9TH INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2008

EDITED BY

Vicente Botti

Antonio Barella

Carlos Carrascosa

NOVEMBER 17-19, 2008

UPV

VALENCIA

SPAIN

A Publication of EUROSIS-ETI

Cover art of Divinity 2: Ego Draconis was reproduced by kind permission of Larian Studios, Oudenaarde, Belgium

9TH International Conference

on

Intelligent Games and Simulation

VALENCIA, SPAIN

NOVEMBER 17 - 19, 2008

Organised by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Binary Illusions

Delft University of Technology

Ghent University

Koala Games

Larian Studios

The Moves Institute

Simulation First

Hosted by

UPV

FIV

Valencia, Spain

GAME ON®
2008

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

General Conference Chairs

Vicente Botti, Universidad Politécnica de Valencia, Valencia, Spain
Antonio Barella, Universidad Politécnica de Valencia, Valencia, Spain
Carlos Carrascosa, Universidad Politécnica de Valencia, Valencia, Spain

Local Programme Committee

Gustavo Aranda, Universidad Politécnica de Valencia, Valencia, Spain
Ramón Mollá, Universidad Politécnica de Valencia, Valencia, Spain

INTERNATIONAL PROGRAMME COMMITTEE

Game Development Methodology

Track Chair: Licinio Roque, University of Coimbra, Coimbra, Portugal
Joaquim Ramos de Carvalho, University of Coimbra, Portugal
Óscar Mealha, University of Aveiro, Portugal
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
Paolo Remagnino, Kingston University, Kingston Upon Thames, United Kingdom
Marcos Rodrigues, Sheffield Hallam University, Sheffield, United Kingdom
Joao Manuel Tavares, FEUP, Porto, Portugal

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

INTERNATIONAL PROGRAMME COMMITTEE

Artificial Intelligence

Artificial Intelligence and Simulation Tools for Game Design

Stephane Assadourian, UBISOFT, Montreal, Canada
Michael Buro, University of Alberta, Edmonton, Canada
Penny de Byl, University of Southern Queensland, Toowoomba, Australia
Antonio J. Fernandez, Universidad de Malaga, Malaga, Spain
Tshilidzi Marwala, University of Witwatersrand, Johannesburg, South-Africa
Gregory Paull, The MOVES Institute, Naval Postgraduate School, Monterey, USA
Oryal Tanir, Bell Canada, Montreal, Canada
Christian Thurau, Universitaet Bielefeld, Bielefeld, Germany
Miguel Tsai, Ling Tung University, Taichung, Taiwan

Learning & Adaptation

Christian Bauckage, Deutsche Telekom, Berlin, Germany
Christos Bouras, University of Patras, Patras, Greece
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
Nicholas Graham, Queen's University, Kingston, Canada

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

Voice Interaction

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

Artistic input to game and character design

Anton Eliens, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
Olli Leino, IT-University of Copenhagen, Copenhagen, Denmark
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

INTERNATIONAL PROGRAMME COMMITTEE

Modelling of Virtual Worlds

Rafael Bidarra, Delft University of Technology, Delft, The Netherlands
Marc van Kreveld, Universiteit Utrecht, Utrecht, The Netherlands

Online Gaming and Security Issues in Online Gaming

Marco Furini, University of Piemonte Orientale, Italy
Pal Halvorsen, University of Oslo, Oslo, Norway
Jouni Smed, University of Turku, Turku, Finland

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
Shea Street, Tantrum Games
Mike Zyda, USC Viterbi School of Engineering, Marina del Rey, USA

Serious Gaming

Wargaming Aerospace Simulations, Board Games etc....
Roberto Beauclair, Institute for Pure and Applied Maths., Rio de Janeiro, Brazil
Henry Lowood, Stanford University Libraries, Stanford, USA
Tony Manninen, University of Oulu, Oulu, Finland
Jaap van den Herik, University of Maastricht, Maastricht, The Netherlands

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

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

Games Interfaces - Playing outside the Box

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

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

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

© 2008 EUROSIS-ETI

Responsibility for the accuracy of all statements in each peer-referenced paper rests solely with the author(s). Statements are not necessarily representative of nor endorsed by the European Simulation Society. Permission is granted to photocopy portions of the publication for personal use and for the use of students providing credit is given to the conference and publication. Permission does not extend to other types of reproduction or to copying for incorporation into commercial advertising nor for any other profit-making purpose. Other publications are encouraged to include 300- to 500-word abstracts or excerpts from any paper contained in this book, provided credits are given to the author and the conference.

All author contact information provided in this Proceedings falls under the European Privacy Law and may not be used in any form, written or electronic, without the written permission of the author and the publisher.

All articles published in these Proceedings have been peer reviewed

EUROSIS-ETI Publications are ISI-Thomson and INSPEC referenced

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

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

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

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

EUROSIS-ETI Publication

ISBN: 978-9077381-45-8
EAN: 978-9077381-45-8

Preface

We would like to welcome you to the 9th edition of the Annual European Conference on Simulation and AI in Games on behalf of all the people and institutions that made this conference possible.

The field of Gaming is an expanding field not only from a business point of view (videogames are the most promising branch in the leisure business sector), but also from a research point of view. In this last case, games not only feature high-definition graphics, but also cover methodologies, AI, communications, and a widespread amount of topics. Some of these topics are reflected in the papers included in this year's conference, including serious games, board games simulations, artificial intelligence, game graphics and game methodology.

On behalf of the Organizing Committee, we would like to thank, first of all to Philippe Geril, the driving force behind the GameOn Conference, without whom this conference would not exist, and last but not least to all the members of the International Program Committee for their excellent work in reviewing and selecting the best papers from all the submissions received. This conference would not succeed without their participation, along with that of the attendees, presenters and session chairs.

As in past editions of Game-On, this year's three-day technical program provides a forum to address, explore and exchange information on the state-of-the-art of all kind of technologies in support of gaming, their design and use, and their impact on our society. Moreover, this year's edition takes place during the first Videogame Week at the Universidad Politécnica de Valencia. During this week there will be a lot of videogame-related activities, including the first Spanish Videogames Developers Conference (CDV), exhibitions, game-programming and playing contest,

Carlos Carrascosa, Antonio Barella and Vicente Botti

Game-On 2008, General Chairs

CONTENTS

Preface ..... **XI**
Scientific Programme **1**
Author Listing **159**

GAME METHODOLOGY

Model Driven Game Development: 2D Platform Game Prototyping
Emanuel Montero Reyno and José Á. Carsí Cubel **5**

HACCS: A Hierarchical Approach to Continuum Crowd Simulation
Christopher Mitchell Deeb and Xin Li **8**

GAME GRAPHICS

Use of LoD Techniques in Luminance Changes on Video Games
Iñigo Barona, Ramón Mollá, Francisco López and Mª Carmen Juan **17**

IntelligentBox as Development System for SaaS Applications Including Web-based 3D Games
Yoshihiro Okada **22**

Transformation and Interpolation with VQM Structure
Xin Li, Cody Luitjens and Michael Beach **27**

ARTIFICIAL INTELLIGENCE

AI BEHAVIOUR

Automatable Evaluation Method Oriented toward Behaviour Believability for Video Games
Fabien Tence and Cedric Buche **39**

Evolving believable behaviours for first person shooter games
Dave Saunders and Colm O'Riordan **44**

GAME AI ANALYSIS

Analysing the Fitness Landscape of an Abstract Real-Time Strategy Game
David Keaveney and Colm O'Riordan **51**

CONTENTS

MASTERGOAL: An Interesting Testbed for AI Techniques
Alberto Samaniego, Alejandro Alliana and Benjamín Barán **56**

An Analysis of Fitness Landscape in the Evolution Social Structures for Computer Games
Alan Cunningham and Colm O'Riordan **61**

AI PROGRAMMING

Rapid Adaptation of Video Game AI
Sander Bakkes, Pieter Spronck and Jaap van den Herik **69**

Compacting a Rule Base into an and/or Diagram for Game AI
Samuel Manier and Olivier Sigaud **77**

NEURAL NETWORKS AND AGENT BASED SIMULATION

Da Vinci Code Game An Neural Network approach to an imperfect information game
Jeroen Boogaard and Leon Rothkrantz **85**

Extending Game Technologies with Virtual Reality and Multi-Agent Systems
A. Barella, C. Carrascosa and V. Botti **94**

Agent Based Serious Gaming; Simulating the Perception Bias on Innovation Diffusion
Marco Remondino, Marco Pironti and Roberto Schiesari **99**

Mshindi – an Awalé winner
Vincent de Boer and Leon Rothkrantz **107**

SERIOUS GAMING

TEAM BUILDING

Emerging Team Collaboration by Social Reinforcement
J. Fabregat-Pinilla and C. Carrascosa **117**

Generating Emergent Team Strategies in Football Simulation Videogames via Genetic Algorithms
Antonio J. Fernández, Carlos Cotta and Rafael Campaña Ceballos **120**

CONTENTS

EDUCATION AND SOCIAL NETWORKS

MyHeart
Juan Albero Sanchis **129**

XIMPEL Interactive Video - between narrative(s) and game play
Anton Eliëns, Hugo Huurdeman, Marek van de Watering
and Winoe Bhikharie **132**

Online Gaming Communities: A Practical Shortlist
Marek R. van de Watering, Winoe Bhikharie, Hugo C. Huurdeman
and Anton Eliëns **137**

MMOG based on MAS: A Game-Independent Ontology
G. Aranda, C.Carrascosa and V.Botti **141**

Edutainment games included as activities in the Summer School of the Technical University of Valencia
M. Carmen Juan, David Furió, Ramón Mollá, M. José Vicent, Roberto Vivó
and Miguelón Giménez **147**

Humming in Facebook: An overview of Social Network platform architectures
Ken Newman **152**

SCIENTIFIC PROGRAMME

GAME METHODOLOGY

MODEL-DRIVEN GAME DEVELOPMENT: 2D PLATFORM GAME PROTOTYPING

Emanuel Montero Reyno and José Á. Carsí Cubel
Grupo de Ingeniería del Software y Sistemas de Información
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camí de Vera s/n, 46022, Valencia, España
E-mail: {emontero, pcarsi}@dsic.upv.es

KEYWORDS

Game Development Methodology, Game Design, Tools, Model-Driven Development.

ABSTRACT

The increasing complexity of game development highlights the need of intellectual and industrial tools to enhance productivity in terms of quality, time and cost. In this paper we propose to apply Model-Driven Development (MDD) methodology to game development, rising the level of abstraction towards game conceptual modelling. As an example, we present a game design prototype tool to prototype 2D platform games for PC, with automatic code generation from UML models.

1. INTRODUCTION

In the last decades, industrial game development has grown exponentially in complexity, highlighting the need of development methodologies to enhance productivity. Game development relies greatly on adhoc development or waterfall methodology, although other software development methodologies have been recently applied to game development: Agile Methodologies, Component-Based Development and Software Product Lines. Withouht leaving aside these previous approaches, we propose to use Model-Driven Development (MDD) methodology in order to rise the level of abstraction of game development, enhancing productivity and shifting efforts from game programming to game conceptual specification, allowing the automatization of the coding tasks in game development.

As an introductory application example of Model-Driven Game Development (MDGD) we present a prototype tool to prototype 2D platform games for PC. Prototyping and playtesting has been previously highlighted (Henderson 2006; Fullerton et. al. 2004) as fundamental engines of game design. Despite being a first approach to productivity enhance of game prototyping, interesting results can be seen working with a higher level of abstraction in game development.

This paper is organized in the following sections: section 2 presents the state of the art in game development methodologies. Section 3 describes the models and transformations that support a prototype tool for MDGD. Section 4 details the conclusions and future lines of research.

2. STATE OF THE ART

Game development is one of the industries with greater expansion of the last decades. The complexity of games has increased exponentially since its origins (Blow 2004). In the mid 80s a game could be developed in 3 months by one programmer who also did the design and art, from conception to final implementation. In 2005 a game can be developed by a team between 20 and 100 multidisciplinary specialists, including programmers, game designers, artists, writers, voice actors, musicians, etc. with a budget over 10 millions of dollars for 4 years of development. Despite the increasing size of the development team and the economic and temporal resources invested, there is a great need of development methodologies to enhance game development productivity.

Game development is a field typically characterized by adhoc low-level development. In the latest years some effort has been done to introduce development methodologies into game development. The first attempt was to use Waterfall Development to cope with the increasing technology demands of game development. Teams scaled up requiring extense game design documentation written in natural language to keep the focus of develpment. But game design documentation wasn't easy to maintain, which caused communication breakdown. Design changes were handled directly at programming level, leading to a difficult maintenance of games and low productivity.

Component-Based Development methodology has been applied to game development (Folmer 2007) in order to allow a greater reuse of recurrent functionality. The use of game development-specific middleware doesn't rise the level of abstraction in development. Games remain programmed with object-oriented languages, script languages and so on. Middlewares are successful and widely extended in game development, becoming an obligation for game development methodologies.

Agile Methodologies have been applied to game development (McGuire 2006; Miller 2008) to enhance change management during development and iterativity in game design, approaching as fast as possible to the core gameplay for the client. Although iterativity enhances game development, Agile Methodologies doesn't rise the level of abstraction in development, keeping the focus on game programming.

In academic research, Software Product Lines have been applied to game development (Furtado and Santos 2004) using graphic domain-specific languages to represent variations in action-adventure games. Individual games are

generated using code templates. The reuse of common characteristics in games of the same genre can be exploited considering this semi-automatic Software Product Line development.

3. MODEL-DRIVEN GAME DEVELOPMENT

In order to cope with the increasing complexity of game development, the level of abstraction have to be increased. With this aim we propose to apply Model-Driven Development (MDD) methodology to game development. As an introductory example a prototype tool has been implemented to prototype 2D platform games for PC from UML models. 2D platform games are a game genre characterized by a protagonist who moves and jumps into platforms, collecting prizes and destroying enemies in various ways. Classic 2D platform games include *Super Mario BROS*, *Sonic The Hedgehog* and *Bubble Bobble*.
The prototype tool uses two Platform-Independent Models (PIM) to define the structure and behaviour of games avoiding development details of the underlying technology platform. A Platform-Specific Model (PSM) describes the mapping of game actions to the hardware control devices by which the player interacts (joystick, keyboard, …). A model transformation from the previous UML models generates the code of the prototypes in C++ using a game development-specific middleware, *Haaf Game Engine*, to provide basic functionality through specialized libraries. Other programming languages and middleware could have been used as development technology platform. Finally, the generated code can be manually completed and the 2D platform game prototype can be playtested iteratively.

3.1. Structure Diagram

2D platform game structure can be specified using class diagrams extended with stereotypes. Stereotypes are UML extensions that allow the creation of model elements suited to the problem domain, which in this case is 2D platform games for PC. The *PlayerCharacter* stereotype describes a game entity controlled by the player. The *Enemy* stereotype defines a game entity controlled by the game system opposing the main character of the player. The *Entity* stereotype describes the other passive game entities such as prizes and platforms. Each stereotype is abstractly represented by a coloured shape in the 2D platform game prototype: the main player character is rendered as a green circle, the enemies as red circles, the prizes as yellow circles and the platforms as orange squares.
In the example *Bubble Bobble* prototype, the UML structure diagram extended with stereotypes specifies all game entities and its relationships. The game level is affected by gravity (attraction towards the ground) and friction (resistence to movement). The main game entities are the player character *Bub*, the bubbles, the *Benzo* enemies, the prizes and the platforms. The player character has a number of lives and a score that increases destroying enemies or collecting prizes. Bubbles can contain a trapped enemy inside.

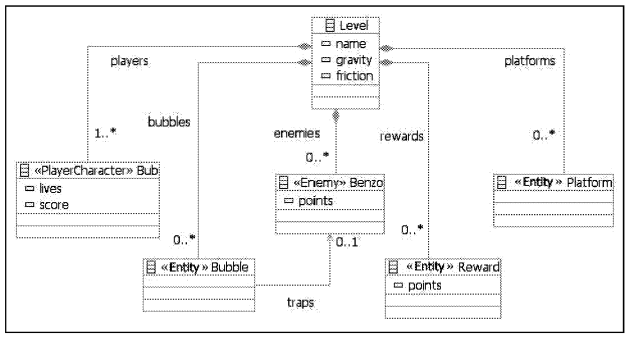


Figure 1: Structure Diagram of *Bubble Bobble*

3.2. Behaviour Diagram

UML state transition diagrams can be used to specify the basic behaviour of each 2D platform game entity. State transitions are triggered by game events which change the internal game entity state. The final game entity state represents the destruction or death of the game entity.
In the *Bubble Bobble* example prototype the behaviour diagram of the player character describes that *Bub* can be alive either on the ground or in the air. In both cases he can blow bubbles but he can only move and jump from the ground. If *Bub* collides with a bubble, he will pop it. If *Bub* collides with a prize he will gain its points. If *Bub* collides with an enemy he will lose a life. Losing all his lives will destroy *Bub*.

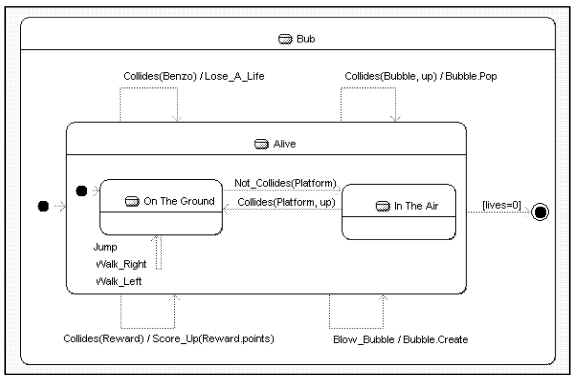


Figure 2: Behaviour Diagram of the Player Character of *Bubble Bobble*

3.3. Control Diagram

Game control can be understood as a hardware device configuration allowing the player to interact with the game. Game control mappings specify which controls are associated to each game action. Game actions are game events triggered by the players. Figure 3 describes a metamodel to capture control diagrams for 2D platform games for PC.

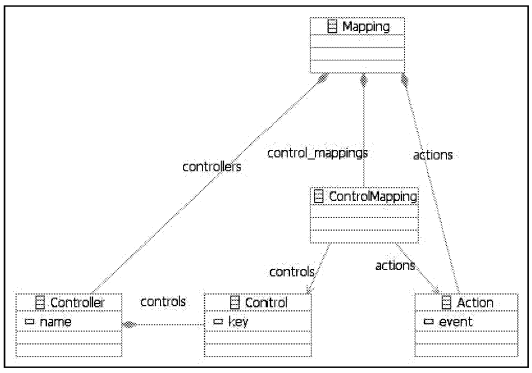


Figure 3: Control Metamodel for 2D Platform Games

The control diagram is a Platform-Specific Model (PSM). Two kinds of techology platforms can be distinguished in game development: development technology and target technology. Development technology platforms include programming languages, SDKs, *middleware* and other technology used in the construction of the game. Target technology platforms include the hardware devices in which the game will be played. In present days there is a great variety of target technology platforms: PC, flatbed and handheld consoles, arcades, mobile phones, etc. Development technology aspects can be automated through the use of Platform-Specific Models (PSM) such as the control diagram.

4. CONCLUSIONS AND FUTURE WORK

Despite the efforts invested in the construction of a prototype tool for 2D platform game prototyping it is fair to declare its shortages and limitations. The UML models used to specify 2D platform games are closer to software engineers than to game developers. It would be of great interest to develop a conceptual model close to game developers terminology, deriving through model transformations the structure and behaviour models.

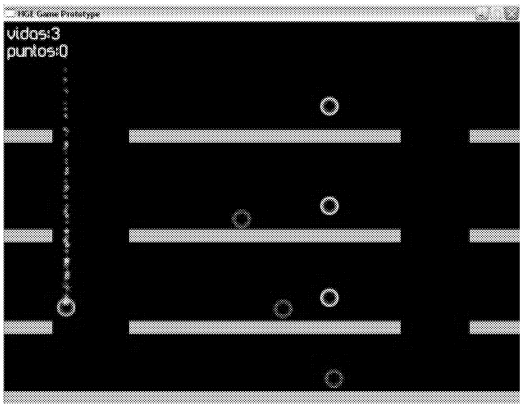


Figure 4: *Bubble Bobble* Prototype

Focusing on the productivity results, a week was invested to manually implement the *Bubble Bobble* example game prototype. Using the model-driven approach, another prototype was automatically built in a day, with 93% of code generation from the UML models. Model to code transformations in *MOFscript* derive automatically most of the game prototype code from the models, notably rising the productivity of the development process. These automated transformations assign each game element semantics, explicitly embedding 2D platform game domain knowledge and good practices in the code generation. In order to informally validate these results, another game (*Super Mario Bros*) was prototyped reusing *Bubble Bobble* models. 94% of the game code was automatically generated. The model-driven approach enhances productivity in game development and reusability of the software artifacts. The programmer only has to manually code aspects not specified in the models (such as the artificial intelligence of the enemies).

Table 1: Automatic Code Generation of 2D Platform Game Prototypes

Proto-type	Structure Code	Behaviour Code	Total %
Bubble Bobble	485 / 495 lines	232 / 274 lines	93%
Super Mario Bros	393 / 393 lines	184 / 215 lines	94%

As future work remains the application of Microsoft's XNA game specific middleware to model-driven game development. The first step is to define a game specific modelling language to precisely describe games using concepts closer to game developers. The next step is to build PSM models for the XNA development techonology platform, and transformations between platform independent and specific models. Finally, code will be generated to each target technology platform: PC and XBOX 360.

REFERENCES

Blow, J. 2004. "Game Development: Harder Than You Think". In *ACM Queue*, 1 (10).

Folmer, E. 2007. "Component Based Game Development: A Solution to Escalating Costs and Expanding Deadlines?". In CBSE.

Fullerton, T. and Swain, C. and Hoffman, S. 2004. "Game Design Workshop: Designing, Prototyping, and Playtesting Games". CMP Books, 157.

Furtado, A. W. B. and Santos, A. L. M. 2006. "Using Domain-Specific Modeling towards Computer Games Development Industrialization". In *DSM Forum*.

Henderson, J. 2006. "The Paper Chase: Saving Money via Paper Prototyping". In *Gamasutra*.

McGuire, R. 2006. "Paper Burns: Game Design with Agile Methodologies". In *Gamasutra*.

Miller, P. 2008. "Top 10 Pitfalls Using Scrum Methodology for Video Game Development". In *Gamasutra*.

HACCS: HIERARCHICAL APPROACH TO CONTINUUM CROWD SIMULATION

Christopher Mitchell Deeb and Xin Li
Computer Science Department
Digipen Institute of Technology
Redmond, Washington, U.S.A

KEY WORDS

HACCS, AI, Crowd Simulation, Path Finding, Continuum Crowds.

ABSTRACT

This paper explores HACCS, a two-dimensional crowd simulation method that uses velocity fields stored inside a quad tree to efficiently move large quantities of crowd members in real-time, interactive framerates. The method is based off of the Continuum Crowds system that attempts to solve the same problem by using a discrete, uniform grid (Treuille et al. 2006). Using a grid of cells contained within a quad tree, the HACCS method creates a final velocity field that moves crowd members in a realistic path toward their goal. The HACCS method is shown to be more stable, accurate, and efficient in many key cases than its Continuum Crowds counterpart. Throughout this paper, the fundamentals of path finding and crowd simulation will be defined and described, leading up to the more accurate and efficient approaches that directly influenced the development of HACCS.

INTRODUCTION

Path planning is a common problem, encountered in some form or fashion in many simulations. While finding an optimal path is a complicated process even on its own, the problem is exasperated by the number of agents and increasing the size of the search area. Planning an optimal path through terrain with dynamic obstacles (such as other moving agents) requires a call to a path-finding algorithm once per actor, per frame. Each actor's path affects every other actor's path in sequence, making the order of path finding important. While one actor trying to find a path to its goal is easily achieved, the problem increases in complexity at a geometric rate as more actors try to find paths through the same area. For all but the smallest crowds, corners must be cut to insure real-time framerates.

Applications of large-scale path planning are numerous. Real-time strategy (RTS) games and military simulations cannot exist if the user has no way of commanding the troops from location to location. In some RTS games, players may be controlling hundreds, even thousands, of troops, and those troops are expected to be able to walk/swim/fly across the game world in a realistic and timely manner. Crowd simulations used for determining evacuation plans from high-occupancy venues such as stadiums and theaters need to be able to realistically show the tendencies of groups of people as they attempt to leave the area. In these kinds of simulations, real-time speeds are not always required. Many crowd simulators favor increased accuracy and complex agent behaviors over a high framerate. Regardless of the use, one trait unifies all applications: as more agents are added to the system, the problem quickly becomes intractable.

RELATED WORK

A good starting point when considering the complex issues of crowd movement is flock and herd behavior (Reynolds 1987). Reynolds' simulation is a combination of a particle system with rule-based actor objects. By adding an orientation, model, and a set of behavioral rules to each particle, Reynolds changes the particles into entities known as "Boids," a collection of actors that are now capable of interacting with each other as well as with their own internal states.

Another important consideration when trying to model crowd movement is pure path finding, such as is described in Dijkstra's "A Note on Two Problems in Connexion with Graphs"(Dijkstra 1959). Dijkstra's algorithm is capable of finding an optimal path between two nodes on a fully connected discrete graph. With the proper use of the heap data structure, the process incurs a time complexity of O(NlogN). Dijkstra's method for finding an optimal path across a graph generates a plethora of extra data in the process. This extra data can be directly used to find additional optimal paths at no added cost. One iteration of Dijkstra's algorithm can be used for any number of actors trying to find a path to a single, shared goal, unlike other path finding methods such as A* that must be executed once per actor.

Many path finding algorithms fall prey to being only as accurate as the connections in their respective graphs allow. What if a simulation requires a higher order of continuity? In order to solve this problem while still maintaining the Dijkstra-like single pass behavior, we enter the realm of level set theory and delve into a method known as Fast March. The Fast Marching algorithm, as first described in J.A. Sethian's "A fast marching level set method for monotonically advancing fronts", uses a finite difference approximation to solve the Eikonal equation that describes a potential field in which the solution does not change signs (Sethian 1996). The accuracy of Fast March is determined by the discretization of the graph. The coarser the graph is, the less accurate the method will be (Sethian 2006). Fast March has many applications, from fluid mechanics to computer graphics. Some of the most recent uses of Fast March can be seen in crowd simulations.

The previous work most influential to HACCS is the "Continuum Crowds" paper (Treuille et al. 2006). In the Continuum Crowds system, based on continuum dynamics, a dynamic potential field integrates global navigation with moving obstacles such as other crowd members, quickly and effectively solving for the movement of large crowds without excessive use of explicit collision avoidance. The notion of agent-based path planning is abandoned in favor of treating agents as mindless particles that an underlying field acts upon. These particle-agents shall henceforth be known as Units. This field is a uniform grid made up of square cells. Once per frame, in each cell, data such as density, flow speed, and velocity is recorded, which is used to find a movement cost from a cell to each of its four neighbors.

Using the Fast Marching algorithm, a dynamic potential field is created and recorded inside the cells of the grid. Final velocity values are assigned to each cell by tracing the gradient of the potential field in the upwind direction, ultimately using the earlier-calculated speed values found in the corresponding direction. Units are moved based on the final velocity value found in the cell currently occupied. To correct for error, the last step of the algorithm is to enforce the minimum distance between Units, preventing overlap (Treuille et al. 2006).

The Continuum Crowds approach yields very impressive results, generating large, interactive crowds (10000+ Units) at real-time framerates. However, it requires uniformity in the size of the cells. Using large cells has the advantage of increased speed and lighter calculations, but compromises accuracy, and can result in "dumb" and rigidly-moving crowds. Indeed, if the cell size grows too large, the minimum distance reinforcement step becomes a quadratic collision-detection problem, which can greatly slow down the system.

Conversely, if the cell size is too small, there will be too many cells, and the simulation will simply run too slowly. Thus, a user must estimate in any given implementation how crowded he or she expects the most congested areas will be, and the cell size must be a good balance between large and small to account for correct accuracy at high traffic areas and good performance overall. Oftentimes much of the grid is unoccupied and unused. Many scenarios result in only a few, select areas of heavy Unit concentration, while the remainder of the grid is sparsely populated. Some may wonder if there is a way to get the best of both worlds: high accuracy at key areas coupled with good performance due to the use of fewer cells.

OUR APPROACH: HACCS

The HACCS crowd simulation method takes an approach similar to Continuum Crowds, but scraps the notion that all cells are uniformly sized. Instead, the simulation's area is encompassed by a quad tree of cells, going down to a user-defined depth. In the quad tree, the root is the largest, and cell size decreases as the tree is traversed toward the leafs. All cells have the same aspect ratio, which is equal to the aspect ratio of the entire rectangular simulation area as a whole. Rather than iterating through all cells as in Continuum Crowds, HACCS creates a small subset of cells from out of the quad tree, designated the "Active List". The Active List's cells comprise an area that completely spans the entire simulation space, with larger cells located at areas of low occupancy and smaller cells at areas of high occupancy. Since only a relatively small number of cells are updated each frame, the most expensive part of the simulation, the creation of the dynamic potential field, is minimized and greatly alleviated. This is the key factor in the performance increase that HACCS offers over Continuum Crowds.

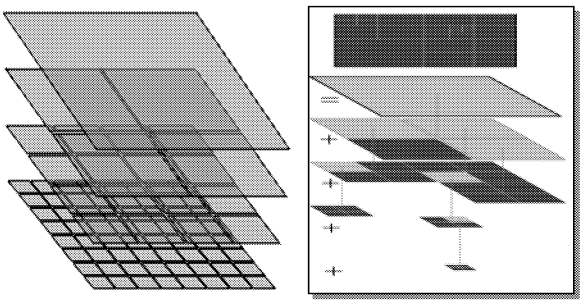


Figure 1(left): A graphical representation of a four-leveled quad tree. This structure is allocated once at the beginning of the simulation.

Figure 2(right): Blue cells represent active cells. As seen by this graphical summation, the combined area of the Active List completely covers the simulation area.

Note that a cell in Continuum Crowds has exactly four neighbors (except edge cases) with a uniform size. The directions and distances from a cell to any of its neighbors are static and deterministic. The directions are always north, south, east, and west, and the distance in all cases is simply the length of the cell. In HACCS, however, a cell can have different numbers of neighbors depending on its size and position. The directions between a cell and its neighbors are also no longer set to the four cardinal directions, nor are they a constant, pre-calculated length.

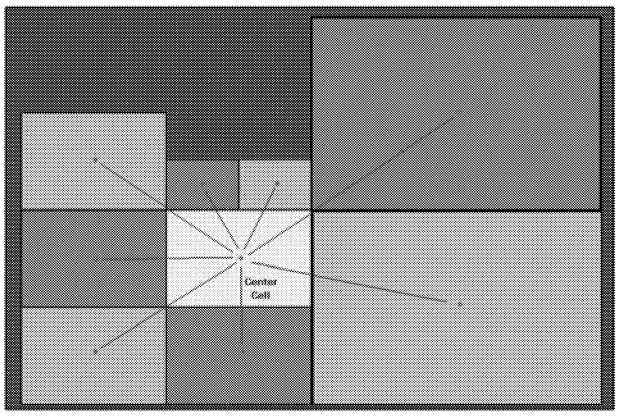


Figure 3: The center cell is connected to eight neighboring cells. The blue lines represent the anisotropic data stored along with the cell pointer leading from the center cell to the adjacent neighbors. Please note that in HACCS, diagonal cells are considered to be adjacent in this context.

These directions and lengths must be calculated and recorded each frame for every new Active List. Fortunately, the additional time it takes to perform these calculations is more than made up for by the savings gained during the dynamic potential field step. Also, even though a vast number of neighbors is possible, most of the time a cell will end up having close to eight neighbors. Due to the increased and

varying number of neighbors, the Fast March technique has been replaced by a version of Dijkstra's algorithm.

In the rest of this section, the actual structure of HACCS is explored, starting with the field's creation, followed by a brief overview of the algorithm and a description of step-by-step breakdown of what happens in a single frame during the crowd simulation. The detailed discussion of each break-down step is then presented in the next section.

Creation of Dynamic Potential Field

To create the dynamic potential field, first set the width and height dimensions of the field. This should correspond to the width and height of the area the crowd will occupy. The aspect ratio will be maintained across all of the field's cells. Next, allocate the space for the cells themselves. The cells must represent a quad tree that spans the simulation area. It will have a user-specified depth. This allocation is done to avoid having to dynamically allocate and deallocate memory during execution. After having created all the cells in the tree, go through each level of the tree, linking neighboring cells with cell pointers. At this point, only cells on the same level of the tree can be considered neighbors of each other; these neighbors are specially designated "same-level" neighbors, and are used only when constructing the list of active neighbors during execution. This completes the creation of our field, and it is done only once at the beginning of execution.

Overview of the Algorithm

A frame begins by resetting the field and constructing the Active List. Next, all the cells in the Active List find their respective Active List neighbors. Once the Active List has found its adjacency information, each cell's density, average velocity, and discomfort is calculated. Then, a linear interpolation is used to find the speed values inside each cell in the Active List. Utilizing the speeds, distances, and discomforts in any given cell in the Active List, the Unit Cost values are calculated.

The next step is the most costly: finding the potential values. In order to find the potential values at each cell in the Active List, a version of Dijkstra's algorithm is executed, using Unit Cost as the path weights. After the potential values have been found at each cell throughout the Active List, they are used to find the final velocity term in each cell. Finally, using the final velocity vectors appropriate to each Unit, the Units are moved.

The step-by-step breakdown of the algorithm is as follows:

(1) Frame Start
(2) Find Active List Neighbors
(3) Density, Average Velocity, and Discomfort
(4) Speed Calculation
(5) Unit Cost Values
(6) Potential (via Dijkstra's)
(7) Final Velocity
(8) Move Units

ALGORITHM DESCRIPTION

Step One: Frame Start

At the beginning of each frame, the Active List must be reset and reconstructed. In order to do this, the field must be wiped clean. Next, the Units are "splattered" onto the field. Finally, in a three-step process, the Active List is constructed from the cells inside the field's tree.

To reset the field, iterate through the tree, resetting each cell's dynamic values. Each cell must be removed from the Active List. Also, each cell's number of occupants, density, and discomfort values must all be set to zero. Then, the "potential" values must be set to infinity (99999.9f will do in C++). Finally, we must clear the Active List.

To splatter the Units onto the field, iterate through the Unit list. Look at each Unit's position, and pair it up with the corresponding cells. A Unit is guaranteed to be on more than one cell; in fact, they are guaranteed to be on exactly H cells, where H is the height of the quad tree. For each Unit, keep track of these H overlapping cells. Later, when the Active List has been constructed, exactly one of the H cells will be Active, and that cell will be the proper "current" cell for the Unit.

The "pre-Active-List" step prepares the tree for the construction of the Active List. During this step, each cell designated as a goal has its occupant count artificially inflated by a large constant (say, 99999). Going up the tree from each goal toward the root, each subsequent parent cell's occupant count is identically inflated. This is done to force the grid around each goal cell to be divided into the smallest possible discretization. In a similar manner, all cells at a specified level have their occupant counts inflated. This level is designated as the "minimum level" for the tree, and it is a user specified constant. The minimum level represents the coarsest level of any grid cell that will be allowed to appear on the Active List.

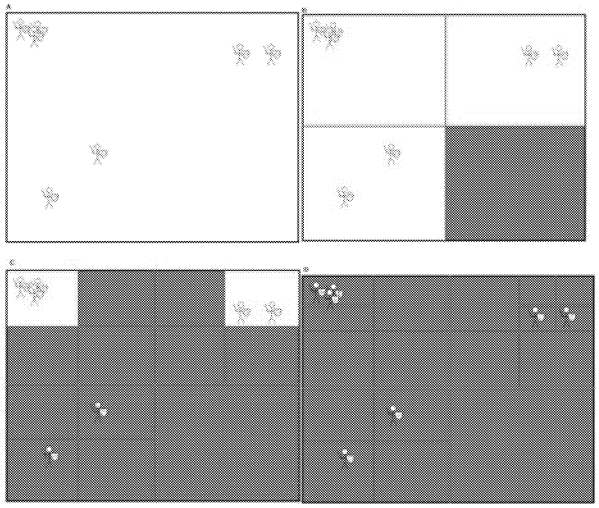


Figure 4: (1) Blue represents cells that do not need to be broken down any further, and white designates cells that have too many Units in them and so must be split into smaller cells. Here, we start in step A, with a top level cell that is filled past capacity. (2) Step B divides the cell into four smaller cells, which finishes the division process for the bottom-right quadrant. The rest of the grid still needs some work. (3) Step C splits the white cells again, which completes all the divisions except the top-right-most and top-left-most cells, which are still beyond capacity. (4) Step D divides the white cells one last time, finishing off the division step entirely. Note that in the top-left cell, there are still three Units. The division ended in this case because in this example we are

working with a tree of depth four, and when the maximum depth is reached, a cell cannot be divided any further. So, those three guys in the top-left will have to make do with being a bit crowded for a frame or two.

The function that constructs the Active List is recursive. A simple description of its functionality is to say that it starts at the root of the tree, steps through each cell, and uses the occupant count to determine which cells will make it onto the Active List. Cells with too many occupants must be "broken down" into finer levels, stopping at the appropriate amount of occupancy.

The "post-Active-List" step is the direct opposite of the "pre-Active-List" step, undoing the artificial inflation of various cells' occupant counts. During this step, each cell designated as a goal has its occupant count reduced by the same large constant as was used in the "pre-active-list" step. Going up the tree from each goal toward the root, each subsequent parent cell's occupant count is identically reduced. This should correct the occupant count of all cells, making it accurately reflect the number of Units present on any given cell. In a similar manner, all cells at the designated "minimum level" have their occupant counts reduced and corrected.

Also, during this step, the system iterates through the Unit list in order to determine each Unit's "current cell". Recall that in the pre-Active-List step, each Unit recorded the list of H cells that the Unit occupied. Now, exactly one of those H cells will be Active. That cell will become the Unit's "current cell".

Step Two: Find Active List Neighbors

In order to do anything useful with the Active List, we must find all the necessary adjacency information. Therefore, in this step, we iterate through the Active List, figuring out which cells are adjacent to which other cells. Consider it a two-step process that is repeated eight times per active cell. Each cell has several pre-calculated neighbors, called "same-level-neighbors", which were defined during field construction. Go through each one of the same-level-neighbors (the four cardinal directions as well as the four secondary directions), look at its parents, then all of its children, searching for any and all Active cells, adding them as an Active List neighbor to the current cell.

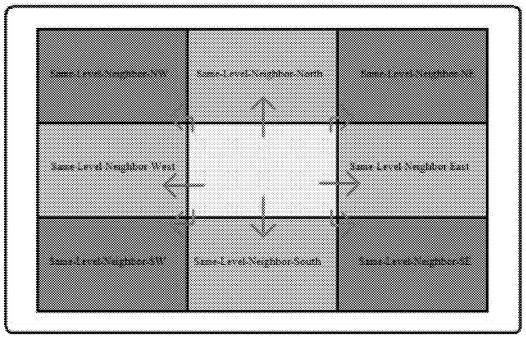


Figure 5: Shortly after the tree's cells are allocated, each level of the tree is iterated through, hooking each cell up with its neighbors on the same level of the tree. During the step where the neighbors for the cells on the Active List are found, these eight same-level-neighbors are used extensively. Each one of the eight is tested; a cell's progeny, ancestry, and even itself, are all tested to find any and all active cells that are adjacent to the center cell.

Step Three: Density, Average Velocity, Discomfort

Once the Active List is constructed and the appropriate adjacency information is found, it can be put to good use. First, the cells in the Active List are cleared of temporary data. To find the density and the average velocity, iterate through the Unit list. Add the Unit's density and velocity terms to the density and average velocity terms of the Unit's specified "current cell". Then, iterate through the Active List, dividing the cells' densities by the appropriate cell area to get the appropriate cell density. Also, divide the average velocity term by the number of occupants to find the appropriate average velocity.

Similar to the way density is calculated, discomfort is also found here. Discomfort from a Unit is that Unit's density projected in the direction of that Unit's velocity; it represents where the Unit will probably be in the near future, or at the very least, where the Unit is trying to go. The discomfort from a Unit is distributed in the direction of its velocity. In that direction, a center cell is chosen. Weighted by its distance away from the center cell and all of the center cell's neighbors, the discomfort will be spread among this set of cells.

Step Four: Find Speed

To find the speed, step through the Active List, calculating each cells' anisotropic data. The key pieces of information obtained for any single anisotropic data set are the distance and the direction of the anisotropy. Once the distance and direction terms are found, a linear interpolation is performed between the maximum Unit speed and the average velocity at the current cell. The parameter for the interpolation is the density at the current cell. At low densities, the maximum unit speed is favored, and at high densities, the average velocity is favored. When a speed is acquired, clamp it between the maximum and minimum possible speeds (the minimum should be a number slightly above zero). Though there is nothing about the HACCS technique that prohibits the calculation of terrain contributions to speed, in this paper's tested implementation terrain was left out, choosing to focus only on the crowd's interaction with itself.

For each cell in the Active List, the speed in the direction of each one of a cell's neighbors is individually calculated by the following equation. It is then stored with the rest of the anisotropic data gathered in this step.

$$Speed_i = S_M(1 - t_i) + t_i(A_i \bullet dir_i) \quad , \text{ where}$$

$$t_i = \frac{D_i}{D_M}$$

D_i = The density of the neighboring cell in the direction of i from the current cell

A_i = Average velocity of the neighboring cell in the direction of i from the current cell

dir_i = The unit-length vector from the center of the current cell pointing to direction i

D_M = Maximum Density

S_M = Top Speed

Step Five: Find Unit Cost

To find the Unit Cost, iterate across the Active List, performing the Unit Cost equation on each cell. The equation makes use of three user-defined weights (alpha, beta, and

gamma) in conjunction with a simple speed-based equation. The method used to find each cell's Unit Cost greatly affects crowd behavior. This is where it is decided whether density, flow speed, or crowd discomfort is weighted more heavily than the other factors. Altering these weights directly changes the behavior of the crowd.

The Unit Cost function is exactly the same as from the Continuum Crowds approach, except that the distances between cells is non-uniform. This fact must be reflected that inside the equation.

$$\int_P C \qquad \text{, where} \quad C \equiv \frac{\alpha f + \beta + \gamma g}{f} d$$

Step Six: Find Potential

The potentials at each cell are found using a version of Dijkstra's algorithm. To start, set the "current cell" to be the goal. Set the current cell's potential to zero, its candidate flag to false, and its done flag to true. Enter the while loop, which continues until the size of the candidate list is zero. Iterate through the current cell's active neighbors list; if a neighbor's done flag equals true, then move on to the next neighbor. Find the anisotropic data that leads from the specified current cell's neighbor to the current cell, which is thusly labeled "correct anisotropy". The projected cost is equal to the current cell's potential plus the correct anisotropy's Unit Cost. If the current cell's neighbor's potential is not greater than the projected cost, then move on to the next active neighbor. If the current cell's neighbor is a candidate, then remove it from the candidate list. Set the current cell's neighbor's potential to the projected cost, and then place it on the candidate list. Note that all insertions into the candidate list should be ordered from smallest to largest potential. Once all neighbors have been considered, set the current cell equal to the cell at the beginning of the candidate list. Then set the current cell's done flag and remove it from the candidate list. Continue with the while loop.

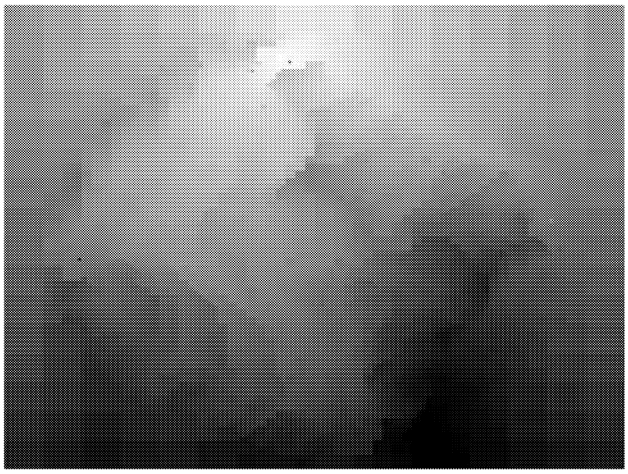


Figure 6: On a simulation area of dimensions 1024x768 pixels, with a tree depth of 10, at a Unit count of 8000, this is the dynamic potential field for the red goal. The lighter areas represent the places that are easier to reach from the goal, and the darker areas represent locations that are more obstructed or farther away. The Units have been made invisible to better examine the dynamic potential field itself in this picture.

Step Seven: Find Final Velocity

After having found the potential values at each cell, next the final velocities are calculated. Iterate through the Active List, finding the correct direction to travel for each cell. This is obtained by searching through a cell's list of anisotropic data and finding the piece with the correct potential. The correct potential is the one that, when subtracted by its neighbor's potential, is equal to zero. The method differs from the method used in Continuum Crowds, where the smallest potential is always chosen. The final velocity is equal to that anisotropy's direction times that anisotropy's speed.

Step Eight: Move the Units

At this point, all cells' final velocities have been acquired. Now, all that must be done is to move each Unit according to its currently occupied cell's velocity field. However, simply doing so without any other considerations will result in choppy, rigid movement. Therefore, for each Unit, take the weighted average of several nearby velocities. The velocities to be considered consist of the velocity at the Unit's currently occupied cell along with all of that cell's neighbor's velocities. The weight applied to each velocity is the Euclidian distance from the center of the Unit to the center of each respective cell. When the averaged velocity has been determined, apply it to the Unit.

If desired, a final step, minimum distance reinforcement, can be executed, which is a simple collision check between all adjacent Units. This minimum distance reinforcement insures that no Units overlap each other, though Units automatically avoid areas of high congestion and thus are unlikely to be overlapping other Units.

Use this final Velocity value to move the current Unit. The term C represents the current cell, N is equal to the number of neighbors at C, and U is the current Unit. The term $C_{n\to i}$ translates to mean "the cell that neighbors C in the direction i from C." The terms V_x and P_x refer to the velocity of object x and the position of object x, respectively. The i term of sigma begins at zero rather than one for the convenience of most computer scientists.

$$finalVelocity = \frac{\dfrac{V_C}{\|P_C - P_U\|} + \sum_{i=0}^{N-1} \dfrac{V_{C_{n\to i}}}{\|P_{C_{n\to i}} - P_U\|}}{\dfrac{1}{\|P_C - P_U\|} + \sum_{i=0}^{N-1} \dfrac{1}{\|P_{C_{n\to i}} - P_U\|}}$$

RESULTS AND COMPARISONS

Having implemented HACCS as well as Continuum Crowds, eight test cases were evaluated. The unifying factor in all test cases is that the simulation area always spanned 1024 by 768 pixels, and the maximum number of Units was always 4500. Please note that the HACCS implementation was set to use a minimum tree depth of four for all tests. The HACCS implementation was pitted against the Continuum Crowds implementation on a laptop containing the following hardware:

Toshiba Satellite X205, Windows Vista Home Premium
Intel® Core™2 Duo CPU T7100 @ 1.80GHz / 1.80GHz
2046 MB RAM
32-bit Operating System

Graphics Card: NVIDIA GeForce 8700M GT

In each test case, the behavior of both implementations was almost identical; the only difference was the performance.

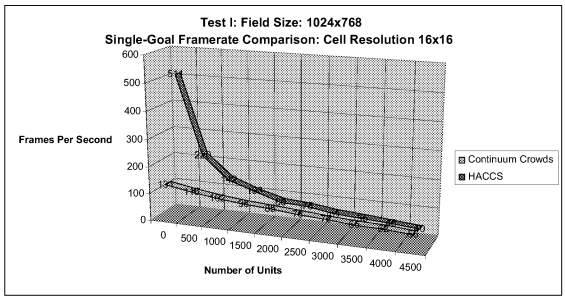


Table 1: To match the resolution of the Continuum Crowds cells (16 by 16 pixels), HACCS generates a tree with a depth of 7, making the smallest cells 16x12 pixels.

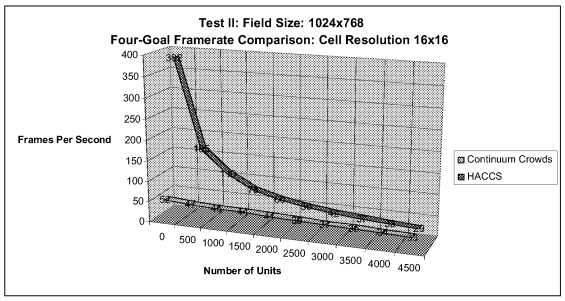


Table 2: As will be seen shortly, Test II marks the last of the test cases where Continuum Crowds even compares with HACCS's results.

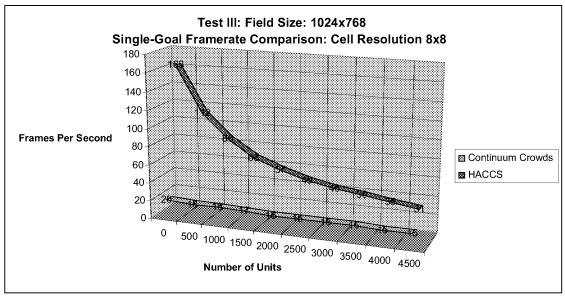


Table 3: To match the resolution of the Continuum Crowds cells (8 by 8 pixels), HACCS generates a tree with a depth of 8, making the smallest cells 8x6 pixels.

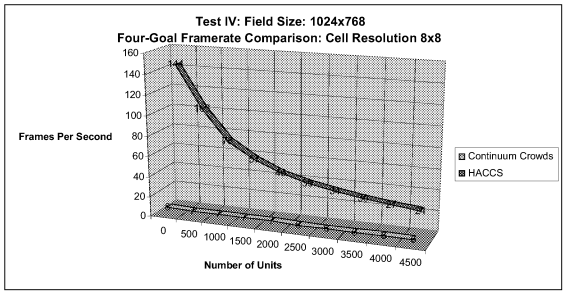


Table 4: At 4500 Units, HACCS performs four times faster than Continuum Crowds. Also, notice that the number of goals present in the simulation has a greater effect on Continuum Crowds than it does on HACCS. In Test III, Continuum Crowds ran at 15 frames per second with 4500 Units, but in Test IV, with the addition of three extra goals, the framerate drops by nearly 66%. Compare this to HACCS, which loses less than 25% of its speed from Test III to Test IV.

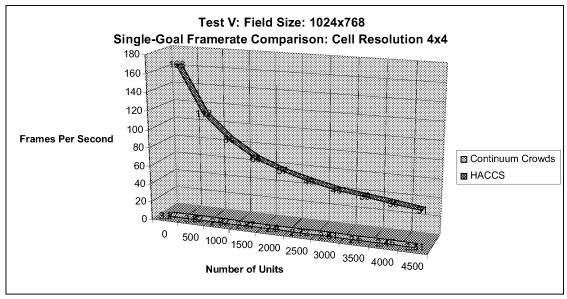


Table 5: To match the resolution of the Continuum Crowds cells (4 by 4 pixels), HACCS generates a tree with a depth of 9, making the smallest cells 4x3 pixels.

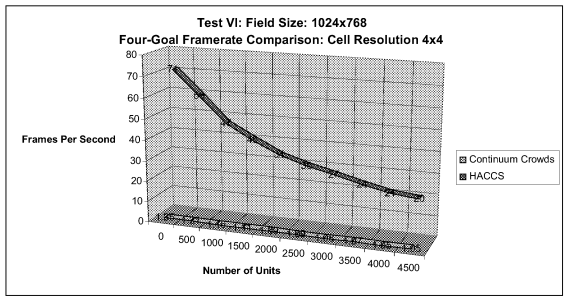


Table 6: As the resolution of the grid increases, the performance gap between HACCS and Continuum Crowds widens. Here, in Tests V and VI, HACCS maintains healthy framerates, while the Continuum Crowds implementation begins to lose its "real-time" status.

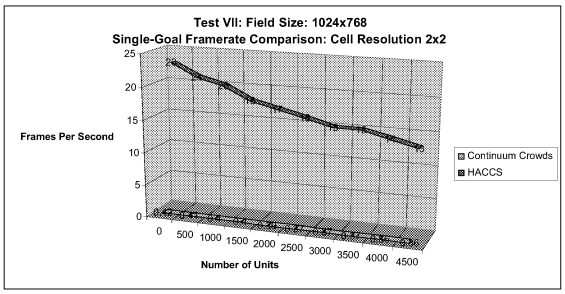


Table 6: To match the resolution of the Continuum Crowds cells (2 by 2 pixels), HACCS generates a tree with a depth of 10, making the smallest cells 2 by 1.75 pixels.

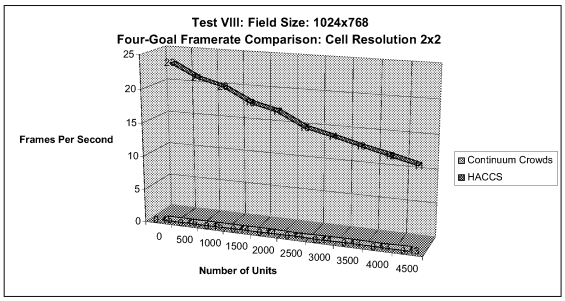


Table 8: Tests VII and VIII push the resolution of the grid to the point where Continuum Crowds takes about seven or eight seconds to complete a single frame of simulation, regardless of Unit count. The HACCS method continues to

flourish, however, maintaining interactive framerates even with 4500 Units and four goals.

From Test III and beyond, the HACCS method consistently displays a higher framerate than the Continuum Crowds method while maintaining equal, if not superior, behavior. After the results of Test VIII, one concludes that once the resolution of the cells becomes sufficiently fine, the Continuum Crowds method simply has too many calculations to make; it cannot stay at interactive framerates. By the end of Test VIII, the HACCS method is over eighty times faster than Continuum Crowds.

CONCLUSION AND FUTURE WORK

The HACCS method has been shown to be superior in performance to the Continuum Crowds method in situations that demand a very high grid resolution across a very large area. However, HACCS might not be the right method to use if memory usage is a big concern. At a tree depth of eight, the HACCS method uses 46 megabytes of RAM, compared to the equivalent Continuum Crowds method at a resolution of 8 by 8 pixels per cell, which uses only 29 megabytes. Though HACCS offers better performance in this case, it uses over 58% more memory than the Continuum Crowds method. However, it is important to note that HACCS does not always use more memory than Continuum Crowds. At depth seven, HACCS has an almost identical memory profile to the equivalent Continuum Crowds implementation at a cell resolution of 16 by 16 pixels.

If memory is a problem, a good goal for future works would be to dynamically allocate only the required cells on the Active List each frame without creating a static quad tree. This may be accomplished with the aid of a memory manager to help avoid the overhead of multiple news and deletes each frame. In fact, it is quite possible that such a method could grant large improvements over the current method of tree creation simply by allowing more useful information to remain in low-level caches.

Also, the HACCS functions that iterate across all Units or all cells in a list lend themselves easily to parallelization. On modern hardware, such parallel processing might yield improvements that could double, even quadruple performance without significantly raising the complexity of the system.

The problem of having too many high-resolution cells across the required grid space is exasperated when the simulation takes on a third dimension. If the purpose is to simulate a crowd across a three-dimensional space, the HACCS method with a little bit of tweaking would, in theory, be a much better choice than other methods, simply due to its ability to severely limit the number of active cells.

REFERENCES

Adrien Treuille, Seth Cooper, and Zoran Popovic. *"Continuum Crowds"*. ACM Transactions on Graphics, 25(3):1160–1168, 2006.

Charles M. Macal, Michael J. North. "*Tutorial on Agent-Based Modeling and Simulation*". Proceedings of the 2005 Winter Simulation Conference.

Craig Reynolds, Sony Computer Entertainment, US R&D. *"Big Fast Crowds on PS3"*. 2006.

E. W. Dijkstra: *"A note on two problems in connexion with graphs"*. In *Numerische Mathematik*, 1 (1959), S. 269-271.

Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). *"A Formal Basis for the Heuristic Determination of Minimum Cost Paths"*. IEEE *Transactions on Systems Science and Cybernetics SSC4* (2): pp. 100–107.

Livingstone, Daniel; McDowell, Robert. 2008. *"Fast Marching and Fast Driving: Combining Off-Line Search and Reactive A. I."*

Reynolds, C.W. *"Flocks, herds, and schools: A distributed behavioral model"*. In Computer Graphics (Proceedings of SIGGRAPH 87), vol. 21, 25-34.

Sethian, J. A., Proc. Nat. Acad. Sci., *"A Fast Marching Level Set Method for Monotonically Advancing Fronts"*, 93, 4, pp. 1591-1595, 1996. Dept. of Mathematics, Univ. of California, Berkeley, California.

Sethian, J. A., Dept. of Mathematics, Univ. of California, Berkeley, California, November 25, 2006. *"Level Set methods and Fast Marching Methods"*.

Soraia Raupp Musse, Branislav Ulicny, Amaury Aubel. "*Groups and Crowd Simulation*". Ch. 14, Handbook of Virtual Humans, John Wiley and Sons, 2004.

GAME GRAPHICS

USE OF LoD TECHNIQUES IN LUMINANCE CHANGES ON VIDEO GAMES

Iñigo Barona, Ramón Mollá, Francisco López and Mª Carmen Juan
Department of Information Systems and Computation
Technical University of Valencia
Camino de Vera, s/n - 46022, Valencia,
Spain
iibale@fiv.upv.es

KEYWORDS
3D Scalability, Rendering

Abstract

The human eye is able to adapt itself to a wide range of luminances from very bright daylight to very dark night light. The human visual system modifies its sensitivity when a variation of illumination occurs. This commonly happen in video games when driving through a tunnel, entering in a dark house or walking in a cave under torchlight. During this transition, known as eye adaptation, our system experiments a temporary blindness. We present a system for accelerating interactive rendering, based on using polygonal simplifications in luminance variations on the scene. We discuss the influence of the preadaptation light, the duration of the transition and the responses to different intensity changes.

INTRODUCTION

Nowadays, 3D scenes are becoming more and more complex so that current graphic hardware is scarcely powerful enough to render some games in real time. In spite of the huge progress on graphic hardware, interactive rendering is still one of the biggest challenges in video games, specially for very low graphic power portable devices such as portable video consoles, smart phones and so on.

A reduction in the graphical payload on the CPU is mandatory in order to improve graphic performance (next gen video games) [1] [2], to reduce power consumption (mobile games) [3] or to increase other video games aspects like behavior (AI), realistic appearance (physics, real time fluids simulation) [4], smoother simulation (increasing sampling frequency) or improving HCI through more complex interactions [5].

The quality of an image depends unavoidably on the visual perception of the viewer. The eyes collaborate with the brain to perceive an object, so visual fidelity is more connected to a perception criteria rather than to a geometrical criteria. In this paper we propose to associate LoD techniques to the eye drop of perception when a sharp change from very bright scenes to very dark ones occurs during the performance of a video game.

Level of detail

LoD techniques offer a way of improving the performance of a video game under certain circumstances by trading visual detail for speed [6].

There are two principal ways of implementing LoD distance: either by calculating the distance to the object from the viewpoint, or by determining the area of the object's bounding volume on the screen. In both cases, lower detail models are employed as the object becomes more distant.

Our aim in this paper is to use any discrete LoD technique to reduce object's complexities, not to present just another LoD algorithm.

Visual perception

The human visual system (HVS) can manage a huge range of luminosities using a process known as *eye adaptation* (from $10^8 cd/m^2$ to $10^{-6} cd/m^2$[7]). The adaptation is not instantaneous and the process can take from 20 to 30 minutes. It is different for rods and cones [8].

As the eye adapts to the darkness (pure scotopic vision starts from $0.034 cd/m^2$ down), its sensitivity improves. On the other hand, as light increases, eye sensitivity is lost allowing to see high levels of luminosity (pure photopic vision starts from $3.4 cd/m^2$ up). Nevertheless, the eye dynamic range is higher in photopic conditions [9].

Aim of this paper

In order to modulate the amount of geometry a LoD technique may discard, there have been some approaches in the past such as relating LoD to an object's motion [10], using the gaze-contingent rendering of the HVS for peripheral location [11] or to automatically balance the system's computational load in an attempt to maintain

a fixed frame rate [12].
Although the eye adaptation process is mastered since long time ago, it has not been used in computer graphics in order to simplify scenes according to light changes during the execution of a real time graphical environment.
Our aim, in this paper, is to evaluate the feasibility of modulating LoD techniques associated to lighting changes. We did a perception test using LoD techniques and changing the lighting of the scene between some given values. We wanted to determine the amount of time a video game can reduce its geometry quality without being noticed by the player. This time can be used to improve other parts of the game: AI, physics, HCI,... or to allow the video game to run at higher frame rates or in lower power consoles.

EXPERIMENTS DESCRIPTION

This section describes the tests made to collect reaction time results. Each person was invited to pass two tests. Each probe was individual and sequential. The first part consisted on a reaction test with the aim of measuring the time elapsed between a visual stimulus and hitting any key as fast as the player could. The second part measured the time of adaptation when the viewer change from a bright environment to a darker environment. Finally, the true adaptation time was calculated subtracting the time of reaction from the time of adaptation.

Reaction test

The main purpose of this test is to know how long it takes to the viewer to react after a stimulus. The test produces a stimulus of maximum contrast, it flips between a white and a black screen. The viewer has to react as soon as possible pressing any key. During the test, there are 5 luminosity changes separated by different time intervals in order to avoid synchronizations. It measures all reaction times and it calculates the average time of reaction. Times between stimulus were randomly generated.

Perception test

This test tried to measure the HVS time of adaptation for different luminosity changes. The idea was to present an image with a certain luminosity during a time of adaptation (approximately 10 seconds). This image showed a non-simplified object (NS object). Afterwards, the viewer pressed a key and it a decrement of luminosity occurred on the image. At this moment, the test flipped an image of the NS object and one of a simplified model (S object) using the same darkness conditions. The flipping was not done at the screen refresh rate because of the persistence effect of the eye. The images were interlaced at a rate of 10fps, 5 for the NS image (odd frames) and 5 for the S image (even images). The viewer had to press a key as soon as possible when he realized any slight vibration or any movement on the object. When the key was pressed, it meant that the viewer's HVS was adapted to the new luminosity conditions and he noticed the biggest polygonal reduction. After the first detection, the test increased the resolution of the S object slightly to the next predefined step using a discrete LoD technique. This new resolution image was interlaced again with the NS object using the same darkness conditions. The viewer had to go on pressing a key when he detected any change. The resolution of the NS Object increased again during 3 loops more until the highest resolution was achieved.

Models choice
We chose two different models to perform the test. The first one was the Stanford Bunny. Simplification and compression algorithms produce very nice results on it. Since this test is focused on interactive rendering acceleration, the number of triangles in the original model is far away too much (69,451 triangles) for real time applications. The test uses a basic bunny simplified up to 5,000 triangles with *Quadric Edges Collapse Decimation* using the tool *Meshlab* [13].
On the other hand, the second model is more complex: the Stanford Dragon. It is extremely rough because of its scales. The dragon was also simplified to a basic dragon with 10,000 polygons without any loss of quality for the distance and size of the image produced for the tests.

Luminosity changes
Luminosity change is the difference of luminance between the initial image and the dark one, in other words, it is the difference of the Y component in the YIQ [14] model between both images. The Y component has value between 0 (black) and 1 (white). In this test we consider that $Y = 2/3$ is the luminosity that provides the brightest image without burnt areas. To calculate the other Y values in the rest of the images used, the maximum luminosity was divided by 2 in every step. This is the same as to move one stop down of the f number of a conventional camera. The minimum luminosity considered was 0.021 since lower values were scarcely visible on the screen and produced no valuable results.
Luminosity changes depend on the preadaption image and the next one. The luminosity change is the difference between both of them. The maximum luminosity change occurs when it flips from the maximum value to the minimum value. Table 1 describes the Y values of the adaptation image (A image), the Y value of the image after the luminosity change (LC image) and the difference of luminosity. Y value translated into candelas are also shown taking into account the maximum

bright of the screen (300cd/m2).

Change Nº	A Image	LC Image	Difference
1	0.666	0.021	-0.645/26.2 cd
2	0.666	0.042	-0.624/25.3 cd
3	0.666	0.083	-0.583/23.7 cd
4	0.333	0.021	-0.312/12.7 cd
5	0.333	0.042	-0.291/11.8 cd
6	0.333	0.083	-0.250/10.1 cd
7	0.166	0.021	-0.145/5.9 cd
8	0.166	0.042	-0.124/5.0 cd
9	0.166	0.083	-0.083/3.4 cd
10	0.083	0.021	-0.062/2.5 cd

Table 1: Description of the luminosity changes.

Models resolutions
The purpose of this paper is to discover if it is possible to simplify a mesh during a luminosity change without affecting the visual quality the observer notices. In this case, the different mesh resolutions for the two models have been chosen following two different criteria: the Hausdorff distance and the RMS error. First of all, the *LoD* has been defined for the base model (Bunny model). The resolutions were: 5000, 4000, 3000, 2000 and 1000 polygons.

Hausdorff distance We measured the mean Hausdorff distance in one direction (base model to simplified model) for each resolution of the bunny. This distance has been calculated with the tool *Metro* [15] which allows to compare pairs of meshes. Next, we found the corresponding *LoD* of the dragon with the same Hausdorff distance to the base dragon model with 10000 polygons. In table 2, different distances and resolutions of the dragon are shown. From now on, Dragon Hausdorff refers to the Stanford dragon model with the *LoD* indicated on table 2. We needed a metric to unify and compare the test made using different models (dragon and bunny) and resolutions. Although these models have different resolutions, they have the same Hausdorff distance in the same step of the test. For example, the bunny *LoD* with 4000 polygons has a Hausdorff distance equal to 0.047 respect to the base model with 5000 polygons.

Error Root-Mean-Square (RMS) In this case, we followed the same method as before but we measure the RMS error of the rendered images of each *LoD* of the bunny. The RMS error has been calculated with a function in *Matlab*. Next, we found the number of polygons necessary to have images of the dragon that produce the same RMS error than those of the bunny. The RMS error and the different resolutions are shown in table 2, . Dragon RMS refers to the different *LoDs* with the resolution of the table .

Bunny	H. Dist	Drag. H.	RMS ϵ	Drag. RMS
5000	0	10000	0	10000
4000	0.047	7100	0.338	9500
3000	0.092	5200	0.339	7000
2000	0.153	3700	0.343	4300
1000	0.291	1970	0.350	3400

Table 2: Hausdorff distance (10^{-3}) and RMS error to the base model and its corresponding resolutions.

Tests organization

First we passed the reaction test and after the perception test. The second one was divided into three parts: one for the bunny and two for the dragon resolution (Hausdorff and RMS). Each part had 10 luminosity changes.
In order to have reliable tests, they were all done in completely darkness so the person was not affected by other environmental factors. The tests were done with the same monitor (21.6" TFT $300cd/m^2$) for everybody and they sat at the same distace (70 cm). Therefore the monitor gave a global luminosity of 40.585cd using its maximum intensity and contrast. The luminosity (cd) associated to every change of the Y component of the whole image was calculated taking into account these numbers. Those values can be seen in Table 1.
Both tests were passed to 25 volunteers and each one took around 15 minutes to complete the test. 15 persons were males and 10 were females. We subtracted the results of the reaction test to the results of the perception test. Those final times are the ones that have been used to study the results.

ANALYSIS OF THE RESULTS

Our hypothesis was that the time of adaptation depended on the difference of lighting between the initial image and the final one. We assumed that no difference would be noticed when changing both initial and final lighting. In order to avoid very long tests that could be boring to the people tested and could discard the validity of the test, we decided to start from different starting intensities dividing the higher one by power of two. Therefore, the starting Y luminance was 0.666, 0.333, 0.166 and 0.083. Since there were ten steps to change this maximum starting Y, we decided to jump to the three lower intensities: 0.083, 0.042 and 0.021. The combination of luminosity jumps can be seen in Table 1.

Models

In this part, three figures are drawn which correspond to the three models. Figure 1 shows the mean time of detection for each luminosity change and for each resolution (1000, 2000, 3000, 4000) for the bunny. Figure 2 represents the same for the dragon Hausdorff model and figure 3 for the dragon RMS model.

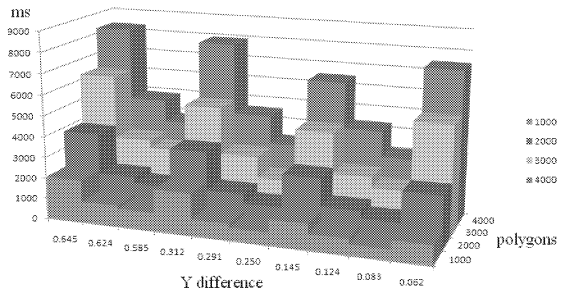


Figure 1: Detection time for bunny model.

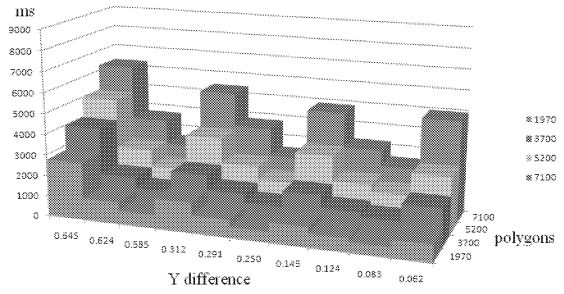


Figure 2: Detection time for Hausdorff dragon model.

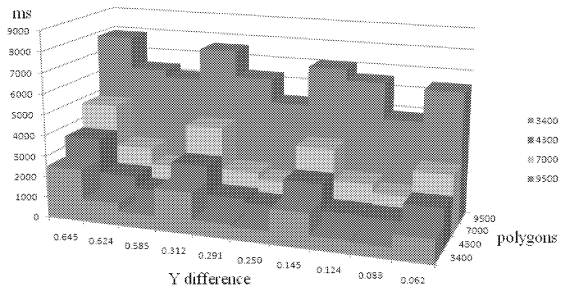


Figure 3: Detection time for RMS dragon model.

Taking into account that the geometry generated using the Hausdorff metric is lower than the one generated using RMS, it is recommended to use Hausdorff metric to simplify a model when using these geometries in real time graphical applications.

Comparing different luminosity changes

All the changes were made from a bright image in the pure photopic vision (26.2, 12.7, 5.9 and 3.4 cd) to one in the mesotopic vision window (0.85, 1.7 and 3,4 cd). All three graphs follow the same tendency:

1. If resolution of a mesh increases, the time of detection increases. It does not depend on the luminosity jump and LoD criteria (Hausdorff/RMS) or model (bunny/dragon)

2. Starting from the same luminosity, as the difference of luminosity increases, the time of adaptation increases. That is the reason why cases 1,2 and 3 show a descendant tendency. It is the same for cases 4, 5 and 6 and for cases 7, 8 and 9. This phenomenon is specially visible when the luminosity of final images is closer to the scotopic vision: cases 1, 4, 7 and 10.

3. Arriving to the same luminosity, as the difference of luminosity increases, the time of adaptation increases. That is the reason why cases 2, 5 and 8 show a descendant tendency. It is the same for cases 3, 6 and 9. This is specially visible when the luminosity of final images is closer to the scotopic vision: cases 1, 4, 7 and 10.

Non-detected cases

In some cases, the volunteers did not press any key. This means that they did not find any difference between the base model and the simplified model under some given luminosity conditions. Those cases are called non-detected.
Figure 4 shows the average number of non-detected for every luminosity change and resolution for all the models. In the first change, an average of 9 people did not find any difference between the base model and the model with the next resolution. Figure 5(A) shows the mean number of non-detected for the same start luminosity and graph 5(B) for the same final luminosity.
The amount of non-detected changes is:

1. Lower as the luminosity jump decreases and the luminosity of the final images move apart from the scotopic vision. Cases 2, 5 and 8 present lower amount of non-detected changes than cases 1, 4, 7 and 10. Cases 3, 6 and 9 are lower than cases 2, 5 and 8.

2. Higher as the resolution of the mesh is closer to the resolution of the original object. This is specially observable in figure 5(A) where the number of non-detected grows as well as the resolution grows. This is specially important when the final image luminosity is closer to scotopic vision. This means the darkest image (cases 1, 4, 7 and 10).

3. Higher depending on the resolution and also on the final luminosity as can be seen in figure 5(B).

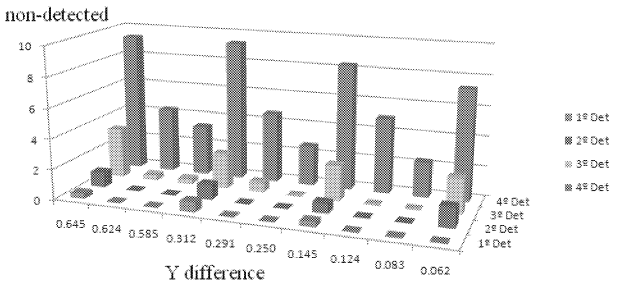


Figure 4: Mean of non-detected for all models.

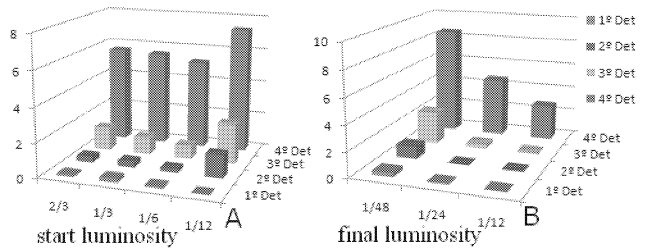


Figure 5: Number of non-detected for the same start luminosity (A) and the same final luminosity (B).

CONCLUSIONS

After the analysis of the results we can extract some conclusions:

1. The adaptation time of adaptation is principally determined by the level of luminosity reached: the conditions at which the image is observed after the luminosity change.
2. The luminosity of the image of preadaption also influences this duration.
3. The results indicate that a few seconds of adaptation exist when the observer can not distinguish some details that will be found afterwards.
4. We can conclude that the type of metric chosen to reduce the amount of polygons is significant in order to increase the time of adaptation.

The adaption time depends on the given mesh of a model and topology, how much the mesh is simplified and many other factors: textures, shadows, background, scene lighting,etc. We emphasize that the timing obtained is really a minimum time since all the player's attention is focused on the object. In real video games, the attention is focused on the main actors. All the objects around these actors could have even higher time of adaptation: backgrounds, furniture,...

References

[1] M. W., "Future graphics architectures," *ACM Queue*, vol. 6, pp. 54–64, March/April 2008.

[2] O. T.J., "Hardware 3d graphics acceleration for mobile devices," *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP*, pp. 5344–5347, 2008.

[3] O. J. B. A. Broll W. Lindt I., Herbst I. and W. R., "Toward next-gen mobile ar games broll," *CG and A*, vol. 28, pp. 40–48, July/August 2008.

[4] G. D. El Hajjar J.F., Jolivet V. and P. X., "A model for real-time on-surface flows," *The Visual Computer, Springer Berlin / Heidelberg*, 2008.

[5] L. Jr. and J. J., "Bringing vr and spatial 3d interaction to the masses through video games," *IEEE CG and A*, vol. 28, pp. 10–15, September/October 2008.

[6] P. M. Astheimer P., "Levelofdetail generation and its application in virtual reality," *Proceedings of the VRST'94 conference*, 1994.

[7] W. C., *Information visualization Perception for design.* Elsevier Inc, 2004.

[8] H. C. Hecht S. and C. A. M., "The influence of light-adaptation on subsequent dark-adapatiotion of the eye," *Journal of general Physiology*, vol. 20, pp. 831–8501, 1937.

[9] B. H. R., "Contrast thresholds of the human eye," *Journal of the Optical Society of America*, vol. 28, 1946.

[10] D. G. T., "Dynamically adaptive animation of virtual humans," Ph.D. dissertation, Faculté des Sciences de l'Université de Genève.

[11] M. H and D. A.T., "Hybrid image-/model-based gaze-contingent rendering," *Applied Perception in Graphics and Visualization*, vol. 253, 2007.

[12] F. T.A. and S. C.H., "Adaptive display algorithm for inter active frame rates during visualization of complex virtual environments," *Computer Graphics Proceedings, Annual Conference Series*, 1993.

[13] C. P., *MeshLab Visual Computing Lab.* ISTI - CNR, 2007.

[14] F. S. H. J. Foley J.D., Dam A. van, *Computer Graphics, principles and practice*, 2nd ed. Addison-Wesley, 1997.

[15] R. C. Cignoni P. and S. R., "Metro: measuring error on simplified surfaces," *Computer Graphics Forum, Blackwell Publishers*, vol. 17(2), pp. 167–174, 1998.

IntelligentBox as Development System for SaaS Applications Including Web-based 3D Games

Yoshihiro Okada

Department of Informatics, Graduate School of ISEE, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, 819-0395 Japan
E-mail: okada@i.kyushu-u.ac.jp

Abstract

This paper treats a 3D graphics software development system called *IntelligentBox* and its aspects as a rapid construction system for interactive 3D games. Especially, the author introduces its new aspects as a development system of SaaS (Software as a Service) applications including web-based 3D games. *IntelligentBox* provides various 3D software components called *boxes* each of which has a unique functionality besides a 3D shape. *IntelligentBox* also provides a dynamic data linkage mechanism that allows users to develop interactive 3D graphics applications only by combining already existing *boxes* through direct manipulations on a computer screen without any text-based programs. This is main feature of *IntelligentBox*.

Application fields of *IntelligentBox* include 3D animation creation, virtual reality software development, 3D interactive simulator development and so on. Besides them, using *IntelligentBox*, it is possible to develop interactive 3D games. However, developed 3D games can not be available on *WWW*. If developed 3D games can be available on *WWW* like SaaS applications, the usefulness of *IntelligentBox* would become higher. So, the author extended the *IntelligentBox* system to make it work as a SaaS application, and then it has become possible to develop web-based 3D games as SaaS applications using *IntelligentBox*.

Keywords: Toolkit system, Component ware, 3D interactive games, Computer graphics, SaaS.

1. Introduction

Advances in recent computer hardware technology have made possible 3D rendering images in real-time. Consequently, 3D software has become in great demand although its development is more laborious work than 2D software development. For this reason, Okada and Tanaka [5-7] developed a 3D graphics software development system called *IntelligentBox*. The *IntelligentBox* system provides various 3D reactive objects called *boxes*. *Boxes* are manually operable objects, which have a 3D visible shape and a unique functionality. The *IntelligentBox* system also provides a dynamic data linkage mechanism called 'slot connection' that allows users to construct interactive 3D graphics applications by combining already existing *boxes* through direct manipulations on a computer screen.

Application fields of *IntelligentBox* include CG animation creation, virtual reality software development, interactive simulator development and so on. Besides them, 3D interactive game development is also included in application fields of *IntelligentBox*. We have already developed several 3D games using *IntelligentBox*. However, developed 3D games can not be available on *WWW*. If developed 3D games can be available on *WWW* like SaaS applications, the usefulness of *IntelligentBox* would become higher. So, we extended our *IntelligentBox* system to make it be used as a development system for SaaS applications including web-based 3D games. In this paper, we clarify the availability of *IntelligentBox* as a rapid construction system for interactive 3D games, especially web-based 3D games, by showing some actual examples.

Related work

Our research purpose is to clarify software architecture that makes it easier to develop 3D graphics applications including interactive 3D games. Its related works are Virtual Reality construction toolkit systems including MR Toolkit[8], MERL[1] and so on. Most of them provide a script language like VRML[9] that allows developers to define the behavior of each object existing in a 3D virtual world. Some of them also provide facilities that enable developers to build distributed 3D virtual environments. However, these facilities are low-level functions so that developers have to make programs when building distributed applications. As well, there are some distributed Virtual Reality systems such as MASSIVE[2], dVS/dVISE[3], DIVE[4]. Although these are very powerful systems, it is not easy to use their essential mechanisms for end-users.

Our research system *IntelligentBox* provides various 3D software components represented as visible, manually operable, and reusable objects. Furthermore the *IntelligentBox* system provides a dynamic data linkage mechanism. These features make it easier for even end-users to develop 3D graphics applications including interactive 3D games, especially web-based 3D graphics applications including web-based 3D games. This is main

difference between *IntelligentBox* and others.

The remainder of this paper is organized as follows. Section 2 explains essential mechanisms of *IntelligentBox*. Section 3 shows example 3D games. Section 4 describes how *IntelligentBox* works as a SaaS application. Section 5 discusses the availability of *IntelligentBox* as a rapid construction system for interactive 3D games, especially web-based 3D games. Finally section 6 concludes this paper.

2. Essential mechanisms of *IntelligentBox*

IntelligentBox employs the following essential mechanisms.

2.1 *Model-Display Object* (*MD*) structure

As shown in Figure 1, each *box* consists of two objects, a *model* and a *display object*. This structure is called *MD* (*Model-Display object*) structure. A *model* holds state values of a *box*. They are stored in variables called *slots*. A *display object* defines how the *box* appears on a computer screen and also defines how the *box* reacts to user operations. Figure 1 also shows the messages passed between a *display object* and a *model*. This is an example of *RotationBox*, which has a *slot* named 'ratio' that holds a double precision number and stands for a rotation angle. This value is normalized into zero to one. One means one rotation. Through direct manipulation on a *box* using a 2D mouse device, its associated *slot* value is changed. For example, 'ratio' *slot* value of *RotationBox* is changed proportionally to the moving distance of the mouse device. Furthermore, its visual image change is simultaneous with the change of the *slot* value. In this way, a *box* reacts to the user's manipulation according to its functionality.

2.2 Message-sending protocol for slot connections

Figure 2 illustrates a data linkage concept among *boxes*. As shown in the figure, each *box* has multiple *slots*. Its one *slot* can be connected to one of the *slots* of other *box*. This connection is called a *slot connection*. The *slot connection* is carried out by some messages when there is a parent-child relationship between two *boxes*. There are three standard messages, i.e., a *set* message, a *gimme* message and an *update* message. These messages

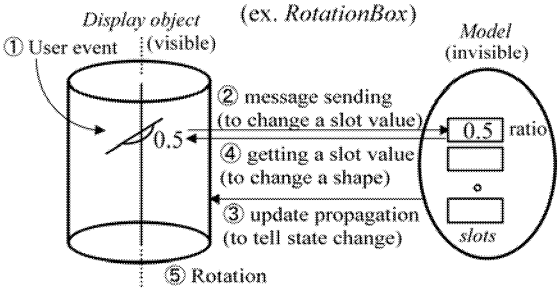


Figure 1. An *MD* structure of a *box* and its internal messages.

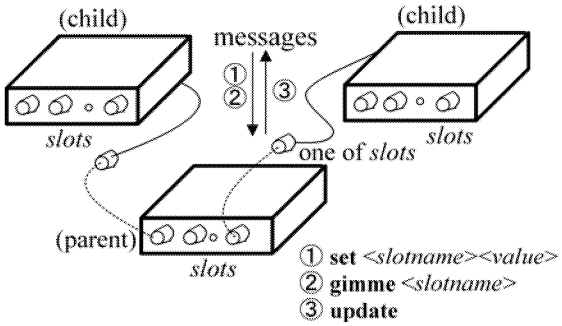


Figure 2. Standard messages between *boxes*.

have the following formats:

(1) Parent box *set* <slotname> <value>.
(2) Parent box *gimme* <slotname>.
(3) Child box *update*.

A <value> in a format (1) represents any value, and a <slotname> in formats (1) and (2) represents a user-selected *slot* of the parent *box* that receives these two messages.

A *set* message writes a child *box slot* value into its parent *box slot*. A *gimme* message reads a parent *box slot* value and sets it into its child *box slot*. *Update* messages are issued from a parent *box* to all of its child *boxes* to tell them that the parent *box slot* value has changed. Each *box* has three main flags that control the above message flow, i.e., a *set* flag, a *gimme* flag, and an *update* flag. These flags are properties of a *display object*. A *box* works as an input device if its *set* flag is set to true. Contrarily a *box* works as an output device if its *gimme* flag is set to true. A *box* sends *update* messages if its *update* flag is set to true. Then child *boxes* take an action depending upon the states of the *set* flag and the *gimme* flag after they receive an *update* message or after they individually change their *slot* values.

2.3 A simple composite *box* example

Figure 3 shows a simple composite *box* example. Figure 4 illustrates its message flow and data flow. In this example, the motor is a counter *box* having a cylindrical shape. Its model has a *slot* with a double precision number value that is increased automatically by a timer process. The toggle button

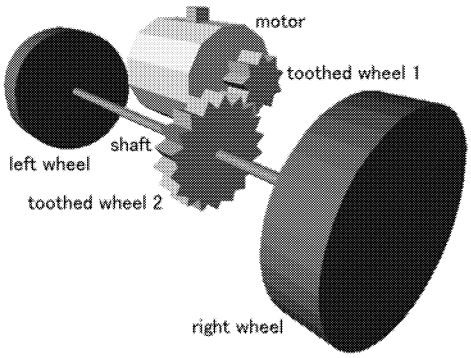


Figure 3. A composite *box* example.

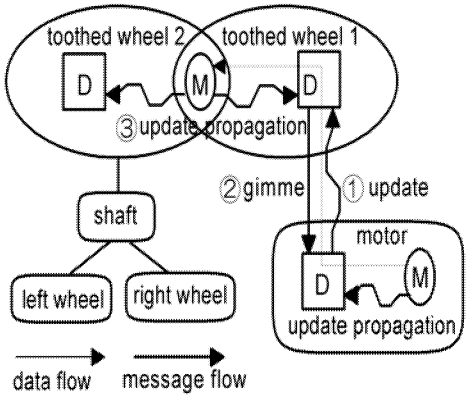


Figure 4. Message and data flow between *boxes.*

attached to the motor works as a switch that changes the state of the timer process. Pushing the toggle button activates the timer process and hence the motor. The *slot* value of the motor begins to increase automatically, and hence the motor begins to rotate. Then according to the data flow shown in the figure, the toothed wheel 1 and toothed wheel 2 come to rotate since both are *RotationBoxes*. Consequently, the toothed wheel 2 makes the shaft with the two wheels rotate.

3. Interactive 3D games

This section presents actual examples of interactive 3D games developed using *IntelligentBox.*

3.1 Table games

The chess is one of the most popular table games. Figure 5 shows a chess game. A player can move his own chessman through mouse-device operations. A left lower figure shows a structure of a chessman in a wire-frame image. Its base is *XYMoverBox*, which moves in the left-right and the near-far directions according to player's mouse-device operations. When a rendering mode is even a shading mode, *XYMoverBoxes* are invisible since their visibility parameter is set into false because they do not need to be seen.

Figure 6 shows a reversi, sometimes called Othello, as another popular table game. A player can move his own chip and put it since its base is also

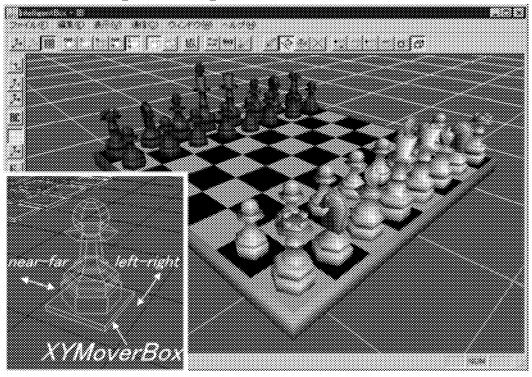


Figure 5. A chess board and chessmen.

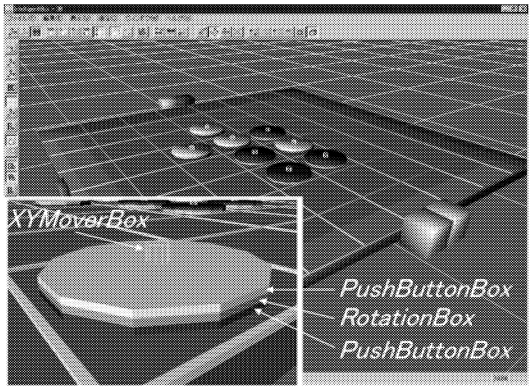


Figure 6. A reversi board and chips.

XYMoverBox. As shown in the left lower figure, each chip is composed from four *boxes*, i.e., *RotationBox*, two *PushButtonBoxes* and *XYMoverBox*. A white colored and a black colored disk are both *PushButtonBoxes* and a sandwiched disk is *RotationBox*. The white and the black colored *PushButtonBox* have a different *slot* value, i.e., 0.0 and 0.5 respectively, to rotate *RotationBox*. If the user clicks the white colored *PushButtonBox, RotationBox* rotates by a half rotation and the black colored *PushButtonBox* becomes top. Similarly if the user clicks the black colored *PushButtonBox*, *RotationBox* rotates by a half rotation and the white colored *PushButtonBox* becomes top. In this way, players can reverse a chip by a mouse click operation.

These table game systems do not have any thinking algorithms so that only person-to-person games are possible. These table game systems are playable on a network by using *RoomBoxes*[6,7]

3.2 Other interactive 3D games

Figure 7 shows other interactive 3D games, for instance, a jack-in-the-box, a slot machine and so on. These applications are constructed using already existing *boxes* without writing any text-based

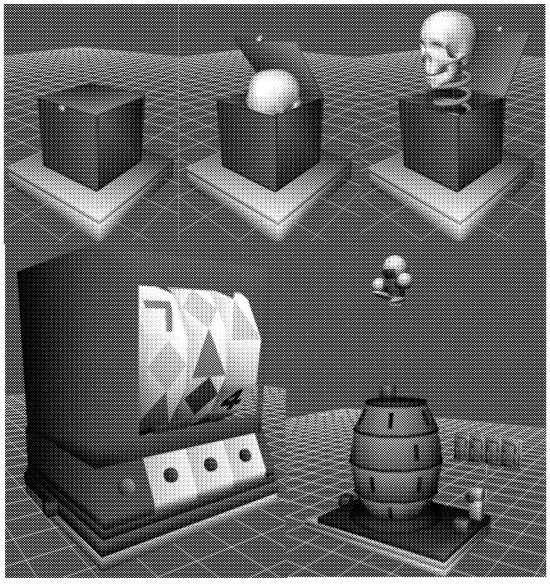


Figure 7. Interactive simple game examples.

programs. A developer develops constructively an application with interactively checking its behavior on a computer screen.

4. *IntelligentBox* as Development System for SaaS Applications

4.1 Basic mechanism

As explained above, using *IntelligentBox*, it is possible to make interactive 3D games only by combining already existing components, *boxes* through direct manipulations on a computer screen. However, developed 3D games can not be available on *WWW*. So, we extended our *IntelligentBox* system to make it be used as a development system for SaaS applications including web-based 3D games. Its mechanism is very simple as shown in Figure 8. The word of "SaaS" is a short term of "Software as a Service" means that any software is installed only at a web-server and the user can receive functionalities provided by the software through a web-browser as a web-service. By the mechanism shown in Figure 8, our *IntelligentBox* works as one of the SaaS applications and already developed 3D graphics applications using *IntelligentBox*, e.g., interactive 3D games, also work as SaaS applications. Figure 9 and 10 show web-based simple 3D game examples.

The *IntelligentBox* system uses OpenGL 3D graphics library which provides an off-screen rendering functionality. Using this functionality, a rendering image of a 3D scene can be generated on a web-server and transferred to a web-browser through the *Internet*. On a web-browser, besides a HTML program, a JavaScript program runs to manage user operation events, i.e., mouse move, mouse button click, down and up. Such user operation events will be transferred to the web-server through a CGI-program of the web-server. The CGI-program (Perl script) once receives the user operation events and applies them to *IntelligentBox*. And them, *IntelligentBox* generates next off-screen rendering image of the 3D scene updated by the user operation events. In this way, the user can interactively manipulate the web-based *IntelligentBox* that runs on the web-server through his/her web-browser.

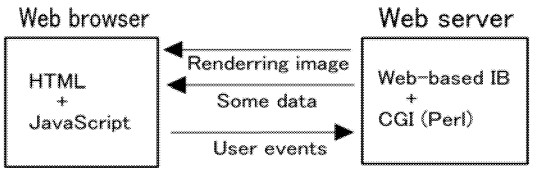


Figure 8. Data-linkage between web-server and web-browser.

4.2 Multiple users of the same web-based IB

Web-based IB can provide multiple users with a web-based collaborative virtual environment as shown in Figure 11. Web-based IB has a System ID No. (SID No.) , and by specifying it, each client user

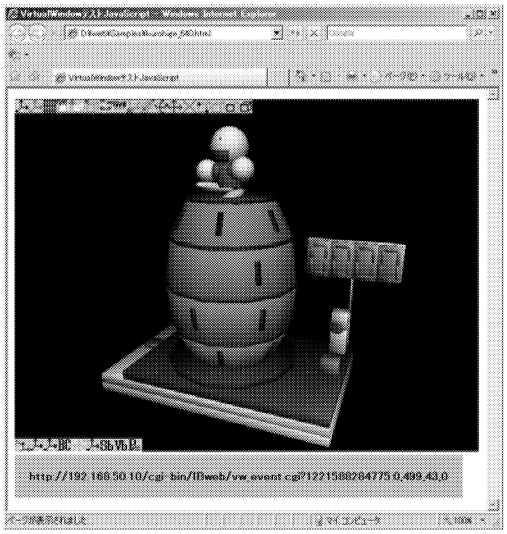


Figure 9. Web-based 3D game example 1.

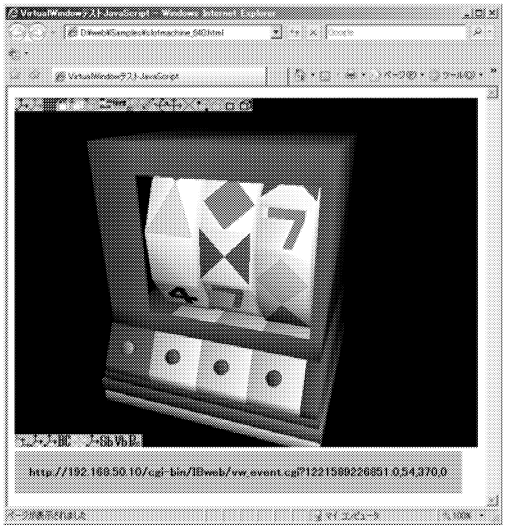


Figure 10. Web-based 3D game example 2.

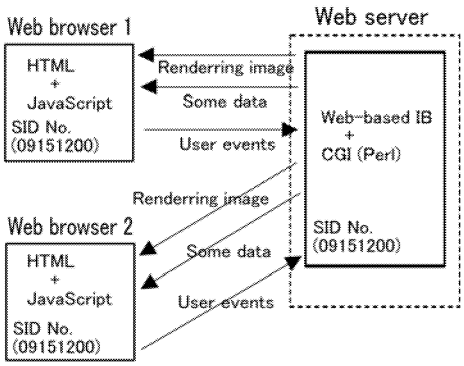


Figure 11. Data-linkage among one web-based IB on web-server and its two clients (web-browsers).

can access its corresponding web-based IB that has the same SID No. through his/her web-browser. Using this data-linkage, on-line network games can be constructed very easily.

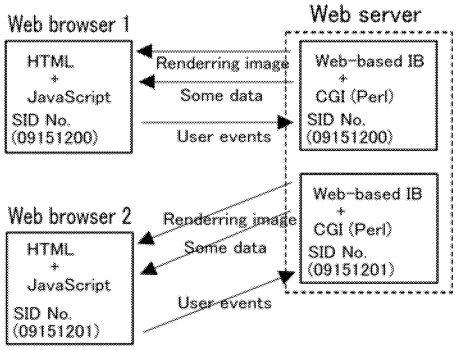


Figure 12. Data-linkage among two individual web-based IBs on web-server and their two dedicated clients (web-browsers).

4.3 Multiple users of individual web-based IBs

Figure 12 shows a case that each of multiple users individually uses his/her own web-based IB. As shown in the Figure, each of multiple IBs can run on the same PC, and using different System ID No.(SID No.), each user can access his/her own web-based IB through his/her web-browser.

5. Discussion

Currently, the *Internet* and *WWW* have become popular and widely common because of broadband network technologies. There are many services on *WWW*. However, there are very few services of 3D graphics applications on *WWW*. One of its reasons is that web-browsers natively do not support the 3D graphics rendering. If we want to receive 3D graphics application services on *WWW*, we have to install any plug-in software like Java3D, Jogl, etc. This is inconvenient especially for end-users who are not accustomed to doing so. Even if this is not inconvenient, there is another reason that 3D graphics applications need 3D polygonal model data but they are usually not allowed to be distributed through the *Internet* due to their user-licenses. Even if the distribution of 3D polygonal model data is allowed, there is a problem that the transmission time of them is very long.

On the other hand, our web-based *IntelligentBox* works as a SaaS application does not have the above problems because only off-screen images of 3D scenes are transferred from a web-server to a client web-browser as explained in Sec. 4. So, interactive 3D games already developed as applications of *IntelligentBox* can be reused as web-based 3D games those work as SaaS applications. Its MashUp with other SaaS applications are also possible. As a result, the usefulness of *IntelligentBox* has become higher.

6. Concluding remarks

This paper proposed software architecture that makes it easier to develop 3D graphics applications including interactive 3D games. Our research system *IntelligentBox* provides various software components. Each of them corresponds to each of required functionalities. Combining these software components by direct manipulations on a computer screen enables users to develop application systems without writing any text-based program. This feature is important for end-users who do not have any programming knowledge. Furthermore, we enhanced the usefulness of *IntelligentBox* by extending it to make it work as a development system for SaaS applications including web-based 3D games.

In this paper, we clarified the availability of *IntelligentBox* as a rapid construction system for interactive 3D games, especially web-based 3D games, by showing some actual example.

References

1. Anderson DB, Barrus JW, Howard JH, Rich C, Shen C, Waters RC, Building Multiuser Interactive Multimedia Environments at MERL. IEEE Multimedia, 2(4): 77-82, 1995.
2. Greenhalgh, C., Benford, S., MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading. Proceedings of IEEE 15th International Conference on Distributed Computing Systems(DCS'95),Van couver, Canada: 27-34, 1995.
3. Grimsdale, C., dVS-Distributed Virtual Environment System. Proceedings of Computer Graphics '91, London, UK, Bleinheim Online, pp. 163-170, 1991.
4. Hagsand, O., Interactive Multiuser VEs in the DIVE System. IEEE Multimedia, Vol. 3, No. 1, pp. 30-39, 1996.
5. Okada, Y. and Tanaka,Y., IntelligentBox: A Constructive Visual Software Development System for Interactive 3D Graphic Applications, Proc. of Computer Animation ''95, IEEE Computer Society Press, pp. 114-125, 1995.
6. Okada,Y. and Tanaka,Y., Collaborative Environments in IntelligentBox for Distributed 3D Graphic Applications, Proc. of Computer Animation '97, IEEE Computer Society Press, pp. 22-30, 1997.
7. Okada,Y. and Tanaka,Y., Collaborative Environments in IntelligentBox for Distributed 3D Graphic Applications, The Visual Computer (CGS special issue), Vol. 14, No. 4, pp. 140-152, 1998.
8. Shaw C, Green M, Liang J, Sun Y (1993) Decoupled Simulation in Virtual Reality with the MR Toolkit. ACM Transaction on Information Systems, Vol. 11, No. 3, pp. 287-317.
9. Ames, A. L, Nadeau, D. R. and Moreland, J. L., The VRML Sourcebook, John Wiley & Sons, Inc. , 1996.

TRANSFORMATION AND INTERPOLATION WITH *VQM* STRUCTURE

Xin Li, *Cody Luitjens* and *Michael Beach*
Computer Science Department
Digipen Institute of Technology
Redmond, Washington, U.S.A.
xli@digipen.edu

ABSTRACT

Quaternions are preferred in 3D graphics and video games as the "weapon of choice for rotations" due to their efficiency, accuracy and elegance to represent rotation and its interpolation. The *VQM* map is defined to combine a rotation by a quaternion with a translation and non-uniform scale in affine transformations [2]. This paper addresses the practicality issues of *VQM* structures when used in character animations and real-time simulations. It introduces simplified procedures for *VQM* transformation and concatenation. The paper also presents an incremental method to interpolation algorithms between two *VQM* key frames. With fixed intervals in distance, angle and size, the proposed approach remarkably reduces the *VQM* transformation and concatenation expenses. Moreover, all interpolations for *VQM* key frames are performed implicitly at no cost. In other words, the computationally expensive trigonometric functions in quaternion interpolation Slerp and exponential function in scalar interpolation Elerp are completely eliminated from the process. The method is efficient, accurate and numerically stable without compromising geometric characteristics of the original transformation methods.

KEYWORDS: *3D Transformation, Simulation and Video Game, Quaternion Rotation, Key Frame Interpolation, Character Animation.*

1. INTRODUCTION

In real-time simulation and video game applications, motions of articulated objects are implemented by 3D hierarchical transformations containing translation, rotation and scale as the components. In practice, to save the storage, animations are often constructed and stored as transformation key frames. At run time, the key frames are interpolated to produce intermediate frames for smooth motion.

There are many different notations to represent the hierarchical transformation. Using a 4x4 homogeneous matrix allows composition of all three transformation components in one date structure. This notation is popular since it is convenient for geometric transformation and concatenation, also since it is supported by hardware of the graphics pipeline and therefore efficient to produce the final image.

However, the matrix notation not only suffers from gimbal-lock when used to represent three consecutive rotations around orthogonal axes (such as in the Fixed-Angle or Euler notations), but also is notorious for its errors during direct interpolations, resulting distorted transformation and undesired results. To overcome this shortcoming, one can decompose the non-singular transformation matrix into translation, rotation and scale components [1]. Then interpolations can be performed between pairs of same type of components. The interpolated results are grouped together to form the intermediate transformation matrix. The decomposition approach allows the components to be processed by different interpolation algorithms, for instance, Lerp for translation and scale, and Slerp for rotation, yielding more accurate data for intermediate animation frames. All of this, of course, is at the cost of extra computation for non-trivial decomposition and composition of matrices.

For this reason, many applications prefer to generate animation key frames with three transformation components stored separately. A notation known as *VQS* is such a data structure to serve the purpose [3]. In a *VQS* structure, the translation is stored as a vector, the rotation as a quaternion, and scale as a scalar. The approach allows three transformation components to be interpolated separately using the aforementioned Lerp and Slerp for translation and rotation respectively, while using Lerp or Elerp for the scaling factors (See discussion in Section 4). It is also easy to support the geometrical operations such as concatenation and multiplication with a vector. The main drawback, however, is that it only allows a single value as the scaling factor in the transformation, which means that all three dimensions of the object have to be stretched uniformly.

The motivation of this research is to introduce a mechanism that, (1) allows to store transformation components separately as animation key frames, (2) utilizes different interpolation algorithms per transformation component for more desired results, (3) is mathematically sounds for supporting hierarchical transformation and concatenation, and (4) yet does so with better efficiency in character animations without losing the geometrical characteristics of original interpolation algorithms.

The rest of the paper is organized as follows: In section 2, a new abstract algebraic system called *VQM* is introduced and some of its algebraic properties are discussed. Section 3 shows how *VQM* is used as the transformation structure to combine a rotation by a quaternion with a translation and non-uniform scaling factors and how it supports transformation concatenation through multiplication. The following section 4 discusses the general interpolation algorithms applied upon each *VQM* component. A more efficient *VQM* interpolation method is then presented in section 5, which allows remarkable reduction of the computational time, assuming a constant incremental interpolation step. Section 6 analyzes the cumulative errors during the incremental approach. The final section of the paper summarizes the outcome of this research.

2. *VQM* STRUCTURE

The *VQM* structure $\mathcal{T}^*$ is an algebraic system, where any $T = [v, q, M] \in \mathcal{T}^*$ is a triplet with v as a *pure quaternion*, q a non-zero quaternion, and M an invertible *homogeneous quaternion matrix* [2]. That is,

$$\mathcal{T}^* = \{T \mid T = [v, q, M],\ q \neq 0,\ M^{-1} \text{ exists}\}.$$

Note that a *homogeneous quaternion matrix* is a 4×4 matrix $M=|m_1, m_2, m_3, m_4|$ with the first three column vectors m_i , (i=1, 2, 3), as pure quaternions and m_4 as the identity quaternion. In other words,

$$M = \begin{vmatrix} (\alpha_{ij}) & \vec{0}^t \\ \vec{0} & 1 \end{vmatrix}, \quad i, j \in \{1,2,3\},$$

where $\vec{0}$ is a 3 dimensional zero vector and (α_{ij}) is the 3×3 sub-matrix formed by the α_{ij}-components of m_i's .

Let $T_1=[v_1, q_1, M_1]$ and $T_2=[v_2, q_2, M_2]$ be *VQM* structures and $M_1 = \begin{vmatrix} (\alpha_{ij}) & \vec{0}^t \\ \vec{0} & 1 \end{vmatrix}$ and $M_2 = \begin{vmatrix} (\beta_{ij}) & \vec{0}^t \\ \vec{0} & 1 \end{vmatrix}$ be homogeneous quaternion matrices. Let c be a scalar. Then *addition, scalar product* and *multiplication* on the *VQM* set are defined in [2] as follows:

$$\begin{aligned} &T_1 + T_2 = [v_1+v_2, q_1+q_2, M_1+M_2], \\ &cT_1 = [cv_1, cq_1, cM_1], \\ &T_2T_1=[q_2(M_2v_1)q_2^{-1}+v_2, q_2q_1, (q_1^{-1}\cdot M_2\cdot q_1)(q_1\cdot M_1\cdot q_1^{-1})] \end{aligned} \tag{E2.1}$$

where $M_1+M_2 = \begin{vmatrix} (\alpha_{ij} + \beta_{ij}) & \vec{0}^t \\ \vec{0} & 1 \end{vmatrix}$ and $cM_1 = \begin{vmatrix} (c\alpha_{ij}) & \vec{0}^t \\ \vec{0} & 1 \end{vmatrix}$.

In the above definition for multiplication, the operator '·' is the multiplication of a quaternion with a homogeneous quaternion matrix. That is, for any quaternion q and a 4×4 homogeneous quaternion matrix $M=|m_1, m_2, m_3, m_4|$ (i.e. m_1, m_2, m_3 and m_4 as column vectors), $q\cdot M\cdot q^{-1} = |qm_1q^{-1}, qm_2q^{-1}, qm_3q^{-1}, qm_4q^{-1}|$ represents multiplication of a quaternion and its inverse with the column vectors (hence quaternions) from the left and right sides respectively. Geometrically, this operation always results in another homogeneous quaternion matrix rotated by the given quaternion [2].

It is proven that the *VQM* structure is closed under defined multiplication. Also it is shown that the multiplication is associative and non-commutative, that it has an identity element with respect to multiplication, and that each structure in the set has an inverse, therefore, that the *VQM* set forms a non-Abelian group [2].

Other algebraic properties on the *VQM* set and its components are also discussed in [2]. We list a few below for the convenience of the discussion in this paper, where q, q_1 and q_2 are quaternions, M and N are homogeneous quaternion matrices, M_q is a matrix representing the same rotation by q, and r is a pure quaternion:

$$q\cdot M\cdot q^{-1} = M_qM \tag{E2.2}$$

$$q(Mr)q^{-1} = (q\cdot M\cdot q^{-1})r \tag{E2.3}$$

$$(q_2\cdot N\cdot q_2^{-1})(q_1\cdot M\cdot q_1^{-1}) = q_2\cdot(N(q_1\cdot M\cdot q_1^{-1}))\cdot q_2^{-1} \tag{E2.4}$$

3. *VQM* TRANSFORMATION AND CONCATENATION

As the homogeneous matrix, the *VQM* structure allows combining a rotation quaternion, a translation vector, and non-uniform scaling factors for a three dimensional transformation. Let $r=[a, b, c, 0]$ be a pure quaternion with $<a, b, c>$ as a vector. Let $T = [v, q, M]$ be a *VQM* structure, where v is a pure quaternion, $q = [x, y, z, w]$ is a non-zero quaternion (with $u = <x, y, z>$ as the vector part and w as the scalar part), and M is a 4×4 homogeneous quaternion matrix. A transformation of r with T is defined as

$$r' = [a', b', c', 0] = [v, q, M]\, r = q(Mr)q^{-1} + v$$

Please note that, if we assume that M is a "diagonal matrix" representing a homogeneous scaling matrix in a transformation of an animation key frame, then M becomes a special case of "homogeneous quaternion matrix". In this case, the *VQM* transformation of vector r is to scale it by M, rotate by q, and then translate by v. It is easy to verify that the result is equivalent to a transformation by a 4×4 homogeneous matrix that contains the same scaling, rotation and translation parameters. The operation always generates another pure quaternion r' with its vector part as the 3D vector after the transformation.

In practice, r, v and r' in the *VQM* transformation can be reduced from quaternions to 3D vectors by removing the zero-scalar part in the pure quaternion, and M can be simplified to a 3×3 matrix by removing the fourth row and column. This modification allows computing the transformation with *VQM* by the following equation:

$$\begin{aligned} r' &= (w^2-u^2)(Mr) + 2(u\cdot(Mr))u + 2w(u\times(Mr)) + v \\ &= M_q(Mr) + v \\ &= (M_qM)r + v \end{aligned}$$

where M_q is the 3×3 matrix corresponding to the same rotation by $q = [u, w]$ (See [1]).

We will assume this notation simplification through the discussion in the paper. Note that all algebraic properties regarding its definitions and operations of *VQM* presented in [2] are still valid under this assumption.

When transforming with a sequence of *VQM* structures, one can concatenate them by *VQM* multiplications as defined in [2]. That is, if $T_2 = [v_2, q_2, M_2]$ and $T_1 = [v_1, q_1, M_1]$ are two *VQM* structures, for any vector r,

$$\begin{aligned} &T_2\,(T_1\,r) = [v_2, q_2, M_2]\,([v_1, q_1, M_1]\,r) \\ &= [q_2(M_2v_1)q_2^{-1} + v_2,\ q_2q_1,\ (q_1^{-1}\cdot M_2\cdot q_1)(q_1\cdot M_1\cdot q_1^{-1})]\,r \end{aligned}$$

We notice that, by definitions and *VQM* properties,

$$\begin{aligned} &T_2\,(T_1\,r) = [v_2, q_2, M_2]\,([v_1, q_1, M_1]\,r) \\ &= [v_2, q_2, M_2]\,(q_1(M_1r)q_1^{-1} + v_1) \\ &= q_2(M_2(q_1(M_1\cdot r)q_1^{-1} + v_1))q_2^{-1} + v_2 \\ &= q_2(M_2(q_1(M_1r)q_1^{-1}))q_2^{-1} + q_2(M_2v_1)q_2^{-1} + v_2 \\ &= q_2((M_2(q_1\cdot M_1\cdot q_1^{-1}))r)q_2^{-1} + [v_2, q_2, M_2]v_1 \\ &= [\,[v_2, q_2, M_2]v_1,\ q_2,\ M_2(q_1\cdot M_1\cdot q_1^{-1})\,]\,r \\ &= (T_2T_1)\,r \end{aligned}$$

where

$$T_2T_1=[[v_2, q_2, M_2]v_1, q_2, M_2(q_1\cdot M_1\cdot q_1^{-1})\,] \qquad \text{(E3.3)}$$

Equation (E3.3) indicates that the concatenation of two *VQM* structures always yields another *VQM* structure, where the translation component is obtained by transforming the first translation vector v_1 with the second *VQM* structure $[v_2, q_2, M_2]$, the rotation component q_2 is exactly the same as that in the second *VQM*, and the scaling component is the concatenation of two scaling matrices $M_2(q_1\cdot M_1\cdot q_1^{-1})$, with the first one M_1 rotated into the second transformation space. Comparing with the definition in [2], (E3.3) generates the same result but allows saving one quaternion multiplication (in rotation concatenation) and two quaternion rotations (in scaling matrix rotation).

4. *VQM* INTERPOLATION

In real-time interactive simulation and video game software, *VQM* structures can be created as key frames and interpolated to generate details of animation sequences. This can be done by various methods. For instance, if $T_0 = [v_0, q_0, M_0]$ and $T_n = [v_n, q_n, M_n]$ represent two key frames in *VQM*, then $T_t = [v_t, q_t, M_t]$, for any $t \in [0.0: 1.0]$, can be calculated by the *VQM* scalar product and addition:

$$T_t = (1-t)T_0 + tT_n \qquad \text{(E3.2)(E4.1)}$$

The equation implements the uniform-linear interpolation of three transformation components (i.e. translation, rotation and scaling factors). It generates another *VQM* structure as the intermediate transformation between two key frame *VQM* structures. This approach is adequate for some applications, but could be insufficient for others since the linear interpolation of unit quaternions generally does not yield a unit quaternion and may introduce variable change rate of rotation angles between two orientations. In practice, we allow the three transformation components to be processed separately by variform algorithms to improve the accuracy and to preserve the geometric characteristics of different operations, for instance, as follows:

(1) Linear interpolation (Lerp):

(2) $$v_t = (1-t)v_0 + tv_n$$

(3) (2) Spherical linear interpolation (Slerp):

(4) $$q_t = \frac{1}{\sin(\alpha)}\left[\sin(\alpha - t\alpha)q_0 + \sin(t\alpha)q_n\right] \qquad \text{(E4.2)}$$

(3) (3) Exponential linear interpolation (Elerp):

(5) $$M_t = M_s^t M_0$$

Note that, in Equation (E4.2) above, α is the angle between two quaternions $q_0 = [u_0, w_0]$ and $q_n = [u_n, w_n]$, M_s^t is a diagonal scaling matrix computed by the powers of ratios of scale parameters in M_n and M_0. That is, if the diagonal elements of M_n and M_0 are $\langle x_n, y_n, z_n\rangle$ and $\langle x_0, y_0, z_0\rangle$ respectively, that of M_s^t will be $\langle (x_n/x_0)^t, (y_n/y_0)^t, (z_n/z_0)^t\rangle$.

The exponential interpolation for the scaling factor intends to compensate the non-linearity of delusional perception of the human eye since it is more sensitive to changes in smaller objects than that in larger ones. Figure 3-1 compares the linear and exponential approaches between scales of 0.1 and 10 in 100 steps.

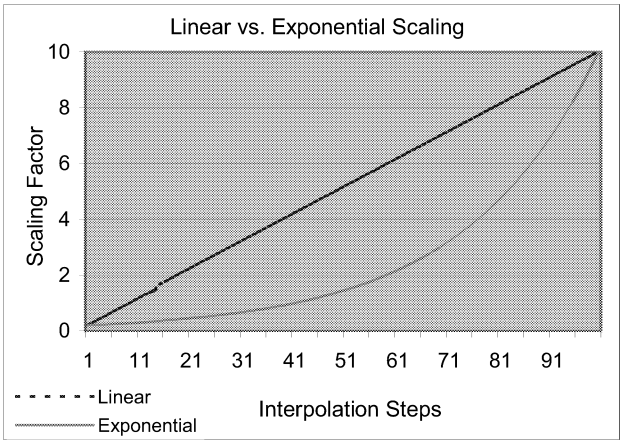


Figure 4-1: Comparison of Linear and exponential scaling

Once interpolated, $T_t = [v_t, q_t, M_t]$ is used to create a smooth animation sequence between the key frames. Please note that, since the *VQM* interpolation is always performed between two key frames that contain only diagonal matrices as non-uniform scaling parameters, therefore M_t is always a diagonal matrix (a special case of homogeneous quaternion matrix before the simplification in the last section). The authors wish, by pointing this fact out, to avoid any confusion.

5. INCREMENTAL *VQM* TRANSFORMATION

Since the spherical linear interpolation is utilized, algorithms in (E4.2) inherit all superior geometric characteristics (constant velocity, geodesic path and unit length) for interpolated quaternions. However, two trigonometric functions in Slerp are too computationally costly and often prohibit this approach to become practical [6]. In this section, we introduce an approach that allows overcoming this disadvantage in many applications, that is, to transform with and interpolate *VQM* structures incrementally.

First we assume that, when interpolating or concatenating between two *VQM* key frames T_0 and T_n, the operations can be done incrementally through n steps with a fixed interval for locations, angles and scales. Under this assumption, the change of parameter t in domain 0.0 to 1.0 can be treated as an incremental procedure with a constant interval $\Delta t = 1/n$. Thus we obtain integer k such that $t = k\Delta t = k/n$, for $k = 0, 1, \ldots, n$.

This assumption allows us to apply an augmentative method to simplify the complexity of the problem and reduce the cost. The approach takes advantage of the gradual and progressive procedure and benefits from the knowledge obtained in the previous steps. In this section, we discuss how this concept is employed first on interpolation algorithms, and then on the transformation concatenation.

We understand that there are applications that do not have the luxury of constant time step. However, there are also applications that require a constant frame rate, for instance, a large number of video game software run at a fixed 60Hz frame rate. In those cases, since the character animations are often pre-constructed and stored in key frames with specific time stamps, the interpolation interval is indeed a constant. Even in the case when the frame rate is not fixed, if the "current frame rate" at a particular time is detectable, one can still calculate the time step for the next key frame interpolation and use it as a close approximation with the incremental method to save time.

5.1 INCREMENTAL *VQM* INTERPOLATION

Under the incremental assumption, after replacing t with k/n, equations in (E4.2) become

$$\begin{aligned} v_k &= k v_c + v_0, \ \text{where } v_c = (v_n - v_0)/n \\ q_k &= q_c^k q_0, \qquad \text{where } q_c = [\cos(\beta),\ \sin(\beta)u] \\ M_k &= M_0 M_c^k, \ \text{where } M_c = \begin{vmatrix} x_c & 0 & 0 \\ 0 & y_c & 0 \\ 0 & 0 & z_c \end{vmatrix} \end{aligned} \qquad \text{(E5.1)}$$

$$\text{and } x_c = \left(\frac{x_n}{x_0}\right)^{\frac{1}{n}},\ y_c = \left(\frac{y_n}{y_0}\right)^{\frac{1}{n}},\ z_c = \left(\frac{z_n}{z_0}\right)^{\frac{1}{n}}$$

Above equations are called incremental interpolations, among which the conversions of the first and the third equations from (E4.2) are straightforward. The derivation of the incremental Slerp for quaternion is discussed in full length in [4]. It states that the slerped quaternion q_k can be computed by multiplication of q_0 and the k-th power of a constant q_c, where $\beta = \alpha/n$ and $u = (w_0 u_n - w_n u_0 + u_0 \times u_n)/\sin(\alpha)$ is a unit vector determined by components of key frame quaternions $q_0 = [u_0, w_0]$ and $q_n = [u_n, w_n]$.

Equations in (E5.1) also allow us to rewrite the incrementally-interpolated *VQM* $T_{k+1} = [v_{k+1}, q_{k+1}, s_{k+1}]$ as recursive functions of $T_k = [v_k, q_k, M_k]$ from the previous interpolation step. That is, with $T_0 = [v_0, q_0, M_0]$ at $k = 0$, we always have the following at any $k = 1, \ldots, n$:

$$\begin{aligned} v_{k+1} &= (k+1)v_c + v_0 = v + (k v_c + v_0) = v_c + v_k \\ q_{k+1} &= q_c^{k+1} q_0 = q_c (q_c^k q_0) = q_c\, q_k \\ M_{k+1} &= M_0 M_c^{k+1} = (M_0 M_c^k) M_c = M_k M_c \end{aligned} \qquad \text{(E5.2)}$$

Note that, in (E5.2), all v_c, q_c and M_c are constants and therefore can be pre-computed once for given T_0, T_n and interpolation steps n. With these recursive definitions, transformations by a sequence of interpolated *VQM* structures, from T_0 to T_n in n steps of fixed intervals on any arbitrary vector r, can be written as

$$\begin{aligned} T_k r &= [v_k, q_k, M_k] r \\ &= q_k(M_k r)q_k^{-1} + v_k \\ &= (q_k \cdot M_k \cdot q_k^{-1}) r + v_k \\ &= A_k r + v_k \end{aligned}$$

$A_k = (q_k \cdot M_k \cdot q_k^{-1}) = |q_k m_{k1} q_k^{-1}, q_k m_{k2} q_k^{-1}, q_k m_{k3} q_k^{-1}, q_k m_{k4} q_k^{-1}|$ is the combined rotation and scaling matrix at the k-th interpolation step. Note that Equation (E2.3) was applied in the above derivation. Carrying the interpolation along the same line for one more step and replacing the interpolated translation, rotation and scaling components at k+1 with the recursive expression (E5.2), we obtain the following:

At *0*-th step:

$$\begin{aligned} T_0 r &= [v_0, q_0, M_0] r \\ &= A_0 r + v_0, \qquad \text{where } A_0 = q_0 \cdot M_0 \cdot q_0^{-1} \end{aligned}$$

At (k+1)-th step (k=0, 1, …, n-1):

$$\begin{aligned} T_{k+1} r &= [v_{k+1}, q_{k+1}, M_{k+1}] r \\ &= q_{k+1}(M_{k+1} r)q_{k+1}^{-1} + v_{k+1} \\ &= (q_c q_k)((M_k M_c) r)(q_c q_k)^{-1} + (v_k + v_c) \\ &= q_c(q_k(M_k(M_c r))q_k^{-1})q_c^{-1} + (v_k + v_c) \\ &= (q_c \cdot (q_k \cdot M_k \cdot q_k^{-1}) \cdot q_c^{-1})(M_c r) + (v_k + v_c) \\ &= ((q_c \cdot (q_k \cdot M_k \cdot q_k^{-1}) \cdot q_c^{-1}) M_c) r + (v_k + v_c) \\ &= A_{k+1} r + v_{k+1} \end{aligned}$$

with

$$\begin{aligned} A_{k+1} &= (q_c \cdot (q_k \cdot M_k \cdot q_k^{-1}) \cdot q_c^{-1}) M_c \\ &= (q_c \cdot A_k \cdot q_c^{-1}) M_c \\ &= M_{qc} A_k M_c \qquad \text{(E5.3)} \\ v_{k+1} &= v_k + v_c \end{aligned}$$

where M_{qc} is the 3×3 matrix corresponding to the rotations by q_c. Again, in the above derivation steps, *VQM* algebraic property (E2.2) and (E2.3) were applied.

Equations in (E5.3) state that, when interpolated incrementally, the *VQM* transformation at each step can be calculated based on the previous step. Taking the advantage of diagonal scaling matrix M_c, A_{k+1} can be computed with a matrix multiplication plus 9 scalar multiplications, while v_{k+1} is merely a vector addition. Since M_{qc}, M_c and v_c are constants, once key frames T_0 and T_n are given, the computational savings for incremental interpolations, including Lerp, Slerp and Elerp for transformation components, are remarkable.

5.2 INCREMENTAL *VQM* CONCATENATION

We now take a look at how *VQM* transformations are concatenated when interpolated incrementally. Let's assume that animations of an articulated object are constructed and stored in a hierarchical structure by key frames. If, at the k-th interpolation step, T_k and S_k represent *VQM* structures of a child node and its parent node respectively in this hierarchy, i.e. $T_k = [v_k, q_k, M_k]$ and $S_k = [u_k, p_k, N_k]$, then from (E3.3), for any vector r defined in the child local space, to apply the child transformation T_k on and then the parent transformation S_k, we have

$$\begin{aligned} S_k(T_k r) &= [u_k, p_k, N_k]([v_k, q_k, M_k] r) \\ &= p_k(N_k(q_k(M_k r)q_k^{-1}))p_k^{-1} + p_k(N_k v_k)p_k^{-1} + u_k \\ &= (p_k \cdot (N_k(q_k \cdot M_k \cdot q_k^{-1})) \cdot p_k^{-1}) r + (p_k N_k p_k^{-1}) v_k + u_k \end{aligned}$$

With the incremental approach, we can compute the concatenated transformation for (k+1)-th step by replacing the components with their recursive expressions as follows:

$$\begin{aligned} S_{k+1}(T_{k+1} r) &= [u_{k+1}, p_{k+1}, N_{k+1}]([v_{k+1}, q_{k+1}, M_{k+1}] r) \\ &= (p_{k+1} \cdot (N_{k+1}(q_{k+1} \cdot M_{k+1} \cdot q_{k+1}^{-1})) \cdot p_{k+1}^{-1}) r \\ &\quad + (p_{k+1} \cdot N_{k+1} \cdot p_{k+1}^{-1}) v_{k+1} + u_{k+1} \\ &= (p_c p_k \cdot ((N_k N_c)(q_c q_k \cdot (M_k M_c) \cdot q_k^{-1} q_c^{-1})) \cdot p_k^{-1} p_c^{-1}) r \\ &\quad + (p_c p_k \cdot (N_k N_c) \cdot p_k^{-1} p_c^{-1}) v_{k+1} + u_{k+1} \\ &= (p_c p_k \cdot (N_k N_c) \cdot p_k^{-1} p_c^{-1})(q_c q_k \cdot (M_k M_c) \cdot q_k^{-1} q_c^{-1}) r \\ &\quad + (p_c p_k \cdot (N_k N_c) \cdot p_k^{-1} p_c^{-1}) v_{k+1} + u_{k+1} \end{aligned}$$

Note that (E2.4) was applied in the derivation. Putting in the recursive terms based on (E5.3), we have

$$\begin{aligned} A_{k+1} &= M_{qc} A_k M_c \\ v_{k+1} &= v_k + v_c \\ B_{k+1} &= N_{qc} B_k N_c \\ u_{k+1} &= u_k + u_c \end{aligned}$$

and

$$S_{k+1}(T_{k+1} r) = B_{k+1} A_{k+1} r + B_{k+1} v_{k+1} + u_{k+1} \qquad \text{(E5.4)}$$

Equations in (E5.4) outline the incremental concatenation based on the recursive functions, which can be implemented with the following data structures and pseudo code:

Data Structures:

- The articulated object is modeled and stored in the hierarchical structure.
- At each node in the hierarchical structure, animation key frames (T_0 and T_n) are specified as *VQM* structures.

- For each pair of $T_0 = [v_0, q_0, M_0]$ and $T_n = [v_n, q_n, M_n]$, M_{qc}, M_c and v_c are pre-computed as constants based on a fixed number of interpolation steps n.
- For each node, $node.A_k$ and $node.v_k$ are used to store temporary results for the current interpolation step k.
- A *stack* is utilized to store the current concatenated pair $[B_k, u_k]$ such that $r' = B_k \cdot r + u_k$ represents the transformation of vertex r at k-th interpolation step of the current node in the tree.

Algorithm:

Initialize the following at interpolation step 0:

For each *node* in the tree

$node.A_k = (node.q_0)\cdot(node.M_0)\cdot(node.q_0^{-1})$;
$node.v_k = node.v_0$;

push $[I, 0]$ on the *stack* top;

Do the following at next interpolation step k:

(1) $currentNode = root$;

(2) Compute the current local transformation:

$currentNode.A_k = (currentNode.M_{qc})\ (currentNode.A_k)\ (currentNode.M_c)$;

$currentNode.v_k = currentNode.v_k + currentNode.v_c$;

(3) push the *stack* top $[B_k, u_k]$ to duplicate;

(4) compute the new *stack* top to concatenate transformations
$u_k = B_k\ (currentNode.v_k) + u_k$;
$B_k = B_k\ (currentNode.A_k)$;
replace the *stack* top with $[B_k, u_k]$;

(5) for each vertex r in *currentNode*, transform with $r' = B_k\, r + u_k$;

(6) loop for each *childNode* of *currentNode*
$currentNode = childNode$;
recursively process currentNode (step (2) ~ step (7));

(7) Pop $[B_k, u_k]$ from the *stack* top to restore the previous transformation.

The algorithm recursively processes each node at the next interpolation step, from the root to leaf nodes. Each step it computes the transformation parameters for the current node based on the previous step at the same node. Once the geometry transformation is done, the algorithm traverses one level down the tree to process the children nodes, calculates the local transformations and concatenates them with the current one stored in the stack.

Note that, since the increments of translations, rotations and scales are all constant and therefore pre-computed only once in matrix forms, there is no cost at all for interpolations of transformation parameters during the concatenation.

6. ERROR ANALYSIS

As mentioned earlier, the incremental approach reduces the computational expense of *VQM* interpolations. It incorporates Lerp, Slerp, and Elerp with no cost. It is immune to the numerical instability and requires no parameter tweak [7] [8]. However, due to the incremental approach based on fixed steps, the implementation on any hardware and software platforms would introduce some cumulative floating-point round-off errors. Transformations by this approach would inevitably deviate from the path of standard *VQM* interpolations.

To quantify accumulated errors and examine the "drifting" behavior, we designed a series of tests using 200 pairs of *VQM* structures containing randomized parameters within controlled ranges: [-999.0 : 999.0] for translations, [1° : 180°] for rotations and [0.1 : 10.0] for each component's scale. 200 random vertices are transformed with Lerp, Slerp, and Exponential interpolations over 100 fixed steps using the standard as well as the incremental approach.

Let $r = \langle x, y, z\rangle$ be the result of a vector transformed by a *VQM* structure computed using the standard interpolation method. Let $r' = \langle x', y', z'\rangle$ be the result of the same vector transformed at the same interpolation steps using the incremental approach instead. For each pair of r and r', four relative errors are computed as follows:

$$\sigma_x = \frac{|x - x'|}{|r|},\ \sigma_y = \frac{|y - y'|}{|r|},\ \sigma_z = \frac{|z - z'|}{|r|},\ \sigma_d = \frac{|r - r'|}{|r|}$$

where σ_x , σ_y , and σ_z are discrepancies of x, y and z components and σ_d is the distance between the vectors computed by the two different approaches. To avoid the "magnitude effect", all errors are normalized by $|r|$. The maximum errors among all 200 transformed vectors are recorded at each interpolation step. For the distance, the average errors at each step are also logged.

As expected, locations and components of the incrementally interpolated *VQM* transformations tend to drift away from the standard *VQM* transformations. The maximum cumulative round-off error increases with the number of interpolation steps. However, our tests also indicate that the drifting behavior is insignificantly trivial. The average relative error for components and distance are less than 0.0003% even after 100 steps of incremental interpolations and transformations. In the worst of cases, all maximum round-off errors are well below 0.004%. It means that, if the

transformed point is 1 meter from the origin in a given coordinate space, the deviation of the incrementally transformed vector would be only 0.04 millimeters away from its "correct location" in the worst case.

Nevertheless, incrementally concatenating *VQM* transformations introduces further cumulative floating-point round-off errors. This is because the multiplications of deviated *VQM* structures would inevitably enlarge any small inexactness. To quantify the "magnified drifting behavior", we conducted a series of tests that are similar to those described earlier on a hierarchical structure with 10 nodes in depth, which hold 200 pairs of *VQM* key frames to interpolate between. The same *VQM* ranges, used in the previous tests, are enforced. 200 random vertices are transformed by concatenating the full hierarchy over 100 fixed steps, both regularly and incrementally.

As expected, the recorded errors are worse when incrementally concatenated. Also like in the previous case, the drifting is insignificant. The average relative distance error is less than 0.0015%, while all maximum round-off errors for each component and the distance are below 0.3% in the worst of cases at the leaf nodes. Note that the single-precision 32-bit floating point was used in all tests. The round-off errors should be further reduced under double-precision 64-bit operations.

7. CONCLUSION

The *VQM* structure is introduced to represent three-dimensional transformation with a non-uniform scaling matrix. Since quaternions are utilized in *VQM*, the transformation does not suffer from the gimbal-lock and interpolation errors. The incremental *VQM* approach discussed in this paper proposes an efficient method for keyframe-based animation with a fix time interval, which combines all interpolation and concatenation together in one single step. Furthermore, it is trivial to convert the recursive form of interpolated and concatenated *VQM* transformation into a homogenous matrix so that the transformation can be easily combined with the camera and projection matrices.

The benchmark test on a Pentium D 2.80 GHZ, 2GB RAM and Windows XP platform demonstrates tremendous savings. It is almost 10 times faster than the non-incremental approach in (E4.2). Besides, the algorithm inherits all superior geometric characteristics (constant velocity, geodesic path and unit length) of the Slerp algorithm.

In conclusion, the proposed incremental algorithms are fast and accurate enough for most character animations in video game and 3D real-time simulation software, while the cumulative errors introduced by incremental steps are insignificant and therefore can be ignored during the interpolations.

REFERENCES:

[1] Ken Shoemake, "Animating Rotation with Quaternion Curves", Computer Graphics, Volume 19, Number 3, 1985.

[2] Michael Aristidou and Xin Li, "The VQM-Group and its Applications", Accepted by International Journal of Algebra, May, 2008.

[3] Warren Robinett and Richard Holloway, "The Visual Display Transformation for Virtual reality", Technical Report (TR94-031), UNC-Chapel Hill, Sept. 1994.

[4] Xin Li, "To Slerp, Or Not to Slerp", Game Developer Magazine, August, 2006.

[5] Ken Shoemake and Tom Buff, "Matrix Animation and Polar Decomposition", Proceedings of the conference on Graphics Interface, 1992.

[6] Jonathan Blow, "Understanding Slerp, Then Not Using It", Game Developer Magazine, April 2004.

[7] Jonathan Blow, "Hacking Quaternions", Game Developer Magazine, March 2002.

[8] F. Sebastian Grassia, "Practical parameterization of rotations using the exponential map", Journal of Graphics Tools, volume 3.3, 1998.

ARTIFICIAL INTELLIGENCE

AI BEHAVIOUR

AUTOMATABLE EVALUATION METHOD ORIENTED TOWARD BEHAVIOUR BELIEVABILITY FOR VIDEO GAMES

Fabien Tencé*,** and Cédric Buche*
* Université Européenne de Bretagne - ENIB
LISyC - CERV
25, rue Claude Chappe
29280 Plouzané (France)
** Virtualys
41 rue Yves Collet
29200 Brest (France)
{tence,buche}@enib.fr

KEYWORDS
Artificial intelligence, agent evaluation, believable behaviour, human tracking

ABSTRACT

Classic evaluation methods of believable agents are time-consuming because they involve many human to judge agents. They are well suited to validate work on new believable behaviours models. However, during the implementation, numerous experiments can help to improve agents' believability. We propose a method which aim at assessing how much an agent's behaviour looks like humans' behaviours. By representing behaviours with vectors, we can store data computed for humans and then evaluate as many agents as needed without further need of humans. We present a test experiment which shows that even a simple evaluation following our method can reveal differences between quite believable agents and humans. This method seems promising although, as shown in our experiment, results' analysis can be difficult.

INTRODUCTION

The vast majority of video games features computer controlled virtual characters, also known as agents, to enrich their environment. For games to be able to suspend disbelief (Bates 1992), those agents should be *believable*. However, this term have two distinct meanings: first, the ability to be a good role-player and second, the ability to be indistinguishable from a human player (Livingstone 2006). We aim at agents taking on the role of players, so we will use the latter definition in this paper.

When modelling or implementing an agent's behaviour, an important step is evaluation. It can be very useful to improve the final result by tuning the model's parameters or by modifying the model itself. Evaluation is almost mandatory to validate work and should be able to answer: to what extent does the agent meet the objectives? or is it better than what have been done before? However, evaluation of believable agents is rather difficult because believability is based on the observers' feelings. Many studies deal with this problem like (Mac Namee 2004; Livingstone 2006; Gorman et al. 2006). However, they still rely on humans to judges agents. As a result, this kind of method cannot be used to evaluate a large number of different behaviours. Moreover, the use of questionnaires for evaluation is criticised (Slater 2004).

Knowing these problems, we propose a method using a different approach: we aim at reducing human intervention as much as possible still helping researchers assessing artificial behaviours' believability. The objective is to have a method which could be automatised and thus which can be used in optimisation algorithms or for tests during the night. For this, we chose to measure artificial behaviours' likeness to human behaviours, a concept close to believability. Of course, this method does not aim at replacing classic ones but rather offers a complementary approach so that the final result looks more believable.

EVALUATION METHOD PROPOSITION

Principle

The evaluation's general principle is to compute vectors which are features of humans' and agents' behaviours. We call those vectors "signatures" as they should be designed to be representative of a behaviour. By measuring the distance between an agent's signature and humans', it should be possible to tell if its behaviour looks like a human behaviour or not.

Protocol

Here are the main stages of the protocol to evaluate an agent with respect to humans:

1. Define behaviour signatures and metrics between them;
2. Monitor humans (the more the better) and compute the signatures;
3. Monitor the agent to be evaluated and compute the signatures in the same conditions as the monitored humans;
4. Compute the distance between each agent's signature and its human equivalent.

The first part is very challenging because we need to find signatures having a low variance for humans, still being able to detect non-believable behaviours. This could be a problem when humans are given a great freedom because their behaviour can vary widely. To determine good signatures, it can be necessary to run the protocol several times with few humans and to modify signatures and/or metrics so that they have nice properties.

For the second and the third steps, human and agents must be studied under the same conditions. Since agents do not play against humans, if virtual characters coexist in the same environment we have to make a concession: for the step 2, humans are together and for the step 3, agents are together. If we introduce agents in step 2, signatures could be flawed by reflecting interaction between humans and non-believable agents.

We presented our method as been automatable. Indeed, step 3 and 4 can be redone without further need of humans. However, automatable does not mean fast as step 3 can take some time. To determine the experiment's duration, a study should be done in order to test how much time the signatures need to stabilise.

Monitoring agents and humans

As explained in the protocol, we must monitor both humans and agents so that we can compute signatures. To be totally independent from the internal architecture, we chose to take the same point of view as a judge looking over the subject's shoulder. Therefore, in the well-known perception-decision-action loop we can only have access to the actions and the perceptions. We assume that the loop can be observed for human as well so that we can compare agents' and humans' perceptions and actions in the same manner.

The principle of the monitoring is to keep track of a subset of these perceptions and actions during simulations so as to build the signatures. For example, in a first person shooter, we could track basic actions and perceptions such as if the actor is jumping and if it sees an enemy. It could then be possible to build a very simple signature measuring the proportion of jumps when an enemy is visible.

There are three main kind of signatures: perception-based, action-based and those linking actions to perceptions. Perception-based signatures are not very useful because players judges agents on their actions. Action-based can be useful but may be too simple to explain complex behaviours. Finally, signatures linking actions to perceptions are the most interesting ones because they may find patterns in the decisions. However, there is a difficulty: if an information is monitored in perceptions, it is not sure that the agent or the human noticed this information.

EXPERIMENT

For the experiment, we used the game Unreal Tournament 2004. It is a first person shooter game, in other words each player or agent controls a unique virtual character and sees through its eyes. The character can, non-thoroughly , grab items (weapons, ...), move (walk, jump, ...) and shoot with a weapon. Each character have an amount of hit points, also known as life points: each time an actor is hit by an enemy fire, a certain amount of hit points are subtracted to the current value. When hit points reaches zero, the character "dies" but can usually reappear at another place in the virtual environment. Although the concept can seem very basic, it can prove challenging for agents to mimic humans.

This experiment's main objective is to give a concrete example and to show the interest of the method. Therefore, the signatures are simple and would be insufficient for a real evaluation. Moreover, for the sake of simplicity, this sample experiment try to assess only the believability of Unreal Tournament 2004 agents' movements. Here are the signatures:

- Velocity change angle: 20-dimensions vector, the value of the i-th component is the number of time we measured an angle of approximatively i between $\overrightarrow{V}_t$ and $\overrightarrow{V}_{t+1}$. Note that the vector as only 20 components so an angle of 20 corresponds to a whole turn, 10, half turn, etc. This vector is then normalised.
- Velocity relative to the direction: 20-dimensions vector, the value of the i-th component is the number of time we measured an angle of approximatively i between $\overrightarrow{V}_t$ and $\overrightarrow{D}_t$. The same unit as the

previous signature is used for angles. This vector is then normalised.

where t is the current time step and $t+1$ is the following one, time steps occurring every 125 ms. $\overrightarrow{V}$ is the velocity vector without its z component, $\overrightarrow{D}$ is a vector pointing toward the character's aiming point without its z component (figure 1). The z component was ignored for the signatures because it simplifies the signatures without losing too much information.

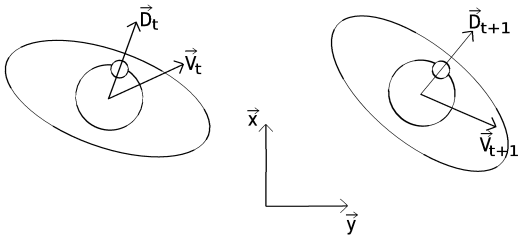


Figure 1: Scheme of a virtual character viewed from top view at two following time steps, t and $t+1$. $\overrightarrow{D}$ is the direction vector, pointing toward the character's aiming point and $\overrightarrow{V}$ is the velocity vector.

The experiment is composed of two steps:

1. 6 low/medium level human played a *deathmatch* game during 20 minutes during which signatures were computed. We chose a duration of 20 minutes because signatures tend to stabilise after 15 to 20 minutes. A *deathmatch* follows the basic rules and the objective is to "kill" as many other players as possible. Each human plays using a keyboard and a mouse to control the virtual character and a classic desktop computer screen to see the virtual environment.

2. Signatures are computed for 8 agents having 8 different skill levels. Those agents fight against each other in 6-agent matches in the same topography, following the same rules and during the exact same time.

We used tools developed by our team and based on Pogamut 2 (Burkert et al. 2007). They are available at `svn://artemis.ms.mff.cuni.cz/pogamut/branches/fabien_tence`. The experiment gave us 2 signatures for each of the 14 subjects, each signature having 20 components. A sample of those results is given in figure 2.

RESULTS' ANALYSIS

In order to visualise the data, we chose to do a Principal Components Analysis (PCA) on the signatures. Subjects are represented using the two principal components in figures 3 and 4.

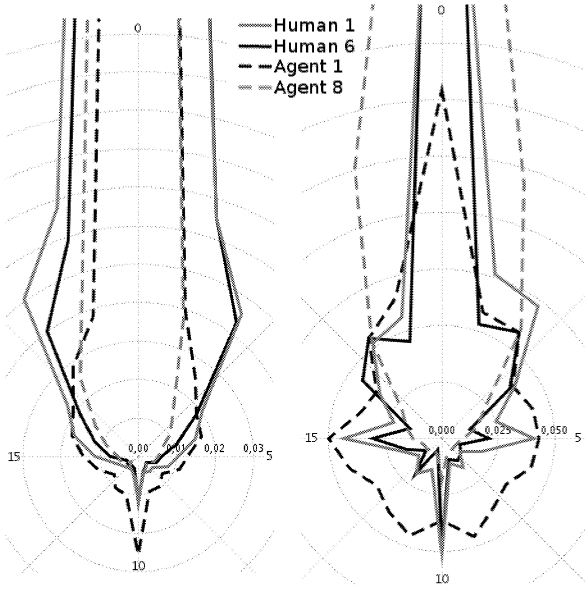


Figure 2: Sample of the signatures. On the left "velocity change angle" signatures and on the right "velocity relative to the direction" signatures. The graphs are truncated for the 0^{th} component for the sake of readability.

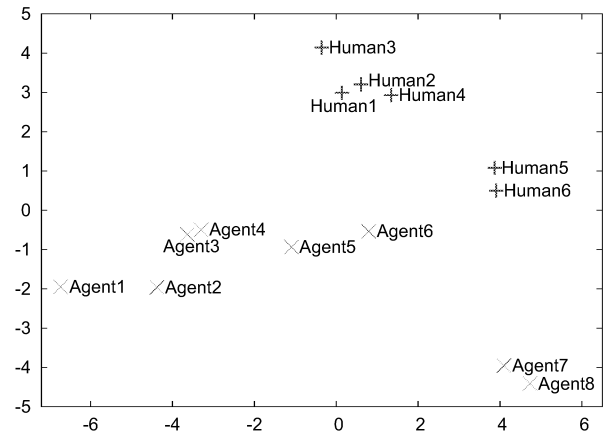


Figure 3: Subjects represented using the two principal components which explain 90.37% of the variance for the "velocity change angle" signatures.

The first chart (figure 3) represents very well the data with about 90% of the variance represented. The first PCA axis is negatively correlated with brutal angle changes and the second PCA axis is positively correlated with small angle changes. The second chart (figure 4) is less accurate with only 78% of the variance represented; however, it does suffice to extract some interesting data's features. The first PCA axis is globally negatively correlated with walking backwards and backwards+sideways. The second PCA axis is positively correlated with walking forward+slightly sideways and negatively with forward+sideways.

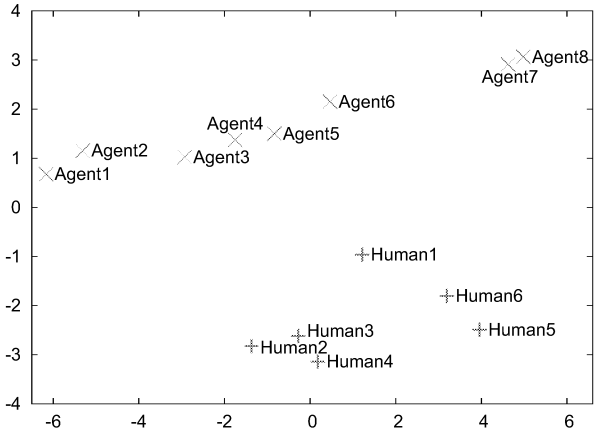
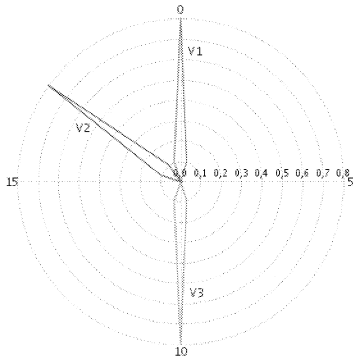


Figure 4: Subjects represented using the two principal components which explain 78.18% of the variance for the "velocity relative to the direction" signatures.

The most important information that both charts reveal is that there is a difference between humans' and agents' behaviours: it is possible to draw a line which separate humans from agents on the two charts. To study the influence of efficiency, agents and humans are ranked depending on their score (number of "kills"). As humans did not play against agents, these ranks apply only within those two groups: Human1 is more efficient than Human2, which is more efficient than Human3, etc. Agent1 is more efficient than Agent2, etc. but we do not know if Agent1 is more efficient than Human1. It seems that the *skill* parameter which influence the agents' efficiency have a quite artificial effect: agents are ordered on the first principal component on both charts.

Even if those results are promising, PCA suffers from a flaw in our case: in the signatures, a component i is much closer in term of angle to the $i + 1^{th}$ and $i - 1^{th}$ than to the $i + 2^{th}$ and $i - 2^{th}$ component, which is not taken into account by the PCA. To calculate the distance between two signatures considering this particularity, we chose to use the Earth Mover's Distance (EMD) (Rubner et al. 2000). Figuratively speaking, the distance between two vectors V_1 and V_2 is equal to the minimum effort made to carry earth from the "relief" V_1 to the "relief" V_2. The interest of this metric is explained in the figure 5.

With the EMD, we can build a matrix of dissimilarities by computing the distance between each signature. To visualise these dissimilarities, we used a method called MultiDimentionnal Scaling (MDS) which consists in representing each vector by a point in a plan. The goal of MDS is to approximate a given dissimilarity matrix between vectors by a matrix of Euclidean distances (L_2) between the points representing the vectors. The result of the MDS method applied to dissimilarities matrices computed with EMD on our signatures are given in figures 6 and 7. Note that there is no need for a scale because it is the relative distance between points that is important, not the real distance.



	EMD	L_2
V_1,V_2	3.0	1.15
V_1,V_3	9.6	1.15
V_2,V_3	7.0	1.15

Figure 5: Example showing the interest of the EMD compared to Euclidean distance (L_2) in our case. The normalised vectors not having any common component, the L_2 distance is then equal between each other. However, it is obvious that V_1 and V_2 are the most similar vectors which is detected by the EMD.

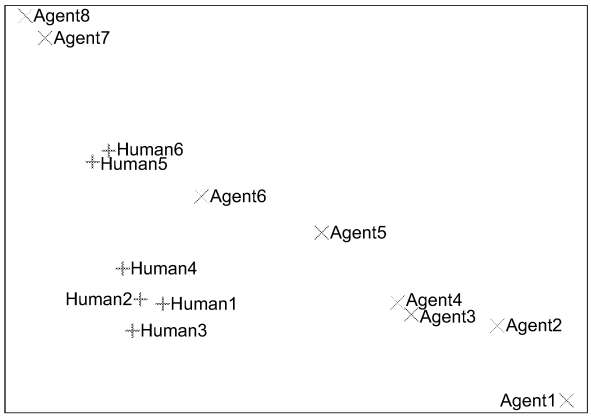


Figure 6: Result of a MDS applied to EMD matrix computed on "velocity change angle" signatures. This representation represents well the original distances as the stress is a bit less than 0.003.

The results are a bit different from what we get with the PCA analysis: humans' signatures are much more mixed with the agents' than with the PCA. In the figure 7 we can see that humans and agents have quite similar behaviours although there are still some differences between the two. We can also note that all agents are on a line in figures 6 and 7 which reinforce the idea that the *skill* parameter have a quite linear effect. It seems that this effect is not pertinent in the case of the figure 6 because as the humans' efficiency rise they tend to move away from the agents'.



Figure 7: Result of a MDS applied to EMD matrix computed on "velocity relative to the direction" signatures. This representation represents well the original distances as the stress is a bit more than 0.003.

The PCA and the MDS analysis seems to give quite different results. The MDS is based on the EMD which measure the difference between global signatures' shapes. From this point of view, agents are quite close to humans so we can think that they might be believable. However, the PCA analysis, being more strict, allows us to clearly distinguish human and artificial behaviours. This difference can come from the fact that humans have limitations to control their virtual character because they use keyboards. Evaluated agents do not copy those limitations, as a result, they may have a global human-like behaviour but they can be recognised if we look closely. Note that our two signatures are designed to study the velocity vector so all our conclusion applies only to the way agents move. We should have designed a lot more signatures in order to study much more aspects of the behaviours.

CONCLUSION

The proposed method seems promising as it could help in assessing the believability of a behaviour. Its main advantage is that it can evaluate a large number of agents, allowing finer improvement of the models' parameters. This advantage is due to the principle of signatures, vectors which characterise behaviours' aspects. We found out in our test experiment that even simple signatures can give interesting results.

However, there is still some questions that should be answered: to what extent do humans notice variations in behaviours? and do signatures have the same sensitivity as humans? Studies confronting the current method to classic ones should be done to evaluate signatures. Even if those studies are time-consuming and complex they will result in useful and reusable signatures.

The next step is to use this method in a behaviour modelling project. The goal is to optimise the models according to an evaluation method based on our present work. The final agents will then be evaluated in a classic way, with subjects judging agents in their environment.

We will, however, need to improve the type of signatures for the evaluation to be more precise. Presented signatures are *global signatures*, they are computed at each time step. *Contextual signatures* are computed only when certain perceptions and/or actions based conditions are met. They will provide more meaningful information about behaviours. Another new type of signature can be useful, *temporal signatures* which will measure time between events. It can be used to measure reaction time which is an important factor of believability.

REFERENCES

Bates J, 1992, The Nature of Characters in Interactive Worlds and The Oz Project, Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon University.

Burkert O, Kadlec R, Gemrot J, Bida M, Havlicek J, Dorfler M and Brom C, 2007, Towards Fast Prototyping of IVAs Behavior: Pogamut 2, *Lecture Notes in Computer Science*, **4722**, 362–363.

Gorman B, Thurau C, Bauckhage C and Humphrys M, 2006, Believability testing and bayesian imitation in interactive computer games, *Lecture Notes in Computer Science*, **4095**, 655–666.

Livingstone D, 2006, Turing's Test and Believable AI in Games, *Computer Entertainment*, **4**(1), 6.

Mac Namee B, 2004, Proactive Persistent Agents: Using Situational Intelligence to Create Support Characters in Character-Centric Computer Games, PhD thesis, University of Dublin.

Rubner Y, Tomasi C and Guibas L, 2000, The Earth Mover's Distance as a Metric for Image Retrieval, *International Journal of Computer Vision*, **40**(2), 99–121.

Slater M, 2004, How colorful was your day? why questionnaires cannot assess presence in virtual environments, *Presence: Teleoperators & Virtual Environments*, **13**(4), 484–493.

Evolving believable behaviours for first person shooter games

Dave Saunders and Colm O'Riordan
Department of Information Technology,
NUI, Galway,
Ireland.
email: colm.oriordan@nuigalway.ie

KEYWORDS
Artificial Intelligence

ABSTRACT

This papers presents research in the use of genetic programming techniques to generate believable behaviours for NPCs in first person shooter games. We use the Quake II game and engine in this work to illustrate our approach. We report aspects of the NPC behaviours deemed unrealistic by a number of human game players. We abstract these scenarios into an abstract game wherein we evolve fit behaviours for our NPCs. These evolved scripts are then placed back into the original Quake II engine. We report on the evolved strategies in the abstract game and also on feedback from a survey of game players on their view of these behaviours in the game engine.

INTRODUCTION

The immersive aspect of a game should not be damaged by unrealistic or unbelievable behaviour in the context of the game. Given the increased number of multi-player games, users are becoming more accustomed to playing against intelligent players who play in a believable manner. This has increased the demand for NPCs that exhibit believable behaviours.

There exist a number of approaches to generating NPC behaviour(hard coding scripts, varying these scripts and use of fuzzy logic in choosing behaviours). These all represent expensive approaches in that a number of programmers and designers are required to create such behaviours.

We hypothesise that genetic programming can be used to generate behaviours that are more realistic (and enjoyable to play against) than the oftentimes monotonous and predictable behaviours exhibited by NPCs in modern computer games.

Genetic programming, an evolutionary computation technique, allows the evolution of successive populations of solutions to problems. One of the big advantages of genetic programming is that its representation is amenable to analysis. The solutions are represented as trees of conditions and actions. Upon evolving good solutions, we can analyse these solutions to gain interesting insights into the problem space being searched.

We use the Quake II first person shooter game as a test bed for our work. We examined the game and identified, through conversations with gamers familiar with Quake II and other games, a number of limitations regarding the believability of the NPCs. We chose two of these limitations and set about evolving NPC behaviours that would overcome these identified limitations and that would also hopefully increase the enjoyment of players playing the game.

As Quake II is a complex game with complex environments with an associated substantial time required to play the game or play a level in the game, we abstracted the game to simple two-dimensional game which still captured the salient features associated with the limitations we identified. In doing so, we were able to reduce the time needed to evaluate a strategy. By abstracting the game, we rendered the problem computationally tractable. We evolved strategies to overcome the identified limitations in our two dimensional world, saved the best strategies (highest fitness) and stored these as scripts to be mapped into the real Quake II game.

In the next section, we describe briefly some related work. In section III, we outline our approach including an overview of the identified limitations, our abstract model, our experimental approach and finally the mechanism used for mapping the evolved behaviours back in to the Quake II engine. We then discuss our results. In the final section we present our conclusions.

RELATED RESEARCH

Evolutionary Computation

Evolutionary computation refers to class of approaches inspired by the notion of Darwin's theory of evolution. These approaches involves the breeding of successive generations of candidate solutions to a particular problem. At each generation, the better solutions are chosen (usually stochastically) and subjected to mutation (effectively a form of perturbation to effect local search) and recombination (a form of global search acheived by combining selected sub-solutions together). There are many approaches which differ mainly in representation.

These include, among others, Rechenberg's evolution strategies (Rechenberg (1973)) which used mutation on candidate solutions, genetic algorithms (Holland (1975)) which use strings of bits or values to represent candidate solutions, and genetic programming (Koza (1992)) wherein the individual solutions are represented as trees.

AI in Current Games

Behaviours in first person shooter games are often described as being monotonous and predictable. The failings in the behaviours of NPCs in such games are due to the type of AI implemented to control them and much work has been undertaken in the academic area to improve on this. In current FPS games, finite state machines (FSM) are the most commonly used NPC behaviour control(Rabin (2002)). With this architecture, the NPC resides in one of a number of finite states which can change due to the NPC's surroundings. For example the Quakebots in Quake had five states in which the NPC could reside; Idle, Search, Attack, Hurt, Dead. Corresponding actions are associated with each state. This method of behaviour control is often chosen, as the size of the FSM is reasonable and therefore readily maintained(Cavazza (2000)). However, the drawback of such an approach is the fact that the number of states the NPC can occupy is finite and will eventually lead to lead to monotonous actions.
Rule-based AI approaches use a set of rules to determine the appropriate action to for a certain scenario. In the case of QuakeII the NPCs have 800 rules which can be examined and a choice of 100 actions can be chosen (Cavazza (2000)).
In order to achieve the goal of abolishing pre-defined actions and achieve a human-like and intelligent behaviour, some form of learning must be implemented in the NPCs. Some commercial games have attempted to implement learning, one such game being Cloak, Dagger and DNA (Diller and Ferguson (2004)). In this game, sequences of the opponent NPCs DNA can compete to form new DNA and hence new behaviours by means of a genetic algorithm. In this way the opponent's skill and actions alter during game play creating a more realistic behavioural set of actions.

Evolutionary Computation in Games

Evolutionary algorithms provide the capability for searching large spaces of possible behaviours and learning the most effective. It is on this model which Hussain has based his Advocates and Critics for Tactical Behaviours (ACTB) model(Hussain and Vidaver (2006)). ACTB creates an initial population of possible tactical plans and evaluates each one. Rather than having a single fitness function this model incorporates a number of evaluation metrics each with different goals. Cole (Cole and Louis (2004)) has also implemented genetic algorithms to improve on NPCs behaviour in the FPS Counter Strike.
Shichel implemented a GP-evolved behaviour in the game Robocode (Shichel and Ziserman (2005)). His evolved behaviours were human competitive in tournaments against strategies designed by humans.
Rather than focussing on the behaviours of a single NPC, Luke (Luke and Hohn (1997)) used GP to evolve the behaviours for the eleven NPCs in a soccer team. Doherty has evolved and analysed a range of team behaviours in FPS shooter games (Doherty and O'Riordan (2007)). More recently, genetic programming has been used to evolve ambient characters in games (Cunningham and O'Riordan (2008)) and has been used successfully to generate interesting strategies for RTS type games (Keaveney and O'Riordan (2008)).

MODEL

Identifying limitations

In the current game, NPCs oftentimes do not seem to value their own health; for example, when under fire instead of trying to evade, the NPCs often just continue their current action which leads to a decrease in their health. Moreover, in the current game NPCs do not compete with player characters for resources such as healthpacks. This would be deemed by many as a flawed behaviour since a realistic character would value their own well being. Providing a means to preserve this well being can be achieved by the introduction of two new actions, namely *Get Healthpack* and *Evade*. Evading provides the NPC with the ability to retreat from combat, therefore reducing the risk of being shot, while retrieving a healthpack prolongs the NPCs life. Providing these new actions allows the NPC the opportunity to learn that the probability of achieving its goal of killing the player character is greatly increased by maintaining its own life.

Genetic Program Design

In the original Quake II, the following parameters or environmental variables (*Fire Visibility of Player, Hunt Distance to Player*) are used to chose an action from the following set (*Move to Target, Crouch, Charge, Strafe*).
In order to evolve behaviours to overcome the aforementioned limitations, we extended these sets to include several more environmental variables (*IF_VISIBLE, IF_PLAYER_HEALTHY, IF_FAR* etc.) to allow NPCs base their actions on issues such as their health, availability of health packs etc. We also extended the list of actions available to include the following (*EVADE, FIRE, HUNT, GET_HEALTH*). This represents a sufficient set of variables and actions to learn more believable behaviours.

Abstract Game

Upon examination of Quake II, the main objects needed in the game environment, with which the NPCs can interact in order to learn believable behaviours, are obstacles (i.e. walls and crates) and healthpacks. With this in mind a simple 2-dimensional environment can be created consisting of walls, healthpacks and free space. One player character and one NPC are placed in the 2-dimensional environment and have access to basic movement and firing commands. As in the original game the player character and NPC starting positions should be chosen so that both characters are not visible to one another. To automate the player character a simple rule set shall be adopted which conforms to the style of play of most, namely to hunt for the NPCs when they cannot see them and to fire when they become visible. NPCs shall behave according to the evolved genetic program being evaluated. Both characters shall take turns at either moving or firing until one character is killed or they each have taken a predefined number of turns. A fitness will then be awarded to the genetic program based on the evaluation metric defined previously.

Mapping evolved behaviours back to Quake II
Once all behaviours have been evaluated, the best is chosen to be inserted back into Quake II. A module was created which interprets the tree structure of the evolved genetic program. This module liaises between the genetic program and game. When a node is encountered which is found to be a conditional statement, the module queries the game for the result and returns it to the genetic program to inform it which branch to choose next. When a terminal node is encountered the module maps the node to an existing function in the game. This module is used as a replacement to the existing behavioural function and so Quake II can now be played using the evolved GP behaviours.

RESULTS

Tests on the effectiveness of the methods evolved in this work were undertaken both analytically and by having testers play through a level of Quake II with the new behaviours implemented. Analytical tests were carried out to measure the improvement of NPC behaviours. Gamers were asked to assess the effectiveness of the new behaviours since believability in games is a perception rather than a measurable metric.

Analytical Results

A number of different experiments were run based on varying parameters (population size, number of generations, mutation and crossover rates). The effects of these parameters were mainly to do with speed of convergence, computational cost and a full description of

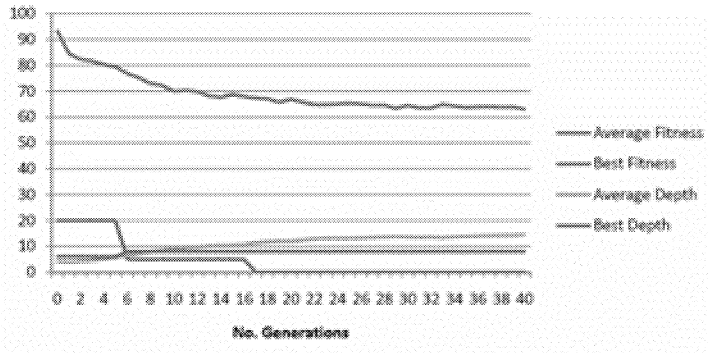


Figure 1: Evolutionary Run: fitness versus generations

these parameters is beyond the scope of this paper. One parameter of particular interest was the length of the solutions generated. Typically, the smaller the evolved solution, the more compact and generalisable the solution. Of course, the computational cost of executing the correspond script in the Quake II engine will be reduced for smaller trees. We include a plot of one typical run to illustrate the convergence in the evolutionary search. The plot included is typical of the experiments and is included to illustrate how the evolutionary search moves from a population of random solutions to solutions capable of defeating simple player strategies.

Survey

Together with the analytical results produced, a survey was conducted with a small sample of people (11) to determine their opinion of the newly created behaviours. In creating such a survey the guidelines suggested by Livingstone (Livingstone (2006)) relating to believability of NPC behaviours were adhered to. Livingstone breaks down believable behaviours into three categories, namely the ability to plan, the ability to act and the ability to react. Testers were asked to describe any variety seen in NPC behaviours in an effort to determine how well these NPC were at planning. Testers were also asked to comment on the realism of behaviours, i.e. if varied behaviours were noticed were these behaviours acceptable under the circumstances. The ability to act is determined by how well an NPC implements chosen behaviours, i.e. is there a noticeable pause while the NPC chooses an action to carry out.
The final category in determining believable behaviour questions the NPC's ability to react to changes in environment. Such changes may include the presence of an enemy or ally or the change in health of a character or the presence of a wall which can be hid behind. Since the GP did not train NPCs to react in the presence of other NPCs, reactions under such circumstances are not expected in these behaviours. Testers were asked to comment on behaviours which seemed to change depending

on the situations encountered. Novel behaviours were also noted, as realistic behaviours can often be uncharacteristic and irrational. Testers were given three versions of the game to test in a manner similar to blind testing, i.e. they were unaware which version of the game they were playing. One version of the game, version 3, was the original Quake II and the other two versions included the newly developed GP behaviours:

Version 1: `(if_healthpack (if_far (if_visible get_health fire) get_health) (if_npc_healthy (if_visible fire hunt) (if_healthpack get_health fire)))`

Version 2: `(if_far (if_visible (if_player_healthy (if_visible fire hunt) (if_healthpack get_health fire)) (if_player_healthy (if_visible get_health fire) (if_healthpack hunt get_health))) (if_player_healthy (if_healthpack get_health (if_visible fire hunt)) (if_far hunt (if_npc_healthy (if_visible hunt fire) (if_healthpack evade fire)))))`

These two strategies were among the best (in terms of fitness) that the evolutionary process discovered. The first one uses a number of environmental variable and primitives regarding their health scores. The latter one also takes into account health scores but also exploits the *evade* parameter.
All three versions were altered to reduce the amount of health which an NPC lost with each hit in order to allow testers time to experience the behaviours of each version. Testers were asked to play all three games for ten to fifteen minutes each. They were then asked to choose what they believed to be the most believable behaviour and to answer the questionnaire based on this version. Out of the 11 completed questionnaires 8 testers chose the new behaviours as being more realistic, with votes split between the two GP versions.

Version 1
As a whole NPCs were deemed "harder and more challenging" by those who chose this version of the three games. NPCs were found to move to get a better line of site and would fire from distance rather than run to the player before firing. It was also noticed that NPCs collected health packs when wounded and that they seemed to shoot from cover to avoid being shot. The function *if_visible* which was implemented in the GP was originally inserted for use in hunting and determining the effectiveness of a shot. However, the GP extended the functionality of this function to allow for shooting from cover. With regard to varied behaviours, all who chose this version agreed that this variance occurred. Some examples include retreating rather than hunting or shooting from cover rather than attempting an all out attack strategy.
The ability of the NPCs to react to their environment was also deemed a success be testers. NPCs were said to move around objects to get a clear line of sight and they would collect health packs when they became visible.

Version 2
Those who found this version to display the best behaviours generally chose it due to the variation in behaviours. Actions were found to vary between shooting, finding cover, evading to a safe distance and collecting healthpacks. With respect to realistic behaviours, all agreed that this goal was achieved in version 2. One tester noticed the unwillingness of NPCs to approach the player character too closely and chose to shoot from cover instead. The collection of healthpacks was also remarked on as a realistic behaviour, although it was noted that on occasion when NPCs were beside healthpacks and had a low health they did not collect the item. While this is a valid observation it must also be noted that NPCs reasoning may have been that turning to collect a healthpack could result in being shot. NPCs were also seen to adjust their distance to the player character to improve accuracy. As with version 1, the ability of NPCs to act was not questioned in this version. The ability of NPCs to react to their environment was seen by all as a success. That behaviour which stood out the most was NPCs reaction to obstacles in the environment. Most testers commented on the strategy employed by NPCs to take cover behind a wall and shoot from there to avoid being hit themselves. Some testers noticed an extreme of this behaviour, where NPCs had a tendency to stay hidden behind the wall until the player character came hunting for them and so, seemed to have the upper hand. Reactions to being shot were also noticed where the NPC turned and ran away. Novelty was noticed in many forms during game play. One tester remarked on the retreating of NPCs when their health was reduced. Due to this variance in behaviours one tester found predicting the next move by the NPC quite difficult and found that repeated playing of the game yielded different outcomes due to this variance in behaviour, which, by todays gaming standards is quite novel. Due to the NPCs willingness to hide, one tester likened their behaviours to online game play.

Version 3
Most of those who chose version 3 did so based on their opinion that the game was more challenging. This challenging effect was achieved due to a continuous attack strategy being employed by the NPC who hunted the player character relentlessly. Aspects of this behaviour such as sudden changes of direction were deemed to be realistic, however, overall the behaviours were seen to be unrealistic with the example given by one tester that weaker NPCs, i.e. those with less health, should be more inclined to hide rather than to resort to all out attack. Opinions on variation in behaviours varied amongst this

group of testers. Some found behaviours to alternate between attacking and hiding while others noticed no change from the gung-ho mentality employed.
Reactions to NPCs environmental awareness were also mixed. One tester observed NPCs hiding behind walls when being attacked, while others saw no change in the NPCs behaviour, from an all out attack plan, no matter what the change in the environment. From a novelty perspective, testers again disagreed, with some stating that behaviours differed from those usually seen in games of this genre while others disagreed due to the opinion that behaviours did not change throughout the course of the game.

Conclusions and Future Work

In this paper, we presented an approach to evolve more realistic, believable behaviours. We evaluated the evolved strategies on a small sample of gamers.
A minimum benchmark of success would be for these newly created behaviours to equal the believability of the original behaviours. Much consideration is put into developing behaviours to be used in modern games. Since each behaviour is scripted, as many scenarios as possible must be catered for prior to release. Therefore to develop behaviours which equal the performance of these behaviours; using a GP which takes seconds to run; is a more than an acceptable outcome.
This work attempts to provide alternatives to behaviour development in modern commercial games a most important next step would be to determine this methods viability as an industry standard. To achieve this, a more recent open source game need be examined or a completely new game written which utilises all the functionality of popular modern games.
The GP would then need to be tested to ascertain its ability to evolve realistic behaviours from a much larger function and terminal set. The length of time taken to evaluate these behaviours would also need to be considered.
Another future modification to this approach would be to introduce more than one NPC to the game environment during GP evaluation. To date the GP software has only been used to evaluate individual behaviours, i.e. NPCs are not made aware of other NPCs. With an increase in environmental variables discussed above such awareness would be made simpler to implement and could provide the means to develop team based strategies rather than promoting a notion of self importance.
A final avenue worth considering would be to introduce a co-evolutionary approach to evolution. Using co-evolution one of the evolved GP behaviours would be used to control the player character and NPCs would play this character during the evaluation. In this way there is an increasing difficulty introduced to evaluations which may lead to the discovery of the best solutions in a faster time.

REFERENCES

M. Cavazza. AI in computer games: Survey and perspectives. *Virtual Reality*, 5(4):223–235, December 2000.

N. Cole and S. J. Louis. Using a genetic algorithm to tune first-person shooter bots. *Congress on Evolutionary Computation*, 2004.

A. Cunningham and C. O'Riordan. Evolution of stable societies and social structures for computer games: An analysis of static versus dynamic environments. *Irish Conference on Artificial Intelligence and Cognitive Science*, 2008.

D. E. Diller and W. Ferguson. Behavior modeling in commercial games. *Behavior Representation in Modeling and Simulation*, 2004.

D. Doherty and C. O'Riordan. A phenotypic analysis of evolved team behaviours. *Genetic and Evolutionary Computation Conference (GECCO)*, July 2007.

John H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

T. S. Hussain and G. Vidaver. Flexible and purposeful npc behaviors using real-time genetic control. *IEEE Congress on Evolutionary Computation*, 2006.

D. Keaveney and C. O'Riordan. Using genetic programming to learn strategies for an abstract real-time strategy game. *Irish Conference on Artificial Intelligence and Cognitive Science*, 2008.

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

D. Livingstone. Turing's test and believable ai in games. *Comput. Entertaintment*, 4(1), 2006.

S. Luke and C. Hohn. Co-evolving soccer softbot team coordination with genetic programming. *Proceedings of the First International Workshop on RoboCup, IJCAI*, 1997.

S Rabin. *AI game programming wisdom.* Hingham, Mass., Charles River Media, 2002.

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

Y. Shichel and E. Ziserman. Gp-robocode: Using genetic programming to evolve robocode players. *8th European Conference on Genetic Programming*, 2005.

GAME AI ANALYSIS

Analysing the Fitness Landscape of an Abstract Real-Time Strategy Game

David Keaveney
Department of Information Technology
National University of Ireland, Galway, Ireland
Email: d.keaveney1@nuigalway.ie

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

KEYWORDS
Real-time strategy, Multiagent system, Genetic programming

ABSTRACT

The aim of this work is to analyse the fitness landscape described by our testbed game for high-level Real-Time Strategy (RTS) games. We wish to identify whether a dominant strategy exists or an evolutionary cycle of best strategies. We briefly discuss the need for better AI in RTS games and how, we believe, techniques used in board-game research can be applied to this domain. We then outline a multiagent system (MAS) based player for this game and our use of genetic programming to allow our player to learn strategies for the game. We perform two seperate runs of co-evolution for this player and then analyse the evolutionary history of both runs to help understand the fitness landscape of our coordination problem and identify important aspect of these successful and robust strategies.

INTRODUCTION

In this paper, we co-evolve strategies for an abstract RTS game and then analyse the co-evolutionary process. Evolutionary computation has been used in a range of domains to generate competitive solutions. In many such domains, we are only interested in the final evolution solutions and the quality of these solutions. In other scenarios, we are only interested in the evolutionary process, for example, whether the evolutionary search brings us to the same location in the search space each time or whether there are epochs of stability in the process. These types of questions and issues often lead to questions regarding our understanding of the underlying fitness landscape. A range of techniques have been developed in this regard to attempt to gain an understanding of the fitness landscape e.g. fitness clouds (V'erel et al., 2003), fitness distance correlation (Vanneschi et al., 2003).

In the vast majority of these domains we are dealing with a fixed landscape, i.e. the measure of the fitness is defined and does not change throughout the evolution. This is often the case in optimisation problems. In some previous applications in computer games, this is also the case; one can define a set of behaviours against which we test our evolved solutions. As the test-set of behaviours does not change we effectively have a fixed environment. This is a perfectly valid approach if we have a test-bed of suitable behaviours against which we can evolve our solutions. In many cases, we don't have access to such a set of behaviours; one approach in this scenario is to adopt a co-evolutionary approach.

In co-evolutionary approaches, the fitness of any individual is dependant on their performance against other individuals, i.e. their ability to compete or outperform other individuals. As the entire population are undergoing evolution, each successive generation will comprise different individuals from the previous generation. Hence, a strategy that is successful at generation t may not necessarily be successful at generation t+1. The landscape is effectively changing.

By analysing the evolutionary trail or path, we can learn interesting concepts regarding the problem and the evolved solutions. For example, for a game with no dominant strategy, we could envisage the existence of cycles of evolution with an ongoing replacement of the current best strategy. However, for games with a definable best dominant strategy, we would expect to see the convergence at a basin of attraction representing the search space where these dominant strategies exist. It is important to note, of course, that justs because we witness convergence, we may not have a dominant strategy; it may be that finding a better one is very difficult.

In this paper, we adopt a co-evolutionary approach to evolving successive populations of strategies for our abstract RTS game. We maintain a record of the best solutions at each generation and then measure these strategies against each other to obtain a measure of fitness removed slightly from the notion of fitness in our co-evolutionary experiment. We identify when the population has converged (or appeared to) for epochs of varying length. We study and discuss the changes in strategy that correspond to these jumps in evolution.

Evolving spatial coordination for our test-bed game represents a considerably difficult coordination problem. We hypothesise, that by considering the evolutionary history (and strategies discovered) in a co-evolutionary setting, we can identify successful robust strategies for this game and identify areas in the search space where these solutions exist. Futhermore, careful analysis of

these robust strategies can help us identify important features of these strategies.
In section we discuss some of the background research involved in our work. We explain evolutionary computation, specifically genetic programming, and the benifits of coevolution. We outline some of the current problems with RTS games and then outline our testbed developed for high-level RTS game research. In section we detail our multiagent system designed automated player for our game and then our use of genetic programming in learning strategies for the game. Section details experiments where we attempt to gain an understanding of the fitness landscape by observing the evolutionary path in coevolving effective strategies. In Section we show the results to our experiments and then go on, in section , to discuss our findings.

BACKGROUND RESEARCH

Evolutionary Computation:

Evolutionary Computation describes techniques inspired by biological evolution and other naturally occurring operations. Genetic Programming (GP) is one such technique that seeks to optimise computer programs based on their fitness (i.e. their ability to perform a given task well). GP is a computationally expensive technique and has not often been used in game development. An important property of GP is that it can search through large search-spaces efficiently. Furthermore, the GP representational language uses a computer program structure which means that neither the size of a solution nor its shape needs to be constrained beforehand, as is the case with other similar techniques such as genetic algorithms. This property of GP allows for a much broader search thereby increasing the potential for better solutions.
Usually solutions within a population receive a fitness by being evaluated against an already known static solution. However, when coevolving, solutions within the same population play against each other. This means that their fitness depends on how well they compete with other solutions within their population. Because of this relative measure of fitness, once fit solutions can become unfit in subsequent generations. "Coevolution has been proposed as a way to evolve a learner and learning enviroment simultaneously such that progress arises naturally with minimal inductive bias." (Ficici and Pollack, 1998)

Problems in Real-Time Strategy Games:

In a real-time strategy (RTS) game, players must gather resources (e.g. gold, wood, ore) in order to build fighting units and buildings and research new technologies, in an effort to destroy all non-allied players buildings and units. Many of the problems in RTS games are similar to those inherent in many multiagent system domains: coordination/cooperation (spatial and temporal), communication, planning and problem solving.
Coordination is a common problem in RTS games. Coordinating the disposition of different types of units within a force for optimal offence/defence can greatly improve the efficiency of an attacking or defending force. Complex plans, such as plans that rely on the success of other plans may provide better solutions for a successful attack or defence, and are rarely seen when playing an automated player. Cooperation and communication between allied players is often very poor, if present at all.

Our Abstract Real-Time Strategy Game - Testbed:

Our testbed game is called Bellus Bellum Gratia (BBG) (Keaveney and O'Riordan, 2008) and was designed for high-level RTS game research. The game can be represented by an undirected graph (like many boardgames). The vertices represent the areas/territories on the board and the edges the borders between areas. Each edge has a maximum capacity allowed across it, and each vertex a maximum occupancy limit. Each player starts the game controlling a vertex known as their 'home-base'. Control of more vertices increase a player's maximum allowed population and if a player's current population is less than their maximum population then that player can receive a new piece from their home-base. Players can move to adjacent vertices that are empty or occupied by friendly units and can attack adjacent vertices that are occupied by enemy pieces. In attacks, the strongest force wins battles, causing the losing force to suffer a penalty (i.e. lose a certain amount of units). Occupying a vertex for a given time, if already controlled by an enemy player will neutralise the vertex (remove the current controller) or if neutral will take control of that vertex.
Although the structure of the map is known, the dynamic information must be observed. Players can see all dynamic information (i.e. occupying units) in areas that are controlled, occupied or adjacent to any occupied area. All players in a team communicate their view of the world to all other members in the team, and so all players in a team have shared beliefs about the state of the world. A player wins the game if all other players have been destroyed or if he controls a certain majority of the map. A player is destroyed when that player has no further pieces on the board and no longer controls his 'home-base'.

MAS AUTOMATED PLAYER USED IN TESTBED

"MABBG" is a multiagent system (MAS) approach to creating a player agent for the game Bellus Bellum Gra-

tia (BBG). All agents in MABBG represent a possible order for the player in BBG. Since all units belonging to the player must be given orders simultaneously and in parallel with all other players' units, the problem of coordination becomes very difficult. We believe that using an iterative plan formation approach can better help coordinate the units of the player. Agents bid for units within their immediate vicinity and the highest bidder, denoted by an order evaluaion function, can commit one unit to itself. After each change in the plan (i.e. unit commited or units released), agents send out new bids. This process continues until all units are commited. At this stage, the least priority agent that is deemed unlike to succeed is removed from the plan and all units that were committed to it are released. If this occurs, it leads back to the iterative bidding stage so that these new units can be used.

Evaluation Function:

All strategies / behaviours in our MAS approach stem from the order evaluation function which determines the value of any agent's bid. In other previous approaches in the domain, the structure and content (i.e. parameters) of the evaluation function are static and known, and the values for these parameters dictate the behaviour of the approach. We attempt to optimise the evaluation function by using genetic programming to learn the structure and parameters important in a successful strategy. This allows for a broader search and increases our chances of finding better solutions.
Furthermore, our approach uses an iterative planning formation approach. After any change to the plan, all orders are re-evaluated using the evaluation function. This is because the function takes certain order agent parameters (e.g. the number of units committed to this order (parameter: COMMITTED) and the number of non-committed units available to this order, located at the order (parameter: NON_COMMITTED) or within a neighbourhood distance of one (parameter: FR_SUPPORT)) and it is possible that the evolved function will give different priorities for the same agent because of a change in the current plan.

Genetic Programming to Learn Strategies:

We use GP to evolve our order agent evaluation function for our MAS player (MABBG). GP takes a set of functions (mathematical operator) and terminals (all parameters that could be used in reasoning something useful about the game) e.g. "EN_STR" which returns the number of enemy units at this order. As stated before, three of our terminal nodes are classified as order agent parameters (COMMITTED, NON_COMMITTED and FR_SUPPORT) as these may change due to a change in the current plan. We believe that if our evaluation function has any hope for the coordination of unit distribution, that the presence of at least one of these three terminals is necessary.

EXPERIMENTS

The purpose of these experiments is to assess the similarity between some of the best strategies coevolved throughout the course of a single run of evolution and to explore whether a cycle of best strategies exist or whether the evolutionary process tends to the same attractors in the search spaces. Two independent runs were performed, each with a different population size and generation length. Tournament selection was used, as was elitism of the best solution found at each generation. All solutions were penalised for length strongly. This high parsimony pressure insures small and more understandable solutions, and help insure the indentification of the more important properties of the game. It also greatly reduces the search space for good solutions, as complex (i.e. long) good solutions are now considered to be poor. Therefore, usually when searching for a good solution it is important not to penalise for length too much but instead find a suitable balance between the complexity of the strategy and it's quality.

Experiment 1: Our first run had a population size of 500 and was 500 generations long.

Experiment 2: Our second run had a population size of 200 and was 1000 generations long.

Tournament Size	6
Number of Games per Fitness Evaluation	50
Max Turns per Game	1000

Table 1: Other GP Settings

Both experiments evolve on the same map/board. This map consists of sixteen vertices, connected in a 4x4 grid pattern. All games played are two player games, with both players' starting positions (position of their homebase) located at opposite corners of the board.
A round robin tournament was performed with a sample set of all the best solutions found (at the end of each generation) for each experiment. This sample set of solutions was biased greatly towards solutions found during the begining generations as we deemed these strategies to have a much higher rate of change from generation to generation than later ones.
Our first run produced five hundred best-of-generation strategies. Twenty eight of these were selected for our first round robin tournament.

'Best-Of' Generations used in Tournament 1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30, 40, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500

Our second run produced one thousand best-of-generation strategies. Thirty three of these were selected for our second round robin tournament.

'Best-Of' Generations used in Tournament 2: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30, 40, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000

RESULTS

Fig. 2 shows a graph of the absolute scores of all strategies that played in our first RR tournament. Each solution plays each other (including itself) 2000 times. Each win gives a score of 1.0 and each draw gives a score of 0.5. This gives a maximum possible score of 56,000.

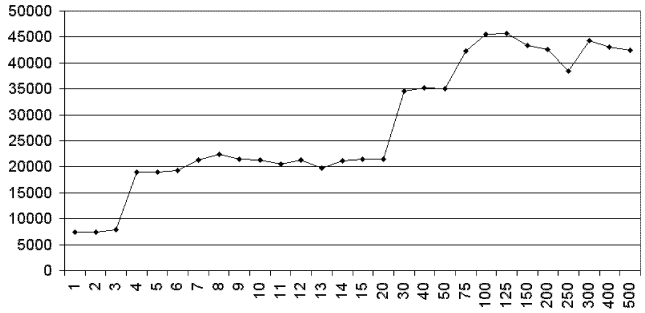


Figure 2: Absolute Scores (Y-axis) for all Strategies (X-axis) in Tournament 1.

The highest scoring strategy in this tournament was the best strategy from generation 125 with a score of 45,723.0.

Best Solution from Generation 125: (DIVIDE FR_SUPPORT (IF FR_CONTROL (IF FR_HOMEBASE (IF ATTACK 50 NON_COMMITTED) 25) (PLUS (PLUS 50 (MULTIPLY (IF ATTACK 25 NON_COMMITTED) (MULTIPLY (MULTIPLY 50 25) FR_SUPPORT))) 50)))

Fig. 3 shows a graph of the absolute scores of all strategies that played in our second RR tournament. Each solution plays each other (including itself) 1000 times, giving a maximum possible score of 33,000.
The highest scoring strategy in this tournament was the best strategy from generation 700 with a score of 27,178.5.

Best Solution from Generation 700:
(MINUS NON_COMMITTED (IF FR_CONTROL EN_CONNECTIVITY (IF FR_CONTROL (MINUS (MINUS UNKWN_CONNECTIVITY FR_SUPPORT) UNKWN_CONNECTIVITY) (DIVIDE 5 (MINUS UNKWN_CONNECTIVITY (MINUS EN_STR FR_SUPPORT))))))

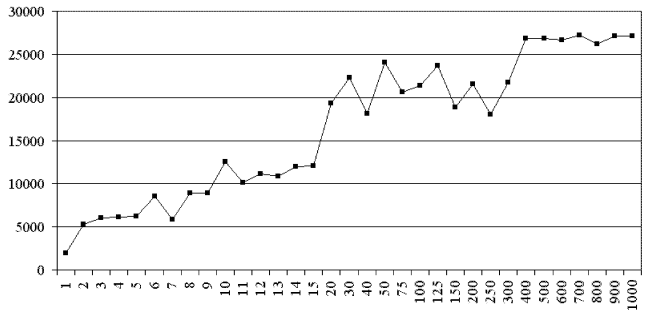


Figure 3: Absolute Scores (Y-axis) for all Strategies (X-axis) in Tournament 2.

The results found in each tournament are then tested for similarity. The scores for each strategy are compared with each other using cosine similarity. The results for both tournaments are shown in Fig. 1. The diagonal shown in both tables show perfect similarity while the areas of the table showing a darker colour represent less similarity between strategies. The large portions of red indicate that many of the evolved strategies are indeed very similar and that often, for generations at a time, there is no marked change and improvement in the evolved strategies.
Using the results of our tournaments, we selected the most robust solutions from our sample sets (those strategies that scored best against other strategies in the sample set).

Robust Strategies in Tournament 1: 50, 100, 125, 150, 200

Robust Strategies in Tournament 2: 50, 125, 400, 500, 600, 700, 800, 900, 1000

Looking at these solution trees, we compile a list of the most common game property terminals found within each robust strategy set.

Terminals Found in Tournament 1: FR_SUPPORT, FR_CONTROL, FR_HOMEBASE, NON_COMMITTED, ATTACK

Terminals Found in Tournament 2: FR_CONTROL, FR_SUPPORT, NON_COMMITTED, COMMITTED, UNKWN_CONNECTIVITY, EN_CONNECTIVITY, FR_CONNECTIVITY, EN_STR

Terminals Common to Tournaments 1 & 2: FR_SUPPORT, FR_CONTROL, NON_COMMITTED

Discussion of Results:

Even given a sample set from 1000 generations, early solutions (i.e. generation 50) still prove to be very robust and in the case of tournament 2, proves the best strategy for playing against generation 150 (which has been

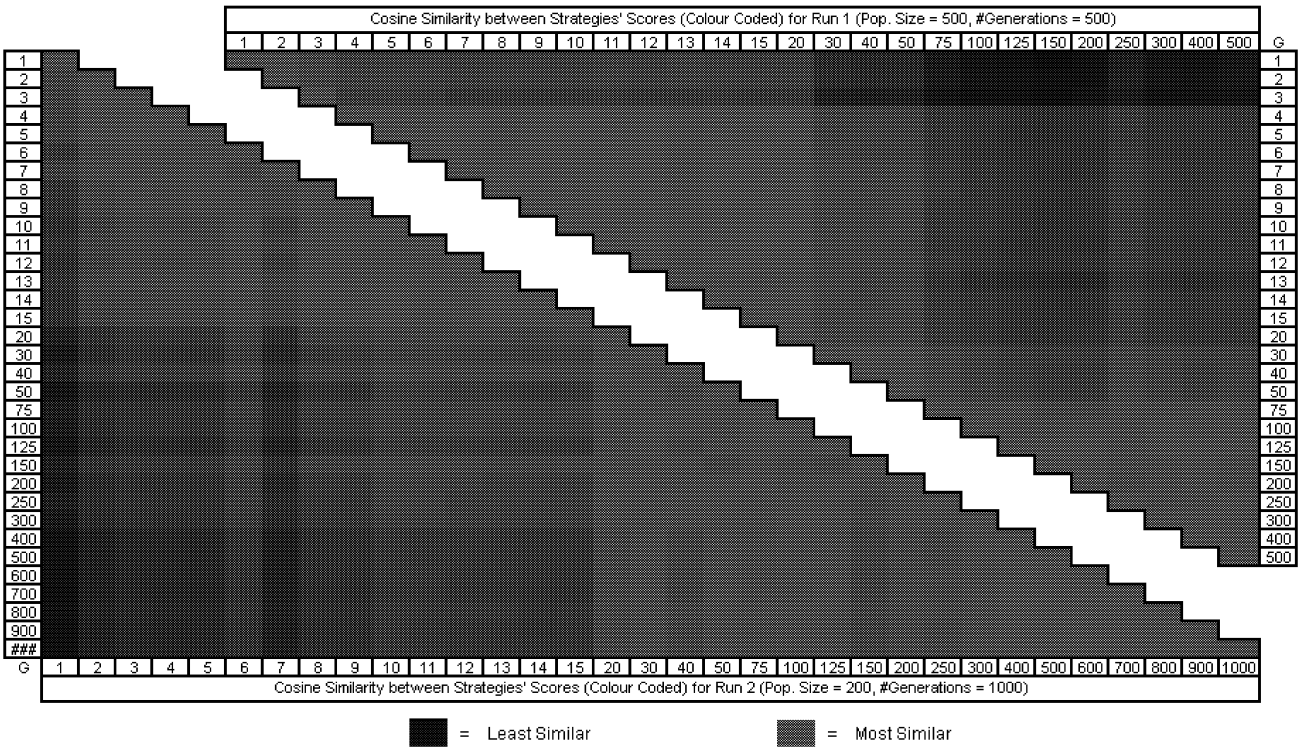


Figure 1: Strategies' Score Similarities for Round-Robin Tournaments 1 & 2

evolved later). This suggests that solutions are moving through different basins of attraction and not towards the same attractors.

Fig. 1 shows that the similarities between strategies are not exactly increasing or decreasing with each generation. These fluctuations are showing different degrees of performance against the full set of strategies.

The common terminals found in all our good solutions, specifically FR_SUPPORT and NON_COMMITTED, suggest that coordination may indeed be evolving in our GP approach. However, further phenotypic analysis and observation of the agent's planning process, iteration by iteration, is required to investigate how much coordination is actually occuring. The FR_CONTROL terminal suggests an importance on whether an area is controlled or not. Since controlling areas increases a player's unit capacity, expanding with this in mind would be essential for any good strategy.

CONCLUSIONS

The results show that there does not appear to be any dominant strategy for this game environment (found within our two evolutionary runs), but more likely a set of different effective robust strategies. Also, by observing the set of best strategies we identified three features important to a good strategy: FR_SUPPORT, NON_COMMITTED and FR_CONTROL.

In future work, we aim to analyse the formation of plans as they occur step by step using our evolved solution, thereby granting us a better understanding of how coordinated the agent's units are. We also hope to add explicit communication between player agents in the same coalitions and to investigate different communication strategies and techniques for plan merging during the iterative planning process.

REFERENCES

Sevan G. Ficici and Jordan B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In *Proceedings of the Sixth International Conference on Artificial Life.*, 1998.

David Keaveney and Colm O'Riordan. Abstract model of a real time strategy game. Technical report, National University of Ireland, Galway, 2008.

L. Vanneschi, M. Tomassini, P. Collard, and M. Clergue. Fitness distance correlation in structural mutation genetic programming. *Genetic Programming, 6th European Conference, EuroGP2003, volume 455-464 of Lecture Notes in Computer Science*, 2003.

S. V'erel, P. Collard, and M. Clergue. Where are bottleneck in nk-fitness landscapes? In *CEC 2003: IEEE International Congress on Evolutionary Computation.*, pages 273–280, 2003.

MASTERGOAL: AN INTERESTING TESTBED FOR AI TECHNIQUES

Alberto Samaniego
Universidad Católica: Nuestra Señora de la Asunción
Asunción, Paraguay
alberto_samaniego@uca.edu.py

Alejandro Alliana
Universidad Católica: Nuestra Señora de la Asunción
Asunción, Paraguay
aalliana@uca.edu.py

Benjamín Barán
Universidad Nacional de Asunción
Asunción, Paraguay
bbaran@cba.com.py

KEYWORDS
Artificial Intelligence, Board Game, Genetic Algorithms, Mastergoal.

ABSTRACT

Games are ideal domains for exploring the capabilities of computational intelligence. Mastergoal© is a strategy board game inspired in soccer with a high branching factor that makes it non-trivial to play. The way tokens are moved inside the board makes it different from traditional board games. This article introduces Mastergoal as a novel game that could be used to explore the effectiveness of emerging artificial intelligence techniques. Some previous work done on Mastergoal is described and future research lines are proposed.

INTRODUCTION

Board games have been of interest for humans since the early days in history. Around 3100 BC, Senet was already a popular game in the ancient Egypt. Go has been being played in China for around 4000 years. Even Chess and Shogi which are more than 1000 years old, derive from more antique games like Chaturanga which remounts to the 5th century BC (Samaniego 2008).

Board games analysis is known as a valid analogy of real life problems, which are often difficult to solve. Most board games have the property of being simplified domains of real life problems, allowing one to focus only in key aspects of the problem such as reasoning, keeping aside issues like physical perceptions or natural language processing. Nevertheless, best humans still surpass most computers in several of these games.

In the field of Artificial Intelligence (AI), games have played an important role since they have been used as testbeds for investigating emerging techniques. The use of games in AI goes back to its earliest days. Already in 1950's, the first Chess programs were written by Claude Shannon and Alan Turing (Shannon 1950). Since then, there has been significant progress in computer game playing strength (Schaeffer 2001). So far, computers already perform better than best humans in simple games like Checkers (Schaeffer et. al 1992) and Othello (Buro 2003). Computers also play at world master's level in Chess (Newborn 1997), and Backgammon (Tesauro 1994). The main exception is the game of Go (Muller 2002), in which computers do not surpass an amateur game-play level, and still remains as a big challenge for AI researchers.

Mastergoal© is a novel board game inspired in soccer, a brand new game in comparison to most popular board games, whose complexity makes it an ideal testbed for AI techniques. This paper introduces Mastergoal as a new valid test domain for AI techniques.

MASTERGOAL

Mastergoal is a two-player strategy board game that reflects a soccer match between two opposing teams. It is similar to chess and checkers in the way that each team has associated with it a number of pieces or tokens that move across squares in the board. The board represents the field and pieces represent the soccer players. There is also a special piece representing the ball.

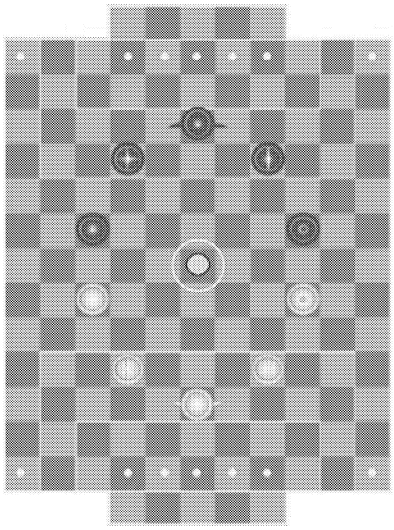


Figure 1: Top view of Mastergoal initial board configuration.

The board consists of 13x11 squares representing the field, and 10 extra squares representing the goals (5 squares on each side). As in soccer, the objective of each team is to score a goal.

Mastergoal could be considered a zero-sum game, in which a player's loss is the other player's gain. It is also a game of perfect and complete information, since the current state of the game as well as the payoffs are known by both players.

During the game, the players take turns to play. A single ply consists of a token move and eventually subsequent ball moves. As a general rule, each token can be moved 1 or 2 squares in horizontal, vertical or diagonal direction. Hence, each token has at most 16 possible squares to move into (see figure 2).

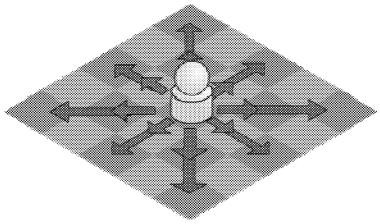


Figure 2: Possible token moves in Mastergoal.

In a similar way, the ball can be moved 1 to 4 squares in the same directions. This implies that the ball can be moved at most in 31 different ways (the ball cannot be left in the square occupied by the token that moves it, see figure 3). The ball can only be moved when there is a majority of tokens of the playing team surrounding it. When this configuration is present on the board, and only in this situation, the ball must be moved by the playing team. The ball can be moved up to 4 times in a single ply, and after the ply is done no team should have a majority of tokens surrounding it. The ball and tokens can only move to empty squares.

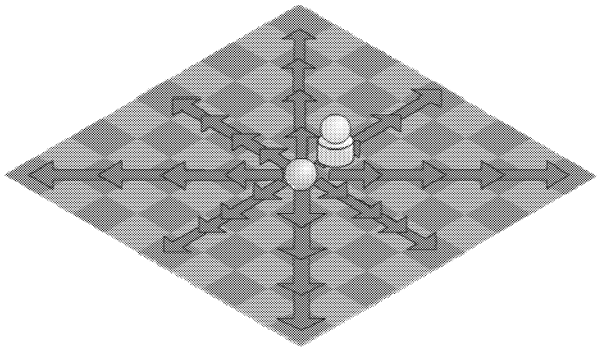


Figure 3: Possible ball moves in Mastergoal.

Obtaining a best playing strategy for Mastergoal is a non-trivial task. Its average branching factor is in most levels superior to the average branching factor of Chess. In fact, at each turn, each player has often more than 100 possible plays, reaching sometimes a branching factor of more than 1000 possible plays in just one turn.

What makes Mastergoal's branching factor so high is the number of passes that can be made in a single ply. A "pass" consists of moving the ball to another square around which the team gets majority (i.e. there are more tokens from the playing team surrounding the ball). As mentioned before, the ball can be moved up to 4 times in one ply, this means that up to 3 passes can be made in a single ply since the final ball move must leave it in a free square. Figure 4 shows an example of a complete ply in Mastergoal for White player. First, White moves one of her tokens one step in lower-left diagonal direction. Since White gets then majority around the ball, she is forced to move it. White makes 3 passes and on the 4th ball move she puts it inside the opponent's goal.

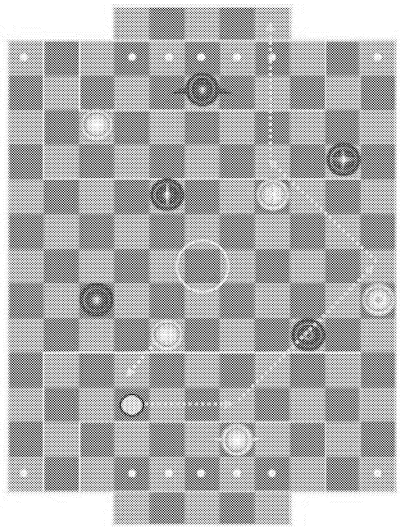


Figure 4: Example of a complete ply that ends up in a goal.

Mastergoal can be played at different "levels" or variations in increasing complexity, each having an increasing number of tokens and slightly different rules associated with them (Samaniego 2008). The most important variations are levels 1 to 3.

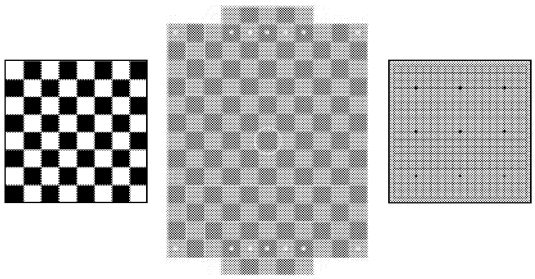
Levels 1 and 2 are simplified forms of the game. Level 1 is played with only 1 token for each team, reducing considerably the branching factor, and hence the complexity of the game. Level 2 consists of 2 tokens for each team, with slightly more complex rules and a higher branching factor than level 1. Levels 1 and 2 are usually played by beginners to get started with the game and its rules.
Finally, level 3 is the most complex modality. This form of the game covers all the rules and has the highest branching factor among all variations. It is the most commonly played modality among strong Mastergoal players. Thus, when we refer to Mastergoal -unless explicitly specifying another form of the game- we assume we are referring to Mastergoal in level 3. In this mode each team has 5 tokens associated with it (see figure 1), being one of them a special token that represents the goalkeeper.

Another interesting aspect of the game Mastergoal is that, unlike games like Chess or Othello, the number of possible moves does not decrease as the game goes by since no pieces are added nor subtracted from the board. Mastergoal's average branching factor remains mostly constant -on

average- through the whole game. Thus, it is practically infeasible to create endgame databases for the game considering the current state of the art of technology.

Mastergoal's branching factor is situated between Chess's and Go's (see figure 5). This property makes it an ideal test domain for many AI techniques. However, despite its high branching factor, the number of legal positions in the game is far less than Chess. This means that nodes, i.e. game states, are frequently repeated in the game tree. Considering the game tree as a graph instead, one could imagine Chess as a graph with a huge number of nodes and fewer edges, in contrast with Mastergoal that could be seen as a graph with fewer nodes and far many more edges.

Chess, Mastergoal and Go



CHESS	*MASTERGOAL*	*GO*
	Board Size	
8 x 8	13 x 11	19 x 19
	Average Branching Factor	
35	more than 100	200 - 300
	Number of Possible Game Possitions	
$\approx 10^{40}$	$\approx 10^{23}$	$\approx 10^{170}$
	Explosion of Choices	
Move 1	35	100	200
Move 2	1225	10 000	40 000
Move 3	42 875	1 000 000	8 000 000
Move 4	1 500 625	100 000 000	1 600 000 000

Figure 5: Comparison between Chess, Mastergoal and Go.

RELATED WORK

A recent work used genetic algorithms (GAs) within a coevolutionary approach to tune the weights of predefined features of the evaluation function (Samaniego 2008). Features for the evaluation function were extracted with the help of the creator of the game.

Since the only parameters that changed during evolution were the evaluation function weights (features remained constant) the best representation for individuals was a real number vector with one element for each weight to tune.

Fitness of individuals was based on the outcome of competition against other individuals. Two kinds of populations where considered during evolution. A population of "hosts" -i.e. individuals whose fitness was under consideration- competed against a smaller population of "parasites" -i.e. individuals that were used to test hosts- (Rosin 1997). Coevolutionary methods such as Competitive Fitness Sharing, Shared Sampling and Hall of Fame -proposed in (Rosin 1997)- were also considered. Competitive Fitness Sharing aims to reward individuals that defeat opponents a few others can, even thou the rewarded individual might not defeat as many opponents as the others can. Shared Sampling was used to conform the parasite population at each generation choosing individuals from the previous host population that had maximal competitive fitness sharing within the current parasite population. Hall of Fame method consists in retaining the best individual from every generation for future testing, so hosts are tested against current parasites and a sample of the hall of fame. This method guarantees progress towards the optimum, since best individuals must beat not only current parasites, but also best individuals of all times.

A 1-ply minimax search was used during evolution. Experimental results showed progress in the playing strength of the solutions. Further comparisons also proved that reaching a good depth in search is as important as having a good evaluation function. Much of the game state information omitted by the evaluation function can be compensated with a deeper search in the game tree. However, other results have shown that weights tuned at 1-ply search depth also perform well at deeper searchers, even thou they are not necessarily optimal at other search depths.

Figure 6 shows the results of a round robin tournament played among best individuals of all times after 300 generations. The graph clearly shows that best individuals from lasts generations won more games than individuals from firsts generations, indicating progress in the playing strength of the solutions.

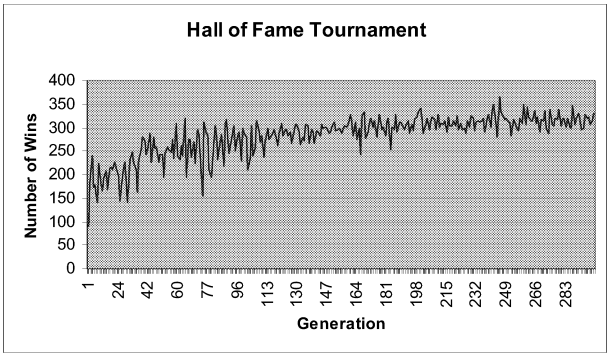


Figure 6: Number of wins of the best individual from each generation in a round robin tournament.

A game agent was built with the obtained evaluation functions to play against the creator of the game, Alberto Bogliaccini. The developed program used negamax search with alpha-beta pruning and search enhancement techniques such as iterative deepening, transposition tables, basic move ordering and some exceptional search extensions. A 3-ply minimax search was enough to make the computer play almost as well as Bogliaccini. However, the computer took around 30 seconds at each turn to make its decision -running on a 1.8 GHz 32 bit processor. This could be a big issue

when playing a game with time constraints considering that a strong human player usually makes its choice in less than 5 seconds.

FUTURE RESEARCH

Much research can still be done using Mastergoal as a test platform. The following are just a few out of many possible lines of research.

Non-Linear Evaluation Functions

Previous work done in Mastergoal has explored the use of linear combination of hand crafted weights as evaluation functions. The use of non linear function approximators such as Multilayer Neural Networks has been tested in other games (Chong 2005) and could be evaluated using Mastergoal. TD-Gammon (Tesauro 1994) achieved the level of play of the top human players using self play to train a Multilayer Perceptron. One of the main reasons for the use of Neural Networks is that they allow generalization: They can evaluate previously unseen board positions based on similar board positions seen during training.

Genetic Programming (GP) can also be used to obtain board evaluation functions. (Ferrer 1996) uses GP to evolve evaluation functions for the game of Senet and Othello, (Anthony 2002) uses GP with a coevolutionary approach to evolve playing strategies for the game known as Acquire. Similar techniques could be applied to Mastergoal. GP can be used to evolve evaluation functions to whatever organizational structure is best suited for the game, without being limited to linear combinations of features. Cellular automata are also a good type of representation for an evaluation function since they are based on pattern-matching primitives. Winning board patterns could be learned by an evaluation function using cellular automata and GP. This method is used in (Rosin 1997) to evolve Go evaluation functions.

Reinforcement Learning

As we previously explained, recent work used GAs to tune the weights of evaluation functions for Mastergoal (Samaniego 2008). Other machine learning techniques such as Reinforcement Learning could be tested using Mastergoal as a testbed. Temporal difference learning was successfully applied to TD-Gammon (Tesauro 1994), but it was not as successful in other game implementations. A theory on the game properties that allow learning using temporal differences is formulated in (Pollack and Blair 1997).

Opening Books

Mastergoal could also be used to experiment with opening books and automatic expansion techniques. (Lincke 2002) presents the drop-out expansion strategy for automatic opening book construction. Drop-out expansion is a best-first expansion strategy which strikes a balance between expanding leaf nodes with good values and expanding leaf nodes at shallow depths reducing the opportunities for the opponent to force the player out of the book. The technique was used to calculate opening books for several games such as Awari, Checkers, Chess and Othello (Lincke 2002).

Forward Pruning Techniques

It is reasonably to think that moving a token towards the ball is a better choice than moving the token away from it in most cases. If the computer can learn which nodes are more promising than others, forward pruning techniques could be used to perform selective and deeper searches in less time than full width searches. Mastergoal could be an ideal game to test forward pruning techniques. (Tsuruoka et. al 2002) presents an approach that selectively expands nodes that are likely to be played, and applies it to the game Shogi. (Lim and Lee 2006) introduces RankCut, a forward pruning technique that makes use of move ordering and can be used in conjunction with other forward pruning methods. RankCut was successfully implemented in an open-source chess playing program along with null-move pruning (Donninger 1993) and futility pruning (Heinz 1998) reducing the game tree size by 10% to 40%. (Buro 1995) presents ProbCut, a forward pruning technique that uses the results of shallow searches to decide which nodes should be expanded to deeper depths. Its improved variant, Multi-ProbCut (Buro 1997), was implemented in Logistello an Othello playing program which defeated the world champion in 1997. ProbCut and Multi-ProbCut were also successfully implemented in games like Chess (Jiang and Buro 2003) and Shogi (Shibahara et. al 2002).

Automatic Feature Construction

Other area that could be explored using Mastergoal is automatic feature construction techniques. Automatic feature extraction is a prominent area in game research. For instance, (Buro 1998) presents GLEM, a framework for the semi-automatic construction of evaluation functions that proposes the idea of constructing evaluation functions by combining conjunctions of boolean features. Other algorithms for discovering features from a formal game description are presented in (Kuhlmann 2006) and (Kaiser 2007).

Graph Search Algorithms

As we explained before, the Mastergoal search tree contains many repeated states, making it a good test for search algorithms that have optimizations that exploit these graph properties. (Plaat et. al 1994) presents a study on how graph optimizations can reduce the search space about 25 %.

Preliminary work on this area shows that MTD-f algorithm using transposition tables reaches higher depths by searching through fewer nodes (Aquino et. al 2005).

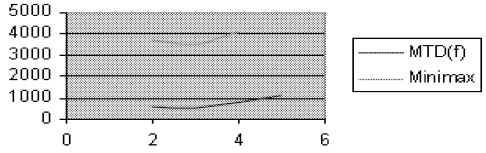


Figure 7: Number of nodes expanded at each depth for the MTD-f and Minimax with Alpha Beta.

SUMMARY

This paper introduced Mastergoal as a novel board game that could be very useful as a testbed for AI techniques. We mentioned some previous work on Mastergoal that has shown encouraging results. Mastergoal properties make it an ideal platform for testing many AI techniques such as neural networks, genetic programming, reinforcement learning, opening books, feature construction and other emerging techniques. The fact that its branching factor is situated between Chess's and Go's makes the game an interesting research topic for forward pruning techniques. Also, the amount of repeated states makes it an interesting test case for graph search algorithms. Eventually, these new techniques could be applied to solve complex real-life situations. There is much research that could be done and much to learn from the study of this novel and challenging game.

REFERENCES

Aquino, H; Milleres, F; Mercado M. 2005. *Implementación y Comparación de Algoritmos Minimax en la Construcción de un Agente que Juega Mastergoal.* JIT-CITA 2005. Asunción, Paraguay.

Buro, M. 1995. *ProbCut: An effective selective extension of the alpha–beta algorithm*. ICCA Journal, Vol. 18, 71–76.

Buro, M. 1997. *Experiments with Multi–ProbCut and a New High–quality Evaluation Function for Othello*. Workshop on Game–tree Search, NECI.

Buro, M. 1998. *From Simple Features to Sophisticated Evaluation Functions.* The First International Conference on Computers and Games. Tsukuba, Japan.

Buro, Michael. 2003. *The Evolution of Strong Othello Programs.* Entertainment Computing - Technology and Applications. Kluwer.

Chong, S.Y.; Tan, M.K.; and White, J.D. 2005. *Observing the Evolution of Neural Networks Learning to Play the Game of Othello.* Evolutionary Computation, IEEE Transactions on Volume 9, Issue 3, 240–251. (Jun.)

Donninger, C. 1993. *Null Move and Deep Search: Selective Search Heuristics for Obtuse Chess Programs*. International Computer Chess Association Journal. Vol 16 No 3, 137–143.

Heinz, E. A. 1998. *Extended Futility Pruning*. International Computer Chess Association Journal. Vol 21 No 2, 75–83.

Hsu, F. 2007. *Cracking Go*. IEEE Spectrum. 44-49. (Oct.)

Jiang, A. and Buro, M. 2003. *First Experimental Results of ProbCut Applied to Chess*. Advances in Computer Games Conference 10, 19–31.

Kaiser, D. M. 2007. *Automatic Feature Extraction for Autonomous General Game Playing Agents*. Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems. Honolulu, Hawaii.

Kuhlmann, G.; Dresner, K.; and Stone, P. 2006. *Automatic Heuristic Construction in a Complete General Game Player*. Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06). Boston, MA.

Lim, Y. J. and Lee, W. S. 2006. *RankCut – A Domain Independent Forward Pruning Method for Games*. Proceedings of Twenty-First National Conference on Artificial Intelligence (AAAI-06).

Lincke, T. 2002. *Exploring the Computational Limits of Large Exhaustive Search Problems*. PhD thesis. Swiss Federal Institute of Technology. Zurich, Swiss.

Muller, M. 2002. *Computer Go*. Artificial Intelligence, Vol. 134, 145–179.

Newborn, M. 1997. *Kasparov versus Deep Blue: Computer Chess Comes of Age*. Springer-Verlag New York, Inc. New York, NY, USA.

Plaat, A; Schaeffer, J. ; Pijls, W. and Bruin, A. 1994. *A New Paradigm for Minimax Search.* Technical report TR 94-18. Technical report TR-85-2. Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada. (Dec.)

Pollack, J. and Blair, A. 1997. *Why did TD-Gammon Work*. Advances in Neural Information Processing Systems, Vol. 9.

Rosin, C. 1997. *Coevolutionary Search Among Adversaries*. PhD thesis, University of California, San Diego.

Samaniego, A. 2008. *Ajuste de la Función de Evaluación del Juego Mastergoal utilizando Algoritmos Evolutivos*. Computer Science Degree Thesis, Universidad Católica "Nuestra Señora de la Asunción". Asunción, Paraguay.

Schaeffer, J.; Culberson, J.; Treloar, N.; Knight, B.; Lu, P.; and Szafron, D. 1992. *A World Championship Caliber Checkers Program*. Artificial Intelligence, Vol. 53, 273-289.

Schaeffer, J. 2001. *A Gamut of Games*. AI Magazine, Vol. 22, 29–46.

Shannon, C. E. 1950. *Programming a computer for playing chess*. Philosophical Magazine, Vol. 41.

Shibahara, K.; Inui, N.; and Kotani, Y. 2002. *Effect of ProbCut in Shogi — by Changing Parameters According to Position Category*. In Proceedings of the 7th Game Programming Workshop, Hakone, Japan.

Tesauro, G. 1994. *TD-Gammon, a Self-teaching Backgammon Program, Achieves Master-level Play*. Neural Computation, Vol. 6, 215-219.

Tsuruoka, Y.; Yokoyama, D.; and Chikayama, T. 2002. *Game-tree Search Algorithm Based on Realization Probability*. ICGA Journal, Vol. 25, 145-152.

AN ANALYSIS OF FITNESS LANDSCAPE IN THE EVOLUTION OF SOCIAL STRUCTURES FOR COMPUTER GAMES

Alan Cunningham and Colm O'Riordan
IT Department, National University of Ireland, Galway
University Road,
Galway,
Ireland
email: alan.cunningham|colm.oriordan@nuigalway.ie

KEYWORDS
Artificial Intelligence, Genetic Programming, Background Characters, Common Pool Problems

ABSTRACT

Using Mutli-Agent Systems (MAS) and Evolutionary Computation (EC), we explore the creation of stable societies and social structures with applications in the computer games domain. Genetic programming techniques are used to automatically generate robust populations of ambient characters. We evolve scripts for a group of Non Player Characters (NPCs) to solve the collective problem of subsistence by way of a common pool. By considering the group of NPCs in this way, we create implicit relationships and varying behaviours for the members. The defined problem requires coordination among the different NPCs and their roles, while containing features of many social dilemmas. We investigate the effect of varying the problem so that it ranges from a relatively easy evolutionary problem to a difficult one.

INTRODUCTION

The motivation for this work stems from the computer games domain and the need for more suitable Artificial Intelligence (AI) approaches. If the NPCs appear to be acting non sensibly for their environment, the user's belief in the game is adversely affected, which is why many users prefer human-controlled opponents (Schaeffer, 2001).

In a typical computer game, such as a Role Playing Game, there may exist a town or a large collection of NPCs. Usually each NPC not central to the story will have a static role assigned to them and this role may have no impact on the game other than servicing the player on request or being part of the game's aesthetic. However, upon revisiting this place multiple times during the course of a game, the characters will be performing the same actions as they did in the first encounter. We argue that even though these characters are not central to the plot, they contribute to the believability of the game. Yannakakis and Hallam (2007) show the behaviours of NPCs are shown to have a direct impact on the interest of the user and entertainment value in predator-prey games, which we argue extends to all games.

We wish to tackle the problem of creating complex, active and robust ambient characters. We investigate the application of Genetic Programming (GP) to solve the problem of creating these populations automatically. GP has been shown to be successful at providing teams with strategies in several domains such as shooter games (Doherty and ORiordan, 2006a,b).

We wish to abstract a fantasy world to base context-relevant actions for each character. We separate a large population into smaller groups and pose each group a dilemma that each member must act in, in order to survive. We hypothesise that large interesting populations can be created by combining a number of these smaller groups, each one striving to solve their problem and maintain their existence. This would create an automatic relationship and dependency between characters and any changes can be reflected through the whole population.

We hypothesise that abstract economic or social problems mapped into the game would provide such opportunities for diverse and important character roles and will provide cooperation and coordination problems for building a society upon. We argue that using GP techniques in this way we bypass the need for extensive training opponents (Doherty and ORiordan, 2006a,b, Spronck et al., 2006) or user controlled opponents (Blair et al., 1999) to evolve our solutions against as is the typical approach when applying GP techniques for games.

In order to specify suitable environments for these populations to be evolved in, and inspired by work detailed by Curran et al. (2007), we look to social dilemma common pool problems where the optimal behaviour for the group is not necessarily the optimal for each individual.

The game we use, combining aspects from both social dilemmas and coordination games, can be described as follows. Agents must chose actions throughout the game (idle, transport, harvest, eat etc.). Each action has an associated cost or benefit. Agents must try to maximise

the benefit while minimising the cost. In order for the group to survive, food must be created. In order to create food a number of steps and associated roles must be fulfilled. This is effectively a coordination game — the agents must coordinate themselves such that all the necessary roles are fulfilled. The game also combines many aspects of well known N-player social dilemmas. The agents are part of a society or group. Some necessary roles are more costly than others. The dominant rational strategy is to only choose activities that convey a high benefit and avoid those with a cost; however if all agents adopt this approach a pareto supoptimal solution is arrived.
We can vary the difficulty of this game in a number of ways. We can vary the costs and benefit such that we can render it an easy game (low costs, high benefits) or a very difficult one (higher costs, lower benefits). This can effectively be achieved by tuning one parameter. In this paper, we are interested in exploring varying the difficulty of the game in terms of varying the costs and benefits. Cunningham and O'Riordan (2008) explored varying the complexity of the game in terms of coordination .
The following paper is divided in several sections. The next section is a discussion of the related research in the games domain, which introduces briefly the concepts seen in the rest of the report. Section three outlines the game and the agent's roles. Section four presents and discusses the results achieved from the experiments. Section five is the conclusion of our findings and discusses the uses for the techniques in the project as well as possible future work.

RELATED RESEARCH

Traditional appraoches implemented in current commercial computer games are rules based systems and finite state machines (Tozour, 2002). However as there are usually only a limited number of states, scripted AI can greatly reduce a game's replay value. Using these methods it is also expensive to fully script each and every character in a large game such that it is performing interesting actions. This is the main reason we are paying special attention to background characters which are usually neglected.
A number of extensions have been made to these approaches, such as, Cutumisu et al. (2006), have shown a model for scripting NPCs more easily. They develop a framework that allows non-programmers to write scenes with proactive and reactive characters. Spronck et al. (2006) have shown the successful application of online learning applied to scripts. This approach has been extended to include dynamic difficulty scaling within their on-line learning (Spronck et al., 2004). Gruenwoldt et al. (2005), Darken and Kelly (2008) add social networks to NPCs to provide more realistic responses from interactions between the player and NPCs.
These approaches however consider each individual as a single insulated entity. We consider the NPCs at group level and use GP techniques to script a whole group collectively. The approaches could be used to enhance the individuals we present here, however with the increase in complexity of each NPC we would expect an increase in the evolution requirements of the group. This is an avenue of research that will be explored as our techniques are tested in a commercial game test bed.

MODEL

Game Definition

The model consists of a population of agents that must adopt various roles in the process of resource collection to survive in the environment. The game is broken up into time periods called ticks and each action in the game takes a tick to perform.
Each agent has a starting value of health; each action that an agent performs has some cost to their health. Agents can replenish health by eating food. Each agent has a set of conditions to reason about the environment and a small set of high level actions.
We will focus on one scenario that has been defined for the agents acting in this environment. The problem we pose to the agents is as follows. There are two resources that need to be combined to create food for the agents. Each resource must be harvested and then refined before they can be used for the production of food.
Harvesters gather the resource at their workplace which they can stock pile to a predefined limit. The resource is then brought to the keep for refinement. A Transporter can then bring the refined resource to the factory to be used for food production. In order to solve this specific problem enough agents must harvest the required amount of each resource in order to create enough food to sustain all the agents involved. There must also be enough agents transporting goods between the keeps and the factory such that the food can be created in a timely fashion.
Thus we have a coordination problem in the role adoption for the group. There must be a sufficient amount of each role to satisfy the demand the group has for the food. We also have a coordination problem as all agents *types* are needed for the creation of the food. Agents must cooperate in order for the group to survive however, not all agents may be needed to contribute. This could lead to instances of individuals exploiting the generosity of the group.
GP is used to create a single tree for all of the NPCs in the group. The first stage is to select each agent's role, each of which will be explained below. Then the the GP process uses the functions IF and AND and terminals from the actions described in B.3 and various environmental parameters that represent the agents' knowledge, to create a tree of actions for each person.

Agent Roles and GP NodeSets

Harvester

Each Harvester is specific to one resource and does not change role during its lifetime. The available actions are:

Harvest: The agent must be at their workplace, or must travel there incurring the cost of moving, in order to harvest. The agent then takes one tick to produce 1 unit of resource which is stored in the workplace.

Transport: The agent must be at their workplace, or travel there incurring the cost of moving, in order to Transport. Once the agent is at the workplace they take 1 unit (the harvester's carrying capacity) from the workplace stock if available, and brings it to the keep. This process takes one tick.

Eat: The agent must be at the factory in order to eat. Once at the factory the agent can consume 1 unit of food if it's available.

Idle: The agent preforms no action for the tick. This is the default action for an agent if the conditions are not met for the other actions, for example if after arriving at the factory there is no food to eat the agent idles.

Transporter

The Transporter is responsible for bringing the refined resources from the keeps to the factory. In this scenario we will assume that two resources are needed ro produce food.

TransportRandomResource: The agent travels to one of the keeps randomly. If a refined resource is there they can transport 1 unit to the factory. Once the agent is carrying the resource they incur the cost of transport.

TransportResource1 and TransportResource2: The same mechanics as "TransportRandomResource" but corresponds to a specific resource.

Eat: The agent must be at the factory in order to eat. Once at the factory the agent can consume 1 unit of food if it's available.

Idle: The agent preforms no action for the tick. This is the default action for an agent if the conditions are not met for the other actions, for example if after arriving at the factory there is no food to eat the agent idles.

Action	Cost
Harvest	-1
Transport	-1
Idle	-0.5
Move	-0.5

Table 1: Action Costs for Agents

GP Actions and Costs

Table 1 shows the associated costs for an agent to perform an action. However each action is dependant on the location of the agent for its true cost as the agent may have to move in order to complete the action.
Equation 1 shows the minimum cost of food production in a two resource game. However, this cost increases after the first units are created when incorporating the return journey to the workplaces. Also of note is that 3 agents are needed for the production of 2 units of food which is now available for the whole group to consume.

$$2Food = \underbrace{(H + T + M) * 2}_{2Harvesters} + \underbrace{(2 * T) + M}_{Transporter} \quad (1)$$

Table 2 shows the GP node sets that are used to create the trees of agents. For each agent in the population a subtree is created with the root node of that subtree from the *Class Set*. This is a function that allows us to load all the required agents into the game by executing the tree at this level. From there each subtree then defines the decisions and actions for each agent which is defined by their own node set. The function If for each agent is combined with nodes from the *Decision Set* and the *Environment Set* and a set of constants(not shown) for the agents to make judgements about their world. The function And allows the agents to add multiple actions onto their action queues.

EXPERIMENTS

The Genetic Program Fitness Function

The evolutionary process is used to measure the effect of varying one parameter on a group of agents. Each group of agents is one member of the GP population, a tree structure containing subtrees for each agent. The fitness of a member is the result of how the group perform. The fitness of each set of agents is computed once a game of 1000 ticks has completed or until all agents have 0 health points remaining. The score for each game is as follows:

> *(Sum of the Worker's Health *1.5)+ Sum of the Worker's Ticks Alive - (depth of tree/4) - (tree length/4)*

We bias the worker's health remaining at the end of the game so that even if the solution found cannot sustain all the agents intended we prefer to have some agents alive even at the expense of others. The bias also helps maintain longer successful solutions where all agents are alive over shorter solutions where only subset of the agents survive.
The number of ticks the agents are alive for represent a measure of progression throughout the evolution process. In the early stages it is highly unlikely that any

Class Set	Farmer	Lumberjack	Transporter	Decision	Environment
Farmer f(1)	Eat	Eat	Eat	Greater f(2)	Health
Lumberjack f(1)	Transport	Transport	Transport Resource 1	Less f(2)	Workplace Stock
Transporter f(1)	Harvest	Chopwood	Transport Resource 2	Equal f(2)	Distance from Workplace
	Idle	Idle	Transport Random Resource		Distance from Factory
	If f(3)	If f(3)	Idle		Factory Stock
	And f(2)	And f(2)	If f(3)		Keep Stock
			And f(2)		

Table 2: Node Sets for GP

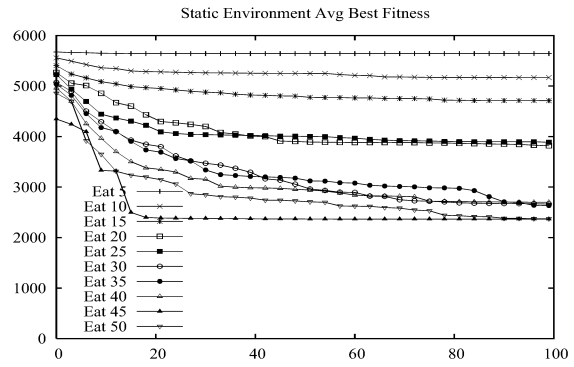


Figure 1: Fitness landscape in a static environment

agent will survive the specified game length. If there are not enough agents contributing for the survival of the group the best the agents can hope for is exploitation. That is agents tending towards idling, an action that incurs the lowest cost, and thus survival for longer.

We take a measure of length and depth of the solutions in order to encourage the finding of the simplest possible solutions to the problems.

Static Environment

This is the initial problem in the environment in which we want to show that the agents can sustain themselves in an unchanging landscape. The experimental setup for this game consists of 6 agents whose role will be chosen by the evolution process and the need for two resources to be combined in equal measure to create food. For one unit of each resource used, 2 units of food is created.

We are varying the payoff or reward for eating food in this simple environment to ascertain the difficulty of the problem. We evolve 10 solutions for each value of reward for *EAT*. We plot the averages of the best fitness over the 100 generations of evolution. We wish to determine a suitable value for reward that will firstly allow the agents to survive and secondly can maintain the a diversity of actions and agents.

While the costs for each individual action may be small in comparison to the reward for *EAT* the action cost in reality also comprises a spatial factor. Each action's cost is its primary cost plus the distance the agent has to move to perform the action times the cost of moving. During the evolution process, as the trees of actions are created, there can be a lot of inefficient moves which cause the agents to perform actions that are detrimental to their health. The actions may involve a lot of moving as the process tries to figure out the best set of actions. If the value of replenishment is too low the agents will find it difficult to evolve to good solution and will tend to choose non-cooperative actions, that is idling until they have no health left.

We use six agents in these experiments to allow for diverse behaviours to emerge within the small group. The minimum number of agents required for this experiment is three, one for each resource and one transporter, but when using this amount of agents in the evolutionary process it is hard to evolve to a sustainable solution. The problem becomes hard as the agents tend towards defection (i.e. idling) which leads to a poor fitness.

In Fig. 1 we plot summary of the average best fitness of 10 GP evolutions for this static environment. As we can see from the graph given an increasing reward value for *Eat* the average fitness increases (closer to 0 is better in this case). However, increasing the reward value does not guarantee a good "usable" solution as is outlined in Fig. 2. We define usable here to mean a strategy with a fitness score of less than 1000. This score implies that most of the agents in the group survived all 1000 ticks of the game evaluation. For the lower values of *Eat* we can see that no solutions match our criteria.

These results indicate the difficulty of our specified problem. There are many reasons for this difficulty. There are 3 agents needed to perform actions to produce 2 units of food for which all 6 members are vying. Without any way to reason about the needs of the other agents it is difficult to share out the food.

Many of the runs that fail to achieve a good fitness score often converge upon a defective state quickly. The GP process has difficulty getting out of this local optimum using the parameters we have specified for this experiment of 95% crossover and 5% mutation. An increase in the mutation rate may prevent the population from stagnating in this way.

We have also investigated this problem in a changing environment where agents lose health at various stages of the game. We found that the average best fitness and the ability to find usable solutions shared a similar

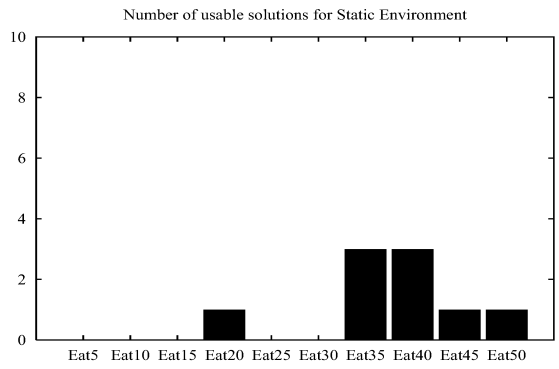


Figure 2: Number of usable solutions for Static Environment

trend to the static environment — as the value of *EAT* increased so did average fitness but a usable result was not guaranteed.

CONCLUSION AND FUTURE WORK

In the broader context of computer games we have shown we can evolve groups of ambient characters for a range of different landscapes resulting in different combinations of NPC behaviours. We have presented an analysis of the difficult problem of evolving cooperating groups of agents. We use problems inspired by game theory and social dilemma common pool problems to provide the environments for the agents to act in. Using genetic programming techniques we have evolved many populations of agents, while varying the reward parameter of the actions, to measure its effect on the group. We have shown that this is a suitably difficult cooperation and coordination problem upon which to build interesting populations of agents.
The current planned future work with this research branches out in many directions. An exploration of the effect of the mutation rate in the GP process and the success in finding good solutions is currently underway. We also wish to increase the complexity of the current problems we are solving by increasing the number of resources and agents involved. We also wish to create an interdependency between the sub-problems in the combination process. Each sub problem would have a solution evolved as in the current version, but the success of each part would rely on some or all of the other sub-problems being fulfilled either partially or wholly.

REFERENCES

A. D. Blair, E. Sklar, and P. Funes. Co-evolution, determinism and robustness. *In Proceedings of the Second Asia-Pacific Conference on Simulated Evolution and Learning, Springer-Verlag*, pages 389–396, 1999.

Alan Cunningham and Colm O'Riordan. Evolution of stable societies and social structures for computer games: An analysis of dynamic environments. *19th Irish Artificial Intelligence and Cognitive Science Conference (AICS)*, 2008.

Dara Curran, Colm O'Riordan, and Humphrey Sorensen. Evolutionary simulations of behaviours in a common-pool resource problem. *18th Irish Artificial Intelligence and Cognitive Science Conference (AICS)*, pages 51–60, 2007.

Maria Cutumisu, Duane Szafron, Jonathan Schaeffer, Matthew McNaughton, Thomas Roy, Curtis Onuczko, and Mike Carbonaro. Generating ambient behaviors in computer role-playing games. *IEEE Intelligent Systems*, 21(5):19–27, October 2006.

Christian J. Darken and John D. Kelly. *Individualized NPC Attitudes with Social Networks.* AI Game Programming Wisdom 4. Charles River Media, 2008.

D. Doherty and C. ORiordan. Evolving agent-based team tactics for combative computer games. *17th Irish Artificial Intelligence and Cognitive Science Conference (AICS)*, 2006a.

D. Doherty and C. ORiordan. Evolving tactical behaviours for teams of agents in single player action games. *In CGAMES 2006 9th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*, 2006b.

L. Gruenwoldt, S. Danton, and M. Katchabaw. Creating reactive non player character artificial intelligence in modern video games. *In Proceedings of the 2005 GameOn North America Conference*, 2005.

J. Schaeffer. A gamut of games. *Artificial Intelligence Magazine*, 22(3):29–46, 2001.

P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma. Difficulty scaling of game ai. *In: Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON)*, pages 33–37, 2004.

Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. Adaptive game ai with dynamic scripting. *Machine Learning*, 63(3):217–248, June 2006.

Paul Tozour. *The Perils of AI Scripting.* AI Game Programming Wisdom. Charles River Media, 2002.

Georgios N. Yannakakis and John Hallam. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence*, 22(10):933–971, 2007.

AI PROGRAMMING

RAPID ADAPTATION OF VIDEO GAME AI

Sander Bakkes, Pieter Spronck and Jaap van den Herik
Tilburg University
Tilburg Centre for Creative Computing (TiCC)
P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands
e-mail: {s.bakkes, p.spronck, h.j.vdnherik}@uvt.nl

ABSTRACT

Current approaches to adaptive game AI require either a high quality of utilised domain knowledge, or a large number of adaptation trials. These requirements hamper the goal of rapidly adapting game AI to changing circumstances. In an alternative, novel approach, domain knowledge is gathered automatically by the game AI, and is immediately (i.e., without trials and without resource-intensive learning) utilised to evoke effective behaviour. In this paper we discuss this approach, called 'rapidly adaptive game AI'. We perform experiments that apply the approach in an actual video game. From our results we may conclude that rapidly adaptive game AI provides a strong basis for effectively adapting game AI in actual video games.

INTRODUCTION

Over the last decades, modern video games have become increasingly realistic with regard to visual and auditory presentation. However, game AI has not reached a high degree of realism yet. Game AI is typically based on non-adaptive techniques [22]. A major disadvantage of non-adaptive game AI is that once a weakness is discovered, nothing stops the human player from exploiting the discovery. The disadvantage can be resolved by endowing game AI with adaptive behaviour, i.e., the ability to learn from mistakes. Adaptive game AI can be established by using machine-learning techniques, such as artificial neural networks or evolutionary algorithms. In practice, adaptive game AI in video games is seldom implemented because machine-learning techniques typically require numerous trials to learn effective behaviour. To allow rapid adaptation in games, we describe a means of rapid adaptation that is inspired by the human capability to solve problems by generalising over previous observations in a restricted problem domain.

The outline of this paper is as follows. First, we discuss the aspect of entertainment in relation to game AI. Then, we discuss our approach to establish rapidly adaptive game AI. Subsequently, we describe an implementation of rapidly adaptive game AI. Next, we describe the experiments that test rapidly adaptive game AI in an actual video game, followed by a discussion of the experimental results. Finally, we provide conclusions and describe future work.

ENTERTAINMENT AND GAME AI

The purpose of a typical video game is to provide entertainment [22, 15]. Of course, the criteria of what makes a game entertaining may depend on who is playing the game. Literature suggests the concept of immersion as a general measure of entertainment [14, 21]. Immersion concerns evoking an immersed feeling with a video game, thereby retaining a player's interest in the game. As such, an entertaining game should at the very least not repel the feeling of immersion from the player [12]. Aesthetical elements of a video game, such as graphics, narrative, and rewards, are instrumental in establishing an immersive game-environment. Once established, the game environment needs to uphold some form of *consistency* for the player to remain immersed within it [12].

The task for game AI is to control game characters in such a way that behaviour exhibited by the characters is consistent within the game environment. In a realistic game environment, realistic character behaviour is expected. As a result, game AI that is solely focused on exhibiting the most *challenging* behaviour is not necessarily regarded as realistic. For instance, in a typical first-person shooter (FPS) game it is not realistic if characters controlled by game AI aim with an accuracy of one hundred per cent. Game AI for shooter games, in practice, is designed to make intentional mistakes, such as warning the player of an opponent character's whereabouts by intentionally missing the first shot [13].

Consistency of computer-controlled characters within a game environment is often established with tricks and cheats. For instance, in the game HALF-LIFE, tricks were used to establish the illusion of collaborative teamwork [12], causing human players to assume intelligence where none existed [13]. While it is true that tricks and cheats may be required to uphold consistency of the game environment, they often are implemented only to compensate for the lack of sophistication in game AI [6]. In practice, game AI in most complex games still is not consistent within the game environment, and exhibits what has been called 'artificial stupidity' [13]

rather than artificial intelligence. To increase game consistency, and thus the entertainment value of a video game, in our research we foremost strive to create an optimally playing game AI, as suggested by Buro and Furtak [6]. In complex video games, such as real-time strategy (RTS) games, near-optimal game AI is seen as the only way to obtain consistency of the game environment [12]. Once near-optimal game AI is established, difficulty-scaling techniques can be applied to downgrade the playing-strength of game AI, to ensure that a suitable challenge is created for the player [19].

APPROACH

For game AI to be consistent with the game environment in which it is situated, it needs the ability to adapt adequately to changing circumstances. Game AI with this ability is called 'adaptive game AI'. Typically, adaptive game AI is implemented for performing adaptation of the game AI in an online and computer-controlled fashion. Improved behaviour is established by continuously making (small) adaptations to the game AI. To adapt to circumstances in the current game, the adaptation process typically is based only on observations of *current* gameplay. This approach to adaptive game AI may be used to improve significantly the quality of game AI by endowing it with the capability of adapting its behaviour while the game is in progress. For instance, the approach has been successfully applied to simple video games [7, 11], and to complex video games [19]. However, this appproach to adaptive game AI requires either (1) a high quality of the utilised domain knowledge, or (2) a large number of adaptation trials. These two requirements hamper the goal of achieving *rapidly* adaptive game AI.

To achieve rapidly adaptive game AI, we propose an alternative, novel approach to adaptive game AI that comes without the hampering requirements of typical adaptive game AI. The approach is coined 'rapidly adaptive game AI'. We define rapidly adaptive game AI as an approach to game AI where domain knowledge is gathered automatically by the game AI, and is immediately (i.e., without trials and without resource-intensive learning) utilised to evoke effective behaviour. The approach, illustrated in Figure 1, implements a direct feedback loop for control of characters operating in the game environment. The behaviour of a game character is determined by the game AI. Each game character feeds the game AI with data on its current situation, and with the observed results of its actions. The game AI adapts by processing the observed results, and generates actions in response to the character's current situation. An adaptation mechanism is incorporated to determine how to best adapt the game AI. For instance, reinforcement learning may be applied to assign rewards and penalties to certain behaviour exhibited by the game AI.

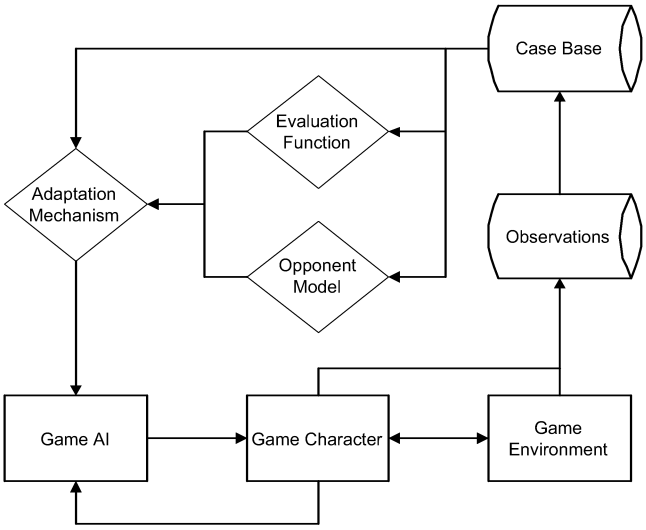


Figure 1: Rapidly adaptive game AI (see text for details).

For rapid adaption, the feedback loop is extended by (1) explicitly processing observations from the game AI, and (2) allowing the use of game-environment attributes which are not directly observed by the game character (e.g., observations of team-mates). Inspired by the case-based reasoning paradigm, the approach collects character observations and game environment observations, and extracts from those a case base. The case base contains all observations relevant for the adaptive game AI, without redundancies, time-stamped, and structured in a standard format for rapid access. To rapidly adapt to circumstances in the current game, the adaptation process is based on domain knowledge drawn from observations of a *multitude* of games. The domain knowledge gathered in a case base is typically used to extract models of game behaviour, but can also directly be utilised to adapt the AI to game circumstances. In our proposal of rapidly adaptive game AI, the case base is used to extract an evaluation function and opponent models. Subsequently, the evaluation function and opponent models are incorporated in an adaptation mechanism that directly utilises the gathered cases.

The approach to rapidly adaptive AI is inspired by the human capability to reason reliably on a preferred course of action with only a few observations on the problem domain. Following from the complexity of modern video games, game observations should, for effective and rapid use, (1) be represented in such a way that stored cases can be reused for previously unconsidered situations, and (2) be compactly stored in terms of the amount of retrievable cases [1]. As far as we know, rapidly adaptive game AI has not yet been implemented in an actual video game.



Figure 2: Screenshot of the SPRING game environment. In the screenshot, airplane units are flying over the terrain.

IMPLEMENTATION

This section discusses our implementation of rapidly adaptive game AI. We subsequently discuss (A) the game environment in which we implement rapidly adaptive game AI, (B) the established evaluation function, and (C) an adaptation mechanism inspired by the case-based reasoning paradigm. Previously established opponent models [17] will be incorporated in future research.

Game Environment

The game environment in which we implement rapidly adaptive game AI, is the actual video game SPRING [10]. SPRING, illustrated in Figure 2, is a typical and open-source RTS game, in which a player needs to gather resources for the construction of units and buildings. The aim of the game is to use the constructed units and buildings to defeat an enemy army in a real-time battle. A SPRING game is won by the player who first destroys the opponent's 'Commander' unit.

Modern RTS games typically progress through several distinct phases as players perform research and create new buildings that provide them with new capabilities. The phase of a game can be straightforwardly derived from the observed traversal through the game's tech tree. Traversing the tech tree is (almost) always advantageous, yet there is a cost for doing so in time and game resources. In SPRING, three levels of technology are available. At the start of the game, a player can only construct Level 1 structures and Level 1 units. Later in the game, after the player has performed the required research, advanced structures and units of Level 2 and Level 3 become available.

Evaluation Function

To exhibit behaviour consistent within the game environment presented by modern video games, game AI needs the ability to assess accurately the current situation. This requires an appropriate evaluation function. The high complexity of modern video games makes the task to generate such an evaluation function for game AI a difficult one.
In previous research we discussed an approach to generate automatically an evaluation function for game AI in RTS games [4]. The approach incorporated TD-learning [20] to learn unit-type weights for the evaluation function, to reflect the actual playing strength of each unit type. Our evaluation function for the game's state is denoted by

$$v(p) = w_p v_1 + (1 - w_p) v_2 \tag{1}$$

where $w_p \in [0 \dots 1]$ is a free parameter to determine the weight of each term v_n of the evaluation function, and $p \in \mathbb{N}$ is a parameter that represents the current *phase of the game*. Our evaluation function incorporates two evaluative terms, the term v_1 that represents the material strength and the term v_2 that represents the Commander safety.
Previous research performed in the SPRING environment has shown that the accuracy of situation assessments are closely related to the phase of the game in which they are made [5]. To distinguish phases of the SPRING game, we map tech levels to game phases and distinguish between when tech levels are "new," and when they are "mature," as indicated by the presence of units with a long construction time. This leads us to define the following five game phases.

- **Phase 1:** Level 1 structures observed.
- **Phase 2:** Level 1 units observed that have a build time $\geq$ 2,500.
- **Phase 3:** Level 2 structures observed.
- **Phase 4:** Level 2 units observed that have a build time $\geq$ 15,000.
- **Phase 5:** Level 3 units or Level 3 structures observed.

Results of experiments to test the established evaluation function showed that just before the game's end, the established evaluation function is able to predict correctly the outcome of the game with an accuracy that approaches one hundred per cent. Note that this is not a trivial result, for two reasons. First, the evaluation function is tuned to make predictions that are good during a large part of the game, not only at the end, and hence it will trade prediction accuracy at the end of the game for higher prediction accuracy earlier in the game. Second,

if the goal of the game was to destroy all the opponent's units, a correct prediction would be easy to make at the end. However, the goal is to destroy the opponent's Commander, and we found that it sometimes happens that a player who is behind in material strength can win, often because the opponent's Commander makes a high-risk move, such as attacking strong enemy units on its own. An evaluation function that is based on comparison of material strength and Commander safety cannot take such moves into account other than allowing for their general statistical likelihood.

In addition, experimental results showed that the evaluation function predicts ultimate wins and losses accurately before half of the game is played. From these results, we concluded that the established evaluation function effectively predicts the outcome of a SPRING game and that the proposed approach is suitable for generating evaluation functions for highly complex video games, such as RTS games. Therefore, we incorporate the established evaluation function in the implementation of our rapidly adaptive game AI.

Adaptation Mechanism

In our approach, domain knowledge collected in a case base is utilised for adapting game AI. To generalise over observations with the problem domain, the adaptation mechanism incorporates an offline means to index collected games, and performs an offline clustering of observations. To ensure that game AI is effective from the onset of a game, it is initialised with a previously observed, successful game strategy. For online action selection, a similarity matching is performed that considers six experimentally determined features. The adaptation mechanism is algorithmically described below, and is subsequently discussed in detail.

```
//Offline processing
A1. Game indexing; calculate indexes for all
    stored games.
A2. Clustering of observations; group together
    similar observations.

//Initialisation of game AI
B1. Establish the (most likely) strategy of
    the opponent player.
B2. Determine in which parameter-band values
    this opponent strategy can be abstracted.
B3. Initialise game AI with an effective
    strategy observed against the opponent
    with the most similar parameter-band
    values.

//Online action selection
C1. Use game indexes to select the N most
    similar games.
C2. Of the selected N games, select the
    M games that best satisfy the goal
    criterion.
C3. Of the selected M games, select the most
    similar observation.
C4. Perform the action stored for the selected
    observation.
```

Game indexing (A1): We define a game's index as a vector of fitness values, containing one entry for each time step. These fitness values represent the desirability of all observed game states. To calculate the fitness value of an observed game state, we use the previously established evaluation function (denoted in Equation 1). Game indexing is supportive for later action selection, and as it is a computationally-expensive procedure, it is performed offline.

Clustering of observations (A2): As an initial means to cluster similar observations, we apply the standard k-means clustering algorithm [9]. The metric that expresses an observation's position in the cluster space is comprised of a weighted sum of the six observational features that also are applied for similarity matching. Clustering of observations is supportive for later action selection, and as it is a computationally-expensive procedure, it is performed offline.

Similarity matching (A2 and C3): To compare a given observation with another, we define six observational features, namely (1) phase of the game, (2) material strength, (3) commander safety, (4) positions captured, (5) economical strength, and (6) unit count. Similarity is defined by a weighted sum of the absolute *difference* in features values. The weighted sum for both clustering of observations and similarity matching is calculated as follows:

$$\begin{aligned} similarity &= ((1 + \text{diff}(phase_of_the_game)) \\ &* (0.5 * \text{diff}(unit_count))) \\ &+ \text{diff}(material_strength) \\ &+ \text{diff}(commander_safety) \\ &+ \text{diff}(positions_captured) \\ &+ \text{diff}(economical_strength) \end{aligned}$$

As observations are clustered, calculating the similarity between observations is relatively computationally-inexpensive. This is important, as similarity matching must be performed online.

Initialisation of game AI (B1-B3): To intelligently select the strategy initially followed by the game AI, we first determine which strategy the opponent is likely to use. In our game environment, perfect information of the opponent is available when loading the opponent game AI (i.e., the settings of all opponent parameters, indicating the strategic preferences, are known). We use this information to determine the initial strategy of the game AI. In a typical game-playing setting, however, such information is not directly available, but has to be established via statistical learning teachniques such as opponent modeling. This will be investigated in future work. When the opponent strategy has been established, we determine in which parameter bands [8] the opponent strategy can be abstracted. We define three bands for each parameter, 'low', 'medium' and 'high'. We subsequently initialise the game AI with an effective strategy observed against the most similar opponent. We consider a strategy effective when in previous play

it achieved a set goal criterion (thus, the game AI will never be initialised with a predictably ineffective strategy), and consider opponents strictly similar when the observed values of the parameter bands are identical.

Action selection (C1-C4): Using the established game indexes, we select the N games with the smallest accumulated fitness difference with the current game, up until the current observation. Subsequently, of the selected N games, we perform the game action of the most similar observation of the M games that satisfy a particular goal criterion. The goal criterium can be any metric to represent preferred behaviour. For instance, a preferred fitness value of 0 can represent challenging gameplay, as this implies that players are equally matched. Naturally, we have to consider that performing actions associated to similar observations may not yield the same outcome when applied to the current state. Therefore, to estimate the effect of performing the retrieved game action, we straightforwardly compensate for the difference in metric value between the current and the selected observation.

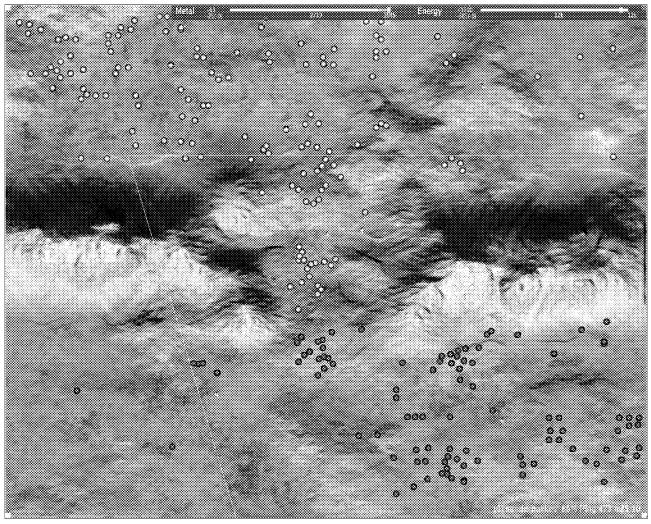


Figure 3: Two game AI's pitted against each other on the map 'SmallDivide'. In the screenshot, the white player has captured the centre position.

EXPERIMENTS

This section discusses experiments that test our implementation of rapidly adaptive game AI. We first describe the experimental setup and the performance evaluation, and then the experimental results.

Experimental Setup

To test our implementation we start collecting observations of games where two game AIs are posed against each other. Multiple SPRING game AIs are available. We found one open-source game AI, which the author called 'AAI' [18]. We enhanced this game AI with the ability to collect game observations in a case base, and the ability to disregard radar visibility so that perfect information on the environment was available. As opposing player, we used the original AAI game AI. We found 27 parameters that define the strategic behaviour of the game AI.[1]

For collecting observation, we simulate different players competing with different players, by for each game pseudo-randomising the strategic parameters of both players. The collection process was as follows. During each game, game observations were collected every 127 game cycles, which corresponds to the update frequency of AAI. With the SPRING game operating at 30 game cycles per second, this resulted in game observations being collected every 4.233 seconds.

We acknowledge that the amount of offline storage should be low for our approach to be considered practical for implementation in a game-production setting. We therefore store game observations in a lightweight fashion, by of each game observation only abstracting the position and unit-type of each unit. This abstraction, of approximately 3 KB per observation, provides a powerful basis for deriving observational features. Accordingly, a case base was built from 213.005 observations of 325 games, resulting in case base consisting of 679 MB of uncompressed observational data. Approaches are available to further reduce the size of the case base, such as offline data compression and subsequent online data decompression [2] and automatic condensation of the case base [3], however these lie outside the scope of the present research.

We collect observations from games played on the map 'SmallDivide'. This map is also used for adaptation experiments. The map, illustrated in Figure 3, is a symmetrical map without water areas. All games are played under identical starting conditions.

For offline clustering of observations, k is set to ten per cent of the total number of observations. Before the game starts, the initial strategy is determined. While the game is in progress, action selection is performed at every phase transition. The parameter N for action selection is set to 50, and the parameter M is set to 5. The game action is expressed by the configuration of the 27 parameters of strategic behaviour.

[1]The concerning parameters determine the game strategy on a high, strategic level, and not so much on a low, tactical level. Three examples of these parameters are AIRCRAFT_RATE (determines how many airplane units the AI will build), MAX_MEX_DEFENCE_DISTANCE (maximum distance to base where the AI defends metal extractors), and MAX_SCOUTS (maximum of units scouting at the same time). The authors happily provide a full list of the parameters on request.

Performance Evaluation

To evaluate the performance of the rapidly adaptive game AI, we determine to what extent it is capable of adapting effectively to game circumstances. We perform three different experiments. First, we test to what extent the rapidly adaptive game AI is capable of adapting to the original AAI game AI, set to play in a medium playing strength. Second, we test to what extent the rapidly adaptive game AI is capable of adapting to previously unobserved opponents, which is simulated by pitting the game AI against the original AAI game AI, initialised with randomly generated strategies. Third, we test to what extent the rapidly adaptive game AI is capable of upholding a tie, when pitted against the original AAI game AI, also set to play in a medium playing strength.

For each of the first two experiments, we perform a trial where the rapidly adaptive game AI is set to win the game (i.e., obtain a positive fitness value). For the third experiment, we set the rapidly adaptive game AI to uphold a tie (i.e, maintain a fitness value of 0, while never obtaining a fitness value less than -10, or greater than 10). To measure how well the rapidly adaptive game AI is able to maintain a fitness value of 0, the variance in fitness value is calculated. A low variance implies that the rapidly adaptive game AI has the ability to consistently maintain a predefined fitness value.

All experimental trials are repeated 25 times, except the trial where the rapidly adaptive game AI is pitted against randomly generated opponents, which is repeated 100 times.

Results

Table 1 gives an overview of the results of the first and second experiments performed in the Spring game. Figure 4 displays the obtained fitness value as a function over time of two typical experimental runs. The results reveal that when pitted against the original AAI game AI, set to play in a medium playing strength, the rapidly adaptive game AI effectively obtains a victory (80% of the experimental runs). This result indicates that rapidly adaptive game AI is effective in play against the original AAI game AI.

In addition, the results reveal that when pitted against the original AAI game AI, initialised with randomly generated strategies, the rapidly adaptive game AI improves on the effectiveness of strictly randomised games played without adaptation mechanism (i.e., a verified effectiveness of 50%). The obtained improvement in effectiveness is 8%. This result indicates that even in randomised play against the AAI game AI, the rapidly adaptive game AI is able to find effective strategies in the case-base.

Table 2 gives an overview of the results of the third experiment. The results reveal that the rapidly adaptive game AI is capable of upholding a tie for a relatively long time (37 minutes on average), while at the same time maintaining a relatively low variance in the fitness value that is strived for.

Opponent	#Games	Goal Ach.	Goal Ach.(%)	Impr.(%)
Original AAI	25	20	80%	30%
Random	100	58	58%	8%

Table 1: Effectiveness of rapidly adaptive game AI.

	Average	Standard deviation
Time to uphold tie	37 min.	15 min.
Variance in fitness value	2.04	0.59

Table 2: Upholding a tie with rapidly adaptive game AI.

DISCUSSION

In the experiments that test our implementation of rapidly adaptive game AI, we observed that the game AI was well able to achieve a victory when pitted against the original AAI game AI, set to play in a medium playing strength. We noticed that the rapidly adaptive game AI was able to find in the case base a strategy that could effectively defeat the original AAI game AI. As the original AAI game AI is not able to adapt its behaviour, the rapidly adaptive game AI could exploit its discovery indefinitely. Note that in some cases, the rapidly adaptive game AI did not win the game, despite it exhibiting strong behaviour. Such outliers cannot be avoided due to the inherent randomness that is typical to video games. For instance, in the Spring game, the most powerful unit is able to destroy a Commander unit with a single shot. Should the Commander be destroyed in such a way, the question would arise if this was due to bad luck, or due to an effective strategy of the opponent. For game AI to be accepted as effective players, one could argue, recalling the previously mentioned need for consistent AI behaviour, that game AI should not force a situation that may be regarded as the result of lucky circumstances.

In addition, we observed that even in randomised play against the AAI game AI, the rapidly adaptive game AI is able to find effective strategies in the case base, and was thereby able to improve on the randomised performance by 8%. This is a satisfactory result. As randomised play may be considered a simulated way to test the game AI against previously unobserved opponents, naturally, the question remains if the performance in randomised play can be further enhanced. We discuss two approaches to enhance the performance in randomised play.

First, note that our case-base currently consists of observations collected over 325 games. For randomised play, determined by 27 pseudo-randomised behavioural parameters, it would be beneficial to collect more games in the case base in order to increase the probability

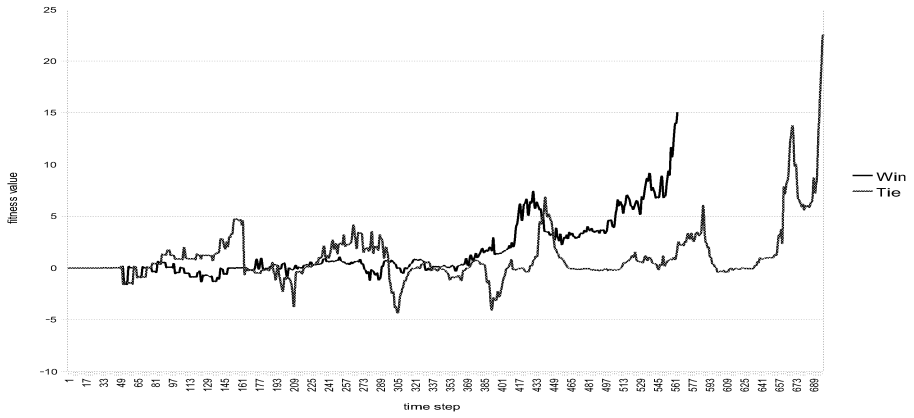


Figure 4: Obtained fitness values as a function over time, when pitted against the original AAI game AI. The figure displays a typical experimental result of (1) the rapidly adaptive game AI set to win the game, and (2) the rapidly adaptive game AI set to uphold a tie.

of it containing effective game strategies. As rapidly adaptive game AI can be expected to be applied in the playtesting phase of game development, and predictably in multi-player games, the case base in practical applications is expected to grow rapidly to contain a multitude of effective strategies.

Second, we observed that the final outcome of a SPRING game is largely determined by the actions performed in the beginning of the game. This exemplifies the importance of initialising the game AI with effective behaviour. In order to do so, one needs to accurately determine the opponent one will be pitted against. In videogame practice, (human) game opponents do not exhibit behaviour as random as in our experimental setup, but will typically exhibit behaviour that can be abstracted into a limited set of opponent models. Previous research has shown that even in complex RTS games such as SPRING, accurate models of the opponent player can be established [17]. We will therefore follow the expert opinion that game AI should not so much be focussed on directly exploiting current game observations, but should rather focus on effectively applying models of the opponent in actual game circumstances [16].

In addition, we found the rapidly adaptive game AI to be able to uphold a tie for a relatively long time, while at the same time maintaining a relatively low variance in the fitness value that is strived for. This ability may be regarded as a straightforward form of difficulty scaling. If a metric can be established that represents the preferred level of challenge for the human player, then in theory the rapidly adaptive game AI would be capable of scaling the difficulty level to the human player. Such a capability provides an interesting challenge for future research.

CONCLUSIONS AND FUTURE WORK

In this paper we discussed an approach to establish rapidly-adaptive game AI. In the approach, domain knowledge is gathered automatically by the game AI, and is immediately (i.e., without trials and without resource-intensive learning) utilised to evoke effective behaviour. In our implementation of the approach, game observations are collected in a case base. Subsequently, the case base is used to extract an evaluation function, and gathered cases are directly utilised by an adaptation mechanism. Results of experiments that test the approach in the SPRING game show that rapidly adaptive game AI can effectively obtain a victory and is capable of upholding a tie for a relatively long time. From these results, we may conclude that the established rapidly adaptive game AI provides a strong basis for effectively adapting game AI in actual video games. For future work, we will extend the established rapidly adaptive game AI with a means to scale the difficulty level to the human player. Subsequently, we will investigate how our approach to rapidly adapting game AI can be improved by incorporating opponent models.

ACKNOWLEDGEMENTS

This research is funded by a grant from the Netherlands Organization for Scientific Research (NWO grant No 612.066.406) and is performed in the framework of the ROLEC project.

REFERENCES

[1] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1), March 1994.

[2] Samir Abou-Samra, Claude Comair, Robert Champagne, Sun Tjen Fam, Prasanna Ghali, Stephen Lee, Jun Pan, and Xin Li. Data compression/decompression based on pattern and symbol run length encoding for use in a portable handheld video game system. US Patent 6416410, 2002.

[3] F. Angiulli and G. Folino. Distributed nearest neighbor-based condensation of very large data sets. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1593–1606, 2007.

[4] Sander Bakkes and Pieter Spronck. *AI Game Programming Wisdom 4*, chapter Automatically Generating Score Functions for Strategy Games, pages 647–658. Charles River Media, Hingham, MA., U.S.A., 2008.

[5] Sander Bakkes, Pieter Spronck, and Jaap van den Herik. Phase-dependent evaluation in rts games. In Mohammad Mehdi Dastani and Edwin de Jong, editors, *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (BNAIC)*, pages 3–10. Universiteit Utrecht, The Netherlands, 2007.

[6] Michael Buro and Timothy M. Furtak. RTS games and real-time AI research. In *Proceedings of the BRIMS Conference*. Arlington VA, 2004.

[7] Pedro Demasi and Adriano J. de O. Cruz. Online coevolution for action games. *International Journal of Intelligent Games and Simulation*, 2(3):80–88, 2002.

[8] Roger Evans. *AI Game Programming Wisdom*, chapter Varieties of Learning, pages 571–575. Charles River Media, 2002.

[9] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[10] Stefan Johansson, Jelmer Cnossen, and Tomaz Kunaver. Spring game engine. http://spring.clan-sy.com/, 2007.

[11] S Johnson. *AI Game Programming Wisdom 2*, chapter Adaptive AI: A Practical Example, pages 639–647. Charles River Media, Inc., Hingham, MA, 2004.

[12] Ronni Laursen and Daniel Nielsen. Investigating small scale combat situations in real-time-strategy computer games. Master's thesis, Department of computer science, University of Aarhus, Denmark, 2005.

[13] L. Liden. *AI Game Programming Wisdom 2*, chapter Artificial Stupidity: The Art of Making Intentional Mistakes, pages 41–48. Charles River Media, Inc., Hingham, MA, 2004.

[14] Lev Manovich. *The Language of New Media*. The MIT Press, Cambridge, Massachusetts, U.S.A., 2002.

[15] Alexander Nareyek. AI in computer games. *ACM Queue*, 1(10):58–65, 2004.

[16] Steve Rabin. *AI Game Programming Wisdom 4*, chapter Preface - What's happening to learning?, pages ix – xi. Charles River Media, Inc., 2008. ISBN 1-584-505230.

[17] Frederik Schadd, Sander Bakkes, and Pieter Spronck. Opponent modeling in real-time strategy games. In Marco Roccetti, editor, *Proceedings of the GAME-ON 2007*, pages 61–68, 2007.

[18] Alexander Seizinger. AI:AAI. Creator of the game AI 'AAI', http://spring.clan-sy.com/wiki/AI:AAI, 2006.

[19] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. Adaptive game AI with dynamic scripting. *Machine Learning*, 63(3):217–248, 2006.

[20] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[21] Laurie N. Taylor. Video games: Perspective, point-of-view, and immersion. Masters thesis, Graduate Art School, University of Florida, U.S.A., 2002.

[22] Paul Tozour. *AI Game Programming Wisdom (ed. Rabin, S.)*, chapter The Perils of AI Scripting, pages 541–547. Charles River Media, 2002.

COMPACTING A RULE BASE INTO AN AND/OR DIAGRAM FOR A GAME AI

Samuel Manier[1, 2] and Olivier Sigaud[1]
(1) Institut des Systèmes Intelligents et de Robotique
CNRS FRE 2507 Université Pierre et Marie Curie – Paris 6
4, place Jussieu, 75005 Paris, France
(2) Cyanide Studio
65, boulevard des bouvets,
92000 Nanterre, France
Samuel.Manier@gmail.com, Olivier.Sigaud@isir.fr

KEYWORDS
Game AI, Rule base, Decision Diagram, And/Or Diagram, Behavior Tree, Cycling simulation.

ABSTRACT

A game Artificial Intelligence defined by if-then rules can become hard to maintain and expand. In this paper we present a method to transform in a compact way a rule base into an And/Or Diagram, a hierarchical structure easier to manage over time. Our method is illustrated in a commercial cycling simulation video game.

INTRODUCTION

In game Artificial Intelligence (AI), basic if-then rules are often preferred over more powerful architectures such as Hierarchical Finite State Machines (HFSMs) (Harel 1987) or Behavior Trees (BTs) (Champandard 2008), because they do not require developing any complex engine or editor. Thanks to their simplicity, developers can quickly compose simple behaviors for game agents. However, their poor maintainability and scalability make them inappropriate when the complexity of these behaviors exceeds a certain degree. But the growth in complexity is not always easy to anticipate, especially for game series, such as sport games, which evolve in sequels in a more and more complex manner. In such a context, developers are often constrained to update their AI by rewriting it from scratch in a better formalism when it becomes too complex, depriving them of the possibility of re-using the knowledge contained in the original AI.

In this paper, based on our experience with a commercial sport video game, we propose a method to transform and compact semi-automatically a rule base in what we call an And/Or Diagram (AOD), whose formalism is close to the one of a BT. This method is divided into successive steps of transformation from a formalism to another:

- the semi-automatic transformation of the rule base into a decision diagram base;
- the automatic conversion of the decision diagram base into an AOD;
- the automatic refining (factorization and simplification) of the AOD;

In the following sections, we will detail those steps, present an application of our method to the Pro Cycling Manager game developed by Cyanide and discuss it, before concluding and giving future work directions.

TRANSFORMING A RULE BASE INTO AN AOD

Our starting point is an action selection system defined by a rule base where each rule has a premise, a conclusion and a priority (see Figure **1** for a small example). A premise is a conjunction of conditions on variables, a conclusion is a conjunction of discrete actions, whereas priorities define the order of rules testing, forbidding the activation of multiple rules at the same time.

r_1: if V1 = true and V2 < 50 and V3 < 10 then do Action1
r_2: if V1 = true and V2 < 50 and V3 > 20 then do Action2
r_3: if V1 = true and V4 = true then do Action3

Figure 1: Three if-then rules forming a rule base, ordered by decreasing priority. V1, …, V4 are variables.

The successive steps to transform and compact a rule base into an AOD are presented in the following subsections.

From A Rule Base To A Decision Diagram Base

The rule base we described is a flat structure that cannot handle common properties of rules without calling upon redundancy (for example, in Figure **1**, the condition *V1 = true* is repeated in each rule). This can slow down considerably the understanding, the modification and the evolution of the AI. In this section, we explain how we reduce this redundancy by representing the rule base into a decision diagram base.

A decision diagram (Akers 1978) is a rooted Directed Acyclic Graph (DAG) composed of:

- decision nodes. A decision node represents a test on the value of a variable;
- edges. An edge coming from a decision node represents a condition on its variable;
- leaves. A leaf represents a solution to the decision diagram problem (for an action selection problem, a leaf is an action or a set of actions).

A decision diagram is executed from the root. At each decision node, the next visited node is designated by the edge whose condition is satisfied, until a leaf is reached.

We propose an algorithm to transform a rule base into what we call an Ordered Multi-Terminal Algebraic Decision Diagram (OMTADD) (designation deriving from Ordered Binary Decision Diagrams (OBDDs), Multi-Terminal Binary Decision Diagrams (MTBDDs) [Drechsler and Sieling, 2001] and Algebraic Decision Diagram (ADD) [Bahar et al. 1993]). An OMTADD is:

- Algebraic, because we use non binary variables (they can be Boolean, integer, float or defined by enumeration);
- Multi-Terminal, because we use non binary answers (in the leaves);
- Ordered, because our algorithm needs an order on variables.

In the rest of this paper, DD stands for OMTADD for short.

Our algorithm is a modified version of the DPLL procedure proposed by Huang and Darwiche (Huang & Darwiche 2004). The DPLL procedure is originally designed to transform a Conjunctive Normal Form (CNF), i.e. a conjunction of disjunctive conditions, into an OBDD. We adapted it to transform a rule base into a DD.

This algorithm requires an ordering on the variables. But, finding a good ordering for many variables is a hard problem. The resulting decision diagram can be huge if this ordering is not carefully chosen. We narrow down this problem by dividing the rule base into smaller rule bases, each one concerning a small enough subgroup of variables. It boils down to do the intuitive operation of regrouping rules using the same variables, or almost the same variables, together. However, this reorganization is constrained to respect rules priorities: the resulting rule bases are ordered by priority, each small rule base getting the priority of its first rule, and the following property is imposed: let r_1 and r_2 be two rules with compatible premises, r_1 belonging to the rule base R_1 and r_2 to the rule base R_2 (with $r_1 \neq r_2$ and $R_1 \neq R_2$). R_1.priority > R_2.priority implies that r_1.priority > r_2.priority.

The transformation of each resulting rule base into a DD is done separately by the *omtadd* algorithm presented below. We first give a brief summary and some definitions.

In this recursive algorithm, variables are instantiated successively with all their possible values. This makes premises of some rules becoming satisfied, creating decision nodes (corresponding to variables instantiations) and leaves (corresponding to rules conclusions). When several nodes are equivalent, only one is kept. The use of cutsets allows anticipating equivalence between nodes, avoiding computing them all.

Our definition of a cutset is the same as in (Huang & Darwiche 2004), with "clause" replaced by "rule": the i^{th} cutset of variable order $\pi = v_1,\ldots,v_n$ for the rule base $\{r_1,\ldots r_m\}$, denoted $\text{cutset}^i_R(\pi)$ or cutset_i for short, is defined as $\{r \in R : \exists\ j \leq i < k$ such that rule r mentions variables v_j and v_k in its premise$\}$.

A cutset value is a bit vector, each bit representing the state (satisfied or simplified) of the premise of a rule of this cutset. When two ore more configurations are found to have the same cutset value, only one of them will be computed, its DD cached, and others will simply generate a cache hit and have their DD immediately returned.

In a DD, the fail action is the one activated when all the rules represented by the DD fail. In that case, the next DD (with a lower priority) is evaluated.

Algorithm *omtadd*(Rules *R*, int *i*) : return the root node of the DD corresponding to the rule base *R*, whose rules are ordered by priority, according to a fixed order on variables. *i* is the current variable index.

---

if at least the premise of one rule is true **then**
  let *r1* be the rule of highest priority among rules with a true premise;
  if there is no rule r_2 with simplified premise such that $r_2.priority > r_1.priority$ and $r_2.actions \neq r_1.actions$ **then**
    return $r_1.actions$;
if all variables are instantiated **then**
  return the fail action;
if ($node_already_computed = cache_{i-1}[value(cutset^{i-1})]$) ≠ nul **then**
  return *node_already_computed*;
result = create_node (R, i);
$cache_{i-1}[value(cutset^{i-1})] = result$;
return *result*;

Function *create_node*(Rules *R*, int *i*) : create a node, or return an already existing node, where the variable of index *i* is tested.

---

let *N* be an empty set of nodes;
for each possible value *val* of the variable v_i, **do**
  N.insert(omtadd(R|v_i = val, i+1));
if all the nodes in *N* are equals **then**
  return the first one;
else
  let *Node[i]* be the set of nodes representing the variable of index *i*;
  for each node *n* of *Node [i]*, **do**
    if the children of *n* are exactly the nodes in *N* **then**
      return *n*;
  result = Node(i, N);
  Node[i].insert(result);
  return *result*;

Function *Node(i, N)* : create a node having *i* as variable index and the nodes contained in *N* as children.

The choice of the variable order in a BDD is hard [Tani 1993]. This is especially true for an MTADD, which is more complex than a BDD. Therefore, we do not automate that part: we proceed manually, by trial and error, to the choice of the variable order of each rule base, until we obtain a DD with a low enough degree of redundancy.

Figure **2** shows a DD resulting from the transformation of the rule base extracted from the rule base of Figure **1**. We can see that the redundancy of conditions *V1 = true* and *V2 < 50* is avoided.

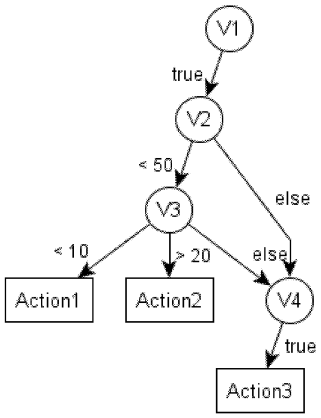


Figure 2: The DD resulting from the transformation of the rule base in Figure **1**, according to the following variable order: V1, V2, V3, V4. The Fail action and edges that lead to it are not represented.

From A Decision Diagram Base To an AOD

DDs can represent disjunctions (see *else* edges in Figure **1** and, for another type of disjunctions, left side of Figure **3**). But when these disjunctions involve numerous conditions, some nodes may have to be accessed from many edges. This contributes to slow down the editing of DDs (e.g. in Figure **3**, replacing the node N by a node N' would require reconnecting to N' all edges leading to N). In this section, we show how we convert DDs into AODs. AODs represent disjunctions by OR nodes which, combined with AND nodes, make the AI modular and easier to reconfigure. The connection of all AODs into a single AOD is also explained.

An AOD is a rooted DAG where nodes can be either leaves (conditions, actions or a named link to other AODs) or composite nodes. There are different types of composite nodes:

- AND nodes. An AND node succeeds only if all its children succeed;
- OR nodes. An OR node succeeds if at least one of its children succeeds. Its children are ordered by priority;
- If-Then-Else nodes. An If-Then-Else node succeeds if both its If and Then children succeed or if its If child fails and its Else child succeeds. An equivalent of an If-Then-Else can be constructed with one OR node and two AND nodes.

The execution of an AOD starts from the root in a depth-first search. When a node fails, we backtrack to its parent and continue its execution (e.g. an OR node will execute its next child whereas an AND node will fail). Leaves representing actions always succeed.

We call successful sub-diagram the result of the execution of an AOD. It is composed of nodes that succeeded and that have a parent (except for the root) belonging to this sub-diagram. Actions contained in the successful sub-diagram form together the final decision.

The conversion of a DD into an AOD is done by searching recursively for patterns upon which we apply transformation rules in a priority order. These rules are explained in the following figures (**3** to **6**) presented in decreasing priority order. In these figures, nodes N, N1, N2 and N3 are not necessarily leaves, and nodes V and V1 are not necessarily roots.

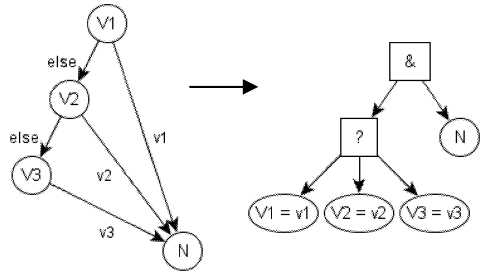


Figure 3: Conversion of a DD with disjunctions into an AOD.

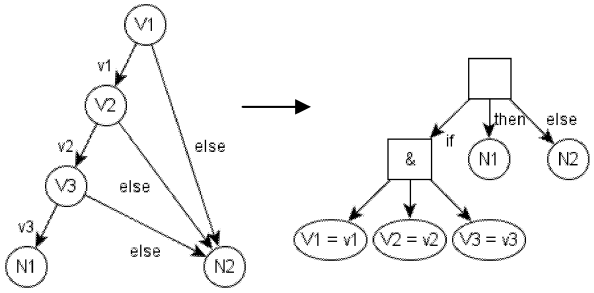


Figure 4: Conversion of a DD with another type of disjunctions into an AOD.

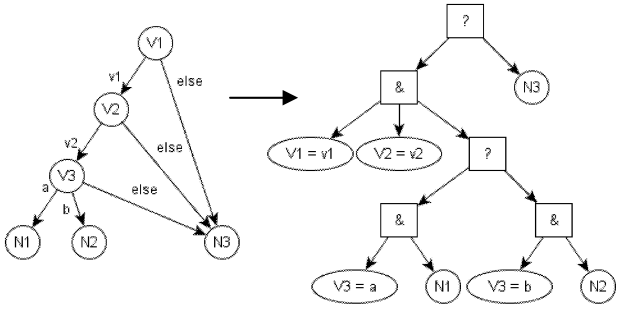


Figure 5: The general case, from the preceding rule, when the last decision node (here V3) has more than two children

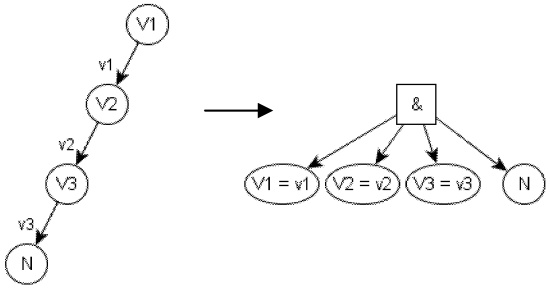


Figure 6: Conversion of a conjunction of conditions leading to the node N into an AND node responsible for the execution of these conditions and N.

AODs benefit from their hierarchical organization (see Figure **7** for an example). A (sub-)diagram can be seen as a

high level condition (if all its leaves are conditions) or a high level action (if at least one of its leaves is an action). Its complexity can be hidden by representing only its root or a named link to its root. We use this property to link all generated AODs under a large OR node and thus regroup the whole action selection system into a unique AOD.

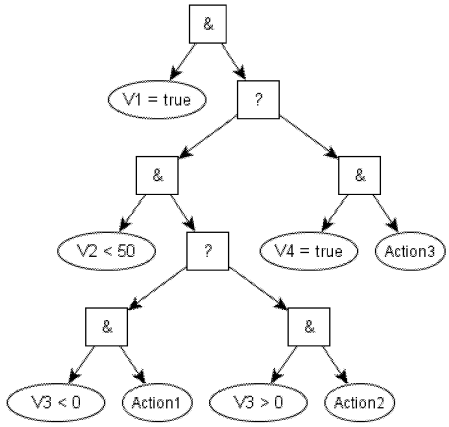


Figure 7: The AOD resulting from the transformation of the DD of the Figure **2** (equivalent to the rule base of Figure **1**).

Refining Tools

We can further clarify an AOD and reduce its size with the following tools.

Factorization
From the definition of the successful sub-diagram and of the different composite nodes, we can infer that:

- when the successful sub-diagram contains an AND node, it necessarily contains all its children,
- when the successful sub-diagram contains an OR node, it contains only one of its children,
- when the successful sub-diagram contains an If-Then-Else node, it contains either its IF child and its THEN child or ELSE child.

Let N be a node with multiple parents. Let A be the common ancestor of parents of N. If all possible successful sub-diagrams coming from A contain N, then we can replace N by an always true condition and replace A by AND(A,N).

Simplification
After its construction and its factorization, an AOD may contain unnecessary nodes that can be eliminated (like true conditions created by factorization), provoking a recursive simplification of the AOD. As this procedure is intuitive, we will not detail it.

Creation of high level conditions and actions
Some too large AODs can remain hard to understand and manipulate. Their hierarchical property can be exploited to hide the complexity of some sub-diagrams into named links to those sub-diagrams. We propose a tool which first selects automatically sub-diagrams that have characteristics making them good candidates to be replaced by named links, and secondly let the user validate their replacement and choose the names of the links. The sub-diagrams proposed to the user are:

- sub-diagrams created from rules of figures **3** to **6**, because they contain a high concentration of conditions that are susceptible, taken together, to reveal a high meaning;
- multi-parented sub-diagrams. As they are accessed from multiple contexts, they may represent relevant concepts.

APPLICATION TO A GAME

We applied the method described in this paper to the high level action selection system of the game Pro Cycling Manager. In this section, we first present the game and its original AI, then we show how we applied our method to it.

Pro Cycling Manager And Its AI

Pro Cycling Manager is a management game where the player supervises a professional cycling team through a career or a simple race in single or multi-player mode. Each race can be simulated or played in real time 3D, as shown on Figure **8**. A race brings about 20 teams of 9 riders together, among which the player's team. The player controls the actions (attack, sprint, relay …) of his/her riders as well as their energy expenditure.



Figure 8: A screenshot of a race in Pro Cycling Manager

Our work focuses on the action selection system of the riders during a race. It is organized into 4 levels:

- **Level 4: Group tactic selection**. A group is composed of riders (up to 3) from the same team on the basis of a leader and whose goals are those of the leader. For example, to catch up with a dangerous opponent, the group tactic could be *Take relays* or *Attack*, with a high effort.
- **Level 3: Individual roles allocation**. This level is in charge of giving a role to each member of the group. For example, if the chosen tactic is *Attack*, only the leader will attack. On the contrary, if the chosen tactic is *Take relays*, all the group will participate to the relay except the leader.
- **Level 2: Role fulfillment**. Each role is controlled by a Finite State Machine (FSM). For example, the states of the FSM of the role *Take relays* are: *Enter the relay queue*, *Take a relay*, *Go to the end of the relay queue*.

- **Level 1: Dealing with physics**. Here are computed the forces applied on a rider, based on its role, its energy, its effort, its competences and its environmental conditions (slope percentage, wind speed and direction, obstacles …).

The player interacts with his/her riders by giving them individual roles. Levels 1 and 2 are therefore shared by player's riders and computer controlled riders.

We applied the transformation method we described in this paper on the highest and most complex level: the group tactic selection. As it was not implemented exactly as a rule base, but rather as conventional procedural programming, we needed a prior transformation stage that we will not describe because of its specificity to that game.

Application Of Our Method: Results

The starting rule base contained 427 rules, bringing together 4506 conditions on 167 variables.

From The Rule Base To A Decision Diagram Base
From these 427 rules, 50 thematic rule bases of different priorities were created by hand. Each small rule base was automatically transformed into a DD on the basis of a manually optimized variable ordering. On that occasion, the total number of conditions dropped down from 4506 to 939 (being the number of edges (1467) minus the number of *else* edges (528)).

From A Decision Diagram Base To an AOD + Refining
The automatic conversion of the DD base into an AOD eliminated the 528 *else* edges but, even after the refining of this AOD, necessitated 741 composite nodes.

Testing
We integrated the obtained AOD in Pro Cycling Manager and successfully verified its functional equivalence with the initial group tactic selection module by comparing their decisions on a very large number of race situations.

DISCUSSION

The transformation of a rule base into a DD base, when the reorganization into smaller rule bases and the variables orderings are carefully chosen, significantly reduces the redundancy of conditions shared by multiple rules. Concerning the conversion of a DD base into an AOD, the diagram size is slightly increased to provide a hierarchical and modular structure allowing a large action selection system to remain easily tunable and expandable.

Our method could be improved by automating the steps that are currently processed manually, particularly the variable ordering step which was the most time consuming. This problem is computationally hard but, for rule bases of reasonable sizes, an automatic solution could help.

AODs are not the ultimate representation of an action selection system. At least they are less powerful than Behavior Trees (BTs). AODs are close to BTs since AND nodes in AODs correspond to parallels in BTs and OR nodes correspond to selectors, but BTs can also represent sequences that endow them with memory and let them behave like HFSMs. Thus, it is possible to create a BT from an AOD by adding sequences to it. Conversely, the automatic factorization and simplification tools we designed for AODs could be adapted to BTs (more precisely to sub-BTs that do not contain sequences) that were written by hand, resulting in more compact behavior definitions.

Our initial AI was hard coded. We used the opportunity of its transformation to separate its formalism from the game code. It made on-the-fly modifications possible and gave the possibility to create tools to help analyzing and editing the AI.

CONCLUSION

In order to rewrite a game AI based on if-then rules into a formalism facilitating its maintenance and its capacity to grow in size, we proposed a method to semi-automatically compact a rule base into an And/Or Diagram (AOD). On a game that continues to evolve over years, our method allows modernizing an action selection system by reusing the work of past generations of AI programmers.

In the future, we plan to integrate fuzzy conditions and composite nodes to allow defining a behavior with a lower number of nodes, and fuzzy actions to prevent crisp changes of behavior under small changes of circumstances. We also plan to develop tools helping the editing process of AODs by masking irrelevant parts to the user.

REFERENCES

Akers. S. B. 1978. "Binary Decision Diagrams", in *IEEE Transactions on Computers*, C-27(6):509–516.

Bahar R. Iris, Frohm E. A., Gaona C. M., Hachtel G. D., Macii E., Pardo A., et Somenzi F. 1993. "Algebraic decision diagrams and their applications". In *Intl. Conf. Computer-Aided Design*, 188-191, IEEE.

Champandard A. 2008. "Getting Started with Decision Making and Control Systems". In *AI Game Programming Wisdom 4*, Charles River Media Inc., 257-264.

Drechsler R., Sieling D. 2001. "Binary decision diagrams in theory and practice". In *Int. Journal on Software Tools for Technology Transfer*, 3(2):112-136.

Harel, D. 1987. "Statecharts: A Visual Formalism for Complex Systems". In *Science of Computer Programming,* 8:231-274.

Huang, J., et Darwiche, A. 2004. "Using DPLL for efficient OBDD construction". In *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing*, 127-136.

Tani S., Hamaguchi K., Yajima S., 1993. "The complexity of optimal variable ordering of a shared binary decision diagram". In *Proc. 45th ISAAC*, *Lecture Notes in Computer Science, Springer*, Vol. 762:389-39.

NEURAL NETWORKS AND AGENT BASED SIMULATION

Da Vinci Code Game
An Neural Network approach to an imperfect information game

Jeroen Boogaard and Leon Rothkrantz
Man-Machine Interaction Group
Delft University of Technology
2628 CD, Delft,
The Netherlands
E-mail: L.J.M.Rothkrantz@tudelft.nl

KEYWORDS
AI, Neural Networks, games, decision making, self-play, interaction, agents, unsupervised learning, reinforcement learning, Hopfield, Kohonen, backpropagation.

ABSTRACT

This paper is about a digital version of the game Da Vinci Code. Different neural network archtectures such as Hopfield and Kohonen have been used to chose an optimal next move. Several rules of action selection have been defined. These rules are tested in some experiments. The results of the experiment are reported
Although the Da Vinci Code game includes the element of chance, it is possible to become a 'good player' after training.

INTRODUCTION

In todays game development, the main focus is on sound and graphics. But AI is becoming more and more important since the game industry realizes the entertainment value of non-playing characters [1,2,3]. For many board games like chess, checkers, and backgammon, AI-players have been developed succesfully. Even human grandmasters have been defeated in e.g. checkers, chess, draughts, and backgammon. The succes
of most of the strongest AI-players in these board games is due to the use of the possibility to look upon many thousands of positions per second by using computer power and efficient search algorithms. AI-players can beat human players in games where the branching factor of the search space is small. AI-players for games which include the element of chance like go, bridge, and poker can be beaten handily by human players. The core of game AI is decision-making; an AI-player behaves intelligent if it consistently makes decisions that we consider appropriate in a given context. This report describes the development of an AI-player for the Da Vinci Code game [5,6], a zero-sum imperfection information game which includes the element of chance. To be succesful in this complex environment we will use of Aritificial Neural Networks since they have the ability to learn from their environment, and to improve their performance through learning.

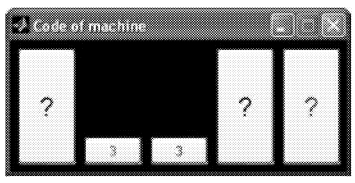
BACKGROUND

In this section we present the game for which we build an AI player. There was not yet a software implementation of the game so we had to program the game itself. There are a few differences between our implementation and the original game but we will still use the name Da Vinci code game.1
The goal of the game is to uncover the code of your opponent.

Environment:
Objects: 12 black and 12 white blocks each labelled with one of the symbols from the alphabet {0,1,…,11}.
Players: Our implementation has to be played with two players; one of them is always the AI-player. The AI-player can play against a copy of itself.
Control: The game will be guided by an independent control program that keeps track of all blocks, moves, and scores, and regulates the communication between the players via graphical user interfaces.

Initialisation

Each player starts with a secret code containing four blocks which are randomly drawn from the bag. The labels are not visible to the opponent player. The blocks are arranged such that they form a sequence in increasing order of their numerical label and colour. If two blocks have the same colour then the red one is considered to be lower valued than its blue counterpart. The sequence forms a unique secret code that has to be uncovered by the opponent player. All blocks that do not belong to a player are stored in the bag such that their labels are not visible to any of the players. The bag is managed by the control.



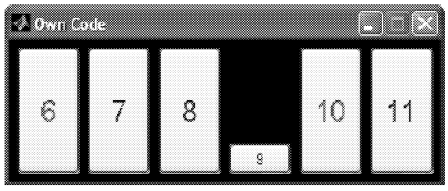
Figures 1: Code of the opponent from which two blocks are visible

States and actions
At the start of each game-round the active player (the player that is in turn) receives a block that has been drawn randomly from the queue. At this point his opponent player is unable to see this block. The active player attacks the code

of his opponent by estimating the value labelled on one of the hidden blocks that it contains. The control compares the estimated value with the actual value. There are two possibilities:

1. The estimated value does not match the actual value:
The active player has to insert his new block in his own secret code such that the label is also visible to his opponent player OR
2. The prediction is right:
The player who owns the uncovered block has to make this block visible. This leads to two possible actions of the active player:

a. He selects another object (not necessarily from the same player) to estimate the value of; OR
b. He finishes his turn by inserting the object in his secret code such that the label is not visible to his opponent.



Figures 2: Own code containing five blocks that are not visible to the opponent

Goal states
The game ends as soon as one of the codes has been cracked i.e. all blocks it contains are uncovered. The owner of this cracked code has lost the game; his opponent has won. If the bag is empty before one of the codes has been uncovered, the game is also ended but without a winner (draw).

TRAINING AN AI-PLAYER

Training an AI-player requires game examples [4]. Playing against human players is very time consuming. A better approach is to use a database of already played games recorded at a tournament being representative of good game-playing. Unfortunately, there are or no such databases available for the Da Vinci Code game. An alternative approach is self-play.
In training the checkers program written by Arthur Samuel (in the early 1950's) two copies of the AI-player were playing against each other without human intervention. The skill of the AI-layer increased during these battles against itself. Self-play simulations also provide a method for the comparison of two or more different AI-players and to qualify the contribution of each individual learning element. Tesauro's backgammon player [8] reached the level of human masters after months of self-play. A drawback of self-play is that it suffers from start-up problems. To evaluate the effect of training, a performance criterion is needed. Patist and Wiering [7] developed a neural network based draughts agent for which
the equity was being measured after each game. In their paper2 they defined equity as the number of wins minus by the number of losses divided by the games played. Note that a stronger criterion would be:

Equity = {#wins–(#losses+#draws)} / #games.

AI-PLAYER

Action Generator
When the AI-player is in turn, it is requested (by the control) for a move i.e. it has to select a not already visible block that he expects to be present at a 'hidden' position in the code of its opponent. Obviously, not all blocks can be a candidate for every hidden position. Therefore, each game-round, the AI-player first collects all available information and then extracts all possible moves from it. In this section we present algorithms to determine all possible moves.

Value/ Position	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11
0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1
1	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1
2	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1
3	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0
5	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	1

Figures 3: Evaluation table for the hidden positions 0,1,2,3,5

The environment is not completely observable; all the agent can sense is
- his own code
- the block that is taken from the bag
- the length of the opponent code
- the positions and colours of hidden blocks contained in the opponent code
- blocks of the opponent code that are made visible (not available during the first round)

From this list, one can extract a list of candidate blocks i.e. blocks that arenot currently visible to the agent and could thus be part of the opponent code. For a complete search space, the candidates and hidden positions must be combined to form a representation of all possible moves. The 'evaluation table' of figure 3.1 is a representation in which the columns correspond to the candidates and the rows to the hidden positions. Each candidate value appears twice; the first value represents the red block, the second value the blue block. Note that the table satisfies the ordering rule discussed in next section. Each table entry or block-position pair is assigned the value one or zero corresponding to a possible or not possible move respectively. Initially, all entries are set to one.

Legal moves
Obviously, all legal moves must be possible but not all possible moves are legal. Therefore a little pre-processing is needed before the actual evaluation of moves.

Theorem 1
Let s be a hidden block, p=position(s), and {s, p} be a move m; colour(s)=0 iff s is a red block, otherwise colour(s)=1; let t be a visible block contained in the same code c; m is called legal if and only if:

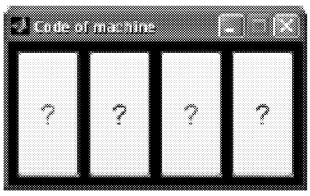
1. colour(s) equals the colour of the hidden block at position p
2. if p<position(t) then value(s) < value(t)+colour(t)
3. if p>position(t) then value(s) > value(t)+colour(t)

Each move that violates the last two conditions will be deleted (see figure 3.2), this can be done by setting the corresponding table entry to zero.

```
if the opponent code contains visible blocks then
        for every visible position
                for every hidden position do
                        1. delete blocks that have a bigger value+colour than a visible
                        block but are candidate for a lower position
                        2. delete blocks with a value+colour smaller or equal to the
                        value than a visible block but are candidate for a higher position
```

Figures 4: Algorithm to test condition 2 and 3 of theorem 1

The secret code of figure 5 trivially has the colour pattern [red,blue,red,blue]. Because of each value both the red and blue block are represented in the evaluation table, we need to eliminate some of the candidates in order to satisfy condition 1 of theorem 1. For each hidden position half of the candidates can be eliminated; all candidates represented by even columns or all candidates represented by odd columns. Since the evaluation table only contains zeros and ones, the elimination can be done by creating two bitvectors: redpattern and bluepattern which can be subtracted from the vector composed of the columns on a given row in the table. Each bitvector contains 24 bits consisting of alternating zeros and ones (see figure 6).



Figures 5: A secret code that contains the colour pattern [red,blue,red,blue]

```
Let redpattern = [1 0 1 ... 0]' and bluepattern = [0 1 0 ... 1]'
For each row corresponding to a hidden position
        if the colour of the block at the hidden position is red then subtract the bluepattern'
        else subtract the redpattern
```

Figures 6: Removing wrong candidates by adding a colour pattern

The remaining moves all satisfy theorem 1 but some are clearly 'better' than others. For example, consider the move {[1 11],1} meaning a blue 11 at position 1. Although this move will be legal if all other blocks are hidden, it will never be a 'good' move. The reason is that if the blue 11 occurs in a code, it will always appear at the rightmost side of the code. A similar reasoning holds for the lowest block; if a code happens to contain the red 0, then this block will always have position 1. Knowledge about the presence or absence of these two extremes is useful as they provide an indication about the range of the code. It is therefore a good strategy
to use the first game rounds to uncover the boundary blocks of the opponent code. This kind of knowledge will not be directly programmed into the AI-player; it is expected that the optimal strategy will be found by one of the learning elements that we discuss in one of the next sections.

Wrong moves
During a game, the agent can get stuck in a kind of selecting the same wrong move every round. Although the code of the opponent changes every two game rounds, there is no guarantee that the estimate will be correct the next round (note that the position of the unknown block does not change if a block from the bag is placed at a higher position). This situation can be avoided by removing the player's last wrong move from the search space. The search space can even be further reduced by considering the wrong moves of the opponent too. Since blocks that are contained in a player's own code are not considered to be candidate blocks for his opponent code, every wrong move of the opponent player
tells us that the estimated block can not be contained in his code or in the bag. So that block can be removed completely from the evaluation table.
But we should be careful for human players who are trying to mislead their opponent. Therefore we consider the pponent's wrong move only during one round. Wrong moves will be stored by the game control and used and updated by the AI-player.

Learning elements
The evaluation table provides us with a list of possible moves. But how do we know if a particular possible move is also a 'good' move? In other words: which moves are more are the most probable moves? In this section we present three different learning elements, each using a different neural network to evaluate all possible moves. The evaluation table is extended such that each table entry is assigned a value representing the 'belief' of being a 'good move', which will result in entry values greater than one.

Kohonen Network to cluster the candidates
In the previous section we discussed how we can create a list of candidate blocks for the opponent code. However, it is not clear which blocks are candidate for which positions. In other words, we do not know how the values are distributed over the code.
Competitive ANN's can be used to subdivide the input space. The Kohonen learning rule tries to associate each neuron with a group of patterns. Assume that we have a Kohonen ANN that is trained one thousand example codes. If we present this network the list of candidates and we create an one-to-one mapping between
neurons/clusters and code positions, we will have an estimate for the distribution of the candidates over the positions. The feature space will only contain the possible values and the number of neurons must equal the number of code positions.

Hopfield Network to restore the code
Imagine that we already have played one thousand games and therefore have seen more than thousands of different codes. We will then have some a kind of intuition about the occurrence of particular values in a particular situation. This is especially true for codes which are partially visible since we will associate them with codes we have already seen

before. The above is the fundamental idea behind learning element ANN2. Hopfield Networks are capable of associating input patterns with patterns that they were trained on [11,14]. Assume that we have a Hopfield Network that is trained on thousands of example codes. If we interpret a partially hidden code as a distorted
pattern and present it to the network, the network will output a code with which the input associates the strongest. Given that the training set contains enough codes, the network will be able to output a code which is very similar or even equal to the opponent code. The performance will depend strongly on the number of blocks that are already visible; we expect this technique to make poor estimates about completely hidden codes.

Reinforcement Learning Network to learn by trial and error

The two previously discussed learning elements we can estimate the likeliness of particular values and their most probable position. They provide us the information we need to select 'good' moves. But if we apply the reinforcement learning approach, we can evaluate the moves directly. A Markov decision problem (MDP) [10] is a controlled stochastic process that satisfies the Markov property and has a reward function. The Markov property states that the transition probabilities to possible next states only depend on the current state and the selected action. For every state in the Da Vinci Code game it holds that the probability of a hidden block to be uncovered only depends on the current state of the code and the action of the player that is attacking the code. The Da Vinci Code game can thus be seen as a MDP. A solution to a MDP is a mapping from states to actions maximizing the reward. Such a solution can be found using the reinforcement learning approach [11,15].
Since each move can be qualified as a success or failure, it can trigger a numeric reinforcement equal to 1 or -1 respectively. Assume we have a list of all moves and their corresponding reinforcement of a finished game. We can use this to train a Feedforward ANN which will serve as a function approximator to evaluate all possible moves during the next games. Unfortunately, there is no tournament database available to train the network a priori. Figure 3.8 shows the functionality of learning element ANN3 which uses a Feedforward ANN for reinforcement learning.

Rules for action selection

In the preceding section Action Generator is discussed which provides the agent with a collection of legal moves. Three different kinds of learning elements were presented that provide the agent with some kind of game experience that is expected to increase its performance. Figure 14 shows the complete functionality of the AI-player. Each of the learning elements is implemented as a separate module indicated by a circle, together they form a series of 'experts' which try to estimate the 'successfulness' of moves. Each learning element is required to output a value between 0 and 1 (like probability and possibility values). The second requirement is that zero valued table entries once zero valued can never be updated to non-zero values to ensure that all moves remain legal. The update values which will be used to modify the corresponding entry values of the evaluation table according to some update rule. This section covers all update rules as well as the (neural) input-output mappings that lead to the update values.

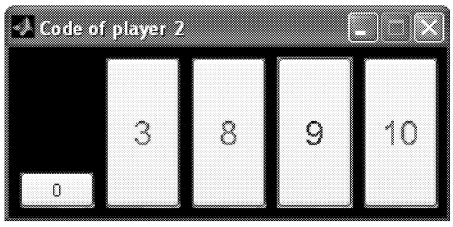
Update rule for ANN1

Learning element ANN1 is responsible for the ordering of candidate blocks according to their expected position. From the list of candidates two vectors are extracted; one contains the colours, the other contains the corresponding values. These two vectors form the feature space that is being clustered by the Kohonen Neural Network. Since the number of hidden positions is variable, 8 Kohonen Networks are created that differ in their number of output neurons. The result of this clustering is a new vector consisting of the estimated positions of the candidates (see table 1). The output vector contains all positions of the opponent code including positions that are not hidden anymore (compare table 1 with fig. 7). All three vectors together will be used to update the evaluation table in such a way that zero entries remain zero. If the Kohonen Network finds fewer clusters than the number of positions, the network will be trained (unsupervised) on the current candidate list and stored for future uses.

Table 1: Clustering 19 candidates with ANN1

Candidate	*Colour*	*Value*	*Winning neuron*	*Estimated position*
1	0	0	3	1
2	1	1	3	1
3	0	2	3	1
4	1	2	3	1
5	0	3	5	2
6	1	3	5	2
7	1	4	5	2
8	0	5	5	2
9	1	5	5	2
10	0	6	2	3
11	1	6	2	3
12	1	7	2	3
13	0	8	4	4
14	1	8	4	4
15	0	9	4	4
16	1	9	4	4
17	0	10	1	5
18	0	11	1	5

The results of ANN1 as shown in table 1 can be evaluated by observing the corresponding opponent code presented in figure 7 (off course this cannot be done by the agent itself causing ANN1 to be unsupervised).



Figures 7: Example code; the stone that lays down is uncovered, the purple border indicates that this stone is selected for estimation

One should notice that the red 3 (candidate 5) is correctly classified as a candidate for position 2. In contrast, the red 8 (candidate 13) is erroneously clustered as a candidate for position 4, probably due to the big 'jump' in value between the second and third position. Both the blue 9 (candidate 16) and the red 10 (candidate 17) are correctly identified as position 4 and position 5 candidates respectively. Although most of the candidates are clustered correctly this does guarantee that moves i.e. these block-position combinations will be selected by the agent. The reason is that for ANN1 all candidates that are clustered for a particular position are considered to be equally probable, e.g. for position 5 the red 18 10 and red 11 equally likely i.e. the probability of a correct estimate for position 5 is 50%. In contrast, the probability of selecting the correct candidate for position 2 is only 20% (5 candidates). How do we use the output of the Hopfield Network to determine which moves are the most promising? We can reason that the best moves are those for which the probability on success is the greatest (in this case: an estimate for position 5). Alternatively we can argue that the best move is a move that will provide us the most information i.e. an estimate for a hidden block with the highest self information (in this case the block at position 2). We choose the first option and therefore the update rule for ANN1 becomes:

```
For every cluster candidate
    Find row number and column number of the corresponding table entry
    If Evaluation table (row number, column number)>0
        Evaluation table (row number, column number) =
        Evaluation table (row number, column number) +
        1 / (number of candidates for this hidden position)
    End
End
```

One of the artefacts of the clustering procedure is that candidates can have a different colour than the block on the position which is associated with (a consequence of being an unsupervised classification problem). These blocks cannot lead to wrong moves since the update rule will not let them to increase the evaluation of the corresponding table entry.

Update rule for ANN2

The goal of learning element ANN2 is to indicate which blocks are more likely to occur in a partly hidden code. Example codes of the same length, colour pattern and visible blocks are presented to a Hopfield Network. The network is triggered by the (partly hidden) opponent code and tries to find associations with the learned example codes. Since the input values must lie between -1 and +1, a new representation of codes is needed which we shall call signcodes. A signcode is 24 bits code that works like an indicator matrix. Although the length of the binary code is fixed, the length of the 'gamecode' it represents can be of any length. The second row of the table in figure 8 shows the signcode for the gamecode in figure 7. Note that the first row is the same as the first row of an evaluation table. It should also be noted that the signcode only represents the blue 0 as this is the only visible block. Because each bit indicates whether or not its corresponding row entry is known to be contained in the gamecode, a signcode can be interpreted as a sequence of 24 Bernoulli variables X~Ber(n,p) where n is the length of the corresponding gamecode.

0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8: Signcode representing the gamecode of figure 7

Next, a list of example codes of the same (gamecode) length are loaded and filtered with the opponent signcode of figure 7. The resulting list only contains those example codes having exactly 5 bits equal to 1 from which one bit indicates the blue 1, and having the same colour pattern as the opponent signcode. Figure 9 shows one of example codes after filtering (blue 1, red 8, red 9, blue 10, red 11).

0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0

Figure 9: One of the example signcodes after filtering the trainset

The resulting list of example signcodes will be used as target vectors for the Hopfield Network. Simulation if the network on the opponent signcode will generate the output presented in the second row in figure 10.

0	0	1	1	2	2	3	3	4	4	5	5
0	1	0	0,078	0	0,078	0	0,078	0	0,078	0	0,078

6	6	6	7	7	8	8	9	9	10	10	11	11
0	0	0,078	0	0,078	0	0,078	0	0,078	0	0,078	0	0,078

Figure 10: Output of Hopfield Network on a barely visible code

Note that output suggests that each block of the proper colour is equally likely to occur in the opponent code. This makes sense since the network is only told that the code starts with a blue 0. Now assume that also the third and fifth blocks are visible, if we repeat the procedure the output will show more variability (see figure 11). Like ANN1, ANN2 also reflects the uncertainty caused by the big jump in value between the second and third position. ANN2 seems to work according to the maximum likelihood principle since it builds a probability model for which a partly hidden code is most likely.

0	0	1	1	2	2	3	3	4	4	5	5
0	1	0,122	0	0,122	0	0,122	0	0,122	0	0	1

6	6	7	7	8	8	9	9	10	10	11	11
0,122	0	0,122	0	1	0,28	0	0,28	1	0,28	0	0

Figure 11: Output of Hopfield Network simulated on a 60% visible code

The evaluation table will be updated according to the following rule:

```
For every row
    For every column
        If Evaluation table (row number, column number)>0
            Evaluation table (row number, column number) =
            Evaluation table (row number, column number) +
            Hopfield output (column number)
        End
    End
End
```

Remember that each row of the evaluation table corresponds to a hidden position in the opponent code. As a consequence, the ones that are present in the output in figure 11 will not be used.

Update rule for ANN3

Learning element ANN3 operates directly on moves. But how do we represent a move? We need a list of features that represent all relevant information about the conditions and action in a particular game state to be able to predict the successfulness of the action. In the list displayed in figure 12, the features p1 and p2 typically represent conditions while p5 clearly characterizes the action. Features p3 and p4 describe the environment as well as the action.

Table 2: List of features

P1	Number of candidates for the hidden position
P2	Length of the opponent code
P3	Position of the hidden block
P4	Colour of the hidden block
P5	Estimate for the hidden block value

Because all features of the list in figure 12 are numerical, they can be composed into a 5D feature vector. The feature vector provides the input for the Reinforcement Learning Feedforward Neural Network (*RLNetwork*). The network outputs values are being normalized to fall in the interval [0,1].

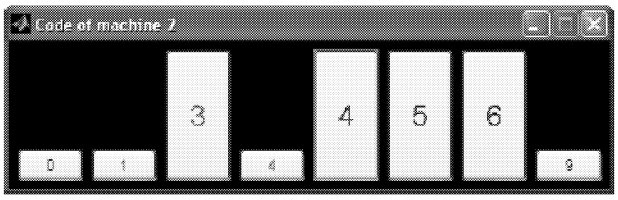


Figure 12: Example opponent code

For example consider the simulation of ANN3 for the opponent code of figure 12. The simulation results displayed in figure 13 show that ANN3 strongly recommends a move in which position 3 or 5 is involved. It learned from experience that value 4 at position 7 is very unlikely. This learning element is most sure about the first proposed move (a red 3 at position 3) which will truly be successful in this situation.

Colour	*Value*	*Position*	*RL-Network output*	*Normalized output*
0	3	3	1,01	0,75
1	4	5	0,94	0,68
1	5	5	0,90	0,64
1	6	5	0,83	0,57
1	4	6	0,35	0,09
1	5	6	0,32	0,06
1	6	6	0,29	0,03
1	4	7	0,26	0,00

Figure 13: Example of simulation ANN3 results

The update rule for ANN3 is:

```
For every legal move
    Find row number and column number of the corresponding table entry
        Evaluation table (row number, column number) =
        Evaluation table (row number, column number) +
        Normalized RL-network output value
    End
End
```

Decision rules for action selection

After each learning element has reported its belief in and preference for a number of moves, the evaluation table contains a certain amount of variation. The agent could simply select the move that has the largest utility i.e. the move corresponding to the maximum valued table entry since most of the time this will yield a single action, but then it will take the risk of throwing away good moves that have a slightly lower evaluation value but can be as good as or even better than the move with the highest evaluation value. Therefore a more *fuzzy* approach would be to state that more than one move can be 'the best move', to a certain degree (of membership). For this we calculate the median and maximum
of all evaluation values that are greater than or equal to one. Hence the decision rule I becomes:

```
For every legal move
    If evaluation value > median(evaluation values) OR
    evaluation value == maximum(evaluation values)
    Then add move to selection
End
```

The OR is necessary to cover the case where there is only one move witha nonzero evaluation causing the median to this evaluation. In all other cases we prefer not to select moves with an evaluation equal to the median.
A disadvantage of decision rule I is that when the evaluation data contains little variance as this will cause many moves to be selected. This situation often occurs at the start when there is a much uncertainty about the opponent code. Decision rule II forces the agent in each situation to only select the 'best few'. The underlying principle of decision rule 2 is that for each game state it holds that the number of moves that equal the real block-position pairs of the opponent code can be at most the number of hidden positions. Because this number can be very low we select the #(hidden positions)+1 best moves. There are no restrictions on the positions of these moves e.g. if positions 1, 4, and 6 are the hidden positions, three moves concerning position 1 and one move for position 6 can be selected ignoring position 4.

Decision rule II:

```
Sort moves in decreasing order of evaluation
Select the min(movenr,hiddenlen+1) best moves
```

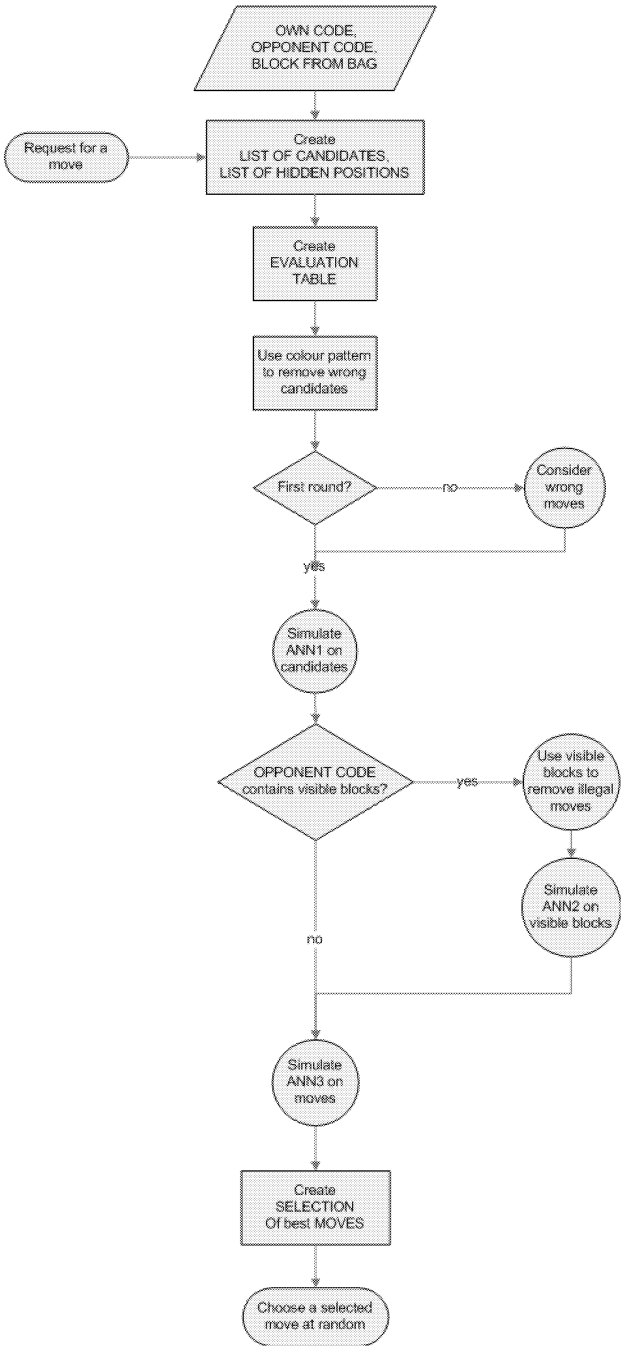


Figure 14: Simulation of AI-player

TRAINING AND SIMULATION

Training the learning elements

From one of the preceding sections we know how to train an agent using self-play. To be able to measure its progress, we let the agent play against a nonintelligent copy of itself i.e. a version of the agent in which only the functionality of section Action Generator is included. The non-intelligent agent contains no learning elements at all and will therefore play at a constant level. In contrast, the intelligent agent will learn by training ANN1 and ANN3. We measure the progress in intelligence by calculating the equity. Both agents will play in offensive mode. The intelligent agent will use decision rule 2 .

Experiments

After training, we would like to investigate the individual behaviour of the learning elements. For this we set up an experiment in which two different implementations of the agent have to compete. A similar experiment can be used to check whether the offensive mode is really always the best strategy. In the final experiment the agent has to compete with a human.

Experiment I: local vs global

In this first experiment we create two different AI-players (agents) and let them play 10 games against each other. The agents we create are:

- **Player1** : a local level online learning agent using only learning elements ANN1 (Hopfield) and ANN2 (Kohonen)
- **Player2**: a global level offline learning agent in which only ANN3 (reinforcement learning) is implemented.

The two AI-players have a completely different game strategy; Player1 considers example codes to derive a kind of statistical model that results in a list of candidates and their most likely position in the code. There is no consideration about which of those moves will be most successful. Each time the number of clusters of the Kohonen ANN is less than the number of positions of the opponent code, the network will first be retrained. Player1 is thus an online learning agent. Player2 operates on a more global level as it evaluates the moves directly. Since this agent learns offline and starts with a 'tabula rasa' we expect this agent to perform very bad during the first games. All moves of both players are stored and used for training the reinforcement learning network so Player2 will also learn from the good and bad moves of Player1. This is why we expect that this agent will learn much faster than his opponent.
In the next experiments the learning ability of the agents will be turned off so they can only use the knowledge they acquired during training.

Experiment II: offensive vs defensive

In this experiment the agents of experiment I are combined into one super AI-player playing 1000 games against itself in another play mode.
After a correct estimation a player has to decide what to do with the block he had taken from the bag (see 2.1.2). There were two options:
a. inserting the block in his code such that is remains invisible to its opponent
b. keeping the block in hand and make a new estimation, taking the risk of having to insert the block in his code such that it is visible to his opponent

From this decision we can derive two AI player types:

- a defensive player who will always choose to protect his code ignoring its closeness to winning (a)
- an offensive player who will always choose to continue its attack no matter close his opponent is to winning (b)

The AI player types are implemented as play modes: offensive or defensive. Thus far, the agents have always played in offensive mod. In experiment II we like to discover which play mode is the most successful. Therefore we create two new agents:

- **Player1**: containing learning elements ANN1, ANN2 and ANN3 and plays in the offensive mode
- **Player2**: implements the same learning elements as Player1, but plays in the in the defensive mode.

One note must be made about Player2: if the situation occurs in which Player2 takes the last block from the bag, it is unwise to end its turn after a good move because the next game round will be a draw. Therefore Player2 is forced to continue its game round if there are no more blocks left in the bag.

RESULTS

Results of training

Figure 15 displays the equity as a function of the number of games. The biggest increase in performance occurs in the area somewhere between 250 and 750 games. The plot shows a little dip between 750 and 900 games and becomes more and more linearly when approximating 1500 games. A linear curve indicates that the agent does not learn but simply plays better than his opponent. So we can conclude that the effect of adaptation is the greatest during the first 1000 games.

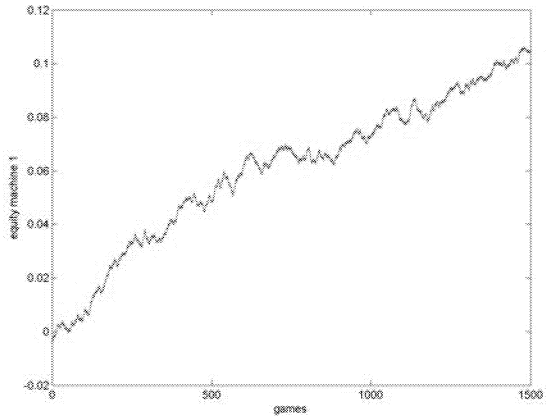


Figure 15: Equity of intelligent agent

In figure 16 one can see the result of the competition between the learning (Player1) and non learning agent (Player2). Player1 829 of the 1500 games and only 7 games resulted in a draw. Clearly, the learning elements lead to better decisions although the difference is not very big.

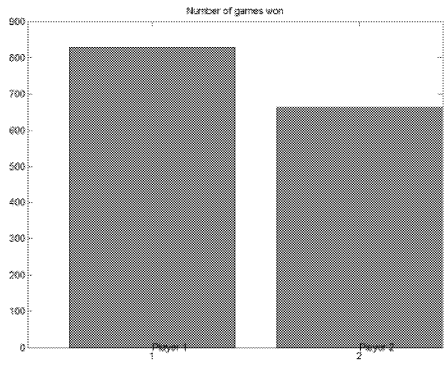


Figure 16: Result of competition between agent with ANN's (Player 1) and agent without ANN's (Player2)

Results of experiments

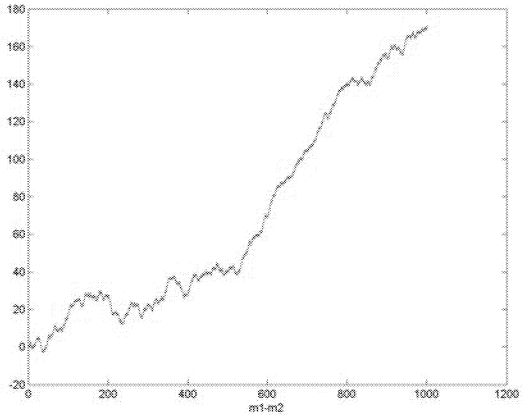


Figure 17: Difference between the number of games won by the local agent and the global agent

Experiment I: local vsglobal

	Global agent			
	Win	Loss	Draw	Result
Local agent	583	412	5	1171-829

Experiment II: offensive vs defensive

	Defensive agent			
	Win	Loss	Draw	Result
Offensive agent	7	3	0	14-6

Experiment III: machine vs human

	Human			
	Win	Loss	Draw	Result
AI Player	5	5	0	10-10

	No learning elements			
	Win	Loss	Draw	Result
ANN1	7	3	0	14-6
ANN2	6	4	0	12-8
ANN3	7	3	0	14-6
Human	6	4	0	12-8

All three Neural Networks contribute to a better play. The local (statistical) agent performs somewhat better than the

global (reinforcement learning) agent, probably due to the fact that the local agent uses two learning elements (ANN1 and ANN2) while the global agent only uses ANN3. The offensive play style is realer better in most cases. The performance in speed of ANN2 is superior over ANN1, ANN2, and a human player. The agent which uses all three ANN's plays like an experienced human player.

CONCLUSION

Although the Da Vinci Code game includes the element of chance, it is possible to become a 'good player' after training. To make sure an agent will strictly follow the rules of the game, it is necessary to implement some game knowledge. One of the big advantages of Neural Networks is that they adjust their weights by training and therefore tune themselves.
Nevertheless when one implements Neural Networks into an agent there is a huge amount of parameters that must be set. An evaluation table provides a suitable representation to monitor the contributions of each individual ANN. Both the Hopfield and Kohonen network operate like statistical estimators which lead to fruitful results. Nevertheless, the reinforcement learning ANN contributes to a better play performance as well. The Hopfield Network makes good estimates but slowed down the entire game speed. The Kohonen Network provides a powerfull unsupervised learning technique. Reinforcement learning is a somewhat mystic AI-technique that can act rather intelligent.

BIOGRAPHY

[1] Vincent de Boer, Leon Rothkrantz, Pascal Wiggers, "Invincible - A Stratego Bot", *the International Journal of Intelligent Games & Simulation*, vol. 5, no. 1, 2008.

[2] J.M.P. van Waveren, L.J.M. Rothkrantz, "Artificial player for Quake III Artena", *International Journal of Intelligent games and Simulation*, vol. 1, no. 1, pp. 25-35, 2002.

[3] Marijn Heule Leon Rothkrantz, "Solving Games, dependence of applicable solving procedures", *Science of Computer Programming*, 1, vol. Special Issue on computer games, no. 67, pp. 105-124, Elsevier, June 2007.

[4] Game AI
http://www.cs.ucla.edu/~ashapiro/ShapiroA_CG2002.pdf#search=%22self%20play%20in%20games%22
http://www.itee.uq.edu.au/~penny/_strategy.htm

[5] The neural net backgammon programs
http://satirist.org/learn-game/systems/gammon/

[6] Da Vinci code game
http://www.anderspel.nl/davinci.html

[7] Jan Peter Patist and Marco Wiering, "Learning to Play Draughts using Temporal Difference Learning with Neural Networks and Databases", Utrecht University.

[8] Gerald Tesauro "Temporal Difference Learning and TD-gammon".

[9] L.P. Kaelbling, M.L. Littman, A.W. Moore
"Reinforcement Learning: A Survey".

[10] M.L. Littman, T.L. Dean, L.P. Kaelbling
"On the Complexity of Solving Markov Decision Problems",

[11] "Neural Network Toolbox, For Use with MATLAB"", Howard Demuth and Mark Beale.

[12] "Structure-Adaptable Digital Neural Networks", AndrésPérez-Uribe.

[13] "Reinforcement Learning: A Tutorial", M.E. Harmon and S.S. Harmon, Wright State University.

[14] Hopfield Networks
http://library.thinkquest.org/18242/associate.shtml
http://fuzzy.cs.uni-magdeburg.de/~borgelt/doc/hopfield/
http://physics.syr.edu/courses/modules/MM/n_net/memory.html

[15] Reinforcement Learning
http://lslwww.epfl.ch/~anperez/rlbj.html
http://researchweb.watson.ibm.com/massive/tdl.html

Extending Game Technologies with Virtual Reality and Multi-Agent Systems

A. Barella and **C. Carrascosa** and **V. Botti**
Dept.de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Campus de Vera, 46022 - Valencia
{tbarella, carrasco, vbotti}@dsic.upv.es

Abstract

Nowadays, videogames, computer simulations and training are very complex applications because users require higher and higher levels of quality. Artificial Intelligence plays nowadays a fundamental role in videogames and virtual reality applications. Agents interacting with humans must have a credible visual appearance and must maintain user immersion. The most important issues of user's interest are both realistic visualization and convincing intelligence. This paper presents a system that allows a full integration of two areas which are becoming increasingly important recently: *virtual reality* (visualization) and *intelligent multi-agent* systems (artificial intelligence), in a flexible and easily scalable way. This system extends the technological possibilities of the videogames.

INTRODUCTION

Computer game development covers many different areas, such as graphics, artificial intelligence and network communications. Nowadays, players demand more sophisticated and credible computer-controlled opponents, but results are sometimes unsatisfactory. This is due to the need of real-time processing constraints to reach an acceptable feeling of *immersion* for the user. Sometimes, these high computational costs are solved by using dedicated hardware, such as graphic cards. However, in the case of artificial intelligence, there is no specific hardware developed to help programmers and users, and outcomes are sometimes poor. The development of systems, tools and development environments can play an important role in the progress of this field. Thus, the combination of artificial intelligence techniques and virtual reality (or virtual environments) has given birth to the field of *intelligent virtual environments* (*IVE*s). An IVE is a virtual environment simulating a physical (or real) world, inhabited by autonomous intelligent entities (Luck and Aylett 2000). These entities have to interact in / with the virtual environment as if they were real entities in the real world. In addition, entities and the virtual environment have to be shown to users in an appropriate way. There are some other typical problems of an IVE that we have to solve, as explained in (Bierbaum et al. 2001). Independence of the underlying technologies is one of the principal requirements to support cross-platform application development (both hardware and operating system). A virtual reality system has to allow user interactions: not only displaying images, but using specific devices such as trackers, gloves, etc. That is, a device abstraction layer is necessary for user interaction. Furthermore, synchronization is a critical issue, specifically both data and image synchronization. Every visualization module must have the same data to create a scene graph, in order to maintain coherence. In case of multiple displays, every display must show the same order of image generated.

The main purpose of this work is to integrate Artificial Intelligence and Virtual Environments. One of the most interesting AI technique for this purpose, due to its features (scalable, autonomous, proactive, reconfigurable, agile, etc...), is the Multi-Agent Systems (MAS) approach.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

BACKGROUND AND MOTIVATION

Using 3D videogame engines to create intelligent virtual environments has been a choice used widely, because of the commercial success and the high level of photorealism that is achieved by current technology (Andreoli et al. 2005). The usual approach for applications to commercial game engines is to integrate an agent as a *bot* player in the game. Quake and Unreal Tournament are the most prefered because of this facility to create and modify *bots*. These *bots* can be implemented using any AI technique; for example, M. van Lent et al. (Laird and Duchi 2000) use SOAR as an inference engine and B. Gorman et al. (Gorman, Fredriksson, and Humphrys 2005), use MATLAB to control the bot. Considering a *bot* as an agent, agent techniques can also be applied; for example E. Norling (Norling 2003) uses JACK for a BDI agent bot.

Other choice is to use simulation packages. There are some powerful packages for the simulation of decentralized systems. One of the most popular ones is Swarm(Minar et al. 1996), but it doesn't provide a framework for 3D simulations or visualizations. An-

other package is Breve(Klein 2002), which is an integrated simulation environment for the implementation of decentralized systems and artificial life simulations in 3D worlds. In Breve, simulations are written in an interpreted language ("steve") and, therefore, they are integrated in this application for execution.

A more laborious choice is to create a framework from scratch. A wide range of approaches may be found in literature following this choice. For instance, DIVA (Vosinakis, Anastassakis, and Panayiotopoulos 1999) (and its evolution, VITAL (Anastassakis, Ritchings, and Panayiotopoulos 2001)) is developed to use a Prolog-based engine for deliberation. Another approach is to develop a virtual environment with only one agent acting as a *wizard* agent for training and educational purposes such as *STEVE* (Rickel and Johnson 1997). It can be found some research in the field of crowd simulations in virtual environments (Ulicny and Thalmann 2001)(Musse and Thalmann 2001). Or even there are approaches that use AI to give human-like expressiveness or movement to their virtual characters (Thalmann and Noser 1999).

In these approaches, artificial intelligence and graphics use to be embedded. Thus, they are *ad-hoc* applications that are not very extensible nor scalable.

This paper presents a case study of a multi-agent system applied to a virtual environment, using JGOMAS(Barella, Carrascosa, and Botti 2006) (as acronym for *Game Oriented Multi-Agent System* based on *JADE*) as framework for IVEs. Specifically, agents playing a *capture the flag* game in a 3D virtual environment while are visualized on different systems. Thus, the purpose of this case study is to show the flexibility of this approach, which allows both distributed intelligence calculation and visualization of the IVE on different devices, such as Head Mounted Displays, CAVE^{TM}, and Powerwall^{TM}.

DESCRIPTION OF THE SYSTEM

According to this approach, both systems, virtual reality and multi-agent, have to work separately, in an independent way in order to create a flexible and versatile framework. Thus, there is a clear separation (figure 1) of the intelligence part (where reasoning is computed) from the visualization part (where the results of reasoning are displayed), as proposed by (Ulicny and Thalmann 2001).

On the one hand, the artificial intelligence problem has been put as a distributed system, breaking the problem down into subproblems. Agents are used as subproblem-solvers. Thus, an agent has only part-information knowledge, that is, it cannot obtain direct access to the complete virtual environment state. Using its reduced knowledge, each agent can figure out what it needs to do in order to reach its designed objectives. This means that nobody has to tell the agent explicitly what to do at a given moment. In this way, a multi-agent system is a system that consists of a number of agents, that communicate messages through a computer network. These agents will cooperate, coordinate, and negotiate with each other, trying to achieve their own goals and common objectives, as a result of emergent behaviour(Wooldridge and Jennings 1995).

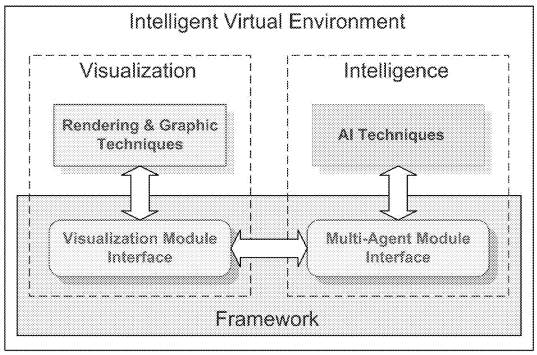


Figure 1: Independent parts in an IVE.

On the other hand, a visualization module is necessary in order to show to users what is happening in the virtual world. This module should cover from standalone users at home to complex virtual reality systems, such as a CAVE^{TM}.

The framework is, at present time, at its first stage. Thus, a *capture the flag* game is proposed as the kind of social interaction to simulate, where agents are grouped into two teams (allies and axis). Allies must go to the axis base, capture the flag, and bring it to their base, in order to win the game. Axis agents must defend their flag against the allies and, if the flag is captured, they must return it to their base. There is a time limit for the allies to bring the flag to their base. If the time limit expires, the axis team wins the game.

This framework can be used as a simulator for validating the coordination, communication, and learning algorithms in the field of multi-agent systems (or artificial intelligence in general). To do this, JGOMAS provides an API that allows designers to add their own code modifications (*mods*), to improve the intelligence of the *player* agents.

JGOMAS ARCHITECTURE

The framework is composed of three subsystems, as shown in figure 2:

Multi-Agent Platform

JGOMAS uses JADE(Bellifemine, Poggi, and Rimassa 2001) as multi-agent platform. JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middleware that complies with the FIPA(FIPA 2002) specifications .

The reason of JGOMAS using JADE as a multi-agent platform is because JADE has become a *de facto* standard in the multi-agent community. Thus, JGOMAS takes advantage of all the resources that JADE offers:

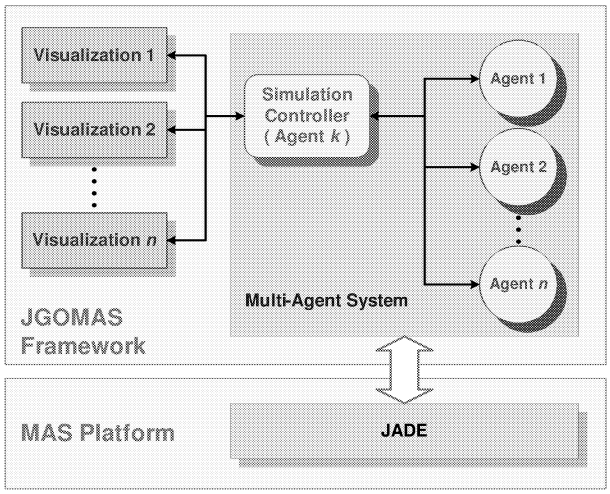


Figure 2: JGOMAS architecture overview.

behaviour mechanisms, message passing, FIPA interaction protocols, etc..., in compliance with the FIPA specifications (a collection of IEEE standards which are intended to promote the inter-operation of heterogeneous agents and the services that they can represent).

Therefore, JGOMAS user has only to implement the intelligence of his agents, avoiding wasting time in low-level technical issues as, for example, inter-agent communication.

Multi-Agent System

JGOMAS' multi-agent system can be viewed as an abstraction upper layer over JADE(Barella, Carrascosa, and Botti 2007). Two main classes of agents are defined: *simulation controller* and *inhabitant* (player) agents.

There is only one *simulation controller* running during a game. It can be viewed as a wrapper of the virtual environment, because it translates all *player* agents' actions to the virtual world. This is the way in which this agent synchronizes the virtual environment's data, maintaining the consistency throughout the game at any time. In addition, *simulation controller* is the interface for *visualization module*, because it sends the game state to graphic viewer clients, acting as a server.

The other agents are *inhabitant* agents simulating *players* in the virtual scenario. These agents can carry out basic actions, such as look, move, etc. The way an *inhabitant* agents achieves a goal is carrying out tasks that changes the virtual world state. Then, these actions are encapsulated into messages to be sent to *simulation controller* using the communication mechanisms of JADE platform.

Next, *simulation controller* performs those actions in the virtual world and, finally, this agent informs *inhabitant* agents affected by the consequences of the actions (*simulation controller* generates events for *inhabitant* agents involved in those changes).

Visualization Module

As mentioned above, one of the main goals is that artificial intelligence and virtual reality systems have to work independently (it is possible to launch the JGOMAS multi-agent system even if there are no graphic viewers connected). In order to make it easy to users, a basic graphic viewer has been implemented. *Render Engine* is the graphic viewer application developed *ad hoc* to display the 3D agents, objects, and the scenario in JGOMAS, and it has been designed as an external module (and not as an agent) and written in C++, using the graphic library OpenSceneGraph (OSG). The OpenSceneGraph is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling.

Render Engine is an important part of the framework, but JGOMAS is not forced to use it, because other graphic engines (both commercial and open source) could be used, for example, to utilize an existing one or to get better image rendering. This is possible because the visualization module in JGOMAS framework is independent of the artificial intelligence module (multi-agent system).

Although *Render Engine* can be used both by standalone users and head mounted display users (stereoscopic view), it is difficult to synchronize multiple displays to perform a virtual reality system, such as a CAVE^{TM}. The CAVE^{TM} is a projection-based VR system that provides real-time head-tracked perspective with a large field of view, interactive control, and stereo display. It used to be a "cube" with images projected onto three walls and the floor. To avoid that effort, *Render Engine* has been extended to use VRJuggler(VRJuggler) as middle-ware to use a complete virtual reality system.

Virtual Reality System VR Juggler is a collection of technologies which provide the tools necessary for virtual reality applications development. VR Juggler allows a user to run an application on almost any virtual reality system. It acts as "glue" between all the other Juggler components. From this, VR Juggler provides a virtual platform for virtual reality application development.

The flexibility of VR Juggler allows applications to execute in many virtual reality system configurations including desktop virtual reality, Head Mounted Displays, CAVE^{TM}-like devices, and Powerwall^{TM}-like devices.

Figure 3 shows the way *Render Engine* uses VRJuggler. The kernel interface provides hardware abstraction, and the draw manager (specifically we have used the OSG draw manager) provides the abstraction for the graphics API. This way we get device abstraction and operating system independence. In addition, VRjuggler provides the ability to run multiple applications simultaneously (image synchronization). As mentioned above, the *simulation controller* is in charge of data

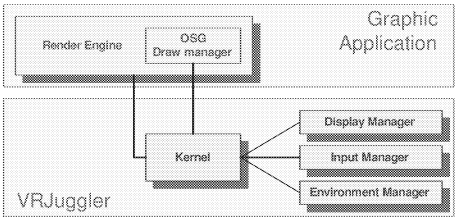


Figure 3: Extending *Render Engine* for use VRJuggler.

synchronization. In this way, synchronization is solved.

EXAMPLE FOR USING JGOMAS

In this section, an example for using JGOMAS is shown, where it is noticed how the virtual reality system and the multi-agent system work independently from each other. In this example, several flexible and versatile configurations have been designed. These configurations cover from stand-alone users at home to complex virtual reality systems, as shown in figure 4.

Since JGOMAS is a distributed system, it is not necessary to use a high-performance computer to support the execution of several agents. In this example, JGOMAS' multi-agent system was executed in a cluster of computers. This way, cheap computers connected through a high-speed network were used in order to get the necessary performance.

Once JGOMAS' multi-agent system is being executed in the cluster, many visualization modules can be used at the same time:

- *stand-alone system*: users can be playing or supervising the game at anywhere.
- *head mounted display system*: users are watching the game in stereoscopic mode.
- *Powerwall^{TM} system*: a group of users viewing the game (perhaps in stereoscopic mode) in a grid of displays which are composing a macro-display (that is, a Powerwall^{TM}).
- *CAVE^{TM} system*: users are each one immersed in a CAVE^{TM} to get an astounding experience in virtual reality systems.

Figure 5 shows an execution of JGOMAS. A *hand made* Powerwall^{TM} with four monitors was composed, and images are rendered by two computers forming a cluster. VRJuggler is used to synchronize the images displayed. At the same time, the CAVE^{TM} system was connected to the current JGOMAS game. Figure 6 shows a picture of a person using the CAVE^{TM}.

CONCLUSIONS

This paper presents a multi-agent framework based on JADE for virtual environments that has been designed to study a full integration of artificial intelligence (specifically multi-agent systems) and real-time graphic applications (specifically virtual reality systems) that can be applied to videogames.

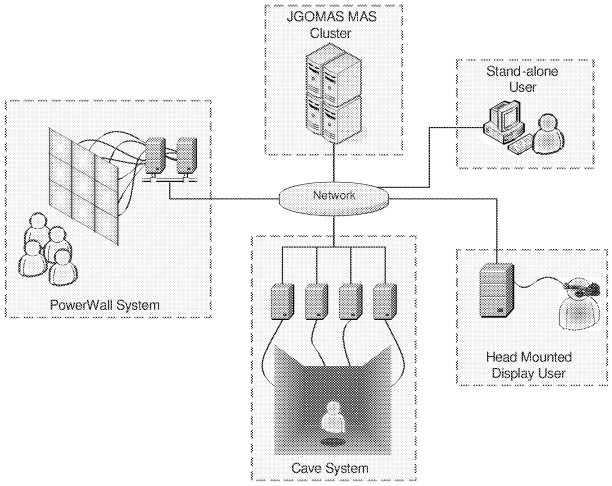


Figure 4: A JGOMAS scenario with some visualization configurations.



Figure 5: A practical Powerwall^{TM} done with four monitors and a *head mounted display* user.

In this way, JGOMAS is designed to have intelligent computer-controlled elements in 3D virtual environments, such as games, training and simulation applications. This means that JGOMAS is not either an artificial intelligence engine or a graphic engine. JGOMAS is simply a framework that allows to have agents in a virtual environment.

Due to the easy-to-use JGOMAS API, and the potentially attractive working scenario, it seems appropriate to use this system as a toolkit to introduce multi-agent systems and artificial intelligence techniques to senior computer engineering students.

JGOMAS package for download is available at http://jgomas.gti-ia.dsic.upv.es

ACKNOWLEDGMENTS

This work is partially supported by the PAID-06-07/3191, TIN2006-14630-C03-01 projects and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022. Furthermore, we thank Área de Sistemas de

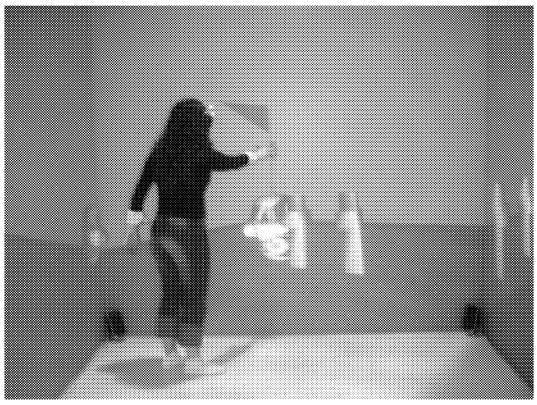


Figure 6: JGOMAS being used in a CAVE^{TM}.

Información y Comunicaciones (ASIC) at Universidad Politécnica de Valencia for providing the CAVE and its infrastructures.

References

Anastassakis, G.; Ritchings, T.; and Panayiotopoulos, T. 2001. Multi-agent systems as intelligent virtual environments. *In Advances in Artificial Intelligence, Joint German/Austrian Conference on AI - KI 2001.*

Andreoli, R.; Chiara, R. D.; Erra, U.; and Scarano, V. 2005. Interactive 3d environments by using videogame engines. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation (IV'05)*, 515–520. Washington, DC, USA: IEEE Computer Society.

Barella, A.; Carrascosa, C.; and Botti, V. 2006. Jgomas: game-oriented multi-agent system based on jade. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in Computer Entertainment technology*, 17. New York, NY, USA: ACM Press.

Barella, A.; Carrascosa, C.; and Botti, V. 2007. Agent architectures for intelligent virtual environments. In *IAT 2007. 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (pendiente de publicación).*

Bellifemine, F.; Poggi, A.; and Rimassa, G. 2001. Jade: a fipa2000 compliant agent development environment. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, 216–217. New York, NY, USA: ACM Press.

Bierbaum, A.; Just, C.; Hartling, P.; Meinert, K.; Baker, A.; and Cruz-Neira, C. 2001. Vr juggler: A virtual platform for virtual reality application development. In *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)*, 89. Washington, DC, USA: IEEE Computer Society.

FIPA. 2002. FIPA abstract architecture.technical report sc00001l. Technical report, Foundation for Intelligent Physical Agents. http://www.fipa.org.

Gorman, B.; Fredriksson, M.; and Humphrys, M. 2005. Qase - an integrated api for imitation and general ai research in commercial computer games. In *Proceedings of 7th ICCG: Artificial intelligence, animation, mobile, educational, and serious games (CGAMES), Angoulême, France.*

Klein, J. 2002. Breve: a 3d environment for the simulation of decentralized systems and artificial life. In *Proceedings of Artificial Life VIII, the 8th International Conferende on the Simulation and Synthesis of Living Systems.* The MIT Press.

Laird, J., and Duchi, J. 2000. Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. In *In M. Freed, editor, Simulating Human Agents: Papers from the AAAI 2000 Fall Symposium.*

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

Minar, N.; Burkhart, R.; Langton, C.; and Askenazi, M. 1996. The swarm simulation system, a toolkit for building multi-agent simulations.

Musse, S. R., and Thalmann, D. 2001. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics* 7(2):152–??

Norling, E. 2003. Capturing the quake player: using a bdi agent to model human behaviour. In *AAMAS '03: Proceedings of the second international joint conference on AAMAS*, 1080–1081. New York, NY, USA: ACM Press.

OSG. Openscenegraph: an open source high performance 3d graphics toolkit. Checked on February 8th, 2007.

Rickel, J., and Johnson, W. 1997. Steve: An animated pedagogical agent for procedural training in virtual environments. In *Proceedings of Animated Interface Agents: Making Them Intelligent.*, 71–76.

Thalmann, D., and Noser, H. 1999. Towards autonomous, perceptive, and intelligent virtual actors.

Ulicny, B., and Thalmann, D. 2001. Crowd simulation for interactive virtual environments and VR training systems. In *Computer Animation and Simulation '01*, 163–170.

Vosinakis, S.; Anastassakis, G.; and Panayiotopoulos, T. 1999. Diva: Distributed intelligent virtual agents. In *Extended abstract, presented at the Virtual Agents 99 workshop on Intelligent Virtual Agents, Univ. of Salford UK.*

VRJuggler. Vr juggler: a platform for virtual reality application development. Checked on February 8th, 2007.

Wooldridge, M., and Jennings, N. R. 1995. Intelligent agents: Theory and practice. *The Knowledge Engineering Review* 10(2):115–152.

AGENT BASED SERIOUS GAMING: SIMULATING THE PERCEPTION BIAS ON INNOVATION DIFFUSION

Marco Remondino, Marco Pironti, Roberto Schiesari
University of Turin
Italy
E-mail: remond@di.unito.it

ABSTRACT

The decisions to adopt innovations, in their widest acceptation, have been investigated using both international patterns and behavioural theories. In this work, an agent-based model is created to study the spreading of innovation in enterprises (namely, the adoption of Information Technology outsourcing). The paradigm of agent-based simulation makes it possible to capture human factors, along with technical ones. This in turn makes it possible to study the influence of perception, and the resulting bias. In particular, this work is focused on small and medium enterprises (SME), in which a restricted managing pool (sometimes just one person) decides whether to adopt a new technology or not, and bases the decision on individual perception only. The model is organized as a serious game, in which the user can change the parameters and see whan happens, so to make the resulting trend as similar as possible to the real (observed) one.

INTRODUCTION

The Theory of Planned Behaviour (TPB) by Ajzen (1991) - an extension of the Theory of Reasoned Action (Ajzen I., Fishbein M., 1980) - supports, the understanding of elements that describe an active and deliberate decisional process, set in a context characterized by constraints, such as social pressures and limited resources.

After a close look, it is a question of central moments in the life on an enterprise which are strictly connected with obtaining positions that guarantee a lasting competitive advantage, also thanks to the exploitation of new technologies (Bakos Y., Treacy M., 1986; Niederman F., King W.R., Sabherwal R., 1992) and, today, of ICTs (Ross J.W., Beath C.M., 2002; Rullani E., 2003).
However, such a great technological development does not seem to be matched by a parallel movement towards adopting and exploiting the opportunities it offers, through economic organizations, especially in SMEs.
Therefore, the goal is to outline the methods through which firm decision makers choose to start innovative investments and identify, at the same time, the criteria with which this choice is made.
A very wide topic, which was first discussed in the studies carried out by Rogers (1983). The author outlined five features of innovation (relative advantage, compatibility, complexity, observability and trialability), which influence its adoption, and consequently affect its circulation rate.

The spreading of certain innovations cannot be modelled simply by taking into account pure technical issues; sometimes the forecasts made with a statistical- or process-based model, which considers just the engineering or financial aspects (i.e.: production process, activity decomposition, scale economies, product life cycle and so on), prove inaccurate or even wrong. This is often the case with cellular telephones, air-cooling systems, personal computers and so on, fields in which some external influences have led to circulation results which are very different from the initial predictions obtained through classical prevision tools. The purpose of this work is to examine the circulation of an emerging technical innovation: using IT services in Application Service Provider (ASP) mode. The spreading of this technology should not be affected by phenomena like fashion – as it happened with cellular phones - and its adoption cannot be considered a status symbol, although, there are some other influences which cannot be taken into account by a traditional process-based or statistical simulator, i.e.: the effects of individual perception about savings, strategic impacts and so on.

IT OUTSOURCING

The ASP mode provides for accessing and using, through the Internet, applications such as, for instance, software designed for enterprise management (ERP), for e-commerce or for customer relations. Therefore, enterprises employing such are not to purchase a licence, but are only required to pay a monthly fee for the program, or to pay for the time they use the service. Furthermore, they are not required to buy the servers or to employ information technologists: both infrastructures and experts are already available in the ASP data processing centres.
The shift from the “Pure Client” to the “ASP” model has taken place on two different levels: a regular technological one and a level that may be defined as deeper, as it involves an “approach to the applications”.
From a merely technological point of view, the “Pure Client” model is based on personal computers that host the applications: each personal computer, client, has an application of its own, which is located and is executed within it. The development of the “Pure Client” model leads to the “Client – Server” model: applications are located on a server within the client enterprise and are executed on the client. The last model is the “ASP” model, in which

applications are transferred outside the company and placed with an Application Service Provider.

The first remark to be made is that the switch from a model to the other is not to be deemed of an evolutionary nature; preference for one or for the other depends on the specific requirements of each individual enterprise, even though the "Pure Client" model has almost completely disappeared from the market.
The ASP model provides for the existence of three crucial elements: the web, the Data Centre and the application. The Data Centre is the real "added value" that the ASP model is able to provide to its clients, because, thanks to the services and technology existing in a Data Centre, the Application Provider may or may not guarantee the high quality of its performances.
Access to applications by means of an ASP has many advantages and a few disadvantages. The main advantage as to do with the reduction in costs and with the considerable savings in information management, chiefly due to the fact that it is not necessary to purchase or update specific hardware and software. The only costs that an enterprise is to bear are those related to client acquisition, the ASP-related fee and the Internet connection. The possibility to process one's data with the support of a public facility such as the Internet is another advantage, as it eliminates all the costs to be born in order to implement a network of one's own. Each enabled user is able to access the application from a remote position, without having to afford expensive updates to improve his/her technological environment.
The enterprises who gain the greatest advantages from this technology are those distributed throughout the territory with many branches. The initial installation cost, which is integrated by the monthly fee, fully justifies the use of an ASP application rather than purchasing software to be independently developed and managed and implementing a network of one's own. The problems related to licences, technical support and installation, to versions updates, to the compatibility with other applications, to the operating system, to the constant developments required in order to catch up with technological changes and the possible need to train highly qualified staff are all handled by the Application Service Provider, who is only paid an annual fee. The fee may involve several different rates, among which we recall flat rates, pay-per-use rates, etc.
The ASP model also displays some disadvantages: first of all, having to switch from an in-house management system and local access to one's data to third-party management and access through the Internet involves giving up direct control over applications and information. Secondly, one needs to take into account the fact that not all application packages are at the moment ASP oriented. It is therefore necessary to take into account the best time to possibly switch to an ASP solution.
Thirdly, some difficulties exist in understanding the proposals and in entering into agreements that may be comprehensive with regards to the services required for the supply. This is basically due to the ASP service suppliers, who mould their proposals depending on the market trend. Above all, it is very difficult to identify which and how many may be the possible critical points of an ASP service, as the critical points are closely related to the type of service offered and to the specific needs of each client. An enterprise that wishes to resort to an ASP is to take into account these new technological opportunities and the initial difficulties involved in creating from the very beginning a good relationship with the new partner. Therefore, the advantages are to be analysed in connection with the consequent strategic and operating risks.
Besides, some not-technological shortcomings also exist, which involve psychology, legal aspects and trust towards the means used. Of course, taking one's information system to an Application and Service Provider represents a big step for an enterprise and involves a certain inclination to externalise certain functions of the company information system, which may also be crucial ones, such as CRM (Customer Relationship Management) or ERP (Enterprise Resource Planning), and many Italian enterprises are not inclined to take this type of approach.
As regards the legal shortcomings, the one which certainly stands out has to do with the management of privacy (governed in Italy by law no. 675/96). Indeed, an enterprise willing to entrust one's data to an Application and Service Provider may have to face the difficulties resulting from the need to entrust the ASP with the management of sensitive data, of whose integrity the ASP itself is to legally answer for.
Lastly, in addition to the psychological inclination to externalise one's applications, the crucial factor an compnay is to take into account in choosing an ASP must be based on trust towards the ASP itself, to whom the enterprise is entrusting the management of part of its business.
This work has been carried out by considering SMEs as clients, in that large or multinational enterprises are unlikely to externalise their IT services, save towards suppliers on which they are able to exert significant (equity or non–equity) control or whatever inside their spam of control.

SME PERCEPTION PROBLEMS

Proper perception of the benefits granted by the adoption of an innovation is a crucial element in order to understand the circulation potential and the growth rate for that innovation.

By following a strictly rational approach, it often proves difficult to understand the reasons for development difficulties and also for the delays in the circulations of innovative goods and services.

Its may therefore prove useful to analyse the reasons that may distort perception of the advantages resulting from the outsourcing of IT services in the ASP mode by SMEs, which are on the other hand the parties which are likely to benefit from the greatest advantages provided by the broadening in the offer of such services.

The reduction in initial investments, the easier and less onerous opportunity to benefit from the improvement of technology, as well as the greater certainty about costs and their manifestation in time should be strong motivations for the spreading of such services among smaller enterprises, once the infrastructure prerequisites related to the performance and reliability of data transmission networks and to the existence of an offer market capable of creating competition have been established.

However, we should take into account the fact that Italian SMEs are late in IT-related expenses and investments.

Indeed, a recent research published in Italy (Confcommercio, 2007) stresses how 53% of IT-related expenses and investments come from enterprises with over 500 employees, or even 76%, if we also take into account enterprises with over 100 employees.

IT-related expenses and investments are therefore restricted to a limited number of larger enterprises than the over 5 million Italian enterprises with a lower number of employees which have invested little or no money, whereas it is known that the ICT-productivity growth ratio is extremely high. This ratio is obvious, not only in industry but also in trade and services. The European committee itself has requested that, at a micro-enterprise level, the element represented by shared organisations be developed.

SMEs' requirement to benefit from IT services that are fit to face existing competition challenges is therefore a priority for their very existence.

However, the growth rate in resorting to outsourcing by SMEs has proved lower than expectations up to very little time ago.

In order to try and understand the reasons that might distort the perception of outsourcing advantages, it may be helpful to mention the fear experienced by smaller enterprises of ending up depending from opportunistic behaviours that might be adopted by the suppliers of such services, and which smaller enterprises fear they may be more vulnerable to, owing to their lower bargaining power and more limited understanding of the alternatives offered by the markets (information asymmetries).

Indeed, Williamson stresses that opportunistic behaviours represent one of the reasons for market failure that cause enterprises to internalize the activity, in order to gain some kind of insurance (Williamson, 1985). The existence of such behaviours causes transaction costs to increase and, in the event of a high level of asset specificity, encourages enterprises to maintain such activities within the company or to internalise them.

This issue is related to the problem of accurately assessing switching costs to be borne in the event that the outsourcing IT services supplier is changed, which are even more difficult to evaluate by an enterprise that lacks specific competence.

The evolutionist theory (Nelson and Winter, 1982) offers a contribution to the analysis. By analysing enterprises as agents designed to acquire understanding, we find that enterprises tend to internalise and understanding or know-how based on their own routine. This know-how, however, lead to the creation of growth trends which depend on what has been learnt in the past (path dependency).

Another problem which significantly hinders outsourcing is related to the confidentiality of information and data contained in the enterprise information system and hence depends on the reliability of the supplier of the service involved in safeguarding the data themselves, both during the course of the cooperation agreement and in the event of termination of services, also if this is related to controversies that may have arisen between the enterprise and the supplier.

A SME may perceive this risk both because it lacks the resources and skills required to asses the reliability of the systems and process implemented by the privacy supplier and because of the belief, especially strong in Italy, as to the non-existence of a prompt and efficient legal protection in the event that one's privacy rights are infringed.

Indeed, information system data are perceived as crucial for the enterprise and their disclosure represents a danger for the survival of the enterprise itself.

The reliability of suppliers over an extended period of time therefore becomes a crucial factor, which is not easy to asses, owing to the many aspects of potential analysis investing not only economic and financial solidity, but also elements related to the ethical nature of dealers.

The fact that many suppliers have only recently appeared on the market and that in the field in question there has been a high failure rate with regards to the enterprises involved, creates significant trust problems.

Also in view of SMEs resistance to change, we should not underestimate the difficulties involved in negotiating outsourcing agreements, which display complexity profiles related to technological aspects, to the rating of service performance and to the possible need to adjust to the requirements of minor enterprises, which often lack adequate strategic and operating planning tools.

A contribution to the assessment of the elements that may distort SMEs' perception also comes from the psychological studies used, amongst others, by behavioural finance and marketing.

Information that is not easily understood or employed by minor enterprises in connection with the choice to be made may represent a greater restraint than the lack of information altogether, which partly contradicts the theories based on rational processes.

The bounded rationality characterising the above mentioend processes is amplified by formalised, but above all informal, relations among the parties involved, within an enterprise, but also among parties belonging to different organisations.

The "confidence" and "trust" problems characterising many SMEs' approach to IT outsourcing represent a restraint to horizontal communication between enterprises, which on the other hand could represent a major innovation propagation factor.

This problem may also involve enterprises that have adopted the innovation but are not fully confident and aware about its benefits and therefore do not officially support it within their community.

Therefore, within settings in which SMEs experience strong difficulties in controlling information, understanding possible threats coming from opportunistic behaviours and assessing the reliability degree of their interlocutors, when faced with a possible radical choice in terms of outsourcing a strategic function such as ICT, they are strongly inclined towards a conservative approach, and maintain this attitude

also in the face of evidence supporting the advantages of a change.

Some of the factors that hinder the perception by SMEs of the advantages provided by innovation may however be eased in a more advanced stage of development of this sector and by behaviours and strategies by suppliers, also of a communication nature, that may be more specifically designed to meet SMEs' requirements. Based on these considerations, we have decided to contextualize the application of the model to SMEs in Italy, in spite of the significance of same compared to the whole Italian production sector (SMEs represent 97% of enterprises as a whole).

AGENT-BASED MODELLING AND HUMAN FACTOR

Agent-based Modelling is one of most interesting and advanced approaches for simulating a complex system: in a social context, the single parts and the whole are often very hard to describe in detail. Besides, there are agent-based formalisms which allow studying the emergence of social behaviour through the creation and study of models, known as artificial societies. Thanks to the ever increasing computational power, it has been possible to use such models to create software, based on intelligent agents, whose aggregate behaviour is complex and difficult to predict, and which can be used in open and distributed systems.

In Franklin and Graesser (1997) we read that:

An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

Another very general, yet comprehensive one is provided by Jennings (1996):

...the term [agent] is usually applied to describe self-contained programs which can control their own actions based on their perceptions of their operating environment.

Agents themselves have traditionally been categorized as one of the following types (Woolridge and Jennings, 1995):

- Reactive
- Collaborative/Deliberative
- Hybrid

When designing any agent-based system, it is important to determine how sophisticated the agents' reasoning will be. Reactive agents simply retrieve pre-set behaviours similar to reflexes, without maintaining any internal state. On the other hand, deliberative agents behave more like they are thinking, by searching through a space of behaviours, maintaining internal state, and predicting the effect of actions. Although the line between reactive and deliberative agents can be somewhat blurry, an agent with no internal state is certainly reactive, and one that bases its actions on the predicted actions of other agents is deliberative.

The agents used in this paper are reactive, but organized in the form of a MAS (Multi Agent System), which can be thought of as a group of interacting agents working together or communicating among each other. To maximize the efficiency of the system, each agent must be able to reason about other agents' actions in addition to its own. A dynamic and unpredictable environment creates a need for an agent to employ flexible strategies. The more flexible the strategies however, the more difficult it becomes to predict what the other agents are going to do. For this reason, coordination mechanisms have been developed to help the agents interact when performing complex actions requiring teamwork. These mechanisms must ensure that the plans of individual agents do not conflict, while guiding the agents in pursuit of the system goals.

Many simulation paradigms exist; agent-based simulation is probably the one that best captures the human factor behind decisions. This is because the model is not organized with explicit equations, but is made up of many different entities with their own behaviour. The macro results emerge naturally through the interaction of these micro behaviours and are often more than the algebraic sum of them. This is why this paradigm is optimal for the purposes of modelling complex systems and of capturing the human factor. In this work the focus is on the individual perception in human behaviour, when facing the decision of adopt a new technology or an innovation. Especially in SMEs, the decision is in the hands of a small managing pool, or even of one person only. Even if the economic value of an innovation is high, sometimes it will not be adopted in an enterprise, since it is not correctly perceived. This seems to be the case of IT outsourcing in Italy, as we will discuss later in the article.

THE SIMULATION FRAMEWORK

A computational framework is built, by using pure Java coding. It consists of a user interface, in which the "player" can set the initial simulation values. According to internal rules, the model computes these values, developing them in order to give final aggregate results. The main purpose is that of analyzing how human (individual) perception can bias real evidences.

A number n of macro-agents is built, representing the SMEs; each of those is composed of a random number of agents (where the range can be set with min/max or can obey to a normal distribution, given an average). This represent the managing pool, that in the SMEs is usually composed of few persons having the power to decid wether to adopt an innovation or not.

The agents belonging to the same enterprise are connected by a social network; the links feature a random weight, with a modifiable threshold. The network has the purpose of transmitting the information about the technology, that's coded through a function describing the perception of the real phenomena. Some of these agents can of course be connected to others, belonging to different enterprises, so spreading their opinion about the new technology over the

network. The perception is surely derived from the real saving allowed by the new technology, but not only that; there could be a bias, representing other factors, like trustfulness towards the company that supplies the IT outsurcing facilities, personal inclination towards IT and so on. The agents spread their own perception on the network, to the other agents connected to them and this will be modified according to the weight of the specific individual link. Each macro-agent (the enterprise) also features a CIO manager, which is the IT expert for that enterprise. Some enterprises could lack a CIO, and in that case, this agent represents an external consultant, which could also be shared by two or more enterprises, thus constituting a bridge among different macro-agents and, indirectly, among the agents involved.

A second link among the macro-agents is represented by social networks among the agents belonging to different enterprises. We can metaphorically depict these links as friendship, job or family relationship and so forth.

One important aspect in the model is that only agents are directly connected, not the enterprises; agents represent people, so they can be connected somehow, while enterprises can be considered "connected" among them if and only if the people belonging to them know each other.

Since the focus is how social relations and individual perception can influence the spreading of IT outsurcing in SMEs, the model, through repeated execution, can show the diffusion over time, by changing just one parameter at a time, while leaving the others unchanged (*ceteris paribus*).

The Parameters in the Model

Here follows a list of the parameters used in the simulation:

1) Total number of enterprises
2) Average number of agents per enterprise (managing pool)
3) Percentage of enterprises already reached by the new technology (if higher than zero)
4) Costs for the innovation (una tantum and variable cost)
5) Average saving allowed by the new technology
6) Perception BIAS (1:double perception, 0:perfect perception, -1:inverse perception, 2:random perception)
7) Satisfaction Threshold (0: always satisfied, 1: satisfied only if perfect conditions occur)
8) Requested majority to adopt the IT outsurcing in the enterprise (percentage, E.g.: 51%)

These parameters can be changed by the user at the beginning of the simulation, in order to mimic, as much as possible, the real world. Some parameters are very general (such as "Average number of agents per enterprise", "Costs for the installation" and so on). They should be set accorging to the values observed on samples taken in the market where we want to set the study.

Other parameters, such as "Satisfaction Threshold" and "Perception BIAS" are more difficult to determine; they are derived from the Cognitive Sciences and define how a single person perceives the environment and the innovations. It is possible to assume that almost nobody has a perfect perception, since this would be similar to the "Homo oeconomicus" assumption in the Neoclassical Economics, i.e.: a stylized and ideal type of the instrumentally rational individual. More realistically, people have a perception BIAS, that's the general tendency of a person to view a situation in a particular light based upon the method of socialization that they were brought up with, their personal experience, their knowledge and so on.

A technician could have almost a perfect perception about how convenient a technical innovation can be, but an average person can't.
Besides, not all the individuals are satisfied about something at the same threshold; somebody could be happy to save 100€ per year, while others don't even bother for that amount of money.
That's why these two parameters must be set according to some general observations and common sense and are the main focus of the present work.

Model Iterations and Steps

In this subsection an overview about how the model works is given. Here follows the sequence of steps performed during the simulation:

1) The first step is the adoption of the new technology by a number of enterprises
2) The new technology has some installation costs and variable costs, and brings a positive or negative saving compared to historical data
3) These values will be perceived by the agents belonging to the enterprise, but filtered by their personal BIAS (which takes into account different income, ecological awareness and so on)
4) Each agent randomly spreads through the social network her/his own perception, and the agents linked with her/him will form their own opinion, according to the average of the received opinions
5) According to the "satisfaction threshold", the agents will individually decide whether to adopt or not the new technology
6) Once all the agents in a enterprise have formed their own opinion, according to the requested majority, the SME will adopt the new technology or not yet. The next step will be a loop from point number 2).

There is not a metaphorical correspondence between the steps and time units. It's possible to realistically think at heach step as one month time.

RESULTS AND COMMENTS

Several experiments have been run on the model, by changing the parameters in order to mimic different scenarios. Two are the parameters studied in this work that affect the diffusion over time: a different connection level and a different perception level. In particular, since the study deals with a localized market (namely, Italian one), sooner or later there will be a saturation level, that's a threshold

beyond which it's not possible to go (i.e.: a limited number of agents can be reached by an innovation, since a limited number of enterprises exist in a market). So we aspect an higly connected network to generally increase the diffusion speed over time. We can think to horizontal connection, like clubs of managers, as an example. In those situations, people will spread their opinions among others working in the same field or, at least, with the same job and position, and this will make their message more effective.

The saturation threshold is represented by the parameter called "Total number of enterprises" that multiplied per "Average number of agents per enterprise" gives a good approximation of the total number of agents in the simulation (not exact, due to sampling effects). On the graphs, the total adoptions versus time step are shown.
A total of 500.000 macro agents (enterprises) are set in the model and an average of 5 is the managing pool hypnotized. In order to simulate a realistic situation, 20% of enterprises are considered as already reached by the new technology at time 0. The costs for the innovation and the average saving allowed by the new technology are desumed by interviews made with a sample of Italian SMEs. The requested majority is set at 51%, since we imagine that all the members of the managing pool have the same importance in the enterprise.

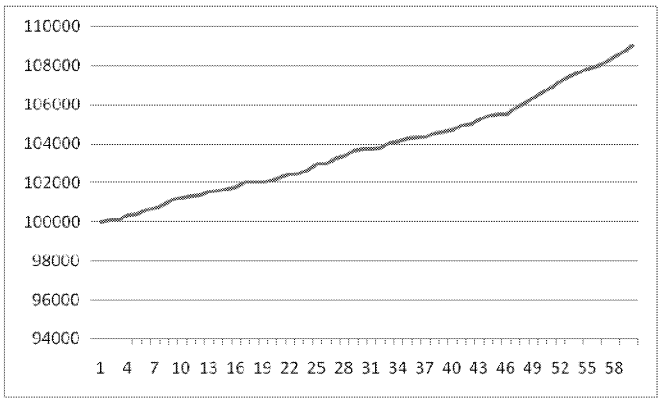


Figure 1: low perception and low connection

In figure 1, the results with a low perception index and a loosely connected network are shown. After 12 simulation steps (the first year, according to the model), from the initial 100.000 enterprises that already adopted IT outsourcing, we have that 1.399 moved to the new technology. This brings to a 1,4% increase. The second year brings an even worse figure, that's an increase of 1,1%. Third and forth years bring an increase of about 1,2% and 1,6% respectively, while the fifth year has much a better figure compared to the previous ones, resulting in a 2,5% increase.
The slow increase, even if the savings allowed by stepping to the new technology, are to be explained taking into account the low perception index. The bias that the perception introduces in the model is enough to make a very good deal to look like an average or poor one to many agents in the model. Besides, the loosely connected network prevents the agents to spread the good news about their savings to many others and so, after 5 simulated years, less than 10% new enterprises stepped to the new technology, from the original 100.000.

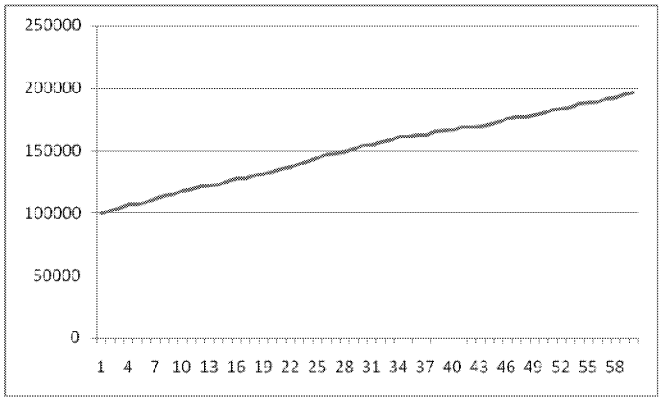


Figure 2: low perception and high connection

In figure 2, the same values were used as in the first experiment, with the exception for the level of connection inside the network, and its topology. Now the network is an highly connected one. This is probably an unrealistic scenario, when referred to people belonging to different enterprises spread throughout a whole country. That would mean that almost every manager knows all the other ones. The perception in this experiment is still very low, meaning that even a big advantage is not perceived as such.

The results are now quite different than before. The first year brings an increase of 21%, stepping from 100.000 to 121.774 enterprises that adopted IT outsourcing. This is a very high figure, considering that the perception level is still very low.

The following simulated years also have a very good trend, even if slightly decreasing compared to the first experiment. In fact, the second year brings an increase of 15%, the third of 12%, the forth of 8% and the last one of 10%, always referring to the preceeding year.

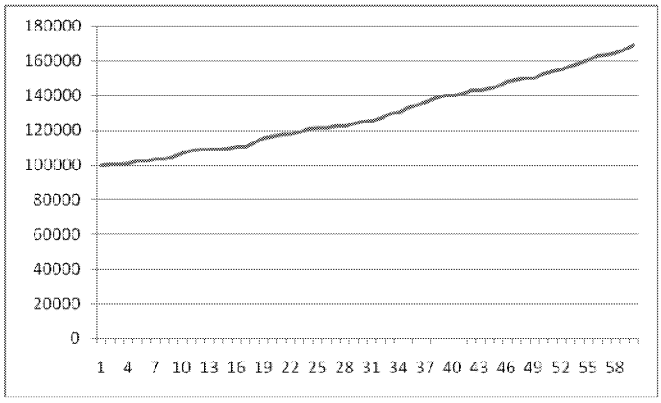


Figure 3: high perception and low connection

The fact that in experiment 1, even if with much lower increase rates, the pattern revealed by analysis of variance is a growing one, while in experiment 2 is a decreasing one can be due to the fact that now, having an highly connected network, the agents are reached right from the start by others's opinions, this forming their own. When the simulation goes on, there are less and less agents still not reached by the news about the innovation. Since the

perception is low, many agents decide not to adopt the technology even if reached by the news, and so even after 5 simulated years, a lot of enterprises are still deciding to stay with the old technology.

The third experiment is about a loosly connected network, as the one in the first experiment, but this time with an almost perfect perception. This means that the few managers reached by the word of mouth given by others, will for sure perceive the correct advantage brought in by the new technology. The results are shown in figure 3.

Thanks to the high perception, we have the following increase rates: 11% (second vs first year), and then 12-14% circa per year, for the following periods. After 5 simulated years, a total of 169.129 enterprises have stepped to the new technology, even if the network is loosely connected, meaning that a lot of agents are never reached by the news about technology. This time the trend is again an increasing one, meaning that when more agents are reached, there is in turn an higher probability to reach other ones, and so on.The last experiment is carried on with high perception and higly connected network; the results are shown in figure 4.

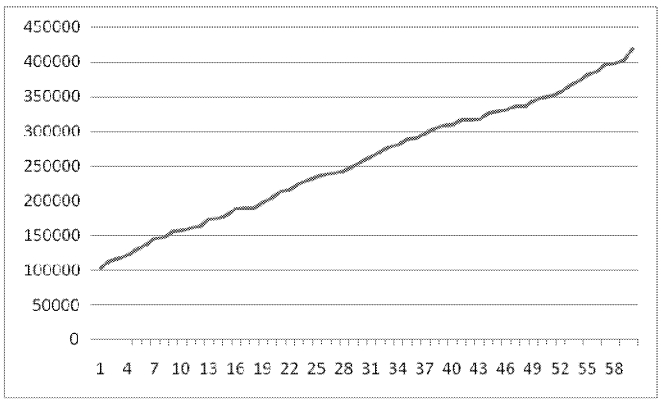


Figure 4: high perception and high connection

Now the spreding of the innovation is pervasive; after one year, we already have an unrealistic 60% increase compared to the time zero, and the trend continues in the following simulated years, so that, after 60 months, 418.274 out of 500.000 enterprises are reached by the new technology.

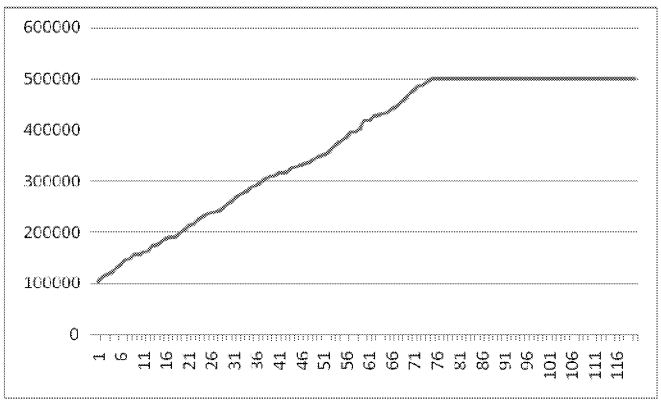


Figure 5: high perception and high connection, 120 steps

In order to further validate the model, an experiment with the same parameters as the last one is carried on, but with a different time limit. This time the simulation is run for 120 months (10 simulated years), in order to see when all the enterprises in the model are reached. The results are shown in figure 5.

As foreseen, the previous trend continues, and all the enterprises are reached at the 75th simulation step. After that, the scenario stands still, since no other enterprises can be reached (a closed marked is simulated here).

CONCLUSION

In the present work, the role of perception is investigated when studying the study of innovation diffusion. In particular, the IT outsurcing is an innovation that's quite crucial in enterprises and is used as an example to empirically validate the presented framework. The studied market is that of SMEs, where a small managing pool can decide whether to adopt an innovation or not. An agent based model is implemented, since it allows to exploit the human factor behind social phenomena. A set of macro agents, representing the enterprises, is built, and a set of sub-agents (the managing pool) for each enterprise is created. Five different experiments are carried on, varying the level of perception and of connection along the social network connecting the agents. Validation for the results is carried on by using tools coming from data-mining. The results obtained from the model are clear: an high perception of the advantages given by an innovation is very important in the spreading mechanism, and so is a good coonection level among the agents. By comparing the real figures (coming from AITech-Assinform/NetConsultin) about Italian market to the experiments shown in this article, we see that they look like the ones obtained in the first experiment (figure 2). In fact, the delta in IT outsurcing adoption from 2003 to 2004 is about 1%, stepping to 1.6% for 2004 to 2005, and to over 2% from 2005 to 2006. This proves to be a market in which there is a low perception of the benefits and a loosely connected network connecting the various enterprises.

BIBLIOGRAPHY

Ajzen, I. (1991) "The Theory of Planned Behaviour", Organizational Behaviour and Human Decision Processes, n° 2.

Ajzen, I., Fishbein M. (1980) "Understanding Attitudes and Predicting Social Behaviour" Prentice- Hall, Englewood Cliffs.

Bakos, Y. Treacy, M. (1986) "Information technology and corporate strategy: a research perspective", MIS Quarterly.

Confcommercio, "Manifesto per l'innovazione", Marzo 2007

Cragg, P.B. King, M. (1993) "Small-firm computing: motivators and inhibitors", MIS Quarterly, March.

Davis, F.D. (1986) "A technology acceptance model for empirically testing new end-user information systems: theory and results", Doctoral dissertation, Sloan School of Management, Massachusetts Institute of Technology.

DeLone, W.H. (1988) "Determinants of success for computer usage in small business", MIS Quarterly, March.

Edwards W. , "Conservatism in human information processing" , Formal Representation of Human Judgment, John Wiley and sons (1968)

Ellsberg D., "Risk, Ambiguity and the Savage Axioms" , Quarterly Journal of Economics (1961)

Franklin, S., & Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, Proceedings of the Agent Theories, Architectures, and Languages Workshop, Berlin (pp. 193-206). Springer Verlag.

Jennings, N. R. (1996). Software agents. IEEE Review (pp. 17-20)

Laumann E.O. et al. 1978, "Community Structure of Interorganizational Linkages", Annual Review of Sociology, 4, pp 455-484

Mataric, M. (1995). Issues and approaches in the design of collective autonomous agents. Robotics and Autonomous Systems 16 (pp.321-331)

Mathieson, K. (1991) "Predicting user intentions: comparing the technology acceptance model with the theory of planned behaviour", Information Systems Research.

Moore, G.C. Benbasat, I. (1991) "Development of an instrument to measure the perceptions of adopting an information technology innovation", Information Systems Research.

Nelson R.R., Winter S.G. "An Evolutionary Theory of Economic change" Harvard University Press. Cambridge (1982)

Niederman, F. Brancheau, J.C. Wetherbe, J.C.(1992) "Information systems management issues for the 1990s", MIS Quarterly, December

Raymond, L. (1988) "The impact of computer training on the attitudes and usage behaviour of small business managers", Journal of Small Business Management.

Rogers, E.M. (1983) Circulation of innovations" Free Press, New York,.

Ross, J.W. Beath, C.M. (2002) "Beyond the business case: new approaches to IT investment", MIT Sloan Management Review, Winter 2002.

Rullani, E. (2003) "Lavorare in rete. L'uso strategico delle ICT per riposizionare la piccola impresa e il made in Italy nel nuovo ambiente competitivo", Economia & Management, n. 4.

Shefrin H., "Beyond Greed and Fear: Understanding Behavioural Finance and the Psychology of Investing", Oxford University Press (2002)

Twersky A., Fox C. "Ambiguity Aversion and Comparative Ignorance" Quarterly Journal of Economics 1995

Williamson O.E. "The Economic Institutions of Capitalism". Free Press. New York 1985

Woolridge, M., & Jennings, N. (1995). Intelligent agents: Theory and practice. Knowledge Engineering Review 10(2) pp.115-152

Mshindi – an Awalé winner

Vincent de Boer and Leon Rothkrantz
Man-Machine Interaction Group
Delft University of Technology
2628 CD, Delft,
The Netherlands
E-mail: L.J.M.Rothkrantz@tudelft.nl

KEYWORDS
Awale, min-max algorithm, Tjip challenge.

ABSTRACT

Mshindi is a program developed for the Tjip Challenge, an open contest to write an Awale program in Java or C# that beats the other participants within time and space constraints. The programs were given two minutes for the whole match and received an extra second for each move. Also the source code and any databases couldn't be larger than 1MB, making it impossible to pre-compute all the game states or even use a very large opening book or endgame database.
The design of the program based on the min-max algorithm with alpha-beta pruning will be discussed. The final scores of the competition will be presented.

INTRODUCTION

The game Awale and its rules

Awalé is an African game that is known worldwide under hundreds of different names (e.g. Wari, Oware, Abapa and Awari) with sometimes slight variations in the rules [1,2,3,4]. The board consists of 12 pits that are 'connected' counter-clockwise in a circle. At the start of the game each pit contains 4 seeds. It is a two player game: Each player owns the 6 pits on his side and can make moves by taking the seeds from one of his pits and distribute them one by one in the consecutive pits. When the pit that is being played contains 12 or more seeds, the pit where the seeds came from is skipped, so it is always empty after the move. If a move ends in one of the opponent's pits and this pit contains 2 or 3 seeds after the move, then these seeds are captured. If the previous pit also contains 2 or 3 seeds they are also captured, until either the player's own side is reached or a pit that contains less than 2 or more than 3 seeds.

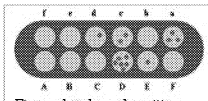


Figure 1: a board position

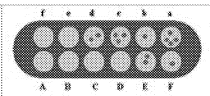


Figure 2: moved pit "D", ends in "d"

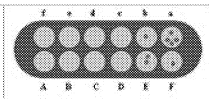


Figure 3: completing move: captured seeds in "d" and "c"

An additional rule is that you are not allowed to make a move which would leave the opponent without seeds unless you have no other option. If this happens and the opponent has no seeds to move, all seeds left on the board are given to the owner of the pits they are in.

A common technique to capture seeds is to save so many seeds in one pit that you can make a circle and pass opponent pits fields twice. If the opponent pits contained beforehand at most 1 seed, then after this move they all contain two or three seeds and are captured.

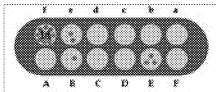


Figure 4: pit "f" contains 14 seeds

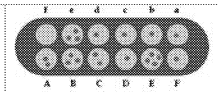


Figure 5: pit "f" is played, move ends when it arrives in pit "C" for the second time

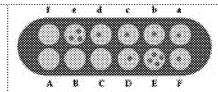


Figure 6: the seven seeds from pit "C", "B" and "A" are captured

The contest

TJIP BV is a software company located at Delft in the Netherlands. Every year the Company organizes a tournament the TJIP Challenge for gameprgammers who like to compete with each other. This year 2008 the the participants were invited to develop an AI software system , which was able to play the game Awale. Next year the Transport game Tycoon has been selected..
The current game Mshindi is a program developed for the Tjip Challenge, an open contest to write an Awale program in Java or C# that beats the other participants within time and space constraints. The programs were given two minutes for the whole match and received an extra second for each move. Also the source code and any databases couldn't be larger than 1MB, making it impossible to pre-compute all the game states or even use a very large opening book or endgame database.
The Oware Society (OWS) [7], is an organization set up to promote the playing of Oware in all its forms worldwide. In the past years they linked many groups together and organized and supported International Tournaments.

Related work

In 2002 J.Romein and H. Bal proved that Awari has been solved [8,9]. With perfect play, the game ends in a draw. They solved the game by searching the entire state space on a parallel computer with 144 processors. The state space contains 889.063.398.406 positions and was searched using retrograde analysis.
Now the game has been solved it is still,challenging to play on computers with limited memory and processor capacity.

In 2002 [10] Lincke developed a system called Marvin which proves to be a strong player. At this moment many commercial systems are available.
Our system fits in the tradition of games to be played on an average computer [5,6,13].

Design of the Mshindi-system

Awale is a highly mathematical game and a perfect game for brute-force searching techniques. Our goal was to make a program that would simply look further ahead than any other participant within these time constraints. For our program, Mshindi, we used Java as programming language.

Min-max/Alpha-beta pruning

The basis of our program is the min-max algorithm with alpha-beta pruning [11,12]. This means we use a depth-first search to find all the possible move sequences till a certain depth and return an evaluation of the board when this depth is reached. One player makes the moves that lead to the best reachable evaluation function in the leaves of the tree, while the other makes the moves that lead to the worst reachable leaves. Because this considers all possible combinations of moves, it has an exponential complexity in the number of moves.
Alpha-beta pruning is a technique to stop the search when you can prove that the result of the tree can't be relevant to the value of the root. With good move ordering this can approximately double the search depth.

Evaluation function

Many of the heuristics in Awalé somehow involve the "moves in hand" principle; the number of moves you can safely make on your own half before you have to give seeds to your opponent. The player who has this advantage can force the opponent to play a move he would rather not make. Many other principles, like making a large stack of seeds will usually only pay off when combined with an advantage in moves in hand. Because we saw no efficient way to calculate the moves in hand we decided to keep the evaluation function simple and count on the extra search depth we could reach that way.

Our evaluation function was therefore initially simply [our score]-[opponent score]. We later made it slightly more complicated by adding for a capture instead of the number of seeds, 100 times the number of seeds plus the remaining search depth after that node. This gave a higher value to capturing seeds early and solved problems where our program would delay taking seeds when he knew he could get the same seeds later as well. Here could arise a problem as a repetition of the board situation forces the game to finish. It was more efficient to change the evaluation function than to check all previous board positions in every node.

It also helped solving the search horizon problem where a capture at the end of the search may seem favorable because the negative positional effect of the capture is not seen. We now give a positive score to getting up two points and losing them later and a negative score to the opposite. This makes sense because capturing often makes your position worse so the one who made the last capture is most likely to have the worst position, even if the effects of this fall outside the search range.

Bitboard

To search the game space for the best move there are 2 actions that must be performed millions of times: determine the legal moves for a given player in a given position and perform one of these legal moves. The board representation should be chosen so that these actions can be performed easily.

We represented the board as a single 64-bits number. This gives you 5 bits for each of the 12 pits and 4 spare bits. Since 5 bits can only represent numbers up to 31 and there are initially 48 seeds in the game that might all end up in the same pit these 5 bits are not always enough. However you do know that only 1 pit can contain more than 24 seeds. You can therefore use the additional 4 bits to represent extra seeds belonging to the pit with more than 24 seeds. Whenever a pit contains 28 or more seeds, 4 seeds are subtracted and added to the 4-bit extra store. This allows you to store 27+15*4= 87 seeds in one pit, which is more than enough. In fact, only a 3-bit store is needed. The bit that is left now, can be used to indicate whether the last move was legal or not. We use the sign bit for this, so that if a move results in a negative number, the previous move was illegal. The captured stones can't be represented in the same number, but since they don't affect the way a move is made nor whether a move is legal this doesn't affect the speed with which moves can be made.

In binary representation the starting position, with 4 seeds in each pit, for example would look like:
s**000**0010000100001000010000100001000010000100001000010
0001000010000100
where the most right 5 bits correspond to pit 0, the next 5 to pit 1 etc. The bit marked with s is the sign bit, which is not used for legal board positions and the 3 bold bits are the extra storage.

Doing moves

The action of doing a move is split up in 3 different parts. First the seeds are taken from the "move" pit and redistributed one by one over the other pits; the possibly captured seeds are removed from the board and counted. The last step is moving seeds that could cause an overflow in later steps to the extra store.

The redistributions of seeds can now be done with a single addition. The difference is a constant number, depending on the pit from which the move is done and the number of seeds in that square. This gives 12*48 different longs that can be pre-computed.

These numbers can be visualized as board positions with in each pit the number of seeds that must be added and in the starting pit a negative number.

0	0	0	1	1	1
0	0	0	-5	1	1

This board position would correspond to moving 5 seeds from pit 9

The code for redistributing seeds is now very efficient:

```
//get the number of seeds in pit "move"
byte seeds = (byte) ((board>>5* move) & 31);
//return negative number for illegal move
if(seeds==0) return -1;
//add 4*store (bit 60,61,62) to the number of seeds
if(seeds>=24) seeds+= (board>>58)&28;
//redistribute seeds
board += moves[move][seeds];
```

This results in only 3 operations for pits with less than 24 seeds (which is the large majority of the moves) and 6 for pits with more than 24 seeds.

The capturing of stones depends on the rest of the board, so it's not feasible to pre-compute this. If the move ends in opponent's territory, a check on possible captures must be made. Checking whether a field is equal to 2 (or 3) can be done by filtering only the bits corresponding to the pit you want to check and comparing it to a long that represents two seeds in that pit (also pre-computed). Emptying a pit is just a single operation: a bitwise "and" with a pre-computed number having 1's for all bits except those representing the pit you want to empty. Note that it is harder to empty pits with more than 24 seeds since part of them may be in the extra storage, but that only happens in the redistribution phase so that effect can be included in the pre-computation of the moves.

A check whether field "end" equals 2 would be:

```
if((board&masks[end]) == two[end])
```

and emptying that pit if this is true:

```
board &= ~masks[end];
```

masks[3] for example would be (in binary representation):
0000000000000000000000000000000000000000001111
1000000000000000
and two[3]:
0000000000000000000000000000000000000000000100
0000000000000000

The last, and most painful, part of a move is to check each pit on overflow in order to possibly move seeds from the largest pit to the extra storage. Fortunately, it's not necessary to check this every move. If a 5-bit number is larger than 24, the 2 most significant bits must be 1, so

```
((board&bits8)<<1) & (board&bits16)
```

//bits 8 has only the 8-bits of each pit on 1, and bits16 has only the 16-bits of each pit on 1.

bits8:
0000010000100001000010000100001000010000100001000
0000000100001000

will return a number larger than 0 if and only if one of the pits has over 24 seeds. If this is not the case, no check is needed. If it is the case, all 12 fields are checked and if any is larger or equal to 28, 4 is subtracted (since the 4-bit must necessarily be 1 for 5-bits numbers larger than or equal to 28, this can be a simple bit operation rather than a real subtraction) and 1 is added to the store. This is an addition with a pre-computed number representing 1 seed in the store.

Determining legal moves

A move is illegal if no seeds are in that pit, or when the opponent has no moves after completion of your move. This means that checking whether a move is legal or not is comparable in complexity to actually doing the move. We therefore first perform the move and then decide whether it was legal, which is the case if any of the bits on the opponent's side is 1 after your move.

Such move can still be legal if none of your moves gives seeds to the opponent, but in that case all your moves have the same effect: you get all the remaining seeds on the board. For searching it is enough to award a player all the remaining seeds if he has no legal moves since that actually meant all his moves were legal, but would then leave the opponent without legal moves in the next turn.

```
//check if north just moved and all the bits on the south side
of the board are 0
if(pmin[move]==0 && (board&south)==0) return -1;
//check if south just moved and all the bits on the north side
of the board are 0
if(pmin[move]>0 && (board&north)==0) return -1;
//legal move, so return the new board position
return board;
```

north:
0000000000000000000000000000000011111111111111
1111111111111111

Note that the bits in the extra storage can be ignored here since they always belong to a pit that has seeds in it anyway and finding a single seed is already enough to conclude that the move was legal.

Move ordering

When using alpha-beta pruning it is very important in what order you evaluate the moves. If you evaluate the worst moves first you will find better results in every step and eventually end up searching as many nodes as in a normal min-max search. If search the best move early you can prune large parts of the board.

Ordering moves high in the tree has the biggest effect, because ending a path high in the tree cuts of a large sub

tree. Ordering near the leaves just makes your algorithm slower because sorting the moves takes more time than what you can win by searching the (few) underlying nodes more efficiently.

In each node where we want to sort the moves, we perform short alpha-beta searches to get an estimate of the value of the different moves. We then sort them and evaluate the best moves first. Because each level contains more nodes we do shorter searches to control the time spent on sorting. At every next level the ordering search is one ply less, so that all ordering searches go to the same point in the game. This means that all ordering searches are sub-trees of the first ordering search so if the results of that search are stored they can be used for all later ordering.

Move ordering made a difference of several plies, something which most other participants underestimated.

Transposition table

A transposition table is a hash table that contains previously calculated board positions. In many games different moves can lead to the same result. If a large sub tree must be evaluated to find the value of this position it can save a lot of time if this value is stored rather than evaluated every time this board position occurs somewhere in the tree.

Storing and retrieving board positions takes some time though and because Awalé has relatively few transpositions that lead to the same board position, the standard implementation of a Hash table could not really improve the search speed. We can search a lot of nodes as the time to update the board, using a bitboard, is very small. Compared to our move function, making a java object to store the position already takes relatively much time. The standard implementation also increases its size when it gets fuller, which is very slow due to the elements that mustj be copied to new array. We therefore made our own implementation that does not use objects, but stores the values in separate arrays (?) and that takes the maximum size at creation and is never rehashed.

For each position we store the long representing the real board position to check whether the retrieved value does not belong to another position with the same hash value, the calculated value, the search depth at which this position was evaluated and a flag whether the value was an alpha value (real value is equal to or lower than the stored value), a beta value (real value is equal to or higher than the stored value) or an exact value. Waarom is the real value different from the stored value?

When a position has the same hash as a position that was already stored, only one of the positions is kept. Exact values have the highest priority, then the values calculated at a higher depth follow and finally the values that were added later. The last condition is because old board positions are less likely to appear again.

The advantage of the transposition table for a single search is still not large, but the table is extremely useful for storing the results of ordering searches and also to reuse results of the previous move when the search depth is decreased because of lack of time.

Time management

Depending on the remaining time, we had a minimum and a maximum target time to be spent in a move. When we still had more than a minute left of our initial two minutes, moves had to be made in between 3 and 12 seconds. When less than 10 seconds time remained moves had to be made between 250 milliseconds and 1 second.

Mshindi then used iterative deepening of searches. When the previous search took less time than the minimum time to be spent on that move, another search was started, 2 plies deeper than the previous. When a search took more than the maximum time, the next move would be limited to 2 plies less than the search that took too much time.

Results

Mshindi won the competition, beating all its 11 opponents. Twice it gave up even 9 seeds in the opening for a better position, to come back from behind and win the game.

	Program		1	2	3	4	5	6	7	8	9	10	10	10
1	Mshindi	11		1	1	1	1	1	1	1	1	1	1	1
2	EMvanDoorn	10	0		1	1	1	1	1	1	1	1	1	1
3	Soultaker	9	0	0		1	1	1	1	1	1	1	1	1
4	Misha Stassen	8	0	0	0		1	1	1	1	1	1	1	1
5	Sebastian	7	0	0	0	0		1	1	1	1	1	1	1
6	Gan	6	0	0	0	0	0		1	1	1	1	1	1
7	Daan	5	0	0	0	0	0	0		1	1	1	1	1
8	Jeroen Bransen	4	0	0	0	0	0	0	0		1	1	1	1
9	AfterSix	3	0	0	0	0	0	0	0	0		1	1	1
10	Mathijs van Altvorst	1	0	0	0	0	0	0	0	0	0		1	0
10	Bas Kloppenburg	1	0	0	0	0	0	0	0	0	0	0		1
10	jvdb	1	0	0	0	0	0	0	0	0	0	1	0	

The stronger opponents could be grouped by those that used evaluation functions to value board positions on more information than just the intermediate score and those that put all their effort into searching as many nodes per second as possible. The first group searched typically at 13 to 15 plies, while the fast searchers reached depths 16 to 18 plies. Mshindi was one of the latter, but because of the better move ordering and the transposition table, which the other programs lacked it could reach depths of 20 to 22 plies from the start. The evaluation function could bridge the search difference of 3 or 4 plies with the other fast searchers, which was proven by the second place program which used genetic programming to improve his heuristic evaluation function, but searched at a modest 14 or 15 plies. He beat everyone except Mshindi, whose difference in search depth with was simply too large.

Sebastian used techniques to think in the opponent's time, but ran out of time against Misha whose program used almost no time at all.

Conclusions

The first implementation of min-max with a straightforward board representation in an array without alpha-beta pruning could search at 7 or 8 plies within the time limits. The addition of alpha-beta pruning increased this depth to 14 or 15 plies. Changing the board representation from an int array to a single 64-bit number added another 4 plies. Improving the move ordering and adding the transposition table finally allowed Mshindi to search deeper than the magical barrier of 20 plies. Especially this last step probably made the difference with the other programs because most used alpha-beta pruning and many used bitboards of either one long, or two ints, but none seemed to have paid much attention to move ordering and neither had a transposition table.

BIOGRAPHY

[1] Agudoawu, K.C., Rules for playing Oware, Winston publishing company, inc. Nashville 1991
[2] Board games round the world, Cambridge University Press 1991
[3] Kovach, R.P., Oware, the national game of Africa, Morrison Quick Print Oakland CA 1995
[4] Murray H.J.R., A history of board games other than chess, Claredon press Oxford 1952
[5] Oware, the national game of Africa, Sapient Software, www.myriad-online.com/en/products/awale.htm
[6] Awale: jeu de reflexion Africain, http://www.cityjeux.com/awale.phtml
[7] The Oware Society, http://www.oware.org
[8] Romein, J.W., Bal, H.E., Awari is solved, Journal of the International Computer Games Association, vol 25, number 3, pp. 162-165, September 2002
[9] Romein, J.W., and Bal, H.E., SolvingSolving Awari using Parallel Retrogade Analysis, IEEEE Computer, vol 36, number 10, pp. 26-33, October 2003
[10] Lincke, T.R. and Goot, R van der (2000) Marvin wins Aware Touranement. Journal of the International Computer Games Association, Vol 23, No 3, pp 173-174
[11] Min-max algorithm: http://en.wikipedia.org/wiki/Minimax
[12] Alpha-beta pruning: http://en.wikipedia.org/wiki/Alpha_beta_pruning
[13] Free Awale: http://savannah.nongnu.org/projects/awale
[14] http://www.brothersoft.com/downloads/awale-game.html

SERIOUS GAMING

TEAM BUILDING

Emerging Team Collaboration by Social Reinforcement

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

ABSTRACT

Artificial Intelligence plays nowadays a fundamental role in videogames and virtual reality applications. Learning is a useful adaptation trait and, in a social context, learning is very often a social feature. The goal of this paper is to emerge a team collaboration policy for a group of agents. To reach this goal, a social component will be added to the reinforcement enabling each agent to reinforce another agent in the team. A case of study will be presented to illustrate this claims.

KEYWORDS

Multi-Agent Systems, Social Learning, Reinforcement Learning, Videogames

Introduction

Artificial Intelligence plays nowadays a fundamental role in videogames and virtual reality applications [7]. Agents interacting with humans in this environments must have a credible visual appearance, but also a behaviour that maintains user immersion [4]. Agents also interact with other agents, and a credible interaction between them is also needed, be it cooperation or confrontation. Learning is a useful adaptation trait and, in a social context, learning is very often a social feature. Learning in videogames has two main advantages, the first one is that it can simplify the development of behaviours, and the second is that it can become a part of the game, as in [3] or [6].

Reinforcement Learning (RL) has been an active research field both in intelligent agents and game theory areas for many years[10]. However, most of the work in multi-agent learning has been done with a game-theory orientation, looking for a strategy that achieves some kind of equilibrium in repeated-games, usually in self-play.

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve.

The goal of this paper is to emerge a team collaboration policy for a group of agents. To reach this goal, a social component will be added to the reinforcement enabling each agent to reinforce another agent in the team. This reinforcement will depend on each agent's criteria, being its own goals or an attribute of its role in the system.

In the second section a brief overview of previous approaches to Multi-agent learning will be described for utter comparison and a social reinforcement approach to coordination in MAS will be presented. The third section contains the description and solution of a case of study using the technique described. The fourth section has some conclusions and intended future work.

Multi-Agent Learning (MAL)

There is an increasing interest in the application of learning techniques to MAS. The community tries to solve the need of adaptive methods that take into account the agents interaction. From a theoretical point of view, this has moved the learning problem from a Markov Decision Problem to game theory and stochastic games [8].

An early application of learning to multi-agent systems is found in [11]. With some experiments in the hunter / prey domain with a team of hunters, solo learning is compared with several kinds of collaborative learning. The sharing of perception, of reinforces and fully learning the same policy together are the strategies tested. Another early work about learning in a multi-agent setting, [5] deals with two main problems. A noisy environment in which learning a reactive behaviour is too difficult, and dealing with the reinforcement allocation between the robots. The first problem is inherent to any real world application. The perception must be abstracted to some state signal, and in this paper the actions learned are also abstracted to a behaviour level. The robot learns to choose between behaviours with a state signal based on its smoothed perceptions. The second one is a problem of reinforcement learning with a

team of agents. It's difficult to determine which agents should take credit for a positive reinforcement, or if the reinforcement comes from a team action, so it should be allocated to the combination of several behaviours in a given state. The paper proposes broking the reinforcement function in an aggregation of several progress estimations with some direct observable facts. The ideas are implemented in real robots with positive results.
Shoham et al.[9] review the different approaches that the research community has taken on MAL. The paper characterizes them in five distinct agendas, according to the problem addressed in each approach. The third agenda, called normative by Shoham et al. is centred in the notion of equilibrium, and which strategies are in equilibrium in repeated-games. It is the most depicted approach in the bibliography, and it's usually centered in how different algorithms converge with others, and if it is possible such convergence in several game scenarios. The fifth agenda describes a MAS learning as a setting in which the agent must optimize reward and which is characterized by the types of agents in it.
The main interest of this paper is benefiting of the learning mechanisms available for an agent immersed in a virtual social setting, so the fifth agenda described is the one which better fits with this goals.

A Social Approach to Reinforcement Learning

Conte[1] explains collaborative learning as seen in nature and explores the different kinds of social learning. Social learning is defined as the phenomenon by means of which a given agent updates its own knowledge base by perceiving the positive or negative effects of any given event undergone or actively produced by another agent on a state of the world which the learning agent has as a goal.
Learning is a useful adaptation trait and, in a social context, it is very often a social feature. Reinforcement learning deals with the adaptation of an agent to an environment. When several agents interact together and with a common environment, learning can be greatly improved with the sharing of common information, being it by teaching, by mutual observation or by any other learning mechanism [1]. Also, learning about the other agents is probably the most interesting piece of information in the setting, and it can't be learnt but from the interaction. Whether an agent wants to collaborate, compete or simply interact with other agents, there are rules and information about them that must be known. Learning relationships between agents can be of a wide range. They can be static and unidirectional, only between two agents, in a simple teacher-learner fashion. They can be multiple, with many teachers for a single learner or an only teacher or a small number of them for a big number of learning agents.
When an agent takes on a role that needs work done previously by other agents, it may give reinforcement for doing this work. It reinforces agents for collaboration, thus creating a source of reinforcement not present in the environment. This social reinforcement can be composed with other sources of reinforcement and thus a policy combining the costs and benefits of collaboration may be learned.
As agents learn how to collaborate, goals and preferences of each agent guide the group behaviour and a set of rules may emerge from learning. In these case, the rules are the outcome of the learning, not the object.[2] The purpose of this paper is to provide a collaboration policy for a group of agents so that they can solve a task together.

Emerging Team Collaboration by Social Reinforcement: A Case of Study

Problem Description

The task to be solved is the building of a house using materials from the environment. Only the choosing of the roles involved has been taken into account. In a full game or virtual reality application each role would trigger a set of tasks for the agent in the virtual environment.
Only one agent -the mason- can work in the building at a given moment, but it needs the rest of the group to provide the materials. For this example, the materials needed will be two, tables and stone. The first one, tables, needs also to be cut by a carpenter from raw wood. Both wood and stone can be collected from the environment by woodcutters and quarrymen. There are three warehouses that keep the materials, if some material is produced when the warehouse is full, it is wasted.
There are, thus, four roles for the agents to take: mason, carpenter, quarryman and woodcutter. There can be only one mason working in the house at a given moment. The mason and the carpenter need materials to work, if they lack any of the materials, they won't be productive. The mason receives positive reinforcement from the environment for building the house, there is no other external reinforcement. After 100 hundred rounds of productive mason work, a house is finished and a new house is started.

Proposed Solution

The proposed solution is the use of reinforcement to learn a collaborative behaviour for the group of agents. Each agent has a state signal with information of the warehouses -if they are full, medium or empty- and whether there is a mason working in the house at the moment. An expected reward table, initialized randomly, guides the role selection for each agent.
The roles of mason and carpenter receive the additional task of reinforcing the agents that provide needed materials for their work to be done. The reinforcement is

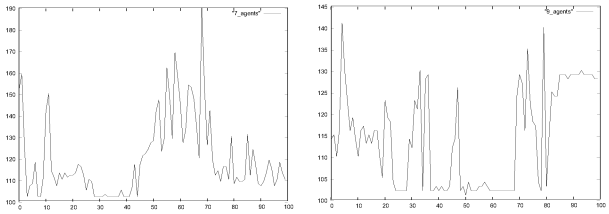


Figure 1: Number of rounds spent in each house built with seven and nine agents

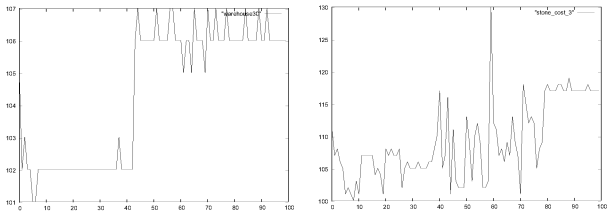


Figure 2: Number of rounds spent in each house built, after the 40th house, conditions has been changed.

positive if the warehouse has free space and negative if it is full.The sources of reinforcement are thus both social and environmental. Woodcutters and quarrymen only receive reinforcement.
The reinforcement learning algorithm used is TD(0)[10]. It only takes into account the last action for the reinforcement received, as is intended in this design.

Results

The proposed solution has been tested with several numbers of agents, warehouse sizes and materials needed for each task. Figure 1 shows some executions with seven and nine agents. After the completion of each house, all warehouses have been emptied, so the work in each case is the same. The minimum number of rounds for a house to be completed is 100 if there are enough agents to obtain the materials at the rate the masons uses them. As the state signal relates the number of the materials with the warehouse capacity, the learned policy is independent of the size of the warehouse and of the needs of materials. In the figure 2 we can see executions with different material needs and size of the warehouses. After the 40th house is built, conditions are changed. In the first case the size of the warehouse changed from 50 to 30, in the second, the stone needed by the mason for each round of building changed from 1 to 3.
It can be seen that the proposed solution works and a team collaboration emerges of the needs of each role.

Conclusions and Future Work

A reinforcement learning mechanism that benefits from the opportunities of a social setting has been described. Social learning can be used as a tool to learn collaboration.
In the case of study an agent learns from the social reinforcement of other members of the organization. The agents reinforce when the needs of their role are met, thus creating a collaboration between the agents and distributing the external reinforcement.
In the results presented it can be seen that the learned policies of the agents emerge a team collaboration.
As a future work it is intended to scale up the learning and integrate it in a full game and test the overhead caused.

Acknowledgements

This work was partially supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022. The financial support received from the Spanish government and FEDER funds under TIN2006-14630-C03-01 project is also gratefully acknowledged.

REFERENCES

[1] R. Conte. Intelligent social learning. In *AISB'00 Symposium on Starting from Society*, pages 1–11, 2000.

[2] J. Fabregat, C. Carrascosa, and V. Botti. A social reinforcement teaching approach to social rules. *Communications of SIWN*, 4:153–157, 2008.

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

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

[5] M. J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[6] Nintendo. Nintendogs, 2005.

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

[8] Y. Shoham, R. Powers, and T. Grenager. Multi-agent reinforcement learning: a critical survey, 2003.

[9] Y. Shoham, R. Powers, and T. Grenager. If multi-agent is the answer, what is the question? *Artificial Intelligence*, 171:365–377, 2007.

[10] R. S. Sutton and B. A. G. *Reinforcement Learning: An introduction*. MIT Press, Cambridge, USA, 1998.

[11] M. Tan. Multi-agent reinforcement learning: independent vs. cooperative agents. In *Readings in agents*, pages 487 – 494, 1997.

GENERATING EMERGENT TEAM STRATEGIES IN FOOTBALL SIMULATION VIDEOGAMES VIA GENETIC ALGORITHMS

Antonio J. Fernández, Carlos Cotta and Rafael Campaña Ceballos
ETSI Informática,
Departmento de Lenguajes y Ciencias de la Computación,
University of Málaga,
Málaga, Spain
E-mails: {afdez,ccottap}@lcc.uma.es

KEYWORDS
AI, Evolutionary Algorithm, Simulation, Robot Football.

ABSTRACT

This paper defends the use of evolutionary algorithms to generate (and evolve) strategies that manage the behavior of a team in simulated football videogames. The chosen framework to develop the experiments is Robocup, an internatonal project that promotes the use of Artificial Intelligence in socially significant areas. One of these areas is related to computer games in the form of a simulated soccer league that allows two teams of 11 simulated robotic autonomous players to play football without human intervention. This paper proposes to generate emergent behaviors for the teams, via an evolutionary training process. The proposal is an alternative to implementing specific AI controllers for both players and teams in football videogames.

INTRODUCTION

The main aim of videogames (VG) is to provide entertainment to the player(s). In the past, research on commercial VGs was mainly focused on having more realistic games by improving graphics and sound (i.e., having higher resolution textures, more frames-per-second, ...etc). However, in recent years, hardware components have experienced exponential growth and players, with higher processing power computers, demand higher quality opponents exhibiting intelligent behavior.

In many simulated sports video games, the opposing team / the opponent (i.e., the enemy) is basically controlled by a fixed script. This is previously programmed and often comprises hundreds of rules, in the form *if the game is in state S then execute action A,* to control the behavior of the components (e.g.. members or players) of the team under specific conditions of the framework (i.e., a specific state of the game). This is quite a problem from both the developer and player point of view. For the former, it is a problem because these scripts are becoming more and more complex and thus it is not easy to program all the possible situations that could potentially happen. In fact, most of the games contain 'holes' in the sense that the game stagnates or behaves incorrectly under very specific conditions. As a consequence, the reality of the simulation is drastically reduced and so too the interest of the player. This problem relies on the category of 'artificial stupidity' (Lidén 2004). Also, for players, these scripts that model the opponent behavior are pre-programmed schemas whose behavior can become predictable for the experienced player, again causing a decrease in player interest.

To solve these problems, existing sports games employ some kind of artificial intelligence (AI) technique with the aim of making the opponents more intelligent, thereby making the game more attractive and increasing player satisfaction. However, even in these cases, the reality with respect to current sports videogames is that game AI is either not really AI and often consists of very specialized scripts (with the same problems as those already mentioned), or else game AI basically mimics a human player behavior. Nevertheless, even in this latter case, a problem remains: the AI controlled strategy rarely evolves according to the behavior of the player during the game. Again the reality is that most videogames are divided into levels, and the opponents are pre-programmed according to these. In the most complex levels the player faces high-quality opponents who behave like humans. Once the player is able to beat all the opponents in each level, they lose interest. In this context, the generation of opponents whose behavior evolves in accordance with the player's increasing abilities would be an appealing feature and would make the game more attractive. For instance, an amateur player expects an amateur opponent (not necessarily pre-programmed) whereas a very experienced player demands high-quality opponents. Moreover, the addition of emergent behavior in a football simulation game can make it more entertaining and less predictable in the sense that emergent behavior is not explictly programmed but simply happens (Holland 2000;Sweetser 2007).

This paper represents a step in this direction and deals with football simulation videogames. It proposes the use of genetic algorithms (GAs), to generate, in a dynamic way, opponents that depend on both the user skills and the game level. The main contributions of the paper are the natural encoding of the team strategies, that make our proposal very simple to manage, and the definition of a fitness function based on two heterogeneous components to guide the processes of learning and improvement of the team strategies inside the genetic algorithm. We report our experience using

Genetic Algorithms (GAs) in the context of Robocup, an international robot soccer simulation framework in which researchers can develop new ideas in the area of AI, and their developments can be evaluated via a competition mechanism in which any AI proposal is tested against another one. It should be observed that our experience can be extrapolated to commercial football simulation videogames.

RELATED WORK

AI can play an important role in the success or failure of a game and some major AI techniques have already been used in existing videogames (Johnson and Wiles 2001; Millington 2006). Traditionally, game developers have preferred standard AI techniques such as Artificial Life, Neural Networks, Finite State Machines, Fuzzy Logic, Learning and Expert Systems, among others (Bourg and Seemann 2004; Mikkulainen et al. 2006).

Evolutionary algorithms (EAs) (we use this term in a broad sense to refer to any kind of evolutionary procedure, including genetic algorithms and genetic programming) offer interesting opportunities for creating intelligence in strategy or role-playing games and, on the Internet, it is possible to find a number of articles related to the use of evolutionary techniques in VGs. For instance, (Sweetser 2004) shows how EAs can be used in games for solving pathfinding problems; also (Buckland 2002) focused on bot navigation (i.e., exploration and obstacle avoidance) and proposed the use of evolutionary techniques to evolve control sequences for game agents. However, in general, most of the work published on the use of EAs in games is aimed at showing the important role EAs play in Game Theory and, particularly, their use in solving decision-taking (mainly board) games. For example, (Kim and Cho 2007) presented several methods to incorporate domain knowledge into evolutionary board game frameworks and chose the board games checkers and Othello to experimentally validate the techniques. Also, it is worth mentioning the works of Fogel, which explored the possibility of learning how to play checkers without requiring the addition of human expertise via co-evolutionary techniques (Fogel 2000; Chellapilla and Fogel 2001). Other decision-taking games that have been handled via evolutionary methods are for example poker (Barone and While 1999), Backgammon (Pollack and Blair 1998) and Chinese Chess (Ong et al. 2007).

Evolutionary techniques involve considerable computational cost and thus are rarely used in on-line games. One exception however, published in (Fernández and Jiménez 2004), describes how to implement a genetic algorithm used on-line in an action game (i.e. a first/third person shooter). In fact, the most successful proposals for using EAs in games correspond to off-line applications, that is to say, the EA works on the user's computer (e.g., to improve the operational rules that guide the opponent's actions) while the game is not being played and the results (e.g., improvements) can be used later online (i.e., during the playing of the game). Through offline evolutionary learning, the quality of opponent intelligence in commercial games can be improved, and this has been proven to be more effective than opponent-based scripts (Spronck, Sprinkhuizen-Kuyper, and Postma 2003). Also, genetic algorithms have been used to evolve combat strategies for agents or opponents in between games (i.e., offline learning) as was done in the classical *Unreal Tournament* (Dalgaard and Holm 2002). Some realistic VGs that have used genetic algorithms are *return Fire II*, *The Creatures Series*, *Sigma and Cloak*, *Dagger and DNA*, and *Quake III*. For more detailed information about the use of EA in games the reader is referred to (Fogel, Blair, and Miikkulainen 2005; Lucas and Kendall 2006).

Regarding sports simulation VGs, readers can find some papers describing the evolutionary experiences mainly in simulated robot soccer frameworks. For instance, (Luke 1998) reported his experience of applying genetic programming (GP) to evolve team strategies. The teams were represented by trees and the GP system used low-level atomic functions designed for the soccer environment to build the strategies. Also, (Agah and Yanie 1997) used a genetic algorithm to evolve a team of 5 robots in a simulated soccer setting. Here, the agents (i.e., the robots) were built based on the approach of the Tropism-Based control architecture. (Nakashima et al. 2005) proposed to encode the set of action rules for soccer agents into integer strings. Although this is similar to what we have done, the approach is radically different as Nakashima et al. divided the soccer field into 48 subareas, and the action of the agent is specified for each subarea. Also, the replacement policy in the evolutionary process was quite different and they used a standard evolution strategy-type generation replacement schema i.e., the $(\mu+\lambda)$-strategy (Eiben and Smith 2003).

GENETIC ALGORITHMS

A genetic algorithm is a population-based optimization algorithm that uses techniques inspired by evolutionary biology such as selection, inheritance, mutation, and recombination. Genetic algorithms manipulate a population of candidate solutions (also known as individuals), traditionally represented by binary strings that evolve towards better solutions. A typical genetic algorithm schema is shown in Figure 1:

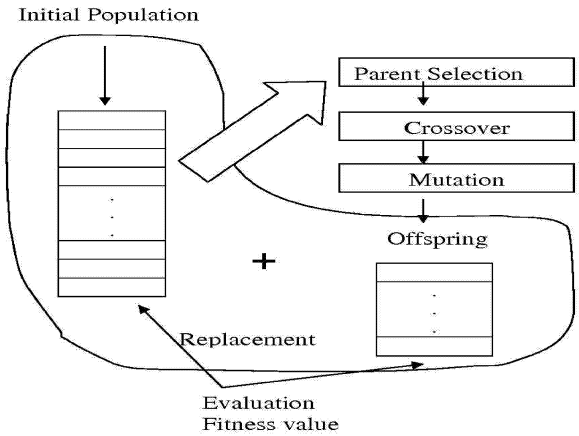


Figure 1: Standard Schema of a Genetic Algorithm

The basic process is as follows: initially a population of individuals is (often randomly) generated and the fitness of each member of this population is evaluated. Then the algorithm is executed until a termination condition is satisfied (e.g., a number of generations –iterations – or evaluations is reached, a solution is found, a desirable fitness value is obtained, etc). In each generation, some individuals (i.e., parents) from the current population are selected stochastically (this selection is usually based on their fitness values) and recombined to produce an offspring. The newly generated individuals can be modified via a mutation process; then the new individuals are evaluated. Finally, the population for the next generation is constructed from the individuals belonging to the current population and the new ones, now referred to as the offspring. This new population is then used as the current population in the next iteration of the algorithm (Eiben and Smith 2003).

ROBOCUP: THE SIMULATION LEAGUE

Robocup is an international framework in which researchers can develop new ideas in the area of AI and their developments can be evaluated via a competition mechanism in which any AI proposal is tested against another one. The basis of the research is robot soccer. Among the five leagues in RoboCup soccer, the simulation league is the most active research domain in terms of the development of computational intelligence techniques. This league is based on a simulator called "soccer server", which is a software system that simulates a football match. The server is a real time system that provides support for the competition between multiple virtual players in a multi-agent framework. It eases the communication with the client program, manages the entire physical environment (i.e., communications, objects, and hardware issues), and allows the visualization of the match in an X-window system. Figure 2 shows an example of a simulated match. The advantage of using the soccer server is that developers can focus on conceptual tasks such cooperation and learning. We will not go into more details as it is preferible to concentrate on the artificial evolution of the team strategies. The reader is referred to (Noda and Stone 2003) for more information on the server.

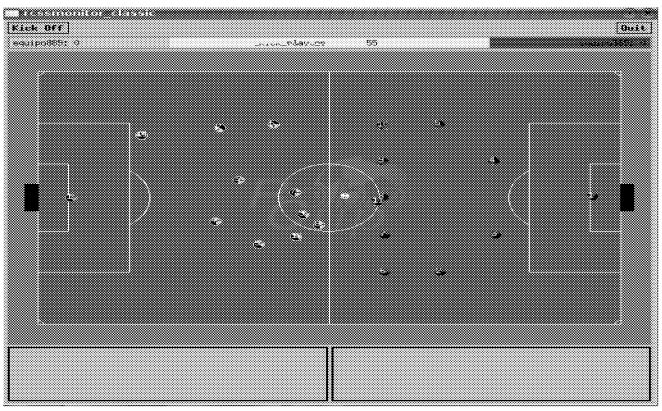


Figure 2: Soccer Server: An Example of a Simulated Match

The rules of a simulated match - very similar to those of the FIFA – (Fédération Internationale de Football Association) - are imposed (e.g., kick-off, goal, off side, half-time, time-up, players must not block the movement of other players, etc.). A Robocup agent (i.e., an autonomous player/bot)) receives, in real-time, information from the environment via three sensors: *aural sensor* (to detect the messages from the referee, trainers and the rest of the players); *visual sensor* (to detect information into a limited region, called the *agent visual range*; the information is about the current state of the match in the form of distance to and direction of the close objects i.e., players, balls, etc, all within a specific area of vision); and *corporal sensor* (to detect the physical state of the agent i.e., energy, velocity and direction).

EVOLUTION OF TEAM STRATEGIES

Our aim is to generate controllers to govern the behavior of an entire team (i.e., a set of 11 autonomours players/bots). A description of the technical issues (e.g., management of communication with soccer server or implementation of basic agent actions e.g., shot, pass, run, turn, etc., among others) of our proposal is beyond the scope of this paper. We concentrate thus on the process of developing controller behaviour.

The first step consists of generating (and evolving) a set of rules that will control the reactions of the agents at each instant of the game. These reactions depend on the current state of play. As already mentioned, each agent is continuously informed about the ongoing state of the game through the communication with the soccer server (via the three sensors mentioned above). Depending on the situation, the agent executes a particular action that may modify the state of the agent itself. The global team strategy is then the result of the sum of the specific behavior of all the agents.

However, the definition of specific strategies for each agent is complex, costly, requires a profound knowledge of the required behavior of each agent, and results in predictable behavior. To avoid these problems, all the agents (except the goalkeeper, who is manually implemented) will be managed by the same evolved controller. This means that we have to produce just one controller (thereby making the evolution process very much cheaper) to devise a global team strategy. Note, this does not mean that all the agents execute the same action because this depends on the individual situation of the agent in the match.

In the following we provide details of the genetic algorithm used to evolve team strategies.

Chromosome Representation (encoding): each individual in the population represents a team strategy, that is to say, a common set of actions that each agent has to carry out under specific conditions of the environment. These conditions depend on the information that the agent receives in its specific visual range (this differs from one agent to another). The information comes in the form of parameters that can take values from a range of values. Then, an individual in the population is represented as a vector v of k cells where k is the number of different situations in which the agent can be, and $v[i]$ contains the action to be taken in a specific

situation. In other words, if there are m parameters, and the parameter p_i (for $0 \leq i \leq m-1$) can have k_i possible values (numbered for 0 to k_{i-1}), then the cell:

$$v[e_{m-1}+e_{m-2}*k_{m-1}+e_{m-3}*(k_{m-1}*k_{m-2})+e_{m-4}*(k_{m-1}*k_{m-2}*k_{m-3})+\ldots]$$

contains the action to be executed when the parameters p_0, $p_1,\ldots,$ p_{m-1} take the values e_0, $e_1,\ldots,$ e_{m-1} respectively. Managing a large number of parameters (such as those provided by the server) is complex. Thus to reduce complexity, in our experiments we have considered the following manageable set of parameters:

- Advantage state (A_s): This paremeter can take two values that evaluate a possible advantage situation. 0: if the agent is supported by more team mates than rival ones. 1: otherwise.
- Ball kick (B_k): can the agent kick the ball? Two values. 0: the agent cannot; 1: the agent can.
- Agent position in the field (A_p): three values. 0:closer to its own goal area; 1: closer to rival goal area, 3: not defined.
- Ball possession (B_p): which agent is closer to the ball ? Four values. 0: the agent, 1: a team mate, 2: a rival agent, 3: not known.

The encoding of an individual in the population consists of a vector of 48 genes i.e., all the possible conditions that can happen from the different combinations of these parameter values. An additional parameter was also considered:

- Position of Ball (P_b). Three values depending on the proximity of the ball to the goal areas. 0: closer to agent goal area, 1: closer to the rival goal area, 2: not defined.

Considering the addition of this parameter the representation is a vector of length 144. We note that some combinations of values in the representations may make no sense (e.g., The combination $A_p = 0$ and $P_b = 1$ makes no sense with $B_k = 1$ because the agent would never be able to kick the ball). The addition of this kind of knowledge can lead to a simplified representation but in our experiments we did not limit any combination (in any case, this kind of optimization can be done at a later stage).

Each cell of the vector encoding a candidate solution will contain an action. In our experiments, 9 actions were considered:

- Go back: each agent has two positions by default: the *starting position* that corresponds to a fixed position in the field in which the agent is placed at the beginning of the match as well as after scoring a goal, and the *required position* that corresponds to a strategic position in which the agent can be placed during the game (e.g., forwards and defenders should be placed close to the rival area or team area respectively).
- Look for the ball: turn the agent body to align it with the ball if this is visible, otherwise turn 15º (an arbitrary value) to the right to modify the environmental conditions received in the visual area of the agent.
- Intercept ball: If ball is visible then the agent accelerates in that direction.
- Look around. Turn 15º right.
- Kick. Shoot in direction to the rival goal area.
- Pass to closest team mate.
- Pass to farthest team mate.
- Kick out: try to put the ball as far as possible from the team goal area.
- Drive ball: conduct the ball in the direction of the rival goal area.

This means that the search space (i.e., the number of different strategies that can be generated from this representation) is $9^{144} = 3^{288}$ if we consider the second representation, and $9^{48} = 3^{96}$ if we consider the 48 length representation. Figure 3 displays an example of a possible encoding of length 48. The optimal solution (if it exists) would be that strategy which always select the best action to be executed for the agents under all possible environmental conditions. In fact, these vast search spaces make this problem impracticable for many exact methods and ideal for genetic algorithms.

	Ball possesion (B_p)			
$A_sB_kA_p$	agent	Team mate	Rival agent	Not known
000	1	6	9	3
001	8	8	3	7
002	5	1	2	4
010	4	2	5	6
011	6	9	3	5
012	1	1	2	7
100	7	4	9	2
101	1	2	3	5
102	6	5	4	8
110	3	6	1	7
111	7	2	3	1
112	2	9	8	5

Example: Execute action 7 (pass away) if the player is not supported by team mates, can kick the ball, is positioned close to its lodge, and is the player placed closest to the ball in the field.

Actions:
1. Go back default position
2. Look for the ball
3. Intercept ball
4. Look around
5. Kick
6. Pass to friend
7. Pass away
8. Kick out
9. Drive ball

Parameters:
As: Advantage state
Bk: ¿ can the player kick the ball ?
Ap: ¿ Where is the player placed in the field?
Bp: ¿ Who is closer to the ball ?

Figure 3: Example of Encoding for an Arbitrary Individual

Fitness function evaluates the adequacy of a team strategy. Let *pop* be the population considered in the GA. Then, evaluating the fitness of an individual *pop[i]* (for $1 \leq i \leq$ population size) requires the simulation (in the soccer server) of the match between the opponent strategy (e.g., the one followed by a human in a previous game) and the strategy encoded in *pop[i]*. The fitness function depends on the statistical data collected at the end of the simulation (e.g., who won, goals scored by both teams, etc). The higher the number of statistical data to be collected, the higher the computational cost will be. A priori, it seems a good policy would be to consider a limited number of data. Five data were used in our experiments during the simulation of a match between *pop[i]* and the opponent.

1. Ball closeness (c_i): distance average from team players to the ball.
2. Ball possession (p_i): average time that the ball is in the rival field.
3. Ball in area (a_i): average time that the ball is in the rival area.
4. Scored goals (sg_i).
5. Received goals (rg_i).

The fitness function to evaluate any individual *pop[i]* in the population is then defined as follows::

$$fitness(pop[i]) = f_1(c_i, p_i, a_i) + f_2(sg_i, rg_i)$$

where

$$f_1(c,p,a) = w_1 \cdot c + w_2 \cdot p + w_3 \cdot a \qquad (1)$$

and

$$f_2(sg,rg) = \begin{cases} 0, & \text{if } sg = 0 \\ w_4 \cdot ((sg - rg) + 1), & \text{if } ((sg \geq rg) \\ (w_4 - 1)/(rg - sg), & \text{otherwise} \end{cases} \qquad (2)$$

The higher the fitness value of a strategy is, the better the strategy. Observe that the fitness function is based on two very different components. The first, defined in (1), has the aim of teaching to the strategy *pop[i]* how to play according to the basic rules of football (note that initially the population is randomly initialized and thus most individuals will not even know how to play football). Weights were assigned as w_1=0.01, w_2=0.1, and w_3=1; the reason for assigning these values was to promote the evolution towards strategies in which the ball is in play more time in the rival field (if possible near the goal area) than in the team field (it seems reasonable to think that this policy will result in less "received" goals). The second component of the fitness function, defined in (2), should help the strategy *pop[i]* to evolve towards better solutions (i.e., those able to beat the opponent). The weight w_4 is assigned to 100 so that in the case of a victory or a draw, the function returns a number that is multiple of 100, thus giving priority to the higher difference of goals; otherwise (in the case of defeat), priority is given to a minor difference of goals.

The fitness function is non-deterministic (i.e., different simulations of the same match can produce different results) as the simulation is affected by random factors that the soccer server adds to provide a greater sensation of realism. This fact, which can be viewed as another reason for not using complete solving techniques, is really an added incentive to use genetic algorithms as it is well-known that GAs incorporate a certain degree of randomness that makes them suitable for handling this problem.

Evolutionary operators: Our GA is a steady state algorithm that uses single point crossover, binary tournament for the parent selection, and elitism for the replacement policy. Mutation is done in the gene – i.e., action – level by changing an action to any other action.

EXPERIMENTAL SECTION

Extensive tests, varying the probability of mutation P_M (i.e., 0.1, 0.01, 0.001, 0.0001), the number of generations (i.e., 150,300), offspring length (i.e., 1,2,3,4), and individual size (i.e., 48 and 144) were carried out. The population length and the crossover probability P_X were set to 30 and 1.0, respectively in all the tests (i.e., instances) and the population was always initialized randomly. Ten runs per test instance were executed. In addition, two type of tests were conducted: one (mode 1) in which the GA was executed with the aim of finding one strategy (i.e., a winning strategy) to beat a manually implemented opponent; and another type (mode 2) in which each winning strategy obtained was incorporated into the objective function (i.e., the objective was not only to beat an opponent but to beat all the opponents in the objective function). The experiments demonstrated the validity of our proposal in the first mode. Due to the large number of experiments done, only some examples of evolutionary advances are shown in the following.

For instance, Figure 4 displays the average fitness (from ten runs) resulting from one of the test instances executed. It is interesting to note that initially the team strategies encoded in the population as individuals play really very badly (in fact, they do not know how to play), but they evolve quickly within a few generations. This behaviour of the evolutionary process is common in all the tests.

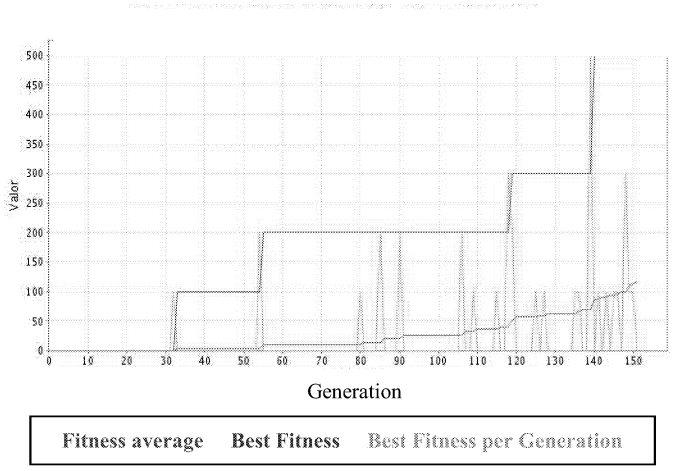


Figure 4: Pop size: 30, Offspring size: 1, P_M = 0.001. Average Results for Fitness Value/Generations

Figure 5 shows for the same instance, the evolution of the data values used to define the fitness function; one can observe for instance that the "received" goals decrease whereas the scored goals increase. Also, note the pressure to place the ball close to the rival area.

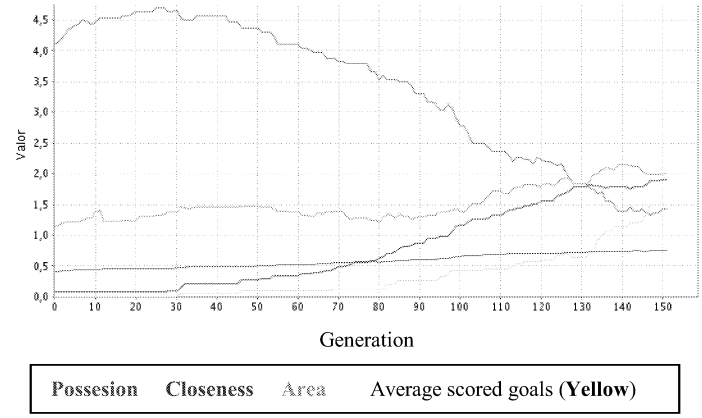


Figure 5. Evolution of Values Defining the Fitness Function

Tests in mode 2 were also developed with poor results: the algorithm could be executed with two opponents in the objective function but not with three. The reason for this bad performance is the cost of evaluating the fitness values since for each individual, three simulations had to be carried out at

considerable computational cost (note that the strategy encoded in the individual has to be evaluated against each of the three teams included in the objective function). Note however that this mode rarely has applications in simulated football games in which there is just one opponent (i.e., the human player). Mode 2 is interesting if one wants to produce strategies to be tested in a competitive environment such as for instance Robocup, but this was not the original motivation for this work.

CONCLUSIONS AND FURTHER RESEARCH

In this paper we have shown that genetic algorithms are a simple mechanism to obtain emergent behavior strategies to control teams in football simulation videogames (such as for example *FIFA*, *Pro-evolution or Football Manager* series). The genetic algorithm described in this paper has the particularity that is guided by a fitness function defined with two very heterogeneous components: one that guides the basic learning of the football principles, and the other that strives to find winning strategies. These two components, although radically different, are complementary however. Experiments were conducted to validate the feasibility of our approach with promising results. The evolutionary learning described in this paper could be used in existing football simulation games taking as opponent the player's game strategy, which can be deduced by collecting statistical data during the game. In this sense, the evolutionary algorithm would produce player's skills-based self-adaptive opponents. This, together with the randomness associated with GAs, would lead the evolutionary process towards the generation of opponents with non-predictable behavior.

The main drawback of our technique is the computational cost associated to the simulation of matches that are necessary to execute in order to evaluate the fitness values of the evolved teams. This however can be minimized in our working framework as the soccer server provides facilities for parallel execution and therefore several matches can be running on different machines at the same time, thus reducing the computational cost. Nevertheless, this is an issue for further research.

ACKNOWLEDGMENTS

This work has been partially supported by projects TIN2007-67134 and TIN2008-05941 (from Spanish Ministry of Innovation and Science) and P06-TIC2250 (from Andalusia Regional Government).

REFERENCES

Agah, A. and Yanie, K. 1997. "*Robots Playing to Win: Evolutionary Soccer Strategies*". In Proceedings of the 1997 IEEE lnternational Conference on Robotics and Automation, 632-637.

Barone, L. and While, L. 1999. "*An Adaptive Learning Model for Simplified Poker Using Evolutionary Algorithm*". In Proceedings of the Congress on Evolutionary Computation, IEEE Press, 153-160.

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

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

Chellapilla, K. and Fogel, D.B. 2001. "Evolving an expert checkers playing program without using human expertise". In *IEEE Trans. Evolutionary Computation* 5, No. 4, 422-428.

Dalgaard, J.and Holm, J. 2002. "Genetic Programming applied to a real time game domain". Master Thesis, Aalborg University - Institute of Computer Science, Denmark.

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

Fernández, A.J. and Jiménez, J. 2004. "Action Games: Evolutive Experiences". In *Computational Intelligence: Theory and Applications*, Bernd Reusch (ed.), Springer, 487-501.

Fogel, D.B. 2000. "Evolving a checkers player without relying on human experience". In *Intelligence* 11, No. 2, 20-27 (2000)

Fogel, D B; Blair, A; and Miikkulainen, R (eds). 2005. "Special Issue: Evolutionary Computation and Games". In. *IEEE Trans. Evolutionary Computation* 9, No. 6,

Holland, J.H. 2000. *Emergence: from Chaos to Order*. Oxford University Press.

Johnson, D. and Wiles, J. 2001. "Computer Games with Intelligence". In *Australian Journal of Intelligent Information Processing Systems* 7, 61-68.

Kim, K-J., and Cho, S-B. 2007. "Evolutionary Algorithms for Board Game Players with Domain Knowledge". In *Advanced Intelligent Paradigms in Computer Gam*es, Baba, N, Jain L.C. and Handa H. (eds), Springer, 71-89.

Lidén, L. 2004, "Artificial Stupidity: The Art of Making Intentional Mistakes", In *AI Game Programming Wisdom 2*, S. Rabin, ed., Charles River Media, Inc., pp. 41-48.

Lucas, S.M., and Kendall, G. 2006. "Evolutionary Computation and Games". In *IEEE Computational Intelligence Magazine* 1, No.1, 10-18.

Luke, S. 1998. "*Evolving Soccerbots: a Retrospective*". In Proceedings of 12th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI). Invited paper.

Miikkulainen, R; Bryant, B.D.; Cornelius, R.; Karpov I.V.; Stanley, K.O.; and Yong, C.H. 2006. "Computational Intelligence in Games". In *Computational Intelligence: Principles and Practice*, Yen, G. Y. and Fogel, D. B. (editors), IEEE Computational Intelligence Society, 155-191.

Millington, I: 2006. *Artificial Intelligence for Games*. Morgan Kaufmann.

Nakashima, N; Takatani, M; Udo, M; Ishibuchi, H; and Nii, M. 2005. "*Performance Evaluation of an Evolutionary Method for RoboCup Soccer Strategies*". In *Proceedings of RoboCup 2005*, LNCS 4020, Springer, 616-623.

Noda, I: and Stone P. 2003. "The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS Research**".** *Autonomous Agents and Multi-Agent Systems* 7 , No.1-2 (July-September), 101-120.

Ong, C.S.; Quek, H.Y.; Tan, K.C.; and Tay, A. 2007. "*Discovering Chinese Chess Strategies through Coevolutionary Approaches*". In Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games, 360-367.

Pollack, J.B. and Blair, A.D.1998. "Co-Evolution in the Successful Learning of Backgammon Strategy". In *Machine Learning* 32, No. 1, 22-240.

Spronck, P.; Sprinkhuizen-Kuyper, I.; and Postma, E. 2003. "Improving Opponent Intelligence through Offline Evolutionary Learning". In *International Journal of Intelligent Games&Simulation* 2, No. 1, 20-27.

Sweetser, P. 2004. "How to Build Evolutionary Algorithms for Games". In *AI Game Programming Wisdom 2*, S. Rabin, ed., Charles River Media, Inc., pp. 627-638.

Sweetser, P. 2007. *Emergence in Games*. Charles River Media.

EDUCATION AND SOCIAL NETWORKS

MYHEART

Juan Albero Sanchis
Universitat Politecnica de Valencia
Camí de Vera,s/n. 46022 València
CPI, Edificio 8G, Itaca-TSB, planta 1, lab 1
E-mail: juaalsan@itaca.upv.es

KEYWORDS
Health care, peripheral, services.

ABSTRACT

Electronic applications can not only be used as "games", this is the prove of how to generate entertainment and health care using "game" methodologies such as custom peripherals and virtual worlds.

INTRODUCTION

The overall perception of electronic entertainment applications is that it's mainly focused on kids and thus is only made to this certain public objective. What the MyHeart project wants to break is that perception, showing how electronic applications can be used to solve cardiovascular and stress problems and to achieve the results estimated in theoretical problems while keeping the users entertained.

ACTIVITY COACH

TakeCare Concept

Cardiovascular diseases (CVD) are due to a combination of several risk factors. Expert panels, like the European Society of Cardiology (ESC), the American Heart Association (AHA/ACC), have written practice guidelines to define those risks, recommended lifestyle, and therapeutic targets for cardiovascular disease prevention. It has been proven that lifestyle interventions, such as regular physical activity or a healthy diet can significantly reduce cardiovascular morbidity and mortality.

TakeCare addresses especially to those who:

- Try to be more active and to lose weight,
- Have sleep and stress related problems,
- Quickly resign from following lifestyle adoptions due a lack of self-motivation.

TakeCare Objectives

TakeCare provides support for customers to carry out self-management of CVD risks in daily life. For a successful self-management, the CVD risks of each individual are assessed allowing to derive appropriate improvement plans and personalized recommendations. With the help and support provided by the TakeCare system (and services), primary prevention will be permanently established in daily life and more customers will participate successfully in risk reduction on the long run. TakeCare concentrates on education, monitoring and coaching in order to provide the best chance of lifestyle improvement.

The positioning of TakeCare is towards a serious medical product that allows reliable and trustful measurements, analysis and support to overcome problems.

Status

The AC Product concept was designed to provide a continuity of care to people who want to stay healthy and therefore prevent CVD. The system provides personalized training information to the user, based on their own personally set goal and previous results.

As main target group is people exercising for fitness & fun. This group has as distinctive characteristics an extrinsic motivation to do physical activity, seeking for optimal exercise results at the lowest effort. The concept is also extensible to more advance users that want an improvement in their performance.

MYHEART

Introduction

MyHeart is how the Activity Coach product plus the TakeCare Concept finally came to life.

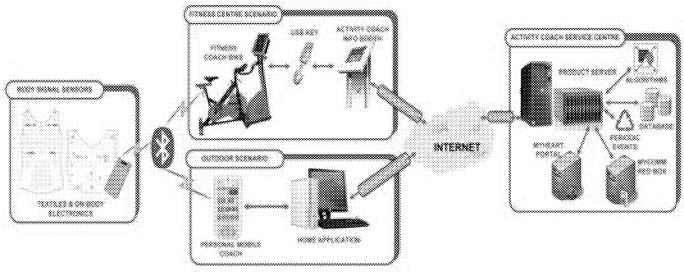


Figure 1. System overview

The overall system functionality (fig. 1) works as follow:

- The *Body signal Sensor* (BSS) it is a vest that contains the on body sensors and electronics. The data obtained is transmitted to the user interaction devices via Bluetooth.
- In the Fitness Center users will have a personal identification device (USB key) which stores the personal profile, authentication data and details for the upcoming training exercise using the *InfoBooth*.
- The USB key is then plugged into the *Fitness Bike* that provides the proper sensor receivers for the BSS. The results are presented via a multimedia 3D graphics environment.

- All information collected and evaluated is then sent to the *Service Center* through the InfoBooth.

To successfully develop the system the following research and development has been done:

- Analysis of previous developed exercise algorithm in order to assess the suitability for the exercise.
- A user profile map, including users' different physiological parameters and lifestyle habits with the aim to better personalize long-term training programs. These parameters have been analyzed to determine how they can influence on the design of a session of exercises and the relationship between them.
- The algorithm for training program based on the user profile map and takes into account the main parameters related to the exercise and training planning.
- Algorithm for fitness assessment has been defined with the aim of extending the applicability of the Heart Rate Recovery test and the algorithm itself to indoor and outdoor running.
- Recruitment of volunteer group that has been trained in both activities (bike and running), performing the assessment test in order to validate the algorithms for training program creation and fitness assessment.

Needed Definitions

Physiological Profile: Defines the task to do in a session from a psychological standpoint. It is designed to suit the user and has as a primary objective to improve the physical performance inside cardiovascular parameters.

It is done thanks to the session creator algorithm that takes:

- Input parameters, like the users age, sex, habits (smokes or not), fitness levels, and possible previous diseases.
- And provides the bike profile (session number, Heart Rate (HR) objective, HR max, HR min and duration.

Motivation Profile: Defines the task to do in a session from a training standpoint. It takes the user profile referencing music preferences with the objective to maintain the user motivated during the exercise.

Those profiles are hand made and stored on a database for reutilization, and based upon:

- Total Duration and number of exercises
- HR range according to selected music
- Music Style (based on user preferences)
- Song for each exercise
- Type of exercise (resistance: slope, plain, descent)
- Motivation signals and associated message

This profile has created an exercise based on temporary placed marks and the associated messages. Those marks will indicate the system that the proper motivation event will have to be fired through the suited channel according to its type and message to transmit.

Session Creation

The data coming from a user profile and feedback from previous sessions is compared with data from the motivation profile based on parameters such as the HR objective in the physiological program. The length of the motivation profile has to be as similar as the physiological profile and that it has not been previously assigned to the user.

The result will be the training session, that the user will download from the server through the InfoBooth and that will be composed of an audio file (with all the songs of the motivation profile) and a configuration file that will contain the physiological profile and the motivation profile.

The training program creation algorithm is self-adaptative to the different physiological needs and conditions.

SYSTEM DESCRIPTION

The Body Signal Sensor

It is a vest made of an intelligent textile the user wears on and that incorporates the proper sensors and electronics. It is responsible for monitoring the required vital signals.

The signals that can be measured are: heart rate, heart rate variability, breath rate, breath amplitude, pace speed and activity level (3D accelerometer). It is also responsible of the sensors integrated in the stationary devices such as pedaling frequency, speed and distance.

Those signals are sent back to the bike through a Bluetooth communication unit.

The Fitness Coach Bike

The Fitness Bike is composed by two main parts, the hardware and the multimedia application.

The hardware of the bike has sensors for rpm data acquisition and a specific keyboard for user interaction. The bike design was focused on to offer a fully global solution, embedding each part of the system into a unique final product. This offers an easier interaction with user, as well as a simple system installation process.

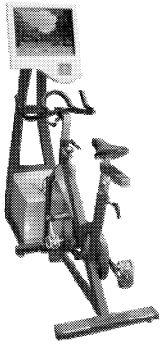


Figure 2. The Fitness Bike.

The FitnessBike Application is the multimedia application that runs in the bike. The *core application* is responsible for the user interaction, hardware communication (rpm sensors, USB port, audio jack, monitor...)

The *3D environment* is a dedicated module that implements a 3D engine capable of navigating through a 3D world. This module is connected to the sensors through the core application in order to give a feeling of reality. It is also responsible for giving feedback and coaching to the user by using both visual and audio content.

The *motivation and coaching algorithm* that runs in the bike creates events to provide appropriate motivation messages and takes as input parameters the physiological profile and the measurements coming from the sensors (user HR, rpm and respiration). The channels of motivation are (fig. 3):

1. Dynamic Paths: to avoid repetitive exercise and to represent different level of difficulty, intrinsic motivation (e.g. slope – higher effort)
2. Layers of information. It is composed by 4 different layers that contain different kind of information:
   a. Layer 1 - Background interface: It is the lower layer and shows the graphical appearance for each phase of the interaction, for example, during the exercise it shows the 3D world where the user is immerse .
   b. Layer 2 – Motivation: It includes the audio, music, messages and graphical coaching needed to guide the user through the exercise session.
   c. Layer 3 – Exercise info: It shows the exercise related parameters, such as, heart rate goals, rpm goals… It is always visible.
   d. Layer 4 – Performance info: It shows the data coming from the sensors and related to how the user is doing his exercises. It is always visible.



Figure 3. The application motivation messages.

InfoBooth

It is the software that connects to the central server and retrieves all the information needed to make an ActivityCoach FitnessBike session. That information includes personalization by user and general purpose data like media messages, music, and so on. It is also used to upload session results to the server.

Service Centre

It is the professional platform that provides the back-end services for the users. It includes the ActivityCoach Databases that store all the information about the users profiles, trainers profiles, training programs, motivation schemas, data gathered from sensors and results; a set of algorithms for training program generation and user status assessment and access communication, security, MyHeart portal, periodic events and questionnaires functionalities.

CONCLUSIONS AND RESULTS

A concept validation study was performed in a real (but controlled environment where the subjects experienced the use of the Activity Coach product over a long period of time.

The protocol used to test the AC product involved 15 subjects (male/female) involved during 3 sessions per week over 8 weeks.

Impact assessment test

It was done to determine if the training programs were effective in improving fitness status. The obtained results are:

- *Workload (watts)*: A significant improvement in the max workload of 13% after 4 weeks and 19% at the end.
- *Aerobic capacity (ml/min O_2)*: improvement of 27% (female) and 10% (male)
- *Lactate (La mmol/l)*: Significant reduction of over 41%, that means a better resistance to perform daily activities.
- *Perceived fatigue*: reduced by 14%

User experience and technical test

The test tried to understand what people experience when interacting with the product and evaluate possible technical issues.

After the test it was concluded that the InfoBooth was easy to use and the information provided was complete, but the users preferred to have everything included in the bike application. The Fitness Bike was very well evaluated concluding that the 3D environment was the correct approach and the information provided was very complete. The major number of flaws where found in the vest, being considered too tight and causing issues due to sweating and short-circuiting the intelligent textile.

Next steps involve the evolution of the product according to the feedback obtained from the users.

REFERENCES

Jörg Habetha, Dr. HArald Reiter. 2005. “MyHeart project IST 507816”http://www.hitech-projects.com/euprojects/myheart/home.html. Leaders of the MyHeart European project.

Alvaro Martinez, Pilar Sala. Deliverables 01-23 Internal documentation, Itaca-tsb.

Alvaro Martinez, Manolo Traver, “Development of Fitness Bike” Internal implemtation documents.

Pilar Sala, Sergio Guillem “Fighting Cardiovascular Diseases by Preventive Lifestyle and Early Diagnosis”

Computerized therapy in speech pathologies, at Research. Stroke forum. Milan, Dec. 13, 2007

XIMPEL INTERACTIVE VIDEO – BETWEEN NARRATIVE AND GAME PLAY

Anton Eliëns, Hugo Huurdeman, Marek van de Watering, Winoe Bhikharie
Intelligent Multimedia Group, VU University Amsterdam
email: eliens@cs.vu.nl

KEYWORDS
ximpel, interactive video, game development

ABSTRACT

In this paper we present the XIMPEL[1] interactive video platform. XIMPEL has been developed for the Clima Futura climate game, in which we use interactive video as a basis for scenario-driven games, with additional mini-games for elaborating on specific topics or tasks that arise during game play. XIMPEL has also been deployed in education, for creating short viral videos which allow for further interactive explorations. In this paper we will discuss the relation of XIMPEL with the Clima Futura game concept(s), describe the XIMPEL Interactive Video Format, and discuss the use of scenarios and storygraphs, in particular from the perspective of what interactive cinema and games have in common. We will also look at how the interactive video format can be deployed for scoring and diagnostics to provide meaningful feedback to the user, and the role argumentation rethorics can play in structuring information and interaction.

INTRODUCTION

Nowadays more and more institutions and organisations feel compelled to be present on one of the popular websites, like youtube.com, to present their message or mission and attract the attention of a wider audience. Commonly used formats are *viral* videos or games, demanding only casual attention, and meant to seduce their audience in buying or exploring additional offerings. However as observed by Alexander et al. (2007), in their Museum on the Web 2008 article Beyond Launch: Museum Videos on YouTube[2], lauching a video or game is one thing, but defining a communication strategy that attracts the attention of users over a longer period of time and creates a bond with potential clients is rather more demanding. Essential questions here are

- *relevance* – what is our message?
- *identity* – who are we?
- *impact* – why would anybody be interested?

And even more importantly, however well-meant and serious the message is, why would people pay any attention whatsoever to it, if it does not immediately arouse curiosity, and why would potential clients explore the further offering if the effort to do so is or may be seen as too demanding, given the fact that competitors are very likely one or two clicks away.
A very helpful set of criteria, summarized below, for distinguishing games from other (online) applications were presented to us in a workshop on educational games in a museum context[3]:

criteria

- *challenge* – relevance, feedback, confidence
- *curiosity* – cognitive or sensitive discrepancy
- *control* – contingency, choice, power
- *context* – intrinsic or extrinsic metaphor(s)

Whereas *context*, *challenge* and *control* seem to be the major parameters for modelling user actions in terms of, respectively, goals or topics and strategy and choices, or in other words scenarios with more or less well-established courses of behavior, *curiosity* seems a distinguishing factor not only for games but also for viral videos.
The notion of context may need some further elaboration, as it indicates the extent to which the game play or media format is intrinsic to the domain, that is inherently related in terms of content and metaphors, or extrinsic, that is relatively independent of the domain and available as a format that may be re-used over and over again for many different domains, as for example a memory game, simply by selecting the appropriate set of images.
Our XIMPEL Interactive Video Format resolves the tension between immediate attention and convenient exploration, by allowing viral clips to act as an interface to a rich repository of other (interactive) video material, as well as other interactive applications including scenario-based games and casual mini-games.

structure The structure of this paper is as follows. We will first discuss the relation between XIMPEL and the Clima Futura game concept(s). Then we will describe the XIMPEL Interactive Video Format in more detail, and discuss the use of XIMPEL in student projects *interactive multimedia* at our university. After looking

[1] ximpel.net
[2] www.archimuse.com/mw2008/papers/hart
[3] www.archimuse.com/mw2008/abstracts/prg_335001733.html

at the relation between (interactive) cinema and games, we will briefly indicate how scoring is dealt with and how user navigation, that is the actual choices made by a user, may be used to provide diagnostics, for example with respect to standpoints taken in the debate on climate issues. We will then explore how argumentation rethorics can be of help in organizing (a large body of) video fragments and imposing structure on interaction. Finally, we will look at future work we plan to do for Clima Futura and draw our conclusions.

GAME CONCEPT(S)

As more fully explained in Eliens et al. (2007b), Clima Futura[4] was explicitly meant to tackle the *pathos* of the media, of which Al Gore's *Inconvenient Truth* was only one example, and bring about a deeper, science-based awareness of the climate issue(s) with a younger audience, primarily high-school students in the age of 14-18 years old. Quoting Eliens et al. (2007b):

> The Clima Futura game addresses the issues of climate change, not altogether without *pathos* nor *ethos*, but nevertheless primarily focussed on bringing the *logos* of climate change into the foreground, in other words the scientific issues that are at play, and the science-based insights and uncertainties that may govern our decisions in the political debate.

Clima Futura is a turn-based game in which the player is confronted with a series of problems and disasters, for which actions are needed to counteract the possibly fatal consequences. All player actions have effect on a score in the result parameters (*people*, *planet*, *profit*). In addition, Clima Futura features *mini-games*, that may be selected on the occurrence of a game event, to acquire additional information, gain bonus points or just for entertainment. Examples of mini-games, are *negotiation with world leaders*, or a climate-related variant of Tetris. The game model for Clima Futura can conveniently be depicted, fig. 1, as three interwoven cycles of activity and events resulting from players' actions or occurrences within the climate simulation model, which keeps track of climate changes due to these events over (game) time.

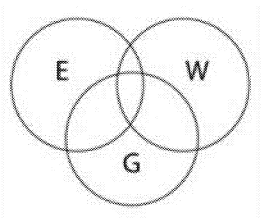


Fig 1. Game play, Simulation (W), Exploration

In summary, the Clima Futura game combines the following elements:

1. game cycle – turns in subsequent rounds (G)
2. simulation(s) – based on climate model (W)
3. exploration(s) – by means of mini-games (E)

Each of the three elements is essentially cyclic in nature, and may give rise to *game events*.
Early on in the development of the Clima Futura game concept, we decided to give interactive video a prominent place, not only in the exploration(s) supporting mini-games, but also as an interface for menu-selection and the choice of actions in response to game-events. In this paper, we will primarily focus on XIMPEL Interactive Video and only briefly indicate the game-oriented features of the XIMPEL platform in the section on future work for Clima Futura.

XIMPEL VIDEO FORMAT

In this section, we will present the basics of the XIMPEL Interactive Video Format, by presenting an example taken from the online tutorial[5] accompanying the XIMPEL distribution. Below, in the first fragment, a declaration is given for the subject *ExpertDebate*:

```
<subject name="ExpertDebate">
<longname>
   Discussion between Dorland and Kroonenberg
</longname>
<score>2</score>
<videos>
<video file="debate-01"></video>
<video file="debate-02"></video>
<video file="debate-03">
  <question>
     Van Dorland is against mitigation
  </question>
  <rightanswer>false</rightanswer>
</video>
<video file="debate-04">
  <question>
     Kroonenberg proposes adaptation to change
  </question>
  <rightanswer>true</rightanswer>
</video>
<video file="debate-05" repeat="true">
  <branchquestion>
     Are you worried about the climate?
  </branchquestion>
  <overlays>
  <overlay
    position="01" label="AntiMeasures"
    position2="31" label2="ProMeasures"/>
  </overlays>
</video>
</videos>
</subject>
```

[4] www.climafutura.nl

[5] www.cs.vu.nl/∼eliens/im/local/intro-iv.pdf

A basic video configuration file defines a playlist, wich consists of a number of *subjects*, containig video fragments related to that subject. Video fragments may include *questions*, which may be either transitory, allowing the user to obtain bonus points, or modal branching, giving the user the choice to switch to different *subjects*. Finally, there are *scores*, to reward the choice of particular subjects, or give bonus points for correctly answering transitory questions.
To complete te example, the continuation for the choice for *ProMeasures*, during the 5th fragement of the *debate* videos, is:

```
<subject name="ProMeasures">
<longname>
   Yes, and I want climate measures!
</longname>
<score>4</score>
<videos>
<video file="nature-01"></video>
</videos>
</subject>
```

Note that the 5th *debate* video fragment has the *repeat* attribute set to *true*, which means that the user has to make a choice either for the *ProMeasures* continuation or for the *AntiMeasures* continuation, which is listed below:

```
<subject name="AntiMeasures">
<longname>
   No, and I don't want climate measures!
</longname>
<score>3</score>
<videos>
<video file="mediocre-01"></video>
</videos>
</subject>
```

The example shows how *subjects* can be connected, to allow for meaningful branches and continuations. Obviously, to give meaningful information to the user, *long names* are needed, in addition to the symbolic names by which the individual subjects are identified.
In summary, the basic XIMPEL Interactive Video Format supports the following elements:

playlist element(s)

- subject(s) – with video fragment(s)
- video(s) – to present the material
- question(s) – transistory or modal branching
- overlay(s) – to indicate choice point(s) for branching
- score(s) – assigning (bonus) points

Some refinements of the format, not treated here, include the indication of time duration for the overlays, as well as more extensive options for defining the size and visual appearance of overlays, for example by using semi-transparant images.

SCENARIO & STORY GRAPH

Having used the XIMPEL Interactive Video Format for a limited number of Clima Futura productions, a real challenge presented itself when we decided to use XIMPEL as the technical platform for a two-week undergraduate course, with as a general assignment:

project interactive multimedia

> To design and develop a moderately complex multimedia application, with both educational and game elements, as part of a communication strategy for some particular goal or issue of societal relevance.

More specifically, the course aims to teach:

- *elementary web-based multimedia technology*
- *programming/tools for interactive animation and video*
- *first principles of information visualisation*
- *basic media and cummunication theory*
- *the design of an effective communication plan*
- *business and societal context of media deployment*

Since XIMPEL was still in a development phase, our major questions were whether the XIMPEL format would be sufficiently *expressive* for students to realize their creative aspirations, and, equally important, whether or not the students would stumble upon *technical issues* that, as developers, we had not foreseen.

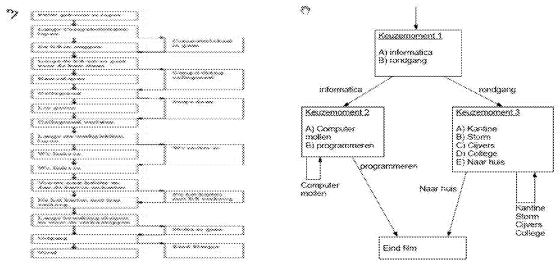
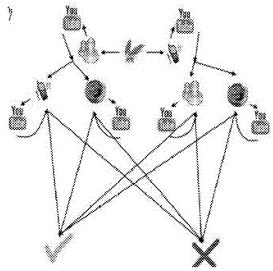


Fig 2. (a) Mario (107) (b) Tour (302)

In all respects the results of the course were very positive and exceeded our expectations. Both the viral clips that were produced, collected in the vumedia[6] channel at youtube.com, as well as the interactive videos[7], not only showed great variety, with some coming close to actually being a game (e.g. Mario, fig 2a), but were also surprisingly original and informative (e.g. Tour, fig 2b), due to the powerful graphics and video facilities of the flex/as3 platform.
An essential part of the assignment was to develop a scenario as well as a storygraph, where students were instructed, following Riedl & Young (2006), that: branching narratives are usually represented as directed graphs in which each node represents a linear, scripted scene, followed by a decision point. Arcs between nodes represent decisions a user can make.

[6] www.youtube.com/vumedia
[7] www.cs.vu.nl/∼eliens/media/view.html



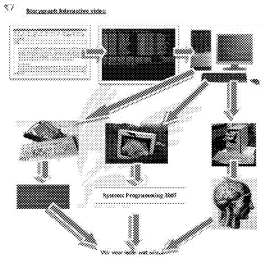
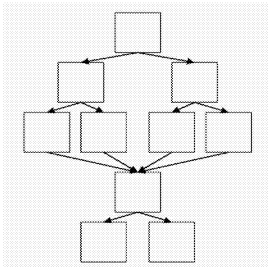


Fig 3. (a) SMS/MSN (205) (b) Course(s) (301)



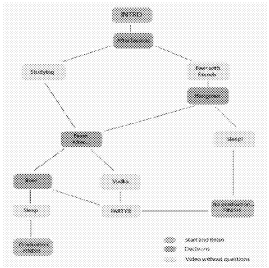


Fig 4. (a) Convergence (b) Divergence

Obviously, students had no difficulty in visualizing the XIMPEL configuration structure(s) in graphs and defining (sub) scenarios for the nodes of the graph. In effect, students showed great originality in presenting their designs, using either plain text or icons, fig. 2 and 3. It proved, however, to be more difficult for students to apply the *transitory questions* and even more so the *score* options in an effective and meaningful way.

During the course, for some of the technically more advanced students the need for extensions became apparent, not only for visual styling, but also functional extensions, with as in the case of the *Mario* application for example a real *SuperNintendo Mario* mini-game. To accomodate these wishes, students were allowed to experiment with the code, and in a later phase we will incorporate some of these extensions in the XIMPEL platform.

CINEMA AS GAME

Where, as we observe in Eliens & Ruttkay (2008), games are increasingly being used to produce video, using what is called *machinima*, the XIMPEL platform has been developed to enable the use of video as the carrier of game play, allowing for what we have characterized as *poor man's immersion* in Eliens et al. (2007b), by deploying the immersive realism of video, based on conventions of storytelling and film editing[8], Bolter and Grusin (2000). In particular, with the large amounts of material available on repositories like youtube.com, and the availability of affordable video cameras and video-enabled cellphones, authoring of video material comes at a fraction of the cost of developing virtual environments for platforms such as Second Life, see Eliens et al. (2007a), and the Halflife 2 SDK, see Eliens & Bhikharie (2006).

From film theory, Bordwell & Thompson (2001), we learn that narratives may be understood as: *a chain of events in cause-effect relationship occurring in time and space.* Obviously, giving the user control, as in interactive narratives, gives rise to new relationships, and defines additional challenges for the user, bringing the experience closer to game play.

Creating *branching structure(s)*, however, gives rise to new issues, and may even jeopardize authorship, in the following ways:

- *redundancy* – large amounts of video material are needed for branching, Sawhney et. al. (1996)
- *closure* – conflict of (user) control and coherence, Riedl & Young (2006)

However, problems of coherence can by and large be circumvented by including *vital story elements* in every possible navigation path, either by enforcing convergent paths, fig. 4a, or by adding redundant scenes necessary to bring about closure in the case of divergent navigation paths, fig 4b.

Following the definition given in Lindley (2005), stating that *a game is a goal-directed and competitive activity conducted within a framework of agreed rules*, we emphasize that the XIMPEL platform provides rich options to include additional challenges and options for the user to exert control. As a good example of a XIMPEL application that comes close to being a game, as well as conforming to the criteria listed in the introction, we have already mentioned *Mario*, which however falls short of being a game by lacking a sufficiently well-defined mission and the omission of a meaningful scoring strategy. Additional challenges, no doubt, may be introduced by realizing a smooth transition to mini-games as an extra option in modal branching questions.

score(s) & diagnostics Inspired by *game interaction pattern*, Björk & Holopainen (2005), in particular the *Score* patterns, we are extending the facilities for scoring to include support for incrementelly adjusting for example the combined result parameters $(people, planet, profit)$ of our climate game. that may conveniently be represented as vectors of features.

However, the facilities currently available already allow for interesting diagnostics, as for example the *moral profile* resulting from our Dante-inspired *Journey to Helf*[9] application.

[8] www.edgecodes.com

[9] www.cs.vu.nl/∼eliens/ximpel/dante

GAME RETHORIC(S)

In Eliens et al. (2007b) we already indicated the need for a more flexible way of including video fragments, and as the necessary background for such extensions we present here the major ideas underlying the Vox Populi system, Bocconi (2006). Vox Populi supports the classification and retrieval of video fragments based on that we may call *rethoric polarity*.
In Vox Populi, video fragments are annotated with meta-information to allow for searching relevant material, supporting or opposing a particular viewpoint. based on the users' preference, either a *propagandist* presentation can be chosen, expressing a single point of view (POV), a *binary commentator*, which shows arguments pro and con, or an *omniscient presenter* (mind opener), which displays all viewpoints. In the future development of XIMPEL we intent to include a *video content module*, that provides flexible access to the collection of video(s) by allowing for tag-based search and takes into account the rethoric content and polarity of the material. We believe that in this way we may provide the user with additional challenges, by presenting either conformant fragments or material that presents an antagonist position with respect to the users' observed preferences.

CONCLUSIONS

In this paper we have presented the XIMPEL Interactive Video platform, that allows for presenting *information*, by means of scenario-based interactive tours, and to provde the users with *game challenges*, by allowing choices in virtual story space, as well as transitory questions for the quick-witted user. An appealing feature of XIMPEL is the support for diagnostics, enabled by the score mechanism and by recording the users' navigation paths, which allows for meaningful feedback about the users' position and preferences with regard to the topics dealt with.
Observing the popularity of online flash video, Larson & Costantini (2008), our choice for the flex/as3 framework seems well-motivated. Moreover, due to the expressiveness of our XML video configuration format, which allows for a far more rich underlying structure than the recent extensions for interactive video introduced by youtube.com, XIMPEL seems to be an excellent vehicle for a wide range of game-like applications, including service management games, Eliens & Chang (2007). In the near future we further hope to realize a fully functional scenario-based version of Clima Futura, that not only provides the user with possible mission(s), but at the same time will also give access to the wealth of video material and information on climate issues, as collected in our Climate Portal[10].

[10] www.climateportal.nl

REFERENCES

Alexander C., Burnette A., Dark D. Hart D, Rossi J. , Minor N. (2007), Beyond Launch: Museum Videos on YouTube, In Proc. Museum on the Web 2008, Montreal, Canada

Björk S. and Holopainen J. (2005), *Patterns in Game Design*, Charles River Media

Bocconi S. (2006), *Vox Populi: generating video documentation from semantically annotated data repositories*, Ph.D. Thesis SIKS 2006-27

Bolter J.D and Grusin R. (2000), *Remediation – Understanding New Media*, MIT Press

Bordwell D., Thompson K. (2001), *Film Art: An Introduction*, 6th edition, McGraw-Hill

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

Eliens A. and Chang T. (2007), Let's be serious – ICT is not a (simple) game, In *Proc. FUBUTEC 2007*

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

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

Eliens A. and Ruttkay Z. (2008), Record, Replay & Reflect – a framework for understanding (serious) game play, submitted to GAME-ON 08, Valencia, Spain

Lisa Larson and Renee Costantini (2008), *Flash Video for Professionals: Expert Techniques for Integrating Video on the Web*, Sybex

Lindley E. (2005), Story and Narrative Structures in Computer Games, in Bushoff, Brunhild. ed. 2005. Developing Interactive Narrative Content: sagas/sagasnet reader. Munich: High Text

Riedl M., Young M. (2006), From Linear Story Generation to Branching Story Graphs , IEEE Computer Graphics and Applications, 26(3), 2006. Special Issue on Interactive Narrative.

Sawhney N., Balcom D., Smith I. (1996), HyperCafe: narrative and aesthetic properties of hypervideo, In: Proc. of the the seventh ACM conference on Hypertext, 1996

ONLINE GAMING COMMUNITIES – A PRACTICAL SHORTLIST

Marek R. van de Watering, Winoe Bhikharie, Hugo C. Huurdeman, Anton Eliëns
Intelligent Multimedia Group VU University, Amsterdam, The Netherlands
E-mail: rvdwate@few.vu.nl

KEYWORDS
Online communities, gaming, preliminary shortlist.

ABSTRACT

Relating to the shifting role of users in becoming active contributors to the design process, in this paper we look into online gaming communities. A preliminary shortlist of conditions for such communities to exist is given, and potential benefits for designers are listed. From these, five guidelines for creating such a community are proposed, which are to be examined in the light of setting up an online community for "Clima Futura", a game about climate change that is being developed.

INTRODUCTION

Thus, as designers we have managed to get our game adopted by a number of users. What comes next? Ideally, we would want these users to create a community; a living social context that revolves around our particular game.

In this paper, we will look into this social phenomenon, as it can help developers in (re)designing and refining an application by for example getting feedback from within the supporting community and lengthen the application's lifespan by ultimately, and under the right conditions, the community taking over the development role.
This examination is based on an initial and limited amount of literature that exists on online gaming communities, and from the author's own experience of active participation in and observation of such a community over a period of 8 years. First, we will look at how to define online communities. Next, the potential benefits for game developers are given, and the conditions for an online community to exist are examined. Based on this, we provide five initial guidelines for setting up an online gaming community, both in theory and from practice, the latter based on a game about climate change, "Clima Futura", and draw a conclusion.

DEFINING "ONLINE COMMUNITY"

To start with, Stolterman et al. (1997) define an online community as "a group of people trying to achieve something as a group of people that are using new information technology as a mean". People come to online communities seeking information, encouragement and conversation (Arguello et al. 2006). They gather around similar interests, they form communities as it lies in man's nature (Kihlman and Johansson 2000), and they contribute to getting quicker answers to their questions, to show their skills, and more conceptually, because of compassion and respect for others (Kollock 1999).

For those not familiar with the topic, think of a gardening convention: participants gather around a topic of their interest, show their skill, and help each other. Talk is not restricted to this particular topic, though; participants will also engage in all kind of small-talk not related to gardening. Newcomers, too, will come to the convention with particular questions, and if not responded to correctly, they will presumably not participate in future conventions.

BENEFITS FOR DEVELOPERS

Creating and supporting a "healthy" online community provides the developers of an application a number of benefits, that can be summarized as follows.
Firstly, contributing online communities lengthen lifetime of an application by expanding it in modifications, refining it with new rule sets, maps and exploring new modes of usage (in games, gameplay). A typical example is the Quakeworld community, which has driven the creation of new game clients, modifications and novel ways of gameplay from within its community, lengthening the game's lifespan from 1996 up to the present day.
Secondly, online communities can provide valuable feedback and ideas on which future development can be based (as is done within the Steam community).
Next to this, online communities provide support (help and possibly patches etc.) to the application's users.
Finally, although marginal, communities can act as talent-pools through which developers can contact talented contributors directly and thereby bypassing the normal pyramid-like model of working oneself up in the business (similar to Frith's pyramid model from popular music in Shuker 2001, pp. 114).
All of this is done at minimal costs for the original developers, as community members use their own resources for participating in the community. The question however is, if developers will put their resources into facilitating the necessary conditions for a community to form and to exist.

CONDITIONS FOR AN ONLINE COMMUNITY TO EXIST

Setting up an online community should first involve taking into account the necessary conditions for such a community to exist (McArthur and Bruza 2001), which are, in short: purpose, commitment, context and infrastructure.

First of all, a community needs a *purpose*, which can either be a practical purpose, such as limiting carbon dioxide emissions, or a shared interest, such as (parts of, or as an activity *playing*) a particular game.
Secondly, another condition is that of *commitment*, which stands for "the repeated, active participation committed to the purpose of the community" (McArthur and Bruza 2001). Here, there are two important interrelated factors in success: (1) that of the community's willingness to respond (reciprocity), and (2) that of the individual's willingness to stick with the group over time. If a first-time participant, a newcomer, does not get a reply within a certain amount of time (which no study has yet attempted to measure), he/or she is likely not to come back ever again.
Thirdly, a community needs a *context*; "attributes whose values remain more or less fixed or stable, thereby providing the secure foundations on which a community can be built." (McArthur and Bruza 2001) This is made up of a shared common knowledge within the community, its popular beliefs, and the social rules and policies it uses.
The fourth and last condition is that of an *infrastructure* being in place, which facilitates communication. This supporting, physical infrastructure is made up of technology and shared resources such as databases, web servers, chat programs, forums and websites.

GUIDELINES FOR CREATING AN ONLINE COMMUNITY

Based on these conditions, and partly following Kihlman and Johansson (2000), a number of practical guidelines for creating and supporting an online community can be defined:

- Create a *virtual meeting space*, which is stable, reassuring and interesting. This mostly means creating an appealing homepage, that attracts the target group. Next, it should be made possible for participants to *add content* and discuss it with each other, which can typically done by setting up a forum. Following our gardener convention metaphor, this forum should minimally contain newcomer information, technical support, content sharing and a place for off-topic discussion.
- Give the members *influence over the information*, which means that participants can contribute without censorship, and have the possibility to get to know others by being able to know who posted what message, and by being able to reply to that message.
- Give the members *influence over the environment*, so that they can modify it in coherence to their needs. Participants should be allowed to create new forum topics, choose a color theme, and even modify the shared interest. In many online communities, the shared interest is modifiable by the community members, making it a major topic of discussion and participation ("modding" for games, for example). This means however, that this shared interest should be made modifiable by its designers, for example by being made up of customizable modules.
- Give the community a *history*, so that discussions and other shared events are recorded and can be referred to in the future. Such a documentation acknowledges what the community can accomplish and acts as a guideline for social rules and behavior.
- Give the community a means of *punishing and regulation*; a small group of participants will inevitably break social rules, and a larger group will ask for drastic changes, all of which have to be regulated.

SETTING UP AN ONLINE COMMUNITY IN PRACTICE – CLIMA FUTURA

Following these guidelines, we can define how an online community for the previously mentioned climate game "Clima Futura" can be set up. Looking at the four conditions stated, we might say that climate change is an uniformly shared purpose. For commitment, context and infrastructure, we can follow our guidelines of setting up a community.
First, we should create a *virtual meeting place*. For Clima Futura, we developed a website that was to appeal specifically to the target group, without losing valuable scientific information (http://www.climafutura.nl), basing color and composition on popular gaming. On linked pages, it facilitates the necessary scientific background of the project and game, a way of contacting the design team (an electronic form) and it shows the design team visually (as if to say "you can be part of this team").
An important missing item is the forum, which should be built to facilitate a community. This can be done using widely used free, open-source packages and creating the topics mentioned above, and adding a topic on scientific background as it is a major part of the project (and game). The forum should be brought to the attention of potential participants by word-of-mouth, both online and in real life. Online, this might be done by posting on existing game forums, by using viral adverts, or by setting up a Clima Futura Hyves or Facebook account.
Next, we should give the members *influence over the information*. Using an open-source forum package, such as PunBB (http://punbb.informer.com) facilitates the posting without censorship, and enables the designers or forum regulators to define the level of configurability of the forum. Getting newcomers to stay can be done by creating newcomer topics at the forum, but also by facilitating tutorials (for example).
Now, we want to give the members *influence over the environment.* In designing Clima Futura, the project team first made an overview of existing games of similar nature. Only one out of the six reviewed games allowed players to contribute to the game, which made the team realize that this was a chance for Clima Futura. Not going into too much detail, Clima Futura is to be open-source and made using a modular architecture, in which participants can contribute video and image material, new scenarios and minigames (Eliëns et al. 2007). As a promotional tool, "design your own minigame" forms were handed out at major events, which allowed participants to share their ideas, and made them feel

contributors already. Ideally, this would be taken over by the forum or a digital form.
Being the fourth guideline we distinguished previously, we would want to give the community a *history*. For a game such as Clima Futura, one would want to implement a feature that records the actual game a player plays. This allows other players to share their moments of interest, and discuss them with other players. Recording programs such as Fraps (http://www.fraps.com) can be propagated to take over this role externally, being an alternative for the costly endeavor of creating recording functions within the game itself.
A forum should allow users to search into older topics, or even refer to those topics. Functions such as celebrating the 100th user, setting up polls, giving out awards for the best contributed movie, and competitions all help to facilitate "shared events" that ultimately create a shared history. Following the *Quakeworld* and *Elderscrolls* communities, if a willing base of contributors is established, a Wiki (such as http://wiki.qwdrama.com/Overview) can be helpful in sharing information on a more steady base.
Finally, we wish to give the community a means of *punishment and regulation.* Mostly referring to the forum, the design team should appoint administrators that regulate the forum in terms of topic creation, grouping of discussions, and punish evil-doers by for example (technically) banning them. Best being made up of members of the design team and the community, weekly or monthly meetings can help in defining rules and regulations for competitions, new topics, polls, etcetera (as is done on the Quakeworld.nu forum).

DISCUSSION

A division that is gaining importance, but which we have not discussed is that of the virtual environment the community uses for communication. This can be external to the community's shared interest; think of an online community about art such as Deviantart (http://deviantart.com). The shared interest does not support communication as a tool.
Next to being external to the community's shared interest, communication can be internally facilitated by it, in other words embedded in, the shared interest, such as a game or an application. Communication then is exclusively facilitated by the application's interface. This often leads to hybrid constructions in which in-game communication takes place and therefore supports in-game collaboration, which is extended by external facilities such as forums, or real-time external communication tools such as available in Valve's *Steam* (http://www.steampowered.com).
Within this paper, we have looked into the first of these. It is however difficult to establish the exact impact of virtual environments themselves allowing for community building, as most research on this topic is very recent and as of yet difficult to analyze unambiguously. Typically, this is an interesting topic for future research in general, but also regarding Clima Futura.
In general, it is important to emphasize that community building is difficult, as it is dependent on human beings that have a multitude of different online activities to choose from. Guaranteeing a community to be successful is therefore never possible, but future research might empirically test the importance of the aspects mentioned in this chapter to provide some sort of measure of successfulness.
Last but not least, the risk of creating an online community is that the community's views can clash with those of the game's development team. When an online community comes up, the developers in a sense lose part of their ownership of the application (Cox 2000). Therefore, the original developers should include a strategy for introducing changes without alienating the existing community, and should therefore be prepared to go into discussion with the community when necessary.

CONCLUSION

Within this paper, the notion of online communities has been discussed. This was done based on the observation that an increasing number of people is contributing to the design process of multimedia applications by providing content created within such communities.
Having assessed possible benefits, such as aiding newcomers and getting feedback from within the community, a number of preliminary guidelines for setting up an online community has been given. These guidelines can be summarized as creating a virtual meeting place, giving community members influence over the information that is shared, giving members influence over the virtual environment, giving the community a history, and finally, giving the community a means of regulation and punishment.
Being a proposal, important next steps in our research are those of evaluating and extending our work based on a larger body of literature, a selection of existing online communities, and on our ongoing venture of creating an online community for "Clima Futura".

REFERENCES

Arguello, J., Butler, B.S., Joyce, E., Kraut, R., Ling, K.S., Rosé, C., Wang, X. "Talk to me: foundations for successful individual-group interactions in online communities", In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, April 22-27, 2006, Montréal, Québec, Canada, 959-968.

Cox, T. 2000 "Online and Multiplayer Gaming - An Overview," In *Virtual Reality*, vol. 5, No. 4, 2000, 215-222.

Eliëns, A., van de Watering, M. R., Huurdeman, H., Bhikharie, W., Lemmers, H., Vellinga, P. 2007. "Clima Futura @ VU – communicating (unconvenient) science". In *Proceedings of GAME'ON 2007*. 20-22 November 2007. University of Bologna, Italy.

Kihlman, D., Johansson, M. "Am I alone? How to form a user group". In *Proceedings of IRIS 23: 23rd Information Systems Research Seminar in Scandinavia "Doing IT Together"*. 12-15 August, 2001, Uddevalla, Sweden.

Kollock,. 1999. "The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace". In *Communities in Cyberspace*, Marc Smith and Peter Kollock (Eds). London: Routledge, 220-239.

McArthur, R, Bruza, P. 2001. "The ABC's of Online Community." In *Proceedings of First Asia Pacific Conference on Web Intelligence*, Lecture Notes in AI, Pittsburgh, PA, Springer-Verlag.

Shuker, R. 2001. *Understanding Popular Music*. Second Edition, London: Routledge.

Stolterman, E., Croon, A. & Ågren, P.O. 1997. “Virtual communities - why and how are they studied”, Department of Informatics, Umea University, Sweden.

MMOG based on MAS: A Game-Independent Ontology

G. Aranda and C.Carrascosa and V.Botti
Departamento de Sistemas Informaticos y Computacion
Universidad Politecnica de Valencia
Camino de Vera s/n 46022 Valencia, Spain
{garanda,carrasco,vbotti}@dsic.upv.es

ABSTRACT

The number of players of Massively Multiplayer Online Games is increasing quite fast, and the mechanics of these virtual worlds are getting more and more complex. Due to the eminently distributed nature of these games and their growing necessity of modern AI techniques, it is time to introduce design methods that take advantage of the power of Multi-Agent Systems in order to face the challenges of designing a modern Massively Multiplayer Online Game. This article follows previous pieces of work in the line of Multi-Agent Systems and Massively Multiplayer Online Games research. A common ontology is presented to represent the information that agents store and share in these kind of systems.

Introduction

Massively Multiplayer Online Games (MMOGs) are an important focus of research, not only because they are economically attractive, but also because a MMOG involves many fields and a large amount of data that is generated by the interactions of many individuals: configuring a MMOG is a relevant source of research. In the field of AI, to model such systems and its dynamics is nowadays a very relevant task[14, 10].

A typical MMOG (such as World of Warcraft™[1] or Age of Conan™[2], two successful games of this kind) gathers a large community of concurrent players in a persistent virtual game world. These players take part in a common game which has no foreseeable end (i.e. the game is never completed or finished) and, along with some synthetic characters and creatures driven by AI, they populate the virtual world and create an online society with its own rules, landscape, economy and idiosyncrasy.

MMOGs are similar in nature to established online virtual communities, like Second Life™[3] or There™[4], as both have three-dimensional virtual worlds inhabited by the users' online personas and both have a virtual environment in which the activities of the users take place. However, virtual communities usually spot an open policy and an enforcement of freedom among its members. They do not completely lack rules or norms, but in general terms, users are left to do as they please and there are no short or long term goals for the users' personas to accomplish.

On the other hand, a MMOG also presents some degree of freedom to its players, but it is normally built around a set of rules, logics, conventions and other game mechanics that the game system itself enforces onto the players and that, together, represent the actual way of playing that particular MMOG and set what the game is about and how it becomes an enjoyable and challenging experience for the players. Besides, MMOGs have a tradition of integrating the development, evolution and upgrade of the players' *alter egos* into the game mechanics, making it a *de facto* mandatory long-term game goal for most players.

Following the ongoing research line in MMOG based on Multi-Agent Systems, this work introduces a common basic ontology that the agents that form the game can use to express semantic knowledge in common terms. Also this ontology is used as the base foundation for other, more specific, ontologies that can be built to represent knowledge for a given game or game genre. An example of this later kind of ontology is also provided. The language chosen to represent these ontologies is OWL-DL the standard content language for the semantic web, sanctioned by the W3C[5].

In section a summary of the recent works in this research line is provided. In section the MMOG ontology is introduced. In section a specific example deriving from the MMOG ontology is presented. Finally, in section some conclusions are presented along with some hints at future works.

Previous Work

This piece of work follows in the footsteps of previous research efforts like [1] and [2] in which games in general and MMOG in particular are researched as natural scenarios for agents and MAS. A MMOG (like most complex systems) can be seen as a system split into several layered subsystems, with each layer being relatively independent and taking care of one aspect of the whole MMOG experience. From the perspective of this work, a MMOG is split into three layers:

[1] http://www.blizzard.com
[2] http://www.ageofconan.com
[3] http://www.secondlife.com
[4] http://www.there.com
[5] http://www.w3.org

- **HCI Layer**: It is the *client-side* of the system, the part of the game running on the players hardware (PC, mobile phone, game console, ...). It is the user interface that the game provides to the player, i.e. the game *client* software, and it provides the player with a gaming experience (i.e. 3D graphics, sound, ...). On a computer, for instance, this software does not differ technically from other types of game software (i.e. offline games), and it is normally built around a game engine, giving it agent capabilities. Due to this layered design, the user is able to interact with the same game in different ways according to the device it is using, that is, the *Interface Agent* it interacts with. Its desirable features are efficiency and good usability.

- **Intelligent Virtual Environment (IVE) Layer:** It is the virtual representation of the game environment itself. It is part of the *server-side* of the system, the part of the game that runs mostly on the game provider's hardware, a controlled environment. The synthetic *place* and scenario where the game takes place: The virtual world. This world is independent of the type of game or simulation it must give support. The IVE layer falls outside the scope of this article, but it is thoroughly described in [5], presenting two kind of agents: *Simulation Controller Agents* and *Inhabitant Agents*. The first ones are in charge of controlling the environment (or pieces of it) and communicating with the graphical viewers along with the agents inhabiting such environment, that is, the *Inhabitant Agents*.

- **MMOG Layer:** It is a complex subsystem where all the game logics and mechanics are implemented and must be solved at run-time. It is independent of the IVE layer. It implements the game rules / norms controlling the game development. It is the *place* where all the game clients connect to play, and along with the IVE layer must facilitate game server scalability. In this line of research, this subsystem is seen as the core of a MAS and requires, at least, one agent platform as its foundation.

The main scope of this paper lies within the **MMOG Layer** that will be detailed in section .

The HCI Layer

The HCI layer is the user interface to the game that runs on the players' hardware. It comprises a full game client, but, for the sake of the focus of this article, only one part of the software is mentioned: the **InterfaceAgent**.
The InterfaceAgent is an agent which represents the interface given to the user to interact with the game. It is an agent that is executed in the **client** machine, that is, the players computer , and that makes it an *external* agent (out of the direct control of the platform). It is a light-weighted agent that provides the user an interface to the MMOG. the InterfaceAgent is capable of triggering a set of actions that output information messages towards the ProfileAgent. A user may have many InterfaceAgents installed: one on a computer, another on a mobile phone, etc. But since all of them represent the same user, all of them end up communicating with the same ProfileAgent.

The MMOG Layer

As stated before, the **MMOG Layer** is essentially a dedicated, open MAS which runs the game. This MAS uses agent technologies like agent services, Electronic Institutions[11, 7] and Agent Organizations[3, 12, 8, 4] to model some game mechanics, and translates the common issues and situations found in MMOGs into problems that can be solved using classic software agent features, such as agent interactions, agent communication protocols (like auctions or call-for-proposals), service-oriented computing, event-driven behaviors, role models, etc. A (rough) translation table between game concepts and agent concepts can be seen in table 1.
Like any other agentification process, one of the key ideas is to identify the agents and types of agents that will conform the system. In this case, the agents are based on the concepts and entities that form the whole game experience of a MMOG, and are explained in more detail in [2]:

- **ProfileAgent**: a personal agent which manages the player status and profile within the game community. A ProfileAgent is executed server-side, that is, inside the agent platform (which makes it an *internal* agent). It manages the player's preferences in the game world, which avatars the player uses and the **role** that the player plays in the system (*Spectator*, *Player* or *GameMaster*).

- **AvatarAgent**: an agent which represents an avatar of a human player within the game (a PC or Player Character). It is a persistent kind of agent: is not deleted and re-spawned often, it bears a long life cycle. It is the agent that holds the PC *stats* (server-side), and so, a malicious player cannot modify them locally (cheat). The AvatarAgent is the kind of agent that actually performs the actions for the player in the virtual world. It is a persona of the player inside the virtual world and it is a very active type of agent. Since it is an internal agent of the game and it is designed by the game developers, it will strictly follow the game rules, much like a *Governor* agent does not break the rules of an Electronic Institution.

- **NPCAvatarAgent**: an agent which represents an avatar of an AI-controlled character. It is similar to the AvatarAgent, as both populate the game world, but it does not obey nor represent a player in the game.

- **GameZoneAgent**: a kind of agent which implements the logics of the game environment and works as a nexus between the MMOG Layer and the IVE Layer's Simulation Controller.

Game Concept	Agent Concept
Players Clan	Agent Organization
Players Clan: Lord	Agent Organization Supervisor
Players Clan: Objective	Agent Organization Goal
Players Clan: Loot Share	Agent Organization Reward Policy
Game Zone	Electronic Institution
Game Zone Hub	Federation of Electronic Institutions
Client Software	Interface Agent *(not in its entirety)*
Player Character	AvatarAgent
Non-Player Character	NPCAvatarAgent
Game World	Intelligent Virtual Environment

Table 1: Translation between game concepts and MAS concepts

- **BankAgent**: an agent that stores information persistently for other agents and helps maintain the consistency and reliability of the system.

As summarized in table 1, this approach makes use of Agent Organizations and Electronic Institutions too. Agent Organizations are used to express what is known as player Clans or Guilds, i.e. player associations, which may or may not be permanent, and that bond players which have something in common. This commonness is usually a goal or set of goals within the game world, e.g. in a medieval fantasy game like World of Warcraft™ a clan may be formed to build an army to participate in a big-scale battle against opposing armies. In agent terms, these clans are really Agent Organizations within the agent platform. Each organization has its supervisors (which double as *Clan Lords* in the game world), its members (which are AvatarAgents), its goals and its reward policies. The later two are expressed within game terms using an ontology (more on this later in section).
Electronic Institutions, however, come handy for another task: modeling the game zones. A game zone is a segment of the virtual game world with its own name, theme and uniqueness. For example, in a medieval fantasy game which has a big game world representing an island, a game zone could be a castle called *'Hidden Palace'* and the dungeon below it. Game zones like that work essentially in a very similar way as Electronic Institutions do: there are clear entry and exit points, interconnected rooms, requirements for entering certain rooms, crowd limits, AI-controlled agents that play different roles, etc. In fact, there is a separate branch of the Electronic Institutions called the '3D Electronic Institutions'[7] that cover precisely this kind of virtual places. The whole game world is envisioned as a set of interconnected (or *federated*) Electronic Institutions. The main institution is called the *'Free Zone'*, an Electronic Institution with relaxed constraints and where the Player Clans may be formed.

A Common Ontology for MMOG agents

One issue left unsolved in previous works, and that this paper addresses, is the need for a common ontology for all these agents to be able to share information in a consistent way (i.e. they need to speak the same content language). It is clear that every game or game genre has its own mechanics, rules and players, but, in essence, all MMOGs share some common concepts, like player avatars. These common concepts can be expressed using a common set of conventions, a common knowledge base. Then, in order to express specific concepts for a given game or genre, custom ontologies that derive from the original one can be described. This common ontology is called **MMOG Ontology**[6]. Besides providing a good starting point for expressing knowledge in MMOGs, the use of a common ontology (or set of ontologies) allows for some interesting developments: first, some parts of the MAS (or even the whole system) can be reused or cloned from game to game with little to no change needed; second, it would be possible to interchange or carry over knowledge between different MMOGs, such as player information, statistics, avatars, etc; third, it settles a common way of working on a MMOG that transcends the current project and helps developer to establish work methodologies; and fourth, it allows for rapid game prototype creation using already available tools for developing ontologies.
The **MMOG Ontology** has been defined using the Web Ontology Language (OWL). More precisely, it is expressed using the OWL-DL (Description Logics) sublanguage for two reasons: one, some of the features of this sublanguage, like defining *one-to-many*-like relational properties, where needed; and two, the full set of the OWL language (called OWL-Full) is not deemed as being computable [6]. Even so, it is a quite light ontology, as it is composed of some base classes that have *datatype*[7] and *object* properties. A summary table of the ontology is presented in table 2. The ontology uses the *'mmog:'* prefix and it presents the following classes: **mmog:Avatar**, **mmog:Clan**, **mmog:GameBeacon**, **mmog:GameItem**, **mmog:GameZone**, **mmog:Goal**, **mmog:Profile**, **mmog:Requirement** and **mmog:RewardPolicy**. Let's review each one of them.

- **mmog:Avatar**: This is the generic class to represent an avatar within the game. It does not matter if its a player-controlled avatar (i.e. played by an AvatarAgent) or an AI-controlled avatar (played by a NPCAvatarAgent). Avatars are uniquely identified by their *hasName* property which ties each of them to a unique name string identifier within the game system.

- **mmog:Clan**: This class represents a Player Clan that the system implements by means of an Agent Organi-

[6] http://gti-ia.dsic.upv.es/ontologies/mmog.owl#
[7] A property whose value is a basic data type such integer, float, string, boolean, etc.

Ontology Class	Property	Type	Inverse Property
mmog:Avatar	atBeacon	*(single mmog:GameBeacon)*	–
mmog:Avatar	hasName	*(single string)*	–
mmog:Avatar	hasOwner	*(single mmog:Profile)*	*owns*
mmog:Avatar	isLordOf	*(multiple mmog:Clan)*	*hasLord*
mmog:Avatar	isMemberOf	*(multiple mmog:Clan)*	*hasMember*
mmog:Clan	hasGoal	*(multiple mmog:Goal)*	–
mmog:Clan	hasLord	*(multiple mmog:Avatar)*	*isLordOf*
mmog:Clan	hasMember	*(multiple mmog:Avatar)*	*isMemberOf*
mmog:Clan	hasName	*(single string)*	–
mmog:Clan	hasRewardPolicy	*(multiple mmog:RewardPolicy)*	–
mmog:Clan	maxMembers	*(single int)*	–
mmog:GameBeacon	hasName	*(single string)*	–
mmog:GameBeacon	placedAtZone	*(single mmog:GameZone)*	*containsBeacon*
mmog:GameItem	hasID	*(single string)*	–
mmog:GameItem	hasName	*(single string)*	–
mmog:GameZone	containsBeacon	*(multiple mmog:GameBeacon)*	*placedAtZone*
mmog:GameZone	hasName	*(single string)*	–
mmog:GameZone	hasRequirement	*(multiple mmog:Requirement)*	–
mmog:Goal	hasCondition	*(multiple owl:Thing)*	–
mmog:Profile	createdWhen	*(single date)*	–
mmog:Profile	hasContact	*(multiple mmog:Profile)*	*hasContact*
mmog:Profile	hasPlayerName	*(single string)*	–
mmog:Profile	owns	*(multiple mmog:Avatar)*	*hasOwner*
mmog:Profile	playsRole	*(single owl:oneOf {'Spectator' 'Player' 'GameMaster'}*	–
mmog:Requirement	atBeacon	*(single mmog:GameBeacon)*	–
mmog:Requirement	hasGoal	*(single mmog:Goal)*	–
mmog:Requirement	playsRole	*(single owl:oneOf {'Spectator' 'Player' 'GameMaster'}*	–
mmog:Requirement	requireClan	*(single boolean)*	–
mmog:Requirement	requireClanMax	*(single int)*	–
mmog:Requirement	requireClanMembers	*(single int)*	–
mmog:RewardPolicy	hasAction	*(multiple owl:Thing)*	–

Table 2: Summary table of the MMOG ontology

zation. It has properties referring its members, lords, goals, reward policies and constraints.

- **mmog:GameBeacon**: A *beacon* is a sort of *'named point'* of the virtual world. A marked position inside a mmog:GameZone which is referred by name. For example, the starting point or one of the exits.

- **mmog:GameItem**: An inanimate object within the game world uniquely identified by a code string. For example, in an adventure game, the items that the players virtually carry in their inventories are instances of a subclass of mmog:GameItem.

- **mmog:GameZone**: A class to represent a game zone. As stated earlier, a game zone is a segment of the virtual game world with its own name, theme and uniqueness. It holds one or more requirements to be entered.

- **mmog:Goal**: A collection of conditions expressed within valid semantic terms of the game. When all the conditions are met, then the goal is reached. For example, a condition could be the equivalent expression of *'Place object A in beacon XYZ'* or *'Avatar B is a member of Clan C'*.

- **mmog:Profile**: The profile of a player within the game community: The player's screen name along with all the information the system wants to keep from the player, the avatars the player controls, which role the player plays within the game system and the contact list for this player (i.e. the *'friends list'*).

- **mmog:Requirement**: A requirement to enter a given GameZone. It can specify terms regarding player clans attributes, goals or player role.

- **mmog:RewardPolicy**: A reward policy to apply once a goal has been achieved. It holds a collection of actions defined within game terms. Actions will normally be specified using terms from derived ontologies of specific games.

And so, this ontology serves as a foundation to build other, more specific ontologies to express the knowledge of a particular game or type of game. In the next section, an example of these kind of ontologies can be found.

An Example: RacingGame

Let's imagine an online racing game where players can assume the roles of drivers of different types of vehicles (Cars, Motorbikes and Trucks) that compete in online races (e.g. a simplified vision of the genre of games like Mini-Racing Online™[8] or Test Drive Unlimited™[9]). Players can also participate in an in-game virtual marketplace that allows them to buy and sell parts to upgrade their vehicles. There are basically two kinds of environments within the game: roads (where the racing takes place) and shops (where the selling and buying takes place). Besides, racers can find two types of objects on the road: gas cans (which replenish their gas tanks) and obstacles (which deal damage to a vehicle if it crashes into them). The developers of the game also wish to incorporate some sort of *'Top Racers Ranking'* or *'Leaderboard'*, and so, they need to collect some data from the races, such as how many races has a given player taken part in and how many of them did that player win.
With that game design in mind, a custom ontology called **'RacingGame'** built on top of the MMOG ontology can be defined. Let's start by refining the mmog:GameItem. Three different kinds of items have been identified: parts,

[8] http://www.miniracingonline.com
[9] http://www.testdriveunlimited.com

gas cans and obstacles, and such, three distinct subclasses of mmog:GameItem can be created to represent them: *VehiclePart*, *GasCan* and *RoadObstacle*. Let's start with the later two. GasCan needs an obvious datatype property called *hasFuel* that indicates how many fuel units are contained in the can. It also needs a property to indicate where is it placed. Looking at the MMOG ontology, the *placedAtZone* property does just that, and so, it is extended to cover the domain of GasCan too. The RoadObstacle class is very similar, but instead of the *hasFuel* property, it has one called *dealsDamage* which indicates how much damage is dealt to a vehicle if it crashes with the obstacle. The VehiclePart class is going to be the superclass of every subtype of part, so it can have the common properties of every part: *hasPrice*, which is a datatype property indicating how much does the part cost, and *canApplyTo*, a multiple property which indicates that the part can be applied (i.e. is *'compatible'*) to a certain subclass of VehicleAvatar (more on that later). The subclasses of VehiclePart are BodyWork, Engine and Wheels, each one representing one of the main parts of a vehicle and each one having some properties identifying its features and advantages and some other aspects, like the color of the BodyWork (see table 3).

Subclass	Property	Type
BodyWork	hasArmour	*(single owl:oneOf* {1 2 3 4 5})
BodyWork	hasColour	*(single owl:oneOf* {'Red' 'Black' 'White'})
BodyWork	hasWeight	*(single float)*
Engine	gasConsumption	*(single float)*
Engine	hasPower	*(single int)*
Wheels	hasGrip	*(single owl:oneOf* {1 2 3 4 5})

Table 3: Subclasses of VehiclePart

It is clear that the avatars in this game are going to be the vehicles that race, and that all of them have common properties that can be reflected using OWL properties. Let's subclass mmog:Avatar creating the new class **VehicleAvatar** that represents an in-game vehicle. Each vehicle is composed by the union of three parts: a bodywork, an engine and the wheels, and so, the VehicleAvatar class has three object properties called *hasBodywork*, *hasEngine* and *hasWheels* that represent this fact. It also has a numeric property called *maxSpeed* that represents the maximum speed that a vehicle can reach based on the properties of the parts that conform it. VehicleAvatar also has a series of subclasses starting with **BikeAvatar**, **CarAvatar** and **TruckAvatar**, each one representing one type of vehicle that the player can control. These classes refine the type of vehicle, but do not add any significant property.

Regarding zones, two kinds of mmog:GameZones are defined: the **CarPartsShop** and the **Road**. The CarPartsShop is the place where player can buy and sell car parts to upgrade their vehicles. It is envisioned to be implemented with an Electronic Institution that resembles those used in other market types (like the one used in the Fishmarket^{TM}system[9]). The other zone is the Road, which represents the virtual courses where the races take place. For this zone, two classes of mmog:GameBeacon have been defined two mark the start and end points of a race: **RaceStartPoint** and **RaceEndPoint**.

Last, the mmog:Profile must be refined in order to represent all the specific data for the races, and such, the **RacerProfile** class is created. It holds an object property called *favoritePlace* which points to the game zone where the player's avatars have spent more time since the moment the player joined the game community (represented by the property *mmog:createdWhen*). It also has a couple of numeric properties to represent the number of races the player has been involved in and how many of them did he actually win: *numberOfRaces* and *numberOfWins*.

Conclusions and Future Work

This work has continued the research line of MMOG Multi-Agent Systems by providing a starting point for building and exchanging semantic content between agents of these kind of systems. The existence of a basic ontology (a common starting point) to represent game-related knowledge is needed in order to achieve greater long-term goals like standardization, inter-systems communication or define a methodology to build these kind of system from the ground up.

In this work, such common ontology has been defined using the standard OWL-DL semantic language, as well as a small example of a more refined ontology for a specific game type. Both of them can be downloaded from the web and are ready for deployment and use at *http://gti-ia.dsic.upv.es/ontologies/*.

There is an ongoing path of work in this research line that involves agent services and service representation. Due to the use of OWL-DL in order to represent agent knowledge, it would be interesting to explore the possibility of using its derived language OWL-S[13] in order to represent the services that agents have to offer.

In other path of work, the agent interactions and protocols that conform the whole communication scheme of the system can now be defined and specified, as now there is an ontology for these agents to use and to express their semantic knowledge.

Acknowledgements

This work is partially supported by the PAID-06-07/3191, TIN2006-14630-C03-01 projects and CONSOLIDERINGENIO 2010 under grant CSD2007-00022.

REFERENCES

[1] G. Aranda, C. Carrascosa, and V. Botti. Intelligent agents in serious games. In *Fifth European Workshop On Multi-Agent Systems (EUMAS 2007)*. Association Tunnisienne dIntelligence Artificielle, 2007.

[2] G. Aranda, C. Carrascosa, and V. Botti. Characterizing Massively Multiplayer Online Games as Multi-Agent Systems.

Proceedings of the 3rd International Workshop on Hybrid Artificial Intelligence Systems (HAIS 2008, In Press), 2008.

[3] E. Argente, V. Julian, and V. Botti. Mas modelling based on organizations. In *Agent Oriented Software Engineering (AOSE08)*. Springer-Verlag, 2008.

[4] E. Argente, J. Palanca, G. Aranda, V. Julian, V. Botti, A. Garcia-Fornes, and A. Espinosa. Supporting agent organizations. In *CEEMAS'07*, 2007.

[5] A. Barella, C. Carrascosa, and V. Botti. Agent architectures for intelligent virtual environments. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 532–535. IEEE, 2007.

[6] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, et al. OWL Web Ontology Language Reference. *W3C Recommendation*, 10:2006–01, 2004.

[7] A. Bogdanovych, H. Berger, S. Simoff, and C. Sierra. Narrowing the Gap between Humans and Agents in E-commerce: 3D Electronic Institutions. *K. Bauknecht, BP oll, and H. Werthner, editors, E-Commerce and Web Technologies, Proceedings of the 6th International Conference, EC-Web*, pages 128–137, 2005.

[8] N. Criado, E. Argente, V. Julian, and V. Botti. Organizational services for spade agent platform. In *IWPAAMS07*, volume 1, pages 31–40. Universidad de Salamanca, 2007.

[9] G. Cuni, M. Esteva, P. Garcia, E. Puertas, C. Sierra, and T. Solchaga. MASFIT: Multi-Agent System for FIsh Trading. *ECAI*, 16:710, 2004.

[10] N. Ducheneaut, N. Yee, E. Nickell, and R. Moore. " Alone together?": exploring the social dynamics of massively multiplayer online games. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 407–416, 2006.

[11] M. Esteva, B. Rosell, J. Rodriguez-Aguilar, and J. Arcos. AMELI: An Agent-Based Middleware for Electronic Institutions. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 236–243, 2004.

[12] E. Garcia, E. Argente, and A. Giret. Issues for organizational multiagent systems development. In *Sixth International Workshop From Agent Theory to Agent Implementation (AT2AI-6)*, 2008.

[13] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, et al. OWL-S: Semantic Markup for Web Services. *W3C Member Submission*, 22, 2004.

[14] M. Matskin. Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games. *Eighth Scandinavian Conference on Artificial Intelligence: SCAI'03*, 2003.

EDUTAINMENT GAMES INCLUDED AS ACTIVITIES IN THE SUMMER SCHOOL OF THE TECHNICAL UNIVERSITY OF VALENCIA

M. Carmen Juan, David Furió, Ramón Mollá, M. José Vicent, Roberto Vivó
Instituto Universitario de Automática e Informática Industrial, Universidad Politécnica de Valencia,
Camino de Vera, s/n. 46022 Valencia, Spain
E-mail: mcarmen@dsic.upv.es

Miguelón Giménez
Escola d'Estiu, Universidad Politécnica de Valencia, Camino de Vera, s/n. 46022 Valencia, Spain

KEYWORDS
Edutainment, Augmented Reality, game, recycling, endangered animals

ABSTRACT

Climate is changing and we can actuate in order to reduce this change. One way of achieving this is to recycle and making people aware of this fact. Due to different factors, several species of animals and plants are endangered. In this paper we present four games for making children aware of these two subjects. Three games use Augmented Reality. Children can learn endangered animals using two games and how to recycle in the other two games. These games have been included as activities in the Summer School of the Technical University of Valencia. 160 children were involved in these activities. We are still analysing the data and in this paper only the analysis of one question has been included.

INTRODUCTION

Many initiatives have been/are being proposed for trying to actuate in order to reduce climate change. In this paper we propose the use of Augmented Reality (AR) for developing games with this purpose.

The games were developed for being used as activities in the Summer School of the Technical University of Valencia. This Summer School is for children who parents or relatives work or study in the Technical University of Valencia. At about 2,200 children attend to this summer school every summer. Every summer there is a motto and many of the activities are focussed on this motto. The motto of this year was 'The earth planet'. Therefore our AR games deal with subjects related to it such as recycling and animals that are in risk of extinction. We collaborated with the Summer School staff for defining the content of the games.

AR has already been applied for learning, entertainment or edutainment and other fields have also used it such as: military; medicine; engineering design; robotic; telerobotic; manufacturing, maintenance and repair applications; consumer design; psychological treatments, etc. (Azuma, 1997; Azuma et al., 2001).

In this introduction we only cite AR applications for learning/entertainment. We focus on AR applications for PC, but there are also several games for mobile phones. For example soccer AR games (Rohs, 2007; Geiger et al., 2004) or tennis AR games, (Hakkarainen & Woodward, 2005; Henrysson et al., 2005). The aim of citing these applications is to highlight that several AR systems have been developed for learning and entertainment, the variety of applications and there has not been presented any system similar to the presented here. Moreover, few of them have included evaluation.

We start with AR application for learning purposes. Construct3D (Kaufmann, 2004). He developed an AR system for constructing 3D geometries. It was designed for learning mathematics and geometry. Construct3D was tested with 14 students from two high schools in Vienna. Results from two evaluations showed that Construct3D was easy to use, required little time to learn and encouraged learners to explore geometry. Shelton & Hedley (2002) presented an AR system for learning the relation between the earth and the sun for geography students. Mixed Reality Lab of Singapore (www.mixedrealitylab.org) has developed several AR systems such as: an AR system for learning the sun system or an AR system for learning how the plants germinate, reproduce or perform photosynthesis. It is also possible to learn how to play the guitar using AR (Cakmakci & Berard,.2003). Billinghurst et al. (2001) presented the Magic Book. It seems like a normal book, but in the pages are markers, when the system recognizes a marker, an image is shown or a story starts. Mixed Reality Lab also presented a similar system. Following this idea, but using a different interface, the Magic Story Cube was presented (Zhou et al., 2004). A cube is used as a tangible interface that is folded or unfolded and depending on the markers that are visible the story is different. In this case, the story of the Noah's ark was presented. With the same idea of a tangible interface using a folded/unfolded cube, Juan et al. (2008) presented the interactive storytelling of the Lion King. These two last works are quite related to one of the games presented here, the magnetic cubes game. All three use cubes as tangible interfaces. Larsen et al. (2005) presented an AR system for learning how to play billiards. The most highlighting aspect of this system is the user interface. The user only touches over the billiards table to interact with the system.

Next, we present some entertainment applications. One of the first AR games was the ARQuake (Piekarski & Thomas, 2003) that is a game that is played in the exterior. Mixed Reality Lab has also presented AR games such as: Jumanji Singapore (Zhou et al., 2004) that is a tangible table game. The game uses the Singapore map as a table game; or an AR version of the traditional Pacman (Cheok et al., 2004). This game is played in the exterior and there are two types of players, Pacman and the ghost. Matyscok et al. (2004)

presented an AR game for playing tenpin bowling. In Kobito - Virtual Brownies (Aoki et al., 2005), the interface is a box and the animations (kobitos) appear in front of it. Kobitos can try to move the box on the opposite side. The box can remove kobitos from the table and the AR visualization is presented on a frame that is over the table.

DEVELOPED GAMES

Magnetic cubes

Hardware

The game requires a camera to capture the real world in order to determine where the virtual elements will exactly have to be drawn. We used an USB colour camera. To be exact a Logitech QuickCam Pro for Notebooks was used. After processing the captured image, the game obtains the real camera position and orientation relative to physical markers, and determines where the virtual elements have to be drawn. A Head-Mounted Display (HMD) was used as visualization system. We used 5DT HMD.

Software

We programmed the game using OsgART (http://www.artoolworks.com/community/osgart). It is a C++ library that allows developers to build AR applications using the rendering capabilities of Open Scene Graph (OSG) and the tracking and registration algorithms of ARToolKit.

OSG, http://www.openscenegraph.org, is a set of open source libraries that primarily provide scene management and graphics rendering optimization functionality to applications. It is written in portable ANSI C++ and uses the standard OpenGL low-level graphics API.

ARToolKit is an open source vision tracking library that enables the easy development of a wide range of AR applications. It was developed at Washington University by Kato & Billinghurst (1999). The required elements for the application are: a USB or FireWire camera, and a marker. Markers are white squares with a black border inside of which are symbols or letter/s. ARToolKit uses computer vision techniques to obtain the position and orientation of the camera with respect to a marker. Virtual elements are drawn on these markers. ARToolKit includes code for image acquisition, the detection of markers in the video stream, the calculation of the three-dimensional position and orientation of the markers, the identification of markers, the computation of the position and orientation of the virtual objects relative to the real world, and the rendering of these virtual objects in the corresponding video frame.

The virtual elements that appear in the game are images and videos of endangered animals.

Description of the system

Before using the game, the player watches a video that introduces the Red List and its categories; and how to play is also explained.

The Red List is published by the International Union for the Conservation of Nature and Natural Resources (IUCN). The IUCN is an international organization dedicated to natural resource conservation. The IUCN publishes the Red List of Threatened Species. This list was created in 1963 and it is the world's most comprehensive inventory of the global conservation status of plant and animal species. Whenever possible, information on the Red List is updated on the web site annually. A full analysis of the data held on the Red List is published once every four years. There are nine categories in the IUCN Red List: Extinct, Extinct in the Wild, Critically Endangered, Endangered, Vulnerable, Near Threatened, Least Concern, Data Deficient, and Not Evaluated. In the developed games, we have included the following categories: critically endangered, endangered and vulnerable. More information can be found in http://www.iucnredlist.org/.

In our game the player is asked for finding an endangered animal. The player uses three magnetic cubes. The cubes are placed one next to the other. The middle cube has 4 markers (four consecutive faces). The right cube has 2 markers (two opposite faces). Combining the faces of these two cubes it is possible to visualize 8 different animals. If the player thinks he has found the right animal, he has to place the third cube on the left by choosing the face where "*" marker appears. If the animal is right, the game asks the player if he wants to watch a video about the found animal. If the answer is yes, the video appears over the two right cubes. The player can get closer to the two cubes and look as near as he desires the video or the images. Figure 1 shows a player using the three cubes and looking the Iberian eagle. Figure 2 shows the same player as in Figure 1, but she is watching the video of the Iberian eagle.

Finding pairs

Hardware and Software

The software used was the same as in the previous game. We also used the same USB camera. The only difference was the visualization system. In this game the visualization system was a table with a transparent glass under where a monitor was placed. That is, the player played the game in the part of the table that was near him and just next was the monitor where the augmented image was shown, see Figure 3.

Description of the system

As in previous case, before using the game, the player watches a video that introduces the Red List and its categories; and again how to play is explained.

This is the typical game where the player has to find pairs. The player has to turn two pieces in order to find a pair. This step has to be repeated meanwhile a pair has not been found. Till here this is the description of a typical game for finding pairs. The difference is that the images do not appear over the pieces, the images of the animals appear over the pieces but on the screen that is in front of the player. Figure 3 shows a player playing the game. In this Figure it is possible to see that over the table near the child are the markers and in the monitor that is in front of the child appears the augmented game. Figure 4 shows the visualization of a step of the game where the player is placing the marker near the camera. If the player finds a pair, the game offers the possibility to watch a video about the found animal. In this video the characteristics of the animal, its classification in the Red List and the causes of extinction are listened.



Figure 1. Player using the three cubes



Figure 2. Player watching the video about the Iberian eagle

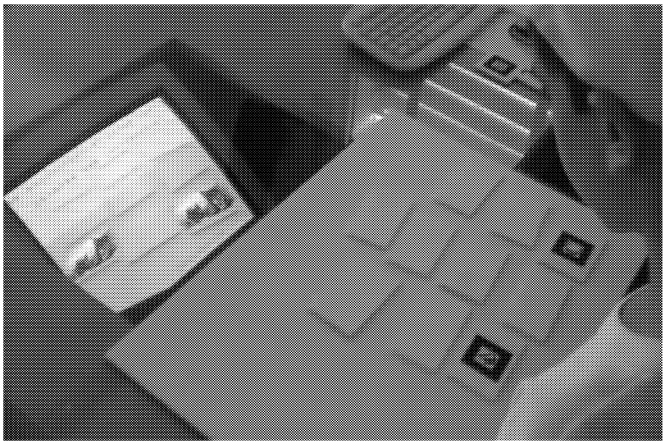


Figure 3. Child playing the pairs game



Figure 4. Looking near a crocodile over a marker



Figure 5. Basic Mobile game

Basic mobile game

Hardware and Software

The hardware used was a mobile phone, Nokia N95 with 8Gb. The game was programmed in Java, using J2ME.

Description of the system

Before using the game, the type of residues that can appear in the game and the colour selected for the recycling bins are presented to the player. Moreover, instructions for playing are also given.

The game recycles 6 types of residues: Papers; Plastics, bricks; Batteries; Glass and crystals; Electronic components; and other organic materials.

This game is divided into three levels. In the first level there are only two recycling bins and only two types of residues. In the second level there are three recycling bins and in the third level there are four recycling bins. In each run the type of residues are randomly selected among 6 possibilities and the residues that are going to appear are also randomly selected. The residues appear on top of the screen and they are falling down. The player has to place them into the right bin. In this case he will add points and on the other hand, he will subtract points. From time to time a question appears and the player has to answer. As in previous case, answering correctly will add points and on the other hand, points will be subtracted. Figure 5 shows an image of this game

AR mobile game

Hardware and Software

As in previous game the mobile phone used was a Nokia N95 with 8Gb. We developed our game using Microsoft Visual Studio 2005. Developing and running PC applications is very easy using this development environment. But for running an application simulating its running in a selected mobile phone, an emulator of the mobile phone is required. For including this type of tools we used the S60 Platform SDK, 3rd Edition. For the AR facilities, we ported ARToolKit onto a mobile phone running on Symbian OS and Series60.

Description of the system

As in previous case, before using the game, the type of residues that can appear and the colour selected for the recycling bins are presented to the player; and instructions for playing are also given.

The AR Mobile game is similar to the previous one, but it uses AR. In this case, we use markers that are placed in the room where the play has to be held. Two types of markers are used. One for the objects. All objects appear over the same marker, but depend on the level and the play the object is different. The bins appear over four different markers. In the first level only three markers are used, one for the objects and two for the bins. As in previous case, the bins and the objects are randomly selected. The bins and objects are similar to the bins and objetcs that appear in the previous game. Figures 6 and 7 show two images of a game. In Figure 6 it is possible to see a cardboard box over the object marker. In Figure 7 it is possible to see a step in the game where the player is in the second level and he is using three recycling bins.

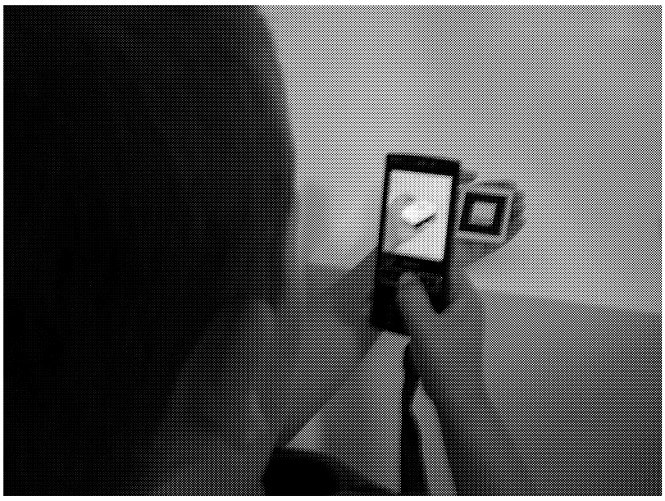


Figure 6. Player using the AR mobile game

RESULTS

The games were included as activities in the Summer School of the Technical University of Valencia.

Parents signed an authorization for letting their children to participate in these activities.

160 children participated in these activities. 32 played the game for finding pairs, 45 played the magnetic cubes game and 83 played the mobile games.

The games were evaluated from different aspects: engagement, fun and sense of presence. As the games are for recycling/knowing endangered animals, we have also checked its perceived value; children's attitude, motivation and intention to change respect to these two subjects. In the endangered animals game the games were compared with a real one. In the case of the mobiles, children used the normal and the AR game.

In the experiments, participants were counterbalanced, that is, half of the participants used first the 1) game and later the 2) and the another half used first the 2), and later the 1). That is, in the case of endangered animals 1) means the AR game and 2) the real one. In the case of recycling, 1) means the basic game and 2) the AR game.

This paper only describes some characteristics of the games and includes some preliminary results. We are still analysing the data.

One of the questions asked after using each game was: I enjoyed playing this game. Score from 1-7. The mean using the AR magnetic cubes game was 6.6 and using the real one was 5.6. The mean using the AR pairs game was 6.74 and using the real one was 6. The mean using the AR mobile game was 6.5 and using the normal mobile game was 6.2.

Related to the two first games (endangered animals), student t tests for this question indicate that there are significant statistical differences. That is, children did not have similar level of enjoyment playing the AR games and the real games. And they had more enjoyment using the AR games.

On the other hand, related to the mobile phone games, student t tests for this question indicate that there is not significant statistical difference. That is, children had similar level of enjoyment playing the AR game and the normal game. Altough the mean for the AR game is higher.

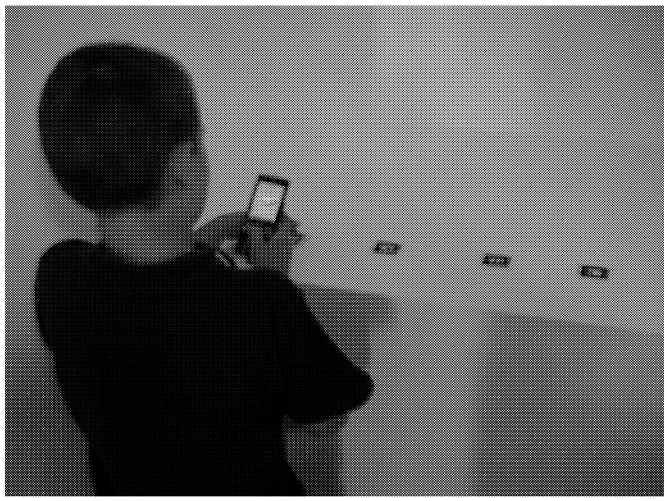


Figure 7. Player using the AR mobile game

CONCLUSIONS

We have presented four edutainment games. Three of them are AR games. We validated our games with 160 children of the Summer School of the Technical University of Valencia. We are still analysing the data and in this paper only one

question has been included (I enjoyed playing this game). Student t tests for this question indicate that there are significant statistical differences for the endangered animal games and there is not significant statistical difference for the mobile phone games. That is, children had more enjoyment using the AR games for the endangered animals. But, children had similar level of enjoyment playing the AR game and the normal game. After we have finished our analysis, further conclusions would be presented.

As mentioned in the introduction, few systems have been evaluated or they have included a very superficial evaluation. Therefore, more work has to be done in the evaluation of educational AR/VR systems. For this evaluation a number of technical, orientational, affective, cognitive, pedagogical and other aspects could be considerered.

Finally, we believe this type of edutainment games has a great potential and we hope soon they will be introduced in our society as games for children.

ACKNOWLEDGEMENTS

We thank:

- Giacomo Toffetti for developing the magnetic cubes game as Final Career Project
- Marta Carrizo for developing the pairs game as Final Career Project
- Carlos Fuster for developing the normal mobile game as Final Career Project
- The staff of the Summer School of the Technical University of Valencia for its collaboration
- Leila Alem and Peta Ashworth for their help in the design of the questionnaires
- Parents of the children of the Summer School of the Technical University of Valencia for signing the agreement for letting them to participate in these activities
- Children for participating in these activities
- 'Facultad de Informática' of the Technical University of Valencia for letting us the room, the table and the monitor used in pairs game activity
- DSIC for letting us the seminar used in the magnetic cubes activity and the room used in the mobile activity

REFERENCES

Aoki, T., Ichikawa, H., Asano, K., Mitake, H., Iio, Y., Ayukawa, R., Kuriyama, T., Kawase, T., Matumura, I., Toyama, T., Matsushita, T., Hasegawa, S. and Sato, M. 2005. "Kobito-Virtual Brownie-Virtual creatures interact with real objects and real people", *SIGGRAPH'05*, vol.1, n. 1, 7.28-7.29, http://rogiken.org/vr/index.php?plugin=attach&refer=FrontPage&openfile=kobito_siggraph2005.pdf

Azuma, R. T. 1997. "A Survey of Augmented Reality". *Presence: Teleoperators and Virtual Environments*, Vol. 6, N. 4, pp. 355 – 385

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. and MacIntyre B. 2001. "Recent advances in augmented reality", *IEEE Computer Graphics and Applications*, 21, 34–37

Billinghurst, M., Kato, H. and Poupyrev, I. 2001. "The Magic Book-Moving Seamlessly between Reality and Virtuality", *IEEE Computer Graphics and Applications*, May/June, pp. 1-4

Cakmakci, O. and Berard, F. 2003. "An augmented reality based learning assistant for electric bass guitar", *10th International Conference on Human - Computer Interaction*, http://students.creol.ucf.edu/ozan/arbass.pdf

Cheok, A.D., Goh, K.H., Liu, W., Farbiz, F., Teo, S.L., Teo, H.S., Lee, S.P., Li, Y., Fong, S.W. and Yang, X. 2004. "Human Pacman: A mobile wide-area entertainment system based on physical, social and ubiquitous computing", *Conference on Advances in Computer Entertainment* Technolog (ACE'04), Poster N. 8

Geiger, C., Paelke, V. and Reimann, C., 2004. "Mobile Entertainment Computing", *Lecture Notes in Computer Science*, 3105, 142-147

Hakkarainen, M. and Woodward, C. 2005. "Symball-Camera Driven Table Tennis for Mobile Phones", *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005)*, pp. 321-324

Henrysson, A., Billinghurst, M. and Ollila, M., 2005. "Face to Face Collaborative AR on Mobile Phones". *International Symposium on Augmented and Mixed Reality (ISMAR'05)*, pp. 80-89

Juan, M.C., Canu, R. and Gimémez, M. 2008. "Augmented Reality Interactive Storytelling Systems using tangible cubes for edutainment", *The The 8th IEEE International Conference on Advanced Learning Technologies (ICALT'08)*, pp. 233-235

Kato, H. and Billinghurst, M. 1999. "Marker tracking and HMD calibration for a video-based augmented reality", *2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pp. 85-94, http://www.hitl.washington.edu/artoolkit

Kaufmann, H. (2004) Geomtry Education with Augmented Reality, Dissertation, march

Larsen, L.B., Jensen, R.B., Jensen, K.L. and Larsen, S. 2005. "Development of an automatic pool trainer", *Conference on Advances in Computer Entertainment Technology (ACE'05)*, pp. 83-87

Matyscok, C., Radkowski, R. and Berssenbruegge, J. 2004. "AR-Bowling: Immersive and realistic game play in real environments using Augmented Reality", *Conference on Advances in Computer Entertainment Technolog* (ACE'04), short paper N. 11

Piekarski, W. and Thomas, B. H. 2003. "ARQuake - Modifications and Hardware for Outdoor Augmented Reality Gaming", *4th Linux Conference Australia*, http://downloads.planetmirror.com/pub/lca/2003/proceedings/papers/Wayne_Piekarski/Wayne_Piekarski.pdf

Rohs, M. 2007. "Marker-Based Embodied Interaction for Handheld Augmented Reality Games", *Journal of Virtual Reality and Broadcasting*, 4, no. 5.

Shelton, B.E. and Hedley, N.R. 2002. "Using Augmented Reality for Teaching earth-sun relationships to undergraduate geography students", *First IEEE International Augmented Reality Toolkit Workshop*, http://depts.washington.edu/pettt/papers/shelton-hedley-art02.pdf

Zhou, Z., Cheok, A.D., Pan, J. and Li, Y. 2004. "Magic Story Cube: an Interactive Tangible Interface for Storytelling", *International Conference on Advances in Computer Entertainment Technology (ACE 2004)*, poster N. 10

Zhou, Z., Cheok, A.D., Chan, T. and Li, Y. 2004. "Jumanji Singapore: An Interactive 3D Board Game Turning Hollywood Fantasy into Reality", *International Conference on Advances in Computer Entertainment Technology (ACE 2004)*, poster N. 9

Humming in Facebook:
An overview of Social Network platform architectures

Ken Newman
Academy of Digital Entertainment
NHTV, Hopmaanstraat
Breda 4810 Netherlands
E-mail: Newman.K@nhtv.nl

KEYWORDS
Social networks, architecture, REST, games, play.

ABSTRACT

This paper reviews the emergence of the social network developer platforms, and the responses of the developers and critics these platforms. Social network application platforms such as Facebook's developer platform have brought a lot of new opportunities for game developers. The platforms themselves and the architecture on which they are built have at the time of writing assembled themselves into two camps, Facebook vs OpenSocial. At a more abstract level again the network application architectures are also divided into two camps - REST vs WebServices architecture. These delineations are reviewed in lay terms and then the REST architecture of the Facebook developer platform is examined with three examples each differing in complexity; a basic Facebook canvas-callback architecture, a Flash-based application, and a Flash-based application implementing streaming video – and details the development of an example of such an application, the Facebook Hummit game.

RISE OF SOCIAL NETWORK PLATFORMS

All the statics of social networks point to astonishing growth in the past five years, and especially in the months from May 2007 to May 2008, as the major social networks have released their application development platforms. The six largest of the social networks are Facebook, MySpace, LinkedIn, Friendster, Plaxo, and Hi5, and all fulfill similar functions though they have evolved with some differences in the user demographics (age, gender split, and multipl-network-membership patterns) (Rapleaf 2007).

In May 2007 Facebook became the first of the Social Networking sites to relase a developer platform allowing developers to create applications which visually run inside a Facebook environment on a *canvas* page, and accessing Facebook data such as the user's profile and the friends lists. The response was staggering with thousands of applications flooding in as developers realized what was possible – instant access to potentially millions of users for their application. In one instant Facebook became the best audience to write for and the best distribution method for developers, as the traditional publishers and distribution gatekeepers were by-passed. Facebook was a completely new concept in distribution - a meritocracy where the distribution depended solely on how much the users wanted to share the application with friends. Developers embraced the opportunities and audiences loved it. Facebook outgrew all it's competitors in this period.

The Facebook platform API (Application Programming Interface) had, and still does have, a number of prorietory features, often to do with security. Developers register and apply for a numeric key for each application; the pages are marked up with a proprietory set of mark-up tags (FBML) and data queries are performed through a reduced set of SQL syntax (FBQL). There are also a number of apparently arbitrary restrictions on functionality, for example a Flash flie embedded with the proprietary FBML tag <fbswf> would not allow the Flash file to communicate with scripts on it's host page. Critics of the Facebook platform complained about,

"the costs of learning yet another markup launguage for every new social network platform, and taking developer time in creating and maintaining the code. " (Tech Crunch, 2007).

Reeling from the astonishing success of the Facebook platform, and also responding to the prorietory elements of Facebook's platform, Google, in November 2007, proposed a universal platform specification for Social Networks which they called OpenSocial. The platform consisted of a data schema and API calls to handle users, the friend networks, and events/activities. The OpenSocial platform did not specify any proprietory tags or queries, but opted instead to use standard HTML and JavaScript. MySpace, LinkedIn, Hi5 and Plaxo signed up as hosts for the OpenSocial platform.

Critics of the OpenSocial platform argued that it was merely a Google widget specification (Googel Gadgets); that it was just as prorietory as the Facebook platform; and that it's intended purpose would be better served by an enhanced version of the vCard Specification (Schofield, 2007). The portability of the format, which it's advocates considered one of it's main strengths, also raised concerns among the platform watchers.

OK, what if I am a friend of someone on MySpace. Cool, my name and picture appears on their friends list and anyone can see it. But what if this MySpace friend joins a PornSpace social network site and wants to import his friends list to that site? Now my name and picture shows up on his PornSpace page as a friend of his? Hey, wait a minute, I didn't agree to that. (Dodge, 2007)

Nevertheless LinkedIn launched it's developer platform in Nov 2007, MySpace in Feb 2008, and Hi5 in April, each

claiming some level of adherence to the OpenSocial specifications. Bebo, the third largest social network, broke ranks in Dec 2007 by launching it's platform to conform not with OpenSocial but with Facebook specification. At the time of writing the big Social Networks are split into the two camps: the Facebook platform and the OpenSocial platform.

PLAYFUL BEHAVIOR IN FACEBOOK

Two months after the Facebook platform launch Tech Digest magazine's top 10 Facebook applications listed such apps as:

1. TopFriends (flatter or insult your friends),
2. Tetris Tournament,
3. Flixter (movie reviews and quizzes),
4. SkypeMe (VoIP from within the full Facebook profile),
5. What's Your Stripper Name?

At the time of writing twelve months later the Facebook applications with the most users include SuperWall, SuperPoke, HotorNot and Texas Holdem Poker, as well as some that have been popular since the beginning such as TopFriends Flixter and iLike. While there appears to be a growing number of serious themes the majority of users of Facebook seem to be using it to engage in playful activity with their real world friends, a sort of extended way of conveniently maintaining friendship by spending time together.

Some of these applications are games in a formal sense of having an identifieable game circle, rules, objectives, and an outcome (Huizinger, 1938). Poker and Tetris are unquestionably games while 'What's Your Stripper Name?' is probably better described as playful activity rather than an actual game. A Facebook application like "Are you a Pirate?" is somewhere in between. In this application the user can earn credit by recruiting others to be pirates, spend the money on ships and weapons and explore the high seas, while role-playing piratey behaviour with their friends. There's no real outcome as a game but there are certainly many game elements. When the formal definitions don't give a clear answer to the 'game-or-not' question, the best alternative is to gauge public perception. In an informal poll, conducted by the author in November 2007, two hundred undergraduate game students were shown the Facebook pirate application and asked if this was a computer game - 57% thought it was.

One pattern that is readily identifiable in Facebook games and game-like applications (as in the WWW at large) is the ubiquity of Flash as the development tool for complex and highly interactive client-side interfaces. In applications like the highly controversial Scrabulous - controversial because it is basically an unlicensed online version of Scrabble (Ramesh 2008) - the entire board game is functionally modeled and runs in a Flash client. Behind the sophisticated interface there is an equally sophisticated distributed system. The management of game challenges calls on the Facebook users and friends lists; the scrabble board's state is managed during the many-sessions of a game by the application's own data writing and retrieving processes; and the validation of words and spelling is handled by third-party server processes.

It is my intention in this paper to discuss, in more detail, the Facebook platform architecture, and to compare different levels of distributed complexity. Before getting into specifics of Facebook game architectures, however, I would like to discuss the platforms themselves in a more abstract way, and in the context of a wider debate in network architecture; the REST vs WebServices debate.

RESTFUL PLATFORMS

Both the Facebook and the OpenSocial platforms state that they conform to the principles of Representational State Tranfer architecture or REST (Fielding, 2000). Described by some as a philosophy or a set of principles more than an architecture, REST has come to embody the radical alternative to over-complicated enterprise architectures by advocating lightweight, simple and elegant solutions. It's advocates argue for re-utilizing and, where necessary incrementally enhancing existing protocols and methods (eg HTTP, HTML, XML) rather than creating completely new architectures such as the Microsoft-IBM-BEA strategy which has resulted in WebServices architecture such as SOAP, OWL, WSDL etc, and implemented in .NET and Java Enterprise Edition environments. WebServices attempts to specify architecture semantics in enterprise level detail. Figure 1 (reproduced from Booth 2004) shows the semantics of the WebServices *resource-oriented* model. As well as *resources*, the WebServices specification contains complex semantic modeling of such constructs as services, policies, messaging, security, reliability, management, meta-data, business processes, transaction and presentation.

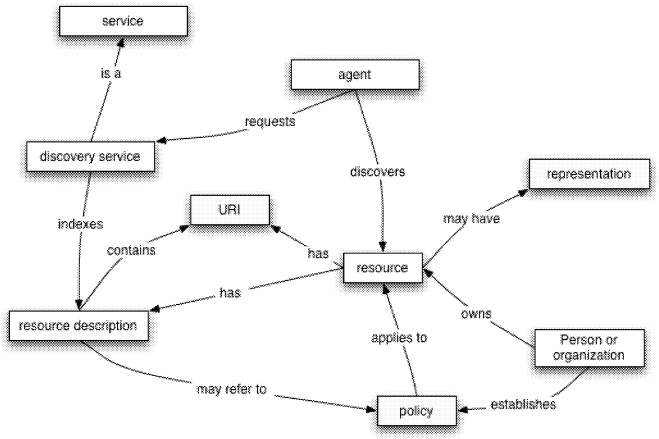


Figure 1. The semantics of the WebServices resource oriented model.

The *simplicity* of REST is in strong contrast to the "bloated, opaque, and insanely complex" specifications of WebServices architecture (Sutherland, 2004).

In a remarkable feat of synthesis Bone sums up the REST ethos in one sentence.

"It turns out that the best argument for REST is this: there are no applications you can think of which cannot be made

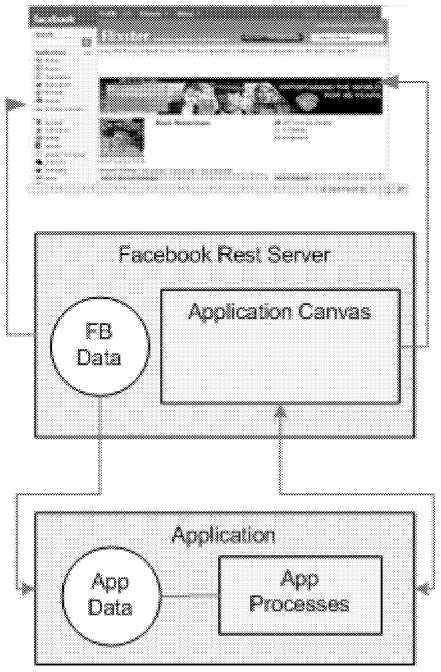
to fit into the GET/ PUT / POST / resources / representations model of the world!... with a little design thought up-front, it is complete enough and flexible enough to subsume all the application semantics you might want" (Bone, 2001)

The REST approach has been advocated by some significant players - Amazon, eBay, Google Maps, and of course the social networks have all subscribed to REST and use systems based on it's principles to execute 10s of millions of reliable transactions every day.

In Facebook, the REST server maintains the browser client-side representation of the Facebook environment while opening up a canvas – a space on the page for an application to represent itself. Severside the application API opens up it's data tables to the application.

EXAMPLE 1 THE BASIC APPLICATION

In it's most basic configuration a Facebook application runs on its application server and sends its content to the Facebook REST server where it is embedded into a *canvas* page.



Figures 1: Basic Facebook Application - Flixter

A large number of Facebook applications, some with quite sophisticated functionality, use this basic configuration. Flixter, a movie-loversapp, includes such functions as:

1. user-related data - films seen, rated and reviewed
2. friend-related data such as comparisons of your ratings with your friends
3. generic user-generated application data such as trivia quizzes by various movie-related themes, titles and topics.
4. information on movies for hire, sale, and screening near to the user
5. a reccomender system based on users who show similar tastes.

The data model for Flixter is relatively complex but the server architecture that implements the application essentially conforms to the most basic model shown in figure 1.

The application, when called, first checks if it is running inside a Facebook canvas page - if it is not the user is first directed to a Facebook login page. The Facebook REST server creates the user's facebook environment with a *canvas* page into which the application can open. The application uses a registered Facebook application key to authenticate itself to the REST server, and then uses FBQL calls to the REST server to retrieve the user's facebook data. Other application processes take the user's facebook id and retrieve the user's application specific data (eg favourite movie lists). Further processes assemble and style the visual representation of the application in FBML and pass this to the Facebook Rest Server where it is inserted into the canvas page. In Facebook terminology an application has a static Facebook address (the canvas url) and a static application address (the call-back url).

The supported application development languages for Facebook are PHP and Java, though the unofficial list also includes Facebook API clients for Python, Pearl, .NET and even ActionScript (GoogleCode 2008) .

EXAMPLE 2 FLASH INTERFACES

The second example demonstrates the architecture of a typical Flash-based, Facebook app. The ChessPro app, illustrated in figure 2. Like the previously described Scrabulous, HessPro uses a Flash client-side interface to model the game space and functionality, in this case the chess board and pieces. The app starts as described in the previous example and the application processes assemble all the required application-specific data for the user; their game stats, current games, lapsed games. For each game there will be data for the position of each piece, who's move it is, who the opponent is, and any current chat messages that accompanied the last move. The application processes assemble the page data into FBML and send it to the REST server. When the user selects a game this FBML assembly will include an embedded SWF (compiled Flash file) representing the visual, functional chess board and the chat windows.

The Flash client communicates with the application server to get the relevant game data to initialize it. The interface functionality such as clicking a piece to move (checking the move is legal, that the user is not in check) is handled by the Flash client. When a move has been made and/or a comment sent, the Flash client communicates this back to the application processes so that the game state can be preserved for future sessions.

Facebook offers two ways to embed an SWF file into a canvas page. One is with the <fb:swf> tag, and the other is with the <fb:iframe> tag. There are some implications as a consequence of this choice. Using the <fb:swf> tags automatically imports a whole set of useful data such as the user's Facebook id, and these become ActionScript variables (eg fb_sig_id). The swf file embedded in an iFrame does not

enjoy this luxury. However an swf file embedded in the <fb:swf> tag, presumably because it is more closely bound with the Facebook server processes, has some extra security restrictions imposed on it. It cannot, for example, communicate with scripts on it's host page, normally handled by the Flash ExternalInterface Object. If, for example, an application needs to redirect the user to another page via the Flash interface then the swf will need to be in an iFrame. Interestingly Facebook seems to have left a loophole in this as a page may contain two swf files, one in an <fb:swf> tag and one in an <fb:iframe> tag. The two flash files can communicate to each other through the LocalConnection object meaning the main <fb:swf> flash interface can enjoy all the benefits of being tightly bound to the data, and the <fb:iframe> flash file can be a one by one pixel square stub for the sole purpose of passing messages to the ExternalInterface object. The Hummit application described in the next section uses this work-around.

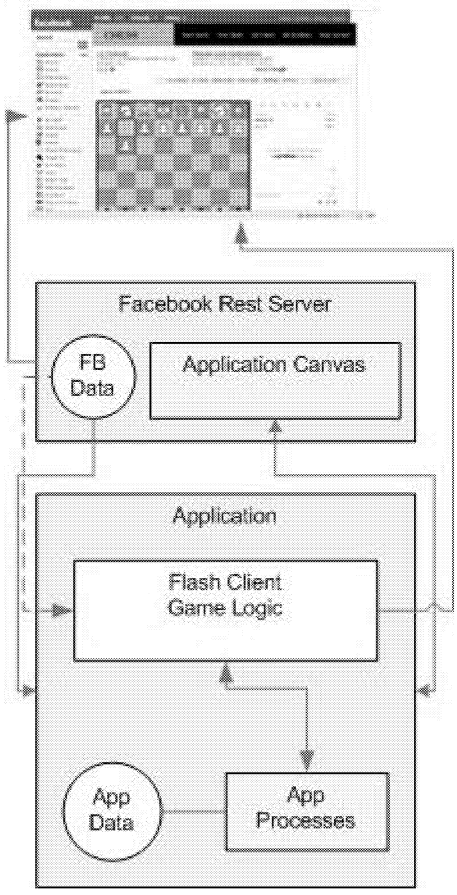


Figure 2: Flash Facebook Application - ChessPro

EXAMPLE 3 VIDEO STREAMING APPS

For the third example in this discussion I have chosen the Facebook *Hummit* application, developed by the author and illustrated in figure 3. While the data model for Hummit is very simple, the server processes demonstrates another degree of distributed complexity in terms of the application architecture. The functionality of Hummit is simple to describe. A user records themselves humming a tune using their own webcam/microphone. The recorded *humm* is uploaded to the streaming server and entered into the application data with a video file name and location, and some data about the tune entered by the hummer; the name of the tune, it's musical genre, and a cryptic clue. The hummer can then challenge one of their friends to guess the *humm*.

The friend will see the challenge in their newsfeed and notifications as, eg "Pam has hummed at you. Try and guess what she is humming". They start the application and choose the *humm* from their current list of unattempted challenges. They listen to the *humm* and get six attempts to correctly guess the title. After two incorrect guesses they get the number of words in the title. After three guesses they get the genre and after four guesses they get the cryptic clue.If they do not correctly guess the title in six attempts they fail. After they have either correctly guessed it or failed, they are asked to rate the hummer's abilities as a hummer (tone-deaf to rock star), and the difficulty of the humm itself (from easy-peasy to totally obscure).

In order to control the video record and playback the stream was recorded via the Flash client with access to the users webcam and microphone, to an Adobe Media Server using the realtime media protocol (RTMP). In terms of playback this is how a Youtube video is played back inside a Facebook application like the FunWall.

In this case the Flash client needed direct communication with the Media Server to;

1. begin a unique session connection
2. create a stream object to receive the users video
3. write the stream to a temporary server file
4. handle numerous attempts at the same humm
5. copy a submitted humm to an appropriate folder
6. clean up unwanted humm attempt files
7. playback humm videos as requested
8. do basic housekeeping on session, stream and connection objects.

As well as working closely with the Media Server the Flash Client displayes the lists of challenges sent, and received and provides the interface for recording a *humm* and for issuing a challenge (NB it was in the process of obtaining a friend selector widget in order to challenge them, that the previously mentioned <fb:iframe> stub workaround to redirect the page was used).

Last but not least the Flash client needed to be closely bound to the Hummit application processes such as:

1. Listing humms a user has recorded
2. listing challenges they have issued
3. listing challenges they have received
4. submitting a new humm they have recorded (with title, genre and cryptic clue).
5. Checking whether they have already guessed a humm
6. checking whether they have rated a humm/hummer from a challenge
7. submitting guess scores and ratings

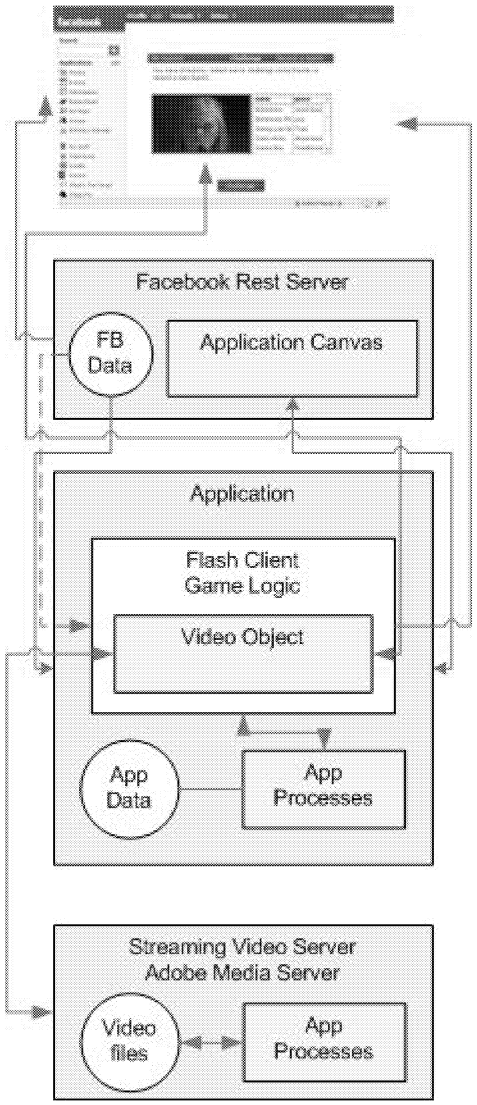


Figure 3: Flash Streaming Video Facebook Application - Hummit

As well as responding to the various data queries from the Flash client, other application processes did such tasks as aggregating hummer and humm statistics, to produce best and worst hummers and humms.

Hummit is a highly distributed application using three server environments (PHP, MySQL, MediaServer) and two client environments (HTML, Flash ActionScript) as well as the three environments used by Facebook to create the canvas page (PHP, MySQL, Apache) – eight component processes to get the application functioning in a browser.

SUMMARY

A lot of games and playful activity in Social Networking environments happens within the context of the Network's application platform. The discussion has briefly and in lay terms reviewed the recent emergence of the application platforms into two camps: Facebook vs OpenSocial; and I have described some of the shared charachteristics of both platforms and discussed them briefly in the context of the REST vs WebServices debate. The architecture of Social Networking applications was further elaborated upon with a discussion of three examples of playful applications which demonstrate a range of application architectures from basic to more complex and distributed.

One of the characteristics of the early emergence of a new technology is the tendency to get an expert group together and specify it to death. It is not inconceivable that the future of Social Network platforms will implement some of the semantic sophistication of the WebServices specifications, hopefully without losing it's present simple elegance.

REFERENCES

Bone J, 2001. REST, RPC, mountains, molehills, and a retraction. http://www.xent.com/pipermail/fork/2001-August/002801.html

Booth D et al, 2004, Web Services Architecture, W3C Publication, http://www.w3.org/TR/ws-arch/

Dodge, 2007. My Friends List is not like an OPML of RSS feeds. Url http://dondodge.typepad.com/the_next_big_thing/2007/11/my-friends-list.html.

Fielding 2000. "Architectural Styles and the Design of Network-based Software Architectures". http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

GoogleCode 2008. "Actionscript API for the Facebook platform", working group site with documentation, http://code.google.com/p/facebook-actionscript-api/

Huizinga J 1938. Homo Ludens: A Study of the Play-Element in Culture. The Beacon Press, Boston, MA, 1955.

Ramesh R 2008. "Why brothers' Facebook homage to Scrabble spells L-A-W-S-U-I-T", The Guardian, Jan 17, 2008, http://www.guardian.co.uk/technology/2008/jan/17/internet.facebook.scrabulous

Rapleaf 2007. "Statistics on Google's OpenSocial platform end users and Facebook users", http://blog.rapleaf.com/2007/11/13/statistics-on-googles-opensocial-platform-end-users-and-facebook-users/; viewed 10/09/08.

Schofield J 2007. Google's OpenSocial looks more like hype than substance. Article, Guardian, url, http://www.guardian.co.uk/technology/blog/2007/nov/04/googlesopensociallooksmore

Sutherland J 2008. "Web Services: Bloated, Opaque and Insanely Complex?", http://jeffsutherland.com/2004/10/web-services-bloated-opaque-and.html

TechDigest (2007). "The 101 Coolest Facebook Application in the World Today", http://www.techdigest.tv/2007/07/the_101_best_fa.html. viewed 10/09/08.

Tech Crunch 2007. Details Revealed: Google OpenSocial To Launch Thursday

Google 2007. MySpace and Google Join Forces to Launch Open Platform for Social Application Development, Press Release, http://www.google.com/intl/en/press/pressrel/myspace_opensocial.html, viewed 10/09/08.

BIOGRAPHY

KEN NEWMAN completed his PhD in this are in 2007 while working for the University of Portsmouth and is now a Senior Lecturer and Researcher at the University of Applied Science of Breda (NHTV) in the Netherlands.

AUTHOR LISTING

AUTHOR LISTING

Alliana A. **56**
Aranda G. **141**

Bakkes S. **69**
Barán B. **56**
Barella A. **94**
Barona I. **17**
Beach M. **27**
Bhikharie W. **132/137**
Boogaard J. **85**
Botti V. **94/141**
Buche C. **39**

Carrascosa C. **94/117/141**
Ceballos R.C. **120**
Cotta C. **120**
Cubel J.Á.C. **5**
Cunningham A. **61**

de Boer V. **107**
Deeb C.M. **8**

Eliëns A. **132/137**

Fabregat-Pinilla J. **117**
Fernández A.J. **120**
Furió D. **147**

Giménez M. **147**

Huurdeman H.C. **132/137**

Juan M.C. **17/147**

Keaveney D. **51**

Li X. **8/27**
López F. **17**
Luitjens C. **27**

Manier S. **77**
Mollá R. **17/147**

Newman K. **152**

O'Riordan C. **44/51/61**
Okada Y. **22**

Pironti M. **99**

Remondino M. **99**
Reyno E.M. **5**
Rothkrantz L. **85/107**

Samaniego A. **56**
Sanchis J.A. **129**
Saunders D. **44**
Schiesari R. **99**
Sigaud O. **77**
Spronck P. **69**

Tence F. **39**

van de Watering M.R. **132/137**
van den Herik J. **69**
Vicent M.J. **147**
Vivó R. **147**