8TH INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2007

EDITED BY

Marco Roccetti

NOVEMBER 20-22, 2007

UNIVERSITY OF BOLOGNA
BOLOGNA
ITALY

A Publication of EUROSIS-ETI

Cover art was reproduced by kind permission of Koala Games, Bologna, Italy

8TH International Conference on Intelligent Games and Simulation

BOLOGNA, ITALY

NOVEMBER 20 - 22, 2007

Organised by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Binary Illusions

University of Bradford

Delft University of Technology

Ghent University

Koala Games

Larian Studios

The Moves Institute

Simulation First

Hosted by

The University of Bologna

Bologna, Italy

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

General Conference Chair

Marco Roccetti
University of Bologna
Bologna, Italy

Local Programme Committee

Stefano Cacciaguerra, University of Bologna, Bologna, Italy
Antonio Corradi, University of Bologna, Bologna, Italy
Gabriele D'Angelo, University of Bologna, Bologna, Italy
Stefano Ferretti, University of Bologna, Bologna, Italy
Marco Furini, University of Piemonte Orientale, Italy
Luca Genovali, Institute for Advanced Studies of Lucca, Lucca, Italy
Vittorio Ghini, University of Bologna, Bologna, Italy
Silvia Mirri, University of Bologna, Bologna, Italy
Claudio Palazzi, UCLA, USA
Laura Ricci, University of Pisa,Pisa, Italy
Paola Salomoni, University of Bologna, Bologna, Italy
Cesare Stefanelli, University of Ferrara, Ferrara, Italy

INTERNATIONAL PROGRAMME COMMITTEE

Game Development Methodology

Track Chair: Licinio Roque, University of Coimbra, Coimbra, Portugal
Joaquim Ramos de Carvalho, University of Coimbra, Portugal
Barbara M. Grüter, Hochschule Bremen University of Applied Sciences, Germany
Óscar Mealha, University of Aveiro, Portugal
Eija Karsten, University of Turku, Finland
Jari Multisilta, University of Tampere, Finland
Esteban Clua, Universidade Federal Fluminense, Brasil

Physics and Simulation

Graphics Simulation and Techniques
Arjan Egges, Universiteit Utrecht, Utrecht, The Netherlands
Magy Self El-Nasr, Penn State University, University Park, USA
Pieter Jorissen, Universiteit Hasselt, Diepenbeek, Belgium
Ian Marshall, Coventry University, Coventry, United Kingdom
Marco Roccetti, University of Bologna, Bologna, Italy

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

INTERNATIONAL PROGRAMME COMMITTEE

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

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
Adriano Joaquim de Oliveira Cruz, Univ. Federal de Rio de Janeiro, Rio de Janeiro, Brazil
Chris Darken, The MOVES Institute, Naval Postgraduate School, Monterey, USA
Andrzej Dzielinski, Warsaw University of Technology, Warsaw, Poland
Maja Pivec, FH JOANNEUM, University of Applied Sciences, Graz, Austria
Martina 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.
Scott Neal Reilly, Charles River Analytics, Cambridge, USA
Marco Remondino, University of Turin, Turin, Italy

Collaboration & Multi-agent Systems
Victor Bassilious, University of Abertay, Dundee, United Kingdom
Sophie Chabridon, Groupe des Ecoles de Telecommunications, Paris, France
Nicholas Graham, Queen's University, Kingston, Canada

Opponent Modelling
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

INTERNATIONAL PROGRAMME COMMITTEE

Storytelling and Natural Language Processing

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

Modelling of Virtual Worlds

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

Online Gaming and Security Issues in Online Gaming

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

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
Richard Ferdig, University of Florida, Gainesville, USA
Tony Manninen, University of Oulu, Oulu, Finland
Jaap van den Herik, University of Maastricht, Maastricht, The Netherlands

Games for training

Ahmed BinSubaih, University of Sheffield, Sheffield, United Kingdom
Michael J. Katchabaw, The University of Western Ontario, London, Canada
Jens Müller-Iden, Universität Münster, Münster, Germany
Roger Smith, US Army, Orlando, USA

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

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

Games Interfaces - Playing outside the Box

Games Console Design

Chris Joslin, Carleton University, Ottawa, Canada

Mobile Gaming

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

INTERNATIONAL PROGRAMME COMMITTEE

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

© 2007 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, Ghent University, Faculty of Engineering, Dept. of Industrial Management, Technologiepark 903, Campus Ardoyen, B-9052 Ghent-Zwijnaarde, Belgium.

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

Printed in Belgium by Reproduct NV, Ghent, Belgium
Cover Design by Grafisch Bedrijf Lammaing, Ostend, Belgium
Cover Pictures by Koala Games, Bologna, Italy

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

EUROSIS-ETI Publication

ISBN: 9789077381373
EAN: 9789077381373

GAME ON®
2007

Preface

Welcome to Game-On 2007, the Annual European Conference on Simulation and AI in Games. On behalf of all the people who made this conference happen, I wish to welcome you to this special event.

During the past years, Game-On offered an opportunity for researchers and practitioners to present their findings and research results in the new and exciting field of gaming and concurrent technologies. This year, the 3-day technical program provides a forum to address, explore and exchange information on the state-of-the-art of AI, simulation, networking, and all the other allied technologies, in support of gaming, their design and use, and their impact on our society.

With the emerging importance of gaming, we need to tackle a broad range of technology, management and design issues, and we need to become familiar with newly introduced techniques and current applications. To this aim, the gamut of papers presented at Game-On will cover topics from game methodology and game AI, to art, design and game graphics; from mobile and online gaming, to games for education and serious gaming, plus others designed to provide a wide range of topics as reflected in the technical program of the Conference.

All those contributed papers have undergone a serious paper review and helped us to achieve this goal. Special recognition goes to each of the contributing authors for their dedication and effort in their field of research. In addition to all the accepted papers, we assembled a program comprising a keynote speech (given by dr Graham Morgan, University of Newcastle upon Tyne), a round-table discussion on serious games and a games projects session

On behalf of the Organizing Committee, I would like to extend my personal thanks to all the members of the International Program Committee for their hard work in reviewing and selecting the best papers to be presented from all the received submissions. The success of this conference is credited to them, as well as to session chairs, presenters and attendees. My sincere thanks also go to Philippe Geril, our *deus-ex-machina*, who has helped us in putting together such an excellent program, as well as for his organizational efforts and input with the Conference.

Finally, I must admit it is my personal pleasure to host, this year, this Conference in the University of Bologna, the most ancient in the Western World. I am confident you will enjoy the Conference and as well as the many treasures of my city!

Marco Roccetti

Game-On 2007, General Chair

CONTENTS

CONTENTS

SCIENTIFIC PROGRAMME

GAME METHODOLOGY

WEBBING: A SMART ARCHITECTURE FOR SNAPPY BROWSER–BASED GAMES

Alessandro Amoroso
Department of Computer Science
Università di Bologna
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: amoroso@cs.unibo.it

KEYWORDS

Smartphone, Wireless, Web applications.

ABSTRACT

This paper presents the main architectural issues of a new platform to implement multiuser interactive games based on the browser. Those games runs "inside"the browser, equipped with standard plug-in, and without installing any program on the client side.
We named this new architecture WEBBING, that stands for WEB Based INterpersonal Games. This architecture is responsive, reliable and scalable. The main characteristics of WEBBING suggest that it could be profitably used to implement browser–based games that could run on a smartphone.
We also sketch the main design issue of a test bed game that we are planning to experimentally study our architecture. The game is aimed to the new iPhone, that presents several programming challenges.

INTRODUCTION

Online games are very popular these days, and their popularity is continuously growing. The increasing diffusion of small portable devices, such as the smartphones, has been one of the factor that stimulated the generation of a branch of online games called *browser–based games*. Those games do not require any software to be installed at the client side, they rely solely on the browser with some common plug-in. The most common of these plug-ins are Java, Flash, and Shockwave. In general, the main advantage of browser–based applications is that they do not require to install software that will run on the client side. Recently, browser–based games use Web technologies, like AJAX, to allow for multiuser interaction.
In this paper we present a novel architecture to implement multiuser browser–based games. We named this new architecture WEBBING, that stands for WEB Based INterpersonal Games. Moreover, we show the main architectural issues of a browser–based game, aimed to the iPhone, that we are going to implement as a test bed for our architecture. We discuss several properties of WEBBING, the main of them are: *responsiveness*, *fairness*, *reliability*, and *scalability*.
The bandwidth available to the Internet users is increasing these days, but the latency of the messages over the network is almost constant. The *best effort* nature of the Internet leads to variable, and unpredictable, messages latency. The responsiveness of an online game strictly depends on the latency of the messages. There are several papers that deal with the responsiveness of the online games. Among the others, in (Claypool and Claypool, 2006) the authors studied how the latency influences the online games, while in (Chen et al., 2006) there is a study on the effects of the network QoS with respect to the quality of the game. Some proposed novel a technique to a smooth and fast delivery of multimedia contents in a wireless home entertainment center, such as(Palazzi et al., 2006). In (Brun et al., 2006b) the authors discuss the problem of server selection, proposing the idea of *critical response time*, they compare the proposed solution with respect to a centralized one.
The consistency of the state of a distributed multiplayer game has been deeply investigated. Among the others, (Li et al., 2004) presents a continuous consistency control mechanism, based on a relaxed consistency control model for continuous events. The fairness of real-time online games is studied in (Brun et al., 2006a).
The available smartphones suffer several hardware limitations with respect to the current personal computers. The main limitations are on the screen size and on the processor performance. Additionally, these devices are particularly sensible to energy consumption, while they are battery equipped. The screens are few hundreds of pixels per side, and are large at most 3.5" in diagonal. The above limitations depict a kind of device that resembles the ones available in early 90's of the last century. Moreover, several smartphones are not equipped with a full keyboard; when present, the keys are quite small. A common feature to almost all the smartphones is a simply accessible pointer control.
The design of a browser–based game, that takes in account the above limitations, strictly resembles the old

arcade games. Usually, those games used a joystick, and few buttons, as means of moves input. This characteristic suits very well the current smartphones.
In addition to the above mentioned features, a modern arcade style game, running on a device connected to the Internet, could offer multi–user interactions. Since almost all of the smartphones are capable to connect to the Internet, the design of a game could take advantage of that. The resulting game could have an interactivity between the users that ranges from soft real-time to loose asynchrony.

ARCHITECTURE

The client device of our architecture presents several challenges, mainly due to software constraints, that strongly suggest a *thin client* design. The game, and client management, run almost entirely on the servers, while the clients provides for graphical user interface.
To implement WEBBING, we are studying a refinement of an architecture, called AIDA (Amoroso, 2007), that supports snappy auctions over the Internet.
The WEBBING architecture inherits from AIDA both responsiveness and fault tolerance. The importance of the scalability depends on the nature of the implemented game. When the implemented game deals with tight interaction between a small set of users, the scalability is not a strong issue. This scenario imposes a constraint to the number of users due to the screen size.
WEBBING is an hierarchical, timed, distributed architecture, arranged in two logical levels of abstraction. The servers at the lower level are called *cohort* servers, and one of them has the role of *leader*, that is the unique server at the higher level. The Fig.1 shows an example of this architecture. The nodes labelled S_1, S_2, S_3 represent the servers implementing the WEBBING architecture, and the unlabeled squared nodes represent the clients connected to them. The solid edges in the graph represent communication channels, used to manage the user interface, while the dashed edges, fully connecting the set of servers, are communication channels carrying coordination messages. The clients have no visibility of each single server, but they perceive the gaming system as a coherent whole. Needless to say, the set of servers shown in the figure is largely oversized with respect to the number of clients. Due to both synchronization and fault tolerance issues, the minimum number of servers is three.

Cohort servers these servers are at the lower level of the WEBBING hierarchy; they accept connections from the clients. In Fig.1 the servers S_1 and S_2 are cohort servers. These servers receive clients moves as them are submitted. Periodically, each cohort server computes the *set of local moves* based on the moves it has received in the last period, if any, and communicate

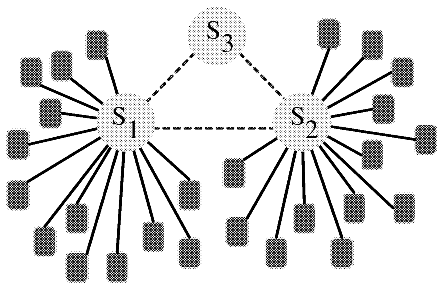


Figure 1: System architecture

that to the *leader*. The cohort servers relay the *state of the game* to the clients, upon receiving the specific messages from the *leader*. Moreover, the cohort servers detect and manage possible leader faults.

Leader this is a unique server, at the upper level of the hierarchy, directly connected to each *cohort* server. In Fig.1 the leader is S_3. The leader receives the set of local moves from the *cohorts* and, periodically, computes the *state of the game* based on the moves it has received in the last period, if any, and broadcasts that to the cohort servers. The leader resolves the possible conflicts between the sets of moves that it has received from the *cohorts*. Possible failures of the leader are managed by the reconfiguration mechanism described later.
The content of the message named *state of the game* depends on the nature of the game. It could be either a mere sequence of the non conflicting last moves, or a full snapshot of the game. When the state of the game contains the last moves, the length of the resulting message could be minimal, but it is mandatory that none of the messages got lost. When the state of the game contains the snapshot of the game, the resulting message could be bigger than the previous, but contains all the information to keep the user up–to–date with respect to the game state.

Client Side The users join the game by means of a browser request to the Web service of the game. The Web service answers with a client program, that connects to a server and manage the user participation to the game. The client program launches sequential requests of availability to a predefined set of cohort servers, and waits for their answers. After the last answer, the client program selects the server that answered in the shortest time, ignoring the other servers. The client then receives the current state of the game from the chosen server.
The client program can submit a move to the chosen server any time it likes, and, almost periodically, it receives the game state message from that server.
The above strategy provides for the selection of the

server that is nearly the most responsive one. Moreover, this measured responsiveness take into account both the server load and the network state, even if these were not detected exactly at the same time. Each client program sticks with the chosen server, unless it detects unsustainable performance, *i.e.* it cannot keep the *user response time* below a certain threshold. In that case, the client program starts a new selection procedure. Note that the above technique allows for a run–time *load balancing* with respect to the servers and the network; requiring a negligible effort for the servers, a small amount of work for any client, and a little overhead of the network traffic. Moreover, the presented technique is effective in case of both server and network faults, enabling the client to autonomously connect to another server.

MAIN PROPERTIES

Timelines The discussion of the main timelines properties of WEBBING by means of the example shown in Fig.2, that is a *space–time* diagram for a simple interaction in the system presented in Fig.1. To simplify the diagram, the figure shows few clients and all the servers. The solid horizontal lines represent the computations of both the servers and the clients; the time flows from left to right. Client C_1 is connected to server S_1, while C_2 and C_3 are connected to S_2. The small vertical lines on the server's computations represent time ticks, *i.e.* when the server computation reaches a tick, it triggers an action accordingly to the server role. The distance between two consecutive ticks is the duration of the *server period*. The arrowed lines represent messages: the beginning of a line represents the sender time, while the pointed end represents the receiver time. Note that those times are measured with clocks that are local to each process, and therefore can solely be assessed by the other servers. At this moment, we may ignore the points labelled with Greeks letters, these will be discussed later.
The Fig.2 shows client C_1 submitting a move to its server S_1, that receives this move just after a tick. Therefore, S_1 will store the received move until either it receives a conflicting move, or its current period expires, *i.e.* its computation reaches the next tick. In case of conflicting moves, it depends on the game criteria how to resolve the conflict. In the meanwhile, the clients C_2 and C_3 submit a move to their server. The server S_2 concatenates these two moves, stores them, and waits until the next tick to notify the leader server. Note that the moves of the three clients are independent of each other, because they are submitted in parallel.
The cohort servers send the messages containing the set of local moves to the leader at a certain pace. There is no synchronization between their sending periods. The duration of the sending periods is a trade–off between the number – and the size – of the messages, and the responsiveness of the gaming system. When a server de-

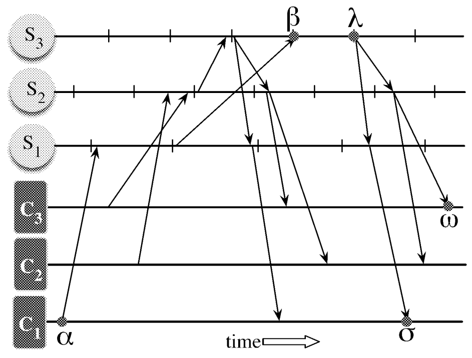


Figure 2: Space–time diagram of some users interactions

tects a deterioration of the network performance it can increase the duration of its period; this leads to both less sent messages to the leader, and to less system reactivity. On the other hand, when a server detects an increasing of the network performance it can reduce the duration of its period; this leads to both higher number of messages and better system responsiveness. The duration of the server's period could be dynamically adapted to take into account both the number of messages and its responsiveness. The servers immediately relay the state messages because there is no convenience into delaying them, and doing that will reduce the system responsiveness.
As shown in Fig.2, the first cohort server that reaches a tick is S_1. The notification from S_1 to the leader S_3 takes a while and will arrive just after a tick, then S_3 will store it until the next tick. In the meanwhile, S_2 reaches its tick, and its notification to S_1 quickly arrives, just before a tick.
When the leader reaches its next tick, it multicasts to the cohort servers the actual state of the game, *i.e.* the one updated by the moves previously received from S_2. When a cohort receives the multicast from the leader, it immediately relays the message to its clients. In the given example, all the clients receive the current state fo the game containing the merged moves of C_2 and C_3. After the above described multicast, the leader S_3 receives the moves from cohort server S_1. Since this new move is still compatible with respect to the actual state of the game, the leader stores it until the next tick. As in the previous case, the new state of the game, containing the move from C_1 finally reaches all the clients after few consecutive multicasts.
The leader decides the state of the game, ad it is the point of synchronization of the system. The ordering of events is forced by the leader receiving time, that manage the consistency of the moves based on the ordering. In order to reduce both the load of the leader and the network traffic, the cohort servers locally manage inconsistent messages, by means of the same criteria of the leader. It could happen that same moves, that are incon-

sistent at local level, become consistent at global level, due to some intermediate moves. This scenario depends on the nature of the game, and could be circumvented by inhibiting the check for consistency operated by the cohorts.

Responsiveness We can express the *responsiveness* R as the time spent by the system to merge a new move with respect to the state of the game, and to communicate the new state to all the clients. In the example shown in Fig. 2, the responsiveness with respect to the move of C_1 is the elapsed time between the points α and ω. Let δ_{max} and δ_{min} be the maximum and the minimum latency for any message in the system. Those values are a parameter of the design; their value could be either assessed by measurement, or defined by a reasonable default. In addition, we name the periods of the servers, *i.e.* the elapsed period between two consecutive ticks, as τ_c, and τ_l for the cohorts, and the leader respectively. Summarizing: $R \leq \tau_c + \tau_l + 4\delta_{max}$
We can divide the responsiveness in three sequential step. The first step is the time that the leader takes to receive a move, named *rise time*, that is at most $\tau_c + 2\delta_{max}$. In Fig. 2 the rise time for the move of C_1 is the elapsed time between points α and β. The second step is the leader latency, called *coordinator time*, that is at most τ_l. In Fig. 2 this is the elapsed time between points β and λ. The third step is the "diffusion" of the current state of the game, called *spread time*, that is the elapsed time between the broadcast of a new state, from the leader, and the last reception of that state by the clients. In Fig. 2 this is the elapsed time between points λ and ω.
Note that this assessment of R does not explicitly consider the load condition of the servers, *i.e.* how long a message waits in the queue of the received messages before being evaluated. Such a delay might occur because a server is dealing with messages previously received.

Fairness A side effect of the responsiveness is the *fairness*, F, of the system, *i.e.* the elapsed time between the first and last reception of the same state message by the clients. In other words, F represents the advantage of the first client that receives a state message with respect to the last client that receives the same message. Fig. 2 shows an example of fairness as the elapsed time between points σ and ω.
To assess the fairness we consider the maximum difference between the fastest and the slowest spread times for the same state message; therefore: $F = 2(\delta_{max} - \delta_{min}) \leq 2\delta_{max}$

Fault Tolerance The WEBBING systems remains available, and properly working, in spite of a predefined number of servers and communications faults. The maximum number of tolerable servers fault is half minus one of the servers. Our system can tolerate the following types of faults:
Channel Performance: a communication channel is overloaded, the connection remains active, the messages are delivered with higher latency than expected;
Channel Disruption: a communication channel does not delivers messages anymore, it has to be re-established;
Processor Performance: a computer is overloaded and responds later than expected;
Processor Crash: a computer becomes not working, then all the programs and communications executed by that computer become unavailable. A crashed processor could, possibly, return available after a fresh restart.
Each client regularly receives the state of the game message. In regular running that message is sent almost periodically, its arrival time gives to each client a gauge of both the server load and the network traffic. The missing of a state of the game message signals to the client the presence of a fault, either of the server or of the network. In that case the client could connect to another server, by re-running the procedure to choose a server that has been described above. This operation, named *client migration*, is completely transparent to the user. The migration does not require any effort to the servers, therefore, the crashed ones neither blocks, or interferes with respect to the migration. In other words, the crashed servers do not influence the reconfiguration of the system. Moreover, the migration requires a lightweight work solely to the clients that are migrating, without any kind of coordination with each other.
The cohort servers elect the leader The leader remains in charge for the whole duration of the game, unless it suffers a fault. In the latter case, the cohort servers elects another leader. Moreover, by means of the same mechanism, the leader can manage the joining of a new server to the group of cohorts.

Scalability As well known in literature, a distributed system is *scalable* if it remains effective despite of a significant increase in the number of both users and resources.
We can assess the amount of resources required by the system with respect to the number of clients. Specifically, we consider both the number of servers and the number of messages exchanged between the servers. The number of cohort servers is linearly proportional with respect to the number of clients. The number of parallel messages exchanged by the servers, is at most twice the number of communication channels between the servers. The workload of any cohort server is almost balanced. As mentioned above, the load of the servers mainly depends on the number of active connections, that could be easily kept balanced during the growth of the system. Due to this property, the single leader does not represent a performance bottleneck.

IMPLEMENTATION DESIGN

We are implementing a client for WEBBING that targets the Apple iPhone. Two are the main motivations for this choice. Firstly, we think that the device is the first one of a new class of smartphones, that will rapidly grow in popularity between the users. Secondly, the programming paradigm proposed by the iPhone is quite unusual, being strongly oriented to browser–based applications; therefore it represents a challenging opportunity to modify the way we traditionally write programs for smartphones. Moreover, a program that runs on the iPhone – avoiding to take advantage of the devices special features – will run without any modification on a large number of platforms, ranging from other portable devices to desktop computers.
The iPhone offers two different means to connect to the Internet: fast Wi–Fi (802.11b/g, at about 1000Kbit/sec) and slow EDGE (about 140Kbit/sec). The device is capable to silently switch between the two connection modes without interfere with the current internet operations. This capability could be used to test WEBBING with client suffering highly variable connection delays.
We are planning to implement, as test bed game, a kind of distributed "pac–man". This game offers important characteristics, against which to test our system; the most relevant of them are:
All-to-all: tight interactions between a small set of users – we suppose they will be no more than ten;
Soft Real–time by the nature of the game;
Simple User Commands: to the avatar, that responds by changing its moving direction in the bi–dimensional constrained space; this allows for small messages, carrying the moves, sent by the client.

iPhone Peculiarities As largely emphasized by Apple, the input device of iPhone are the user's fingers that directly touch the sensitive screen. The main difference with respect to a built–in device is a resulting lower precision in the pointing. The iPhone does not send to the shown Web page some events traditionally generated by a pointing device, such as the "mouse–over" events. Moreover, the iPhone screen manage *multi touch* gestures performed by means of multiple fingers on the screen at the same time. For portability reasons, in our design we do not plan to take advantage of the latter.
The touch–screen of the iPhone is $3'' \times 2''$, at the resolution of $160dpi$. The device self detects its orientation, and, automatically swap between landscape and portrait mode, accordingly with respect to its orientation. The iPhone is equipped with a "software" keyboard, that is shown on the screen upon user request. Since the keyboard fills part of the screen, it is not a valuable means of user input for the application that we are designing.
The browser available on the iPhone is Safari 3.x. The browser has some restrictions, such as the size of embedded JavaScript limited to 10MB, with respect to the equivalent version running on personal computers. At the moment, the operating system of iPhone does not allows for any access to the file system, and the installation of any software is unsupported. Therefore, the application that we are designing can run solely as a browser script. Moreover, it is not possible to add any plug-in to the browser, that nowadays does not support technologies such as Flash. The only technologies available to build a browser–based application that runs on the iPhone are: HTML 4.01, XHTML 1.0, CSS 2.1 (and some 3.x), JavaScript 1.4, DOM, AJAX/Web 2.0 (included the *XMLHTTPRequest*).

CONCLUSION

We presented the main characteristics of WEBBING, and sketched the main design issue of a test bed game that we are implementing to experimentally study our system.
WEBBING is responsive, reliable and scalable. Moreover, the fairness of the system depends, almost entirely, on the latency of the messages sent over the Internet.
We believe that the system could be well suited to implement browser–based games. At this very preliminary stage, we have developed part of the WEBBINBG architecture, and some proof–of–concept tests seems to confirm our study.

Acknowledgment This work has been partially funded by the Italian MIUR Project (MoMa).

BIBLIOGRAPHY

Amoroso A. 2007. "Aida: Responsive and available auctions over the internet." In *GLOBECOM 07*, Washington DC, USA, IEEE.

Brun J.; Safaei F. and Boustead P. 2006a. "Managing latency and fairness in networked games". *Commun. ACM*, 49(11):46–51.

Brun J.; Safaei F. and Boustead P. 2006b. "Server topology considerations in online games". In *NetGames '06: Proc. of 5th ACM SIGCOMM workshop on Network and system support for games*, New York, NY, USA. ACM Press.

Chen K.; Huang P. and Lei C. 2006. "How sensitive are online gamers to network quality?" *Commun. ACM*, 49(11).

Claypool M. and Claypool K. 2006. "Latency and player actions in online games". *Commun. ACM*, 49(11):40–45.

Li F.W.B.; Li L.W.F. and Lau R.W.H. 2004. "Supporting continuous consistency in multiplayer online games". In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 388–391, New York, NY, USA. ACM Press.

PalazziC.E.; Pau G. Roccetti M.; Ferretti S. and Gerla M. 2006. "Wireless home entertainment center: reducing last hop delays for real-time applications". In *ACE '06: Proc. of 2006 ACM SIGCHI intl. conf. on Advances in computer entertainment technology*, New York, NY, USA. ACM Press.

Towards a High-Level Design Approach for Multi-Server Online Games

Alexander Ploss, Frank Glinka, Sergei Gorlatch, and Jens Müller-Iden
University of Münster, Germany
email: {plotzer | glinkaf | gorlatch | jmueller}@math.uni-muenster.de

ABSTRACT

The development of scalable online games is a complicated problem that often requires a large amount of low-level and error-prone programming. We analyse and classify currently used development methodologies for games, and propose a novel, high-level development approach. As a possible base for a high-level game design, we describe the *RTF* (Real-Time Framework) middleware system. The RTF middleware enables an easy and flexible design of distributed, multi-server game software with minimized development efforts. We explain how RTF supports flexible implementation of single-server online games and how the RTF allows to switch to practically proven multi-server distribution concepts of zoning, instancing and replication for increased game scalability. The RTF facilitates flexible and problem-oriented adaptation and combination of these techniques in a seamless development approach.

KEYWORDS

Online games, Game middleware, Game development methodology, Real-Time Framework

INTRODUCTION

Current practice of online game development usually relies on using a company's in-house expertise for a custom game design. Such custom development employs low-level programming and networking tools, which makes it time-consuming, risky and often very expensive.

This paper aims at raising the level of abstraction in the development of online games, in order to simplify the development process and improve its productivity. We suggest a high-level development approach that shields the human developer from the underlying low-level details, e.g., allowing him to transfer objects using communication channels instead of transferring low-level byte arrays through system-dependent sockets.

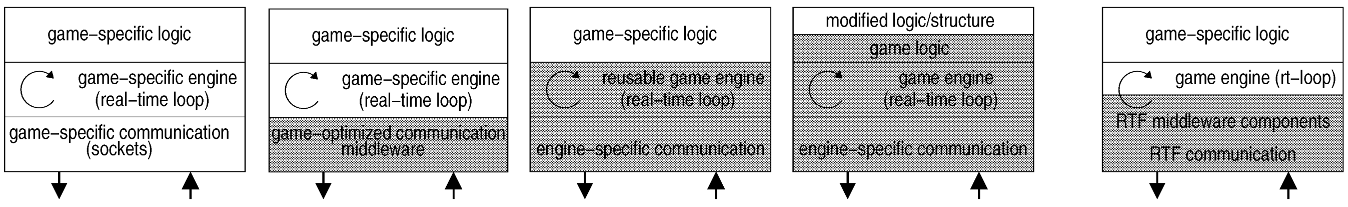
We aim both at single-server and multi-server online games, with the goal of seamless integration of these concepts: this makes our approach generally usable by all online game genres and allows for easy and flexible combination of multi-server network architecture concepts. Handling multiple servers in our approach enables high-level development of *Massively Multiplayer Online Games* (MMOGs) that pose especially difficult challenges. These include the efficient realization of the communication between game participants, as well as techniques that allow to scale up the player numbers in the application by using multiple servers. This should substantially improve the current situation of custom development, in which every new multi-server game is designed and implemented almost from scratch.

This paper presents our *RTF* (Real-Time Framework) middleware system for seamless and consistent game engine development and operation for single- and multi-server engines. The RTF middleware provides integrated solutions for a variety of development and runtime problems in multiplayer online games:

- Design optimized, object-oriented serialization and communication for game entities and events;
- Integrate fast algorithms and efficient data structures, e.g., for area of interest management;
- Efficiently distribute and consistently maintain the game state across servers and clients;
- Parallelize game state processing among distributed servers to achieve scalability;
- Provide the players with an overall seamless and responsive virtual environment.

Although there has been extensive work specifically and exclusively targeting most of these problems, relatively little research has been conducted so far on integrated high-level libraries and middleware for games. It is our goal in this paper to study which middleware concepts can be adapted and enhanced for a variety of online games types, ranging from fast-paced and small action games to large MMOGs and how this can be done.

The contributions and the structure of the paper are as follows: We develop a comprehensive taxonomy for current online game design approaches, with respect to their complexity and flexibility. We sketch a high-level game development approach and describe the basic concepts of our corresponding game middleware. We show how RTF is employed for multi-server game processing, give an overview of a first case study and conclude the paper by summarizing our contributions in the context of related work.


game–specific logic
game–specific engine (real–time loop)
game–specific communication (sockets)
game–specific logic
game–specific engine (real–time loop)
game–optimized communication middleware
game–specific logic
reusable game engine (real–time loop)
engine–specific communication
modified logic/structure
game logic
game engine (real–time loop)
engine–specific communication
game–specific logic
game engine (rt–loop)
RTF middleware components
RTF communication

(a) Custom Development (b) Comm. Middleware (c) Using Existing Engine (d) Modding Game Game (e) RTF approach

Figure 1: Main Approaches to Game Development. Unfilled: Self-Developed, Shaded: Use Existing Components

GAME DEVELOPMENT APPROACHES

The central part of a game software system consists of the *game state*, i. e., the collection of all objects that form the virtual environment, and the continuous *processing* of the game state. In this paper, we focus on the development of the game state and its processing, rather than on the game user interface, i. e., the representation of the virtual environment the player interacts with.
In order to compare different development approaches of the overall distributed architecture of online games, we identify the following three major aspects of online game software systems:

- **game logic**: *entities*, *events* (data structures), and processing rules describing the virtual environment;
- **game engine**: *real-time loop* which continuously processes (user) events, according to the rules of the game logic, to compute a new game state;
- **game distribution**: logical partitioning of the game world among multiple servers, computation distribution management according to actual game state, and communication.

The third aspect, game distribution, can be further split up into two levels of distribution: a) distribution of the user interface and game state processing between client and server, and b) distribution of game state processing in the multi-server architecture.
These three aspects are treated differently, depending on the requirements and properties of a particular game genre. For example, fast-paced action games rely on efficient communication and engine implementation while using only relatively simple game logic and -content. The complexity of the game distribution aspect usually increases with the number and density of the participating users within a game and is thus particularly challenging for MMOGs.
Our classification in Fig. 1 distinguishes common approaches a) – d) to game development, according to how they treat these three aspects. In each approach, the aspects shown in white are managed by the human developer, whereas the shaded areas are provided automatically by the development system:

(a) Custom Development: The most direct approach used for game development is to design and implement the entire software system individually. The development team designs and implements all three major aspects of the game software system: game logic, game engine and game distribution. This allows the developers to have full control over their code and optimized implementation with focus on the individual performance needs of the game. While the custom development of an entire game is very complex, hence cost-intensive and error-prone, it is sometimes the only way to achieve the particular objectives of the game design because of its flexibility.

(b) Game Communication Middleware: This approach uses special communication libraries and middleware systems (like Quazal Net-Z (Quazal, 2006)) for game development. As shown in Figure 1(b), the game developer employs the middleware to realize the communication between clients and servers in a distributed game while implementing the game engine and logic on his own. Using this approach, the developer has enough flexibility to design and implement the aspects of game logic and game engine whilst the middleware deals with the game distribution. However, available libraries usually focus on a particular architecture setup, decreasing flexibility of the engine development. Furthermore, this approach has been used only rarely for the development of multi-server based MMOGs since a pure communication library is not sufficient for these games. A middleware for MMOGs also has to deal with the difficult task of distributing the game processing among multiple servers, for which only a few middleware systems are available (e.g. Emergent Server Engine (EGT, 2007) or BigWorld (BIG, 2006)).

(c) Using Existing Engine: With this approach, shown in Figure 1(c), an existing game engine, i.e., the processing component of a game, is re-used to develop a completely new game. This reduces the complexity of development. Some game studios design their game engines primarily for the purpose of reselling and licensing the engine afterwards. Examples of popular and often used engines are the *Quake 3* engine or the *Unreal* engine. However, a particular engine is quite inflexible because it is usually closely tied to a specific game genre.

(d) Game Modding: Figure 1(d) outlines the approach of game modding (community jargon for modifying an existing game) via a dedicated interface for programming the game logic. This was first done by hobby developers who modified the actual game content. Nowadays, the creation of mods is based on high-level tools created and also used by the game development studios themselves. Such tools allow the creation of game content by designers with minimal programming effort. The primary aspect of modding is the creation of new game content within the constraints of an existing game logic; hence it is rather inflexible. Nevertheless, modding allows to develop innovative game concepts, and sometimes a mod becomes even more popular than the original game as, for example, the mod *Counter-Strike* based on Valve's game *Half-Life*.

(e) RTF Multi-Server Middleware: Our Real-Time Framework, as illustrated in Fig. 1(e), allows a novel game development approach which provides more processing support than using only a communication middleware, but does not constitute a complete game engine, allowing higher flexibility. Thus, the RTF can be classified in between the approaches (b) and (c). The characteristics and usage of the RTF, justifying this classification, are discussed in the next sections.

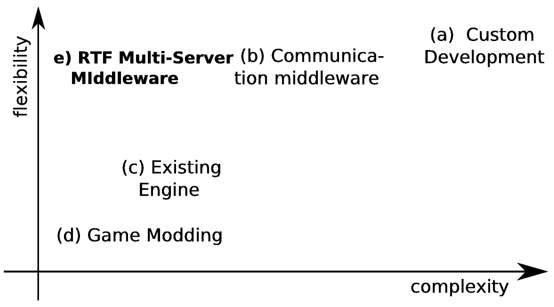


Figure 2: Taxonomy of game development approaches

Figure 2 illustrates our taxonomy of the five discussed development approaches with respect to their flexibility and complexity. The most simplicity in terms of distributed software infrastructure is offered by existing game engines (c) or modding toolkits (d). However, these approaches have the remarkable drawback of being quite inflexible. Obviously, the fully custom development (a) offers most flexibility while being rather complex. The use of special middleware (b) is a promising alternative for particular tasks: its use reduces the complexity of game development. Pure communication support is not enough for MMOGs: for such large distributed systems, the multi-server management is quite extensive and increases the development complexity. As indicated in the taxonomy, the RTF is designed to provide the developer with most flexibility in game design while freeing him from complex low-level implementation tasks in the game development process.

RTF: MULTI-SERVER MIDDLEWARE

The Real-Time Framework provides a high-level communication and computation middleware for single-server and multi-server online games. RTF supports both the server-side and client-side processing of an online game with a dedicated set of services which allows developers to implement their optimized engine at a high, entity-based level of abstraction in a flexible manner. Figure 3 shows a generic multi-server example of a game developed on top of RTF.

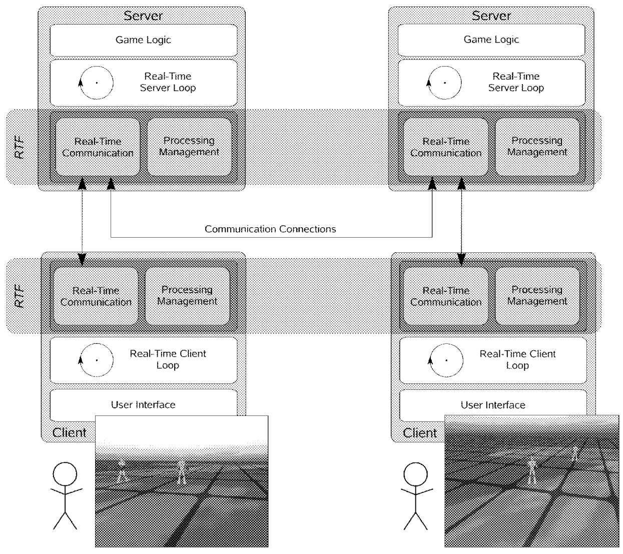


Figure 3: RTF multi-server Middleware

The RTF middleware deals with entity and event handling in the real-time client loop and the continuous game state processing in the real-time server loop, and the distribution of the game state processing across multiple real-time server loops. The developer implements the game-specific real-time loop on client and server, as well as the game logic, using the RTF middleware to exchange information between the processes.

GENERAL DEVELOPMENT TASKS

The development of the game state processing in online games consists of several tasks, as shown in Figure 4. Regardless of developing a multi-server MMOG

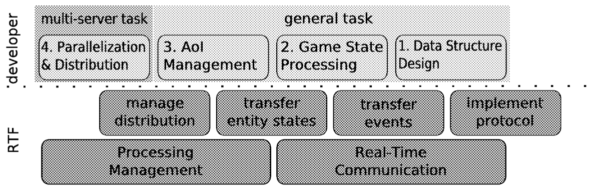


Figure 4: Development tasks for a multi-server game: distribution between RTF and developer

or a single-server, small-scale action game, the developer has to care about three *general tasks* – AoI management, game state processing and data-structure design – when building the game on top of RTF. If the game uses multiple servers, then multi-server parallelization and distribution also have to be taken care of by developers. Underneath these tasks for the developer, RTF provides a variety of low-level functionality like optimized event and entity serialization and communication, management of the game state and its possibly distributed processing. Overall, this separation of tasks among the developer and RTF reflects the overall approach of RTF sketched in Fig. 1(e): Providing high-level game engine-related functionality on top of an optimized communication middleware. While a technical discussion of the RTF can be found in (Glinka et al., 2007), the following subsections focus on the developer tasks and present the overall development methodology provided by RTF.

1. Task: Data Structure Design

The dynamic state of an online game is usually described as a set of *entities* representing avatars, NPCs or items in the virtual world. Besides entities, *events* are the other important structure in an online game engine for representing user inputs and game world actions. Both events and entities require hierarchical data structures for designing complex game worlds; they also have to be serializable in an optimized manner for efficient network communication. When using only a communication middleware, developers have to build data structures and serialization mechanisms from scratch, while using an existing engine requires the use of predefined entities and events, which reduces flexibility.
The RTF provides an optimized high-level entity and event concept enabling automatic serialization while still providing full design flexibility. When using RTF, entities and events are implemented as object-oriented C++ classes. The developer defines the semantics of the data structures according to the game logic. The only semantics of entities that is predetermined by the RTF is the information about their position in the game world. Entities, therefore, are derived from a particular base class `Local` of the RTF that defines the representation of a position for entities. This is necessary since the distribution of the game state processing across multiple servers is based upon the location of an entity in the game world. Besides this requirement of inheriting from `Local`, the design of the data structures is completely customizable to the particular game logic, as illustrated by the example of an avatar entity shown in Listing 1.

```
class Weapon : public RTF::Serializable;
class Avatar : public RTF::Local {
  private:
    rtf_int32 _ser_state_flags; // ducked, jump, etc
    rtf_int8  _ser_damageCount; // health value
    Weapon    _ser_weapon; // yielded weapon
    Avatar*   _ser_target; // the current target
    RTF::Vector _ser_velocity;
    RTF::Vector _ser_orientation;
    RTF::Collection _ser_inventory;
};
```

Listing 1: An entity written in the manner of RTF

In order to enable platform independence and the required introspection, RTF defines primitive data types to be used (e.g., `rtf_int32`). Also, easy-to-use complex data types for vectors and collections are provided to the developer. Overall, more complex entity and event data structures can be easily defined using these existing primitives.

Automatic serialization and network transmission
In online games, entities and events are continuously transmitted over a network. Therefore, these hierarchical data structures have to be serialized in an optimized manner. However, there is no standard serialization mechanism in C++, such that the developer has to define and implement a network-transmittable representation for each entity and event of a game when using a traditional communication middleware. As an alternative, most engines provide high-level scripting capabilities with automatic serialization, but they decrease flexibility and possibly also performance due to the abstraction overhead from native C/C++.
RTF provides automatic and native serialization of the entities and events defined in C++, considers marshalling and unmarshalling of data types and optimizes for bandwidth consumption of the messages. The RTF solves this problem for the developer by providing a generic communication protocol implementation for all data structures following a special class hierarchy. All network-transmittable classes inherit from the base class `Serializable` of the RTF. It is possible to have primitive types, pointers to Serializables, and Serializable objects themselves as attributes of a class. For all entities and events implemented in this manner, the RTF automatically generates network-transmittable representations and uses them at runtime. This code generation is seamlessly integrated in the data compilation process, as indicated in Figure 5. To enhance a class with the

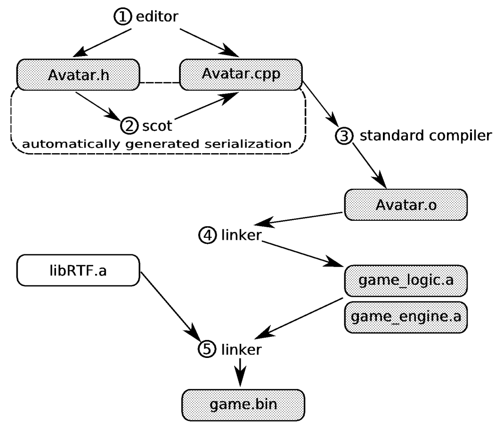


Figure 5: Schema of the development cycle from the class sources to the binary of the game

serialization functionality of RTF, the particular source files are processed with our tool called *scot* (*S*erialization

*CO*de *T*ool). Figure 5 illustrates that the usage of scot is seamlessly integrated in the development cycle, where it generates optimized serialization and communication code directly into the compiled classes. This figure also illustrates how the RTF is linked into the game software. Items that are part of the RTF are shown unshaded. Other items belong to the application, such that the developer of the application is responsible for them. The first step (① in the figure) is to implement the class as designed for the game logic. The next step ② is to generate the implementation of the network-transmittable representation of this class using *scot*. This step circumvents the usual implementation of class serialization by hand which would be necessary for fully custom development. The following steps ③ and ④ are the usual compilation and linking steps for the application level sources. The final step ⑤ is to link the application binaries for game logic and game engine against the RTF library to the complete executable.
Overall, this approach allows to use native C++ data structures for entities and events, while avoiding to implement the cumbersome, network-specific serialization by hand. Additionally, our approach is open to be combined with custom, engine-specific scripting capabilities: for example, LUA-bindings for high-level behaviour scripting can easily be added into the C++-based core data structures.

2. Task: Game State Processing

Using RTF, the developer still has to implement an own real-time loop for computing the game state updates, the so-called *ticks*. However, RTF provides substantial support functionality for implementing and running this real-time loop; in particular, it provides event and entity manager classes the developer can directly work with, as illustrated in Figure 6 for a server loop.

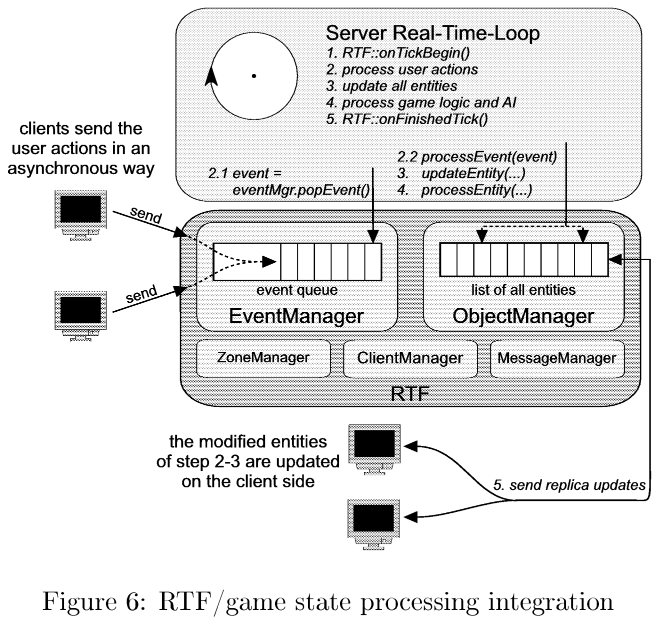


Figure 6: RTF/game state processing integration

The figure shows the following standard steps of the real-time loop to be implemented by the game developer: process user actions (step 2. in the figure), update entities (3.) and process game logic (4.). In each runtime step, the processing code interacts with components of the RTF. The runtime communication and distribution tasks handled by the middleware happen before and after the processing steps of the games engine. Therefore, the developer informs the RTF about the begin and the end of each *tick* using the methods `RTF::onTickBegin()` (step 1. in the figure) and `RTF::onFinishedTick()` (5.). The particular tasks – transferring events, transferring entity states, and managing distribution – are handled within the RTF in these methods and are described in the following. This schema of integrating a communication and distribution middleware into continuous processing of the game state is an important finding of our studies on using a distribution middleware in online games: it frees the developer from low-level network programming as is the case when using a conventional communication middleware, but still provides full design flexibility for the real-time loop as opposed to using an existing game engine with a predefined processing loop.

3. Task: AoI Management

An *Area of Interest (AoI)* concept assigns each avatar in the game world a specific area where dynamic game information is relevant and thus has to be transmitted to the avatar's client. AoI optimizes network bandwidth by omitting irrelevant information in the communication. If done in a fine-granular manner, it avoids wallhack-like cheating (Yan and Randell, 2005; Choo, 2001) at the client side which makes walls semi-transparent and reveals hidden opponents outside of the AoI. Unfortunately, determining the relevant set of entities for a particular client can be quite compute-intense, such that the AoI management, for which different algorithms are compared in (Boulanger et al., 2006), has to be implemented in an efficient and optimized manner.

RTF supports the custom implementation of arbitrary AoI concepts by offering a generic publish/subscribe interface for inter-entity visibility. The engine determines continuously which entity is relevant for a client avatar and notifies the RTF of each change of an "interested" relation through a `client.subscribe(...)` and `client.unsubscibe(...)` call. The RTF automatically takes care that the entity is available and always updated at the client or is removed from the client, respectively. The RTF also takes care that entities are removed from AoI of all participating clients if they disappear at a certain server. This is important, as entities can move from one zone into another and thus maybe leave the AoI of clients implicitly.

Transferring entity states

Every time the game engine has finished the processing of a new game state, the RTF automatically synchronizes the state of entities between the distributed processes depending on the indicated AoI relations. When an entity is replicated to another process (for example, all the entities within the AoI of a particular avatar to its client), the state of the remote copy has to be updated. Since the computations are usually performed in repeatedly executed cycles (*ticks*), the best way to perform state updates is after a computation cycle has finished, thus preventing propagation of intermediate states and read-write conflicts between the middleware and game engine.

The use of RTF simplifies this task of transferring entity states for the developer. He only has to inform the middleware that a computation cycle of the game engine has ended by invoking `RTF::onFinishedTick()` (step 5. in Figure 6). The necessary flow of information to update the game state on all participating processes is determined by the RTF upon the specified distribution. At runtime, the middleware automatically creates messages for changed objects and transmits them. This is done using the network-transmittable representations that were generated for the data structures using scot during the development cycle. The RTF-part of the receiving process of such an update message automatically determines the object related to the messages and writes the updated data directly to the right object. Since the data is directly written to the objects used inside the game engine, this writing step is again triggered by the developer, e. g., directly before a computation cycle, by invoking `RTF::onTickBegin()` (step 1 in Figure 6), to prevent read-write conflicts.

MULTI-SERVER TASKS

The general development tasks described in the previous section are fundamental for any client-server based game. However, when the game should be massively multiplayer, a multi-server approach is required for achieving a high scalability for supporting a large amount of users. This section describes what parallelization approaches are supported by RTF and discusses how developers can easily use them for building MMO worlds.

Parallelization Concepts supported by RTF

RTF currently supports three parallelization concepts for scaling virtual world environments: *zoning* (Cai et al., 2002; Rosedale and Ondrejka, 2003) and *instancing*, as commonly used in contemporary MMORPG, and *replication* (Bharambe et al., 2006; Müller and Gorlatch, 2006a; Müller et al., 2007), an alternative parallelization approach recently discussed in academia. All these approaches aim at different scalability dimensions: zoning allows large user numbers in large MMORPG worlds, instancing runs a large number of game world areas independently in parallel, and replication targets high user density for action- and player-vs-player-oriented games. Figure 7 illustrates the overall combination of these approaches in a single game as provided by RTF.

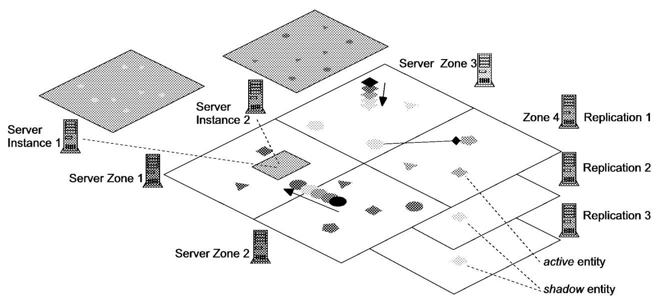


Figure 7: Combination of zoning, instancing, and replication for a single game world in RTF

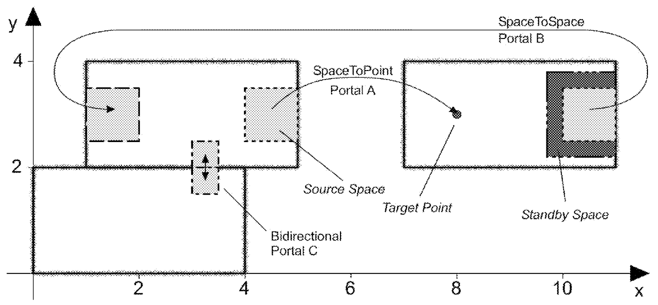
The overall goal of integrating these approaches into RTF, as discussed in detail in (Müller and Gorlatch, 2006b), is to provide general and dynamic scalability for all game genres within a single framework, which can be operated on demand in a Grid computing environment. The following discussion sketches the envisaged methodology for developers for using these multi-server parallelization concepts.

4. Task: Parallelization and Distribution

If the multi-server capabilities of RTF are used, then, in addition to the general tasks 1-3 (data structure design, state processing and AoI management), the developer has to segment the game world into zones, instances and replication areas and to define the connections between them in form of portals. Using this information, RTF automatically assigns servers to each of the segments and connects each client to the particular segment the associated avatar resides in. If the user moves his avatar through a portal area, RTF will recognize this and automatically issue a connection transfer, making the server of the new segment responsible for processing the avatar. Each of the participating servers runs the normal server real-time loop discussed in the previous section for its associated segment – the RTF internally handles connection migration and distributed entity management.

RTF offers a dedicated interface for specifying how the overall game world is segmented into a combination of zones, instances and/or replication areas. Figure 8 illustrates a two-dimensional game world example with three zones and portals of various types.

The definition of a zone, as illustrated in Fig. 8, consists of an ID, the occupied space and a flag if it is allowed to replicate the zone across multiple servers. For the portals, an entrance area and a connected destination area are given. During runtime, all zones are assigned to the set of available servers. Fig. 8 shows different portal



Code example:

```
Zone a = Zone(0, Space(1, 2, 0, 4, 2, 0), PLAIN);
Zone b = Zone(1, Space(7, 2, 0, 4, 2, 0), REPLICATE);
Zone c = Zone(2, Space(0, 0, 0, 4, 2, 0), PLAIN);
Portal& pA = *new SpaceToPointPortal(entranceA,
   destinationA);
Portal& pB = *new SpaceToSpacePortal(entranceB,
   destinationB);
Portal& pC = *new BidirectionalPortal(spaceOne,
   spaceTwo);
```

Figure 8: Segmentation Specification Example

types supported by RTF for expressing various transfer relations (uni-/bidirectional, space to space, space to point) how to move over to a different area of the game world.

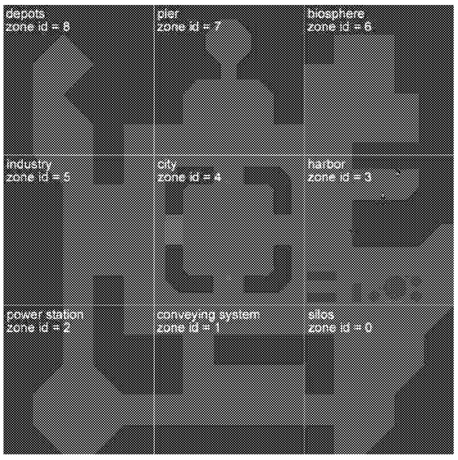
Overall, game developers only have to implement mechanisms at a high level of abstraction in the RTF multi-server task. In particular, they can start developing any multi-server game engine as a single-server engine at begin and then easily switch over to a scalable multi-server engine. For this switch, developers, in most cases, only have to segment the game world into zones, instances and replication areas, possibly implementing segment-related game logic mechanisms on top of the already existing specified entity and event data structures.

IMPLEMENTATION AND "OFFSHORE" CASE STUDY

RTF is currently under development with a strong emphasis on studying and optimizing mechanisms in the area of distributed real-time computation and communication, continuous processing parallelization and development methodology of distributed virtual environments and online games. The work on RTF is part of the *edutain@grid*[1] project funded by the EC IST, where it provides the fundamental real-time computation and communication middleware for interactive applications and online games operated in a Grid computing infrastructure.

Based on the current version of RTF, we are developing *Offshore* as a MMOG case study taking place in an aquatic metropolis. Figure 9(a) illustrates the corresponding overall game world being segmented into nine zones, while Figure 9(b) shows a screenshot of the current client prototype giving an overview on the game world from an elevated position.

[1]http://www.edutaingrid.eu/



(a) Aquatic Metropolis Segmentation

(b) Overview Screenshot of Current Prototype

Figure 9: Offshore: Aquatic Metropolis Case Study

The Offshore case study has been used so far for technical evaluation and for verification of the game development process and methodology. Here, RTF first supported the general development tasks for single-server operation, after which the game engine has been successfully switched over to multi-server processing by segmenting the game world.

CONCLUSION AND RELATED WORK

In this paper, we have studied and summarized development methods for online games and demonstrated to what extent the low-level custom development can be substituted by a high-level approach using game middleware for single-server and for scalable multi-server engines. We described our RTF system which is used to support the human developer in the development process. The RTF enables a smooth transition of single-server online games to the multi-server architecture by its integrated distribution capabilities. Since RTF focuses on the processing part of games, it puts no constraints on the remaining development tasks as, e.g., graphics or game logic implementation.

In comparison to existing approaches in the field of communication middleware like *Net-Z* (Quazal, 2006),

HawkNL (Hawk Soft, 2006) or *RakNet* (Rakkarsoft, 2003), RTF provides a much higher level of abstraction featuring automatic entity serialization and hides nearly all of the technical network communication aspects. On the other hand, RTF is much more flexible than reusable game engines like the *Quake* or *Unreal* engines, because it is not bound to a specific graphics engine and leaves the real-time loop implementation to the developer, supported by high-level entity and event handling mechanisms. The multi-server capability of RTF allows to easily incorporate three different parallelization and distribution approaches and is open to be extended to future approaches. This flexible support of different parallelization concepts allows RTF to be usable for a wider range of MMOG concepts than existing multi-server middleware like Emergent Server Engine (EGT, 2007) or BigWorld (BIG, 2006) focusing mostly on the concept of zoning.
The current implementation of RTF already provides important features for developers to implement online games at a high level while still preserving design flexibility for single- and multi-server engines. Summarizing, RTF offers the following integrated functionality:

- The serialization mechanism liberates the developer from the details of network transmission programming.
- Communication is optimized with incremental updates to reduce the data sent over the network.
- Segmentation and distribution of the game world are described on an abstract level in game design.
- The proven multi-server distribution concepts zoning, instancing and replication, as well as their combinations, are supported.
- Distribution management and parallelization of the game state processing is fully handled by the RTF.
- The game logic and entities are implemented using C++ in a usual object-oriented way and are open to be integrated with state-of-the-art scripting capabilities.

Besides developing case studies using the offered multi-server segmentation approaches in combination for in-depth scalability evaluation and comparison, we plan to integrate additional features into RTF in the future. In particular, audio and video streaming as well as automatic game state persistence are highly interesting to be integrated for further enhancing the RTF as a comprehensive middleware for online games.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive comments helping to improve this paper. The work described in this paper is supported in part by the European Union through the IST 034601 project "edutain@grid".

REFERENCES

Bharambe A.; Pang J.; and Seshan S., 2006. *A Distributed Architecture for Multiplayer Games.* In *PACM/ USENIX NSDI.* San Jose, USA.

BIG, 2006. *BigWorld Technology www.bigworldtech.com.*

Boulanger J.S.; Kienzle J.; and Verbrugge C., 2006. *Comparing interest management algorithms for massively multiplayer games.* In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games.* ACM Press, New York, NY, USA. doi: http://doi.acm.org/10.1145/1230040.1230069.

Cai W.; Xavier P.; Turner S.J.; and Lee B.S., 2002. *A Scalable Architecture for Supporting Interactive Games on the Internet.* In *Proceedings of the 16th Workshop on Parallel and Distributed Simulation.* IEEE, Washington, D.C., 60–67.

Choo C., 2001. *Understanding Cheating in Counter-Strike* http://www.fragnetics.com/articles/cscheat/print.html.

EGT, 2007. *Emergent Game Technologies, www.emergent.net.*

Glinka F.; Ploss A.; Müller-Iden J.; and Gorlatch S., 2007. *RTF: A Real-Time Framework for Developing Scalable Multiplayer Online Games.* In *NetGames 2007: Proceedings of 6th Annual Workshop on Network and System Support for Games.* Melbourne, Australia, 81–86.

Müller J. and Gorlatch S., 2006a. *Rokkatan: scaling an RTS game design to the massively multiplayer realm. ACM Computers in Entertainment*, 4, no. 3, 11. ISSN 1544-3574. doi:http://doi.acm.org/10.1145/1146816.1146833.

Müller J. and Gorlatch S., 2006b. *Scaling Online Games on the Grid.* In M. Merabti; N. Lee; K. Perlin; and A.E. Rhalibi (Eds.), *GDTW 2006 - Fourth International Game Design and Technology Workshop and Conference.* Liverpool John Moores University, Liverpool, UK, 6–10.

Müller J.; Gorlatch S.; Schröter T.; and Fischer S., 2007. *Scaling Multiplayer Online Games using Proxy-Server Replication: A Case Study of Quake 2.* In *16th IEEE International Symposium on High Performance Distributed Computing (HPDC 2007).* IEEE, Monterey, California, USA.

Quazal, 2006. *Net-Z www.quazal.com.*

Rakkarsoft, 2003. *RakNet www.rakkarsoft.com.*

Rosedale P. and Ondrejka C., 2003. *Enabling Player-Created Online Worlds with Grid Computing and Streaming <http://www.gamasutra.com/resource_ guide/20030916/rosedale_ 01.shtml>.*

Hawk Soft, 2006. *HawkNL www.hawksoft.com/hawknl.*

Yan J. and Randell B., 2005. *A systematic classification of cheating in online games.* In *NetGames '05: Proceedings of 4rd ACM SIGCOMM Workshop on Network and System Support for Games.* ACM Press.

A FRAMEWORK FOR NETWORK-AGNOSTIC MULTIPLAYER GAMES

Patric Kabus
Alejandro P. Buchmann
Databases and Distributed Systems Group
Technische Universität Darmstadt
64289 Darmstadt, Germany
{pkabus,buchmann}@dvs1.informatik.tu-darmstadt.de

Abstract

Producing computer games is a complex and resource-intensive task nowadays. Since this task involves many people with a great variety of expertise, a clear separation of concerns within the project is essential. Especially multiplayer online functionality, which is probably the most popular aspect currently, raises the complexity significantly. Getting this aspect separated allows game developers to focus on design issues rather than on writing sophisticated network code. This paper presents a framework that provides a complete abstraction from network related implementation details.

1. Introduction

"Ten or twenty years ago it was all fun and games. Now it's blood, sweat, and code."[4] In the early days, computer games could be developed by a only few people or even a single person. Most of the work was about writing optimized game code for hardware with very limited resources. Due to these resource limitations other aspects of a game, like design, graphics or sound, had to remain very simple. Today's games are multi-million dollar projects including dozens of highly specialized professionals like 3D artists, level designers, musicians or storytellers. Yet at the core of a game remains a large piece of code, the game engine. It's purpose is to combine all the digital content (called *assets*) created by various artists into a playable game.

Since asset creators are mainly artists, they usually have a very limited knowledge about writing code. Thus, the interface to the game engine must require a minimum of programming skills. But at least when creating assets that exhibit behavior (like an enemy which is controlled by the computer) one usually cannot avoid getting in touch with coding. For this purpose, easy-to-learn scripting languages are incorporated into the game engine. Together with predefined methods, which handle common in-game functionality (e.g. a *move(x,y)* method, which moves a game object to a certain position and automatically performs path finding and collision detection) and can be called from within a script, the programming task is kept as simple as possible.

Hiding complexity gets even more difficult when network gaming comes into play. Network functionality is probably the most important gaming feature today, with networks ranging from a few nodes in a LAN environment to a few thousand nodes in Massively Multiplayer Online Games. Providing a sufficiently consistent view of the game on all nodes of the network is non-trivial. Consequently, asset creators should not be burdened with the task of handling inconsistencies or performing manual synchronization of game objects. However, even programmers will benefit from being shielded from consistency issues. Modern game engines are complex systems composed of various modules. It is generally a good idea to keep consistency-related code within a single module, allowing developers of other modules to focus on their specific tasks. Finally, a clean separation of concerns is a good basis for reusability. Nowadays many game engines are reused by multiple game projects and selling engine licenses is even part of the business model of some producers.

In this paper we present a gaming framework that completely shields game developers from network and consistency issues. Unlike existing game engines, our system does not only abstract from a specific network architecture. Games built using our framework can be deployed in many different environments by simply changing a configuration file. Besides running the game in single player mode locally, we currently support three modes: classic Client/Server, a Peer-to-Peer mode usually known as *Replicated Simulation* [2] and a hybrid one, called *Mutual Checking* [13]. In the following we will refer to these modes as *CCS*, *RS* and *MC* respectively. All three modes provide some protection against cheating, an essential property for today's games. The underlying abstraction allows developers to extend the framework with their own custom network modes, if necessary. Without the need to commit to a specific network architecture, it is much easier to reuse a game engine in different projects. Furthermore, game developers may allow players of a certain game to change the network architecture by simply altering a configuration file. If a group of players doesn't trust a single node to host a server for a Client/Server session, they could switch to Peer-to-Peer mode where each node maintains its own local copy of the game state. Finally, home-brewn or independent games as well as academic projects may benefit from the possibility of playing around with different network settings without having to change their game code.

In the following section we discuss related work. Section 3 describes the architecture of our framework, while in Section 4 we delve into some implementation details.An example game that we have implemented to show that the framework can actually be used for games is shortly presented in Section 5. Since cheating is an omnipresent issue in online games, Section 6 briefly examines this topic. Finally, Section 7 concludes the paper.

2. Related Work

To our knowledge, no scientific work exists that deals with the complete abstraction from different network architectures within a gaming context.

Kaneda et al. [14] propose a middleware that allows the reuse of Client/Server-based games in a Peer-to-Peer setting. The authors argue that this may be necessary if, for some reason, the producer of a game discontinues to provide the necessary servers. Each player has to install an application on his node which connects to the other player nodes in a P2P fashion. The application acts as a fake server to the local game application by capturing and answering the game related traffic. The global state is synchronized between all nodes, making it appear as if all players were connected to the same server. A major drawback of this approach is that the game's network protocol must either be openly specified or reverse-engineered. Every implementation of this middleware is specific to a certain game and hardly reusable for other games.

The Real-Time Framework (RTF) [11] also aims at providing an abstraction from the underlying network, but from a different perspective. It does not address pure P2P or hybrid architectures. Instead, it abstracts from the way a multiplayer game is distributed in a multi-server architecture. RTF supports three distribution concepts, namely *zoning*, *instancing* and *replication*. Similar to our framework, RTF provides a way for game developers to deal with game objects without concerning about synchronization issues. The paper does not go into detail about the underlying network

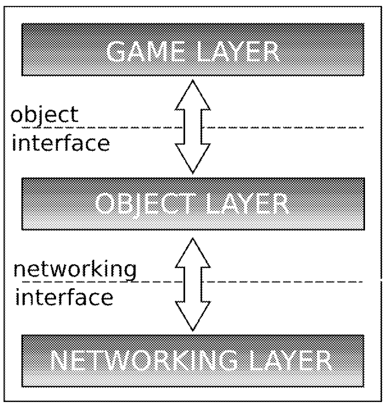


Figure 1. High-level overview

architecture. Thus, it is currently difficult to say in which parts our works complement each other.

Modern commercial game engines usually provide some level of network abstraction, but are mostly tied to a certain network architecture. The technology overview of the upcoming *Unreal 3 Engine* [10] states that it will be possible to run games either in a C/S or P2P mode. Unfortunately, the architecture is not openly documented and details thus unavailable. It is uncertain whether the engine supports a transition from P2P to C/S or vice versa without altering code. Moreover, it is very unlikely that the engine easily supports hybrid or custom network architectures.

3. Architecture

Our proposed architecture can be divided into three layers and two intermediate interfaces, as shown in Figure 1. The discussion in this section remains on a rather abstract level; important implementation details are addressed in the following section. We start on the highest layer, the *game layer*, and work our way down to the lowest one.

3.1. Game Layer

This layer contains components like the input manager, the presentation manager and the game engine. The input manager is responsible for accepting commands issued by the player via keyboard, mouse, a gamepad or any other kind of input device. The presentation manager provides the player with an audiovisual representation of the game and probably even some haptic feedback. At the core of any game there is a game engine which manages all the assets that the game is composed of and controls their behavior which is defined by the game logic. The engine may also manage the other components of the game and perform additional tasks like logging in and out of a network game. Although virtually every game is made of components like those mentioned above, actual implementations may show a great variety. Professional games today will most likely consist of much more components, while simple games may combine everything into a single one. Note that these components do not necessarily have to be implemented by the game developers themselves. There are many implementations that can be bought off the shelf or are available for free. The game layer is connected to the lower layers via the *object interface* which serves as the top-level abstraction for our framework.

3.2. Object Interface

The central element of a game is a collection of objects that constitute the state of the virtual world. The *game objects* may represent nearly every aspect of the game: the players' avatars, computer-controlled enemies or allies, interactive objects (like vehicles and machines) or completely static objects (like trees and walls). Even purely logical entities that have no perceptible representation (at least none that is perceived by a human player), like containers that aggregate game objects into a logical unit or triggers that activate in-game actions, may be modelled as game objects.

In a multiplayer game, multiple participants share the same game world and thus need to have a consistent view of its state. If the players are located on different nodes of a network, local copies of the game objects, which as a whole represent the state, need to be synchronized. Our architecture hides this synchronization effort completely, allowing a game developer to access and manipulate game objects as if they were local. Consequently, the interface that is presented to the developer allows the creation and deletion of game objects as well as reading and changing their state. The components of the game that run on a player's node may work as usual. E.g., the input manager translates input events into appropriate changes of the player's avatar object. The presentation manager may read the state of the game objects and generate audio-visual and haptic feedback. And last not least, the game engine changes game objects whenever the rules and the logic of the game require it. Furthermore, the interface provides methods that perform the necessary bootstrapping when setting up or joining a network session as well as methods that leave or shut down a session. The following subsection describes the implementation of the object interface.

3.3. Object Layer

The *object layer* is responsible for holding up the illusion that all game objects seem to be local and can be manipulated without concerning about synchronization. Furthermore, it has to handle the necessary bootstrapping when a new node joins the network or cleanup when a node leaves.

In our framework, every game object has an owner which keeps a master copy of it. Whenever a node wants to change a local copy of an existing game object it must send a *request* to the owner. If the request is granted, the owner changes the object state accordingly and sends an *update* to every node that keeps a local copy (including the one which has sent the request). Whenever a node receives an update sent by the owner of an object, it will perform the contained change on its local copy. This way we achieve a single-copy consistency since the owner of an object serializes all operations on it. Note that in the MC example a group of region controllers acts as the owner of a game object. Each region controller in the group receives a request, processes it independently and sends an update. Whichever node has a local copy will receive the updates and elect the one which holds the majority. Please refer to [12] for details, including a discussion on consistency.

Until now we have only talked about existing game objects which contain the owner information in their metadata. What remains is the question of how ownership is determined when creating a game object. Burdening a game developer with this task when creating an object would break our abstraction. To avoid this, the object layer has to provide a factory method for each supported architecture which encapsulates the knowledge about determining ownership. A game developer simply creates an object (using the object interface) and, depending on the network configuration, an appropriate factory is chosen. In our CCS example, the server is the owner of all game objects and whenever a client needs to create one, the respective object factory determines the server as the owner of this object. In contrast, in the RS example a peer node always takes ownership of objects it creates. Finally, in the MC setting, the owner id addresses the whole group of region controllers. As we can see, a node does not only create objects for itself but it may also request the creation on another node. Thus, the creation of a new game object is treated the same way as the manipulation or deletion of an existing one: it is sent as a *request* to the future owner. Upon receiving and processing a creation request, the owner sends an *update* to all nodes the creation may concern.

Note that all operations needed for the management of an object can be mapped onto two types of messages, namely a request message and an update message. We still need a third kind of message to inform nodes about organizational events like the joining and leaving of nodes. Whenever a node joins the network it sends an *announcement* to the existing nodes. Every node that owns a game object which is relevant for the newly joined node may now send an update containing the current state of this object. This way, a new node can be provided with the current state of the game. When the node leaves again, it may inform the other nodes that it won't process request or updates anymore. If the objects it owns are still needed, it may request the creation of replacements on remaining nodes.

3.4. Networking Interface

The discussion above showed that the messages needed for game object synchronization and node housekeeping may be divided into three categories: *requests*, *updates* and *announcements*. What we have to make sure is that messages are sent to the appropriate recipients. For instance, a client in the CCS example is never interested in receiving request messages, since it doesn't own any objects. On the contrary, the server doesn't care about updates since — due to the fact that it owns all the objects — it is the only one to send them. To complicate matters, nodes join and leave and thus the list of senders and recipients changes dynamically.

However, this problem is not new and a solution for it is well-established: the *Publish/Subscribe* (pub/sub) paradigm [9]. One of the main advantages of pub/sub systems is the decoupling of message senders from message receivers. Participants of such a system only need to know what *kind* of messages they want to send. They do not need to know who are actually the recipients of these messages. The other way round, receivers only need to know what kind of messages they are interested in, not who may actually be sending them. The sending of messages of a certain kind is called a *publication*, while registering interest for a certain kind is called a *subscription*. The pub/sub system matches every publication to its respective subscriptions and thus takes care that a message will reach its intended recipients. Both, publishers and subscribers, may join and leave dynamically without requiring other participants to take notice of this.

Applying this concept to our framework avoids that owners of game objects and keepers of local copies have to be aware of each other. Any node which wants to manipulate an object simply publishes an appropriate request message. Owners of game objects are subscribed to this kind of message and thus will automatically receive change requests. After processing the request they publish an update and nodes which keep a local copy will receive the change since they are subscribed to update messages. To sum it up, the networking interface has to provide means to issue publications and register subscriptions.

3.5. Networking Layer

The lowest layer of our framework's architecture is responsible for implementing the pub/sub methods that are offered by the networking interface. Publications have to be routed over the network to the appropriate subscribers. This layer also has to take care of managing publishers and subscribers which dynamically join and leave the network. Please refer to Section 4.2 for a detailed discussion on implementation issues regarding the networking interface and layer.

3.6. Concluding Overview

Figure 2 gives a more detailed overview of our three-layer framework including its two interfaces. On top is the game layer which connects to our framework via the object interface. Within the game layer, one may simply manipulate game objects as if they were local without paying attention to the layers below. The only thing that may be noticeable is a delay until a manipulation actually takes effect. (This delay may be hidden from the player by using commonly known techniques like *Dead Reckoning* [18].) Below the object interface is the object layer where the configuration of the desired network architecture takes place. A node has to define to which topics it publishes and subscribes and which factory it uses for creating objects with the correct ownership. Supporting different network architectures means providing the appropriate definitions and factories. This layer is also responsible for handling the login and logout of nodes. Finally, the networking interface serves as an abstraction to the message handling. By using a generic interface one may use different implementations in order to fulfill certain performance or scalability requirements or simply to experiment.

4. Implementation Issues

This section will give more insight on some of the implementation issues of the framework architecture. In order to be able to speak of a complete framework, we must provide more than merely a networking middleware. A gaming framework should also provide standard components that are located on the game layer, like the input and presentation managers and the game engine. However, our research focus lies on the network transparency which is not directly related to these components. Additionally, there exists a vast amount of — free and commercial — implementations that may easily be integrated. Consequently, we only provide implementations of these components to the extent they are necessary for our example game (see Section 5). In the following, we focus on the object model and the pub/sub system implementations. The former is the part that game developers have to deal with if they want to create a game that is agnostic of the underlying network architecture. The understanding of the latter is important if one wants to extend the framework with new network architectures or optimize existing ones.

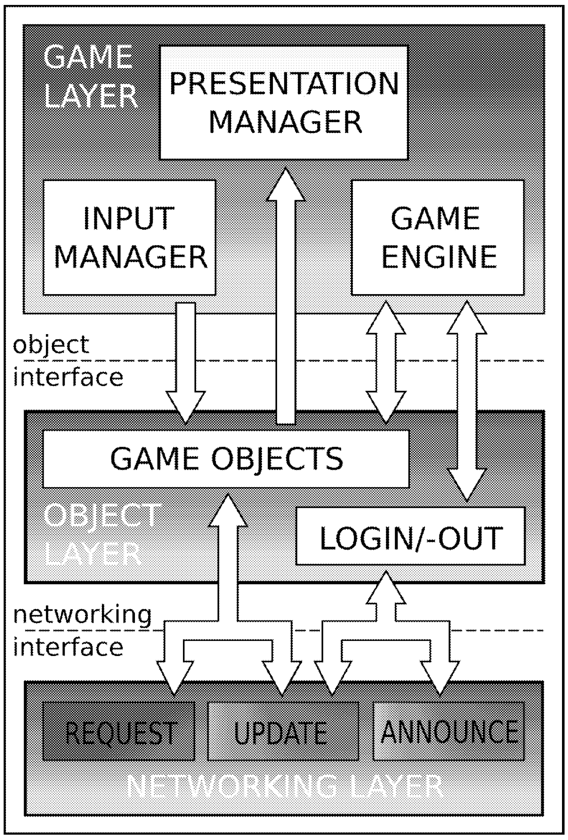


Figure 2. Detailed overview

4.1. Object Model

Many different ways exist to manage objects within a virtual gaming environment [3, 5, 7, 8].We have chosen an approach that provides high flexibility as well as ease of use. It is completely data-driven, i.e. every aspect of a game object can be changed at any time dynamically without the need for a recompilation. This speeds up the development process and should make it easy to integrate this framework into the workflow of a game developer.

The type system of the game object model doesn't rely on static types defined by the programming language's class hierarchy (our prototype is implemented in Java). Instead, a generic GameObject class is used which is assigned a *game object type* dynamically. The type itself consists of a number of state variables and methods plus possible base classes. Every type inherits all of the states and methods from its base types and thus new types can be easily composed of existing ones. Game object methods may be defined in any scripting language which is available for the Java Scripting Platform [16]. The type definition itself is currently written in XML, but by providing an appropriate import plugin, any format may be used. Figure 3 shows how a simple type definition may look like. The example shows

```
<type id="bomb">
  <state name="x" default="50.0"/>
  <state name="y" default="50.0"/>
  <state name="countdown" default="10"/>
  <script name="tick" lang="js">
    <![CDATA[
      countdown -= 1;
      if (countdown == 0)
      {
        go.execute("explode");
      }
    ]]>
  </script>
</type>
```

Figure 3. Example of a game object type definition

a definition of a bomb which contains a two-dimensional position state and a detonation counter. The “tick“ method, which is always called by the node owning the object, allows to trigger time-dependent behavior like decrementing the internal counter of the bomb. Note that in our example game, the tick method doesn't decrease the counter every time it is called but takes into account a variable that contains the amount of time that has passed since the last call. From within the script additional references may be used, e.g. the current game object (“go”), the object manager (allowing access to other game objects) or the object's state variables.

All game objects are stored hierarchically within a tree. The object interface allows insertion, manipulation and deletion of objects. Furthermore, a query method allows finding objects that match a certain regular expression. Finally, it is possible to define arbitrary groups of objects as *views* which provide an easy way to access objects that match certain criteria.

Inserting and deleting objects as well as changing their state triggers the notification of registered listeners. A certain listener is responsible for automatically publishing appropriate requests and updates whenever necessary. Listeners may also be used to manage the membership of views. If a new object is created or an existing one changes its state, any listening view may add the object if it matches the view definition. The same way, objects can be removed if they are deleted or do not match the view criteria anymore.

4.2. Publish/Subscribe

To demonstrate how a pub/sub messaging service can be integrated into our framework we have chosen a simple form of pub/sub, a *topic-based* approach. Later on we will discuss how more powerful approaches may be used to lower bandwidth consumption or improve scalability.

4.2.1 Topic-based Publish/Subscribe

As the name implies, in a topic-based pub/sub system participants publish and subscribe to *topics* and each topic represents a certain kind of message. The obvious way to model our communication is to assign each type of message — requests, updates and announcements— its own topic. We first demonstrate how requesting a change and sending an update works within the three example architectures we have implemented. Next, we will show how the announce topic may be used for handling nodes joining and leaving the network.

The following is a short overview of how those architectures distribute the ownership of game objects. For more detailed information please follow the references given in Section 1.

Classic Client/Server (CCS) The central server is the owner of all objects and thus keeps all master copies. Clients only store local copies which are updated by the server.

Replicated Simulation (RS) Each peer may own certain objects for which it keeps the master copies. It stores local copies of the objects owned by other peers.

Mutual Checking (MC) In order to avoid arbitrary manipulations by malicious nodes, each object is owned by multiple region controllers (RCs). Thus, each RC keeps its own master copy of an object and any change request has to be sent to each RC. After changing the state of a master copy, each RC sends an update to the local copies on the clients. The client compares the update messages and elects the one that holds the majority.

Figure 4 shows the request/update process in the CCS context. Client 1 wants to change an object and publishes a message to the request topic. The server which owns all objects has subscribed to this topic and thus receives all requests. After performing the requested changes the server publishes a message containing the changes to the update topic. All clients, including the one that has sent the request, are subscribed to this topic and receive the update.

In the RS context (Figure 5), a peer that wants to change an object publishes a request. All peers within the system are subscribed to the request topic, but only the owner of that object needs to process the request. The state update is then published and received by all peers, since each of them is subscribed to the update topic.

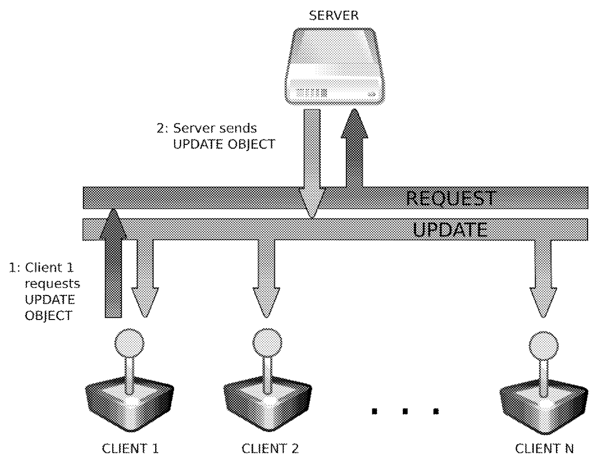

SERVER
2: Server sends UPDATE OBJECT
REQUEST
UPDATE
1: Client 1 requests UPDATE OBJECT
CLIENT 1
CLIENT 2
CLIENT N

Figure 4. Request/update in CCS mode

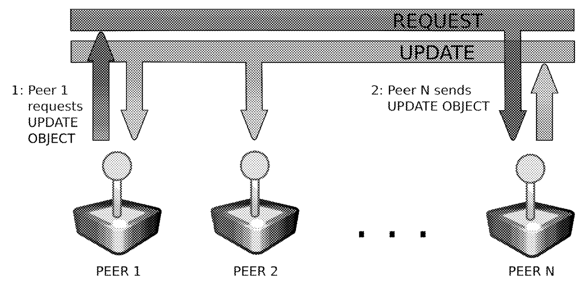

REQUEST
UPDATE
1: Peer 1 requests UPDATE OBJECT
2: Peer N sends UPDATE OBJECT
PEER 1
PEER 2
PEER N

Figure 5. Request/update in RS mode

Our last example, the MC context (Figure 6), is very similar to the CCS setting. Instead of having a single server, all RCs are subscribed to the request topic. After performing the requested change, each RC publishes an update. The clients, which are subscribed to the update topic, receive all updates from the RCs. Before an update will be performed, the correct one is elected out of the received updates.

To handle events like nodes logging in and out of the system, a third topic, called *announce*, is used.

Whenever a new player joins the game, an object has to be created that represents that player. The nodes already in the system need to be informed about the state of this new player object. Figure 7 illustrates this process in the CCS context. The server, which is subscribed to the announce topic, receives a login announcement published by the new client. It creates a new avatar object representing that player and publishes an appropriate update. This update is received by all clients, since they are subscribed to the update topic.

After logging in, the new client needs to be supplied with the current state of the game. For this purpose, every node that owns game objects must be subscribed to the announce topic. Upon receiving the login message, the owners may publish an update containing the complete state of their

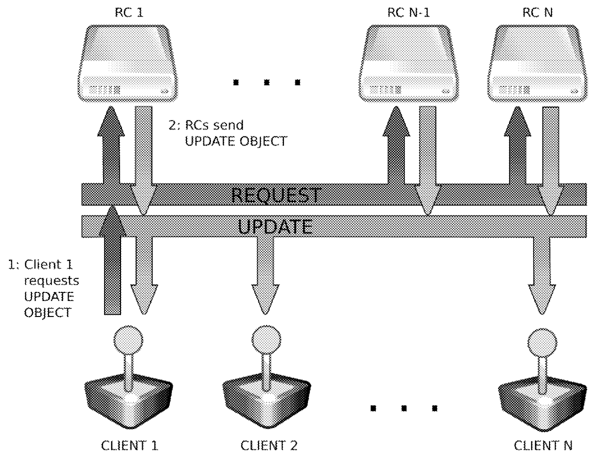


Figure 6. Request/update in MC mode

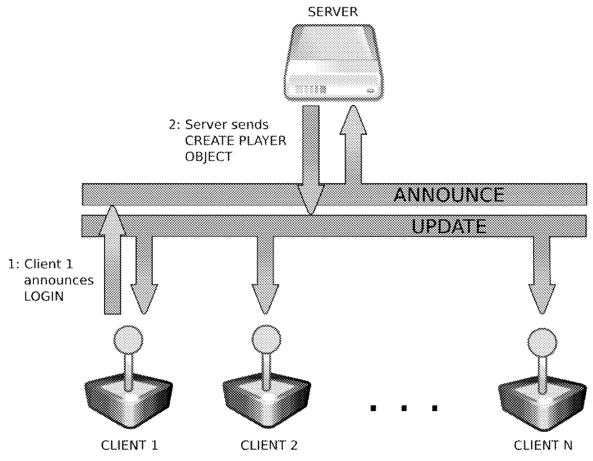

SERVER
2: Server sends CREATE PLAYER OBJECT
ANNOUNCE
UPDATE
1: Client 1 announces LOGIN
CLIENT 1
CLIENT 2
CLIENT N

Figure 7. Client login in C/S mode

master copies. Unfortunately, publishing the whole state of all master copies every time a node joins the game would be a waste of bandwidth. Every node subscribed to the update topic would receive the current state, even if its local copy is up-to-date. Optimizations that avoid this are discussed in the following subsection.

If a node wants to leave the network it simply publishes a log-out announcement. After receiving this message, the server publishes an update that removes the avatar object of the corresponding player from the game. Note that in our RS setting things are slightly more complex, since a leaving peer node may be itself the owner of certain game objects which are still needed. Bevor leaving the network, the node has to make sure that these objects are transferred to other peers. In order to do so, it can request the creation of an object on another peer by specifying this peer's id as the owner id.

4.2.2 Optimizations

An important way to reduce network bandwidth requirements in online games is to restrict the amount of updates a certain node receives. Obviously, a node does not need to be informed about changes of game objects that the local player can neither perceive nor interact with in any way. Limiting the update message to ones relevant for the player is commonly known as *Interest Management*. Instead of subscribing to all messages that are published to the update topic, a filtering based on the in-game position of objects may be performed.

For example, the Java Message Service [20] combines a topic-based pub/sub approach with filtering based on key/value pairs. Every update published may be enriched with additional properties that contain the position of the updated object. Only when the player's avatar is in the interaction range of that object the update will be sent to that player's node.

Instead of using a flat topic space, a hierarchical one may be employed to restrict messages to certain game regions. This approach is usually referred to as *subject-based* filtering [17]. E.g. in a game that uses a real-world setting, subjects like Earth, Earth.Europe and Earth.Europe.Germany could exist. Whenever an avatar enters a region (e.g. Germany) the node subscribes to the corresponding subjects. On the one hand, this makes sure that the node won't be bothered with unrelated messages of events that happen in a different country or even on a different continent. On the other hand, the node will receive messages of events that are relevant for the whole continent or even globally. Naturally, changes made by the node will be published to the appropriate subjects in the same manner, depending on their relevance.

Not only the addressing model but also the implementation of a specific model has an impact on performance and scalability. One very important performance criteria of network games is the latency when propagating updates of game objects. Usually nodes of gaming networks talk directly to each other, be it a client talking to a server or peers talking to each other. The delay of changing an object (i.e. issuing a request and getting a reply) equals the roundtrip time between nodes. In an implementation that wants to avoid higher latencies, a node that requests the change of an object must send the request directly to the owner node. Afterwards, the owner has to send its updates directly to all nodes which keep a local copy of the updated object. This way, extra delay caused by additional hops on the network path is avoided. In such an implementation a local software component running on each node can provide the pub/sub interface to the object layer. Internally, this component stores for all topics it publishes messages to a list of all subscriber nodes. Whenever a node publishes a message it can send it directly to the appropriate nodes. The subscription management service may be located on a separate node. Every time a node subscribes for a topic, the management service can inform the publishers about it. By sending a so called *advertisement*, a node can inform the management service about its intention to act as a publisher for a certain topic.

A further optimization is that whenever a node wants to change a game object that it owns, it may directly publish an update without the need to send a request first. But one should be aware that this may affect fairness. While the change is propagated to other nodes with the delay of a single hop it is perceived nearly instantly on the local node. This may enable the local player to react much faster than players on remote nodes. To avoid this, an artificial delay may be introduced (e.g. *Local Lag* [15]).

While the implementation above minimizes latency caused by network delays, it severely limits scalability. Think of a node in a Replicated Simulation which has to send updates to a very large amount of other nodes in the game. This way a node will soon reach the limits of its network connection, especially when using an asynchronous DSL connection with a very limited upload bandwidth. This is where pub/sub systems that rely on intermediate brokers play out their strength. While introducing additional delays for message delivery, the intelligent routing and filtering mechanisms can minimize bandwidth and connectivity requirements on the game nodes.

5. Example Game

For demonstrating the feasibility of our approach, we implemented a game that includes many important aspects found in today's games. These aspects include a graphical representation, changes in object state through player input or progress of time and interaction between game objects. While in our example they remain very basic, our framework imposes no limits onto their implementation. Rich three-dimensional graphics and sound are possible as well as control of game objects through complex artificial intelligence.

Our game is a simplified version of a famous multiplayer game concept that has been implemented by the open-source game XBlast [1]. Every player controls an avatar which may move freely around the game field. By pressing a button, he can place a bomb at his current location. Placing the bomb starts a timed detonator and when the countdown reaches zero the bomb explodes. All avatars that are in the vicinity of the detonation are removed from the field and, as in the original XBlast game, the last remaining player wins. Figure 8 shows a screenshot of the game.

As intended, the same game code can be used within all three network architectures without any modifications.

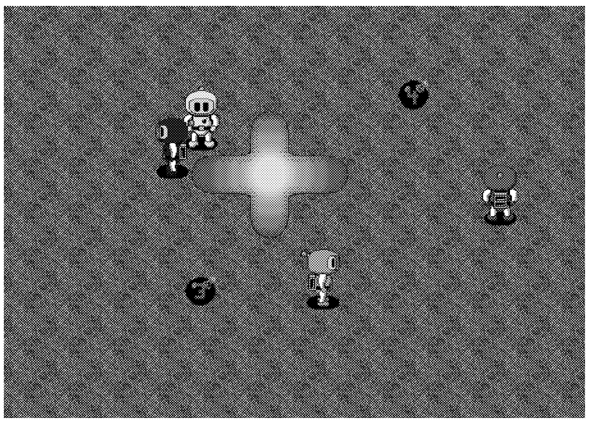


Figure 8. Example game

6. A Word on Cheating

No multiplayer online game today can come along without some protection against cheating, since the possibility to cheat poses a major threat to the fairness of the game.[6, 12, 19] Fairness is a critical factor for enjoying a game and consequently cheating may drive away paying customers. However, we will not delve into that topic. Instead, we only want to point out that the level of cheat-resistance is determined by the implemented architecture, not by our framework. In the classic C/S setting, all trust is imposed on the server and our framework doesn't change this. A P2P node within the Replicated Simulation is responsible for the object it owns. However, all peers receive updates about changes of that object and they may check themselves if those changes conform to the rules of the game. Otherwise they may reject an update. In the Mutual Checking scenario, each RC votes for a certain update. The larger the group of RCs is, the less likely it is for cheating nodes to insert a falsified update.

The only thing the framework has to guarantee is that no one is able to forge messages. E.g., if a node receives an update, it must be sure that the sender is really the owner of that object. Nodes may simply identified by IP addresses or, if a higher level of security is necessary or object ownership must outlast network sessions, cryptographic signatures may be used. For this purpose a public key infrastructure is necessary which can be run by the game provider.

7. Conclusion

In this paper we have presented a framework that provides a game developer with a complete abstraction from network related issues. The framework can be divided into three layers: on the highest level the game layer, underneath the object layer and at the bottom the networking layer.

On the game layer, standard components, like the game engine and components managing audiovisual feedback and player input, are located. This is also where a game developer has to implement the rules and the logic of a specific game. All components on this layer communicate through an interface to the layer below, the object layer. Game developers can create, manipulate and delete all game objects as if they were local; network consistency as well as ownership management is handled automatically. The networking interface below hides network related issues behind a publish/subscribe abstraction. If it is necessary to optimize the network layer for different quality requirements, like higher scalability or lower latency, custom implementations can be used.

With network implementation details hidden, game developers can focus more on game design rather than writing specialized code. Implementation details like data-driven game objects further emphasize this approach.

References

[1] XBlast. `xblast-center.com`.

[2] P. Bettner and M. Terrano. 1500 archers on a 28.8: Network programming in age of empires and beyond. In *GDC Proceedings*, 2001.

[3] S. Bilas. A data-driven game object system. In *Proceedings of the Games Developer Conference*, 2002.

[4] J. Blow. Game development: Harder than you think. ACM Queue vol. 1, no. 10, February 2004.

[5] D. Church. Object systems: Methods for attaching data to objects and connecting behavior. In *Proceedings of the Game Developers Conference*, 2002.

[6] S. B. Davis. Why cheating matters - cheating, game security, and the future of global on-line gaming business. In *Proceedings of the 2003 Game Developers Conference*, March 2003.

[7] M. Doherty. A software architecture for games. Technical report, University of the Pacific Department of Computer Science, 2003.

[8] A. Duran. Building object-systems: Features, tradeoffs and pitfalls. In *Proceedings of the Game Developers Conference*, 2003.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[10] E. Games. Unreal 3 engine. `www.unrealtechnology.com/html/technology/ue30.shtml`, 2007.

[11] F. Glinka, A. Ploß, J. Müller-Iden, and S. Gorlatch. RTF: A real-time framework for developing scalable multiplayer online games. In *Proceedings of NetGames '07*, 2007.

[12] P. Kabus and A. P. Buchmann. Design of a Cheat-Resistant P2P Online Gaming System. In *Proceedings of DIMEA '07*, 2007.

[13] P. Kabus, W. W. Terpstra, M. Cilia, and A. P. Buchmann. Addressing cheating in distributed MMOGs. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–6, New York, NY, USA, 2005. ACM Press.

[14] Y. Kaneda, H. Takahashi, M. Saito, H. Aida, and H. Tokuda. A challenge for reusing multiplayer online games without modifying binaries. In *Proceedings of the 4th ACM Workshop on Network & System Support for Games*, 2005.

[15] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia*, 6(1):47–57, Feb. 2004.

[16] S. Microsystems. JSR-223 Scripting for the Java Platform, 2006.

[17] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen. The information bus: an architecture for extensible distributed systems. In *SOSP '93: Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68, New York, NY, USA, 1993. ACM Press.

[18] W. Palant, C. Griwodz, and P. Halvorsen. Evaluating dead reckoning variations with a multi-player game simulator. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 20–25, May 2006.

[19] M. Pritchard. How to hurt the hackers: The scoop on internet cheating and how you can combat it. *Gamasutra*, 2000.

[20] Sun Microsystems Inc. Java Message Service (JMS) Specification Version 1.1. http://java.sun.com/products/jms/docs.html, 2002.

MOBILE VIRTUAL WORLDS: A PROXIMITY BASED EVOLUTION

Stefano Cacciaguerra, Gabriele D'Angelo
Department of Computer Science
University of Bologna
Via Sacchi 3, Cesena (FC), Italy
E-mail: {scacciag, gdangelo}@cs.unibo.it

KEYWORDS
Virtual Worlds, Pervasive Entertainment, Agent-based Entertainment, Game Design.

ABSTRACT

The wireless revolution has enabled a new generation of applications for nomadic users. In this work we propose a new paradigm for the creation of games based on virtual worlds that are hosted on mobile devices. Each time mobile devices *"get in touch"*, their virtual worlds have the opportunity to interact. This form of interaction is based on the remote control of a subset of the agents that populates the virtual world. In accord with this, it is possible to create games with an unpredicted and unforeseeable evolution. Finally, we introduce PReDA, a prototypal implementation of the proposed mechanism that is based on the Netlogo environment.

INTRODUCTION

The pervasive diffusion of the wireless technology has lead to wide effects on the ICT field. First of all the wireless access to the Internet telephony and, then, the use of wireless mobile devices to browse the Web, anytime and anywhere. Thanks to more capable hardware, pervasive networks and adaptive software protocols, the wireless technology is fostering a new generation of applications, as an example: sensors' networks, wearable computers, ubiquitous and context aware applications (Chen at al., 2003; Kanter, 2003).

In this scenario, it is easy to predict a future where mobile users will daily use many forms of Internet access (e.g. wireless hotspots, private networks, ad hoc networks etc.), in order to share contents and to take advantage of the resources offered by the broadband connectivity. As an example, the participative virtual worlds are gaining more and more popularity: they allow the users to keep in touch with friends and colleagues, to collaborate in the resolution of shared tasks, to run brainstorming meetings and to share common resources. It is worth noting that this kind of social environments supports both forms of collaboration and competition between users, building a new kind of interactive and immersive metaworld. To some extent, using these technologies is possible to build virtual worlds that mimic many of the daily activities (Linden, 2007). In this case, the world is virtual and under some viewpoints it is safe: it represents a sort of sandbox.

In this field, an important role has been played by the participative simulation (Colella et al., 1998; Wilensky et al., 2007) that is a gaming activity often used to explore complex systems. As an example (Terna, 2003), a virtual marketplace where users can be sellers and customers, at the same time. As another example, the road transportation: we could imagine a system where each user is in charge of managing a specific traffic light, with the capability to trigger the "green" and "red" lights. In this case the users would be able to choose a strategy based on collaboration or competition. For the sake of simplicity and clearness, all these examples are immediate and obvious, but it is worth noting that participative simulation has many fields of application in both scientific and technical areas (e.g. diffusion of viruses, vehicles behavior, modeling of molecules in a membrane and the prisoners' dilemma).

The main part of a participative simulation is the underlying Multi-Agent System (MAS). It offers a programmable system to model and simulate complex behaviors. In a MAS, the simulation developer can define groups of agents, to each group can be assigned autonomous or shared tasks, and can be instructed to achieve a specific goal. The global state of a multi-agent simulation is obtained as the result of a large set of interactions among the agents. During the simulation lifespan each agent will be part of many local interactions, data exchanges, cooperation and competition activities. Each agent (during its artificial life) can be driven by a form of artificial intelligence or by a human being. In both cases, the agents will play following their capabilities, in example their past knowledge, the ability to explore the environment and the available activities.

In this scenario, the wireless technologies enable the participation of nomadic users, extending the access to MASs also to users with mobile devices. The first and more direct consequence of the wireless technology is that human users can remotely control agents that are within a MAS, anytime and anywhere. Furthermore, less immediate and more complex consequences can be foreseen. Despite of a human being, the player could be a remote MAS that controls one or more external agents. In this case the interaction would be between MASs. Using wireless connectivity, would it be possible to imagine that a MAS running on a mobile device would connect to another MAS, to take control of a part of its agents?

The main goal of this work is to demonstrate that this scenario is realistic, and to propose a new framework based on the Netlogo environment. Following this approach, different MASs implemented by Netlogo will be able to interact together, based on their proximity, and to affect the evolution of the whole system. In this case, we are not proposing a new game based on MASs, but a new paradigm for the creation of games based on mobile virtual worlds.

Following this approach, the evolution of the game is determined by new factors, as the proximity of gamers and the consequent random interactions between MASs. In our vision, this work is the first step in the direction of a new class of mobile games.

The next Section explains why it is important to give the possibility to remotely drive agents. The third Section shows similarities and differences with computer entertainment applications. The fourth illustrates the system architecture. The fifth suggests a case study based on Netlogo. Finally, we conclude this paper with some final remarks and future works.

REMOTE DRIVEN AGENTS

A MAS allows to represent, mimic and study complex systems where different components interact among them in a cooperative or competitive way. It promotes the understanding of a complex system by means of the description of its rules and the representation of its evolution. Modern MASs are based on 2D raster graphical functionalities (North et al., 2006; SWARM, 2007; Wilensky, 2007) that, in many cases, are inadequate for the human perception. Only lately, the introduction of recent 3D rendering engines (Cacciaguerra et al., 2004; Wilensky, 2007) has lead to higher expressivity and a better representativeness of the system (see Fig. 1). Expressivity becomes either the capacity to mimic, with a higher detail of accuracy, a complex system (if this is necessary for the modeling effort) or to show to the viewer (i.e. the player) another dimension in order to enhance his comprehension. Essentially, we are introducing a new dimension to our discussion, referring either to another physical dimension (i.e. geometric plane) or an improvement of the representation (i.e. expressive power). In accordance with these considerations, we believe that enabling a MAS to control the agents of another one, would permit to add a new dimension in the evolution of complex systems. The idea behind this approach is not related on the opportunity to decrease the computational load, distributing the agents on other computers (that is a well known approach in literature, e.g. Riley, 2003). It refers to the opportunity that two or more MASs get in touch and interact when are close, that is, under the wireless coverage area of one or more network adapters. This translates to a system that can evolve in a "less deterministic" (i.e. unpredictable) mode: that is because the interaction due to the *proximity* with other systems would be able to change their evolution. This kind of system will be by far more unpredictable than a system where all agents are driven by a single piece of local software (i.e. the standard approach). In the first case, the movements of mobile devices are the basis for unplanned meetings and the availability of a wireless network is the media that allows the interaction. In this scenario, the unplanned meetings add a new degree of indeterminism to the whole system. Furthermore, following this approach, the game modelers can define different behaviors of agents when reacting to the same perception, implementing different course of action. In any case, all the implementations are bounded by a set of game-related roles. For example, in the wolf-sheep predation model, the sheeps should adopt different strategies to eat the cabbage and to flee from the wolves. In any case, it is not acceptable that a sheep eats a wolf! In other words, this means that the freedom in the implementation of a specific behavior have always to be coherent with the specific role of the agent. Following this approach, it is possible to mix the behaviors of agents that have been implemented by different parties. This will allow to generate combined actions that can lead to results that are unpredicted and unforeseeable in the original system. In a causal meeting, the exchange of behaviors among systems that are hosted on mobile devices, can be described as the spread of a virus in an epidemiologic scenario. In this metaphor, the remote system can affect the behavior of many agents due to the *contact* (i.e. the proximity). A posteriori, if the resulting effect is seen as interesting then it would be possible to analyze the log files, in order to trace the interactions and inspect step-by-step the evolution of the system. In most cases, videogames can be considered as complex systems. Therefore, very often the participatory simulation is seen as a form of game-based e-learning. In this sense, we think that our approach could be a first step in the direction of a new paradigm for mobile gaming.

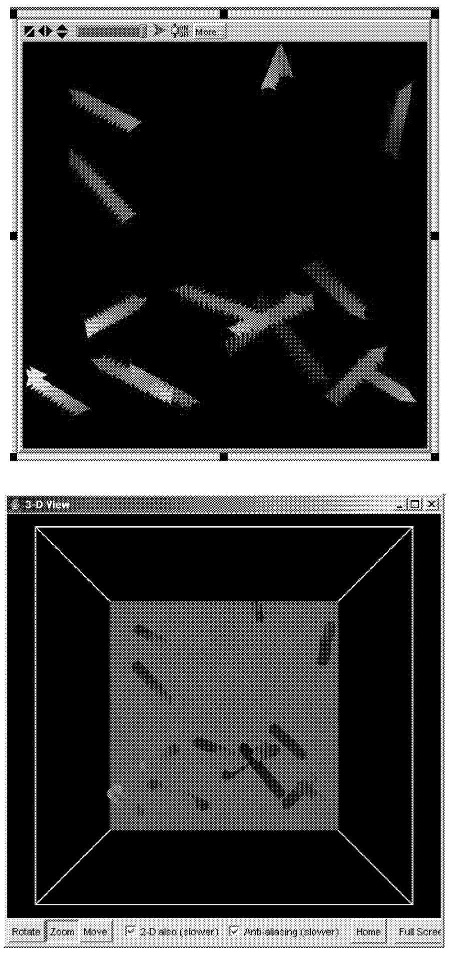


Fig. 1 Display of NetLogo bouncing balls model: 2D and 3D

RELATED WORK

In the state of art of computer entertainment there are some applications partially based on the proposed paradigm, such as: Ubiquitous Monster (Kawanishi, 2005), Insectopia

(Peitz et al., 2007), WSNMP (Liu et al., 2006) and Pirates! (Björk et al., 2001).

Ubiquitous Monster is a monster collection videogame where the players wander about the real world to collect monsters that will be used in the virtual world. In this case, the transfer of monsters is based on the RFID technology. Within the game, the behaviors and aptitudes of monsters are predefined, and they appear in the virtual world in relation with the geographical position of the player. The monsters, accommodated in the virtual world running on a mobile device, can: born, make friends, evolve, breed and die on the basis of the weather conditions (i.e. lightness, temperature, pressure, electric potential). All these conditions are detected by a sensors network in the area where the user is located. For example, given a monster that uses the light to obtain its vital energy, it is simpler to capture it in a sunny place instead of a dark zone. Further, the weather and light conditions change along the day: so, it is very difficult to find such kind of monster during the night! In the game, when two players meet, some monsters may migrate from one virtual world to another, in order to find more comfortable environments, and therefore to obtain as much energy as possible. This game promotes the movement of the players in the real world in order to collect different monsters and to exchange them, by means of the migration process. Since that the virtual world is an ecosystem with limited resources, it is not possible to capture a great number of monsters. Therefore, the aim of the game is to reach a sort of instable equilibrium in each local ecosystem, trying to support the higher possible number of monsters of different breeds.

Similarly, Insectopia is an insect collection video game running on mobile phones. Each player must collect and domesticate his insects. The lifespan is limited to a fixed amount of time. After a week, each insect dies and the player must capture a new one. The catch of a specific insect depends on the type of mobile devices that are in proximity of the player. This game adopts the Bluetooth technology in order to discover the different types of mobile devices.

Wireless Sensor Network based Mobile Pet game (WSNMP) is a game where the user must control domestic animal by means of a mobile device. In this case, the game is based on a wireless sensor network. The players can interact with virtual pets, feeding them, taking care of them and playing with them. And furthermore, they can share their pets with others, trade them, watch them compete against each other, become friends, create offspring and develop a virtual pet society. The players can also communicate each other through their shared pets. Each virtual pet is represented by a sensor node. The sensor nodes are composed by many sensors; each sensor is an organ of perception, such as, light detector, smoke detector and microphone for eye, nose, and ear, respectively.

Pirates! is a videogame where the player takes the role of a captain pirate sailing his ship in a fantasy archipelago. The ship permits to transport commodities from the different islands in order to be sold at markets. Each ship has a crew and is equipped with cannons. If the captain successfully completes the missions, then he can sturdier the ship thanks to the gained rewards. There are some dangers such as sinking in a battle, meeting cannibals or getting lost during the exploration of an island. The aim is to find treasures and commodities in each visited island. Islands are different and provide many kinds of merchandise and dangers. Further, at the free harbor it is possible to recruit new crew members, to repair a ship, to trade for goods and to obtain a new mission. Each ship is represented by a PDA equipped by an IEEE 802.11 WLAN card and a RF proximity sensor, while the islands are physical locations in the real world (e.g. different rooms in a building).

The similarities of these applications with our proposal are: i) the dynamism in the evolution of the ecosystems, and ii) the migration of agents among mobile devices. The above introduced applications use sensor networks to detect the ambient conditions and to get the geographical position. Differently in our approach, the possible evolutions of the virtual world are, *a priori*, less predictable. In fact in the other approach, the behavior of monsters, insects, pets or pirates does not change during the lifespan of the game: all of them are defined and implemented by the application developer and can not be changed at runtime.

To the contrary, in our approach, each player can modify the behavior of its agents (e.g. implementing new actions) in each moment, also if the game has already started. The only imposed limitation is to respect the general rules of the virtual world that, above, we have called roles. In this way, it will be impossible to define a priori the evolution of the simulated ecosystem. A posteriori, it will be very important to study the evolution of the system, inspecting the different phases of the evolution and taking care to study emergent patterns. This analysis will be possible using the log files that trace the evolution of the whole system.

SYSTEM ARCHITECTURE

In our vision, we have a set of virtual worlds, each one runs on a different mobile device. The virtual worlds should be able to interact together depending on their proximity. The position of a virtual world is due to the mobile devices that hosts it. Each virtual world is composed by a set of agents living in a defined environment. When a virtual world gets in touch with another one, it can take control of a sub-set of the agents in the other one.

In accord with this vision, we implemented the Proximate Remotely Driven Agents (PReDA) framework. PReDA is a prototypal communication framework, based on the proximity of mobile devices, that is in charge of: i) discovering devices that host a PReDA virtual world; ii) managing the communication among PReDA virtual worlds and iii) enabling the remote control of agents.

Given such requirements, the proposed architecture is based on ultra-portable notebooks, tablet PCs and PDAs, Java-enabled and with Bluetooth connectivity. The discovery phase of PReDA takes advantage of the Bluetooth discovery mechanism. Each device continuously searches other devices within its coverage area. Each time a new device is detected, an inquiry scan is performed to obtain the list of available services (i.e. a virtual world based on the PReDA framework). If the new device is running PReDA, then it is possible to start a direct communication between the local and the remote virtual worlds. The communication is implemented using the Bluetooth Logical Link Control and Adaptation Protocol (L2CAP). PReDA uses the L2CAP protocol to pair the local and the remote virtual worlds. Each instance of PReDA verifies if there are remotely controllable agents that are flagged as available. An agent is available if no other PReDA systems are now controlling it. If at least one available agent is found then the local virtual world will send a set of commands to a subset of them. Due to the nomadic nature of the hosting devices, only a limited amount of time (in order of a few seconds) would be available for the interactions among virtual worlds. Therefore, in a short time frame, it is possible to transfer a single command or a complex behavior (in form of a set of actions). In the following section we will introduce a prototypal implementation of PReDA based on Netlogo.

A CASE STUDY WITH NETLOGO

NetLogo (Wilensky, 2007) is a programmable tool that allows to simulate the evolution of complex systems. This tool permits to the modeler to give instructions to a high number of independent agents all operating concurrently, either in a cooperative way or in a competitive one. Therefore, it promotes the exploration of the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interaction of these individuals. Further, the users can "open simulations" (i.e. explore the internal state), can play with them, in order to explore their evolution under various conditions and can create their own models (i.e. implementing new functionalities). This tool is simple enough that any user can easily run simulations or even implement the behavior of his agents. The possibility to see the code of other models and to access an elevated number of high-level primitives promotes the reuse of the code, allowing everyone to implement its own routines. The simple approach, that can be used to program the tool, does not reduce the expressive power of the models that can be simulated, making it an interesting tool for many research fields (i.e. the simulation of many natural and social phenomena). Moreover, the Netlogo community is very active and has made a large number of models freely available, models that are related to many fields as: biology and medicine, physics and chemistry, mathematics and computer science, economics and social psychology. This wide adoption demonstrates that Netlogo is very easy to use and that can cope with many different topics and problems. One of the most interesting features (introduced in version 2.0) is the "extensions" module, it allows the developers to introduce new commands and reporters (Wilensky, 2007) that can be used inside the Netlogo environment. The idea behind this module is to extend the primitives by means of Java code, that is archived in a .jar file. In this way, it is possible to write high-level functions but also to integrate the Netlogo environment within other projects! In accord with this consideration, we have integrated the Netlogo environment within a new framework. The goal of this new framework is to support communication among many Netlogo environments that are executed on different devices. The framework exploits the Netlogo extensions module to obtain this result. The framework has to provide two main functionalities: i) it must supply an access point to each local instance of the environment by means of an agent discovery system; ii) it must support the exchange of commands among different mobile devices. The communication between different environments needs a protocol that permits to exchange commands (as string of characters) from different worlds (e.g. Netlogo instances). Given the "A world" and the "B world", that are different environments accommodated on two mobile devices, our mechanism provides a form of addressing (i.e. to make the environments reachable) and a communication protocol.

In detail, Netlogo classifies agents into two types: passive and active. The virtual world is divided in square pieces of ground, each piece is called "patch". Netlogo classifies the patches as passive agents. These agents can be affected only by active ones and by the Observer (that is the Demiurge of the world). The active agents (called turtles) can interact among them and with the patches.

In the following, we report some details about the implementation of our framework. Firstly, we report a piece of code from a Netlogo model. It is worth noting that the example contains an include of the *PReDAextension.jar* archive. This archive provides the basic functionalities for communication and discovery. In this way, the developer can directly use its own Java routines inside Netlogo. Each Netlogo model begins with the pressure of a button that starts the *setup* of the ecosystem, initializing the agents and the environment variables. In particular, in the code example reported below, after the initial setup, the setup bootstraps the discovery system specifying which agents can be remotely controlled. In detail, the *rmt-crt-turtles* and *rmt-crt-patches* routines report the set of turtles and patches that will be remotely controllable. By means of the definition of these routines, the user can determine which agents can be remotely controllable while leaving untouched the others. The *startCommFrmwrk* initializes the communication framework. The *GO* button runs the body of a Netlogo model. The *ask* construct is used to specify commands that are to be run by a set of agents. The *run* routine allows an agent to interpret the given string as a sequence of one or more NetLogo commands and runs them. The routine *recvmsg* that has been declared inside *PReDAextension.jar* receives a message, that is a string sent by a remote Netlogo environment accommodated on a mobile device. Obviously, this means that the remote Netlogo environment will use the *sendmsg* routing (also in this case defined in *PReDAextension.jar*) to send messages, that are strings dispatched to one or more remotely controlled agents.

In last part of the code, it is possible to analyze the approach used in the implementation of our framework. In the first case (see Example 1), the received string will contain the number of steps that a set of turtles must cover. In the second case (see Example 2) is defined the new color of a set of patches. For example, if the string returned by *recvmsg* is "3", then all turtles will be moved straight of 3 steps in their direction, while, if it is "yellow" then all patches will become yellow colored.

It is worth noting that in the third case (see Example 3 and Figure 2), the string directly reports a sequence of commands, respectively: i) "*ask turtles with color = red [fd 3]*" - all red turtles will be moved straight on their direction of 3 steps, while, ii) "*ask patches with pcolor = yellow [set energy 0]*" - the energy level of all yellow patches will be decreased to zero.

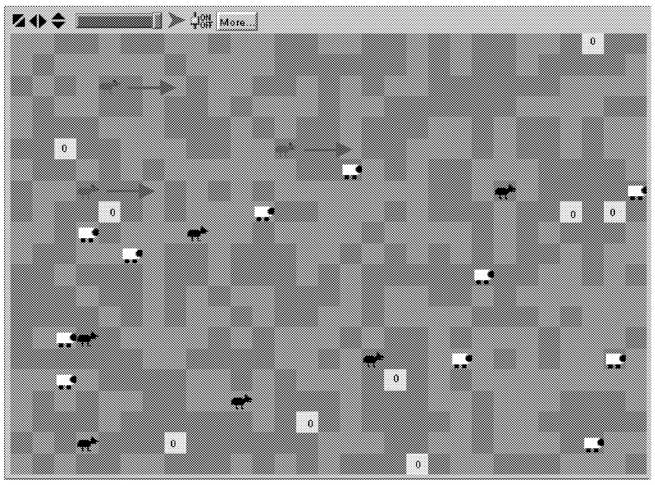


Fig. 2 NetLogo Wolf – Sheep Predation

```
__extensions["PReDAextension.jar"]

to setup
  …
  setup-turtles
  setup-patches
  set clock 0
  …
  Discovery(rmt-ctr-turtles, rmt-ctr-patches)
  startCommFrmwrk
end

to GO
  …
  ask turtles
  [
    …
       run fd recvmsg  ;; (Example 1)
    …
  ]
  ask patches
  [
    …
       run set pcolor recvmsg  ;; (Example 2)
    …
  ]
  …
  run recvmsg  ;; (Example 3)
  …
  set clock clock + 1
end
```

CONCLUSIONS AND FUTURE WORK

In this work, we have introduced the Proximate Remotely Driven Agents (PReDA) framework. PReDA is a prototypal framework based on the Netlogo environment and its "Extensions" module. PReDA exploits the Bluetooth connectivity in order to discover other PReDA copies running on top of mobile devices. Each mobile device runs a local virtual world composed by an environment and a set of agents. In this way, the PReDA-based virtual worlds running in each mobile device can interact together, taking control of a part of the remote agents. Since each player can easy modify the behavior of agents and since the game is subject to random interactions, the evolution of the game will result very unpredictable.

As a future work, we aim to build a real game based on the proposed paradigm. Furthermore, we plan to develop a light version of the PReDA framework that will run on Java-enabled mobile phones.

Another direction of this research will involve the study of the behavior of nomadic users and the impact of the proposed paradigm on social sciences.

REFERENCES

Björk S., Falk J., Hansson R., Ljungstrand P., 2001 Pirates! - Using the Physical World as a Game Board. *in proc. of Human-Computer Interaction conference* (July), Tokyo, Japan.

Cacciaguerra S., Mirri S., Pracucci M., Salomoni P., 2006. "Wandering about the City, Multi-Playing a Game" *in proc. of IEEE International Workshop on NIME* (January), Las Vegas (NV-USA).

Cacciaguerra S., Roccetti M., Roffilli M. Lomi, A, 2004. "Wireless Software Architecture for Fast 3D Rendering of Agent-Based Multimedia Simulations on Portable Devices" *in Proc. of the First Consumer Communications and Networking Conference (CCNC)*, IEEE Communications Society (January), Las Vegas (NV-USA)..

Chen H., Finin T., 2003 "An Ontology for a Context Aware Pervasive Computing Environment", *in Proc. of IJCAI Workshop on Ontologies and Distributed Systems.*

Colella V., Borovoy R., Resnick M., 1998 "Participatory Simulations: Using Computational Objects to Learn about Dynamic Systems" *in Proc. of Computer Human Interface Conference*, (April) Los Angeles (USA - CA).

Kanter T. G., 2003 "Attaching Context-Aware Services to Moving Locations" *in IEEE Internet Computing Magazine*,(March-April) Vol. 7, N. 2.

Kawanishi N, Kawahara Y., Morikawa H., Aoyama T., 2005 "Prototyping a Real-World-Oriented Monster-Collection Game" *in Proc. of 5th International Workshop on Smart Appliances and Wearable Computing*, (June) Columbus,(USA - OH).

Linden Lab 2007, "Secon Life", http://www.secondlife.com/.

Liu L., e Ma, H., 2006 "Wireless Sensor Network Based Mobile Pet Game", *In Proc. of NetGames*, (October), Singapore.

North M.J., Collier N.T., Vos J.R., 2006 "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit", i*n ACM Transactions on Modeling and Computer Simulation*, Vol. 16, Issue 1, pp. 1-25, (Jannuary), New York (USA - NW).

Peitz J., Saarenpää H., Björk S., 2007 "Insectopia - Exploring Pervasive Games through Technology already Pervasively Available" *in Proc. of Advanced in Computer Entertainement Technology*, (June) Salisburg (Austria).

Riley P., 2003 "SPADES: System for Parallel Agent Discrete Event Simulation", in AI Magazine.

Terna P., 2003 "Decision making and enterprise simulation with jES and Swarm.", *in Proc. of the Seventh Annual Swarm Users/Researchers Conference (April)*, Notre Dame, Indiana (USA - IN).

Swarm Development Group. 2007 "Swarm" http://www.swarm.org Swarm Development Group, Santa Fe (USA – NM).

Wilensky U., 2007 "NetLogo" http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky U., Stroup W., 2007 "HubNet" http://ccl.northwestern.edu/netlogo/hubnet.html. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

JOURNEY FROM THE MAGIC CIRCLE TO THE THERAPEUTIC GAMEPLAY EXPERIENCE

Balázs Józsa
Institute of Psychology,
University of Debrecen
4032 Egyetem tér 1. Debrecen, Hungary
E-mail: jozsabalazs@yahoo.co.uk

KEYWORDS
player's experience, presence, avatar, methodology

ABSTRACT

This paper combines several theories which try to define player's experience. The three component player's experience model is unique in that way, that it gives a complex and full explanation how player's experiences come into exist. As avatar makes interaction with game elements, game events arise and its interpretation has affect on player's emotion and opinion creating an experience. This experience will influence the avatar's behaviour. Series of player's experiences merged into gameplay experience, which can be analysed by content analysis and Experiential Analysis Technique.

INTRODUCTION

Researching player's experience is quite a new field of game researching, but its root date back to the past. In this paper the question I am trying to answer is: What are the components of player's experience, how they establish and how can we examine them?

Previous theories

According to Huizinga (1955) games are one of the building stone of our society. If we observed a playing animal or a human, we would find that their behaviour is very similar to their everyday activity. So player's game experience - in my opinion - could be very similar to player's everyday experiences. Whereas Huizinga said, games have their own space, time and rules, they are not in connection with everyday life. This is what game research called magic circle. Later, number of researchers attacked this opinion (e.g. Callois 1961) but they also did not mention what kind of experiences player could go through during a game session.
If the magic circle was traversable, games would have long term effect on players, so it would be worthy to plan player's experience in advance. There are several theories trying to uncover components, which can influence player's experiences. One type of theories underlines **immersion or presence**. Ermi and Mäyrä (2005) set up a model called SCI to explain how player's experiences come into exist. It has three main components: Sensory Immersion, Challenge-based immersion and Imaginative immersion. Later, Arsenault (2005) modified this model, and used Systemic immersion instead of Challenge-based immersion and Fictional immersion instead of Imaginative immersion (SSF-model). Despite its complexity the model distinguishes only three types of experience: sensorial-, systemic- and fictional type.
Other theories' main concept is **gameflow** as game researchers use it. Flow is a state of consciousness that is sometimes experienced by people who are deeply involved in an enjoyable activity (Pace, 2004). It has 8 components which existence is necessary establishing flow (Csikszentmihalyi 1990). Later Novak at al. (2000, in. Fernandez 2007) criticized GameFlow theory that some of its base components just consequences of flow and have no any role in establishing it. GameFlow model tells how flow could come into exist - which is a complex experience-pack. So this model is not useful to distinguish different type of experience.
Other theories emphasize player's **motivation** and its role in establishing experience. For example Mitchell's Situational Interest model (1993) separates catch and hold triggering conditions. He said computer games are catch conditions, which can capture player's attention but cannot hold it. I am not agree with Mitchell, I think computer games could provide both catch and hold experiences - that is why some players replay their favourite game. These are two different types of experiences.
Fernandez (2007) tried to explain emotional aspect of player's experience. She create the **Game Experience Model** to describe elements of digital game experiences especially elements that determinate fun experience. According to this model, as player playing with a computer game it is elicit a cognitive and emotional response from him/her and these two components are responsible for both fun experiences and the evaluation of the game.

Bobko et al. (1984) asked people to compare ten computer games pairwise how similar are they. **Multidimensional scaling** showed 3 underlying dimensions: destructiveness, dimensionality and graphic quality. However it is a quite acceptable model it interpreted only 30% of the result.

PLAYER'S EXPERIENCE

According to Fernandez (2007) fun is the main result of player's experience. I think fun is just one type of player's experience. One of my interviewee said about a computer game: "*it is possible to learn a lot from games*". I think players have not got main experience, they have different kind of experiences (e.g. fun, growing knowledge, aesthetic experience...etc.).

If we try to understand player's experience, we have to make a model how experiences come into exist. At a game situation there are two main components: the game and the player. Computer is the transmitter medium between game and player. Games have elements and interface, players have opinions and emotions. This is a very simple model and every element in it can be taken to further pieces. *Every game has an avatar* (a special game element mostly controlled by the player): hero or heroine, a car driver, a mayor...etc. Even Tetris has one: the force with which I can manipulate tetris elements. So *avatar is the player's representation in the game (reality).* Avatar is a game element with which a player can identify his/herself. Figure 1. shows components of player's experience.

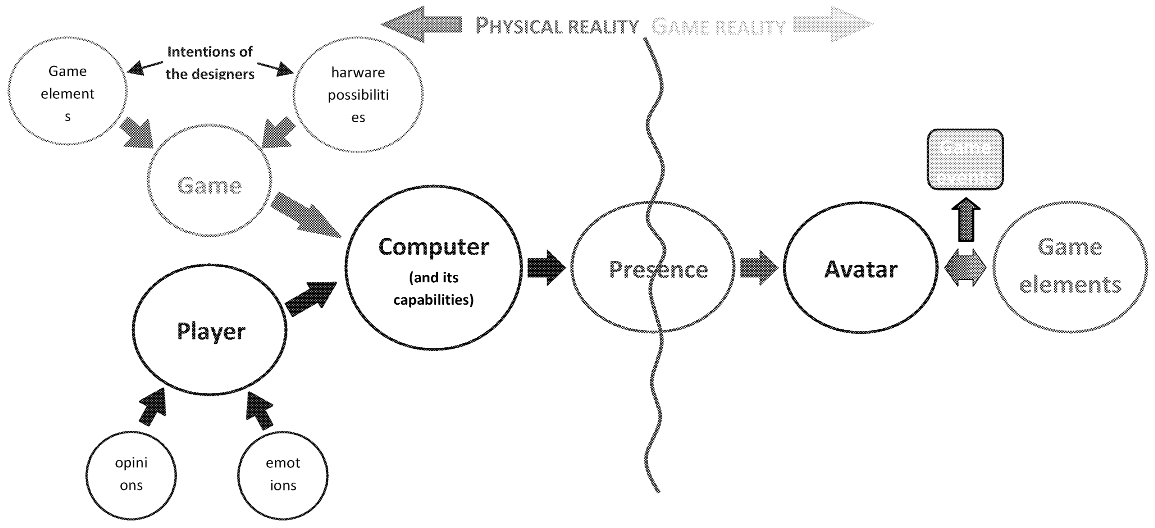


Figure 1: Components of player's experience

Process of player's experience

There are two different realities: avatar is in the game reality (virtual reality) while player is in the physical reality. In this model, *presence is a function which makes possible that player indentifies him/herself with the avatar* and that time the two realities become one. According to this model, player's experience can establish in that way: Player starts to play, then with the help of presence, (s)he becomes one with the avatar. Then avatar makes different kind of connections with game element, thus game event come into exist. Finally, this game event takes effect on player and creates an experience. So *player's experience is a subjective interpretation of a game event(s) in the player's mind.*

Levels of Player's Experience

The definition of player's experience has two components: game events and its subjective interpretation in the player. Subjective interpretation means game events can change player's opinion and emotions about what happens. After player interprets a game event (s)he behaves (with his/her avatar) according to it (in game reality). This model is similar to the phenomenon of attitude, which is known from social psychology. We can say player's experience is a special attitude which can change from game event to game event. Attitude has 3 components: cognitive (intellectual), affective (emotional) and conative (behaviour). In case of computer games, behaviour level is in the game reality, while the other two are in the physical reality. Figure 2 shows the
arrangement of levels and theirs connection with reality.

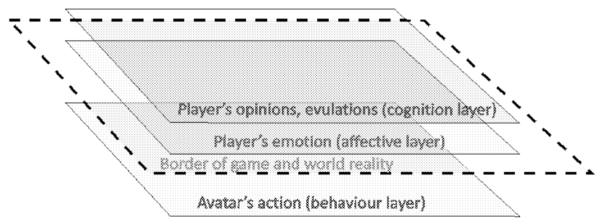

Player's opinions, evulations (cognition layer)
Player's emotion (affective layer)
Border of game and world reality
Avatar's action (behaviour layer)

Figure 2: Three Levels of player's experience and their Connection with reality

Cognitive level means what are *my thoughts about the game, about my avatar*...etc. *Affective level* means *my emotions about game events, about my avatar and its action*…etc. *Conative level connects to avatar because avatar "does things"* (walk, steer…etc.). Let's see an example: I meet with a dragon *(avatar makes connection with a game element)*. I think I am strong enough *(opinion)* to defeat that dragon and I feel myself in safe *(emotion)*. I attack *(game event)* the dragon. But dragon wound me a lot *(game's reaction to my behaviour)*. I think I am not strong enough *(modified opinion)* to defeat that dragon and I am starting to fear *(modified emotion)*. Finally I run away *(my/avatar behaviour)*. Fernandez (2007) also stressed, that players have cognitive and emotional responses but she did not define the way how they relate back on gameplay. In my model player's cognitive and emotional responses manifesting in the behaviour of the avatar. Figure 3. shows how components of player's experience connect to each other.

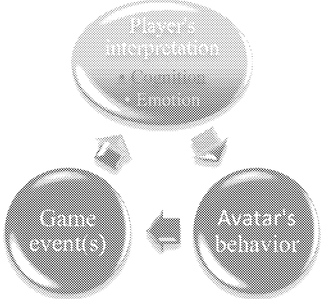


Figure 3: Relations of the three levels of player's experience

Gameplay Experience

The combination of the players' personality and game events results lots of different player's experience. *Gameplay experience is a series (or combinations) of player's experiences during a gameplay* - like an exciting race in Need for Speed. This experience consist series (or combinations) of game events *(overtakings, drifts)*.
If I ask you to tell your opinion about a concrete game, you will not speak about your gameplay experience. *Gameplay experience consists cognitive, emotional and behavioural level at the same time.* If you mention some remarkable moments from the game, that will be a gameplay experience. It is possible that every game, game genres or even players have their own map of gameplay experience.

DISCUSSION– METHODS FOR EXAMINING GAMEPLAY EXPERIENCES

Researching player's experience is very difficult task, because if a researcher wants to know what players experiencing during a game session, (s)he keeps player from experiencing anything. *Content analysis* is good for research this issue by ask players about their remarkable memories. With this method gameplay experiences can take to pieces. These will be player's experiences. After coding them we have to identify its cognitive, affective and conative level and theirs connection with game elements. *EAT* (Experiential Analysis Technique, Sheehan et al. 1978) can be useful in it. Using EAT, subjects are asked to play with a computer game. A camera records game events and players' behaviour at the same time. After it a questioner asks subjects to play the recorded video, comment it, stop it whenever they want and explain what is happening at that moment. Theirs explanations will be analysed the same way as has been mentioned at content analysis.
Although this model is preliminary yet, If we connected game events to player's experiences, some day we would be able to create (or integrate) healing/therapeutic experiences into a computer game. Finally, we will be able not just play with games, but use them in favour of a good matter.

BIBLIOGRAPHY

Arsenault, D. 2005. "Dark Waters: Spotlight on immersion." In *Proceedings of Game-On North America 2005 Conference.* http://www.le-ludophile.com/Files/Arsenault%20-%20Dark%20Waters.pdf, 2007. 11. 03.

Bobko, D., Bobko D.J. and Davis M.A. 1984. "Multidimensional scaling of video games." In *Human Factors* 26, No. 4, 477-482.

Callois, R. 1961. *Man, play and games* (M. Barash, Trans.). Free Press of Glencoe, New York

Csikszentmihalyi, M. 1990. Flow: The Psychology of Optimal Experience. HarperPerennial, New York

Ermi, L. and F. Mäyrä. 2005. "Fundamental Components of the Gameplay Experience: Analysing Immersion." In *Proceedings of DiGRA 2005 Conference: Changing Views – Worlds in Play.* http://www.gamesconference.org/digra2005/viewabstract.php?id=267, 2007.09.20.

Fernandez, A. 2007. "Fun Experience with Digital Games: a Model Proposition." In *Proceedings of Interact 2007 Workshop.* http://www.fun-of-use.org/interact2007/papers/FunExperienceWithDigitalGames.pdf, 2008.11.03.

Huizinga, J. (1955). *Homo ludens: A study of the play element in culture*. Beacon Press, Boston

Mitchell, M. 1993. "Situational interest: Its multifaceted structure in the secondary school mathematics classroom," In *Journal of Educational Psychology* 85, No. 3, 424-436.

Pace, S. 2004. "A grounded theory of the flow experiences of Web users." In *International Journal of Human-Computer Studies* 60, 327-363.

Sheehan, P.W., McConkey, K.M. and Cross, D. 1978. "Experiential analysis of hypnosis: Some new observations on hypnotic phenomena." In *Journal of abnormal psychology*, No. 87, 570-573.

BIOGRAPHY

BALÁZS JÓZSA was born in Debrecen, Hungary. Hungary and went to the University of Debrecen. First he studied as a programmer mathematician, then he was admitted to psychology and he obtained his degree in 2005. He has made two researches in relation with computer games: one was about absorption effects on computer game playing; second one (his final thesis) was about computer game playing situation and its relation with hypnosis and meditation. Now he is a PhD student at the same university. His research area is gameplay experiences, identifying player's experience and map of digital game experiences.

GAME AI

DATA ANALYSIS FOR GHOST AI CREATION IN COMMERCIAL FIGHTING GAMES

Worapoj Thunputtarakul and Vishnu Kotrajaras
Department of Computer Engineering
Chulalongkorn University Bangkok Thailand
worapoj.t@student.chula.ac.th, vishnu@cp.eng.chula.ac.th

KEYWORDS: Ghost AI, Fighting Game, Case base

ABSTRACT

In this paper we present a simple, rapid and efficient method for creating a ghost AI, an Artificial Intelligence that can imitate playing styles of players in fighting games. The created ghost AI can perform combination actions and make a decision about any movement in a similar fashion to a player it is copying. We scan a player's battle data, and then create situation-action pair cases for its corresponding ghost AI to use in actual battles. A ghost AI can be created and run swiftly, using small amounts of memory, making it suitable for console games. Our method is general enough to be used in most 2D and 3D fighting games. We carried out our experiment on Street Fighter Zero 3, one of the most well crafted fighting games, using AI-TEM testbed engine.

1. INTRODUCTION

1.1 Ghost AI

In fighting games there have been various attempts at ghost AIs (AIs that imitate players). Virtua Fighter 4 allowed players to train computer AIs to fight like them. Such ghosts could then be assigned to fight another player. However, feedback from players was not good at the time the game was released because it was hard to train their ghosts case by case. But in recent years, a ghost AI system has been used once more, in Tekken5: Dark resurrection. This time many things have been changed. Players do not need to train their ghosts in a training mode. They just play the game normally and the system will mechanically create their ghosts. This method makes fighting games more interesting because there will be many fighting styles for computer controlled opponents. Despite the fact that the ghost AI system is being acknowledged as the definitive AI for fighting games, the method for ghost AI creation remains undisclosed. In this paper, we propose a method for ghost AI creation using data obtained from game memory. Our method can be used in most fighting games. It also requires very small amounts of memory and therefore is suitable for console games.

1.2 Street Fighter Zero3 (SFZ3)

Street Fighter Zero3 is regarded as one of the best fighting games of all times. In a fighting game, a player must select one character from many characters, and fight one by one with an opponent character (another player or computer AI). A character can perform normal action such as move, crouch, jump, guard, punch or kick. There are also special attacks, such as firing bullets or executing a powerful flying punch. These special actions can be performed when a player presses a correct sequence of commands at the right time. A player must choose to perform actions in various situations based on the status of his character and opponent character. Getting into action with SFZ3 requires only a few minutes of tutorial. Nevertheless, the game has many ways to play a single character. For that reason, we have chosen SFZ3 as our game for experimenting with the ghost AI.

1.3 Testbed Environment

For the reliability of experimental results, game researchers may want to test their AI on real commercial game environments (Graepel et al 2004). But such environments are scarcely available. Results obtained from a researcher created game may not be convincing enough to warrant an actual use of discovered techniques in genuine games. Some researchers used mod of a commercial game (Spronck et al 2004), or a clone game (Ponsen et al 2005). Some developed test games on their own (Kendall and Kristian 2004) or used a testbed (Bailey and Katchabaw 2005). But none of those methods fit our experimental goal. (Thunputtarakul and Kotrajaras 2006) proposed a system to test AI modules in real commercial games without using any source code. They implemented a testbed from VisualboyAdvance (VBA), a Nintendo GameboyAdvance emulator. The testbed was called AI-TEM. An overview of AI-TEM is presented in figure 1 and its workflow diagram is presented in figure 2. By accessing the memory pool of the emulator, AI-TEM users are able to know states of the game at any particular moment. For fighting games, a state can consist of characters' positions, current animation frames, health points, etc. Users can insert their AI modules, in the form of C/C++ code or python script, into the testbed to control the game characters by providing controller signals. Our work uses AI-TEM as its testbed.

2. OUR APPROACH FOR CREATING GHOST AI

The main concept of our ghost AI creation is case based AI construction. We extracted a player character's reaction in various situations from battle log data created while playing, then produced situation-action pairs for the ghost of that character. Our experiment was made using SFZ3 training mode with character Ryu versus Ryu. AI-TEM was modified to suit our experiment. The ghost AI creation processes are displayed in figure 3. The following subsections describe each component in the process.

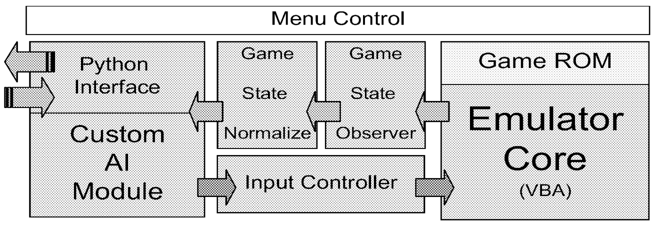


Figure 1: AI-TEM Testbed System Overview. The Light Blue Modules are VBA Original Modules.

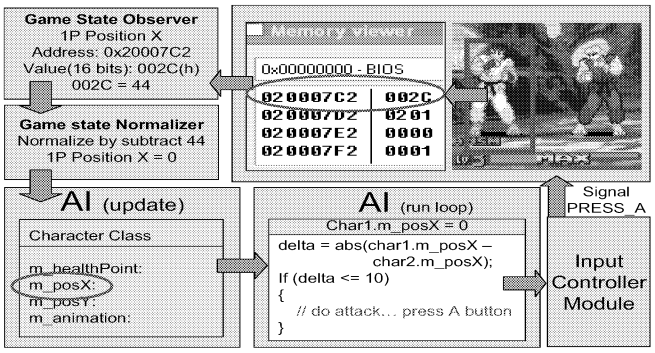


Figure 2: Workflow Diagram of AI-TEM System in SFZ3.

2.1 Obtaining Player Battle Log Data

First, while a player is playing, game states data need to be dumped from memory onto a battle log file. The data are used to identify each case in the case based AI system. The data consist of characters animation, characters positions in x and y axes, characters health points, characters bullet positions in x axes, damage that characters obtain in that frame, player character's facing direction and the corner status of characters. Recorded battle log data is in the following form:

```
Frame Data no: 00001
P1:Ani=002,X=120,Y=40,bullet=0,damage=0,HP=90
P2:Ani=002,X=240,Y=40,bullet=0,damage=0,HP=90
  :
Frame Data no: 00720
P1:Ani=016,X=150,Y=40,bullet=0,damage=0,HP=30
P2:Ani=030,X=560,Y=40,bullet=0,damage=5,HP=20
```

These criteria can change depending on game or user. Creating the ghost AI while the game is running without creating the battle log file is possible if complete information about the game mechanic is known (such as short or shared animation frame, that will be described in section 2.3). For SFZ3 on AI-TEM, we did not have such information. Therefore we had to use the log file.

2.2 Animation Set Database

An animation set database is used for identifying whether a character animation frame belongs to an animation set. An example is illustrated in Figure 4. Ryu animation frame number 0 to 6 belong to animation set ID 0, which represents Ryu's standing animation, while frame number 707 to 713 belong to Ryu's medium punch action, set ID 15. Together with the battle log file, the animation sets are used to create situation-action pair cases. In our experiment, we manually defined this database. There are totally 912 frames for character Ryu. This seems daunting. However, it is relatively easy for a game company to do because any game development team usually has access to animation data.

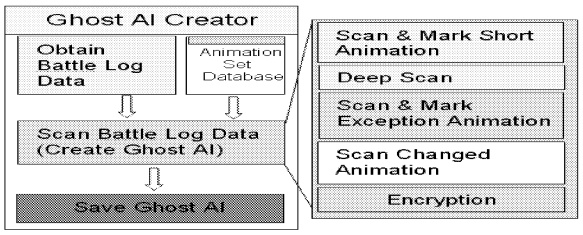


Figure 3: Ghost AI Creation Processes.

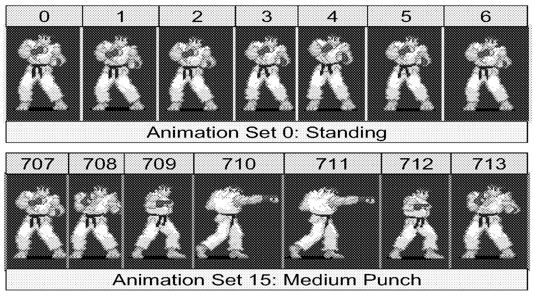


Figure 4: Example of Animation Set Database.

2.3 Scanning Battle Log Data

This process scans through every frame of a player's battle log data, trying to find which situation the player decided to begin his new animation set. For example, in situation *A* player1 is standing on the ground at position x=120 and player2 approaches player1 by jumping in the air at position x=150, both characters have full health bars and no bullets. Player1 decides to perform the special anti-air attack called Shoryuken punch. In short, the following situation-action pair will eventually be created:

```
if (Situation == A) do SHORYUKEN;
```

Now we look at this process in more detail. The process contains the following subtasks:

2.3.1 Finding Short Animation

Short animation means any animation that occurs for a very short period of time. It takes place mostly when a character is changing over from any standing animation loop to crouching animation loop. See an example animation time frame in figure 5. In figure 5, our character is standing then intends to do a crouching kick, but the crouching kick is not performed immediately. Before the crouching kick is carried out, a short period of moving forward and crouching animation is performed. This can happen due to the player not inputting the right command. For a crouching kick to be performed correctly without any prefix animation, the player needs to press down and kick at the same time on his control pad. In figure 5, the player presses down before kick and also unintentionally presses forward at the same time as down. Therefore extra animation is triggered. Nevertheless, the crouching kick is eventually performed and the prefix animation is so fast a human eye cannot see. We cannot avoid such minor mistakes made by players.

In our ghost AI model, detected animation frames tell us about a player's intention. Therefore, having the short animation taking place before the intended animation can misinform us. We must either identify a player's intention from the overall animation or get rid of the short animation before processing. In our experiment, we chose to do the latter.

All battle log data need to be scanned to find which animation set appears unusually brief, then that set is marked. Marked animation will not be considered when creating the AI. For a set of animation to be considered short, it depends on the set. In our experiment with SFZ3, short animation was no longer than 6 frames for most of the animation sets. The only exception was the crouching animation, of which short animation was no longer than 14 frames because changing from standing to crouching already took 8 frames.

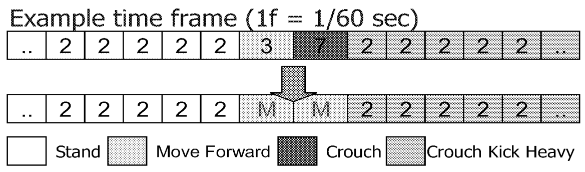


Figure 5: Short Animation Marking.

2.3.2 Deep Scanning

Some animation frames are shared between many animation sets. In such case, scanning ahead becomes necessary in order to identify the correct animation set. For example, jump straight, jump forward and jump backward begin with the same animation frames at the beginning. With the first frame obtained, we can only conclude that the character is doing an anonymous jump. With further scanning, we then know which jump the player intends to do and can go back to change from an anonymous jump to a specific jump. This step can be omitted if the controller signal can be completely analyzed. But this is not always the case.

2.3.3 Exception Animation Sets

Some animation sets should be omitted from our case base because they do not take place under players' control. Obvious examples are various damage animation sets. They occur as the results of opponent attacks. This type of animation that appears in the battle log data will be marked here.

2.3.4 Scanning Changed Animation

This step is the core of our ghost AI creation. After matching all animation frames to their corresponding animation sets and marking useless animation, it is time to scan the battle log data once more to find the situation that causes the player character to change its animation. Such situation and the changed animation set that it causes will be paired to create a situation-action case.

An example is shown in figure 6, where a player executes a crouching heavy kick. In 7th-8th frame, our character changes its animation set from standing to moving forward. But moving forward lasts only 2 frames so it is a short animation. It is marked useless and the next animation to consider will be crouching. However, this crouching is also a short animation and therefore marked useless (a proper crouching must last 14 frames or more). As a result, the next animation (crouching heavy kick) will be taken into account. The crouching heavy kick does not fit the useless animation category, so it is regarded as the changed animation set. Therefore the (situation at 7th frame, crouching heavy kick) is added to the case based AI.

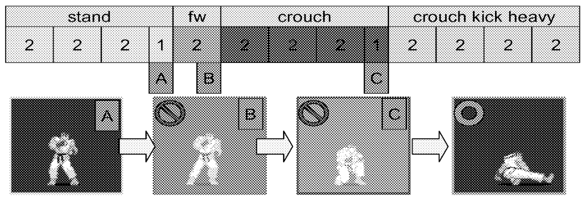


Figure 6: Scan Animation Change.

2.3.5 Situation Encryption

If the game needs to compare ten or more criteria (animation, position, bullet, etc.) to judge whether the current situation in the game is the same as any existing condition in our situation-action database, it will be a waste of processing power. Any game situation should be defined in simple form for easy comparison and discovery. We propose a method to encrypt a fighting game state situation into a 32-bit integer (capable of holding 4,294,967,296 values). The bits can be divided into small 1-8bits sections as shown in table 1.

Table 1: Detail of Situation Encryption.

Bit no.	nBits	nValues	Meanings
1-8	8	256	Player character animation set ID. [As said in section 2.2]
9-12	4	16	Delta position in X axis. [Divide distance into 9 ranges]
13-14	2	4	Delta position in Y axis. [Divide distance into 4 ranges]
15-18	4	16	Enemy character state. [Group into 6 stages: Normal, Attacking, Blocking, Dizzy, Damaged, Invulnerable]
19	1	2	Character's bullet state. [Have or not]
20-22	3	8	Delta position in X axis between player character and enemy's bullet. [Divide distance into 8 ranges]
23-29	7	128	Damage value that enemy got in that frame (use for combination attack decision).
30	1	2	Player character side. [Left or Right]
31	1	2	Is player at corner. [Yes or No]
32	1	2	Is enemy at corner. [Yes or No]

There are ten criteria that we use for identifying the game state (ten rows in table1). Bit 1 to 8 store the animation set ID of the action that the player character performs in that frame situation. The animation set value comes from the animation set database described in section 2.2. When the player character is in any normal standing frame (frame id 0 to 6), the value in the first 8 bit will be 0. As a brief example, the situation that two characters are standing at the beginning of a battle will be encrypted as "1,792". Every criterion for this particular scene will have 0 as its value, except the delta position in the x axis, which will have its value equal to 7 due to the distance between characters at the beginning of battle (150 units). Details of this encryption can be changed to match other games or other platforms.

2.4 Creating Ghost AI File

When the scanning process discovers that animation set change takes place, the situation in the frame before that discovered frame is encrypted into 32-bit data (integer) by the process in section 2.3.5. Its corresponding case base can now be created by combining the situation ID (32-bit situation encryption result) with its response action list. An example of our case base is shown below.

```
SituationID: 0000000000
TotalRatio: 03 TotalNextAni: 02
    NextAni: Punch-Light-Close Ratio 2
    NextAni: Kick-Heavy-Close  Ratio 1
:
SituationID: 2684356352
TotalRatio: 01 TotalNextAni: 01
    NextAni: Hadouken          Ratio 1
```

Each case will have `situationID` for representing each game situation. `TotalRatio` is the number of incidents the player encounters that situation. `TotalNextAni` is the number of different animation sets that the player performs when facing that situation. It is followed by the list of those animation sets and the number of times the player performs each animation set. The ratio of each animation set and the total number of sets will be used in response selection while the ghost AI is actually running.

From above example cases, the player encountered situation 0 three times and decided to do a light-punch twice and a heavy kick once. These cases should be kept in a data structure that is convenient and fast to insert and find because we need to know whether the situation is a new situation that player never encounters (so we can add new data from scratch), or an old situation that updates the response action list. In our experiment we chose *map* of standard template library (STL), which is a balanced binary search tree, to store the cases. The tree was written into our ghost AI file. Using file allows for future modifications of the knowledge base.

3. USING GHOST AI

To run the ghost AI, first, the game needs to load any required database such as the animation set database. Then it needs to load the ghost AI case base into some data structure that allows quick finding and matching. A new case is never inserted while running the ghost AI.

From the data in section 2.4, the game first loads all cases into the *map*. When the `situationID` 0 takes place, the case that has `situationID` 0 in the *map* is searched. It will be found and returned. That case has a total ratio of 3 and has two next animations (light-punch with ratio 2 and heavy-kick with ratio 1). The game then randomly selects one of these actions corresponding to the ratio value and sends a command to perform that action.

When a ghost AI is running, if used with a suitable data structure such as a balanced binary search tree, searching any case is guaranteed to use *O(log n)* amount of time (when n is the number of cases). A ghost AI with one thousand cases should find a result in the tenth search. Each case based data uses approximately 40 bytes of memory. Therefore, a thousand-case ghost requires only 40KB of memory. In short, creating and running our ghost AI does not slow down the game or consume much memory at all.

4. VERIFYING METHOD AND RESULTS

The best way to evaluate a ghost AI's similarity to its creator should be: letting its creator verify with his own eyes. But sometimes, people can make incorrect judgments, forgetting even their own playing styles. Therefore we designed a measurable method for evaluating the ghost AI.

4.1 The Experiment

We appointed thirty two SFZ3 players and let them play the game for approximately 2 to 10 minutes. We recorded their game events in VMV file format (recording the beginning game state and controller sequence) and created their ghost AI. After that, we let the player semi-play the game again two more times, while their ghost AI was playing and while their own playing movie was playing. The term semi-play means players see their ghosts or their own movies playing while pressing the controller, imagining that they are controlling their characters in that situation. We wanted to compare the controller signals of the ghosts with the players' signals. We also wanted to compare the players against their video.

Controller signals should not be compared frame-by-frame, because only 1 frame delay (1/60 second) will cause the rest of the matching process to fail.

Therefore the controller signals need to be normalized before any comparison can be done. In our approach, we normalized the signals by splitting the signals into parts. Each part contained approximately 5 to 15 signals. After that, we combined all the same signals that appear continuous into one signal (when a player presses one button normally, it takes approximately 6-8 frame, so it gives out 6-8 continuous signals). For example, if the signals are as follows:

```
Raw ghost signal:
16,16,16,32,32,32,64,64,64,64,128,128,256,256,256
Raw player signal:
16,16,16,16,16,16,32,32,32,32,64,64,128,128,128
```

After normalized they will be like these.

```
Normalized ghost signal: 16,32,64,128,256
Normalized player signal:16,32,64,128,0
```

It can be seen from the example that if we compare raw signals directly the result will be 3 of 15 signals match. The matching result is not correct because identical commands that are pressed for slightly different amount of time will be regarded as being different. However, if we compare the two signals after our normalization, the match is 4 out of 5.

We had two methods for slicing controller signals. In the first method, we sliced every 15 frames. We had tried several values and this value gave the best result. Too small values made the normalization meaningless, while too large values put more than one signals in the same frame, making the result unreliable. In the second method, we performed the slicing every time the signal of the ghost AI or the player movie changed values, based on the assumption that matching signals should occur in the same frame time period as its counterpart. With the second method, we always had one signal per slicing window. We also gave score if there were some similarity between controller signals. For example, if the ghost AI was pressing down-forward and the player was pressing forward only, we gave similarity score of 0.5 (50%) to the ghost AI.

4.2 Result

The result of our experiment is illustrated in figure 7 and table 2. *Player_Player%* is the similarity (in percentage) between each player's own movie and his actual control when re-playing the situation in the movie. *Ghost AI_Player%* compares each ghost AI with its corresponding player's re-play. *Delta%* is the difference between the two comparisons. Table 2 displays the overall statistical summary. *Delta A* and *Delta B* indicate delta percentage points between the result of [player's own movie vs. player] and [ghost AI vs. player]. *Score* is the score that the players evaluate their ghosts' similarity to their fighting styles based on their feelings.

Both signal slicing methods gave similar results. But the second method gave less matching percentage points. This is likely because the number of signals after the normalization was less than in the first method. With many long signals in play, such as idle signals, the first method scored better because it did not compress long signals into one signal. For the first method, the average similarity between ghosts and the players is 26.33%. This may seem small. But if we look at the comparison between the players and their own movies, the similarity is only 34.96%. The ghosts' performances were therefore very close to players' performances (75.31% close). Some ghosts even scored better than their corresponding players.

An average satisfactory score given by players is 72.2%, which is good. The players thought that the ghosts sometimes performed more attacks and fewer defenses than their creators. Some players could not distinguish between their ghosts and their own movies while semi-playing. (We did not tell the players which engine was really controlling the characters).

5. CONCLUSION AND FUTURE WORK

We propose a method and concept for creating ghost AI without having to know game source code. We used AI-TEM, an emulator based testbed to provide a commercial game testing environment. Our concept for ghost AI creation is general for all fighting games. Using SFZ3, which is a very well respected commercial game, with its basic systems being used in almost all fighting games, our findings are guaranteed to be applicable to other commercial games.

Our method produces good results. Ghost AIs display their creators' playing styles even when the training time is short. The two-minute average training time we used is equal to a match time in an average fighting game.

For future experiment we are interested in exploring techniques for ghost AI in team based fighting games, where characters can cooperate. Another interesting future work is developing AI that can adapt and counter an opponent's play style.

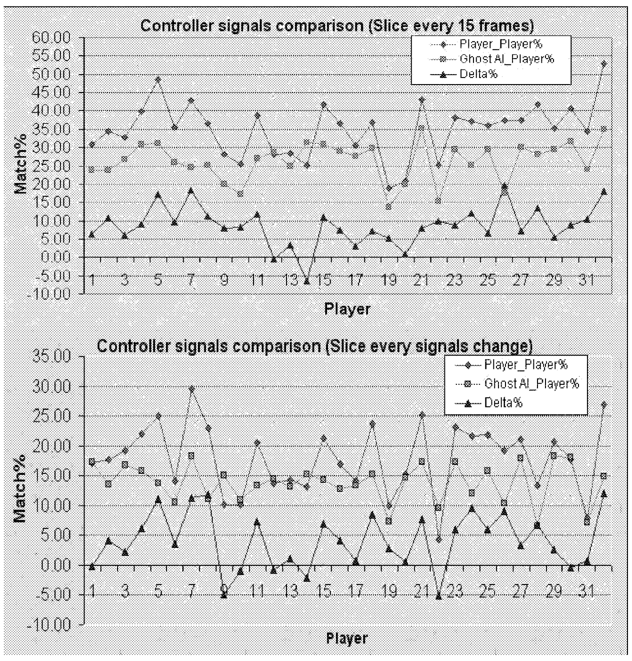


Figure 7: Players vs. Movies and Players vs. Ghosts.

Table 2: Summary Result of Experiment. A: Slice Every 15 Frames, B: Slice Every Time When Signal Change.

Summary	Player_ Player A	Ghost AI_ Player A	Delta A		Player_ Player B	Ghost AI_ Player B	Delta B		Score
Min	18.81	13.6	-6.45		4.37	6.54	-5.18		44
Max	52.84	35.18	19.82		29.51	18.27	12.03		90
Average	34.96	26.33	8.62		17.93	13.81	4.12		72.2

REFERENCES

Bailey, C. and M. J. Katchabaw. 2005. An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games. *Proceedings of the 2005 GameOn North America Conference.* Montreal, Canada.

Graepel Thore, Ralf Herbrich, Julian Gold. 2004. Learning to fight. *International Conference on Computer Games: Artificial Intelligence, Design and Education*

Kendall Graham, Kristian Spoerer. 2004. Scripting the Game of Lemmings with a Genetic Algorithm. *Proceedings of the 2004 Congress on Evolutionary Computation,* IEEE Press, Piscataway, NJ, pp. 117-124

Ponsen Marc J.V., Hector Munoz-Avila, Pieter Spronck, and David W. Aha. 2005. Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning. *Proceedings The Twentieth National Conference on Artificial Intelligence.*

Spronck Pieter, Ida Sprinkhuizen-Juyper, Eric Postma. 2004. Online Adaptation Of Game Opponent AI With Dynamic Scripting. *International Journal of Intelligent Games and Simulation*, Vol. 3, No. 1, University of Wolverhampton and EUROSIS, pp. 45–53.

Thunputtarakul Worapoj and Kotrajaras Vishnu. 2006. AI-TEM: Testing Artificial Intelligence in Commercial Game using Emulator. *8th CGAMES International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games.* Louisville Kentucky, USA.

TEMPORAL DIFFERENCE CONTROL WITHIN A DYNAMIC ENVIRONMENT

Leo Galway
Darryl Charles
Michaela Black

School of Computing & Information Engineering
University of Ulster at Coleraine
Cromore Road
BT52 1SA
E-mail: {galway-l1, dk.charles, mm.black}@ulster.ac.uk

Colin Fyfe

School of Computing
The University of Paisley
Paisley
PA1 2BE
E-mail: colin.fyfe@paisley.ac.uk

KEYWORDS
Reinforcement Learning, Sarsa, Digital Games, Pac-Man.

ABSTRACT

The aim of this paper is to investigate reinforcement learning, specifically the use of Temporal Difference learning methods for the generation of player character movement, within a dynamic, digital game environment. Using a variation of the classic arcade game Pac-Man, the Sarsa and Sarsa(λ) algorithms have been utilised for the control of a Pac-Man game agent, with results indicating that the chosen learning algorithms are successful in achieving the underlying objectives of the game agent. However, a number of trade-offs between the objectives of the game agent must be made during the selection of parameter values for the learning algorithms. In the experiments presented herein, the incorporation of a priori game information into the chosen learning algorithms has shown an improvement in the performance of the game agent in terms of both the score obtained and time taken per game.

INTRODUCTION

Digital games provide an interesting test-bed for machine learning research due to the characteristically non-deterministic, dynamic nature of digital game environments (Spronck 2005). The incorporation of both traditional and modern Artificial Intelligence (AI) techniques into a game engine's AI sub-system (*game AI*) often result in predictable and static responses from computer controlled game agents. In order to generate reactive and believable game agent behaviours the use of effective machine learning techniques is required (Charles 2003). However, the effective use of machine learning algorithms within digital games is restricted by a number of operational requirements (Galway et al. 2006). These include ensuring that the computational time taken for a learning algorithm's operation is as efficient as possible (Maes 1995; Spronck 2005; Baekkelund 2006), the learning algorithm must be able to learn new game agent behaviours in response to a changing game environment (Maes 1995; Van Lent and Laird 1999) and any game agent behaviours learned should be both appropriate within the context of the underlying game and visible to the player (Maes 1995; Van Lent and Laird 1999; Spronck 2005; Baekkelund 2006). By contrast, not all operational requirements for the integration of machine learning techniques within digital games are purely restrictive (Galway et al. 2006). By incorporating prior knowledge about the learning task into the learning algorithm and knowledge representation used, the efficiency of a machine learning algorithm within a digital game environment may be improved (Baekkelund 2006). Although a large variety of techniques exist within the machine learning domain, reinforcement learning provides an approach to agent-based learning which focuses on an agent's interactions with its environment (Sutton and Barto 1998), thus providing a learning methodology appropriate for use within digital game environments.

This paper investigates the use of reinforcement learning as a potentially suitable machine learning technique for controlling a game agent within a dynamic environment. The classic arcade game 'Pac-Man' is used as the test-bed for the integration of a reinforcement learning-based controller for the Pac-Man agent. Experiments with regard to the setup and implementation of the Temporal Difference control methods, specifically the Sarsa and Sarsa(λ) algorithms, have been performed. These include investigations into the utilisation of a priori information within the chosen learning algorithms. Details of the experiments will be presented along with analysis of the results obtained by the game agent controllers. Results shall be discussed in terms of the game objectives of the Pac-Man agent together with the efficiency and believability requirements necessary for the effective incorporation of machine learning techniques into digital games.

BACKGROUND

Pac-Man is a well-known digital game in which the primary objective for the player is to achieve as high a score as possible by manoeuvring the Pac-Man agent around a 2D, grid-based environment in order to eat 'dots' while at the same time avoiding being eaten by four opponent ('ghost') agents. Consisting of a maze of corridors, containing a predefined number of dots and 'power-pills', the reduction in the number of dots, along with the deterministic behaviour of the ghosts and the Pac-Man agent's ability to eat the ghosts for a finite period of time after consuming a power-pill, gives rise to the dynamic nature of the game environment (Gallagher and Ryan 2003; Yannakakis and Hallam 2004; Gallagher and Ledwich 2007). From the point of view of the player, the overall aim of maximising the score obtained within such an environment may be considered as the problem of developing game-play strategies through a combination of navigation, task prioritisation and risk assessment (Koza 1993; Gallagher and Ryan 2003; Gallagher and Ledwich 2007). As a suitable test-bed for real-time machine learning research, the dynamic environments presented by such predator/prey style games also offer the advantage of being easily broken down into a finite set of states (Hartley et al. 2004). Within the academic digital game research literature a number of attempts have been made to incorporate machine learning techniques into variations of the Pac-Man game in order to implement controllers for both the Pac-Man agent and ghost agents. A variety of neuro-evolutionary approaches to the generation of game agent controllers have been proposed. These include the generation of a Pac-Man agent move evaluation mechanism (Lucas 2005), the utilisation of local neighbourhood 'windows' of game features as input vectors to an evolved multi-layer network (Gallagher and Ledwich 2007), the use of an evolved probabilistic rule-base coupled with a finite-state machine (Gallagher and Ryan 2003), evolved neural networks for controlling the ghost agents (Yannakakis and Hallam 2004), and the use of genetic programming to evolve a series of primitive movement operators for control of the Pac-Man agent (Koza 1993). In the majority of these approaches however, a number of distance metrics have been used (Koza 1993; Gallagher and Ryan 2003; Yannakakis and Hallam 2004; Lucas 2005). This is explicit information which may not necessarily be known to a player during game-play. Although using a variation of the Pac-Man game, the focus of the work by Yannakakis and Hallam (2004) is to determine a metric for a player's interest in the game. One aspect of their proposed interest metric that provides a suitable mechanism for establishing the believability of a game agent's behaviour is the use of entropy. By determining the entropy of the moves made by a game agent, the spatial diversity of the agent over the game environment may be measured, with higher values of entropy indicating a more interesting range of movement by the agent over the course of a game (Yannakakis and Hallam 2004). Alternative approaches to the real-time control of game agents within a 2D predator/prey game include the use of Markov Decision Processes (MDP). By decomposing the game environment, based on a variation of Pac-Man, into a number of 'states' with associated 'reward' values, opponent agents were successfully controlled by learning a near-optimal policy which reflected the maximum expected utility to be gained by the agent for any transition between the states of the game environment (Hartley et al. 2004).

Temporal Difference Learning Methods

Based on the discovery of an optimal policy for an agent performing actions within the discrete time, mathematical model of a MDP, reinforcement learning comprises a set of algorithms and techniques that involve learning a sequence of actions in order to maximize an accumulated discounted reward over a period of time, where each action is associated with a reward or penalty. Through exploration and exploitation of the agent's environment a *control policy* can be learned that maximises the environmental feedback for a sequence of actions without requiring explicit training from a domain expert (Sutton and Barto 1998; Duan et al. 2002; Graepel et al. 2004; Pfeiffer 2004; Baekkelund 2006). Although a MDP assumes that the environment is stationary, deterministic and contains only one agent, a class of algorithms, known as *Temporal Difference (TD) methods*, which include the *Sarsa* and *Sarsa(λ)* learning algorithms, can be used to overcome the limitations of deterministic MDP environments (Sutton and Barto 1998). These learning algorithms are characterised by on-policy, model-free learning methods in which an agent repeatedly evaluates the results of its actions in terms of the reinforcement signal it receives and estimates of the values of both current and future state transitions that occur. Typically, the estimated value for each state-action pair (*Q-value*) is stored in a look-up table (*Q-Table*). Sarsa(λ) extends the Sarsa algorithm by incorporating the use of eligibility traces in order to maintain an accumulating trace of state-action pairs visited, which signify the amount of reward a state-action pair is eligible to receive. As each step of learning proceeds, the eligibility trace values for each state-action pair decay in relation to the *trace-decay* parameter (λ), unless the agent has visited the state-action pair. Although the use of eligibility traces gives rise to larger computation times, they offer the advantages of faster convergence to an optimal policy and permit the use of temporal difference methods when the learning task is partially non-Markov (Sutton and Barto 1998). Within the academic digital game research literature, reinforcement learning techniques have been applied to a variety of games in order to learn control policies for game agents. In particular, a near-optimal control policy for agents in the fighting game "Tao Feng" has been generated using Sarsa(λ) with a linear function approximation for Q-value representation (Graepel et al. 2004). Similarly, Sarsa(λ) has been used in the strategy game "Settlers of Catan" in order to generate a control policy for game agent strategies. By pre-training the value function representation with domain specific knowledge, improvements were found in the time taken to learn the control policy, subsequently resulting in an overall improvement in the game agent behaviours (Pfeiffer 2004).

EXPERIMENTAL SETUP

Pac-Man Game Environment

For the experiments discussed within this paper, the game environment employed was a variation of the classic 2D arcade game Pac-Man. The configuration of a 20x20 grid of game dependent 'features' establishes the 'level' of the game used and comprises; walls, dots, power-pills, tunnels, inaccessible spaces (i.e. grid cells for the starting position of the ghost agents) and empty spaces (i.e. grid cells where a dot/power-pill has been eaten). Unlike the arcade version of the game, a single level has been used for all games played and was initially populated with a total of 176 dots and 4 power-pills, each worth a respective score of 5 and 50 points. Both the Pac-Man agent and 4 ghost agents begin each game in predefined starting locations. Throughout the course of a game, if the Pac-Man agent eats a power-pill the state of the game environment temporarily changes from the default 'Attack' state to the 'Evade' state, during which the Pac-Man agent may eat the ghost agents. 300 simulation steps have been specified as the duration of the Evade state, regardless of multiple power-pills being eaten or successive power-pills being eaten when in the Evade state. A score of 100 is obtained for each ghost agent eaten by the Pac-Man agent during the Evade state, with all eaten ghosts being regenerated (in the current game state) within the predefined ghost starting position in the next simulation step. If a ghost agent eats the Pac-Man agent, the Pac-Man agent loses 1 out of 5 lives and is regenerated in its predefined starting location. For both the Pac-Man and ghost agents, the range of movement available is in the 4 directions; North, South, East and West. Similar to the original version of the game, the Pac-Man agent makes a move 50% more often than the ghost agents however, the choice of moves for the ghost agents are non-deterministic, with each ghost agent's moves being generated randomly, thus preventing the learning algorithm from simply learning the ghosts' movement patterns (Gallagher and Ledwich 2007). In order to allow for a direct comparison of games played during experiments, the Pac-Man agent has been restricted to a maximum of 1000 moves per game. A game ends when all the Pac-Man agent's lives are gone, the total set of dots has been eaten, or the maximum number of moves for the Pac-Man agent has been reached. Based on the entropy metric proposed by Yannakakis and Hallam (2004), the 'raw' entropy of the Pac-Man agent's moves over the course of a game, E_r, is given by Equation (1). Each raw value is subsequently normalized using Equation (2) in order to produce an entropy value, E_n, within the range [0,1].

$$E_r = -\sum \frac{p_i}{P} \log_2\left(\frac{p_i}{P}\right) \qquad (1)$$

$$E_n = \left(\frac{E_n}{\log_2(P)}\right) \qquad (2)$$

where p_i is a count of the number of times a specific grid cell is visited by the Pac-Man agent and P is the total number of moves made by the Pac-Man agent during the course of a game (Yannakakis and Hallam 2004).

Game Agent Control Algorithm Setup

For all games played, the choices of moves for the Pac-Man agent were determined using an on-policy TD control algorithm. The game environment was broken down into a 20x20 grid, where each grid cell was represented as a state with 4 possible actions, each action corresponding to one of the possible moves that may be made by the Pac-Man agent. Due to the finite number of state-action pairs, a look-up table has been used for value function representation. While attempting to learn an optimal control policy, state-action pairs are selected using a ε-greedy action selection policy. In order to help maintain a high degree of exploitation of learned state-action pairs, a low exploration rate, $\varepsilon = 0.1$, was used. For each new state that occurs as a result of taking the chosen action, the game environment provides positive and negative feedback directly corresponding to the game feature (i.e. wall, dot, power-pill, tunnel, inaccessible space, empty space or ghost agent) which exists in the grid cell represented by the new state. Every time a move is required for the Pac-Man agent the TD learning algorithm is run for 100 episodes of learning, with a predefined number of steps per episode, and the Q-values updated. A move was then chosen for the agent by selecting the action corresponding to the Q-value for the current state that had achieved the largest value during learning. At the start of each game, 1000 episodes of learning were performed as a *prediction* phase in order to obtain an initial, optimistic set of Q-values. By beginning the *control* phase of each game with a pre-initialised set of Q-values, the Pac-Man agent is less likely to repeatedly explore areas of the game level for which initial moves made during the control phase provide positive rewards. For all episodes of learning during the prediction phase a static version of the game level was used, consisting of stationary ghost agents and randomly generated positions for the Pac-Man agent. This process of a single prediction phase during game initialisation, followed by a control phase before each Pac-Man agent move, was repeated for every game played.

RESULTS

In the initial set of experiments the Sarsa algorithm was used in order to determine a suitable number of steps per episode of learning for the control algorithm. Based on the results obtained, a second set of experiments were performed using the Sarsa(λ) algorithm to investigate the use of eligibility traces in terms of the score, number of dots eaten, time taken for the control phase and the normalized entropy values obtained. A final set of experiments were then performed in order to investigate the effect on the game agent's performance caused by incorporating a priori information into the control algorithm. For these experiments two forms of a priori information were used: (a) game state-based reward values, with a negative reward value received by the Pac-Man agent for a ghost agent encounter when the current game state was the Attack state and a positive reward value being received when the current game state was the Evade state, (b) valid Pac-Man agent moves, where any action chosen during learning which results in an invalid move leads to the new state associated with the action being disregarded and subsequently replaced by the current state. For the third set of experiments, investigations were performed using: (1) no a priori information, (2) game state-based reward values, (3) valid Pac-Man agent move information, (4) both game state-based reward values and valid Pac-Man agent move information.

Number of Steps Per Episode of Learning Test Results

During all experiments the best game agent performance was achieved using a relatively high learning rate ($\alpha = 0.1$) along with a high discounting factor ($\gamma = 0.9$). In the first set of experiments the Sarsa algorithm was used and the number of steps per episode of learning tested with values in the range [1, 10]. For each test of the step size parameter 20 games were played. Figure 1 illustrates the mean scores obtained while Figure 2 shows the mean number of dots eaten for the range of steps per learning episode. Table 1 lists the mean time taken, in seconds, for the control phase of all games played alongside the mean normalized entropy values obtained. From Figure 1 and Figure 2 it can be observed that the highest mean score was obtained using 8 steps per learning episode however, the highest mean number of dots eaten was obtained using 2 steps per learning episode.

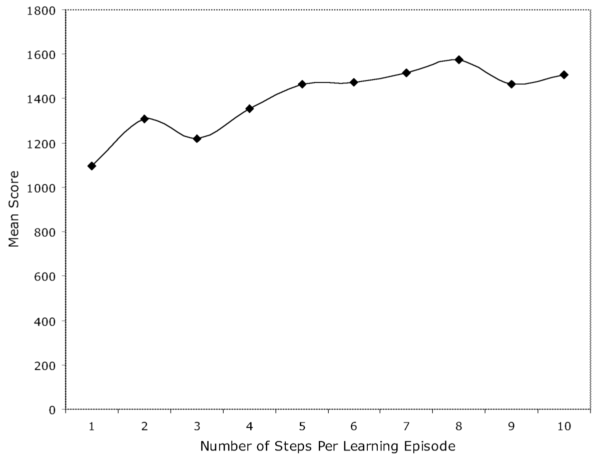


Figure 1: Mean Score

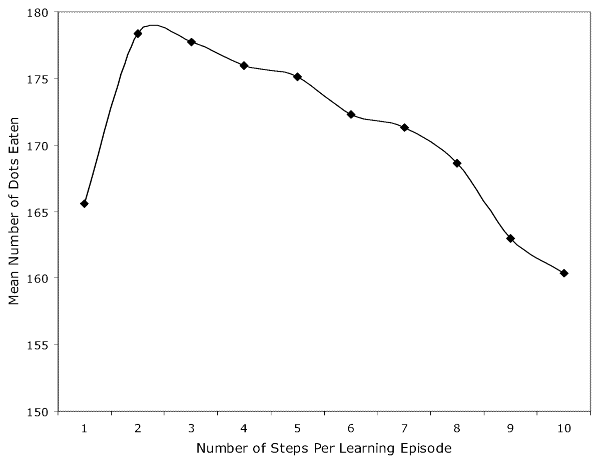


Figure 2: Mean Number of Dots Eaten

Table 1: Mean Time Taken & Mean Normalized Entropy

Number of Steps	Mean Time (sec)	Mean Entropy
1	124.33	0.681
2	122.58	0.733
3	123.55	0.733
4	131.07	0.717
5	131.30	0.713
6	131.65	0.700
7	132.01	0.700
8	132.00	0.699
9	132.12	0.686
10	132.04	0.677

As Table 1 shows, the lowest mean time taken during the control phase also corresponds to the use of 2 steps per learning episode, which also achieves a high mean normalized entropy value. Although the range of mean entropy values is relatively small, higher entropy values suggest greater coverage of the game level by the Pac-Man agent. However, a trade-off between the number of steps that give rise to a high score and the number of steps that achieve a high normalized entropy must be made in order for the Pac-Man agent to successfully prioritise between the tasks of eating all the dots and eating the ghost agents when applicable. The number of steps used also has a visible effect on the behaviour of the Pac-Man agent; a value of 1 step per learning episode results in visibly erratic behaviour, with the Pac-Man agent frequently oscillating between neighbouring grid cells as each move is made. As the number of steps used was increased, the erratic movement was visibly reduced. Figure 3 illustrates the normalized entropy values obtained over the course of a single game for both 1 step and 10 steps per learning episode. As can be seen, the entropy values obtained over a single game are, in general, higher with less variation when a larger number of steps per learning episode are used. Again, a trade-off is required between suitable on-screen behaviour of the Pac-Man agent and the potential result obtained.

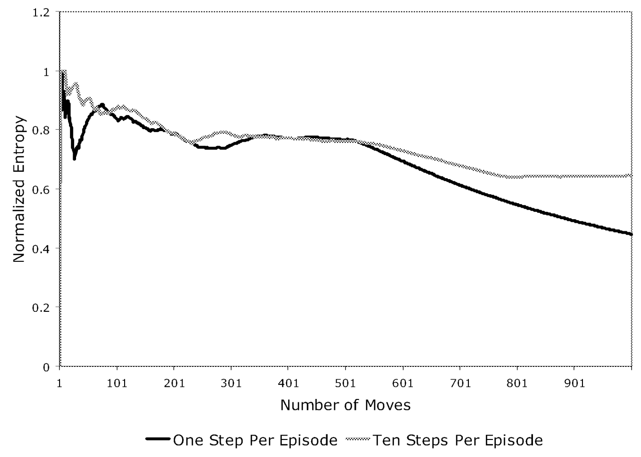


Figure 3: Normalized Entropy Values Over a Single Game

Trace-Decay Parameter Test Results

In the second set of experiments the Sarsa(λ) algorithm was used and a range of values for the trace-decay parameter, λ, tested using 2 steps per learning episode. The values tested for λ were in the range [0.1, 0.9]. Again, 20 games were played for each value specified for λ. The results obtained for the mean score and mean number of dots eaten are shown in Figure 4 and Figure 5 respectively, with the mean time taken, in seconds, for the control phase and mean normalized entropy of the Pac-Man agent given in Table 2. From Figure 4 and Figure 5 it is apparent that high values of λ result in a sharp decline in both the mean score and mean number of dots eaten, suggesting that a lower trace-decay is required in order for the Pac-Man agent to successfully navigate the game environment. With less influence from future updates on recent moves during learning, a lower value for λ would seem preferable given the dynamic nature of the game environment. Again, the test which results in the highest mean score ($\lambda = 0.7$) does not necessarily equate to the test with the highest mean number of dots eaten. For both $\lambda = 0.3$ and $\lambda = 0.4$, the Pac-Man agent consumed all 180 dots (including power-pills) for 10 out of 20 games played using each λ setting. The highest mean number of ghost agents eaten by the Pac-Man agent and the lowest mean number of Pac-Man agents eaten by the ghost agents also occurred when λ = 0.7, hence the increase in the mean score obtained. This would suggest a trade-off is required for λ that results in a large coverage of the game level by the Pac-Man agent and a value that enables the Pac-Man agent to successfully prioritise between the tasks of avoiding the ghost agents and eating the ghost agents when appropriate.

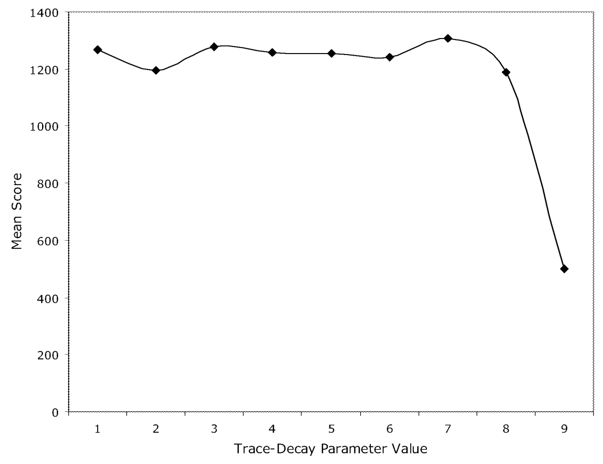


Figure 4: Mean Score

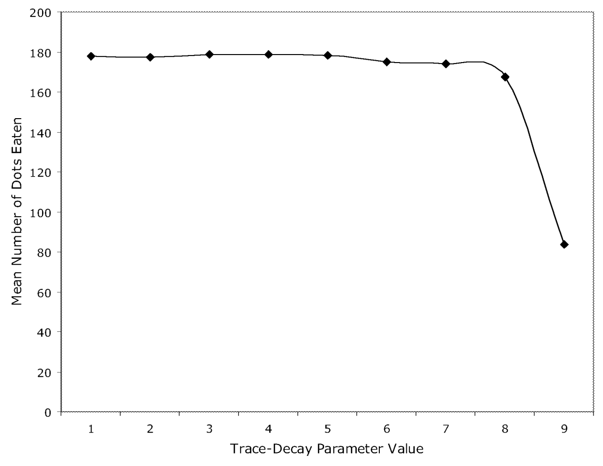


Figure 5: Mean Number of Dots Eaten

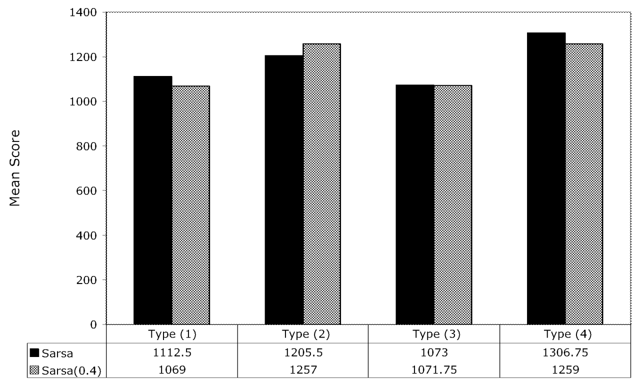
Table 2: Mean Time & Mean Normalized Entropy

Trace Decay (λ)	Mean Time	Mean Entropy
0.1	124.14	0.729
0.2	123.88	0.731
0.3	120.16	0.734
0.4	118.72	0.741
0.5	122.71	0.732
0.6	226.07	0.705
0.7	223.52	0.722
0.8	229.78	0.705
0.9	226.05	0.548

Almost twice as many Q-value updates occurred when λ was in the range [0.6, 0.9]. Correspondingly, the mean time taken for the control phase almost doubles in these cases. In general, the use of eligibility traces results in a more consistent range of scores over all games played. The decision to use eligibility traces must balance any performance gain, in terms of the computation time taken and consistency of results, over the overall performance of the Pac-Man agent.

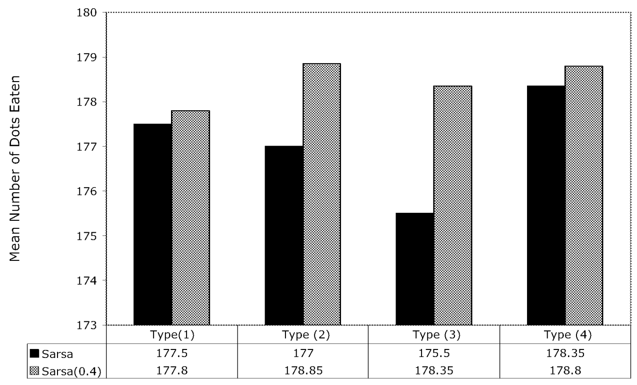
A Priori Information Test Results

In the final set of experiments the integration of a priori information within both the Sarsa and Sarsa(λ) algorithms was tested. For both algorithms the number of steps per learning episode was set to 2 and for Sarsa(λ) the value of λ was set to 0.4. Four types of a priori information were used during the experiments, as previously outlined at the start of the Results section, with 20 games being played for each type of a priori information used. The resulting mean score and mean number of dots eaten for both algorithms are given in Figure 6 and Figure 7 respectively.



	Type (1)	Type (2)	Type (3)	Type (4)
■ Sarsa	1112.5	1205.5	1073	1306.75
▨ Sarsa(0.4)	1069	1257	1071.75	1259

Figure 6: Comparison of Mean Score



	Type(1)	Type (2)	Type (3)	Type (4)
■ Sarsa	177.5	177	175.5	178.35
▨ Sarsa(0.4)	177.8	178.85	178.35	178.8

Figure 7: Comparison of Mean Number of Dots Eaten

Correspondingly, Table 3 gives the mean time taken for the control phase and the mean normalized entropy values obtained.

Table 3: Mean Time & Mean Normalized Entropy

	Sarsa		Sarsa(0.4)	
A Priori Test	Time	Entropy	Time	Entropy
Type (1)	124.73	0.733	121.20	0.739
Type (2)	122.41	0.733	120.46	0.737
Type (3)	129.09	0.723	120.47	0.735
Type (4)	122.55	0.733	118.71	0.741

From Figure 6 and Figure 7 it can be observed that for both Sarsa and Sarsa(λ) the integration of Type (4) a priori information leads to a higher mean score and higher mean number of dots eaten. By making use of game state-based reward values, in conjunction with valid move information, the impact on the mean score and mean number of dots eaten is greater, in the case of the Sarsa algorithm, than the use of either type of a priori information alone. For Sarsa(λ) there is an improvement in game agent performance when using Type (4) a priori information however, only a relatively small improvement is gained over the use of Type (2) a priori information. When only Type (3) a priori information is integrated into the learning algorithm less coverage of the game level occurs by the Pac-Man agent, as indicated by the normalized entropy values shown in Table 3. In general, the use of Type (4) a priori information may improve the overall results obtained for both algorithms.

CONCLUSION

It has been shown that the Temporal Difference learning methods Sarsa and Sarsa(λ) may be successfully used to control a game agent within a dynamic game environment. A period of learning within a static game environment may be utilised in order to bootstrap the initial Q-values used during game agent control learning. Both versions of the Sarsa algorithm have been shown to provide suitable controllers for the game agent however, given a suitable choice of value for the trace-decay parameter, Sarsa(λ) has been shown to marginally outperform Sarsa in terms of the coverage of the game level by the game agent and the time taken for control learning. Regardless of the Temporal Difference learning method used, a number of trade-offs regarding the algorithmic parameters used are required, including a trade-off between the number of steps used per learning episode that permits the game agent to achieve the task of obtaining a high score and the task of fully exploring and navigating the game environment. A further trade-off is required between the number of steps per learning episode that results in a high overall level of game agent performance and the appropriate on-screen behaviour of the game agent. Similarly, when using Sarsa(λ) a further trade-off is required between the trace-decay parameter resulting in task prioritisation by the game agent and a suitable degree of coverage of the game environment. Subsequently, by incorporating a priori information into both the learning algorithm and reward value function used, the overall performance of the game agent may be improved. One drawback of the experiments presented is the use of only one game level. Further research should be conducted into the use of multiple game levels, thereby attempting to obtain a generalization of the control policy. In addition, further research into the retention of the Q-Table between successive games in order to overcome the need for Q-Table initialisation using a static environment should be performed.

REFERENCES

Baekkelund, C. 2006. "A Brief Comparison of Machine Learning Methods." In *AI Game Programming Wisdom 3*, S. Rabin (Ed.). Charles River Media, Hingham, MA. 617-631.

Charles, D. 2003. "Enhancing Gameplay: Challenges for Artificial Intelligence in Digital Games." In *Proceedings of Digital Games Research Conference* (University of Utrecht, Netherlands).

Duan, J., Gough, N. and Mehdi, Q. 2002. "Multi-Agent Reinforcement Learning for Computer Game Agents." In *Intelligent Games and Simulation*, Q. Mehdi, N. Gough and M. Cavazza (Eds.). University of Wolverhampton, UK. 104-109.

Gallagher, M. and Ledwich, M. 2007. "Evolving Pac-Man Players: Can We Learn From Raw Input?" *IEEE Symposium on Computational Intelligence and Games*. 282-287.

Gallagher, M. and Ryan, A. 2003. "Learning To Play Pac-Man: An Evolutionary Rule-Based Approach." *IEEE Congress on Evolutionary Computation*, 2462-2469.

Galway, L., Charles, D. and Black, M. 2006. "A Set of Guidelines for the Evaluation of Real-Time Machine Learning Techniques for use in Digital Games." In *Proceedings of 9th Annual Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games* (Dublin, Ireland). 52-56.

Graepel, T., Herbrich, R. and Gold, J. 2004. "Learning To Fight". In *Computer Games: Artificial Intelligence, Design and Education*, Q. Mehdi, N. Gough, S. Natkin & D. Al-Dabass (Eds.). University of Wolverhampton, UK. 193-200.

Hartley, T., Mehdi, Q. and Gough, N. 2004. "Applying Markov Decision Processes To 2D Real Time Games". In *Computer Games: Artificial Intelligence, Design and Education,* Q. Mehdi, N. Gough, S. Natkin & D. Al-Dabass (Eds.). University of Wolverhampton, UK. 55-59.

Koza, J.R. 1993. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, MA.

Lucas, S. 2006. "Evolving A Neural Network Location Evaluator To Play Ms. Pac-Man." 2005. *IEEE Symposium on Computational Intelligence and Games*, 203-210.

Maes, P. 1995. "Artificial Life Meets Entertainment: Lifelike Autonomous Agents." *Communications of the ACM* 38, No.11, 108-114.

Pfeiffer, M. 2004. "Reinforcement Learning of Strategies for Settlers of Catan." In *Computer Games: Artificial Intelligence, Design and Education,* Q. Mehdi, N. Gough, S. Natkin & D. Al-Dabass (Eds.). University of Wolverhampton, UK. 384-388.

Spronck, P. 2005. "A Model for Reliable Adaptive Game intelligence." In *Proceedings of 2005 Workshop on Reasoning, Representation, and Learning in Computer Games* (Washington, VA).

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

Van Lent, M. and Laird, J. 1999. "Developing an Artificial Intelligence Engine." In *Proceedings of 1999 Game Developers Conference* (San Jose, CA). 577-588.

Yannakakis, G. and Hallam, J. 2004. "Evolving Opponents For Interesting Interactive Computer Games." In *Proceedings of 8th International Conference on Simulation of Adaptive Behaviour*, 499-508.

BIOGRAPHY

LEO GALWAY was awarded a Masters with Distinction in Computing & Intelligent Systems from the University of Ulster in 2005. Having completed his undergraduate studies in 1998, receiving a Bachelor of Science degree in Computer Science from The Queen's University of Belfast, he was employed as a system level software engineer for a period of 6 years before returning to full time education. He is currently pursuing a PhD degree at the School of Computing & Information Engineering from the University of Ulster, with an emphasis applied intelligent techniques for digital games. His research interests include artificial intelligence, evolutionary computing and machine learning.

EVALUATION OF MULTIAGENT TEAMS VIA A NEW APPROACH FOR STRATEGY GAME SIMULATOR

Vicente V. Filho, Clauirton A. Siebra, José C. Moura, Renan T. Weber, Patrícia C. Tedesco and Geber L. Ramalho
Centro de Informática
Universidade Federal de Perrnambuco
Av. Professor Luis Freire s/n, Cidade Universitária
50740-540, Recife – PE – Brazil
{ vvf,cas,pcart,glr }@cin.ufpe.br

KEYWORDS
Strategy games, Simulation, Multiagent Systems, Benchmarks.

ABSTRACT

This paper discusses some practical experiments using JaRTS, the *Java Real-Time Strategy* simulator, which is our proposal of simulator for real-time strategic games. The main goal of this simulator is to support the evaluation of different team approaches, implemented as multiagent systems, during pre-defined game contests. In fact, we show that the use of a simulator is important to evaluate several individual aspects and components of a team design, rather than only the final product as a whole. The experiments were carried out during a student's competition, where we could testify if JaRTS was actually able to detect positive and negative aspects of different AI approaches to Real-Time Strategy (RTS) games.

INTRODUCTION

The evaluation of any computational system is an important phase of its development process and this is not different for systems that use Artificial Intelligence (AI). For such systems, in particular, there is a trend (Hanks et al 1993) in employing simulation environments as benchmarks, which enable the evaluation of such system in different test scenarios.

Another interesting trend is the use of such environments in academic competitions (Stone 2003). This trend brings two advantages to the AI research. First, competitions motivate the investigation of solutions in specific areas. Second, they integrate the research community, providing an environment where approaches are discussed and results, raised from contests, can be analyzed in a comparative way.

The aim of this work is to describe the use of a real-time strategic games simulator, called JaRTS (*Java Real-Time Strategy*), during the evaluation of several AI techniques used to implement teams as Multiagent Systems (MAS). The experiments described in this paper were carried out during a competition among students of the Intelligent Agents Course of the Universidade Federal de Pernambuco (UFPE), and the idea was to verify if such students were able to take conclusions about their approaches via tests using our simulation environment. The students' conclusions were captured via reports, as discussed along this paper.

The remainder of this work is structured as follows. Section 2 introduces the idea of using benchmarks and simulators as option to validate and evaluate AI approaches and, in particular, multiagent systems (MAS). Section 3 discusses the use of RTS games as evaluation approach and summarizes the main RTS environments and competitions currently available. Section 4 presents JaRTS, our proposal of simulator aimed at real-time strategic games. Section 5 describes the use of JaRTS in a competition involving students of the Intelligent Agents course of the Informatics Centre – UFPE. Finally, Section 6 concludes this paper, stressing the main ideas and results of our experiments.

BENCHMARK AND MULTIAGENT SYSTEMS

The use of benchmarks, as evaluation method for multiagent systems, has become a common practice for AI systems. In fact, this method is able to evaluate the performance of such systems and provide a basis for comparative analysis. However, some criticisms (Hanks et al 1993) have been directed to the use of benchmarks. First, such method is an empirical form of evaluation, so that its users have to account for distinguishing the important evaluation events, as well as interpreting the results of such events. Second, there is not a consensus about what a representative benchmark is. Finally, results from benchmarks experiments are not general. Rather, they are related to a subgroup of possibilities from the total set of scenarios.

Simulators are a particular type of benchmark, whose generation of new states is carried out at runtime and such states depend on the activities performed inside the environment. In this way, final results are commonly unpredictable. An interesting trend related to simulators is its use in academic competitions (Stone 2003). Apart to motivate research and integrate the scientific community, competitions determine deadlines to the creation of functional systems and periodic events, using the same platform, enables the improvement of past systems and their approaches.

One of the main competitions related to multiagent systems is the *RoboCup Rescue* (RCR) (Kitano and Tadokoro 2001). RCR is an appropriate example of benchmark to multiagent research because it implements several of the prerequisites that such systems require. These prerequisites are:

- Agents don't have control on the environment so that their actions aren't the unique events can change it;
- Agents are not able to ensure that a sequence of actions will lead to a specific state, or if these actions are valid because changes can happen over the environment between decision and execution moments;
- RCR environments are complex and each of their objects presents several attributes whose values can affect the ongoing simulation;
- The environment considers communication and coordination among agents as an important simulation issue, so that there are specific rules to control such communication;
- There are several ways to measure the efficiency of approaches via, for example, number of victims or total area of fire;
- The RCR simulator has a well defined temporal model, which is based on configurable cycles;

A last and important RCR feature is its high level of parameterization, which enables an evaluation of MASs considering a significant diversity of problems and conditions. In this way, RCR users can configure the environment in such way that it can be more useful during evaluations of some particular aspect.

RTS AS BENCHMARKS

As discussed in the last section, the use of benchmarks as an alternative to evaluate MAS has received several criticisms, which are mainly based on the fact that such systems are implemented to be used in real situations. In this way, independently of the specification level of a benchmark, it will still represent a limited number of situations that could happen in real scenarios.

There exist cases, however, in which realism is not the main requirement to be considered. In such situations, the goal could be focused on the comparative evaluation among different approaches. For example, benchmarks currently used in the *Planning Systems Competition* (McDermott 2000) and the *Trading Agent Competition* (Wellman et al 2001) corroborate this idea.

Other kind of competition, which is recently receiving more attention from the research community, is related to RTS games. Note that the main competition in this area (*Orts RTS Game AI Competition*) does not have the same maturity than other AI competitions. However, several benefits can already be observed, such as the creation of a community with common interests in RTS problems.

One of the main advantages of using RTS environments as benchmark is the broad variety of problem types that they can generate. For example, consider a classic RTS game of battle between two teams. Some of the problems that can be investigated in this game are:

- Pathfinding: teams need to move along the best routes so that they decrease time and effort. It is common more elaborated versions of this problem, such as involving routes along unknown lands or facing dynamic obstacles (e.g., enemy team);
- Patrolling: a team can keep a specific area on control, or cover an area to find resources or enemies in an optimized way;
- Coordination: the components of a team ideally need some kind of coordination so that the whole work can be improved and they do not disrupt each other. For example, during an attack maneuver the team need to decide if they will attack via flank, or if the infantry should wait by the intervention of the artillery before moving forward;
- Strategic and tactical decisions: each team must plan and conduct the battle campaign (strategy) in the same way that must organize and maneuver forces in battle to achieve victory (tactics).

Then, RTS environments enable an ample set of problems or situations in which we can apply AI techniques. These environments must be configurable, following the RCR model, so that users can create more appropriate scenarios to each kind of problem.

MAIN SIMULATORS AND COMPETITIONS

There are currently several simulators to RTS games and some of them are used in open competitions. In this section we discuss four examples: ORTS, Stratagus, Boson and Glest.

Open Real-Time Strategy (ORTS) (Buro 2003) is a simulation environment for studying real-time AI problems such as pathfinding, reasoning with imperfect information, scheduling and planning in the domain of RTS games. Users of ORTS can define rules and features of a game via scripts that describe several types of units, buildings and interactions. The ORTS server accounts for loading and executing such scripts, sending the world state to their clients. These clients are applications that generate actions to be performed over the simulation server, according to their objectives and the current state of the world.

The ORTS competition comes up in cooperation with the *Artificial Intelligence and Interactive Digital Entertainment Conference* (AIIDE). There are four different contests in this competition:

- Game 1: agents must develop strategies to perform resource gathering. The idea is to get the maximum of resources within 10 minutes;
- Game 2: tank agents must destroy as many opponent buildings as possible within 15 minutes. However, at the same time, they must also defend their home bases;
- Game 3: apart the resources gathering and battle actions, agents must also deal with resource management, defining if the team should collect more resources or spend it. The decision making process involves, for example, actions to create more fighters or to attack/defend specific positions. The main goal is to destroy all opponent buildings within 20 minutes.
- Game 4: marine agents must destroy as many opponent units as possible within 5 minutes.

Stratagus (Stratagus 2007) is an engine to RTS games that supports that development of both multiplayer online and single player offline games. Stratagus is configurable and can be used to create customized games.

Boson (Boson 2005) is a RTS engine to games similar to *Command & Conquer* and *StarCraft*. The battles take place in both land and air, with a high number of units fighting at the same time. An interesting feature of this simulator is the effect of world features (e.g., wind speed, natural obstacles such as trees, and terrain variety) on actions that are performed along the scenarios. Individual units have their own moving pattern, animation, multiple weapons and intelligent pathfinding, so that they can fight against other agents or human players.

Glest (Glest 2007) is a 3D RTS engine available for several platforms. The engine can be customized via XML files, so that it is possible to define basic parameters such as life, magic, protection, and so on. The current version includes single player game against AI controlled players, providing two factions for player control: *Magic* and *Tech*, each of them with their corresponding tech trees, units and buildings.

We can discuss some main points of this analysis. First, Boson and Glest simulators are not centered in AI issues, so that we could certainly need an additional effort to define and create test scenarios to evaluate AI techniques. Differently, Stratagus provides a script language (LUA) that allows AI researchers to modify the game AI without having to change the engine code. LUA employs many familiar programming paradigms from 'common' programming languages such as C (e.g., variables, statements, functions), but in a simpler fashion (Ponsen et al 2005). Stratagus was also integrated to TIELT - *Testbed for Integrating and Evaluating Learning Techniques* – (Aha and Molineaux 2004), which is a freely available tool that facilitates the integration of decision systems and simulators. Currently, the TIELT focus is on supporting the particular integration of machine learning systems and complex gaming simulators.

ORTS has initially presented itself as a good candidate to be a MAS benchmark. However, after some trials, we have identified that ORTS is very complex, presenting several problems of configuration and behavior during its performance. Furthermore, ORTS does not provide a proper documentation. In this way, after an analysis of all our alternatives and problems, we have decided to create a new simulator that could be simpler and oriented to AI problems. This means, users just need to be worried about solutions related to AI problems. This simulator is presented in the next section.

JARTS

JaRTS (JaRTS 2007) is a RTS game simulator that enables users to focus their development uniquely on the agents' behavior. This means that users do not need to be worried about the simulation and evolution of the environment in simulation. The approach of JaRTS is very similar to the *Robocode* simulator (Li 2002), where there are basic classes that implement the main behaviors of their agents, such as move to, twirl around, or shoot in. The task of programmers is to extend such classes so that they can generate more complex behaviors.

JaRTS presents three basic types of agents: (1) the W*orker* whose role is to collect resources to fill up the *Control Center*; (2) the *Tank* whose objective is to attack or defend specific positions along the environment, and; (3) the *Control Centers* that represent a save place to resources collected by Workers and are also the target (to protect or to destroy) of *Tank* agents. These type of agents, as well as their functions and goals, were based on the ORTS agents.

Each type of agent in JaRTS can be individually simulated. For example, we can create a *MyWorker* agent and simulate its behavior in an environment populated only with *Worker* agents. This means, every agent is collecting resource without interference of enemies so that we can focus our attention on this task in specific. This is the scenario of ORTS Game 1.

In the same way, it is possible to create a *MyTank* agent and simulate it in a world where the unique goal is to fight, without considering resources gathering or management, as happens in ORTS Game 2. Furthermore, users can implement both *Worker* and *Tank* agents and evaluate their behaviors in a more realistic RTS environment that involves all the challenges such as resource gathering, fight, resource management and dynamic pathfinding.

The JaRTS modeling aimed the specification of a simple and intuitive architecture, which could reduce the development time of agents. The main component of this architecture is the *World class, which* represents an environment via components from the *Element* class. The *Unit* and *Terrain* classes extend the *Element* class and, while instances of the *Unit* class represent agents specified by users (Workers, Tanks and Control Centers); the Terrain class represents elements of the scenario such as *Obstacle* (places where agents cannot move across), *Resource* (objects to be collected by *Workers*), and *Plain Terrain* (place where agents can move over). Each *Unit* has an *Instruction* object that represents the target action. For example, the instruction *Mine* has as target a *Resource* representing the place where an agent will perform this instruction (or action).

The implementation of an agent's behavior is carried out in a similar way than in the Robocode simulator. Users just extend one of the unit classes (*Worker*, *Tank* or *ControlCenter*) to specify a particular behavior. For example, users can create the class *MyWorker* extending the class *Worker*. At simulation runtime, users can choose instances of classes that will connect to the simulation environment.

EVALUATION OF RTS TEAMS

The evaluation experiment has involved 65 students of the Intelligent Agents course in the Informatics Centre – UFPE. The final project was the implementation of multiagent teams that could run and be evaluated via JaRTS. This project was split up in two parts. First, groups of students should focus on approaches to the resource gathering problem (Figure 1). After that, the focus was on approaches to the tank battle problem.

After the conclusion of their projects, students should report the agents' issues/features that were evaluated via simulator. Furthermore, the reports were also important to clarify the approaches used for each team. The remainder of this section consolidates the information of such reports.

The first issue is related to *which information agents are able to sense and actions that they can perform*. In general, the teams have implemented sense routines to the following game features: obstacles location, resource locations, control center locations, enemy and fellow agents' locations, amount of resources available in a mine, amount of resources already collected, hit points of agents and control centers. In fact, the evaluation of such routines is not a complex task because JaRTS is an accessible domain. This means that agents have complete access to the environment features (e.g., location of mines or other agents), so that any kind of limitation rule was implemented (note that some strategic games have rules to sense limitation via, for example, the fog of war concept). Differently, the simulator is very useful to test patterns of behaviors (actions). Despite the fact that students have implemented the same basic actions, such as moving and resource gathering, the simulator can provide several configurations of environments (e.g., open fields or narrow roads) where such behaviors can be evaluated.

The second issue is related to *which constraints/variables teams must consider during the decision making process of agents,* or in other words, which information is required and appropriated to make a decision. For example, a pathfinding algorithm can initially be implemented to only consider the surrounding area. In this approach only the status of the nearby area is required. In other approach, such as a dynamic pathfinding, the position of other agents could also be used (e.g., to avoid enemies). In this context, the simulator clarifies if the sense of more information actually corresponds to more efficient actions.

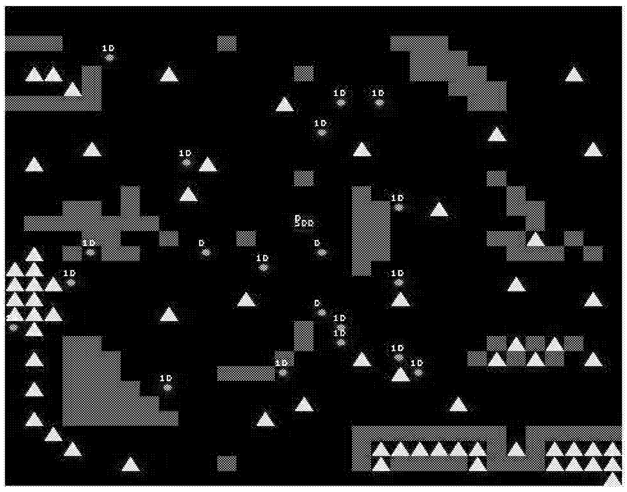


Figure 1: JaRTS scenario to the first problem, where triangles represent mines, circles represent agents that must collect minerals in such mines and squares represent static obstacles.

The third point is related to the *process of decision making* itself. Teams have implemented different strategies of reasoning and the simulator evaluates their performances in different situations. It is common to observe that a strategy does not have a good performance in all situations. Rather, there are particular strategies that are more appropriate to specific situations.

The fourth issue is related to the choice between *reactive and long term goals decisions*. These properties are classical extremes in Artificial Intelligence and the most common approach is a hybrid implementation. The application and evaluation of both approaches via simulator has assisted the teams in finding a balance between such extremes. For example, the moving to a control centre is a long term goal decision, while the avoidance of coalitions during this moving is a reactive behavior.

The fifth issue is about *communication*. In this case, the simulator has evaluated the performance of agents regarding the amount and content of information exchanged between agents. Note that the important point is not to exchange lots of information, but information with useful content. In this point the simulator helps to figure out which piece of information is in fact useful.

Three other issues were not very explored by the teams: approaches for *coordination* regarding techniques (planning, negotiation, etc.) and architectures (centralized, emerging or hybrid); use of *homogeneous versus heterogeneous agents* in terms of roles, permissions and goals; and the *influence of the team goal on each individual agent goal*. Such aspects will be analyzed in future directions of our work.

Note that a simulator only can carry out all these types of evaluations if it is configurable in terms of scenarios/maps and features. In this way, teams can set the simulator to provide a more appropriate situation to the kind of test that they intend to perform.

CONCLUSION

We can highlight some points about the evaluation of the students´ teams in both scenarios. In the first problem, resource gathering, we have noticed that the main tactic used by the teams was to collect mineral from the mines closer to the control centre. In this way, a significant issue of these approaches was a good specification of the pathfinding algorithm.

An interesting observation was that some teams had a very good performance in some of the game maps, however such performance did not come up again when other maps were employed. Thus we could clearly conclude that particular features of the environment, such as mines among obstacles, have significant impact on the algorithms because such features could configure “logical traps”. This means, situations where general algorithms cannot deal with in an efficient way. Such fact was important to stress the usefulness of RTS environments as benchmarks, which can realize faults and advantages of an algorithm in specific configurations and scenarios.

This issue is more critical in the second scenario. As happens in real world, an attack/defense strategy is strongly dependent on features of the environment (battlefield). For example, a combat strategy for an open battlefield cannot be used in a scenario with several obstacles (e.g., trees), which can make difficult the movement of military divisions. Again, the use of a RTS environment as benchmark enables the evaluation of strategies, such as the advantages and disadvantages that each strategy can offer for specific configurations of the environment.

Unfortunately the experiments have also stressed some criticisms previously discussed. The process of choosing maps, for example, has not used a proper methodology, so that the maps only cover a small part of the existent possibilities. In addition, the analysis of events is an exclusive task of evaluators, so that they own need to infer, for example, if a specific approach is (not) efficient in a specific configuration of the simulation scenario.

REFERENCES

Aha, D. and Molineaux, M. 2004. "Integrating learning in interactive gaming simulators". In D. Fu & J. Orkin (Eds.) Challenges in Game AI: Papers of the AAAI’04 Workshop. San José, CA: AAAI Press.

Boson. (2005). *Boson Homepage*. Available in http://boson.eu.org/ (accessed on 10 September 2007)

Buro, M. (2003). Real-Time Strategy Games: A new AI Research Challenge. *Proceedings of the 2003 International Joint Conference on Artificial Intelligence*, Acapulco, Mexico.

Glest. (2007). *Glest: A Free 3D RTS Project*. Available in http://www.glest.org/ (accessed on 10 September 2007)

Hanks, S., Pollack, M. and Cohen, P. (1993). Benchmarks, Testbeds, Controlled Experimentation, and the Design of Agent Architectures. *Technical Report 93-06-05*, Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA.

Kitano, H. and Tadokoro, S. (2001). RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI Magazine*, 22(1):39-52.

Li, S. (2002). Rock 'em, sock 'em Robocode!. *IBM developerWorks*, Available in http://www.ibm.com/ developerworks/java/library/j-robocode/ (accessed on 10 September 2007)

McDermott, D. 2000. The 1998 Planning Systems Competitions. *AI Magazine*, 4(2):115-129.

Ponsen, M., Lee-Urban, S., Muñoz-Avila, H., Aha, D. and Molineaux, M. 2005. “Stratagus: An open-source game engine for research in real-time strategy games”. Proceedings of the IJCAI Workshop Reasoning Representation, and Learning in Computer Games. Edinburgh, UK.

Stone, P. (2003). Multiagent Competitions and Resarch: Lessons from RoboCup and TAC. In Gal A. Kaminka, Pedro U. Lima, and Raul Rojas, editors, RoboCup-2002: Robot Soccer World Cup VI, *Lecture Notes in Artificial Intelligence*, pp. 224–237, Springer Verlag, Berlin.

Stratagus Team. (2007). *Stratagus: A real Time Strategy Engine*. Available in http://www.stratagus.org/ games.shtml (accessed on 10 September 2007)

Wellman, M., Wurman, P., O’Malley, K., Bangera, R., Lin, S., Reeves, D. e Wash, W. (2001). A Trading Agent Competition. *IEEE Internet Computing*, 5(2):43-51.

JaRTS Site. (2007). *JaRTS: A Java Real Time Strategy Engine*. Available in http://www.cin.ufpe.br /~vvf/jarts/ (accessed on 10 September 2007)

MAP-ADAPTIVE ARTIFICIAL INTELLIGENCE FOR VIDEO GAMES

Laurens van der Blom, Sander Bakkes and Pieter Spronck
Universiteit Maastricht
MICC-IKAT
P.O. Box 616
NL-6200 MD Maastricht
The Netherlands
e-mail: l.vanderblom@student.unimaas.nl, {s.bakkes,p.spronck}@micc.unimaas.nl

ABSTRACT

This paper proposes an approach to automatically adapt game AI to the environment of the game (i.e., the so-called map). In the approach, a particular map is first analysed for specific features. Subsequently, an automatically established decision tree is applied to adapt the game AI according to the features of the map. Experiments that test our approach are performed in the RTS game SPRING. From our results we may conclude that the approach can be used to automatically establish effective strategies dependent on the map of a game.

INTRODUCTION

Throughout the years, video games have become increasingly realistic with regard to visual and auditory presentation. However, artificial intelligence (AI) in games has not yet reached a high degree of realism. Typically, game AI is based on non-adaptive techniques [9], which prevents it to adequately adapt to changing circumstances. Adaptive game AI , on the other hand, has been explored with some success in previous research [2, 5, 8].
An important component of adaptive game AI is the ability to automatically establish effective behaviour dependent on features of the game environment (i.e., the so-called map). This is called 'map-adaptive game AI'. In this paper we will investigate how to analyse and exploit features of the game environment for the purpose of establishing effective game strategies.
The outline of the paper is as follows. We will first present our approach to establish map-adaptive game AI. Subsequently, the experiments that test our approach are discussed. The experimental results are discussed next. Finally, we provide conclusions and describe future work.

APPROACH

Our approach to establish map-adaptive game AI consists of three components: (1) definition of the features of a map, (2) determination of compound actions of map-adaptive strategies, and (3) automatic construction of a decision tree of map-adaptive strategies.
We establish map-adaptive game AI in an RTS game environment, i.e., a simulated war game. Here, a player needs to gather resources for the construction of units and buildings. The goal of the game is to defeat an enemy army in a real-time battle. We use RTS games for their highly challenging nature, which stems from three factors: (1) their high complexity, (2) the large amount of inherent uncertainty, and (3) the need for rapid decision making [1]. In the present research we use SPRING, illustrated in Figure 1, which is a typical and open-source RTS game. A SPRING game is won by the player who first destroys the opponent's 'Commander' unit.

Features of a Map

To automatically establish map-adaptive game AI, we start by defining a basic set of features that will play an essential role in the strategy of a player. For our experiments, we decided to use the following five features of a map.

1. RN: Number of metal resources.

Figure 1: Screenshot of the SPRING game environment. In the screenshot, the artillery unit on the left attacks a target across the river.

2. RD: Resource density.
3. NR: Presence of relatively narrow roads (e.g. due to obstacles such as mountains and rivers).
4. CL: Number of cliffs.
5. DE: Distance between base locations.

Feature values will be used to automatically construct a decision tree of map-adaptive strategies. If we would allow all possible feature values, an explosion in the number of nodes in the decision tree would occur. We therefore divide the range of feature values in bands [3], such as 'None', 'Few' and 'Many'. Naturally, game maps can vary in size. Therefore, feature values are scaled proportionally to the size of the map.

Map-Adaptive Game AI Actions

After analysis of features of a map, game AI is established on compound actions of map-adaptive strategies. For our experiments, we decided to use the following seven compound actions.

1. Construction of metal extractors at near metal resources.
2. Construction of metal extractors at far away metal resources.
3. Placement of offensive units at relatively narrow roads.
4. Placement of offensive units at own base.
5. Placement of artillery on cliffs.
6. Protection of operational metal extractors.
7. Protection of artillery on cliffs.

The defined actions can be used to establish offensive as well as defensive stances of game AI.

Decision Tree of Map-Adaptive Strategies

A decision tree is a tree where each internal node analyses a feature, each branch corresponds to a band of feature values, and each leaf node assigns a classification. Since we are dealing with discrete feature values, the decision tree is called a 'classification tree'. The leaves of such a tree represent classifications and the branches represent conjunctions of features that lead to those classifications. Classification trees also enable disjunctive descriptions of the features of the map.
In the Spring game, each feature of the map can be expressed by discrete values (e.g., the number of resources and the resource density). For constructing a decision tree for the Spring game, we employ the ID3 learning algorithm [4, 6].

The ID3 algorithm performs a simple-to-complex, hill-climbing search through the hypothesis space, which consists of all possible decision trees for the given features and their values. That is, the hypothesis space consists of all possible disjunctions of conjunctions of the features. The algorithm maintains only a single decision tree, performs no backtracking in its search and uses all training instances at each step of the search during the training process. Its evaluation function is the information gain, which is defined as

$$G(E,a) = I(E) - \sum_{v \in V_a} \frac{\mid E_{v,a} \mid}{\mid E \mid} I(E_{v,a}) \qquad (1)$$

where E is the set of all training examples and a is an attribute from the set of all features A. V_a is the set of values corresponding to feature a; that is, it is a set of values, such that

$$V_a = \{v \mid value(a,x) = v\} \qquad (2)$$

for all $x \in E$, where $value(a,x)$ defines the value for feature $a \in A$ of a specific example x. Moreover, $E_{v,a}$ is a subset of E, such that

$$E_{v,a} = \{x \in E \mid value(a,x) = v\} \qquad (3)$$

The function I is defined as

$$I(E) = \sum_{c \in C} -\frac{\mid E_c \mid}{\mid E \mid} \log_2 \frac{\mid E_c \mid}{\mid E \mid} \qquad (4)$$

where C is the set of all possible classifications (i.e., the actions that the game AI should perform) and E_c is a subset of E with classification c; that is, it is a set of training examples, such that

$$E_c = \{x \in E \mid class(x) = c\} \qquad (5)$$

for all $x \in E$, where $class(x)$ defines the classification of a specific example x.
The function I is also called the entropy, which measures the impurity of the set of all examples. In other words, information gain is the expected reduction in entropy caused by partitioning the instances according to a given attribute. This implies that the learning algorithm has a preference for short trees, with the features with a high information gain located near the root of the tree.

EXPERIMENTS

This section discusses the experiments that test our approach. We first describe the process of constructing the decision tree and then experimental setup.

Constructing the Decision Tree

We use the ID3 learning algorithm to construct the decision tree from experimentally determined training data, which consist of input data with values for attributes of the map and the corresponding target output data in the form of actions of the game AI. The training data is given in Table 2 (Appendix A). The learned decision tree is displayed in Figure 2 (Appendix B). We give two observations on the constructed decision tree.

First, the feature 'number of metal resources' is placed at the root of the tree. This implies that the number of metal resources is the most important feature when analysing a map. This result fits our expectation, for metal resources are vital to expanding the base and building the first units.

Second, if the number of metal resources is low, the constructed decision tree considers the 'resource density' as the next-most important feature of the map. If the number of metal resources is high, however, the resource density is the least important feature.

Experimental Setup

To test our approach, the map-adaptive game AI will be pitted in a game against the same game AI without map-adaptive capability. We found one game AI which was open source, which we labeled 'AAI' [7]. We enhanced this AI with the capability to adapt its behaviour dependent on the features of the map.

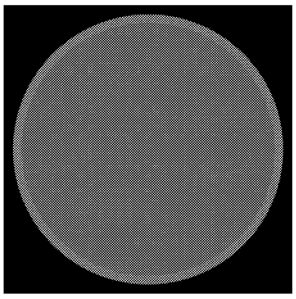
The map-adaptive game AI is tested in real-time games, which took place on five maps that are described in the next section, namely (1) Speed Ball, (2) Speed Ball Ring 8-way, (3) Mountain Range, (4) Small Supreme Battlefield v2, and (5) No Metal Speed Metal. Each experimental trial is repeated five times.

In addition to pitting the map-adaptive game AI against a computer opponent, our approach was also tested against a human opponent. Our expectation was that when pitting it against a superior human opponent, different behaviour will be evoked from the map-adaptive game AI.

RESULTS

In this section we provide a detailed discussion of the obtained results. For each map, we present the following three items: (1) the characteristics of the map, (2) the obtained results, and (3) a discussion of the obtained results.

Map 1: Speed Ball



The Speed Ball map has many resources and the players are always relatively close to each other. This allows us to determine whether the map-adaptive game AI attempts to protect its own units. In the black area nothing can be constructed. When two AI players compete against each other, they may become more offensive and attack each other early in the game. It is therefore expected that different AI behaviour will be observed when a relatively idle human player is involved as the second player.

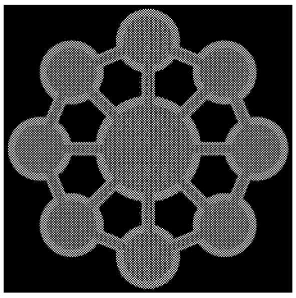
Gameplay on this map is focussed on effectively responding to the following features of the map: 'Number of Resources', 'Resource Density' and 'Distance between base locations'. Action '1+2,4,6' is expected to be executed, which corresponds to example X37 from Table 2 (Appendix A).

Obtained Results

We observed that the players did not focus on gathering resources and building an army when the map-adaptive game AI played against the computer-controlled player. Players became offensive early in the game and continued battling while constructing buildings and units (usually cavalry and artillery), which assisted them in defeating their opponent. This occurred in each of the five times that they played on this map.

When a human player was involved, the map-adaptive game AI behaved differently. One time the human player played offensively and successfully defeated the game AI, but in the meantime the game AI countered by also playing offensively. Another time the human player stayed in the background and remained relatively idle for a long period of time, only taking actions in order to defend himself. As a result, the game AI focused more on gathering resources before actually attacking its opponent.

Map 2: Speed Ball Ring 8-way



The Speed Ball Ring 8-way map has many similarities with the Speed Ball map. Instead of one big circle, however, this map is divided into eight equally sized circles and one larger circle in the middle. All of the circles are interconnected. The pathways that connect the circles are not very wide, which implies that they are considered as 'Narrow Roads'. There is a relatively large number of resources available on this map. Players are positioned randomly at the beginning of the game, thus the distance between the players can vary.

In this map we focus on all of the features other than 'Number of cliffs'. It is expected that actions '1+2,3' or '1+2,3,4,6' will be executed, corresponding to the examples X16 and X40 respectively, from Table 2.

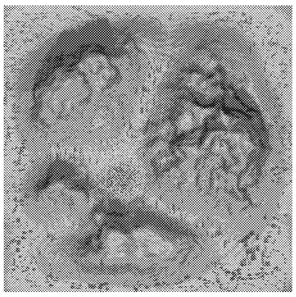
Obtained Results

When the map-adaptive game AI played against another computer-controlled player, the game AI often performed action '1+2,3', as expected. There was only one occurrence of action '1+2,3,4,6' being performed, when the game AI was in the middle circle and decided to place offensive units at the entrance of the pathways connecting to the other circles, while the other player was on one of the outer circles.

The map-adaptive game AI fared well against the human player with the same actions as expected, but again the human player had to remain inactive for the most part in order to provoke the game AI to exhibit behaviour that was expected of it. Otherwise, more offensive subroutines of the game AI would take over and battles occurred early in the game. There were two occasions where the game AI was located in the middle of the map, resulting in a different action, namely '1+2,3,4,6'.

Because starting positions of both players were random, classifications by the decision tree were also different each time we started a new game. This is caused in particular by the distance between both players, explaining the large difference in classifications.

Map 3: Mountain Range



The Mountain Range map does not have many metal resources. Additionally, the mountains are obstacles that obstruct the roads, making navigation relatively difficult. Each location for constructing a base on this map has its own advantages and disadvantages. The use of metal storages is recommended on this map, because of diminishing metal resources.

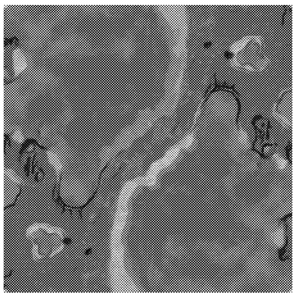
The map is relatively large. Therefore, the distance between the base of two players will typically remain large in a one-versus-one battle, as is the case here. The focus of the decision tree lies on the features 'Number of Metal Resources', 'Resource Density', and 'Narrow Roads'. We expected that the game AI would execute action '1,3', corresponding to example X4 from Table 2.

Obtained Results

We observed that on this map the map-adaptive game AI was strongly focussed on gathering nearby metal resources early in the game. Additionally, the game AI was blocking the passageways between the mountains and, if applicable, between the edges of the map and the mountains. The game AI protected the expanded position, and launched an attack on the opponent when it constructed a relatively large army. This strategy was consistently observed against both the computer-controlled as well as the human opponent.

We observed that early in the game, relatively few attacks would take place. This is caused by the relatively large distance between the base of each player. As lower-level subroutines will not be called for handling an attack, the map-adaptive game AI can focus on its first priority: establishing and expanding a strong base. Thus, we observed that in all cases action '1,3' was executed, as expected.

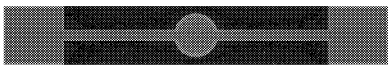
Map 4: Small Supreme Battlefield v2



The Small Supreme Battlefield v2 map contains a long bottleneck in the centre of the map. On each side of the map there is a large area of water. On the other two sides of the map there are small areas of land. The map has relatively few metal resources, some of which are available in the water areas. We expected that the game AI would execute action '1−2,3', corresponding to example X10, or '1−2,3,4,6', corresponding to example X34, from Table 2.

Obtained Results
When in competition against the computer-controlled opponent, the map-adaptive game AI had a preference for executing action '1−2,3'. In most of the cases the game AI blocked the bottleneck with offensive units. In other cases the focus of the game was not so much on the bottleneck, but more on the water area, on which the battle continued by sea units and submarine units. The computer-controlled player always built its base on land located in the corner of the map, which implied that the distance between the bases remained fairly large.
Identical behaviour was observed when in competition against a human player. On one occasion the human player constructed the base nearby the base of the map-adaptive game AI. This led the AI to increasingly protect its base and the metal extractors by use of offensive units.

Map 5: No Metal Speed Metal



The No Metal Speed Metal map is based on the Speed Metal map. As the name implies, however, this particular map does not contain any metal resources. The lack of metal resources makes it difficult to quickly produce units and expand the base. A challenge for the established map-adaptive game AI was that it was not trained under circumstances where no metal resources were present on the map.
It was expected that the game AI would choose action '1−2,3', corresponding to example X10, or '1−2,3,5', corresponding to example X11 and X12, from Table 2. Actions '1−2,3,4,6', corresponding to example X34, and '1−2,3,5,6,7', corresponding to examples X35 and X36, were considered alternative possibilities.

Obtained Results
In competition against the computer-controlled game AI, both players focused on constructing offensive units to oppose their opponent. In addition, the map-adaptive game AI utilised units for protecting the entrance of its own area. Similar behaviour was observed when competing against the human player. In one case, the human player constructed the base in the centre of the map, near the map-adaptive game AI. This led to a different classifaction, and thus different behaviour from the map-adaptive game AI.
Though the map-adaptive game AI was not trained for circumstances where no metal resources are present, it was able to utilise the learned decision tree by traversing the node for 'few metal resources'. However, it did not exhibit behaviour suitable for defeating the computer-controlled player.
A summary of the experimental results is provided in Table 1.

DISCUSSION

Our approach to map-adaptive game AI should not be confused with machine-learning techniques that allow the game AI to adapt to novel situations. Rather, in our approach, we implemented the map-adaptive game AI as a high-level planner of strategic actions. We allowed low-level actions, such as handling an imminent threat of the opponent, to interfere with the established high-level plan.
In a typical RTS game, early phases of the game are focussed on planning the construction and expansion of the base. Later phases of the game are typically focussed

	Computer	**Human**
Map 1		
Speed Ball		
Classification	X37 (100%)	X37 (100%)
Win:Loss	3:2	0:5
Map 2		
Speed Ball Ring 8-way		
Classification	X16 (80%)	X16 (60%)
	X40 (20%)	X40 (40%)
Win:Loss	4:1	0:5
Map 3		
Mountain Range		
Classification	X4 (100%)	X4 (100%)
Win:Loss	4:1	0:5
Map 4		
Small Supreme Battlefield v2		
Classification	X10 (100%)	X10 (60%)
		X34 (40%)
Win:Loss	3:2	0:5
Map 5		
No Metal Speed Metal		
Classification	X10 (100%)	X10 (80%)
		X34 (20%)
Win:Loss	0:5	0:5

Table 1: Classifications of the map-adaptive game AI in competition against the computer-controlled player and against the human player.

on engaging in offensive or defensive actions. However, if an opponent would decide to attack relatively early, a player would be forced to abandon the established plan and focus on combat. Therefore, our implementation of map-adaptive game AI as a high-level planner of strategic actions, is particularly suitable for RTS games.
For games from other genres, our implementation of map-adaptive game AI may not necessarily be the most suitable one. Game developers should consider that to apply our approach in practice, a balance should be found between pursuing a high-level map-adaptive plan, and allowing the game AI to respond to low-level actions.

CONCLUSIONS AND FUTURE WORK

In this paper we proposed an approach for establishing map-adaptive game AI. In our approach the game environment, in the form a so-called map, is first analysed for specific features. Subsequently, a decision tree of map-adaptive strategies is constructed automatically. Experiments to test our approach were performed in the RTS game Spring. Our experimental results show that against a computer-controlled opponent, the map-adaptive game AI consistently constructed effective game strategies. The map-adaptive game AI was outperformed by a human opponent. However, observations showed that the map-adaptive game AI responded to strong play of the human player by adapting its own strategy. From these results, we may conclude that the established map-adaptive game AI can be successfully used to construct effective strategies in RTS games.
In future work, we will incorporate a self-adaptive mechanism to enable the game AI to automatically refine the constructed decision tree. This mechanism will be particularly suitable for games where effective map-adaptive game AI should be established online, on the basis of relatively little training data.

ACKNOWLEDGEMENTS

We extend our gratitude to the anonymous reviewers for their insightful feedback.
This research was funded by a grant from the Netherlands Organization for Scientific Research (NWO grant No 612.066.406).

REFERENCES

[1] Michael Buro and Timothy Furtak. RTS games and real-time AI research. In *Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2004.

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

[3] Roger Evans. Varieties of Learning. In S. Rabin, editor, *AI Game Programming Wisdom*, pages 571–575. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, United States, 2002.

[4] Daniel Fu and Ryan Houlette. Constructing a Decision Tree Based on Past Experience. In S. Rabin, editor, *AI Game Programming Wisdom 2*, pages 567–577. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, United States, 2004.

[5] Thore Graepel, Ralf Herbrich, and Julian Gold. Learning to fight. In Quasim Mehdi, Norman Gough, and David Al-Dabass, editors, *Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*.

[6] Tom Mitchell. *Machine Learning*, chapter 3: Decision Tree Learning, pages 52–80. McGraw-Hill, 2 Penn Plaza, New York 10121-0101, United States, 1997.

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

[8] Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma. Online adaptation of game opponent AI with dynamic scripting. *International Journal of Intelligent Games and Simulation*, 3(1):45–53, 2004.

[9] Paul Tozour. *AI Game Programming Wisdom (ed. Rabin, S.)*, chapter The Perils of AI Scripting, pages 541–547. Charles River Media, 2002. ISBN: 1-58450-077-8.

A - TRAINING DATA

Example	Attributes					Action(s)
	RN	RD	NR	CL	DE	
X1	Few	High	No	None	Far	1
X2	Few	High	No	Few	Far	1
X3	Few	High	No	Many	Far	1,5
X4	Few	High	Yes	None	Far	1,3
X5	Few	High	Yes	Few	Far	1,3
X6	Few	High	Yes	Many	Far	1,3,5
X7	Few	Low	No	None	Far	1−2
X8	Few	Low	No	Few	Far	1−2
X9	Few	Low	No	Many	Far	1−2,5
X10	Few	Low	Yes	None	Far	1−2,3
X11	Few	Low	Yes	Few	Far	1−2,3,5
X12	Few	Low	Yes	Many	Far	1−2,3,5
X13	Many	High	No	None	Far	1+2
X14	Many	High	No	Few	Far	1+2
X15	Many	High	No	Many	Far	1+2,5
X16	Many	High	Yes	None	Far	1+2,3
X17	Many	High	Yes	Few	Far	1+2,3
X18	Many	High	Yes	Many	Far	1+2,3,5
X19	Many	Low	No	None	Far	1+2
X20	Many	Low	No	Few	Far	1+2
X21	Many	Low	No	Many	Far	1+2,5
X22	Many	Low	Yes	None	Far	1+2,3
X23	Many	Low	Yes	Few	Far	1+2,3,5
X24	Many	Low	Yes	Many	Far	1+2,3,5
X25	Few	High	No	None	Close	1,4,6
X26	Few	High	No	Few	Close	1,4,6
X27	Few	High	No	Many	Close	1,5,6,7
X28	Few	High	Yes	None	Close	1,3,4,6
X29	Few	High	Yes	Few	Close	1,3,4,6
X30	Few	High	Yes	Many	Close	1,3,5,6,7
X31	Few	Low	No	None	Close	1−2,4,6
X32	Few	Low	No	Few	Close	1−2,4,6
X33	Few	Low	No	Many	Close	1−2,5,6,7
X34	Few	Low	Yes	None	Close	1−2,3,4,6
X35	Few	Low	Yes	Few	Close	1−2,3,5,6,7
X36	Few	Low	Yes	Many	Close	1−2,3,5,6,7
X37	Many	High	No	None	Close	1+2,4,6
X38	Many	High	No	Few	Close	1+2,4,6
X39	Many	High	No	Many	Close	1+2,5,6,7
X40	Many	High	Yes	None	Close	1+2,3,4,6
X41	Many	High	Yes	Few	Close	1+2,3,4,6
X42	Many	High	Yes	Many	Close	1+2,3,5,6,7
X43	Many	Low	No	None	Close	1+2,4,6
X44	Many	Low	No	Few	Close	1+2,4,6
X45	Many	Low	No	Many	Close	1+2,5,6,7
X46	Many	Low	Yes	None	Close	1+2,3,4,6
X47	Many	Low	Yes	Few	Close	1+2,3,5,6,7
X48	Many	Low	Yes	May	Close	1+2,3,5,6,7

Table 2: Training data of the Spring real-time strategy game with respect to features of the map. Legend: "–" means first executing the action before the dash sign, then the action after the dash sign. "+" means executing the actions on both sides of the plus sign simultaneously. "RN" means number of metal resources. "RD" means resource density. "NR" means presence of relatively narrow roads (e.g. due to obstacles such as mountains and rivers). "CL" means number of cliffs. "DE" means distance between base locations.

B - DECISION TREE FOR MAP-ADAPTIVE GAME AI

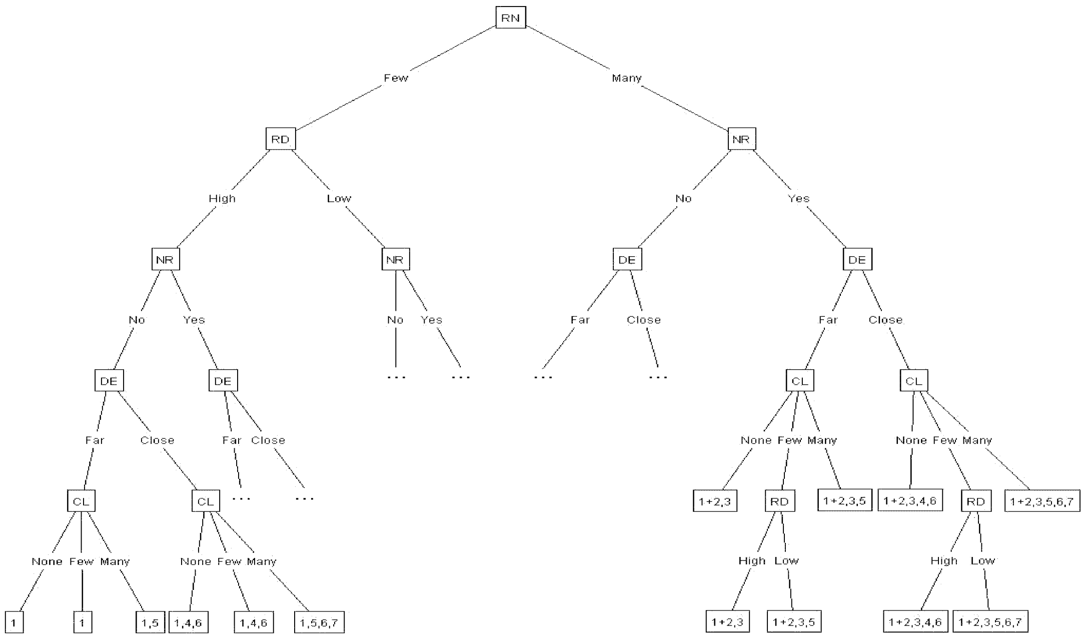


Figure 2: The automatically constructed decision tree. Only a portion of the tree is shown. Legend: "RN" means number of metal resources. "RD" means resource density. "NR" means presence of relatively narrow roads (e.g. due to obstacles such as mountains and rivers). "CL" means number of cliffs. "DE" means distance between base locations.

OPPONENT MODELING IN REAL-TIME STRATEGY GAMES

Frederik Schadd, Sander Bakkes and Pieter Spronck
Universiteit Maastricht
MICC-IKAT
P.O. Box 616
NL-6200 MD Maastricht
The Netherlands
e-mail: f.schadd@student.unimaas.nl, {s.bakkes,p.spronck}@micc.unimaas.nl

ABSTRACT

Real-time strategy games present an environment in which game AI is expected to behave realistically. One feature of realistic behaviour in game AI is the ability to recognise the strategy of the opponent player. This is known as opponent modeling. In this paper, we propose an approach of opponent modeling based on hierarchically structured models. The top-level of the hierarchy can classify the general play style of the opponent. The bottom-level of the hierarchy can classify specific strategies that further define the opponent's behaviour. Experiments that test the approach are performed in the RTS game Spring. From our results we may conclude that the approach can be successfully used to classify the strategy of an opponent in the Spring game.

INTRODUCTION

In computer gaming, real-time strategy (RTS) is a genre of simulated wargames which take place in real time. In RTS games, the player needs to construct a base and build units for the purpose of destroying the opponent. The opponent is either a human player, or a player controlled by an artificial intelligence (AI). Each unit-type has particular strengths and weaknesses. To effectively play an RTS game, the player has to utilise the right units in the right circumstances.

An important factor that influences the choice of strategy, is the strategy of the opponent. For instance, if one knows what types of units the opponent has, then typically one would choose to build units that are strong against those of the opponent. To make predictions about the opponent's strategy, an AI player can establish an opponent model. Many researchers point out the importance of modelling the opponent's strategy [2, 3, 9, 10, 12, 14], and state that opponent models are sorely needed to deal with the complexities of state-of-the-art video games [8].

Establishing effective opponent models in RTS games, however, is a particular challenge because of the lack of perfect information of the game environment. In classical board games the entire board is visible to the player; a player can observe all the actions of the opponent. Hence, assessing the opponent's strategy and building an opponent model is possible in principle, for instance by using case-based reasoning techniques [1]. In RTS games, however, the player has to deal with imperfect information [5]. Typically, the player can only observe the game map within a certain visibility range of its own units. This renders constructing opponent models in an RTS game a difficult task. In this paper we will investigate to what extent models of the opponent's strategy can be established in an imperfect-information RTS-game environment.

The outline of this paper is as follows. We will first introduce the concept of opponent modeling. Then, our approach to establish effective opponent models in RTS games will be discussed. Subsequently, our implementation of the approach will be presented. The experiments that test our approach are described next, followed by a discussion of the experimental results. Finally, we provide conclusions and describe future work.

OPPONENT MODELING

In general, an opponent model is an abstracted description of a player or a player's behaviour in a game [8]. Opponent modeling can be seen as a classification problem, where data that is collected during the game is classified as one of the available opponent models. A limiting condition is the fact that in RTS games, these classifications have to be performed in real-time, while many other computations, such as rendering the game graphics, have to be performed in parallel. This limits the amount of available computing resources, which is why only computationally-inexpensive techniques are suitable for opponent modeling in RTS games.

Preference-based modeling is a commonly used computationally-inexpensive technique [4]. The technique identifies the model of an opponent by analyzing the opponent's choices in important game states. Due to the visibility limitations in RTS games, however, it is common that choices of the opponent cannot always be observed.

In the present research we use Spring, illustrated in

Figure 1: Screenshot of the SPRING game environment. In the screenshot, airplane units are flying over the terrain.

Figure 1, which is a typical and open-source RTS game. A SPRING game is won by the player who first destroys the opponent's 'Commander' unit. We used the freely-available 'AAI' artificial intelligence player [15] to combat opponents.

APPROACH

A straightforward approach to opponent modeling is the following. First, a number of possible opponent models is established, then the confidence level of each opponent model is calculated, and finally, the opponent model with the highest confidence level is selected. An enhancement of this approach is to apply an hierarchical ordering on the possible opponent models [7]. This *hierarchical approach* allows the division of a relatively complex classification problem, into several relatively simple classification problems. In addition, the hierarchical approach makes it possible to use different classification methods in each level or the hierarchy. For establishing opponent modeling in RTS games, we follow the hierarchical approach.

Our opponent models will describe the strategy of a player. We define a strategy as *the general play style combined with the player's choice of units built.* The most defining element of an opponent's strategy is the general play style. We therefore place the general play style at the top of the hierarchy. Each play style has its own subcategories that further define behavioural characteristics.

For instance, if it is known that the opponent follows an aggressive general play style, a logical response would be to improve one's defenses. If also the opponent's choice of units is known, the defenses can be specialised to be effective against those specific units.

IMPLEMENTATION

This section discusses our implementation of hierarchical opponent modeling in RTS games. In our implementation, we establish a hierarchy consisting of two levels. The top-level of the hierarchy classifies the opponent's general play style. The bottom-level of the hierarchy classifies the opponent's choice of units built.

In the SPRING game we discriminate between an aggressive, and a defensive play style. For an aggressive play style we discriminate at the bottom level between predominantly using the following four unit-types: (1) K-Bots, (2) Tanks, (3) Ships, and (4) Airplanes. Each unit-type has specific strengths and weaknesses, and is therefore used to execute a particular strategy. For instance, K-Bots are relatively fragile but can cross mountains, and are therefore useful for a strategy against an opponent which attempts to exploit chokepoints between mountains. Tanks can only manoeuvre on plain terrain but are relatively sturdy, and are therefore useful for a strategy against an opponent who constructs strong defenses.

For a defensive play style we discriminate at the bottom level between the following three building preferences: (1) Super Weapon, (2) Tech, and (3) Bunker. These three building preferences are commonly observed in actual SPRING games.

Figure 2 displays the hierarchy of the opponent models. The hierarchy defines the following strategies.

- Aggressive→K-Bot. The opponent will attack early and will typically use K-Bot robots.
- Aggressive→Tanks. The opponent will attack early and will typically use tanks.
- Aggressive→Air. The opponent will attack early and will typically use airplanes.

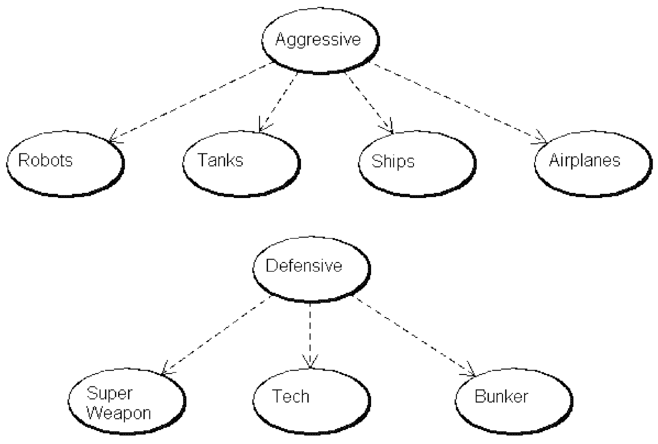


Figure 2: Hierarchy of the opponent models.

- Aggressive→Sea. The opponent will attack early and will typically use ships.
- Defensive→Super weapon. The opponent will attempt to construct a super weapon (e.g., a ballistic missile).
- Defensive→Tech. The opponent will attempt to reach a high technology level in order to have quick access to superior units.
- Defensive→Bunker. The opponent will construct a massive wall of static defenses around his base so that he has time to construct an army.

Top-level Classifier

A way of performing opponent-model classifications in a computationally inexpensive fashion, is by using fuzzy models [16]. Fuzzy models create models of several classifiable classes based on a single numerical feature. The choice of the numerical feature is crucial, as it should allow to discriminate between the defined classes.

In our hierarchy, the top level classifier has to discriminate between the classes 'aggressive' and 'defensive'. An aggressive player typically will spend a large part of game time attacking. A defensive player, on the other hand, typically will use most of the game time for preparing an army, and only needs a small amount of the game time for an actual attack. As a result, an appropriate numerical feature to discriminate between the two classes would be the relative amount of game time that the opponent spends attacking.

We define an attack as the observed loss of a player's own units. When a loss of units is detected, then the time span around this moment can be regarded as the time that an attack took place. Because information about a player's own units is always available, this definition is suitable for use in an imperfect information environment.

An example of a top-level player model is shown in Figure 3. The figure illustrates the confidence of the opponent following an aggressive and defensive play style, as a function of the percentage of game time spent on attacking.

Bottom-level Classifier

The bottom-level classifier has to discriminate between the subcategories that further define behavioural characteristics. The dynamic nature of RTS games implies that a player's strategy may change during the game. Therefore, the bottom-level classifier needs to emphasize recent event more than past events. To achieve this, the principle of discounted rewards is applied.

Theoretical Background

The concept of discounted rewards origins from the principle of repeated matrix-games [6]. A player can make a choice between several actions and depending on the effectiveness of an action, a reward is received. In the field of repeated matrix-games, the most important considerations are how to select the initial strategy and how to evaluate if a deviation from the initial strategy is needed. A strategy in repeated matrix-games can be a simple sequence of actions, which is played repeatedly. It can also include certain rules for cases that take the actions of the enemy into consideration. In order to determine whether a deviation from the original strategy is feasible, the expected rewards of the game with and without deviation are calculated. Here the discount factor is applied, since it is assumed that a player prefers to receive a reward early in time rather than in the distant future. Hence, rewards in the future are valued less. This valuation is expressed by multiplying the future reward with the discount factor. The more distant the reward is, the more often it is multiplied with the discount factor. If δ is the discount factor and π_t is the reward received at time t, then the expected reward can be computed as follows [6]:

$$(1-\delta) * \sum_{i=1}^{\infty} \pi_i * \delta^{i-1} \qquad (1)$$

When the expected rewards are calculated, a selection mechanism typically will select the strategy with the highest expected reward.

Applied Modification

Analogously to calculating the expected rewards, for discriminating between strategies we need to calculate the confidence value of the opponent applying a particular strategy. The confidence value is calculated on the basis of observations during game events. In classical matrix-games, a game event would be one move of both players.

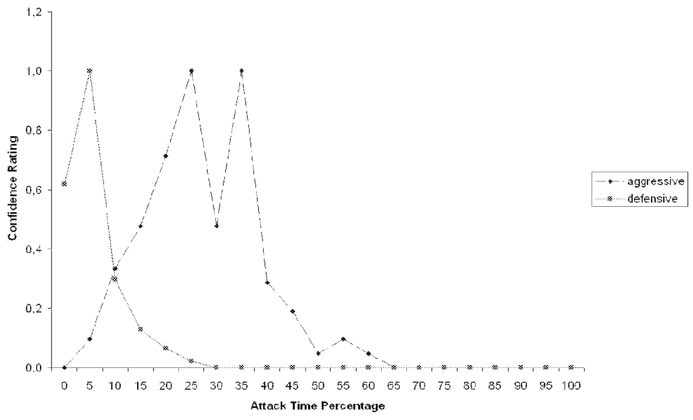


Figure 3: Example of a top-level player model.

Since RTS games do not operate in moves, the term of an event must be redefined. When playing against an aggressive opponent, the occurrence of an attack is a suitable event since the player can then observe the army of the opponent.
When playing against a defensive opponent, however, an attack is not a suitable event because a defensive opponent will rarely attack and hence not many observations can be made. Also, waiting for an attack of a defensive opponent is typically an unsuccessful game strategy. When playing against a defensive opponent, it is typical that the player will 'scout' around the base of the opponent. Since scouting will provide improved information about the opponent's actions, a scout event is a suitable moment for calculating confidence values against a defensive opponent.
When an event is detected, the confidence values of each possible strategy will be updated according to the observed game information. If δ is the discount factor, $\psi_{s,t}$ the belief that the opponent uses strategy s at event t, ranging between 0 and 1, π the total reward added at each event and i the most recent event, then the confidence c_s that the opponent uses strategy s is computed as follows:

$$c_s = \sum_{t=i}^{0} \psi_{s,t} * \pi * \delta^{i-t} \tag{2}$$

The parameter $\psi_{s,t}$ is acquired by inspecting all visible units and structures during event t. Each unit or structure has a value representing a tendency to a certain strategy. The unit-tendency values were determined by the experimenter, using his own knowledge of the game [13]. To give three examples of unit-tendency values: (1) a common defensive tower has a relatively small tendency towards an opponent using the defensive bunkering strategy, (2) a super-weapon building has a relatively high tendency towards the defensive-super-weapon strategy, and (3) an amphibious tank has a tendency towards both the aggressive tank and the aggressive sea strategy.

EXPERIMENTS

This section discusses the experiments that test our approach. First we test the top-level classifier, and then the bottom-level classifier. We finish the section by providing a summary of the experimental results.

Top-level Experiment

As discussed in the previous section, the top-level classifier requires the AI to detect if an attack took place. This is implemented as follows. The AI will register all visible units every N seconds, where N is the size of the time window. The detection algorithm will scan each frame for lost units. If the amount of lost units is above a certain threshold, then each frame inside the analysed time window is labeled as a moment in which the opponent was attacking.
Two parameters determine the accuracy of the attack-detection algorithm. The first parameter is the size of the time window. The second parameter is the unit threshold, which is a percentage value. We will first analyse the sensitivity of the parameter settings on the obtained confidence values. Next, we will discuss the obtained confidence values as a function over the game time.

	Time Window		
Threshold	20 seconds	30 seconds	40 seconds
10%	94,92166	99,2271	93,64321
15%	99,99083	99,57988	96,22472

Table 1: Average defensive confidence values against a defensive opponent

	Time Window		
Threshold	20 seconds	30 seconds	40 seconds
10%	45.10261	76.27533	57.83488
15%	33.18209	52.74819	53.14834

Table 2: Average aggressive confidence values against an aggressive opponent

Sensitivity Analysis
For each configuration of parameters, the AI was matched ten times against an aggressively playing AI and ten times against a defensively playing AI. As opponent the 'NTAI'-AI [11] was chosen, since it is relatively straightforward to implement one's own strategy into this AI. For the matches where an aggressive opponent is required, the 'NTAI'-AI was configured with an aggressive strategy. Analogously, for the matches where a defensive opponent is required, the AI was configured with a defensive strategy. For the time window, the sizes of 20, 30 and 40 seconds were tested. For the unit threshold the percentages with value 10% and 15% were tested.
For each match played, the aggressive and defensive confidence values were recorded at each time point. When the match was played, the average confidence values of the match were calculated. Note that as in the first five minutes of a game a player rarely performs an action that would reveal his strategy, these first minutes were not taken into account.
Table 1 shows the defensive confidence values of matches against an defensive opponent, and Table 2 shows the aggressive confidence values of matches against an aggressive opponent.
The obtained confidence values reveal that all configurations of the top-level classifier perform well when

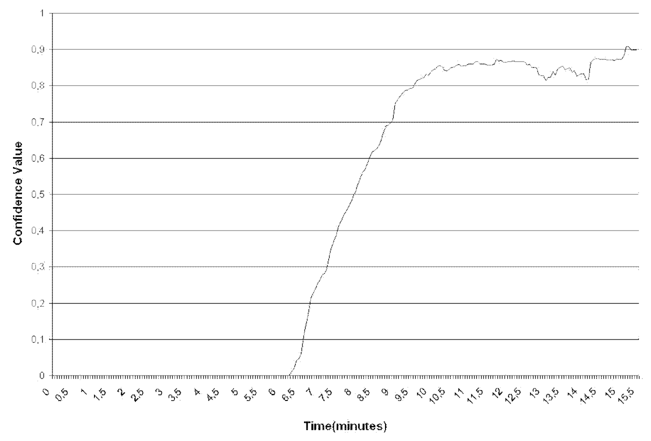


Figure 4: Average confidence value over time against an aggressive opponent

recognizing defensive players. The best configuration obtained a confidence value of nearly 100%. When recognizing aggressive players, the obtained confidence values are lower in comparison, as will be elaborated upon shortly. The best configuration, with as parameters a units-lost threshold of 10% and a time window of 30 seconds, obtained a confidence value of 76%. These obtained configurations will be used for the remainder of this research.

Confidence Value over Time

Using the obtained parameter values, we match the AI fifty times against an aggressive opponent, and fifty times against a defensive opponent. We again used the 'NTAI'-AI as an opponent. For each game, we record the confidence values as a function over time. The average confidence values of all test games against an aggressive opponent are displayed in Figure 4.

In the figure we observe that the average confidence value is low in the beginning of the game. This is due to the fact that the opponent is hardly able to attack at this stage of the game, since he needs to construct a base first. Therefore, one can safely disregard the confidence values of the beginning of the game. After approximately seven minutes of game time, the average confidence value increases until it stabilizes at approximately 85%.

A similar effect can be observed when examining the average confidence value over time of the games against a defensive opponent, as displayed in Figure 5. In the beginning of the game, the confidence values are nearly 100%. This is because the enemy does not attack in the beginning of the game. The top-level classifier will therefore respond with the maximum defensive confidence value during this game stage. One observes that after about six minutes of game time, the average confidence value stabilizes between 96% and 97%.

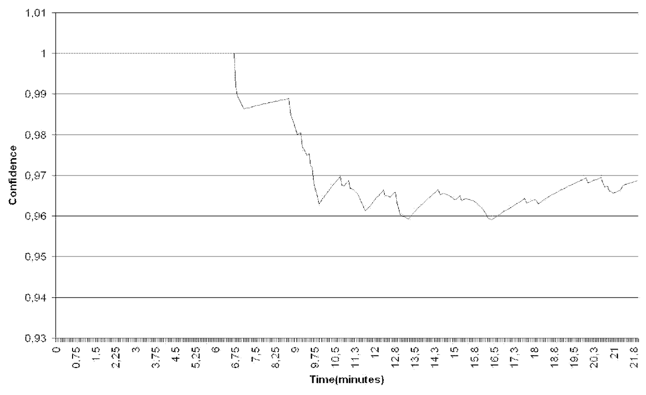


Figure 5: Average confidence value over time against a defensive opponent

Bottom-level Experiment

For testing the bottom-level classifier, the 'NTAI'-AI has been configured such that it resembles each of the specific aggressive opponent models. However, since NTAI is not able to adequately play the defensive strategies, a human opponent was chosen to play against the 'AAI'-AI [15], by following the defensive strategies. For each opponent model, ten experimental matches have been performed. A correct classification of the top-level classifier is assumed for this experiment. For the discounted reward algorithm, the parameters were set to $\delta = 20\%$ and $\pi = 0.8$ by the experimenter. We test the bottom-level classifier's ability to discriminate between the K-Bot and tanks aggressive sub-model, and between the bunker, tech and super weapon defensive sub-model.

Aggressive Opponent

Figure 6 shows the average K-Bot confidence over time of an opponent using the aggressive K-Bot strategy as well as the average Tank confidence over time of an opponent using the aggressive tank strategy. It is observed that both confidence values eventually approximate a value over 90%. We note that the average confidence of the aggressive tank-strategy increases more slowly and at a later stage than the average confidence of the aggressive K-Bot-strategy. This can be explained by the fact that producing tanks requires more resources, and therefore more game time will be needed to attack with tanks.

Defensive Opponent - Bunker

Figure 7 displays the confidence values obtained against an opponent using the defensive bunker-strategy. We observe that the bunker confidence rating increases rapidly after approximately five minutes of game-time. Over time the obtained average confidence value is 83%. The instabilities that occur after 35 minutes of game-time can be explained by the fact that at this moment the AI has discovered structures that may also be used

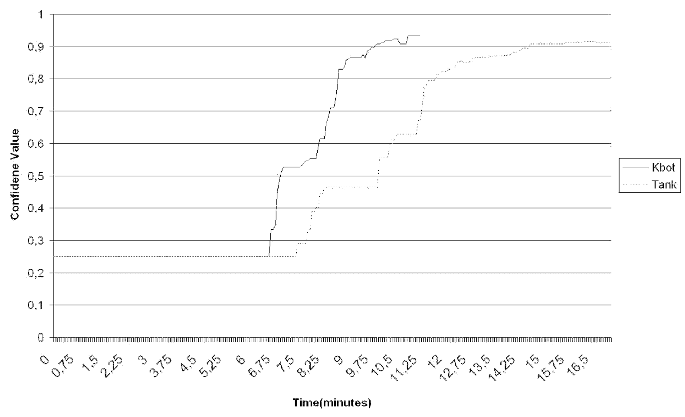


Figure 6: Average confidence value over time for the aggressive sub-models.

by an opponent using the tech-strategy.

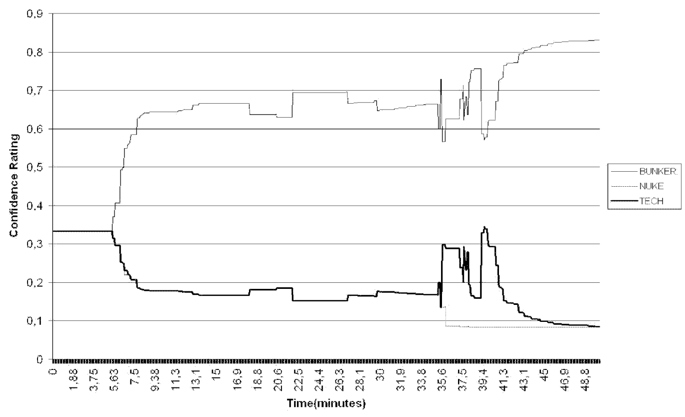


Figure 7: Average confidence value over time for an opponent using the defensive bunker-strategy.

Defensive Opponent - Tech

Figure 8 displays the confidence values obtained against an opponent using the defensive tech-strategy. We observe that for the largest part of the game, the confidence values of the bunker-strategy are higher than the confidence values of the tech-strategy. This can be explained as follows. First, we observed that scout units were destroyed before they were able to scout the base of the opponent. Second, high level units and structures that define the tech-strategy can only be constructed in later stages of the game. This implies that in the earlier stages only structures that belong to a different strategy can be observed. When at a later stage of the game the AI is able to observe structures that are characteristic for the tech-strategy, the confidence value of the tech-strategy increases.

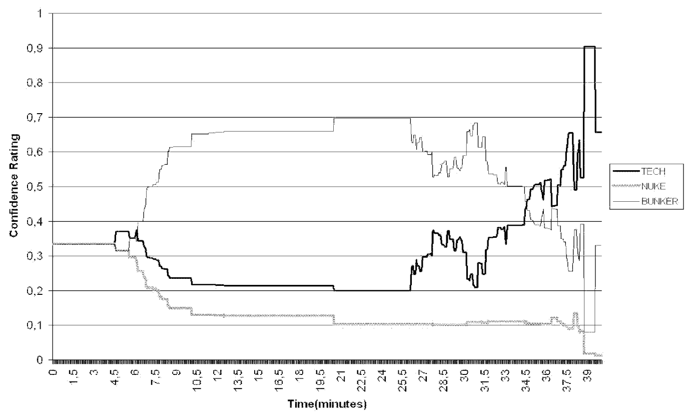


Figure 8: Average confidence value over time for an opponent using the defensive tech-strategy.

Defensive Opponent - Super Weapon
Figure 9 displays the confidence values obtained against an opponent using the defensive super-weapon-strategy. Analogously to the results obtained against an opponent using the defensive tech-strategy, we observe that the confidence values of the bunker-strategy are higher than the confidence values of the super-weapon-strategy. After approximately 25 minutes the confidence value of super-weapon-strategy steadily increases. After approximately 41 minutes, the discounted-reward algorithm temporarily decreased the confidence value because the AI did not observe super-weapon structures any more. Eventually the bottom-level classifier obtains a confidence value of 75%.

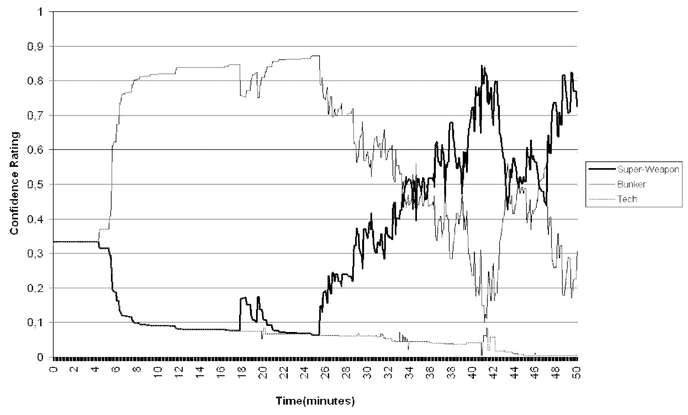


Figure 9: Average confidence value over time for an opponent using the defensive super-weapon-strategy.

Summary of the Experimental Results

Experimental results obtained with the top-level classifier show that the top-level classifier can accurately discriminate between an aggressive and a defensive player. Experimental results obtained with the bottom-level classifier show that the bottom-level classifier can accurately discriminate between the established sub-models in later stages of the game. In early stages of the game, the bottom-level classifier was not always able to accurately discriminate between the established sub-models. This is discussed next.

DISCUSSION

Opponent modeling will typically be implemented in an actual RTS game for the purpose of automatically classifying the strategy of the opponent. Ideally, an accurate classification of the opponent's strategy is available relatively early in the game, at a time when the player is still able to counter the opponent's strategy.
In the experiments that test our approach, we observed that the bottom-level classifier was not always able to accurately discriminate between the established sub-models in an early stage of the game. This phenomenon can be explained by the fact that, typically, the AI cannot directly observe units and structures that are characteristic for a particular bottom-level strategy. To observe these units and structures, the AI relies on scouting.
A straightforward approach to achieve improved results, therefore, is to adapt the AI's scouting behaviour dependent on the need for information of the opponent's activities. For instance, in competition against an aggressive opponent, scouting is relatively unimportant. In competition against a defensive opponent, however, intensive scouting is vital. Analogously, to emphasise the information obtained during a scout event, one may choose to adapt the parameters of the delayed reward algorithm dependent on the top-level classification.

CONCLUSION

In this paper we proposed an approach for opponent modeling in RTS games. In the approach, a hierarchical opponent model of the opponent's strategy is established. The top-level of the hierarchy can classify the general play style of the opponent. The bottom-level of the hierarchy can classify strategies that further define behavioural characteristics of the opponent. Experiments to test the approach were performing in the RTS game Spring. Our experimental results show that the general play style can accurately be classified by the top-level of the hierarchy. Additionally, experimental results obtained with the bottom-level of the hierarchy show that in early stages of the game it is difficult to obtain accurate classifications. In later stages of the game, however, the bottom-level of the hierarchy will accurately classify between specific strategies of the opponent. From these results, we may conclude that the approach for opponent modeling in RTS can be successfully used to classify the strategy of the opponent while the game is still in progress.

For future work, we will incorporate a mechanism to adapt the scouting behaviour of the AI dependent on the top-level classification of the general play style of the opponent. Subsequently, we will investigate to what extent the classification of the opponent's strategy can be used to improve the performance of a computer-controlled player, and will investigate how new models to describe the opponent's strategy can be established automatically.

ACKNOWLEDGEMENTS

The authors wish to extend their gratitude to Marcel Ludwig and Alexander Miesen for being the human opponent in the experimental games. Furthermore, we extend our gratitude to the anonymous reviewers for their insightful feedback.
This research was funded by a grant from the Netherlands Organization for Scientific Research (NWO grant No 612.066.406).

REFERENCES

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

[2] Bruce Abramson. Expected outcome: A general model of static evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:182–193, 1990.

[3] Hans Berliner. Search and knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 77)*, pages 975–979, 1977.

[4] Jeroen Donkers and Pieter Spronck. Preference-based player modeling. In S. Rabin, editor, *AI Programming Wisdom 3*, chapter 8.4, pages 647–659. Charles River Media, 25 Thomson Place, Boston, Massachusetts 02210, 2006.

[5] Robert Gibbons. *A Primer in Game Theory*, chapter 2.3B. Pearson Education Limited, Edingburgh Gate, Harlow, Essex CM20 2JE, England, 1992.

[6] Robert Gibbons. *A Primer in Game Theory*, chapter 2. Pearson Education Limited, Edingburgh Gate, Harlow, Essex CM20 2JE, England, 1992.

[7] Ryan Houlete. Player modeling for adaptive games. In S. Rabin, editor, *AI Programming Wisdom 2*, chapter 10.1, pages 557–566. Charles River Media, 10 Downer Avenue, Hingham, Massachusetts 02043, 2004.

[8] Jaap van den Herik, Jeroen Donkers, and Pieter Spronck. Opponent modelling and commercial

games. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (eds. Graham Kendall and Simon Lucas)*, pages 15–25, 2005.

[9] Donald Knuth and Ronald Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[10] Richard Korf. Generalized game trees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 89)*, pages 328–333. Detroit, MI, August 1989.

[11] Tom Nowell. AI:NTAI. Creator of the game AI 'NTAI', http://spring.clan-sy.com/wiki/AI:NTAI, 2007.

[12] Arthur Samuel. Some studies in machine learning using the game of chechers, ii - recent progres. *IBM Journal*, 11:601–617, 1967.

[13] Frederik Schadd. Hierarchical opponent models for real-time strategy games. Bachelor thesis. Universiteit Maastricht, The Netherlands, 2007.

[14] Jonathan Schaeffer, Joseph Culberson, Norman Treloar, Brent Knight, Paul Lu, and Duane Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53:273–289, 1992.

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

[16] Michael Zarozinski. An open-fuzzy logic library. In S. Rabin, editor, *AI Programming Wisdom*, chapter 2.8, pages 90–101. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, 2002.

ART DESIGN AND GRAPHICS

PANORAMA – EXPLORATIONS IN THE AESTHETICS OF SOCIAL AWARENESS

Anton Eliëns
Intelligent Multimedia Group
VU University Amsterdam
email: eliens@cs.vu.nl

Dhaval Vyas
Human Media Interaction Group
Twente University
email: D.M.Vyas@ewi.utwente.nl

KEYWORDS
social awareness, visual semotics, interaction aesthetics, video games

ABSTRACT

In this paper we reflect on our experiences in developing PANORAMA, a playful application meant to promote and support social awareness in a work environment, through art-inspired visualisations of social processes and personal contributions. With respect to the design of PANORAMA, we found common notions of visual semiotics helpful in determining the overall composition of the screen layout. More in general, however, the development of PANORAMA proved to be an exercise in *interaction aesthetics*, which as we will argue in this paper may greatly benefit from common notions in interactive video game play. In this paper we will only briefly discuss technical and deployment issues, since our main contribution here is to establish the relation between the aesthetics of interaction and game play.

INTRODUCTION

Since the 1970's, Dutch universities have enormously grown in size, due to the ever larger number of students that aim at having university level education. As departments become bigger, however, staff members no longer know eachother personally. The impersonal and anonymous atmosphere is increasingly an issue of concern for the management, and various initiatives have been taken, including collective trips into nature, as well as cultural events, not to much avail for that matter. An additional problem is that more and more members of the staff come from different countries and cultures, often only with a temporal contract and residence permit. Yet, during their stay, they also have the desire to communicate and learn about the other staff members and their culture(s).

In september 2006, the idea came up to use a large screen display in one of the public spaces in our department, to present, one way or another, the 'liveliness' of the work place, and to look for ways that staff members might communicate directly or indirectly with eachother through this display. Observing that communications often took place during casual encounters at the coffee machine or printer, we decided that monitoring the interactions at such places might give a clue about the *liveliness* of the work place. In addition, we noted that the door and one of the walls in the room where the coffee machine stood, was used by staff members to display personal items, such as birth announcement cards or sport trophees. In that same room, mostly during lunch time, staff members also gathered to play cards.

Taking these observations together, we decided to develop a system, nicknamed PANORAMA, to present these ongoing activities and interactions on a large display, which was to be positioned in the coffee room. The name of our system is derived from the famous Mesdag Panorama[1] in The Hague, which gives a view on (even in that time nostalgic rendering of) Scheveningen. However, it was explicitly not our intention to give an in any sense realistic/naturalistic remdering of the work place, but rather, inspired by artistic interpretations of *panoramic* applications as presented in Grau (2003), to find a more art-ful way of visualizing the social structure and dynamics of the work place.

At this stage, about one year later, we have a running prototype (implemented in DirectX), for which we did perform a preliminary field study as well as a first user evaluation, Vyas et al. (2007), and also we have experimented with a light-weight web-based variant, allowing access from the desktop. In this paper, our primary focus, however, will be to establish the relation between interaction aesthetics and game play, as inspired by our experiences in developing PANORAMA.

structure The structure of this paper is as follows. First we will give a brief sketch of the PANORAMA system, that is the ideas underlying it and the realization of a first prototype. Then we will present, in a more general fashion, possible guidelines for the design of PANORAMA, followed by a discussion of common issues in interaction aesthetics and game play. We will then introduce the notion of *dialectics of awareness*, and in conclusion identify the primary dimensions of aesthetic experience.

[1] www.panorama-mesdag.nl

BEING SOCIAL @ WORK

The PANORAMA system is meant to support social awareness, in non-work related ways, using a large screen display in a public room in our faculty. To achieve social awareness, we ask the staff to contribute items of *self-reflection*, such as holiday postcards or birth announcements. In order to reflect the liveliness of the workplace, we monitor places where *occasional encounters* may take place, for example during a break at the coffee machine or in the printer room, waiting for the printer queue. Encounters in such places are often of an informal, personal nature, but may be mixed with work-related interests. As an experimental feature, we consider to allow for direct interaction using the system, for example, to play a game, possibly with a mobile phone as an input device. In summary, the PANORAMA system is determined by the following contributions of its users, contributions that are not necessarily direct or even do require explicit activity.

- self-reflection(s) – e.g. picture/postcard(s)
- casual encounter(s) – at coffee machine or printer
- occasional battle(s) – optional direct interaction

For a deeper understanding of what role the system would play in the (working) life of the staff, we engaged in several field studies and used *cultural probes* to determine what could be valuable contributions to ask for and how to display these on the PANORAMA screen, Vyas et al. (2007b).
We have developed a first prototype implementation in DirectX 9, using ViP technology, based on the system described in Eliens (2006). In this realization, we deploy a moving virtual gallery, containing video and image feeds. The gallery acts like a moving scroll, displaying information in a continuous manner, in a panorama-like way. The images in the gallery are fed by channels, containing information that is either due to explicit contributions (self-reflections) or ongoing activity in the work place (casual encounters or occuring events), monitored by cameras or other sensors.
Obviously, as we will discuss later, the PANORAMA system is subject to a dialectic of awareness, that is it will be present, but the staff will only occasionally pay atention to it, dependent on their interests and also on what visual cues and effects the system presents to draw attention to ongoing activity. Although we would like the system to be autonomous in the decision how to present information, we cannot hope to do this by computational means only, Eliens (1988), and hence we need to provide interaction markers to invite the users to contribute actively to the system, or influence the way information is displayed according to their preference.
For the display of information, we provide a rich context of material, including videos showing the faculty and its surroundings, fragments from video clips, and of course the material resulting from the occassional encounters and self reflections. In PANORAMA we use particle systems displaying the information in a pictorial way by images flowing according to the rules of the particle system chosen to represent that particular type of information. To organize this material we took conventions governing our interpretation of 2D displays as a guideline in designing the flow of particle systems. A more detailed discussion of these conventions will be given in the next section. Identifying *bottom* with *plain*, *top* with *ideal*, *left* with *given* and *right* with *new*, we arrived at the following identifications.

semiotic rules

- self reflections: plain $\Longrightarrow$ ideal/new
- casual encounters: plain $\Longrightarrow$ ideal/given
- contextual stories: ideal/given $\Longrightarrow$ plain/new
- personell faces: ideal/new $\Longrightarrow$ plain/given
- occurring events: ideal $\Longrightarrow$ plain

For example one may remark that people's faces become more familar in time, and that in the process of getting to know them we see more of the plain reality of people. Naturally, different interpretations and different designs are possible.
Apart from the spatial characteristics of these flows of information we also used the speed with which the images move accross the screen as a parameter of design. For example events and occurrences move very fast, while both casual encounters and self reflections move slowly. Faces come across the screen with intermediate speed. To give self reflections more visual salience, the images are displayed in a non-transparent way, whereas all other flows of images merge with the background due to transparency. Although it is debatable whether the interpretations given above hold, we found the heuristics given by semiotic theory extremely helpful in deciding how to represent the information as flows of images in space/time.

MEANING OF COMPOSITION

Aesthetic awareness is common to us all. Having an understanding of aesthetic awareness, can we isolate the relevant design parameters and formulate rules of composition that may help us in developing interactive applications? According to our philosophical credo, Eliens (1979), no! However, the history of art clearly shows the impact of discoveries, such as the discovery of perspective, as well as conventions in the interpretation of art, as for example in the iconic representation of narrative context in 17th century Dutch painting. Moreover, the analysis of the visual culture of mass media may also give us better understanding of the implied meaning of compositional structures.
The notion of *perspective*, described in Alberti (1435), is an interesting notion in itself, since it describes both the organisation of the image as well as the optimal point of

view of the viewer. The normal perspective as we know it is the central perspective. However, there are variants of perspective that force the viewer in an abnormal point of view, as for example with anamorphisms.

Perspective had an enormous impact on (western) art and visual culture. It defines our notion of naturalist realism, and allowed for the development of the panorama as a mass medium of the 19th century, Grau (2003). Art that deviated from central perspective, such as cubism or art from other cultures, was often considered naive. Photography and its pre-cursors had a great impact on the perfection of perspectivist naturalism, and what is called *photorealism* became the touchstone of perfection for early computer graphics, Bolter and Grusin (2000).

Apart from perspective, other conventions regulate the composition of the 2D image, in particular, following Kress and van Leeuwen (1996), the *information value* related to where an object is placed in the image, and the *salience* of the object, determined by its relative size, being foreground or background, and visual contrast. Also *framing* is used to emphasize meaning, as for example in the close-up in a movie shot. In analysing a large collection of image material, Kress and van Leeuwen (1996), somewhat surprisingly found that *lef/right* positioning usually meant *given* versus *new*, *top/bottom* positioning *ideal* versus *real*, and *centre/margin* positioning *important* versus *marginal*. It is doubtful whether these meaning relationships hold in all cultures, but as a visual convention it is apparently well-rooted in western visual culture.

For 2D images, Kress and van Leeuwen (1996) further identify narrative elements, that is relations between objects in the image that suggest a story, such as a diagonal line from a person to a door, or a relation of an object to the viewer, such as a gaze towards the viewer, a technique that has been used only since late renaissance painting.

More than paintings or 2D images, film is the medium for conveying narrative structures. The art of storytelling in film has been perfected in such a way that Hollywood films may seem more real than life. However, as emphasized in Bolter and Grusin (2000), this is not due to any inherent form of naturalism, but to the fact that we have got accustomed to the conventions applied, that is the techniques of cutting, montage, camera movements, close-ups, etcetera. In a highly recommended book, Arnheim (1957), Rudolf Arnheim gives an extensive analysis of the principles of montage and film technique, and he explains why film is such an effective medium:

> It is one of the most important formal qualities of film that every object that is reproduced appears simultaneously in two entirely different frames of reference, namely the two-dimensional and the three-dimensional, and that as one identical object it fulfills two different functions in the two contexts.

Due to the subtle play between these two *frames of reference* film may be considered an art form, and as such perhaps the dominant art form of the 20th century. As a mass medium, film may be characterized by what Arnheim, following Benjamin, called the *aesthetics of shock*, replacing reflective distance with immersive thrill. As an art form, however, it is the dominant paradigm for aesthetic awareness, lacking however still one dimension, *interactive dynamics*.

As observed in Bolter and Grusin (2000), interaction is what distinguishes video games from film. Current day technology allows for high-resolution photorealist graphics, that make video games or virtual applications almost indistinguishable from film. Virtual reality technology as applied in video games adds arbitrary choice of perspective, as exemplified in first-person shooters or fly-overs, as well as an arbitrary mix of the imaginary and real, as in CG movies, in an interactive fashion.

Now, should we take the aesthetics of interactive video games as the standard for interactive applications? Not necessarily, since the naturalism strived for in most games may at best be characterized as naive realism, mostly photorealism. As observed in Kress and van Leeuwen (1996), realism is a social construct, and hence the program for developing an aesthetics for interactive applications should perhaps include the development of appropriate *realisms*. Again with an eye to the history of art, where we have for example *impressionism*, *cubism*, *expressionism*, as a guideline in the design of interactive systems, it might be even better to look for appropriate interaction-*isms*, styles of developing interactive systems and games from a particular perspective. Not excluding provocative perspectives! Cf. Burger (1981).

AESTHETICS OF GAME PLAY

Where an arbitrary interactive system may differ from a game played for entertainment is obviously the actual outcome, that is the value attributed to using the system in the real world, and probably the effort required and the possible consequences. You would not like to run the risk to die a virtual death when answering your email, would you? However, when interactive systems replace task-bound functionality with fun, the difference becomes less clear.

As we indicate in Eliens & Chang (2007), one element not sufficiently captured by a classic game model, as introduced in Juul (2005), is the narrative aspect of the game play.

We may observe that many games already have a strong relation to reality in what narrative context they supply, or else in the realities of the media industry, in particular Hollywood. For *serious* interactive systems, we may assume an even stronger and in some sense more straightforward relation with reality, by the use of media content that is relevant for the life of the individual.

interaction markers Where in game playing the variety of interaction modes seems to be well understood within each community of game players, for the development of more general interactive systems we will have to think seriously whether the target user will be able to learn the various modes of interaction, either by explicit instruction or during play. And as designers we must be concerned with the *rules of interaction* as well as issues of visualisation and interaction mappings, that is in other words which affordances the application offers for a particular group of users.

DIALECTICS OF AWARENESS

In the course of our field study for the *PANORAMA* system, Vyas et al. (2007), we tried to establish what relation users would have to the system, not only in the way they interact with it, but also in terms of what role the system plays in their lives, and when and how they would be aware of the system.
Due to the intrinsic properties of the *PANORAMA* system, as a system meant to support social awareness in a work environment, we could not assume direct focussed attention. Instead, we must take the various forms of awareness or attention into account.
Our thoughts in this direction were triggered by a lecture of Linda Stone (former vice-president of Microsoft) at the Crossmedia Week[2] September 2006 in Amsterdam, entitled *Attention – the Real Aphrodisiac*. In that lecture Linda Stone made a distinction between applications popular before 1985, applications which were in general meant for self-improvement, for example language-learning, applications that were popular between 1985 and 2005, applications that she characterized as supporting *continuous partial awareness*, such as email and news-feeds, and applications of the period thereafter, from now into the future, which may be characterized as applications that allow the user to be creative, take part in a community, and are in other words more focussed and less dependent on the external environment.
Admittedly, it takes a few more steps to formulate a theory of the *dialectics of awareness*. However, with the function of the *PANORAMA* system in mind, we may make, following Benjamin (1936), some interesting distinctions between the experience of art and architecture. Where art is usually experienced in a delimited time span, and is similarly delimited in space, that is the position of the observer, architecture is everywhere and always there. As a consequence, art receives focussed attention and may be appreciated with reflective distance, whereas architecture is often not perceived consciously, but merely present and subject to an almost sub-conscious sensibility, which is only brought to the focus of attention when it is either aesthetisized, for example when taking photographs, or when something surprising is sensed, for example in the change of skyline in New York.
As argued in Hallnäss and Redström (2002), many of the new interactive systems, whether in the category of *ambient media*, *ubiquitous computing* or *calm technology*, will fall somewhere inbetween the spectrum spanned by art and architecture, or more likely even alternate between the forms of awareness associated with respectively art and architecture.
In designing the new interactive systems and games, we need to be explicitly concerned with the actual phases of awareness that occur, simply because it is not clear what role these systems play in our life. When introducing a new system or artefact, we may distinguish between the following phases:

phases of awareness

- *initiation* – appeal to curiosity
- *promotion* – raising interest
- *progression* – prolonged involvement

As designers we must ask ourselves the following questions. *How do we appeal to the users' curiosity, so that our system is noticed? How do we get a more sustained interest? How de we get the user to interact with or contribute to the system? And, how do we obtain prolonged involvement, and avoid boredom?* These questions are not simple to answer, and require also an understanding of the actual context in which the system is deployed as well as an understanding of the level of (aesthetic) literacy of the user(s).

AESTHETIC EXPERIENCE

In Hallnäss and Redström (2002) the notion of *expressional* is introduced, to convey the expressive meaning of objects, that is the shift of attention from *use* to *presence*, subject to a dialectic process of appearance and gradual disappearance, when objects gradually become more familiar. For the *design of presence*, *aesthetics* is then considered as a *logic of expressions*, in which *expressions* act as *the presentation of a structure in a given space of design variables.*
However appealing the notion of *expressional*, following idealist aesthetics, Kant (1781), where a distinction is made between aesthetic awareness as a given, or a priori, sensibility and aesthetic judgement as being of a more empirical nature, we would prefer to consider *aesthetics* as a *logic of sensibility*, which includes a dimension of self-reflection in the sense of its being aware of its own history. Put differently, to characterize the contextual aspect of aesthetics, as it certainly applies to art, we may speak of *aesthetic literacy*, that is aesthetic awareness that is self-reflective by nature.

[2]www.picnic06.org

Assuming a notion of aesthetics as a *logic of sensibility*, we may distinguish between three dimensions of *form*, extending Kant's original proposal, as indicated below:

dimensions of aesthetic experience

- spatial – topological relations, layout of image
- temporal – order, rhythm, structure
- dynamic – interaction, reflection, involvement

The dimension of *dynamics* clearly is the great unknown, and more in particular it is the dimension we have to explore in the context of interactive systems, not in isolation but in relation to the other dimensions, not so much to establish definite criteria, but to understand the forces at work, or in other words the relevant parameters of design. Sartre (1936) gives an existential foundation for the dimension of *dynamics*, by observing that the human body is instrumental in gaining awareness, as the *centre of both obscurity and reflection from which consciousness emerges*, through selection and action. It is in the existential dimension of aesthetic awareness that we come most close to the experience of the new digital artefacts, since it concerns both involvement and human action.

CONCLUSIONS

The PANORAMA system, as presented in this paper, may be regarded as one of the new interactive systems, with game playing – in the form of *occasional battle(s)* – as an intrinsic element. PANORAMA, and similar systems alike, presents us not only with a technical challenge, but more importantly also with a design challenge, which requires a new way of looking at the aesthetics of interaction, or perhaps we should say the meaning of such systems in our day to day experience, amplifying our awareness.
As our initial prototype was received with much interest, we see as important targets for future research, firstly the deployment of alternative platforms, Si & Eliens (2007), and secondly the development of suitable games, that fit within the aesthetic framework determined by the primary *raison d'être* for PANORAMA, to promote and support social awareness.

REFERENCES

Alberti L.B. (1435), *On painting and on sculpture*, Phaidon, edited by C. Grayson, 1972

Arnheim R. (1957), *Film as Art*, University of California Press

Benjamin W. (1936), *The Work of Art in the Age of Mechanical Reproduction*, Online Archive – http://www.marxists.org

Bolter J.D and Grusin R. (2000), *Remediation – Understanding New Media*, MIT Press

Burger P. (1981), *Theorie der Avantgarde*, Edition Suhrkamp

Eliens A. (1979), Creativity: reflection and involvement, Ninth Int Conf of Aesthetics, Dubrovnic, August 1979

Eliens A. (1988), Computational Art, First Int Symposium on Electronic Art, Leonardo Supplementary Issue, Pergamon Press 1988, pp. 21-26

Eliens A. (2006), Odyssee – explorations in mixed reality theatre, In Proc. GAME'ON-NA 2006, P. McDowell ed., Eurosis-ETI, pp. 62-64, Sept 19-21, Monterey, USA

Eliens A. and Chang T. (2007), Let's be serious – ICT is not a (simple) game, In *Proc. FUBUTEC 2007*, April 2007, Delft

Grau O. (2003), *Virtual Art – From Illusion to Immersion*, MIT Press

Graves-Petersen M., Iversen O.S., Gall Krogh P., Ludvigsen M. (2004), Aesthetic Interaction – A Pragmatist's Aesthetics of Interactive Systems, Proc. DIS2004, Cambridge Mass. USA, pp. 269-276

Hallnäss L. and Redström J. (2002), From Use to Presence: On the Expressions and Aesthetics of Everyday Computational Things, ACM Trans. on Computer-Human Interactions, Vol. 9, No 2, pp. 106-124

Juul J. (2005), *Half Real – Video Games between Real Rules and Fictional Worlds*, MIT Press

Kant E. (1781), *Kritik der reinen Vernunft*, Felix Meiner Verlag, ed. 1976

Kress G. and van Leeuwen T. (1996), *Reading Images: The Grammar of Visual Design*, Routledge

Sartre J.P. (1936), *L'imagination*, Presse Universitaire de France, ed. 8, 1971

Si Yin and Eliens A. (2007), PANORAMA – A Rich VRML Application Platform For Online Gaming, Workshop Web 3D Games, Web3D Symposium 07, Perugia Italy, April 15-18

Vyas D., van de Watering M., Eliens A., van der Veer G. (2007), Engineering Social Awareness in Work Environments, Proc. HCI International 2007, 22-27 July, Beijing, China

Vyas D. van de Watering M., Eliens A., van der Veer G. (2007b), *Being Social @ Work: Designing for Playfully Mediated Social Awareness in Work*, Proc. *HOIT 2007*, Chennai, India in August 2007

ISSUES FOR MULTIPLAYER MOBILE GAME ENGINES

Abhishek Rawat
Indian Institute of Technology
Kanpur – 208016
Uttar Pradesh, India
E-mail: rawata@iitk.ac.in

Michel Simatic
GET — INT
9, rue Charles Fourier
91011 Evry Cedex, France
E-mail: Michel.Simatic@int-edu.eu

KEYWORDS

Game, Game engine, Mobile, Development.

ABSTRACT

This study analyzes the different functionalities of Multiplayer Mobile Game (MMG) Engine- that is game engines dedicated to development of multiplayer games on mobile phones. The different functionalities are broadly divided into- core functionalities and extra functionalities. The core functionalities are necessary and essential for MMG development and have been identified as- networking, graphics and animation, inputs management, context awareness, sound and data management. The extra functionalities on the other hand deal with non gaming aspects such as portability issues, deployment and development tools. We illustrate how these functionalities are provided by some of the available MMG engines. The concerned MMG engines are *Edgelib*, *GASP* and *MUPE*. We also throw light on the remaining issues for future MMG engines. This paper concludes by listing key issues related to MMG engines: portability, context awareness, monolithic aspect of current MMG engines, networking, and deployment. Portability has limited research perspective. Other issues offer attractive research opportunities for further development of MMG engines.

INTRODUCTION

Within few years, games have become more and more popular on mobile phones. Nevertheless, currently most of available games correspond only to the porting of games existing on PCs, video game consoles (*e.g.* XBOX 360), or handheld game consoles (*e.g.* PSP). In particular, games do not take advantage of mobile phone specificities: always on and always with user, networking capabilities, etc (Costikyan 2005).
In order to help game developers in their game development, game engines provide a suite of visual development tools in addition to reusable software components. This paper presents the different functionalities of MMG engines. First it introduces the three available engines which we studied in order to illustrate these functionalities: *Edgelib*, *GASP* and *MUPE*. Then it analyzes the core functionalities of MMG engines. Afterwards it considers their extra-functionalities. Finally it concludes by listing key issues.

STUDIED MMG ENGINES

In order to illustrate the different functionalities of MMG engines, we chose three MMG engines among the available ones: *Edgelib*, *GASP* and *MUPE*. We selected *Edgelib* as it is a game engine dedicated to C++ (thus high-performance) games, *GASP* because we have a lot of experience on it, and finally *MUPE* for its interesting handling of context-awareness.
The methodology involved in our study was primarily theoretical. First of all, a thorough study of the concerned platform's documentation was done. This was followed by experiments in order to further evaluate each platform. The experiments involved playing some already existing games (which were developed using the concerned platform) and making simple test games.
The following paragraphs briefly introduce each MMG engine.
Edgelib is a commercial[1] multi-platform game engine for mobile devices (Elements Interactive B.V. 2007). It features a platform independent API, enabling the developer to make use of all of its features for all supported platforms. Games are written in C++. *Edgelib* is the engine itself, containing all the modules and the interface APIs. An *Edgelib*-powered game hooks up on *Edgelib* using the entry and wrapper functions to take advantage of the functionalities and gameloop from *Edgelib*.
GAming Services Platform is a common project of CNAM — CEDRIC and GET — INT laboratories (Pellerin et al. 2007; 2005). Its result is *GASP*- an Open Source Java middleware for the development of multiplayer games for Java Micro Edition (J2ME) phones based on Connected Limited Device Configuration (CLDC) profiles, such as Mobile Information Device Profile (MIDP) and NTTDocomo Java (Doja). This middleware implements the Open Mobile Alliance Games Services (OMA GS) specifications v1.0 (Open

[1]The license for *Edgelib*'s full edition costs 950$ per year (it entitles developer to create and release as many commercial games and applications as desired).

Mobile Alliance 2007).
Multi User Publishing Environment (*MUPE*) is an Open Source application platform for rapid development of mobile multi-user context-aware applications, games and services (MUPE 2007). *MUPE* application framework mainly contains: *MUPE* client, *MUPE* core, and *MUPE* server. *MUPE* client (a J2ME MIDlet) contains all phone related functionalities and extension plugins. *MUPE* core deals with client-server communication, external context information and other utilities. *MUPE* server provides the game/service logic and a simple virtual world.

MMG ENGINES CORE FUNCTIONALITIES

While studying *Edgelib*, *GASP* and *MUPE* engines, we identified several core functionalities, that is functionalities which are directly taking care of some gaming aspects. In the following subsections, we study each one of these functionalities, illustrate them with these MMG engines and introduce remaining issues.

Networking

A cellular networks environment is fundamentally different from a fixed networks environment. The development of MMGs offers more challenge than their fixed/desktop counterparts (multiplayer computer games) mainly because of more limited resources. Indeed resource limitations compensation techniques of multiplayer computer games (Smed and Hakonen 2006) can be reused: message compression and aggregation, dead reckoning, area-of-interest filtering, etc. But these algorithms have to be adapted to take into account cellular network specificities: latencies around 1 second and data transfers charged with a pay-per-byte policy (Nokia Forum 2003). The use of Bluetooth improves the situation (although it is still worse than a fixed network environment), but then the number of players becomes limited (typically to a maximum of eight players). Moreover players must be physically located in a 10 meter radius area.
Cellular network (like 2G or 3G) experience two other constraints:

- Even though IP APIs (*e.g.*, TCP sockets) are provided by the mobile phone, they cannot be used in many countries because these protocols are restricted to HTTP communications by the operator.
- Connectivity is frequently intermittent: The MMG should be playable even in the absence of network connectivity. Indeed this constraint has an influence on the game design[2]. For instance, some games are designed so that players do not consider disconnection periods as system malfunction, but rather as a special game situation, which may even be advantageous for a certain period (Broll et al. 2006). Notice that this connectivity constraint, as well as the latency problem, motivates the necessity for data consistency sub-functionality (module). Such a module guarantees a consistent view of the game data for all players even when exchanged messages between gamers mobiles experience high latencies or when some players are temporarily not playing

Illustration
GASP enables to create MMGs running over cellular networks and over Bluetooth. In the former case, because of protocol restrictions enforced by operators, HTTP protocol is the basis of client-server communication: i) Game events are stacked on the server side, ii) Clients send periodic requests to the server in order to get these events.
To limit volume of data transfers, *GASP* relies on a light-weight binarization of message objects provided by MooDS protocol (Pellerin 2007). In the context of multiplayer gaming experiment, MooDS data is about 1.2 times longer than raw data length values, whereas JSR-172 and kSOAP2 are respectively about 12.2 times and 14 times longer. Moreover generation of binarization code is designed to scale down communication application code size on mobile side. As an example, this compilation mode was successfully used on a game prototype to reduce code size by 8 kilobytes, representing about 25% of the total application code size in Doja 1.5.

Issues
Despite Wi-Fi or other wireless technologies deployment, latency will remain a problem in the few coming years. MMG architectures mixing Bluetooth networks and 2G/3G networks may be a palliative solution.
With the increase in number of MMGs, the data consistency sub-functionality will play an important role in dealing with the latency problem (somehow it is a way of hiding latency). Integrating it easily in the game development process is an issue (Khan et al. 2007).
In-game communications between users will be more and more required: easy communication with one another through chat messages or, even better, by speaking. Network libraries based on SIP are currently being studied. SIP is indeed a neat solution to this requirement. But it is not sufficient, as SIP-based phones will not be commonly deployed among users before a few years. Other solutions should be studied.
Finally fairness is also an issue. Its goal is to guarantee that all players have the same chances of winning, even if some of them have a less powerful mobile phone or a

[2]Notice game design solutions to this problem may also solve constraints related to highly interruptible nature of MMG sessions (The mobile devices are mainly intended for person to person communications, and thus phone calls and SMS messages can constantly interrupt the MMG being played).

slower access to the network. Solutions have been proposed in the context of PC/consoles, but studies should be done in order to find solutions appropriate for mobile environments.

Graphics and animation

Graphics are used to represent the characters and creatures in a game, as well as background worlds and other interesting objects that factor into the overall game design. Graphic functions must be provided to game developers. These functions include drawing of text, basic shapes (lines, rectangles, etc) and displaying/rendering of images on the screen. Transparency is an important aspect of graphics. Without transparency, superimposed images would have distinct visible rectangular borders, which would make the game less realistic.
Animation is the illusion of movement. It is the heart of graphics in almost all mobile games (Morrison 2005). Since mobile phones have limited resources, it is important that when determining the frame rate for a game, it should be low enough to yield a smooth animation and at the same time it should not overload the processor. Even though in recent years the graphics power of the mobile phones have increased a lot (making it possible to develop high performance 3D games with eye catching graphics), for most of the existing mobiles, 2D animations are still more straightforward and efficient than 3D animation and thus are more suited[3]. Sprites are important in games because they provide simple, yet effective means of conveying movement while also enabling the objects to interact with one another. Indeed an important aspect of sprite animation is collision detection: It is used to determine whether sprites physically interact with one another. This is a critical requirement of virtually every action game.
Lastly it is desirable for game developers to have tools to help screen design: MMGs generally require the screen area to represent a large amount of information related to game world and players. The small screen size poses strong challenges to designers, as experienced for instance by TibiaME developers (Nokia 2003).

Illustration
Edgelib contains a graphics engine, optimized for various platforms. It provides functionality to draw both 2D and 3D graphics. It supports color key blitting[4] and optimized opacity blitting, RGBA surfaces, different display orientation (both portrait and landscape layout), rotation in all four orientations (0, 90, 180 and 270), surface rotation and scaling, dynamic clipping, collision detection, native image loading, custom pixel shaders and various optimized color filters. It also supports a broad range of image formats: BMP, PNG, etc. *Edgelib* is equipped with native image loaders making the engine fully independent from external libraries. These image loaders are implemented as plugins and so unused image formats can be disabled to decrease the filesize of executables.
3D surfaces are available to store 3D models. *Edgelib* provides support for hardware accelerated 3D graphics through OpenGL ES and OpenGL. It lets developers fully utilize GPUs (graphics processing units) on mobile devices supporting OpenGL (ES). Devices without hardware acceleration can use an OpenGL (ES) software implementation or *Edgelib*'s "fast" internal 3D renderer. *Edgelib* can read 3D Studio Max files and MilkShape 3D files.

Issues
Handling the small screen size will remain a key issue. Some studies propose to take advantage of public displays to overcome the problem faced by small screen size (Leikas et al. 2006). Such displays can have a more detailed version of game world. Moreover non-players can also enjoy the game. However, public displays should just add value to the game and must not be a requirement. Otherwise it would hinder the mobility of MMGs.
In future more and more users will be equipped with mobile phones with 3D accelerator hardwares. This will allow eye catching graphics in games. But this will lead to having two classes of players: The ones equipped with new mobile phones and the others using old ones. The issue for game developers will be to develop two completely different versions of the same game, but allowing people to play together
Flash Lite will become a popular technology for mobile game development as it progressively becomes more common on mobile phones (Koivisto 2007). But what will be available to make *multiplayer* Flash Lite games? Indeed there are solutions like SmartFoxServer (Gotoandplay 2006) in the PC environment. Are they applicable to mobile environments?

Inputs management

Inputs management functionality is responsible for taking into account player actions related to the game.
Keyboard (or stylus, if available) is the most common means by which a user interacts with MMGs.
The mobile phone has several inbuilt features like camera, microphone, video player, phone book, which can be used by MMG's game design. (Koivisto 2007) gives some examples: i) The phone book provides contact details of friends and it can be exposed to the application so that players can contact/invite their friends during game sessions; ii) The mobile phone camera can be used for creating contents. The Nintendo DS allows players to interact with games by using its integrated micro-

[3]Now, many 2D animation techniques simulate 3D animation by altering the look of the objects to simulate depth.

[4]Surface content can be copied from one surface to another by this method.

phone. For example player can take an action in the game by blowing air through the microphone. This idea can be reused with mobile phones microphones.

Even if they are less common, hands-free input methods start to be experienced thanks to accelerometers, player's body monitors, etc. For instance, Rexplorer game uses the following method: Holding their mobile phone in their hand, players draw signs in the air. Meanwhile their mobile uses the camera to analyze gestures and thus recognize these signs.

Illustration

MUPE provides access to keyboard, but also phone inbuilt features such as camera, sound, video, Bluetooth IDs, location.

Issues

The available input interfaces are too limited on current mobile phones.

One research direction is to have game designs which cope with these limits. For instance, *One button* games are successful mainly because of their simplicity (Sheffield 2006). But they are more suitable for casual mobile games and truly do not satisfy the fast-paced MMGs' requirements.

Actually most promising improvements will come from human-computer interaction (HCI) research field. Indeed, today, more and more mobile phones integrate a touchpad functionality on their display. Motorola intends to use it for character recognition. Apple has patented the simultaneous use of two fingers on its iPhone's display/touchpad: Thanks to this functionality, users can simultaneously move their thumb and forefinger on the touchpad as if they wanted to reduce the size of a displayed picture. In the near future, ongoing HCI studies will provide easier and more natural means of interacting with mobile phones, possibly through body monitors hardware.

Context-awareness

Context-awareness can be described as the device's (mobile phone) ability to react and adapt to changes in the user's environment (Suomela et al. 2004). Context awareness allows the environment to have an effect on the gameplay. Actually it is the main differentiating factor of mobile services compared to online services on PCs: New gameplays can be imagined as MMGs can be aware of the user's surroundings/environment. For instance, in *Samurai Romanesque* game (Potel 2003), actual weather conditions are integrated into the game world: If it rains in a real region, avatars located in the corresponding game region cannot use their muskets (due to wet gunpowder).

Illustration

Context-awareness in *MUPE* is built into two separate parts (Suomela et al. 2004). First, the connected user clients can send their context information to the server, as they are constantly connected to the server. Second, any information in the Internet can be sent to the *MUPE* application by writing a context producer that formats context information according to the Context Exchange Protocol (CEP) (Lakkala 2002).
The context producer sends its information to *MUPE* core. *MUPE* core triggers scripts related to this context information change. These scripts may change some data of the different games managed by *MUPE* server. Possibly this evolution is notified to the different *MUPE* clients.

Issues

First issue concerns collecting of context information: How to gather it efficiently in order to avoid network saturation because of collecting data? Middlewares like JORAM look like an interesting solution.
Another issue concerns the processing of this context information: How to avoid saturation of the processors (especially if these are processors of limited terminals)? Middlewares like COSMOS provide a promising answer (Conan et al. 2007).
Another question is: How to take into account context evolution into the game world? More precisely what programming environment should be provided to the game developer to ease game development? We presented *MUPE* development model previously. In Mediascape, context-awareness is limited to geolocalization (mscape 2007): The terminal plays the appropriate bits of media (sound, pictures, etc.) according to its position. The program is designed as the combination of a map and a storyboard. OpenMEE goes one step beyond by providing configurable/extensible design representations (Biswas et al. 2006): maps, storyboards, state-charts, algebraic representations, etc.
Final issue concerns correct interpretation of context data. For instance, imagine a mobile which detects it is in a hot, moist place. It could deduce it is besides a swimming pool whereas it is in the stomach of a dog who just swallowed it. How to improve context inference?

Sound

Game developers are also concerned with the efficient playback of audio (and possibly video at some point). In mobile environments they have the option to use tones or the more advanced sound options (such as wave sounds, MIDI music, and MP3 audio), or possibly some combination of the two.
Game studios usually do not pay too much attention to the sound (They concentrate most of their energy on the graphics). This may be a mistake as good sound features on its own can make the gameplay more interesting and

appealing. For instance, initial versions of PC game Astronoid were not overwhelming because their sounds did not "sound" realistic.

Illustration
Edgelib's sound module provides an interface for existing multi-platform sound libraries. It provides default support for the multi-platform sound engine: Hekkus Sound System[5] library. Other sound libraries can be added by creating a custom wrapper. The default silent interface is also available if no external sound library is used. It is possible to add additional effects to sound and music playback. These include: fade in, fade out and cross-fade effects. Also it is possible to adjust pitch and panning when playing sound effects.

Issues
The quality of sound hardware will improve as there will be more and more hi-fi mobiles (like Walkman phones, for instance).
There are studies on games (like Demor, for instance) for blind people who play only with sounds. These studies could make sound functionality an interesting alternative to the small screen size issue.

Data management

Considering the highly interruptible nature of mobile phones (phone calls, messages, unexpected network failure or battery loss), it is necessary to store game related data. The game data such as high scores, game state variables, game saves, must be saved and retrieved efficiently when required.

Illustration
GASP provides services to store or retrieve data from a distant server. It enables basic SQL requests to a distant HTTP persistency server (Pellerin 2007).
Moreover *GASP* can rely on MIDP RMS mechanisms (Knudsen 2003), which give access to a device persistent storage. *GASP* can also rely on JSR-75 (PDA Optional Packages for the J2ME Platform) primitives: They provide access to device file system.

Issues
There are on-going studies on how to use mobile database to implement games. For instance, (Mottola et al. 2006) presents the design and implementation of a pervasive game on top of TinyLIME, a middleware system supporting data sharing among mobile and embedded devices: Data is distributed between laptops (carried by the players) and motes (present in the different rooms where the game is played). Laptops store the equipments which players hold in the game. The motes store the objects and possible monsters present in the room. When a player takes an object in a room, corresponding data is moved from room's mote to player's laptop.

Miscellaneous

PC/console game engines also integrate several other functionalities: artificial intelligence, terrain generation and physics engine. Do these have a counterpart in the context of MMG engines?
Concerning artificial intelligence, as MMGs are currently rather simple, developers usually develop their own solution (Morrison 2005). None of our studied MMG engines proposes such functionality.
The same is true for terrain generation and physics engines, probably because these latter functionalities are too resource consuming. Moreover the current simplicity of MMGs do not justify their use.

Issues
As terminal processing power increases, these functionalities will be more and more requested. Main issue is how to make these functionalities available on a mobile terminal despite its limited capabilities.

MMG ENGINES EXTRA-FUNCTIONALITIES

Previous section was dealing with MMG engines core functionalities. This section presents functionalities which ease the development of the game even though they are not directly linked to gaming aspects.

Deployment

Deployment is the functionality dedicated to the installation, update and uninstallation of the game on the user's mobile (and possibly on servers in the case of a multiplayer mobile game). For editors, an important sub-functionality of deployment is payment management.

Illustration
MUPE client uses a custom script language, which encapsulates J2ME functionalities. The end-users are required to do a single install only. Service discovery is built in: A single install provides end-users with access to many services.
When developers want to deploy their *MUPE* game (first installation or update), they need to copy their program files in directories used by *MUPE* core and *MUPE* server. Afterwards, when necessary, *MUPE* client will download part of its code from *MUPE* core. Moreover, if developers' game is a context-aware game, they have to install, on the sensors, the code responsible for transmitting context data to *MUPE* core.

[5]http://www.shlzero.com

Issues
Current mobile phone's deployment capabilities are too restricted (mainly for security reasons). For instance, it is not possible to install a J2ME application and also the shared library which it will use: The whole code must be in the same jar file. JSR-124 (J2EETM Client Provisioning Specification) is a first (incomplete) answer to these limitations.
Another issue is related to distributed deployment: How to guarantee that the different parts of an application is completely (and coherently) installed, updated or removed from the miscellaneous equipments which should host them?

Dealing with portability

In mobile world, portability of games between different phones with different features is by itself an issue. Software development environments (such as Java or C++) may vary as well as screen sizes and keypads. Even the mobile phones developed by same manufacturer may vary. This issue is of much more concern for MMGs, because people living in different geographical locations may want to play the same game. And there is a high risk that their phones vary a lot in terms of features. Thus MMG developers need help for making their game available on many different hardware.

Illustration
Edgelib is referred as a "true multi-platform game engine", mainly because it can be used to create multi-platform games smoothly through the generic interface and all of its key features are available on each platform. *Edgelib* currently supports the following platforms: Windows desktop (2k/XP), Windows Mobile Pocket PC, Windows Mobile Smartphone, Symbian UIQ (such as the Sony Ericsson P900 and M600), Symbian Series (such as Nokia N-Gage and N70, E60). *Edgelib* provides support for most of the input and graphic features which are available on these platforms.
Nevertheless *Edgelib* does not provide any support for Java-only mobiles.

Issues
Tools like Mobile Distillery or UMAK help in the porting process. Nevertheless they do not prevent developers from installing their game on dozens of mobiles to check that the ported Java game is running fine. Porting process is still costly.
Some countries offer a much healthier situation. In the USA, BREW is a *de facto* standard. In Korea, government has imposed WIPI. In the rest of the world, there are on-going initiatives to make J2ME more strict, thanks to, for instance, compliance testbeds.
Portability issue will mostly be solved by industrial companies (there is limited research perpective). For instance, in the last years, Nokia simplified some aspects of portability by standardizing the size of its mobile phones' display.

Development tools

Development tools help in the development process.

Illustration
MUPE Developer Tool plugin for Eclipse offers instructions and helps in application development. Developer Tool contains template projects for getting started with the different applications. In addition, there are code assist, code examples and tools for linking the XML and Java files.

Issues
In a previous section, we presented the issues related to context-aware game's development.
Another issue concerns monolithic aspect of MMG engines. For instance, to develop a game there is no way to use functionality A of MMG engine 1 and functionality B of MMG engine 2. This raises several issues: i) How to glue MMG engines functionalities easily and efficiently (especially if they are coming from various providers)? ii) How to replace easily and efficiently one MMG engine's functionality by another MMG engine's functionality? Indeed solutions exists (and are applied) in PC/console world, but they are not compatible with mobile's limited resources.

CONCLUSION

This paper presented the different functionalities of Multiplayer Mobile Game engines. For each functionality, it presented the specificities of mobile environments which are taken into account by current MMG engines, illustrated it with existing MMG engines and introduced remaining issues. The key issues are: portability, context-awareness, monolithic aspect of current MMG engines, networking, and deployment. Portability has limited research perspective. Context-awareness has many open and interesting issues, especially the one related to the easy integration of application's context-awareness in the development process. Working on the limits of monolithic aspect of current MMG engines seems an attractive research direction. In the networking field, architectures mixing various networks, data consistency sub-functionality, as well as in-game communications, offer worthwhile research opportunities. Finally, efficient solutions for deployment issues also need to be found.

ACKNOWLEDGEMENTS

The authors thank *GET — INT* school and *JEMTU* project who funded this study.

REFERENCES

Biswas A.; Donaldson T.; Singh J.; Diamond S.; Gauthier D.; and Longford M., 2006. *Assessment of mobile experience engine, the development toolkit for context aware mobile applications.* In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology.* ACM Press, New York, NY, USA, 8.

Broll W.; Ohlenburg J.; Lindt I.; Herbst I.; and Braun A.K., 2006. *Meeting technology challenges of pervasive augmented reality games.* In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games.* ACM Press, 28.

Conan D.; Rouvoy R.; and Seinturier L., 2007. *Scalable Processing of Context Information with COSMOS.* In J. Indulska and K. Raymonds (Eds.), *Proc. 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems.* Paphos, Cyprus, vol. 4531, 210–224.

Costikyan G., 2005. *Mobile games: Medium and style.* Keynote Talk at 4th workshop on Network and System Support for Games (NetGames), Available online at: http://www.research.ibm.com/netgames2005/papers/costikyan.pdf.

Elements Interactive B.V., 2007. *Edgelib Mobile Game Engine.* http://www.edgelib.com/.

Gotoandplay, 2006. *SmartFoxServer: Socket server for Flash multiplayer games and applications.* http://www.smartfoxserver.com/.

Khan A.M.; Chabridon S.; and Beugnard A., 2007. *Synchronization Medium : A Consistency Maintenance Component for Mobile Multiplayer Games.* In *Proceedings of 6th Annual Workshop on Network and System Support for Games 2007 (NetGames 2007).* Australia, ACM.

Knudsen J., 2003. *Wireless Java: Developing with J2ME.* Apress, second ed. ISBN 1–59059–077–5.

Koivisto E., 2007. *Mobile games 2010.* Tech. rep., Nokia Research Center Finland, http://research.nokia.com/tr/NRC-TR-2007-011.pdf.

Lakkala H., 2002. *Context Exchange Protocol (CEP).* http://mupe.nrln.net/files/cep_1_0.pdf.

Leikas J.; Strömberg H.; Ikonen V.; Suomela R.; and Heinilä J., 2006. *Multi-User Mobile Applications and a Public Display: Novel Ways for Social Interaction.* In *PerCom.* IEEE Computer Society, 66–70.

Morrison M., 2005. *Beginning Mobile Phone Game Programming.* SAMS, first ed. ISBN 0–672–32665–5.

Mottola L.; Murphy A.L.; and Picco G.P., 2006. *Pervasive games in a mote-enabled virtual world using tuple space middleware.* In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games.* ACM Press, 29.

mscape, 2007. *mscape - You are here.* http://www.mscapers.com/.

MUPE, 2007. *Multi-User Publishing Environment Application Platform.* MUPE website. Available online at: http://www.mupe.net.

Nokia, 2003. *TibiaME Case Study (Version 1.0).* Nokia forum, http://www.forum.nokia.com/main.html.

Nokia Forum, 2003. *Overview of Multiplayer Mobile Game Design (Version 1.1).* Tech. rep., Nokia, http://www.forum.nokia.com/main.html.

Open Mobile Alliance, 2007. *Open Mobile Alliance: Games Services Working Group.* OMA web site, http://www.openmobilealliance.org/tech/wg_committees/gs.html.

Pellerin R., 2007. *The MooDS protocol: a J2ME object oriented communication protocol.* In *Proceedings of International Conference on Mobile Technology, Applications and Systems (Mobility Conference).* Singapore.

Pellerin R.; Delpiano F.; Duclos F.; Gressier-Soudan E.; and Simatic M., 2005. *Gasp: an open source gaming service middleware dedicated to multiplayer games for J2ME based mobile phones.* In *7th Int. Conference on Computer Games CGAMES'05 Proceedings.* 75–82.

Pellerin R.; Delpiano F.; Gressier E.; and Simatic M., 2007. *GASP – a middleware for mobile multiplayer games.* http://gasp.objectweb.org.

Potel M.J., 2003. *Samurai Romanesque, J2ME, and the Battle for Mobile Cyberspace.* *IEEE Computer Graphics and Application*, 23, no. 1, 16–23.

Sheffield B., 2006. *GDC: Success Factors of One-Button Casual Mobile Games.* http://www.gamasutra.com/features/20060322/sheffield_01.shtml.

Smed J. and Hakonen H., 2006. *Algorithms and Networking for Computer Games.* Wiley, first ed. ISBN 0–470–01812–7.

Suomela R.; Räsänen E.; Koivisto A.; and Mattila J., 2004. *Open-Source Game Development with the Multi-user Publishing Environment (MUPE) Application Platform.* In *ICEC.* Springer, *Lecture Notes in Computer Science*, vol. 3166, 308–320.

YeAST: THE DESIGN OF A COOPERATIVE INTERACTIVE STORY TELLING AND GAMEBOOKS ENVIRONMENT

Paola Salomoni
Silvia Mirri
Department of Computer Science
University of Bologna
Mura Anteo Zamboni, 7 40127 Bologna, Italy
{mirri, salomoni}@cs.unibo.it

Ludovico Antonio Muratori
Corso di Laurea in Scienze dell'Informazione
University of Bologna (Cesena)
Via Sacchi 3 47023 Cesena (FC), Italy
muratori@polocesena.unibo.it

KEYWORDS

Interactive cooperative tools, Interactive story telling, Web 2.0 technologies.

ABSTRACT

Sharing knowledge and creativeness is one of the fulfilled potentials of IT. The light hearted dimension of this so called "collective intelligence" is furthermore enforced by the arising Web 2.0 technologies together with the possibilities of massive multimedia employment. Interactive Story Telling is an example of such an assumption. This paper presents a system gathering IST, multimedia presentations and role game features to allow the creation and the modification of digital tales on an open easy-to-use environment.

1. INTRODUCTION

Marcel Proust wrote "*In reality, every reader is, while he is reading, the reader of his own self*" and this is possibly more glaring if dealing with a storyteller and an oral process of sharing stories. Voice sound, gestures, rhythm, newer and newer (often different) details, audience feedbacks, cast the teller inside *his* story as if he were a player of some kind of role or game, rather than a simple go-between.
According to the IT jargon, we might describe such an approach as a sort of multimedia versioning of a tale, an adaptive interface and, indirectly, a collaborative long-term work. Between the fixed structure and order of a written tale and the uniqueness of a told story, recent history shows the so-called "gamebooks" and Interactive Story Telling (IST) environments (Crawford 2007, Cacciaguerra et al. 2006). The formers branch into different sceneries, according to the reader choices along a series of plot steps, while the latter ones shift the narration from a linear form to a dialectical one through dialogues and discussions on a typically collaborative environment. On the other hands, role games can be easily conceived as unique performances of a story, where players from time to time build a sort of a newer and newer story plot (Romero et al. 2004).
This paper describes a system (named *YeAST, Yet Another Story Told*) where IST and role games features are exploited, together with multimedia resources, in order to offer an absolutely open playground for creating, modifying and ruling stories, not only through textual statement, but also with images, video and audio clips. The main goal of such a system is providing a game environment where players participate in building a plot, by adding, editing and timing multimedia contributions. It tries to be a sort of virtual movie-set, where any possible situation is added and evolves while it is realistically seen and heard by any character, player and audience.
As it happens with IST, individuals can share their abilities and generate a more creative final product. The collaborative dimension of such an environment has implied the choice of a suitable, friendly and standardized interface. One of the several flavour of a Wiki Content Management System has been adopted to answer such a basic instance. It has been integrated with some syntax extensions with a suitable parsing add-on to the Wiki engine, in order to add and edit multimedia resources. The SMIL standard has been used as a sub-framework to describe added resources and codify the extended syntax. Moreover, an interface has been designed and implemented inside the Wiki, to show any possible branching of each story plot.
Finally, a subsystem has been added to the Wiki architecture to define roles and policies for the players and to inherit versioning built-in features (Leuf and Cunningham 2001). It's worth noticing that, no rules are established beforehand. From time to time, a sort of moderator can decide the acceptance of new (or modified) "events" on the story. In this sense, the existing versioning system of the Wiki, allows to roll back the story whenever it were necessary.

The reminder of this paper is structured as follows: Section 2 illustrates main design issues and some motivations which have driven our work. Section 3 details the architecture of our system, according to the framework provided by the Wiki CMS. Section 4 describes some aspects of the implementation, particularly referring to the syntax extensions of the players interface. Finally, section 5 points out some open trends of the system, concluding the paper.

2. ON THE DESIGN OF A WIKI-BASED IST APPLICATION

The main goal in designing the architecture of our collaborative system is the integration of Web 2.0 technologies and interactive story telling features. Our idea is based on the provision of a unique application in order to offer easy-to-use editing and user managing, as well as a sort of multimedia repository mechanisms.
Today, Wiki engines are certainly one of the most widely utilized content management systems and represent one of the key technologies for enabling user collaboration in the Web. These Web 2.0-based systems embrace an approach where contents are collaboratively edited by a multitude of users (Vossen and Hagemann 2007). In Wikis, a specific syntax must be employed to edit a new content; the source format that is produced to create and organize textual contents is called *Wikitext*. Users are allowed to edit a simple markup language which exploits plain text with a few simple conventions for creating links and for giving some structure and/or style to the edited contents (Ferretti et al. 2007). Then, the inserted source code is automatically converted to a final HTML document (Désilets et al. 2006).
While similar among each other, each Wiki system has its own syntax, grammar, structure and keywords. No standard has been provided yet. Hence, for instance, different Wiki engines typically have different syntax conventions to specify links. While the very first Wiki systems were only able to produce spare HTML pages with fixed structure and only some simple graphic styling (e.g., bold, italic, acronyms), more recent versions add support for a more complex editing, which allows, for example, to insert text decorations, tables, images and formulas.
Wikis provide a means to verify the validity of recent additions to the body of pages. The most prominent, on almost every Wiki, is the "Recent Changes" page, a specific list numbering recent edits, or a list of all the edits made within a given time frame. Some Wikis can filter the list to remove minor edits and edits made by automatic importing scripts.
From the change log, other functions are accessible in most Wikis: the Revision History showing previous page versions; and the difference feature, highlighting the changes between two revisions. Using the Revision History, an editor can view and restore a previous version of the article. The difference feature can be used to decide whether or not this is necessary. A regular Wiki user can view the differences of an edit listed on the "Recent Changes" page and, if it is an unacceptable edit, consult the history, restoring a previous revision; this process is more or less streamlined, depending on the Wiki software used.
In case unacceptable edits are missed on the "Recent Changes" page, some Wiki engines provide additional content control. It can be monitored to ensure that a page, or a set of pages, keeps its quality. A person willing to maintain pages will be warned of modifications to the pages, allowing him or her to verify the validity of new editions quickly. A similar control can be applied also to the content and its format. All the above described features provide a support to a sort of versioning system for the content and its format of any Wiki-page.
Many Wikis are open to the users without the need to register any account. Sometimes session log-in is requested to acquire a "Wiki-signature" cookie for autosigning edits. Many edits, however, can be made in real-time, and appear almost instantaneously online. This can lead to abuse of the system. Private Wiki servers require user authentication to edit, sometimes even to read pages. Some Wikis allows user managing, defining different groups of users with different levels of permission for each Wiki-page.
Besides traditional Wiki characteristics, our system YeAST needs some other additional feature. In order to provide a sort of digital gamebook and interactive story telling editing and enjoying, a privileged group of users should be able to define and fix some step, such as the first one, some intermediate ones or the end of the story. More over, it should be possible to provide alternatives which follow such fix steps of the story: as well as in reading a gamebook, the users can choose the preferred alternative in order to compose their own story.
Users should upload not only images and link to other resources (such as pdf files), but also audio and video. On one hand the Wiki engine has to be adapted in order to accept such media files and to embed them in the Wiki-pages. On the other one, the Wiki-syntax should be enrich in order to provide a easy-way to add media and multimedia to the story, to synchronize them each other (in a parallel or sequential way), to plot steps and to define following alternatives, if users' permissions allow it.

3. SYSTEM FEATURES AND ARCHITECTURE

YeAST has been designed as an interactive, collaborative, open environment to build multimedia stories. It has been implemented as an instance of the DokuWiki CMS (Dokuwiki, 2007), by exploiting its plug-ins system to extend the Wikitext syntax. As it is shown in Figure 1, actors of such a system can be classified into three general sets: as audience of the story, as players/tellers arranging the sceneries and as directors/moderators surveying the consistency of any plot. Each set has its own collecting rules, which define the policies of access to the story. Audience can simply access the sequence of scenes the story is made up; players can add, edit, modify events on scenes, while moderators can accept or refuse any variation of the plot. An RSS feed (Really Simple Syndication) is provided by the system about every novelty the players have tried to introduce (RSS 2007). A mechanism for deferring the publication of any submitted new event can be activated or get disabled. In the first case, no one except the moderators is able to see any evolution of the story before it has been approved. The second case (truly closer to the Wiki philosophy) implies possible roll backs of a universally visible version of the story. Each story is quantized in scenes (or *story unit*) structured as a graph. The versioning system of Wiki allows to rewind the story, while a proper policy on new events makes the deferred publication possible.

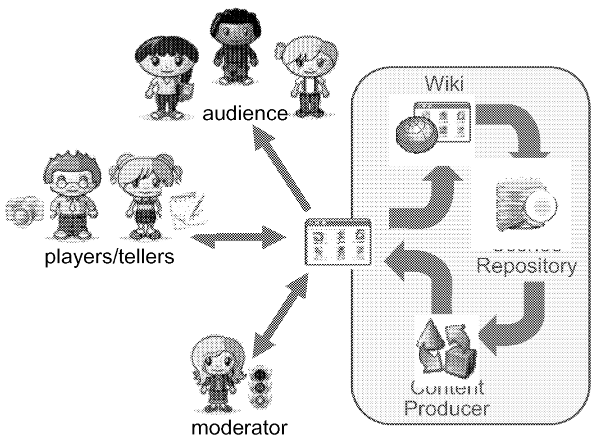


Figure 1: YeAST actors

At the server side, the YeAST manages multimedia content (stored in units, or *scenes* inside the Scenes Repository) in a Wiki-like modality:

- A wiki editing interface is used to add or modify a scene, by using a text-based language that extends the traditional Wiki syntax, by adding some multimedia synchronization features (named EWT, *Extended Wiki Text*).
- A Content Producer composes scenes, described in EWT and related to multimedia files, by composing an appropriate HTML and SMIL content.

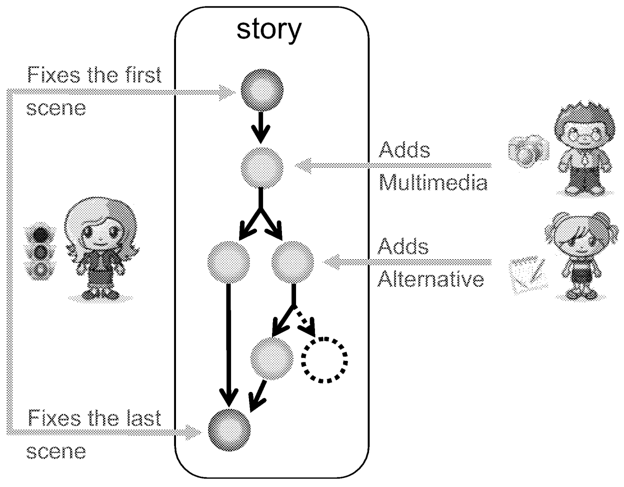


Figure 2: The collaborative editing process

As it is shown on Figure 2, a (possibly cyclic) graph of scenes represents the varying flow of a story. Any user can insert or edit a contemporary or sequential new resource to the flow of the story. The moderator fixes some step of the story to the YeAST system (in Figure 2 the first and the ending scenes are fixed). Contemporary events are collected as grained resources of a single scene or suitably mapped, starting from an entire story unit, which is shown at the same time of another one. Such a meta information is simply presented with a proper tag between two or more contemporary scenes. According to the time and space limits of any single story unit the (quantized) flow of each story can also be beaten by a scene unit as a whole or, once again, as a resource of it.

4. EXTENDED WIKI TEXT

Two different tags exist in SMIL for specifying the media contemporaneity and the sequentiality: `par` identifies a set of parallel elements while `seq` identifies a set of sequential media. With this in view, we added to the Wikitext a specific syntax for each of these two tags (W3C 2005).
In particular, in our Wiki syntax a +++ symbol corresponds to the par SMIL element and the ^^^ symbol corresponds to the seq SMIL tag.
To add a set of parallel media, a user should exploit a syntax which is as follows (see Figure 3).

```
+++ "first media"
+++ "second media"
...
+++ "last media"
```

Figure 3: EWT code fragment for `<par>`

Instead, to add a set of sequential media, a user should exploit a syntax which is as follows.

```
^^^ "first media"
^^^ "second media"
…
^^^ "last media"
```

Figure 4: EWT code fragment for `<seq>`

For instance, instead of updating directly the SMIL code, through our system it is possible to add media in a Wikipage (as a gamebook) by writing the following lines into the specific text form of the Wiki interface.

```
+++ [[http://www.criad.unibo.it/
audio/rainfalling.mp3]]
It can't rain forever
+++
   ^^^ {{rain.jpg}} 10
   ^^^ [[http://www.youtube.com/
   watch?v=oyHevb1eOrk]]
```

Figure 5: The EWT example

To obtain this result, once the user has added a media using our Wiki-like interface, the system automatically adds to the document the SMIL markup code needed to display that caption for the specific time interval, i.e., it adds the `<par>` and `<seq>` elements, reported in Figure 5, in the proper position of the SMIL document. In this example an audio (`rainfalling.mp3`) is reproduced as a background for a visual sequence composed by an image (`rain.jpg` still for 10 seconds) and a video resource from YouTube (Youtube 2007). In order to represent nesting, a +++ element is used to open the sequence of visual resources. This example is limited to the main methods (parallelizing, sequencing and duration) used as extension to a traditional Wiki syntax. Obviously the whole syntax includes all the elements needed to format text pages.
The editing interface for the code fragment of Figure 5 is reproduced in Figure 7.
Besides +++ and ^^^, we have added other new symbols in order to support authors in creating sequences and alternative scenes. Obviously the hypertextual structure of the story graph could be obtained by using traditional link mechanisms but we have tried to provide high-level primitives to facilitate authors and, at the same time, strength the perception of the story flow.
With this aim, we have added:

- !!!, which identifies the one and only scene that follows the current one.
- ???, which identifies all the (more than one) alternative scenes following the current one.

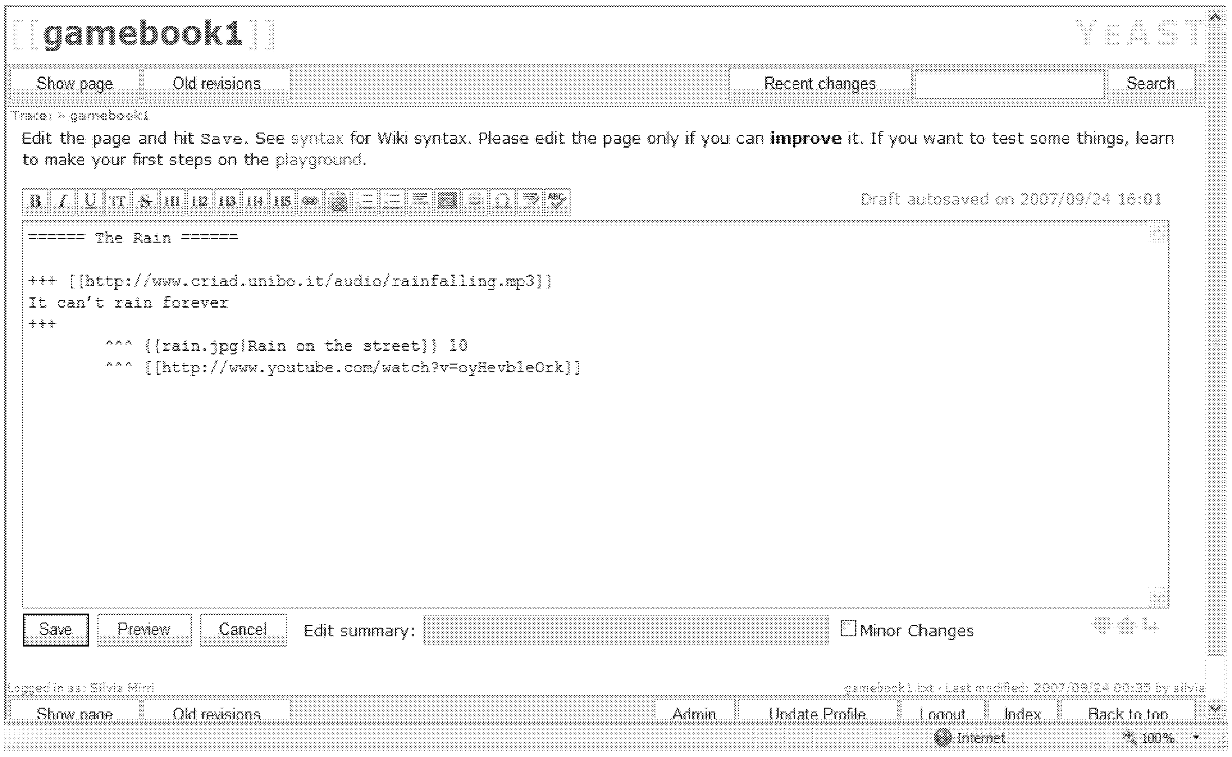
```
<par>
  <audio src="http://www.criad.unibo.it/
   audio/rainfalling.wav" />
  <seq>
    <image src="rain.jpg"/>
    <video src=" http://www.youtube.com/
     watch?v=oyHevb1eOrk "/>
  <seq>
</par>
```

Figure 6: The SMIL code fragment corresponding to EWT code of Figure 3

Generally, on YeAST, players can upload a SMIL-based archive (as a common .zip file) or edit any existent one, arbitrarily entering the scene. The interface on the Wiki editor shows any available resource (audio, video, images, or text) of the SMIL document and let to modify or add further branches (as contemporary or sequential new resources) to the multimedia flow. A contribution can be either a multimedia file or, simply, a link to any multimedia resource somewhere over the Internet. As a final result, the Wiki parser produces a Web Page where the possibly new scene is shown. As shown on Figure 1, YeAST has a typical client/server architecture where a "scenes forest" is server-side produced to describe a story which user build and publish on a client side editor.

5. CONCLUSIONS AND FUTURE WORK

The novelty of YeAST is the possibility of telling and playing a newer and newer story, not only by adding or modifying text pieces of a plot, but also enriching the scenery with images, video and audio clips. The aesthetic and emotional impact of such an approach draw up the final result to the storytelling tradition and cast players and audience into a playground, which is more realistic than usual role games scenarios. Wikis characteristics of openness and collaboration shift the "collective intelligence" to the "collective creativity". Future trends of YeAST are addressed toward the use of Web 2.0 tagging systems (like, for example, Youtube tags) to allow users finding resources through proper keyword. Furthermore, An existent work about extending syntax for multimedia noting on a Wiki, named LAUGH - LAbelling Unbound Grained Hypermedia (Ferretti 2007) is being evaluated as an alternative to the current syntax. Finally, overflow of media as contemporary resources, as well as possible ruling mechanisms and management of spatial synchronization (SMIL regions) remain as open questions for the presented system.



Edit the page and hit Save. See syntax for Wiki syntax. Please edit the page only if you can **improve** it. If you want to test some things, learn to make your first steps on the playground.

Draft autosaved on 2007/09/24 16:01

```
====== The Rain ======

+++ [[http://www.criad.unibo.it/audio/rainfalling.mp3]]
It can't rain forever
+++
        ^^^ {{rain.jpg|Rain on the street}} 10
        ^^^ [[http://www.youtube.com/watch?v=oyHevbleOrk]]
```

Figure 7: A screenshot of the editing system

ACKNOLEDGEMENTS

This work is supported by the Italian M.I.U.R. under the MOMA and DAMASCO initiatives. Authors want to thank Marco Roccetti e Stefano Ferretti for their cooperation during this work.

REFERENCES

Cacciaguerra, S.; Roccetti, M.; and P. Salomoni. 2006. "Multimedia Entertainment Applications – Interactive Story Telling". In *Encyclopedia of Multimedia*, (B. Furht Ed.), Springer, January, 510-518.

Crawford, C. 2007. Storytron Interactive Storytelling Home Page. http://www.storytron.com/index.html.

Désilets, A.; Gonzalez, L.; Paquet, S.; and M. Stojanovic. 2006. "Translation the Wiki way". In *Proceedings of the ACM International Symposium on Wikis*, Denmark, 19-32.

Dokuwiki. 2007. Dokuwiki Home page. http://wiki.splitbrain.org/wiki:dokuwiki.

Ferretti, S.; Mirri, S.; Roccetti, M.; and P. Salomoni. 2007. "Notes for a Collaboration: On the Design of a Wiki-type Educational Video Lecture Annotation System". In *Proceedings of the First IEEE International Workshop on Semantic Computing and Multimedia Systems* (IEEE-SCMS'07), Irvine, CA, IEEE Computer Society, 651-656.

Leuf, B. and W. Cunningham. 2001. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley.

Romero, L.; Santiago, J.; and N. Correia. 2004. "Contextual information access and storytelling in mixed reality using hypermedia". In *Computers in Entertainment (CIE)*, v. 2 n. 3, 12-45.

Vossen, G. and S. Hagemann. 2007. "*Unleashing Web 2.0: From Concepts to Creativity*". Morgan Kaufmann.

RSS Board. 2007. "RSS 2.0 Specification". http://www.rssboard.org/rss-specification.

W3C. 2005. *Synchronized Multimedia Integration Language (SMIL 2.1)*. W3C Recommendation 13 December 2005, http://www.w3.org/TR/SMIL/.

Youtube. 2007. Youtube homepage. http://it.youtube.com/.

SIMULATING INFINITE CURVED SPACES USING VERTEX SHADERS

M. C. Bouterse,
A. Eliëns,
Department of Computer Science
Faculty of Sciences, Vrije Universiteit
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
E-mail: eliens@cs.vu.nl

KEYWORDS
Shader Programming, 3D In-Game Animation, Vertex Manipulation, Curve Simulation

ABSTRACT

Rendering dynamically curved meshes can be complicated if not impossible using traditional methods. Either it involves creating pre-curved meshes and fixed animation sets or processor intensive calculations to deform mesh data on the fly. A model is presented here that can be used to create infinite dynamic curved spaces in real-time using a vertex shader. This makes dealing with curved meshes in real-time rendering easier and makes procedural animations of curves possible. Applications include dynamically curving environments and objects for games.

INTRODUCTION

Shader programming has acquired an important position in the field of real-time rendering. The ability to control the rendering process by executing custom programs has provided new possibilities for numerous applications. Many articles have been published in the *ShaderX* and *GPU Gems* book series describing new algorithms for graphics hardware. Most of these articles demonstrate techniques to improve the visual quality of the rendering process. The ability of the vertex shader to manipulate vertices to animate or deform meshes is less often explored. A few articles dealing with this topic use the vertex shader to render ocean water (Isidoro 2002a and Finch 2004), fields of grass (Isodoro 2002b and Pelzer 2004) or soap bubbles (Isodoro 2002c). These articles show the potential of the vertex shader in vertex-based animation and deformation for very specific applications. This paper builds on the concepts presented in these articles and describes a general technique for applying curves to vertex spaces.

This paper presents a method for using vertex shaders to create infinite curved spaces; to apply curves in real-time to meshes. The algorithm provides an easy way to apply multiple curves in any direction to an arbitrary vertex space. First a straightforward method for creating a single curve is explained. This method is then used as a starting point for creating a more generic model. Finally our conclusions regarding this model are presented.

APPLICATIONS

The method described here is originally developed to create curved tunnel systems for a yet unpublished game. The method proved to be very useful for creating dynamically curved environments and can also be used for procedurally applying curves to in-game objects. The method allows for smooth curve animations that are hard to create with traditional technology.

APPLYING A SINGLE CURVE

To be able to apply a curve to an arbitrary vertex space, we need a per-vertex algorithm that translates each vertex from the original vertex space to the desired position in the curved space. For a single curve this algorithm is relatively straightforward to find. To illustrate the concepts presented here we use a cylindrical mesh centred on the positive z-axis starting from the origin. A cylinder is used here for illustrative purposes, the algorithm works on any vertex space, no matter what the distribution is. A cylinder is appropriate, because one of the most obvious applications, creating curved tunnel systems, uses meshes that resemble cylinders.

In figure 1 the result of applying a ninety degrees curve to the sample vertex space is shown. The length of the segment is denoted by the letter d.

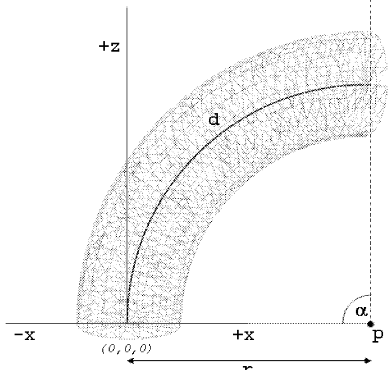


Figure 1: Applying a 90° Curve to a Cylinder

The length of d is preserved in the middle of the cylinder. Preserving the length in the middle of the vertex space is desirable in most cases and leads to minimal stretching and squeezing of the original mesh. The length of the curved space will be the same as that of the original space.

Calculating this curve can be done on a per-vertex basis. The length of the segment (d) and the angle of the total curve (α) must be known beforehand. From this information the main radius (r) of the curve can be calculated using the formula for the circumference ($2\pi r$). Taking into account the part of a full circle that the curve covers, the formula for the radius becomes:

$$r = d / \alpha \text{ (with } \alpha \text{ in radians)}$$

This radius is the distance from the pivot point (p) to the centre of the vertex space (z-axis). This value is not used directly, but used to calculate the per vertex radius as will be shown in the next section.

To apply the 90° curve per-vertex the z-value of the position is used in the vertex shader to determine the position within the curve. For each vertex new coordinates are calculated. Because the curve is applied in the xz-plane, the y-coordinate of the vertex will be preserved. The x and z coordinates can be computed using basic trigonometry with the following formulas:

$$x' = x + r' - \cos(\beta) * r'$$
$$z' = \sin(\beta) * r'$$

Where r' is the per-vertex radius ($r' = r - x$) and β is the per-vertex curve angle that depends on the z position of the current vertex ($\beta = (z/d)*\alpha$); the further away from the origin, the more the vertex will be displaced.

These are the calculations needed to apply a curve to a single segment with a certain angle in the xz-plane. To implement this in a vertex shader the following code snippet can be used:

```
// Compute per-vertex angle
float beta = (pos.z / d) * alpha;

// Compute per-vertex radius
float radius = r - pos.x;

// Calculate curved positions
pos.x += radius - cos(beta) * radius;
pos.z  = sin(beta) * radius;
```

The resulting shader is capable of applying a curve to a segment of arbitrary length by a variable angle. Although this might have some value in practise, the model is very restrictive; only one segment, starting at the origin can be curved and the curve is always in the same direction (in the xz-plane and towards the positive x-axis). A generic model for creating infinite curved spaces is needed and will be presented next.

GENERIC CURVE MODEL

To overcome most of the restrictions of the single curve model, we present a generic model that has the following additions:

- Curves in any direction
- Multiple curves that seamlessly connect

The algorithm we have so far will be used as a basis for creating such a generic model.
First the curve algorithm will be extended to support any curve direction. A naive approach is to simply rotate the model along the z-axis after the curve has been applied to turn it into the desired direction. This allows for curves in any direction, but introduces the problem that vertices are not in the same position as they would be if a curve was directly applied in that direction. A solution to this problem is to first rotate the vertices over the z-axis in the opposite of the desired direction, then apply the curve as shown before and finally rotating it over the z-axis again to its final position. The curve algorithm is now extended with a new parameter, a rotation angle over the z-axis. From now on we will use γ to denote this angle.
The second addition in the generic model is the support of multiple curves. Eventually the following algorithm needs to be executed for each vertex:

1. Determine the curve this vertex belongs to
2. Translate start of the curve to the origin
3. Rotate around z-axis by $-\gamma$
4. Apply the single curve algorithm
5. Rotate around z-axis by γ
6. Align with end of previous curve
7. Connect to previous curve

We have already shown how to implement steps 3-5; the remaining steps are needed to support more than one curve. Before we can implement the remaining steps we must divide the vertex space in segments (each segment has separate curve parameters). The first segment starts at the origin and ends at the plane $z = d_1$, the second starts at $z = d_1$ and ends at $z = (d_1+d_2)$, etc. Each of these segments has its own values for α, γ, and d.

The first step of the algorithm can now be implemented by comparing the z value of the current vertex to the start of each segment until the right segment has been found. Translating the segment to the origin is done by subtracting the start value of the segment from the z value of the current vertex. After the translation, steps 3 to 5 are applied as described before. This can be implemented by a single matrix multiplication. This matrix is the result of concatenating a rotation matrix ($-\gamma$) with a curve matrix (described next) and finally another rotation matrix (γ). The curve matrix needed here can be created from the calculations shown for the single curve algorithm. By substituting the radius equation ($r' = r - x$) into the curve calculations, we get the following matrix multiplication:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ (1-\cos(\beta)) \cdot r & 0 & \sin(\beta) \cdot r & 1 \end{bmatrix}$$

Finally the segment must be translated and rotated in such a way that it seamlessly connects to the previous segment (which can have an arbitrary curve as well). A transformation matrix must be computed that takes care of this. Given the per-segment parameters (γ, α and d), such a transformation matrix can be computed for each separate segment. The transformation matrix for segment N can then be computed by concatenating the matrices from segment `0, 1, …, (N-1)`.

The transformation matrix for each segment is computed by first orienting the vertices in the right direction and consequently translating them to the end of the previous segment. Aligning segment `N` with the end of segment `N-1` can be done by rotating the segment using the curve parameters of segment `N-1`. First segment `N` is rotated along the z-axis with the negated γ value of segment `N-1`, then it is rotated along the y-axis with the α value of `N-1` and finally it is rotated along the z-axis again using γ from `N-1`. The final step is to connect segment `N` to `N-1`, which can be done with a translation vector from the origin to the centre of the end of segment `N-1`. Using basic trigonometry we can calculate the components of this vector T:

$$\begin{aligned} T_{xy} &= r - r * \cos(\alpha) \\ T_x &= \cos(\gamma) * T_{xy} \\ T_y &= \sin(\gamma) * T_{xy} \\ T_z &= \sin(\alpha) * r \end{aligned}$$

This is the final piece of the algorithm for a generic curve shader. Implementing this algorithm in HLSL or another shader language should be straightforward using the presented calculations. The per-segment parameters and the segment offset matrices need to be supplied as shader constants once per frame.

CONCLUSIONS

In this paper a method was presented for creating infinite curved spaces on the GPU that can be used to dynamically apply curves to meshes. In a few steps a flexible model was presented that is capable of applying multiple curves in any direction. The scalability of the implementation is depending on the available shader constants, which is probably only an issue on targets lower than shader model 3.0. Applications in game development consist of creating curved tunnel systems and other curved objects without the need for more assets or complicated animations. The presented technique works best on fairly dense and equally distributed vertex spaces. Meshes with few polygons will not look good when curved because the algorithm doesn't add vertices and curved objects need relatively many vertices. This method might be used as a basis for creating interesting dynamically curving game objects and environments. Further research is needed to determine the full potential and scalability of the presented technique. The per-vertex calculations are relatively heavy, but because most modern games are not bottlenecked by vertex processing capabilities, this method should not lead to dramatic loss of performance.

FUTURE POSSIBILITIES

The presented algorithm and implementation runs sufficiently efficient on SM 3.0 hardware, but is still limited by the amount of vertex constants available. This limit will be almost gone on DirectX 10 hardware. Shader model 4.0 supports considerably more constants, making this limit much less important. Also the geometry shader introduced by SM 4.0 could lead to interesting new applications of the algorithm such as the generation of curved objects on the fly or generating extra vertices to curve meshes with few polygons smoothly.

BIBLIOGRAPHY

Isidoro, J.; A. Vlachos, C. Brennan. 2002a. "Rendering Ocean Water" In *Direct3D ShaderX*, edited by W.F. Engel. Wordware Publishing.

Isidoro, J. and D. Card. 2002b. "Animated Grass with Pixel and Vertex Shaders" In *Direct3D ShaderX*, edited by W.F. Engel. Wordware Publishing.

Isidoro J. and D. Gosselin. 2002c. "Bubble Shader" In *Direct3D ShaderX*, edited by W.F. Engel. Wordware Publishing.

Finch, M. 2004. "Effective Water Simulation from Physical Models" In *GPU Gems*, edited by R. Fernando. Addison Wesley.

Pelzer, K. 2004. "Rendering Countless Blades of Waving Grass" In *GPU Gems*, edited by R. Fernando. Addison Wesley.

AUTHOR BIOGRAPHY

ANTON ELIENS studied art, psychology, philosophy, and computer science. He is lecturer at the Vrije Universiteit Amsterdam, where he teaches multimedia courses. He is also coordinator of the Master Multimedia for Computer Science. He has written books on distributed logic programming and object oriented software engineering.

MARCO BOUTERSE received a Master degree in Computer Science/Multimedia at the Vrije Universiteit Amsterdam. His master thesis was about the development of games with a focus on shader technology. Currently he is working as a game programmer at Two Tribes, a game company located in Harderwijk, The Netherlands.

MOBILE GAMING

MOBILE GAMES: WHAT TO EXPECT IN THE NEAR FUTURE

Marco Furini
Computer Science Department - University of Piemonte Orientale
Via Bellini 25/G - 15100 Alessandria, Italy
Email: `furini@mfn.unipmn.it`

KEYWORDS
Mobile Gaming, Mobile Development Platforms.

ABSTRACT

The current mobile gaming market is filled with the so-called casual games. Simple and easy, these games are well suited for devices with limited resources and for people who play games now and then. This is why *Tetris* and *Pac-man* are among the best sellers games of several mobile games charts. In this paper we analyze current developing platforms, networking technologies, delivery models and game characteristics of the current mobile gaming scenario, in order to identify directions that will lead to the near future of the mobile gaming scenario.

THE MOBILE GAMING MARKET

The success and the popularity of mobile games are making the mobile gaming a successful market. Expectations are high as reported by different research reports, which forecast the mobile gaming market between 10 and 20 US$ billions by 2011 (see, Gibson (2006)).
Nowadays the mobile gaming scenario is filled with a wide range of games (from very simple graphic games to cutting edge 3D graphics; from single to multi players games) (see, Figure 1). However, looking at today's mobile gaming market sales, it is interesting to note that customers prefer the so-called *casual* games, easy and simple to play. Just see the best sellers mobile game charts of different cellphone network operators, to note that most of the titles are simple and easy to play: Tetris, Pac-Man, Pong, Frogger, just to name a few. The success of casual games has various reasons: i) mobile games are basically used by normal people, the so-called *casual gamers*, to kill time or to mitigate boredom and are not meant to replace powerful gaming console, ii) limited screen size and input keyboard are a burden for complex games, whereas they are sufficient for casual games, iii) old titles, usually simple and easy to play, are familiar to a large segment of current mobile device's owners and familiarity is an important factor in a game purchase decision (see, Telephia (2007)), iv) the computational power of current cellphones is limited with respect to gaming console, but is comparable to the one of old gaming console, and hence sufficient to support simple games.
Will casual games play a fundamental role in the future of the mobile gaming market? Will casual gamers be principal target of the game industry? Will hardcore gamers be more considered in the future mobile gaming scenario? In this paper we analyze the current mobile gaming scenario in order to identify directions that will lead to the near future of the mobile gaming scenario. In doing this, we focus on developing platforms, networking and graphics technologies, delivery models and games characteristics. Based on the highlighted directions, the future mobile gaming scenario is likely to be no longer focused on casual gamers, but hardcore gamers will be the main target; casual games and complex games will coexist and mobile games will massively use features like multi-player capabilities, social aspects, location and proximity information and high quality graphics. Figure 2 summarizes a possible evolution of the mobile gaming scenario.

DEVELOPING PLATFORMS

Today, the mobile scenario suffers from a considerable platform fragmentation problem (see, Koivisto (2006)), which is a real burden for the popularity of mobile applications and of mobile gaming in particular. In addition, since the number of different mobile devices is enormous, game developers are practically unable to release a game version for any different platform and for any different mobile device. Needless to say, making a game available to the entire mobile market is almost impossible, as the inter-platforms porting cost may be higher than the developing cost of the entire game. Currently, Java ME, Brew, Symbian and Flash Lite are the most popular software platforms used to develop mobile games.
Java Micro Edition (Java ME) is a collection of technologies and specifications to develop software for devices with limited resources and is currently the most ubiquitous application platform for mobile device. The main advantage of using Java ME is that an application can be written once and can be used on every device Java ME compatible. However, the only way to guarantee that the developed application will run on a particular device is to test it on that particular device. Needless to say, this is a limitation. Furthermore, Java ME applications run slowly compared to applications specifically

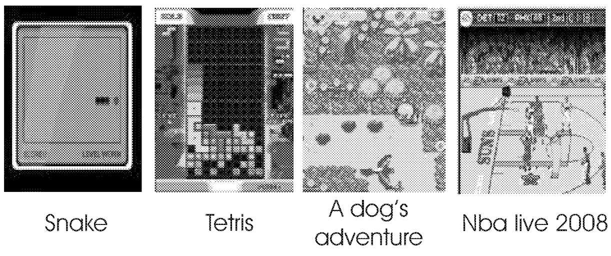
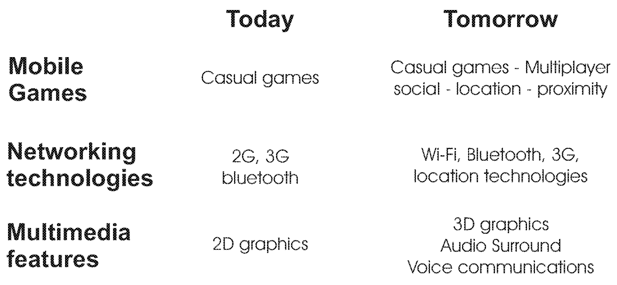


Figure 1: Mobile games evolution: from *snake* to NBA Live.



	Today	Tomorrow
Mobile Games	Casual games	Casual games - Multiplayer social - location - proximity
Networking technologies	2G, 3G bluetooth	Wi-Fi, Bluetooth, 3G, location technologies
Multimedia features	2D graphics	3D graphics Audio Surround Voice communications

Figure 2: The evolution of the mobile gaming scenario.

designed for a particular device/platform and this may be a problem for games that need a fast system response.
Symbian is a proprietary operating system designed for mobile devices with limited resources. Symbian applications, usually written in C++, are designed for a particular device and hence they are more reliable and faster than applications written for generic devices. Needless to say, they are more complicated to write.
Brew (Binary Runtime Environment for Wireless) is a development platform created to run between the application and the wireless device's chip operating system; therefore BREW allows programmers to develop applications without bothering themselves with system interface or networking details. However, to develop a BREW application it is necessary to submit the application for testing, and this introduces a significant additional cost (both in terms of time and money).
Flash Lite is a lightweight version of Adobe Flash Player optimized for mobile phones and consumer electronics devices. This approach is ideal for applications that massively use audio/graphics features and is becoming very popular as it has been adopted by several cellphones operators, but currently, the main drawback is that applications are not capable of communicating with technologies like Bluetooth and infrared.

Far from identifying the best platform, we simply note that, from the mobile gaming market point of view, a unique platform would be necessary. However, it is unlikely that the platform fragmentation problem will find a solution. What is very likely to happen are porting solution to automatically move from one platform to another. A first step toward porting solutions is *alcheMo*, an automated Java ME-to-BREW porting solution developed by Innaworks.

TECNOLOGIES

Currently, mobile devices are more and more similar to hand-held computers, with increasing processing power, considerable storage space and interesting multimedia features. However, to transform a mobile device into a mobile gaming console, graphics features and networking technologies need to be improved.
In the near future, free-of-charge networking technologies like Wi-Fi and Bluetooth will be available also in entry-level mobile devices, causing the network latencies to decrease (a latency below 150 ms is mandatory to support interactions). In such a scenario, gaming applications should automatically choose the most appropriate communication technology in a tranparent way from the user point of view.
Location technologies will play an important role in gaming applications and hence technologies like GPS and RFID are likely to be available in entry-level mobile devices.
3D Graphics should be supported; A first step toward this direction is the OpenGL ES (OpenGL for Embedded Systems) project, which is a subset of the OpenGL 3D graphics API designed for embedded devices such as mobile phones, PDAs, and video game consoles (see, Group (2007)). Also the Java community is working to provide 3D graphics in mobile devices with the Mobile 3D Graphics API (M3G for short).
Multimedia technologies should better support interactions among users (the usage of the small keyboard is a real burden). A step toward this direction has been done by *Pathway to Glory* a world war II game that makes use of voice communication to allow gamers to send voice messages to each other during the game. VoIP is a mature technology than can be embedded in mobile games.

DELIVERY MODELS

Currently, cellphone operators are offering customers an easy way of downloading mobile games (in US, on-portal mobile game revenues account for 74 percent of total mobile game revenues Telephia (2007)), but the main burden in downloading mobile applications is the cost of the data traffic.
In the future, with more complex mobile games, this downloading cost will be a major problem and hence this delivery model is likely to be coupled with bricks-and-mortar retailers, where mobile games can be sold on different supports like multimedia memory cards. This different delivery model will account for 9.1 percent of the total global revenues for mobile games by 2010[1].

[1]http://www3gcouk/PR/May2005/1459htm

MOBILE GAMES

Today, with an average play out of 11 minutes (see, Telephia (2007)), the most successful games are the so-called *casual* games (a.k.a. *snack* games) , while the popularity of multi-player games is still quite limited.

Casual games are very simple to play since they are based on very simple rules, basic techniques, simple strategies and do not require special skills. These games are played in short bursts, during work breaks or, in the case of portable and cell phone games, on public transportation. Due to their simple characteristics they can be played on the majority of current cell phones and hence they are immediately available to *casual consumer*, people who cannot be defined as typical gamers, instead, they play games when they come across them. *Tetris*, one of the best sellers game in the mobile gaming scenario, is an example of casual game.

Multi-player games allow thousands (or even more) players to play at the same time, but currently, the mobile versions of these games are not very popular. As previously mentioned, the network latency is the main burden for the popularity of these games, as well as the fact that the data mobile traffic is quite expensive. By 2010, online multiplayer games will generate 20.5 percent of total global revenues.

As recently happened, mobile games will be developed exploiting the characteristics of the mobile device: they will use advanced graphics and advanced communication technologies, but also social aspects will play an important role in the development of mobile games.

Due to the success of Web 2.0 social applications in the Internet world, it is very likely that the **social aspect** will be part of many mobile games of the near future. As an example, you can think of games similar to the popular *Second Life*[2], a game that emphasize the social aspect of a multi-player game.

Mobile games based on **location technologies** will be part of the mobile gaming market. An example is *Pac-Manhattan*[3], a game that aims at creating a real live version of Pacman around Manhattan. Although this game is simply based on mobile phones to locate users, it shows the potentiality of these games.

Proximity games are an interesting field of next-generation multi-player gaming, as mentioned during the PlayStation Portable presentation (see, Ackerman (2004)). These games make use of close-range wireless networking technologies (e.g., Bluetooth and Wi-Fi) and they differ from classic MMOGs as they require players to be close in space and also differ from classic location-based games since they don't require the knowledge of the player's absolute or relative position, but the knowledge of proximity is sufficient (see, Sderlund (2005)). Note that the proximity is not always referred to player, but can also be referred to objects.

[2]http://secondlife.com/
[3]http://pacmanhattan.com/

CONCLUSIONS

In this paper we analyzed different aspects of the current mobile gaming scenario in order to identify possible directions. As a result, it is likely that the games of the near future will include social aspects, location-based features, alternative delivery models and multi-player capabilities, moving the target on hardcore gamers.

ACKNOWLEDGEMENTS

This work has been partially supported by the Italian M.I.U.R. under the MOMA initiative.

REFERENCES

Ackerman K., 2004. *Sony on Hardware: PlayStation 2 Add-Ons and the PSP. http://wwwfrictionlessinsightcom/Articles/ GDC2004SonyKey/GDC2004SonyKeyhtm.*

Gibson B., 2006. *Casual Gamers and Female Gamers to Drive Mobile Games Revenues Over the $10 Billion Mark by 2009.* In *http://www.juniperresearch.com/shop/ viewpressrelease.php?id=19&pr=16.*

Group K., 2007. *OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. http://wwwkhronosorg/opengles/.*

Koivisto E.M.I., 2006. *Mobile games 2010.* In *CyberGames '06: Proceedings of the 2006 international conference on Game research and development.* Murdoch University, Murdoch University, Australia, Australia. ISBN 86905-901-7, 1–2.

Sderlund T., 2005. *Proximity Gaming: New forms of wireless network gaming. http://wwwdifferentgameorg/detailasp?item=672.*

Telephia, 2007. *Preloading Game Demos is a Key Merchandizing Lever to Drive Purchases.* In *http://www.telephia.com/html/ GDC07_press_release_template.html.*

SIMPLE, CHEAP AND QUICK: THREE URBAN GAMES FOR COMMON MOBILE PHONES

Helena Karsten, Jan-Erik Skata and Sebastien Venot
Department of Information Technologies & TUCS, Åbo Akademi University
Joukahaisenkatu 3 A, 5th floor, 20520 Turku, Finland
e-mail: eija.karsten@abo.fi

Nhut Do
ICT Turku
Itäinen Pitkäkatu 4 B, 20520 Turku, Finland

Janne Konttila and Joonas Peltola
Zetanol Ltd
e-mail:firstname.lastname@zetanol.fi

KEYWORDS

urban game, mobile phone, Bluetooth, game development

ABSTRACT

We designed and implemented three location based games for mobile phones. The overall design principles were: (1) design for device platform with wide penetration; (2) create an architecture that supports different types of games and services, scalability; and (3) aim for cost efficiency and quick application development. The resulted three games are different in nature, but each of them introduces a potential design approach for future pervasive or location based games. Player feedback also supports further development. We see several opportunities for extending our ideas from gaming to various other location-based applications.

INTRODUCTION

Location based gaming and pervasive gaming take into account the physical characteristics of the real world as well as the computer-maintained virtual game environment. Also, they usually have a strong social aspect.

Today, the characteristics, features and technology platforms in pervasive gaming are still often prototypes resulting from ambitious long-term research projects (such as IperG, Uncle Roy All Around Us, or Human PacMan), and thereby complex and, quite likely, also expensive. Prototypes made for testing the ideas are often comprehensive and well designed from the game point of view. By mixing manufactured and custom-made devices together with advanced game logics, the result can be an immersive game experience breaking the bounds of traditional computer games. When the technologies intended to be utilized in later, preferably commercial, game realizations are still under development, they seldom are accessible to potential game adopters. Also the robustness of technologies in the making may be less than optimal.

The challenge is therefore to make game realizations that are actually adoptable by users with those devices they already have. The main obstacle is the use of multiple devices, and the one-purpose nature of the device combinations. Thus, we decided to ground our project on simplicity, low cost, and quick development. In this paper we will describe the case of developing location-based systems for three different games.

First, the background for our project is briefly discussed, and the design principles are outlined. Then the game development project and the technical architecture are introduced. The game realizations are analyzed briefly including feedback from the players. We conclude by discussing the strengths and weaknesses of our approach, ideas for future work and the business opportunities foreseen.

BACKGROUND AND PRINCIPLES FOR GAME DESIGN

Pervasive games often include location awareness or other elements from physical world. Also, the context of player and perhaps qualities of the environment are taken into account. Some, but not all, game objects may be physical. Some actions or game events take place in a virtual world, some in the real world (Magerkurth et al 2004). Access to the game world can happen with use of various devices. In pervasive games, the game experience can be trans-medial: inputs and outputs between a player and the game system can occur on multiple different media. This emphasizes the role of the player as an interpreter of information from various sources (Walther 2005). These approaches emphasize the technological as well as the social aspect of pervasive gaming, originating from the concept of pervasive computing. Montola et al. (2006) use the classic definition of play by Huizinga (1938) as basis for defining pervasive games. Huizinga proposes that play is playful, not serious, voluntary action that is distinct from everyday life in terms of time, space and people. A game occurs in a magic circle of certain place, certain time with certain people. Montola et al. then define a pervasive game as a game that extends beyond this circle socially, spatially or temporally.

Our game development project was stimulated by the idea of expanding the game experience to be part of everyday life, not just a separate activity. Today, virtually everyone is carrying a device capable of running games and other multimedia applications – a mobile phone. When the game device is equipped with data transmission capability, a multi-player network game is one obvious development trend. Further, if the playing occurs when the players are mobile, their location could be used as a factor in the game state at any given point of time. Likewise, the game moves can be made dependent of physically visiting certain physical locations.

As key design principles we emphasize the following three:
1. Design for a device platform with a wide penetration.
By designing applications for a device platform with a large user base, the applications are more likely to spread out and gain popularity. With the large user base, a multiplayer game gets a bigger social factor. Further, the threshold to try the game or other application is significantly lower if it does not need the purchase of a new device.

2. Architecture that supports different types of games and services, scalability.
From the beginning, we saw it necessary to design a basic architecture that would suite different types of location based game models and other services, offering different kinds of experiences. Although the main idea was to design a model for a persistent multiplayer game, we found room also for solo gaming and event-based one-time experiences.

3. Cost efficiency and quick application development.
The project team had previous experience in J2ME programming, so beginning with it was the obvious choice. We also wanted to harness the players own imagination in the game play experience. No expensive 3D modelling was implemented, and the users were provided mainly with textual information. The hardware infrastructure followed the principle of cost efficiency, consisting of standard mobile phones and battery powered Bluetooth beacons. Since the data transferred during the game play was mainly textual, the network rates were not a problem.

Game play is a natural motivator to participate in something that is not immediately necessary or beneficial. A game that coordinates the public to do things with useful side effects would allow gathering large amounts of information from large geographical and social space. By controlling game events and perhaps game logics and rules, the agencies that ultimately use the gathered data can steer the players to do tasks supporting their needs.

We agree with the suggestion of Capra et al. (2005) that pervasive games can be used to support research groups who use, for example, environmental data. With appropriate devices and networks, ordinary people could collect field data by means of game play. The game would keep the lay people interested in a continuous effort. This game must be easy to participate in. The devices used should be either very familiar to the players (like their own mobile phones) or easy to manipulate. The game rules should not be overly complex, as the play will take place in a variety of environments. At the same time, however, it is crucial that the researchers do get the data they need from where they need it. The amount of data received in this way could be very large. Even though the experts would guide – via the game rules and feedback – the players, there is still a likelihood of getting less than optimum quality data. The data would need cleaning, but the replicated data items can help in this. For a future research project making use of this idea, we found it necessary to begin a game development project. However, the game project was soon considered a separate project.

GAME DESIGN PROCESS

The idea of using game play as a motivation for players to participate in something not directly beneficial, or harmful, for them, guided the design of the first game. The aim was to create a game that would lead players to certain places repeatedly. The game system should be stand-alone, run the game and guide players automatically. Also, it should allow intervention by administrative personnel to redefine the important places in the game. Ultimately, we designed three games based on the same basic architecture. Each design and development cycle started from a game concept idea.

Game Logic Design

The idea developed through a simple resource management game: Players should invest their game credits to physical places by actually visiting them. Then, the investment starts to pay back in form of interest. This interest can later be re-invested. The investment can be lost if another player pays more for the same place. To protect the investments made, the players need to once in a while visit also places they already hold, in addition to investing in a new place. That way the prices can be kept up and the investments secured. The players are organized in teams to ensure that even newcomers get similar resources to other players. A persistent game would then be a continuous battle of domination of most game spots.

In the first one, Turku-game, the original idea was left aside and the goal for the players was to solve a murder mystery, aided by a virtual private detective. The game narrative was based on the works of a local novelist. The game took place during the Arts Night in Turku, a medium sized city in South-Western Finland. The players visited a number of attractions during the evening. Once a player detected a hotspot, she or he received a question related to the site. After replying to the question, the player received a clue about the mystery. Each clue helped to get closer to solving the murder.

After the success of the Turku-game, the original game idea was implemented and named Conquer the Quarter. The game idea was to eliminate other players by conquering all the quarters marked by Bluetooth beacons or by buying them out. In the game, each Bluetooth-tagged hotspot had a certain value in the beginning and they were all distributed in corners of the game area. A player could freely increase the value when capturing the corner by investing in it. Other teams then had to pay this new value to capture it. The investment also paid back in interest. The longer the player owned the corner, the more money the team earned. By capturing corners and making wise investments, teams would soon have a property that earns quickly, ultimately enabling the team to conquer the whole quarter. Conquer the Quarter was played in Manhattan, New York as a part of the "Come Out and Play Festival 2006".

The game concept for a third game, the Gnome-game, was designed combining the ideas of the earlier games, but it was never played. According to the plan, there are named hotspots on the map. The player first goes to one of them, and gets a short explanation or a story about the current spot. The player is then asked to take a picture, and finally receives a hint to find a nearby hidden spot. Once the player finds this hotspot, they either get a new hint, or they can proceed with the hotspots already marked on map. At each hotspot, they take a picture, and get short story that links to the current location. The aim is to visit as many hotspots as possible and document the adventure by taking pictures. Pictures are published in real-time on the web. The aim of this game was to work as a guide showing different paths between interesting places in the Turku city centre during Christmas time. Also, the players, by taking pictures, would produce material to the Web.

As in earlier realizations, the game spots were to be digitally marked using Bluetooth radio beacons. The player has to be in physical proximity of a particular beacon to make a move in the game. When in the location, the client application prompts the user with sound and vibration, and a context-based screen appears. The application suggests the player to make a game move that is possible at the current game state.

Software Design

All the games share the same architecture with five elements: Bluetooth beacons, mobile phones with game software (client) installed, service provider network (GPRS), the game server, and a database.

Each Bluetooth beacon consisted of a standard USB Bluetooth adapter, wired up to a custom-made external power adapter that enables running them stand-alone. Beacons initially need to be activated while connected to a PC, and they can then be unplugged. From that point on, they will be running and transmitting the needed signal so that they can be detected by the phones.

The client software runs on the mobile phone. The game platform is Java 2 Micro Edition (J2ME). The minimum requirement for the game to work properly is to have a Java-enabled mobile phone that supports J2ME MIDP 2.0/CLCD 1.0, Bluetooth API, multimedia API (for the camera) and a GPRS connection. The game uses the GPRS connection to communicate with the game server; therefore a service provider supporting this is needed. The game management is run on the server with the help of the database.

THE GAME EXPERIENCES

Turku-game

When playing takes place among other people in an urban environment, the players cannot be expected to have their focus only on the game and the game device. The first game realization, "Turku-game" was tied to events of the Arts Night. This gave a fruitful yet laborious approach to game design. We tied the narrative to the events of the night and aimed to design a mixed reality interactive story that would expand the overall cultural experience.

The Arts Night, when the streets, galleries, restaurants, bookstores and others were full of events, gave an ongoing atmosphere for the game experience. Deeper immersion in the game was reached when the game story and virtual events were tied to ongoing live events. Even if playing the game was for some players the main activity, they ended up to places and events they otherwise would not have found. So, in addition to the gameness, the game worked as a guide to the evening.

The players started all at same place, where they were given the game devices and briefed about the idea of the game and told about the events that had taken place, i.e., that a performance artist had been murdered after a play in an art gallery. They also got a printed map, where the game starting points – the murder scene and the current location of some key witnesses – were marked. The organizers guided them how to receive a first clue and to get the hang of how to solve the murder mystery. Then, the players started the adventure at their own pace. At the game hot spots, players got further clues that helped them to build a big picture of the mystery, and to find new hot spots with new hints. Two of the 15 hot spots of the game were carried by actual persons, who then gave the required information to the players. Some of the clues in the game required awareness of the other events during the night. The clues sometime referred to an event, not directly to a place.

If participants dead-ended in the game, they had a phone number for "private detective Vares" who would help them out. He was also helping in technical issues that were expected since the game was a prototype. During the game, "Vares" received about a dozen calls. Half of them were about game situations, the other half technical issues.

The overall experience was positive. As the organizers recommended before the game, the players enjoyed also other events of the night. About half of the players were able to figure solve the murder mystery. A couple of days after the experience we approached the 17 players with questionnaire about the game event and received fourteen answers. The questionnaire was semi-structured leaving room for players to freely describe their experience and thoughts about it (see Table).

Table. Evaluation questions

Before the game:

1.From where did you hear about the game?
2.What did you expect the game to be like?
3.Was the guidance given prior to the game sufficient? If not, what was missing?

During the game:

4.With your own words, describe how you game play went and what did you do during the game?
5.How much did you communicate with other players during the game? Did you get help from them?
6.Did you play alone or with company?
7.Did you talk about the game with non-players during the game?
8.Did you manage to immerse to the game? What affected this the most?
9.What kind of technical problems did you experience?

After the game:

10.What are the most important things you remember from the game?
11.What was the most fun thing in the game play?
12.What was the worst thing in the game play?
13.How did you experience this kind of combining mobile gaming with a physical playing environment?
14.How would you suggest this kind of game to be developed?
15.Would you be interested to play similar, but further developed game in future?
16.Other comments

Many told that they found interesting events because the game led them to places they otherwise would not have found. On the other hand, some claimed that the game felt like just running from one place to another, and not true gameness existed. They yearned for more complex tasks or tricky puzzles to solve to go ahead, whereas the murder could be solved by collecting enough information by visiting a sufficient number of spots.

The technical problems were another issue. A few players had continuous problems with detecting the Bluetooth beacons. This was really annoying, and made the experience nothing but trying to figure out why it does not work. This also reflected on the player-to-player interactions: most of the conversations between players were about the technology and the whereabouts of the hard-to-find beacons, not the game story or the experience. In a few cases, the players were frustrated while they knew they were at right position, but the hotspot was not detected in the timeframe they expected. Some players were more sensitive to latencies in the system, while some others took it calmly and had a drink near the place they knew was a hotspot. In one game location, the Bluetooth beacon had been removed by the Arts Night staff, because some person (not a player) had mistaken it for a bomb! In another case, the Bluetooth dongle was unplugged from the power supply, thus it stopped working. These incidents caused some players to dead-end in the game.

CONQUER THE QUARTER

The players of Conquer the Quarter were participants of the Come Out and Play urban game festival in New York and therefore quick to get the idea. The game begun with a briefing, where players were provided with a game device (a mobile phone) and a paper map with the 15 game corners marked. Similar information was also in electronic format available through the user interface of the game application, but due to the small screen size and bright sunlight, the paper map was found to be more comfortable.

The players were divided into three groups. One group had four players and the other two had three. All players were in their mid-twenties. The game involved running competitions, spontaneous strategy meetings, and laughing. The fast-paced game got its culmination right after half the time was spent: the green team had lost all of its corners and most of the game credits. From that on, the blue and red teams fighted for the domination of the Quarter.

The game parameters were fixed to support approximately one hour game. The adjustable game parameters are the number of game corners, interest rate and the length of the interest cycle. In this case, the rate was 10% interest plus 1% for each corner the team was holding. This rule was added after early tests where some players ended up investing all the credits in one single corner in the beginning, and later buying everyone else out. Without this rule and with this strategy, there would not have been challenge left (cf Juul 2005). The interest cycle was set to one minute. That way the teams had continuous credit flow (in case they had made investments), and the game pace was kept fast.

As the game was set up in an area covering a few city blocks, it was suitable for an intensive game. The players, however, agreed that one hour was too short, and that the area was too small. They also would have preferred to play this kind of a game in a more populated area. All the players had a very enthusiastic attitude, and as the teams were formulated, they liked the idea of leaving the communication between players on face-to-face basis. However, some players said that after a while, when the teams had begun to conquer the game spots, it felt that a little bit more complicated game logic would have given extra excitement to the game. One suggested a model of ability to build combinations of corners, like three in a row or similar and giving something extra if the team managed to do so.

As we expected, in this type of a game, when the actual game events happened occasionally and the player did not need to stare the game device all the time, further simulation was unnecessary. Thus, the game could be defined as mental game (Nilsen et al. 2004) where the players have to resolve all the simulation themselves. One player suggested giving the virtual corners some qualities of the real world to strengthen the tie between real and virtual worlds.

CONCLUSIONS

We were not able to play the Gnome game in public, but the prototype has been demonstrated to several audiences, with interested feedback. We are now working in our new company, Zetanol, on several further projects with local partners to create urban games to suit their particular needs.

Complex games with high computing capacity consumption, and need for long-term, intensive attention, are not ideal entertainment for all players. A growing number of players are so called casual players, who are not willing to spend neither much time nor money to games or playing. The age and gender of playing audience will diversify. It can be proposed that easily accessible and playable games with little need for time will gain in popularity. Furthermore, digital games could learn from traditional tabletop board games when it comes to the social domain of game experience. To take heed of this, an innovative approach is needed, and a current idea of what is a game is perhaps to be left behind.

With our project, we wanted to contribute to the pervasive games research from our own vantage point. Our aim was to design a location based game that would enable a mixed reality experience by using a common mobile phone as a game device. We had made it clear to ourselves that use of any combination of devices, such as palmtop computer and GPS (Global Positioning system) device, was not an option. The use of Bluetooth was chosen to make the location-based events possible.

Creating a mixed fantasy (Nilsen et al 2004) experience with a combination of virtuality, reality and imagination, is not an easy task. When the qualities of physical surroundings are claimed to be tied to the computer-maintained game world, the players expect such a connection to exist, but delays and other technical issues might loosen this connection, and the immersion suffers. Also, if advancing in the game requires combining information from the physical and the virtual worlds, reconciliation of those two has to be smooth. A mixed-reality game environment would require a much more careful game design than we were able to evoke during the rather fast-paced development project. We claim that a coherent mixed reality environment might not be needed, after all, to create a game offering a new kind of interesting experience and even a model for mobile entertainment. Also, we argue that a location-based game can introduce a story, and the player can actually be part of it, contrary to what has been claimed earlier (Rashid et al 2006).

Pervasive games can be seen as socially, spatially or temporally expanded games. The game prototypes introduced here do not fully meet these challenges. They already meet the challenge of spatial expansion and they have the potential to be either persistent in time or unlimited in the number of players. Our model can be scaled up to harness a wider area of physical space to game play. A future research question would then be how a game could be set to a city centre in a persistent manner: for example, would the popularity of certain places in the physical city affect the popularity and virtual price of the virtual corners? We also consider players as potential writers of stories and mysteries.

A general issue concerning our approach is also how to create a design that reacts to the real world environment and causes a shift in the game in a manner that would guide players to new locations. Our three cases, we propose, give one example of possible direction for future game development. By creating simple, interesting and a little bit tricky games that give something extra to the players' everyday environment, it might be possible to interest and engage them to a new type of gaming. This can be a starting point of creating new types of mobile services and business models.

ACKNOWLEDGEMENTS

We thank the organizers of the Turku Arts Night and the New York urban game festival for inviting us to try our games. We also thank several offices of the city of Turku for many kinds of support and encouragement. Special thanks to Nokia Research Centre for providing the devices for the first experiment.

REFERENCES

Huizinga, J. (1938). *Homo Ludens*. Basel.

Capra, M., Radenkovic, M., Beford, S. Opperman, L., Drozd, A. & Flintham, M. (2005). "The Multimedia Challenges Raised by Pervasive Games." *MM'05*, November 6-11, Singapore: ACM. 89-95.

Juul, J. (2005). *Half Real: video games between real rules and fictional worlds*. Cambridge: MIT Press.

Magerkurth, C., Engelke, T., & Memisoglu, M. (2004). "Augmenting the Virtual Domain with Physical and Social Elements: Towards a Paradigm Shift in Computer Entertainment Technology." *ACM Transactions on Computers in Entertainment*, 2(4), Article 5b.

Montola, M., Waern, A., & Nieuwdorp, E. (2006). "Domain of Pervasive Gaming." Deliverable D5.3B. *Integrated Project on Pervasive Gaming*. January 2006. Retrieved 14.3.2006 from http://www.iperg.org

Nilsen, T., Linton, S., Looser, J. (2004). "Motivations for Augmented Reality Gaming." *New Zealand Game Design Conference,* 26.-29. June, Dunedin, New Zealand.

Rashid, O., Mullins, I., Coulton, P., & Edwards, R. (2006). "Extending Cyberspace: Location-based Games using Cellular Phones." *ACM Transactions on Computers in Entertainment*, 4(1), Article 3C.

Walther, B. (2005). "Atomic Actions – Molecular Experience: Theory of Pervasive Gaming," *ACM Transactions on Computers in Entertainment*, 3(2), Article 4B.

ONLINE GAMING AND SECURITY

Towards Swift and Accurate Collusion Detection

Jouni Smed Timo Knuutila Harri Hakonen

Department of Information Technology
FI-20014 University of Turku, Finland
{jouni.smed, timo.knuutila, harri.hakonen}@utu.fi

ABSTRACT

Collusion is covert co-operation between participants of a game. It poses serious technical, game design, and communal problems to multiplayer games that do not allow the players to share knowledge or resources with other players. In this paper, we review different types of collusion and introduce two measures for collusion detection. We also propose a model and a simple game, implemented in a testbench, for studying collusion detection.

INTRODUCTION

When the rules of a game forbid the players to co-operate, any attempt of covert co-operation is called *collusion*. The players who are colluding (i.e., whose goal is to win together or to help one another to win) are called *colluders*. Collusion poses a major threat to games that assume that the players aim at individual goals individually, because many types of collusion are impossible to prevent in real time. Even detecting collusion can require discerning and understanding the player's motivation – which is often an impossible task for human beings, too. For this reason, collusion is usually detected only afterwards by studying the behaviour of the players and recognizing characteristic patterns that indicate forbidden co-operation.

Apart from games suspectible to collusion such as poker [2, 8, 10] and bridge [11], collusion have been addressed also in the context of tournaments [3] and multiple choice examinations [1]. In our previous work [7] we introduced a classification for different types of collusion, which we will present in the next section. We argued that different types of attacks have been lumped together under the same collective title "collusion" and that they have been commonly dismissed as unsolvable in the literature. We showed that although there are collusion types that are indeed impossible or very hard to detect, there are also cases where automatic recognition is possible. In this paper, we take one step further and present a model and a simple game with which collusion detection methods can be tested and evaluated. Our motivation is that only when we understand how to detect collusion, we can proceed further to its prevention.

The plan of this paper is as follows: We begin by presenting classifications for collusion. They line out the types of collusion and give us the terminology that we will use throughout this paper. After that, we look at the problem statement of collusion detection. It gives us measures upon which the models and testbench game presented next will rely. Finally, we will conclude the paper with a discussion on how the model presented in this paper helps the research and what are the steps for future work.

CLASSIFYING COLLUSION

When the players of a game decide to collude, they make a agreement on the terms of collusion [7]. This agreement has four components:

Consent How do the players agree on collusion?

- *Express collusion*: The colluders make an explicit hidden agreement on co-operation before or during the game.
- *Tacit collusion*: The colluders have made no agreement but act towards a mutually beneficial goal (e.g., try to force the weakest player out of the game).

Scope What areas of the game the collusion affects?

- *Total collusion*: The colluders co-operate on all areas of the game.
- *Partial collusion*: The colluders co-operate only on certain areas and compete on others (e.g., sharing resource pools but competing elsewhere).

Duration When does the collusion begin and end?

- *Enduring*: Collusion agreement lasts for the duration of the game.
- *Opportunistic*: Collusion agreements are formed, disbanded, and altered continuously.

Content What is being exchanged, traded, or donated in the collusion?

- *Knowledge*: The colluders share expertise (e.g., inside information on the game mechanics), in-game information (e.g., the colluders inform one another the whereabouts of the non-colluding players) or stance (e.g., the colluders agree on playing "softly" against one another).
- *Resources*: The colluders share in-game resources (e.g., donating digital assets to one another) or extra-game resources (e.g., a sweatshop is playing a character which will be sold later for real-world money).

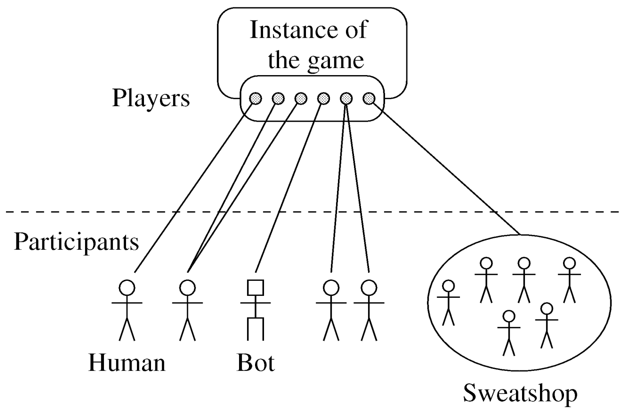


Figure 1: Players and participants are the partakers of a game. The relationship is usually assumed to be one-to-one, but one human participant can control two or more players, a player can be controlled by a computer program (i.e., a bot), or two or more participants (e.g., a sweatshop).

This classification is not sufficient for on-line computer games, because we must also discern the roles of the partakers – players and participants – of the game [7]. A player in a game can be controlled by one or more participants, and a participant can control one or more players in a game (see Figure 1). This means that there are two types of collusion: (i) collusion among the *players* which happens *inside* the game, and (ii) collusion among the *participants* which happens *outside* the game. To detect player collusion, we have to analyse whether the players' behaviour diverges from what is reasonably expectable. To detect participant collusion, we have to analyse the participants behind the players to detect whether they are colluding.

This gives a fine-grained classification of collusion types:

Participant identity collusion How a single player is perceived to participate in a game?

- *Player controller collusion*: Many participants are controlling one player (e.g. two players controlling the same character alternatively).
- *Self-collusion*: One participant is controlling many players (e.g. one participant controls many player in a poker table).

Inter-player collusion How the participants are affecting the game?

- *Spectator collusion*: Co-colluder provides a different type of information (e.g., ghost scouting, post-game information).
- *Assistant collusion*: Co-colluder plays (possibly sacrificingly) to assist the other to win (e.g., as a sidekick, passive scout, or spy).
- *Association collusion*: Colluders achieve individual goals through co-operation.

Game instance collusion How factors outside the game instance affect the game?

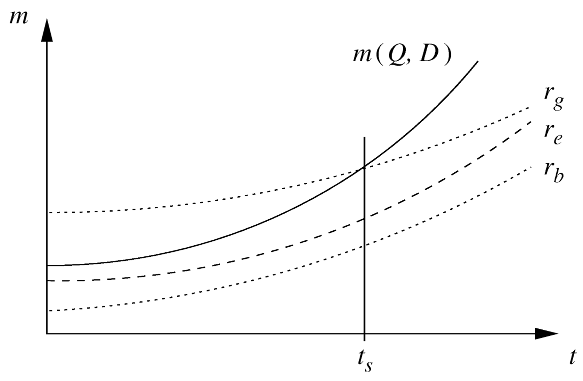


Figure 2: Collusion is detected when the observed results using a measure m deviate significantly from the expected results r_e. Suspicion arises at the moment t_s when the results are getting either too "good" and cross the threshold r_g or they are getting too "bad" and cross the threshold r_b.

- *Multigame collusion*: Players of different game instances collude (e.g., studying the game properties, finding suitable server, fixing tournament match results).
- *Insider collusion*: The co-colluder is a game administrator or game developer that reveals or modifies the workings of the game instance.

Because collusion prevention requires that collusion gets first detected, let us next take a closer look at what is required from collusion detection.

COLLUSION DETECTION

When comparing collusion detection methods, we should observe the following two properties:

Accuracy How justified is the suspicion raised by the detection method?

Swiftness How early does the suspicion raise?

Naturally, accuracy is important so that normal behaviour does not set off an alarm and cause uncalled for inspection or unjust punishment. Swiftness is usually related to accuracy so that the less accurate the detection is, the swifter the suspicion is detected.

Let us try to interpret these properties in a somewhat more formal – but simple – manner (see Figure 2). Suppose that our detection is based on applying some numeric function m upon the participants P of the game and some collected game data D. Let $Q \subseteq P$ and r_g is some chosen threshold value of the best possible play. If $m(Q,D) > r_g$, we decide that the players in the set Q are colluding. In this framework the questions to be asked are:

Accuracy How is the value of m related to the probability that Q really contains colluders?

Swiftness How much data D is needed before r_g is exceeded?

Ideally, we would like to have a measure that indicates as early as possible when players are colluding or when their

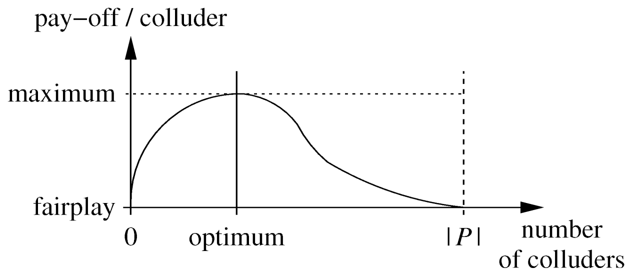


Figure 3: The pay-off of collusion per colluder increases until the optimum number of colluders is reached, after which it approaches asymptotically fairplay pay-off.

behaviour is showing suspicious traits. Should the detection happen before collusion actually gives any notable gain for the colluders, we have managed to prevent it altogether. How then to find such methods? From an intuitive point of view, any abnormal behaviour in a game should raise a suspicion. This is the case especially when some of the players get too good (i.e., exceeding the threshold r_g) or too bad (i.e., going under the threshold r_b) results in comparison to their playing skills (the latter would indicate a case of assistant collusion). Function m could then indicate the (absolute) difference between the expected behaviour (e.g., wins in a card game) against the observed one.

How to select Q then? Instead of inspecting all $|\wp(P)| - |P|$ different colluder sets, we can limit $|Q|$ to a certain range, which depends on the collusion pay-off of the game. Figure 3 illustrates the pay-off of collusion with respect to the number of colluders. As the number of colluders increases, the total amount of pay-off also increases. However, when the pay-off is divided among the colluders, there exists an optimum where the pay-off per colluder is the greatest. For example, robbing is more effective when the gang of robbers is big, but a big gang of robbers has to focus on big heists to provide everyone with a big share of the loot. When we are detecting colluders, $|Q|$ can be limited near to this optimum. For the game design this means that it is possible to discourage large-scale collusion by pushing down the peak of the curve. For example, if robbing is allowed in the game but a part of the loot gets damaged (or otherwise loses its value), the optimum size of a gang of robbers gets smaller.

Next, let us limit ourselves to inter-player collusion, where the players of the game co-operate by exchanging in-game resources or information. This type of collusion is what is "normally" understood as collusion, where we can assume players and participants have one-to-one relationship. For a review of methods proposed for preventing other types of collusion, see [7].

INTER-PLAYER COLLUSION

If the content of the collusion agreement is an in-game resource, it is possible to detect by analysing the game session logs [2]. Detecting shared knowledge, however, is more difficult, because we can only observe the decisions made by the players in order to discern the intention behind the decision-making. To analyse this kind of collusion, we present a simple game, *Pakuhaku*, in the next section, but before that we need to consider two attributes of a game.

The first attribute divides games into *perfect information games* (such as chess), where the players can always access the whole game situation, and *hidden information games* (such as poker), where the players can access only a part of the game situation [6, §4]. Naturally, hidden information is worth colluding, because it gives the colluders benefit over the other players. But collusion is beneficial even in a perfect information game, because the decision-making process can always be seen as "hidden" information.

The second attribute is based on the properties of the game world, which can be *continuous* or *discrete*. If the central attributes of the game world are continuous, there usually is a well-defined metric to compute the distance between two game world locations. Since players try to dominate some geometric sub-area of the game world, the winnings of the game are related to the scope of the dominated area. Collusion can allow the players to govern a larger area than they would obtain by individual effort alone. If the game world consists of a set of discrete locations, the colluders can try to increase their joint probability of winning in the game by maximizing the subset of states they dominate.

When we consider the measuring and estimating collusion in some game environment, we could start by collecting real-world data for the purposes of analysis. However, it would be hard to ascertain what has been the driving force behind the human players at a given time. Another approach is to use synthetic players [5] some of which have been programmed to collaborate. Clearly, it is easier to create a large amount of test data with known co-operative properties with the latter approach, and we believe that it is the more fruitful one in this early phase of this research. The results obtained for artificial data should naturally be later evaluated and verified using real examples.

The idea behind our approach is:

(i) Generate game data with different number of players, colluders, game types, and collusion strategies.

(ii) Devise detection methods.

(iii) Run the detection method against the data to get results.

(iv) Compare accuracy: How many (if any) of the colluders got detected.

(v) Compare swiftness: How quickly the colluders were detected.

Naturally, this creates a competition surroundings where creating colluding synthetic players fights against devising detection methods.

In this paper, we limit ourselves to step (i). The subsequent steps will naturally be the focus of the our future work. Moreover, we intend to provide ready-to-use test data (akin to the Calgary Corpus [9]) for anyone interested in developing and testing their collusion detection methods as well as the possibility to fine-tune the synthetic players and develop new game types using our testbench system, *Pakuhaku*, which we will describe next.

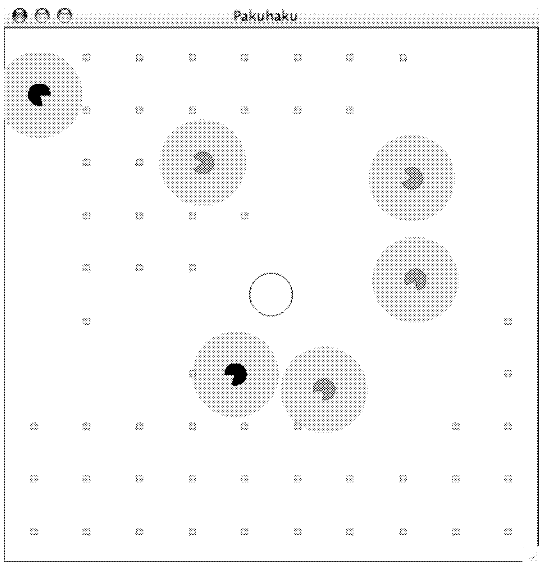


Figure 4: Screenshot from Pakuhaku with fog-of-war. The black players are colluding by dividing the game world into non-overlapping interest domains, while grey players search pills individually.

TESTBENCH FOR COLLUSION DETECTION

The basis for our testbench, *Pakuhaku* (see Figure 4), is the classical computer game *Pac-Man* [4]. We have omitted many features of the original game – such as the maze, ghosts, and power-up cherries – but we allow multiple players to take part in the game. Moreover, we have parameterized the number of directions the players can take (which ranges from three to infinite) and the area visible to the players can be limited by a fog-of-war. The goal of the game is simple: eat as many pills scattered in the game world as possible.

At each turn, each player makes a decision on the direction where to go. This decision is based on knowledge about the game world, which can be perfect (i.e., not limited by the fog-of-war) or hidden (i.e. limited to immediate surroundings by the fog-of-war). The system provides a communication channel, where the colluders can exchange one message in each turn. The communication can be used to assist, restrict and guide the co-colluders.

The game type can be one of the following:

- *Preset game world*: A given number of pills are positioned in the game world. The game ends when all the pills have been eaten.
  - Evenly distributed pills: The pills are positioned into rows and columns.
  - Randomly distributed pills: This pills are positioned randomly from a given distribution.
- *Regenerating game world*: Pills are repositioned to game world after they have been eaten. The game ends when the leader has eaten a given number of pills.
  - Dispenser competition: The game world has only one pill, which is dropped into a random position whenever it gets eaten.
  - Triple competition: A variation of the dispenser competition, where three pills are dropped randomly in a line somewhere in the game world.

Table 1: Results from 1,000 game instances with 100 randomly distributed pills and a fog-of-war. Of the five players A and B collude whereas C, D and E play fair. (a) The colluders share only knowledge of their whereabouts. (b) The colluders divide the playfield into interest domains. (c) Player A plays normally whereas player B tries to hamper the other players by following and eating pills in front of them.

test	player	min	max	mean	variance
(a)	*A*	6	41	20.38	34.88
	B	5	44	20.14	35.85
	C	5	42	19.91	36.77
	D	3	41	19.87	33.74
	E	5	41	19.69	34.34
(b)	*A*	3	50	21.89	44.31
	B	5	47	20.20	39.36
	C	3	40	19.40	34.21
	D	6	38	19.43	32.92
	E	4	40	19.07	35.42
(c)	*A*	4	48	24.17	43.76
	B	2	37	13.65	27.41
	C	3	47	21.19	56.95
	D	1	46	20.76	62.21
	E	1	55	20.23	58.99

The *Pakuhaku* system runs on the Java platform. The testruns can be done in a batch mode, where the system creates log data for further analysis (see Table 1 for a simple statistical analysis of the log), or the actions can be observed on screen. The player logic (including colluders and non-colluders) is freely programmable. The system can also be extended to include new game types.

DISCUSSION

Let us first consider collusion and the effect of the fog-of-war. If we have a perfect information game (i.e., no fog-of-war), colluders do not get any benefit by informing about the position of the game entities. Instead, they can agree on dividing the game world into non-overlapping interest domains (e.g., as a Voronoi diagram) so that each colluder eats the pills within the respective interest domain (see Figure 4). While non-colluders potentially target all available pills, thus competing with other non-colluders as well as with colluders, the colluders focus only to the subset that belongs to them and avoid competition with other colluders.

If the game has hidden information (i.e., the fog-of-war limits the visible area), the colluders get advantage by sharing the positions of the entities visible to them. This benefit is not as great in preset game worlds as in the changing ones. For example, in the triple competition, if the colluders know the position of two pills, the possible locations of the third pill are limited to a single line.

To present an example how collusion can be detected, let us consider the dispenser competition without the fog-of-war: Let the game area size be A and the number of players

p. Normal players would most likely try to balance between the following strategies:

- Patrol in the middle of the game field to minimize the average direction to a random location.
- Try to find an area that can be dominated and which is larger than A/p (i.e., an area where there are not many other players around).

In either case, whenever a pill drops, the player starts rushing to it. Note that if all players follow the same strategy, they have equal chances of winning ($1/p$). If both strategies are followed by some players, the latter strategy is the more profitable one.

Let there be q collusion participants, who divide the game field into q disjoint interest domains. If all non-colluders are patrolling in the middle, the colluders quite likely get most if not all the pills. Even in the latter case, the even distribution of colluders makes all areas of the game field equally uninviting to normal players, so their decisions will be more or less random.

CONCLUDING REMARKS

Collusion cannot be prevented, but some of its forms can be detected and punished afterwards. Therefore, the countermeasures are effective only if we can detect collusion accurately and swiftly. In this article, we focused on inter-player collusion and presented a simple game where collusion detection methods can be tested. The testbench creates game data that can be used to evaluate collusion detection methods. This paves the way to the future work, which will focus on designing detection methods, analysing them formally and improving them experimentally.

REFERENCES

[1] A. Ercole, K. D. Whittlestone, D. G. Melvin, and J. Rashbass. Collusion detection in multiple choice examinations. *Medical Education*, 36(2):166–172, 2002.

[2] U. Johansson, C. Sönströd, and R. König. Cheating by sharing information—the doom of online poker? In L. W. Sing, W. H. Man, and W. Wai, editors, *Proceedings of the 2nd International Conference on Application and Development of Computer Games*, pages 16–22, Hong Kong SAR, China, Jan. 2003.

[3] S. J. Murdoch and P. Zieliński. Covert channels for collusion in online computer games. In J. Fridrich, editor, *Information Hiding: 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*, pages 355–369, Toronto, Canada, May 2004. Springer-Verlag.

[4] Namco. *Pac-Man*. Namco, 1979.

[5] J. Smed and H. Hakonen. Synthetic players: A quest for artificial intelligence in computer games. *Human IT*, 7(3):57–77, 2005.

[6] J. Smed and H. Hakonen. *Algorithms and Networking for Computer Games*. John Wiley & Sons, Chichester, UK, 2006.

[7] J. Smed, T. Knuutila, and H. Hakonen. Can we prevent collusion in multiplayer online games? In T. Honkela, T. Raiko, J. Kortela, and H. Valpola, editors, *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, pages 168–175, Espoo, Finland, Oct. 2006.

[8] C. Vallvè-Guionnet. Finding colluders in card games. In H. Selvaraj and P. K. Srimani, editors, *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, volume II, pages 774–775, Las Vegas, NV, USA, Apr. 2005.

[9] I. H. Witten and T. C. Bell. Calgary text compression corpus, accessed Aug. 16, 2007. Available at ⟨ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/⟩.

[10] R. V. Yampolskiy. Online poker security: Problems and solutions. In P. Fishwick and B. Lok, editors, *GAME-ON-NA 2007: 3rd International North American Conference on Intelligent Games and Simulation*, Gainesville, FL, USA, Sept. 2007.

[11] J. Yan. Security design in online games. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*, pages 286–297, Las Vegas, NV, USA, Dec. 2003.

Behavioral Biometrics for Recognition and Verification of Game Bots

Roman V. Yampolskiy and Venu Govindaraju
Center for Unified Biometrics and Censors, University at Buffalo
2145 Monroe Ave. #4
Rochester NY, 14618
rvy@buffalo.edu

Abstract- Intelligent bots are quickly becoming a part of our everyday life. Virtual assistants, shopping bots, and game playing programs are used daily by millions of people. As such programs become closer in their abilities and intelligence to human beings the need will arise to verify and recognize such artificially intelligent software just like it is often necessary to authenticate and confirm identity of people. We propose applying techniques developed for behavior based recognition of humans to the identification and verification of intelligent game bots. Our experimental results demonstrate feasibility of such methods for both game Bot verification and even for recognition purposes.

Keywords- Game Bots, Behavioral Biometrics, Bot Recognition.

1. Introduction

Artificially Intelligent (AI) programs are quickly becoming a part of our everyday life. Virtual assistants, shopping bots, game playing programs, and smart search engines to give just some examples are used daily by millions of people. As such programs become closer in their abilities and intelligence to human beings the need will arise to verify and recognize such artificially intelligent software just like it is often necessary to authenticate and confirm identity of people. With respect to human beings identification and verification is mostly done for security reasons, to prevent certain people from unauthorized access to some resources and to allow it to the authorized personal.

Similarly, with respect to artificially intelligent programs reasons exist for determining the "identity" of a program or to verify the program is what it claims to be. Such reasons include but are not limited to:

- Finding out which program has actually performed a given task in case a number of possible agents exist, either for assigning blame or reward.
- Determining who has the authorship rights to the results of computation and creative output produced by AI software.
- Securing interaction between different pieces of intelligent software or between a human being and an instance of intelligent software.
- Preventing malicious intelligent software from obtaining access to information or system resources and granting it to authorized intelligent agents.
- Indirectly proving possession of unlicensed software on a system or network based on the observed capabilities of the system.

In addition to verifying and identifying people it is also possible to classify human beings into a number of groups based, for example, on race and gender. Similarly, in addition to verifying and identifying an instance of intelligent software we can verify and recognize a particular version or type to which this software belongs, and from which it differs only in so far as it was customized and trained differently from the original release. Since all instances of particular software release before being customized are exactly the same it is much more valuable to be able to identify or verify a software version than it is to classify a human being according to gender since larger variability exists within genders than between.

We propose applying techniques developed for behavior based recognition of humans to the identification and verification of intelligent programs. Those techniques are commonly called behavioral biometrics. Section 2 of this paper provides an overview of existing behavioral biometric technologies. Section 3 reviews existing AI software to which we can apply such techniques and covers previous work in the field. Section 4 presents a strategy based behavioral biometric used to verify identities of poker players. Finally, sections 5 and 6 report results of our experiments with automatic verification and recognition of artificially intelligent poker players.

2. Behavioral Biometrics

Purely behavioral biometrics are those which measure human behavior directly not concentrating on measurements of body parts or supposedly innate, unique and stable muscle actions such as the way an individual walks, talks, types or even grips a tool [18]. In accomplishing their everyday tasks human beings employ different strategies, use different style and apply unique skills and knowledge. Pure behavioral biometrics researchers attempt to quantify such behavioral traits and use resulting feature profiles to successfully verify identity (see Table 1).

While many purely behavioral biometrics are still in their infancy some very promising research has already been done. The results obtained justify feasibility of using behavior for verification of individuals and further research in this direction is likely to improve accuracy of such systems. Table 2 summarizes obtained accuracy ranges for the set of purely behavioral biometrics for which such data is available [71].

Table 1 summarizes what precisely is being measured by different behavioral biometrics as well as lists some of the most frequently used features for each type of behavior.

Behavioral Biometric	Measures	Extracted Features
Car driving style [26, 24, 25] [35] [47] [54, 44]	Skill	pressure from accelerator pedal and brake pedal, vehicle speed, steering angle
Email Behavior [64, 65] [68]	Style	length of the emails, time of the day the mail is sent, how frequently inbox is emptied, the recipients' addresses
Calling Behavior [34] [31] [17] [27]	Preferences	date and time of the call, duration, called ID, called number, cost of call, number of calls to a local destination,
Programming Style [62] [30] [28]	Skill, Style, Preferences	chosen programming language, code formatting style, type of code editor, special macros, comment style, variable names
Text Authorship [32] [63] [39, 41, 42]	Vocabulary	sentence count, word count, punctuation mark count, noun phrase count, word included in noun
Game Strategy [69, 75] [56]	Strategy/Skill	count of hands folded, checked, called, raised, check-raised, re-raised, and times player went all-in
Credit Card Use [15]	Preferences	account number, transaction type, credit card type, merchant ID, merchant address
Biometric Sketch [16, 4] [67] [37] [58]	Knowledge	location and relative position of different primitives
Command Line Lexicon [60, 51] [21] [76, 50, 45, 46]	Technical Vocabulary	used commands together with corresponding frequency counts, and lists of arguments to the commands
Painting Style [49]	Style	subtle pen and brush strokes characteristic
Soft Behavioral Biometrics [36]	Intelligence, Vocabulary, Skills	word knowledge, generalization ability, mathematical skill

Table 2: Comparison of behavioral biometrics based on their verification accuracy

Behavioral Biometric	Reported Verification Accuracy
Car driving style	68.8-73.0%
Email Behavior	86.2-90.5%
Calling Behavior	87.6-92.5%
Text Authorship	95.7-98.6%
Game Strategy	53.0-78.33%
Credit Card Use	80.0-99.95%
Biometric Sketch	98.7-100.0%
Command Line Lexicon	66.0-99.0%

Certain AI programs perform actions similar to those performed by human beings so it is logical to try to apply similar techniques to the task of profiling and recognizing such programs. This allows us to utilize an already existing technology in a novel and interesting way and possibly better understand mechanisms underlying the human behavior. As a consequence we might obtain improvements in the recognition of human beings, resulting from breakthroughs obtained in the recognition of intelligent machines.

3. Research Domain Overview

3.1. Previous Work

Existing research can be classified into three groups: recognizing patterns in the output of software and behavior of robots, distinguishing between people and computers, and identifying people (biometrics). To the best of our knowledge no research in output-based software recognition or verification exists; some work has been done in program recognition [55] and program understanding [57], in which the source code of the program is analyzed with the purpose of understanding the original purpose behind the creation of such software. Others have researched possibility of robot's behavior prediction and recognition never applying discovered trends to the recognition of robots exhibiting the observed behavior [8, 33].

In 1950 Alan Turing published his most famous work "Computing Machinery and Intelligence" in which he proposes evaluation of abilities of an artificially intelligent machine based on how closely it can mimic human behavior [66]. The test, which is now commonly known as the Turing test is structured as a conversation and can be used to evaluate multiple behavioral parameters, such as agent's knowledge, skills, preferences, and strategies. In essence it is the ultimate multimodal behavioral biometric, which was postulated to make it possible to detect differences between man and machine.

In 2000 Luis von Ahn et al. [2, 3] proposed the concept of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). It is a type of challenge-response test used in computing to determine whether or not the user is human. It relies on ability of human beings to perform pattern recognition at a level which is beyond that currently achievable by artificially intelligent programs. Both the Turing test and CAPTCHAs take advantage of the differences in abilities between human beings and intelligent machines to identify to which group an agent being tested belongs. Majority of previous research falls under the heading of biometric research and it is the utilization of biometric methodologies for recognition of AI software which forms the cornerstone of our research.

3.2. Existing AI software and Robots

The artificially intelligent programs we currently have are still years away from being as intelligent as human beings, but there are some programs to which we can attempt to apply our verification/identification techniques. Some examples are provided below, but the list will unquestionably grow as our success with AI technologies progresses and we obtain programs which are as creative and unique as human beings. We already have programs capable of composing inspiring music [20], drawing beautiful paintings [19], and writing poetry [14] and no limits are know to the abilities which a machine can eventually obtain. Table 3 lists some well developed artificially intelligent programs and behavioral biometrics we can apply to their verification [70].

Table 3: AI software and behavioral biometrics we can apply

AI software	**Behavioral biometric we can apply**
Game Playing Software (Chess, Poker, Go)	Profile of the game strategy, frequently used moves, openings, aggression level, etc.
Chat Bots	Linguistic profile based on frequently used words, common phrases, topics of conversation.
Text-to-Speech Software	Voice recognition, based on acoustic features such as voice pitch and speaking style.
Translation Software	Linguistic profile based on frequently used phrases, idioms, etc.
Speech Recognition	Text authorship combined with error rate analysis.

Biometrics refer to biological measurements collected for the purpose of identifying human beings, however this name is not appropriate with respect to non-biological agents. We propose that the research aimed at recognition and verification of software programs, robots and other non-biological agents be known as *Artimetrics* after the word "artilect", which is a shortened version of "artificial intellect" [29].

4. Verification and Recognition of Players

Yampolskiy et al. [69, 75, 72] proposed a system for verification of online poker players based on a behavioral profile which represents a statistical model of player's strategy. We propose that this approach can be expanded to the task of verification of the AI poker players. In the strategy-based-behavioral biometric the profile consists of frequency measurements indicating range of cards considered by the player at all stages of the game. It also measures how aggressive the player is via such variables as percentages of re-raised hands. The profile is actually human readable meaning that a poker expert can analyze and understand strategy employed by the player from observing his or her behavioral profile [56]. For example just by knowing the percentage of hands a particular player chooses to play pre-flop it is possible to determine which cards are being played with high degree of accuracy. Table 4 demonstrates a sample profile for a player named Bot.

Table 4: Strategy based behavioral biometric profile [75]

Player Name: Bot			Hands Dealt: 224	
	Pre-Flop	Flop	Turn	River
# of Hands Played	224	68	46	33
Folded	67%	28%	24%	18%
Checked	7%	54%	52%	52%
Called	21%	32%	28%	33%
Raised	4%	1%	4%	6%
Check-Raised	0%	4%	0%	0%
Re-Raised	0%	1%	0%	0%
All-In	1%	3%	4%	39%

A combination of statistical variables taken together produces a feature vector which is used by a pattern recognition algorithm to determine if a current profile is consistent with the actions previously seen from this particular player. In the Table 4 we see a 24 dimensional feature vector (number of hands played is only used to determine if we have enough information to put confidence in our statistical profile and is not analyzed as a part of a profile). Explanation for the meaning of the first 7 variables in our feature vector follows. The rest of the variables represent similar strategic ideas but at a later stages of the game [75].

- **pre-flop fold** Percentage of times this particular player has decided to give up his claims to the pot
- **pre-flop check** Percentage of times this particular player has decided to check
- **pre-flop any call** Percentage of times this particular player has paid an amount equivalent to the raise by some other player ahead in position in order to see the flop
- **pre-flop raise** Percentage of times this particular player has chosen to raise before seeing the flop
- **pre-flop check-raised** percentage of times a player has checked pre-flop allowing another player to put some money into the pot, just to come over the top and raise the pot after the action gets back to him
- **pre-flop re-raise** Percentage of times this particular player has chosen to re-raise somebody-else's raise before seeing the flop.
- **pre-flop all-in** Percentage of times this particular player has chosen to invest all his money in the hand

Once feature vectors describing players' strategies are obtained it becomes necessary to compare such multidimensional feature vectors. A comparison score needs to be generated using a similarity distance function. The distance score has to be very small for two feature vectors belonging to the same poker playing agent and therefore representing a similar strategy. At the same time it needs to be as large as possible for feature vectors coming from different players, as it should represent two distinct playing strategies.

The most popular similarity measure is Euclidian Distance, which is just the sum of the squared distances of two vector values (x_i, y_i) [75, 73, 74].

$$d_E = \sqrt{\sum_{i=1}^{n} (x_i, y_i)^2}$$

5. Intelligent Bots-Poker Players

Historically games were a sandbox used for testing novel AI theories and tools. It is a restricted domain, which allows techniques, which are not yet ready for the real world to be examined under controlled conditions. From the AI perspective game of Poker provides opportunities for working with Neural Networks, Genetic Algorithms, Fuzzy Logic, and Distributed Agents to solve problems with probabilistic knowledge, risk assessment, deception, and other real world situations. Ideas from game theory, pattern recognition, simulation and modeling are also come into play, not even mentioning mathematics, statistics, probability and other areas of mathematics [10, 38]. As a result research of artificial poker players enjoys a long and fruitful history. We will begin with a short overview of existing work in the field of developing artificially intelligent poker players, followed by our approach to the creation of AI poker players used as subjects in our experiments on AI player verification and recognition.

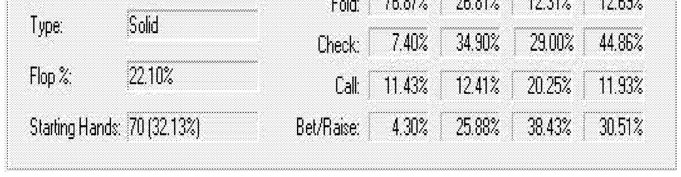
- **Full Scale Texas Hold'em Poker** Billings et al. [9, 10, 11, 13, 23, 22, 12] have investigated development of a complete poker playing program for the game of Texas Hold'em. They use opponent modeling, statistical analysis, semi-optimal pre-flop strategy and even neural-network-based opponent's action prediction to construct a world-class poker-playing program. Their research is still in progress, but their best program Poki has already proven itself as a reasonable strength opponent against both computers and people. It has been playing online and consistently winning, but since the games were not real money games the quality of opponents remains questionable.
- **Bayesian Poker** Korb et al. are developing a Bayesian Poker (BP) program which uses a Bayesian network to model the program's poker hand, the opponent's hand and the opponent's playing behavior based on the hand, and betting curves which govern play given a probability of winning. The history of play with opponents is used to improve program's understanding of their behavior [43]. BP is written to play a two-player five-card stud poker game and is still a work in progress.
- **Evolved Poker Players** Genetic algorithms provide an automated way to solve complex problems without explicitly solving every particular sub-instance of the problem. Many researchers have attempted to evolve good players for the game of poker, typically for one of the simplified versions, making search space more reasonable.
- **Barone and While** Barone et al. [7, 5, 6] use a simple poker variant where each player has two private cards, access to five community cards and there is only one round of betting. Their solution takes into account hand strength, betting position and risk management. The approach shows how a player that has evolved using evolutionary strategies can adapt its style to two types of game: loose or tight.
- **Noble and Watson** Noble et al. [52, 53] use Pareto co-evolution on the full scale game of Texas Hold'em and show that as compared to the traditional genetic algorithm their approach shows promise. Pareto co-evolution treats players as dimensions and attempts to find optimal playing strategies for a multidimensional space of potential strategies.
- **Kendall and Willdig** Kendall et al. [40] have also attempted to evolve a good poker player and showed that a simple reward system of adjusting weights is sufficient to produce a player capable of beating its opponents after playing them for some time and adapting to their style of play.

5.1. Our Methodology

Our implementation of poker bots was done using the statistical package called Online Hold'em Inspector version 2.26d4 [1]. By specifying such conditions as tendency of bots to bluff, slow play, check raise and their aggressiveness level as well as their pre-flop card selection we were able to obtain numerous valid artificial poker players.

Overall we exercised control over: Pre-flop hand selection, on the flop action based on position, number of players, and number of raises in the pot; check raising on the flop, turn and river, bluffing frequency based on a type of opponent, blind protection and blind stealing, slow-playing versus different opponents based on the strength of our hand, opponent respect levels, and many others.

By manipulating those variables associated with bots playing strategy and combining them in numerous ways we were able to generate a multitude of realistically behaving poker players. By statistically analyzing bot's strategy we were able to predict some characteristics of the bot's behavioral profile. The figure below demonstrates such statistical estimation of the behavioral profile for a bot called Solid, representing a somewhat tight but relatively aggressive strategy.


Stats (for a typical full game)

Name:	Solid
Type:	Solid
Flop %:	22.10%
Starting Hands:	70 (32.13%)

	Preflop	Flop	Turn	River
Fold:	76.87%	26.81%	12.31%	12.69%
Check:	7.40%	34.90%	29.00%	44.86%
Call:	11.43%	12.41%	20.25%	11.93%
Bet/Raise:	4.30%	25.88%	38.43%	30.51%

Figure 1: Behavioral profile statistically predicted based on chosen strategy

For our experimental set of artificially intelligent poker players we have taken a number of built-in profiles which came standard with the Hold'em Inspector namely: Solid, Rock, Maniac, Fish, and Typical. We have also included some profiles available via Internet poker forums particularly: Vixen70, GoldenEagle, BettyBot v1.1, and poker_champ [48]. Finally we have programmed in an additional artificial poker player, called AIbot, based on the game theory research presented by Sklansky et al. [61]. Validity of our poker bot was tested at low-stakes real-money online poker tables against human opponents where our bot consistently scored around 3 big bets per hour in profits. In total we ended up with 10 artificial poker players which is representative of the current number of competing programs in many AI sub-fields such as character recognition, chess, translation software, etc.

6. Results and Conclusions

Each artificially intelligent poker player had played two long poker games, each one of at least 150 hands, against a mixture of human and artificial opponents. First game was played to establish the biometric template for the player and second one to perform the verification and identification experiments.

6.1. Verification Experiment

In a databank of 10 AI player's signatures each one was compared with one profile taken from the same player as the one who generated the original signature and with another profile taken from a randomly chosen player, for a total of 20 comparisons. This gave us an experimental set up in which authentic users and imposters are equal in number. Using Euclidian distance similarity measure and an experimentally established threshold of 75 the algorithm has positively verified 90.00% of users. The only false verification was the positive verification of the Solid bot as the Rock bot, which could be explained by the fact that both bots are programmed to play a conservative type of poker strategy.

7.1 Identification Experiment

For this experiment we used a databank of 10 AI player signatures. Each player's record contains an original signature from the enrollment period and a second signature from the testing period. Each testing signature was compared against all original signatures in the databank, for a total of 100 comparisons. The highest matching profile with respect to the similarity measure was recorded as either belonging to the same player (a successful identification) or to a different player (a false identification). From the total of 10 highest matching profiles 3 were correctly identified and 7 were false matches. This gives us artificial player identification with overall 30.00% accuracy, which is a good result considering we are using a soft behavioral biometric technique, which are typically only used for verification not for identification purposes.

Our results demonstrate possibility of using strategy-based behavioral biometrics for accurate verification of Intelligent Agents. We further believe that it is feasible to apply other behavioral biometric techniques to additional domains in which artificially intelligent programs are becoming a major force. In the field of output authorship recognition of artificially intelligent software we expect a lot of progress as AI programs become capable of many tasks usually only attributed to humans such as telling a joke [59], composing music [20], painting [19], or creating poetry [14].

As AI technologies become more commonplace in our society it will be necessary to determine which program has actually performed a given task, assign the authorship rights to software, secure interaction between different pieces of intelligent software, and prevent malicious software from accessing certain information. We have demonstrated that it is possible to verify and even identify artificially intelligent game bots based on their observable behavior. We further propose that the research aimed at recognition and verification of software programs, industrial and personal robots and other non-biological agents be known as *Artimetrics* to distinguish it from the traditional biology centered research in biometrics.

6. References

[1] -. -, *Online Holdem Inspector*, Available at: http://www.pokerinspector.com/, Retrieved May 2, 2006.

[2] L. v. Ahn, M. Blum, N. Hopper and J. Langford, *CAPTCHA: Using Hard AI Problems for Security*, *In Eurocrypt*, 2003.

[3] L. v. Ahn, M. Blum and J. Langford, *How Lazy Cryptographers do AI*, *In Communications of the ACM*, Feb. 2004.

[4] S. Al-Zubi, A. Brömme and K. Tönnies, *Using an Active Shape Structural Model for Biometric Sketch Recognition*, *In Proceedings of DAGM*, Magdeburg, Germany, 10.-12. September 2003, pp. 187-195.

[5] L. Barone and L. While, *An adaptive learning model for simplified poker using evolutionary algorithms*, *In proceedings of the Congress of Evolutionary Computation (GECCO-1999)*, 1999, pp. 153-160.

[6] L. Barone and L. While, *Evolving adaptive play for simplified poker*, *In In proceedings of IEE International Conference on Computational Intelligence (ICEC-98)*, 1999, pp. 108-113.

[7] L. Barone and L. While, *Evolving computer opponents to play a game of simplified poker*, *In In proceedings of the 1998 International Conference on Evolutionary Computation (ICEC'98)*, 1998, pp. 108-113.

[8] D. Barrios-Aranibar and P. J. Alsina, *Recognizing Behaviors Patterns in a Micro Robot Soccer Game*, *In Proceedings of the Fifth international Conference on Hybrid intelligent Systems*, IEEE Computer Society, Washington, DC, December 06 - 09, 2005.

[9] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg and D. Szafron, *Approximating game-theoretic optimal strategies for full-scale poker*, *ijcai-03*, 2003.

[10] D. Billings, D. Papp, J. Schaeffer and D. Szafron, *Opponent modeling in Poker*, *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, AAAI Press, Madison, WI, 1998, pp. 493-498.

[11] D. Billings, D. Papp, J. Schaeffer and D. Szafron, *Poker as testbed for ai research*, *AI '98: Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, Springer-Verlag, London, UK, 1998, pp. 228-238.

[12] D. Billings, L. Pena, J. Schaeffer and D. Szafron, *Learning to play strong poker*, *Machines that learn to play games*, Nova Science Publishers, Inc, Commack, NY, USA, 2001, pp. 225--242.

[13] D. Billings, L. Pena, J. Schaeffer and D. Szafron, *Using probabilistic knowledge and simulation to play poker*, *In AAAI/IAAI*, 1999, pp. 697-703.

[14] J. Boyd-Graber, *Semantic Poetry Creation Using Lexicographic and Natural Language Texts*, Available at: http://www.cs.princeton.edu/~jbg/documents/poetry.pdf, Retrieved July 2, 2006.

[15] R. Brause, T. Langsdorf and M. Hepp, *Neural Data Mining for Credit Card Fraud Detection*, *In Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, 1999, pp. 103--106.

[16] A. Brömme and S. Al-Zubi, *Multifactor Biometric Sketch Authentication*, *In A. Brömme and C. Busch, editors, Proceedings of the BIOSIG 2003*, Darmstadt, Germany, 24. July 2003, pp. 81-90.

[17] M. Cahill, D. Lambert, J. Pinheiro and D. Sun, *Detecting fraud in the real world*, *Technical report, Bell Labs, Lucent Technologies*, 2000.

[18] Caslon-Analytics, *Available at: http://www.caslon.com.au/biometricsnote6.htm*, Retrieved October 2, 2005.

[19] H. Cohen, *HOW TO DRAW THREE PEOPLE IN A BOTANICAL GARDEN*, Available at: http://crca.ucsd.edu/~hcohen/cohenpdf/how2draw3people.pdf, 1988.

[20] D. Cope, *Virtual Music: Computer Synthesis of Musical Style*, The MIT Press, Cambridge, Massachusetts, 2001.

[21] V. Dao and V. Vemuri, *Profiling Users in the UNIX OS Environment*, *International ICSC Conference on Intelligent Systems and Applications*, University of Wollongong Australia, Dec. 11-15, 2000.

[22] A. Davidson, *Using artifical neural networks to model opponents in texas hold'em*, Available at: http://citeseer.ist.psu.edu/460830.html, Retrieved May 25, 2005.

[23] A. Davidson, D. Billings, J. Schaeffer and D. Szafron, *Improved opponent modeling in poker*, *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI'2000)*, Las Vegas, Nevada, 2000, pp. 1467-1473.

[24] H. Erdogan, A. Ercil, H. Ekenel, S. Bilgin, I. Eden, M. Kirisci and H. Abut, *Multi-modal person recognition for vehicular applications*, *N.C. Oza et al. (Eds.) MCS 2005, LNCS 3541*, Monterey CA, Jun. 2005, pp. 366 - 375.

[25] H. Erdogan, A. N. Ozyagci, T. Eskil, M. Rodoper, A. Ercil and H. Abut, *Experiments on decision fusion for driver recognition*, *iennial on DSP for in-vehicle and mobile systems*, Sesimbra Portugal, Sep. 2005.

[26] E. Erzin, Y. Yemez, A. M. Tekalp, A. Erçil, H. Erdogan and H. Abut, *Multimodal Person Recognition for Human-Vehicle Interaction*, *IEEE MultiMedia*, April 2006, pp. 18-31.

[27] T. Fawcett and F. Provost, *Adaptive Fraud Detection*, *Data Mining and Knowledge Discovery*, Kluwer Academic Publishers, 1997, pp. 291-316.

[28] G. Frantzeskou, S. Gritzalis and S. MacDonell, *Source Code Authorship Analysis for Supporting the Cybercrime Investigation Process*, *1st International Conference on eBusiness and Telecommunication Networks - Security and Reliability in Information Systems and Networks Track*, Kluwer Academic Publishers, Setubal Portugal, August 2004, pp. 85-92.

[29] H. d. Garis, *The Artilect War*, ETC publications, 2005.

[30] A. Gray, P. Sallis and S. MacDonell, *Software Forensics: Extending Authorship Analysis Techniques to Computer Programs*, *In Proc. 3rd Biannual Conf. Int. Assoc. of Forensic Linguists (IAFL'97)*, 1997.

[31] H. Grosser, H. Britos and R. García-Martínez, *Detecting Fraud in Mobile Telephony Using Neural Networks*, *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2005, pp. 613-615.

[32] H. v. Halteren, *Linguistic profiling for author recognition and verification*, *In Proceedings of ACL-2004*, 2004.

[33] K. Han and M. Veloso, *Automated robot behavior recognition*, *In Proceedings of IJCAI-99 Workshop on Team Behaviors and Plan Recognition*, 1999.

[34] C. Hilas and J. Sahalos, *User Profiling for Fraud Detection in Telecommunication Networks*, *5th International Conference on Technology and Automation (ICTA 2005)*, Thessaloniki, Greece, 15-16 October 2005, pp. 382-387.

[35] K. Igarashi, C. Miyajima, K. Itou, K. Takeda, F. Itakura and H. Abut, *Biometric identification using driving behavioral signals*, *Proc. 2004 IEEE International Conference on Multimedia and Expo*, 2004, pp. 65-68.

[36] B. A. JACOB and S. D. LEVITT, *To catch a cheat*, *Education next*, Availablet at: www.educationnext.org, 2004.

[37] I. Jermyn, A. Mayer, F. Monrose, M. K. Reiter and A. D. Rubin, *The Design and Analysis of Graphical Passwords*, *Proceedings of the 8th USENIX Security Symposium*, Washington, D.C., August 23-36, 1999.

[38] p. Jonathan Schaeffer. In, *The games computers (and people) play*, *AAAI/IAAI*, 2000, pp. 1179-.

[39] P. Juola and J. Sofko, *Proving and Improving Authorship Attribution*, *Proceedings of CaSTA-04 The Face of Text*, 2004.

[40] G. Kendall and M. Willdig, *An investigation of an adaptive poker player*, *In proceedings of 14th Australian Joint Conference on Artificial Intelligence*, Adelaide, Australia, Dec 10-14, 2001, pp. 189-200.

[41] M. Koppel and J. Schler, *Authorship Verification as a One-Class Classification Problem*, *in Proceedings of 21st International Conference on Machine Learning*, Banff, Canada, July 2004, pp. 489-495.

[42] M. Koppel, J. Schler and D. Mughaz, *Text Categorization for Authorship Verification*, *Eighth International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, Januray 2004.

[43] K. Korb, A. Nicholson and N. Jitnah, *Bayesian Poker*, *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Morgan Kaufmann, San Francisco, CA, 1999, pp. 343-35.

[44] N. Kuge, T. Yamamura and O. Shimoyama, *A driver behavior recognition method based on driver model framework.*, *Society of Automotive Engineers Publication*, 1998.

[45] T. Lane and C. E. Brodley, *An Application of Machine Learning to Anomaly Detection*, *20th Annual National Information Systems Security Conference*, 1997, pp. 366-380.

[46] T. Lane and C. E. Brodley, *Detecting the Abnormal: Machine Learning in Computer Security*, *Department of*

Electrical and Computer Engineering, Purdue University Technical Report ECE-97-1, West Lafayette, January 1997.

[47] A. Liu and D. Salvucci, *Modeling and Prediction of Human Driver Behavior*, *Proc. of the 9th HCI International Conference*, New Orleans, LA, Aug. 5-10, 2001, pp. 1479-1483.

[48] C. LOONIES, *Texas Holdem Poker*, Available at: http://cyberloonies.com/poker.html, Retrieved May 2006.

[49] S. Lyu, D. Rockmore and H. Farid, *A Digital Technique for Art Authentication*, *Proceedings of the National Academy of Sciences*, 2004, pp. 17006-17010.

[50] J. Marin, D. Ragsdale and J. Surdu, *A hybrid approach to the profile creation and intrusion detection*, *DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, 2001.

[51] R. A. Maxion and T. N. Townsend, *Masquerade detection using truncated command lines*, *In International conference on dependable systems and networks(DNS-02)*, IEEE Computer Society Press, 2002.

[52] J. Noble, *Finding robust texas hold'em poker strategies using pareto coevolution and deterministic crowding*, *In In Proceedings of the 2002 International Conference on Machine Learning and Applications (ICMLA'02)*, 2002.

[53] R. A. Noble and J. Watson, *Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection*, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, 2001, pp. 493-500.

[54] N. Oliver and A. P. Pentland, *Graphical models for driver behavior recognition in a SmartCar*, *In Proceedings of the IEEE Intelligent Vehicles Symposium*, 2000.

[55] D. Ourston, *Program Recognition*, *IEEE Expert*, Winter 1989, pp. 36-49.

[56] Poker-edge.com, *Stats and Analysis*, Available at: http://www.poker-edge.com/stats.php, Retrieved June 7, 2006.

[57] A. Quilici, Q. Yang and S. Woods, *Applying Plan Recognition Algorithms To Program Understanding*, *Automated Software Engineering: An International Journal*, Kluwer Academic Publishers, July 1998, pp. 347--372.

[58] K. Renaud, *Quantifying the Quality of Web Authentication Mechanisms. A Usability Perspective*, *Journal of Web Engineering, Vol. 0, No. 0*, Rinton Press, Available at: http://www.dcs.gla.ac.uk/~karen/Papers/j.pdf, 2003.

[59] G. Ritchie, *Current Directions in Computational Humor*, *Artificial Intelligence Review*, 2001, pp. 119-135.

[60] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus and Y. Vardi, *Computer Intrusion: Detecting Maquerades*, Statistical Science, 16 (2001), pp. 1-17.

[61] D. Sklansky and M. Malmuth, *Hold'em Poker for Advanced Players*, Two Plus Two Publishing, March 2004.

[62] E. H. Spafford and S. A. Weeber., *Software Forensics: Can We Track Code to its Authors? 15th National Computer Security Conference*, Oct 1992, pp. 641-650.

[63] E. Stamatatos, N. Fakotakis and G. Kokkinakis, *Automatic authorship attribution*, *in Proc. nineth Conf. European Chap. Assoc. Computational Linguistics*, Bergen, Norway, Jun. 1999, pp. 158--164.

[64] S. J. Stolfo, S. Hershkop, K. Wang, O. Nimeskern and C.-W. Hu, *A Behavior-based Approach to Securing Email Systems*, *Mathematical Methods, Models and Architectures for Computer Networks Security*, Springer Verlag, Sept. 2003.

[65] S. J. Stolfo, C.-W. Hu, W.-J. Li, S. Hershkop, K. Wang and O. Nimeskern, *Combining Behavior Models to Secure Email Systems*, *CU Tech Report*, Available at: www1.cs.columbia.edu/ids/publications/EMT-weijen.pdf, April 2003.

[66] A. Turing, *Computing Machinery and Intelligence*, *Mind*, 1950, pp. 433-460.

[67] C. Varenhorst, *Passdoodles; a Lightweight Authentication Method*, Available at: http://people.csail.mit.edu/emax/papers/varenhorst.pdf, July 27, 2004.

[68] O. D. Vel, A. Anderson, M. Corney and G. Mohay, *Mining Email Content for Author Identification Forensics*, *SIGMOD: Special Section on Data Mining for Intrusion Detection and Threat Analysis*, 2001.

[69] R. V. Yampolskiy, *Behavior Based Identification of Network Intruders*, *19th Annual CSE Graduate Conference (Grad-Conf2006)*, Buffalo, NY, February 24, 2006.

[70] R. V. Yampolskiy, *Behavioral Biometrics for Verification and Recognition of AI Programs*, *20th Annual Computer Science and Engineering Graduate Conference (GradConf2007)*, Buffalo, NY, April 13, 2007.

[71] R. V. Yampolskiy, *Human Computer Interaction Based Intrusion Detection*, *4th International Conference on Information Technology: New Generations (ITNG 2007)*, Las Vegas, Nevada, USA, April 2-4, 2007.

[72] R. V. Yampolskiy, *Improving Accuracy of a Behavior-Based Network Intrusion Detection System*, *IEEE Upstate NY Workshop on Communications and Networks '06*, Rochester, NY, November 6, 2006.

[73] R. V. Yampolskiy and V. Govindaraju, *Dissimilarity Functions for Behavior-Based Biometrics*, *Biometric Technology for Human Identification IV. SPIE Defense and Security Symposium*, Orlando, Florida, April 9-13, 2007.

[74] R. V. Yampolskiy and V. Govindaraju, *Similarity Measure Functions for Strategy-Based Biometrics.*, *International Conference on Signal Processing (ICSP 2006)*, Vienna, Austria, December 16-18, 2006.

[75] R. V. Yampolskiy and V. Govindaraju, *Use of Behavioral Biometrics in Intrusion Detection and Online Gaming*, *Biometric Technology for Human Identification III. SPIE Defense and Security Symposium*, Orlando, Florida, 17-22 April 2006.

[76] D. Y. Yeung and Y. Ding, *Host-based intrusion detection using dynamic and static behavioral models*, *Pattern Recognition*, pp. 229-243.

THE ANATOMY OF AN INTER-VEHICULAR GAMING COMMUNICATION SUBSYSTEM WITH EXPERIMENTS

Emiliano Manca, Fabio Parmeggiani, Claudio E. Palazzi, Stefano Ferretti, Marco Roccetti
Department of Computer Science
University of Bologna
Mura Anteo Zamboni, 7 40127 Bologna, Italy
{mancae, fparmegg, cpalazzi, sferrett, roccetti}@cs.unibo.it

KEYWORDS
Online Games, Inter-Vehicular Communication, Networked Interactive Entertainment, Multi-hop Broadcasting, VANET.

ABSTRACT

We report on an extensive experimental evaluation of a new scheme we recently devised for a fast broadcast of messages in Inter-Vehicular Communication (IVC) systems. Our scheme is specifically designed for online gaming contexts. It is composed of two distributed algorithms, working in parallel. The first one is a dynamic transmission range estimator, which provides each vehicle with a measure of the distance that a broadcast message can cover in the specific geographical area where the car is traveling. The second component is a broadcasting scheme that, based on the transmission range estimation, identifies vehicles that should be selected to forward the message, so as to cover the whole vehicular network in a minimal time. Our transmission range estimator works by exploiting information extracted from game events distributed during the game evolution. Thus, only game related messages are utilized in the IVC. Simulation results confirm the efficacy of our approach.

INTRODUCTION

Online gaming is becoming a pervasive activity due to the widely available wireless connectivity (Farber 2004, Pantel and Wolf 2002). Several mobile games have been recently developed which allow users to play while being on the move (Benford et al. 2005, Bjork et al. 2001, Piekarski and Thomas 2002). The next step will surely be that of enabling car passengers to play online games through Inter-Vehicular Communication (IVC) systems.

Indeed, "on-car entertainment" is a blooming market, with an ever increasing number of new technologies, typically available for "in-home entertainment", ready to be mounted in people's cars. For example, DVD systems and game consoles are now commonly installed on vehicles. Therefore, the current research trend is that of investigating on means to make cars able to communicate among each other, by resorting to some ad-hoc networking technology.

New protocols for IVC are now under development. The most prominent one is the DSRC/IEEE 802.11p standard (DSRC 2007). This new technology will allow to distribute application messages among a group of vehicles spread over an area of few kilometers (i.e., a *car platoon*) through ad-hoc communication. In this highly dynamic context, it becomes particularly interesting to study how fast paced and interactive applications such as online games can be supported by the IVC infrastructure.

In particular, to provide cars passengers with interactive online gaming experiences, a main open issue is that of identifying a fast and smart broadcast scheme, to be applied in IVC systems, able to guarantee a rapid delivery of game events produced by players. This is a crucial aspect which demands solution in order to quickly deliver game events among all players (Palazzi et al. 2006).

As a matter of fact, the fastest and less resource-consuming way to broadcast game events in a vehicular network is that of forwarding messages through a multi-hop broadcast over the whole car platoon (Palazzi et al. 2007, Palazzi et al. 2007b). This way, each vehicle (equipped with a wireless communication infrastructure) can hear the message and, if there is any player on that vehicle, pass the game event to the application layer.

The rationale behind the need for a fast and efficient multi-hop broadcasting scheme lies on the fact that even if game events are typically limited in size to few tens of bytes, each player generates several of them every second (Farber 2004, Pantel and Wolf 2002). Thus, a shared wireless network could become quickly congested with a high number of participants. Moreover, for the sake of game interactivity, this message delivery must be accomplished as soon as possible, possibly within a temporal range of about 150-300 ms, depending on the game type (Palazzi et al. 2006).

A first simple solution to implement such a scheme would be that of let any vehicle to re-broadcast every message as soon as it is received, so as to cover the whole vehicular network. Yet, this is not a good solution, since it would generate an excessive number of forwarding messages being transmitted, leading to collisions, network congestion, augmented delays, and even to a transmission paralysis.

Summing up, an efficient approach is needed to quickly propagate messages by guaranteeing a limited utilization of network resources. Basically, the idea is that of ensuring as few redundant transmissions as possible, so as to keep the channel available for other transmissions (Bisvas 2006, Fasolo et al. 2005, Korkmax et al. 2004, Palazzi et al. 2007).

With this problem in view, we have recently devised a new fast broadcast scheme which aims at ensuring that, upon a message broadcast, the farthest vehicle in the sender's transmission range becomes the next forwarder (Palazzi et al. 2007b). Our solution exploits two different components: i) a transmission range estimator, which estimates the maximum distance reached by a broadcast transmission in a particular portion of the vehicular network, and ii) a broadcasting scheme, which let vehicles select those cars that should forward a given message, based on the estimated transmission range.

It is important to point out that the transmission range estimation is performed by exploiting information extracted from game events distributed during the online game-play evolution. Thus, no additional messages are sent through the vehicular network that could increase the network traffic.

In this paper, we report on an extensive experimental evaluation we conducted to assess the efficacy of our proposed solution. Simulations have been carried out by resorting to the well known NS-2 simulator (NS-2 2007). Specifically, we measured the influence of the network diameter and of the number of players on our approach. Moreover, we have contrasted it against other mechanisms proposed in scientific literature. The simulation study confirms the efficacy of our devised solution.

The rest of this paper is organized as follows. Section 2 describes our scheme. Section 3 reports on results obtained from the simulative assessment we conducted to verify its efficacy. Finally, Section 4 concludes the paper.

FAST BROADCAST FOR INTER-VEHICULAR GAME COMMUNICATION

We designed *Fast Broadcast Algorithm* (FBA) to quickly deliver multi-hop broadcast messages to players belonging to the same car platoon and engaged in a certain game (Palazzi et al. 2007b). Two main components characterize our scheme: i) a transmission range estimator, and ii) a broadcasting scheme, which are outlined in the following. These two components work at a IVC session layer, interleaved between the application (game) layer and the transmission layer that factually performs the message broadcast. It is important to notice that our approach can be interfaced with classic transmission layers such as transport, network or MAC layers, depending on the specific architectural and communication requirements of the game and of the IVC system (Palazzi et al. 2007, Palazzi et al. 2007b).

We assume that a positioning system, such as a GPS, is available at each car which participates to the game and to the broadcasting activity. This technology serves to measure distances among vehicles and transmission ranges of broadcast game events.

Transmission Range Estimator

During the game evolution, each game event generated by a player is encapsulated in a message. Together with the game event, the message includes parameters (managed at the IVC layer) useful to obtain an estimation of the transmission range in the particular area of transmission. These parameters are: i) position of the sender vehicle; ii) its driving direction; iii) the maximum distance from which another vehicle was "recently heard", from backward, denoted as *backward maximum distance* (BMD); iv) the maximum distance from which another vehicle was "recently heard", from frontward, denoted as *frontward maximum distance* (FMD); v) the backward transmission range estimation, denoted as *backward maximum range* (BMR) estimation; vi) its *frontward maximum range* (FMR) estimation. BMR and FMR represent how far a transmission is expected to go before the signal becomes to weak to be intelligible and are the main parameters utilized by our FBA.

To compute a correct transmission range estimation, each vehicle needs to hear several messages from other cars around. Indeed, this naturally happens when considering interactive online games, where each node generates several messages per each second (Palazzi et al. 2006). This guarantees that fresh information is always available at each vehicle to compute estimations of transmission ranges.

Specifically, to estimate BMR and FMR, vehicles exploit an heuristics that uses the estimations obtained for BMD and FMD. Upon reception of a message, BMD and FMD are updated by resorting to the following equation:

$$xMD = \max(xMD_{current}, d) \qquad (1)$$

where xMD represents FMD or BMD, depending on whether the message arrives from frontward or backward, $xMD_{current}$ is the current value to be updated, and d is the distance of the vehicle that broadcast the message (measured by exploiting a location system such as a GPS). In simple words, each time a message is received from a vehicle farther than others previously heard, this new information is stored through this parameter.

Needless to say, since cars are moving, transmission conditions dynamically change. Thus, old stored values are meaningless after the vehicle has covered a given distance. With this in view, the estimation of $xMD_{current}$ expires after a tuned timeout, to be promptly updated based on more recent messages.

Based on the obtained FMD and BMD, at each vehicle, FMR and BMR can be computed. In particular, BMR is heuristically calculated by considering messages coming from vehicles behind the considered one; this value is computed as the largest among all received FMDs and all distances from vehicles that generated them. Similar, specular considerations can be made for FMR. In other words, the two transmission range estimations are updated as follows:

$$xMR = \max(xMR_{current}, d, msg.\bar{x}MD) \qquad (2)$$

where, xMR represents FMR or BMR, $xMR_{current}$ is the current value to be updated, d is the distance of the vehicle that broadcast the message, and $msg.\bar{x}MD$ is the data contained in the received message, related to the maximum hearing distance, i.e., FMD is considered if the message is received from backward, BMD is considered in the opposite case.

As demonstrated in our simulations, these heuristics provide vehicles with accurate estimations of transmission ranges.

Broadcasting Scheme

Once a given player generates a new game event, the corresponding message is broadcast to be received by all other players in the vehicular network. Since with high probability a single broadcast procedure cannot cover the whole area of interest, a forwarding procedure is activated in both directions. Thus, each vehicle is engaged in a forwarding procedure in a given direction (frontward if the message comes from the back, backward conversely).

To avoid cyclic back and forth transmissions of the same game event, each message includes information on the position of the vehicle that generated the event and information on the position of the last forwarder.

We now go into some details on the employed broadcasting scheme. When receiving a message, based on the described parameters, each vehicle assigns itself a priority in becoming the next forwarder of the received message. The transmission range estimation (xMR) is used by vehicles to determine which one among them will become the next forwarder. With the aim of minimizing the number of hops in order to reduce the propagation delay, the next forwarder should be the farthest possible vehicle with respect to the sending one. Therefore, the longer the relative distance of the considered vehicle from the sender, the higher the priority of the considered vehicle in becoming the next forwarder.

In particular, vehicles' priorities to forward a message are determined by assigning different waiting times from the reception of the message to the time at which they will try to forward it. This waiting time is randomly computed based on a contention window value, as inspired by classical backoff mechanisms in IEEE 802.11 MAC protocol (IEEE WLAN MAC 1999).

If, while waiting, some farther vehicle along the same direction of multi-hop propagation already forwarded the message, vehicles between the sender and the forwarder abort their countdowns to transmission as the message has already been propagated "over their heads". Instead, vehicles which are located even farther re-activate the forwarding procedure for the next hop.

The contention window (*CW*) used by each vehicle is measured in time slots and varies between a minimum value (*CWMin*) and a maximum one (*CWMax*), depending on the distance from the sending/forwarding vehicle (*Dist*) and on the advertised estimated transmission range xMR, as shown in equation (3).

$$CW = \left\lfloor \left(\frac{xMR - Dist}{xMR} \times (CWMax - CWMin) \right) + CWMin \right\rfloor \quad (3)$$

Using (3), the farthest vehicle in the sender's transmission range is privileged in becoming the new forwarder. Indeed, the nearer the vehicle to the sending car, the larger the contention window; larger contention windows make more likely that a larger timeout value will be chosen and, hence, that somebody else will be faster in forwarding the game event.

A Practical Example

A simple example can permit a clearer understanding of our solution. To this aim, we use Fig. 1 to show a broadcast procedure during the distribution of a game event. Suppose an event has been generated by a given vehicle, not shown in the figure. The event has been propagated in both directions (backward and frontward) in the car platoon. We consider only a portion of the platoon backward with respect to the vehicle that generated the game event. This means that the message needs to be propagated backward, till reaching the end of the area of interest. We suppose that, based on the scheme we propose, vehicle *d* forwards the message backward. In this example, suppose that only vehicles within the shaded oval receive the message.

If we assume that the broadcast procedure is reliable, all cars in the shaded area receive the message. Cars *a*, *b* and *c* already received the game event in the previous turn, since the message comes from frontward and *d*, which has the message, is behind them. Then, upon the reception of the message (from car *d*), the game event contained within the message is simply ignored at cars *b* and *c*, and the event is not passed at the application (game) layer to be processed, since it has already been processed. Instead, the message is utilized to update the estimations of BMD and BMR, using equations (1) and (2).

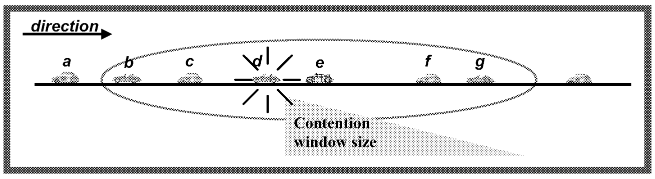


Figure 1: A practical example of our scheme.

As to vehicles *e*, *f* and *g*, values for FMD and FMR are updated, by exploiting equations (1) and (2). Moreover, these vehicles are engaged in a new broadcast procedure to forward the message backward. Thus, based on equation (3), a contention window is calculated, which will be proportional to the distance between *d* and the considered vehicle, as shown in the figure (observe the triangle: the height at each vehicle corresponds to its contention window). Each car then selects a random timeout within its contention window. This way, it is more probable that *g* becomes the next car that tries to forward the game event.

EXPERIMENTAL ASSESSMENT

We carried out an extensive simulation assessment to test our solution. The main tool utilized for our experiments is the well known NS-2 simulator (NS-2 2007). For each tested configuration, 40 simulations were run and their outcomes were averaged.

In these experiments, the length of the vehicular network varies from 1 to 8 Km and for each length we compared different densities of vehicles having communication capabilities. As a scenario, we considered a freeway with multiple lanes and the simultaneous presence of vehicles that had no communication capabilities. Cars are randomly positioned in a lane with a minimum distance of 20 m.

Focusing on the parameters, we have set *CWMin* and *CWMax* equal to 32 and 1024 slots, respectively, as inspired by the standard IEEE 802.11 protocol (IEEE WLAN MAC). Different slot sizes have been compared: 9 µs, which corresponds to the value utilized by IEEE 802.11g (IEEE 802.11g), and a larger of 200 µs, which allow a larger time distribution of contention delays. We set the actual transmission range to vary from 300 to 1000 m, in order to test boundary values that have been declared by the IEEE 802.11p developing committee (Guo et al. 2005). Due to the similarity of obtained results, we report only on the case in which the actual transmission rate is one game event sent every 300 ms.

We consider a number of players, among the vehicles with communication capabilities, which vary from 2 to 50. (When not differently stated, in the simulation this value is set to 50.) This means that game events are periodically generated from each of these vehicles and broadcast, even through multi-hop, to all other players in the network. As to the generation rate at each player, we considered different values, which correspond to different kinds of games. In this work, we report on the classic setting of a game event every 300 ms. The size of each game event was 200 Bytes (Farber 2002, Palazzi et al. 2006).

Influence of Network Diameter

We evaluate the ability of our architecture in ensuring interactivity to the online game application with diverse lengths of the car platoon.

Obviously, the smaller the vehicular network is, the faster message delivery results. Indeed, outcome values are generally proportional to the length of the vehicular network and Fig. 2 shows how the number of hops that messages have to traverse to cover the whole gaming car platoon is about 3-4 hops for each km, regardless of the vehicular network's size and of the slot duration. This is coherent with the fact that the actual transmission range was around 300 m.

The number of slots a message has to wait before being forwarded on the next hop also depends on possible collisions that forces vehicles to retransmit the message, even multiple times. This happens more often when the time width of a slot is very small, i.e., 9 µs, as demonstrated by Fig. 3 and Fig. 4. Indeed, Fig. 3 shows that with 9 µs, the number of slots that a game event experiences in its path through the vehicular network is much higher than when 200 µs is employed; while Fig. 4 demonstrates that this is due to a higher number of transmissions or, in other words, retransmissions caused by message collisions.

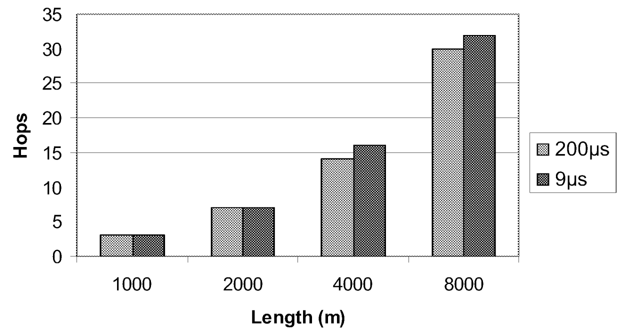


Figure 2: Average number of hops to cover the whole car platoon.

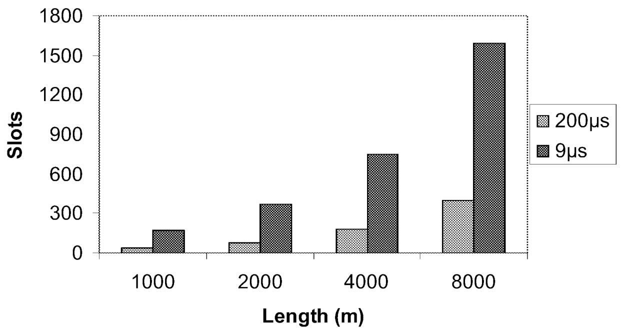


Figure 3: Average number of slots cumulatively waited at forwarding vehicles by a message.

On the other hand, having time slots about 20 time smaller allows the 9 µs setting to result in shorter total transmission time for each broadcast game event (see Fig. 5). Yet, we deem that 200 µs represents a better tradeoff among the requirements for limiting the number of message collisions and for having game events quickly covering the whole car platoon. Indeed, it has to be said that all the reported total delivery times are low enough to be acceptable for many interactive online games.

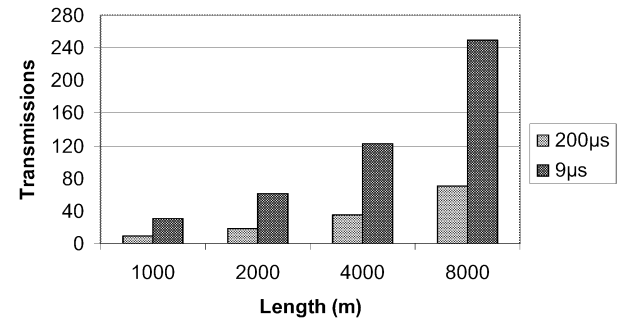


Figure 4: Average number of transmissions that a single game event experiences while covering the car platoon.

If, for instance, we considered just fast paced games (e.g., first person shooters, car races), the only case where the average transmission time of the 200 µs configuration surpasses their 150 ms interactivity threshold is when we consider long vehicular networks (i.e., 8 km). Therefore, it would be enough to limit the car platoon to a shorter maximum length, for instance 4-6 km to obtain game interactivity when employing the 200 µs time slot.

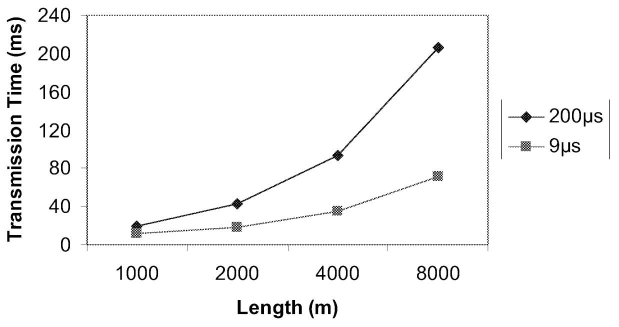


Figure 5: Average transmission time to cover the whole car platoon.

These outcomes and the need for conciseness encourage us in considering just the case of a 8 km long vehicular network for the tests reported in the rest of the paper.

Finally, as the main component of our scheme is represented by its transmission range estimator, we evaluate in Fig. 6 its efficiency in terms of the amount of time required to dynamically compute the factual transmission range that is currently available on each vehicle. As expected, with higher game message generation rates, our approach needs less time to compute the correct estimation. This is a logic consequence of the fact that the estimation is based on information about vehicles' positions and "hearing" distances included in exchanged messages. Therefore, the more the messages, the more the information received, and the quicker the correct estimation can be built. However, all the evaluated configurations show very little delays, 20 ms at most, which is a really encouraging value. This proves that our scheme is able to adapt itself extremely rapidly, which is a crucial feature in a highly dynamic scenario such as a vehicular network, especially when considering highly delay sensitive applications such as online games.

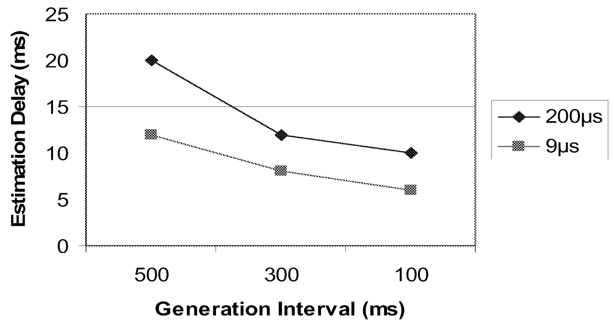


Figure 6: Average time to dynamically compute the factual transmission range.

Influence of the Number of Players

The simultaneous number of players that a game platform can support is important as it relates to higher revenues for game or network providers or just because humans are social beings: "the more we are, the funnier it is". Focusing on this aspect, we report in Fig. 7, Fig. 8, Fig. 9, and Fig. 10 results achieved by employing our scheme with a different number of simultaneous players, from 2 to 50.

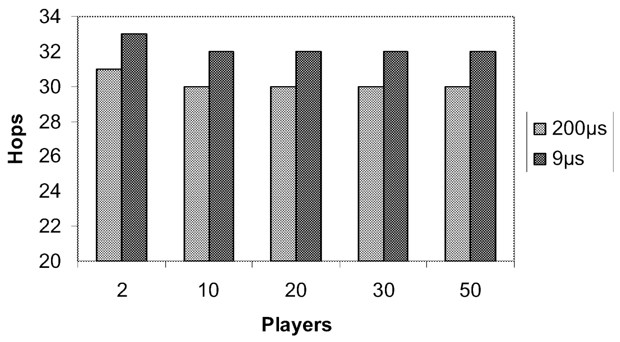


Figure 7: Average number of hops to cover the whole gaming car platoon.

As it is evident, results do not change significantly when increasing the number of players that are engaged in the online game. This resilience to a higher number of players, and hence to a more intense traffic on the channel, is obtained thanks to the ability of the approach in limiting the amount of game events simultaneously "on air". Indeed, each message is forwarded by only few vehicles and is quickly broadcast over the whole network, thus limiting the amount of time a message spend "on air" occupying shared resources. Analogous results in terms of resilience to player scalability are obtained even when considering different vehicular network's length. We hence omit to present them here as they would not bring any further information for our evaluation.

Considerations expressed in the previous subsection about the different slot durations hold even in this series of tests. Indeed, the case with 50 players in Fig. 7, Fig. 8, Fig. 9, and Fig. 10 corresponds to the case with 8 km of network length in Fig. 2, Fig. 3, Fig. 4, and Fig. 5, respectively.

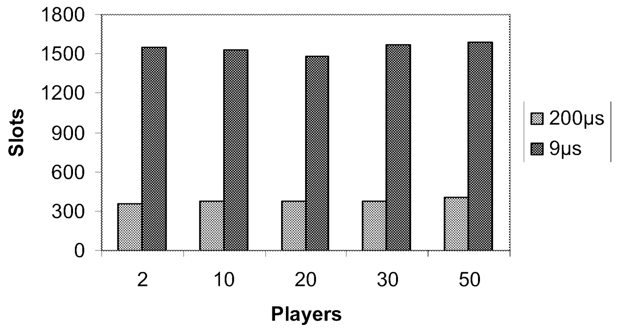


Figure 8: Average number of slots cumulatively waited at forwarding vehicles by a message while covering the car platoon.

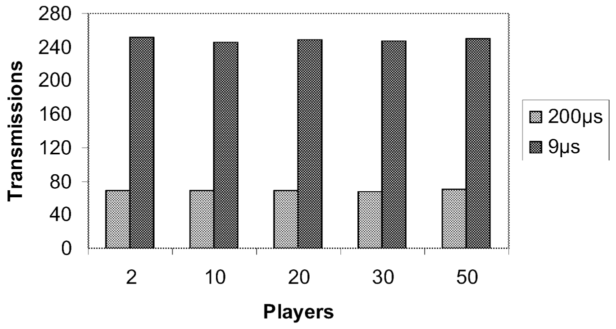


Figure 9: Average number of transmissions that a single game event experience while covering the car platoon.

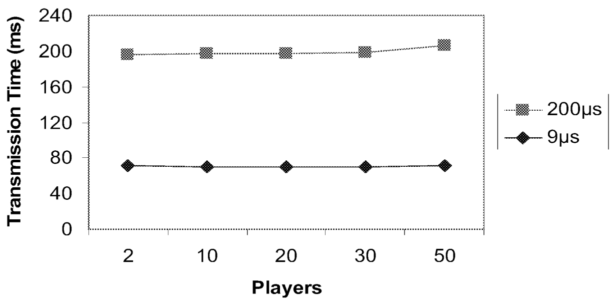


Figure 10. Average transmission time to cover the car platoon.

Our Scheme Against Other Approaches

We have compared FBA against other three possible solutions. For two of them, we have taken inspiration from the solution presented in (Fasolo et al. 2005). This scheme is similar to ours in that it attempts to have the farthest node in the transmission range to become the next-hop forwarder. The difference lies on the fact that the scheme in (Fasolo et al. 2005) simply assumes to have the transmission range parameter constantly set equal to a known predetermined value, rather than being able to dynamically compute it according to the factual channel conditions. We name this solution *Fixed300* if it utilizes 300 m as the transmission range parameter, and *Fixed1000* if it employs 1000 m.

Needless to say, Fixed300 and Fixed1000 perform ideally when the factual transmission range is indeed 300 m and 1000 m, respectively. In any other situation, the utilization of a wrong parameter could result in performance degradation.

We also evaluated *Random*, a solution that does not employ any distance prioritization. Simply stated, every car computes a random waiting time within the contention window before forwarding the message (if no one else already did it). The adopted contention window is initially set to *CWMin* and follows a general backoff mechanism by which its value doubles every time a transmission attempt results in a collision and decreases linearly with every successful transmission.

Specifically, we evaluate here the impact of the dispersion of networking vehicles on the system's performances. Fig. 11 shows the total transmission time when vehicles that participate in the generation or forwarding of game messages are placed every 20 m or every 80 m.

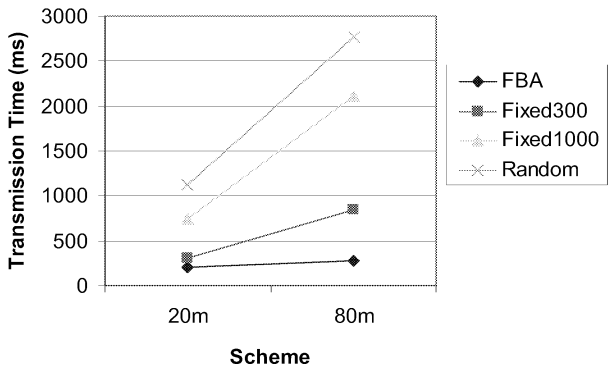


Figure 11. Average transmission time required to cover the whole gaming car platoon; 9 µs slot.

From the figure, our approach is the only scheme that is not heavily affected by a more dispersed set of networking nodes on which count on to forward the message. This is due to the fact that the simulative transmission range was equal to 300 m, with a factual one often around 280 m. Indeed, having a car every 20 m corresponds to have the farthest car in the factual transmission range exactly at 280 m from the sender, whereas with 80 m of space between successive cars, the farthest one in the factual transmission the transmission will be located at 240 m from the sender. Therefore, Fixed300 and, in particular way, Fixed1000 will be utilizing a transmission range parameter wrong with respect to the factual one; whereas FBA will adapt itself in virtue of its transmission range estimator, correctly computing 240 m.

CONCLUSIONS

In this paper we have evaluated FBA, a fast broadcast scheme which allows to quickly distribute game events in IVC systems. FBA exploits a transmission range estimator that, based on the information extracted from messages containing game events, provides a good measure of the maximum transmission distance in a given portion of the vehicular network. Then, a priority scheme is exploited to privilege farthest vehicles in taking charge of forwarding the message to other cars. Simulation results confirm the viability of our approach.

ACKNOWLEDGMENTS

Partial financial support for this work is provided by the Italian MIUR under the MOMA, and DAMASCO initiatives.

REFERENCES

Armagetron: a Tron clone in 3d, 2007, available at http://armagetron.sourceforge.net/.

S. Benford, D. Rowland, M. Flintham, A. Drozd, R. Hull, J. Reid, J., Morrison, K. Facer, "Life on the edge: supporting collaboration in location-based experiences", In *Proceedings of the SIGCHI Conference on Human Factors in Computing*

Systems (Portland, Oregon, USA, April 02 - 07, 2005). CHI '05. ACM Press, New York, NY, 721-730.

S. Biswas, R. Tatchikou, F. Dion, "Vehicle-to-Vehicle Wireless Communication Protocols for Enhancing Highway Traffic Safety", *IEEE Communication Magazine*, 44(1):74-82, Jan 2006.

S. Bjork, J. Falk, R. Hansson, P. Ljungstrand, "Pirates! Using the Physical World as a Game Board", *Proc. Interact 2001, 2001*, IFIP.

DSRC, *Dedicated Short Range Communications (DSRC) Home.* [Online]. 2007, Available at: http://www.leearmstrong.com/dsrc/dsrchomeset.htm

J. Farber, "Traffic Modelling for Fast Action Network Games," *Multimedia Tools and Applications*, vol. 23, no. 1, pp. 31-46, 2004.

E. Fasolo, R. Furiato, A. Zanella, "Smart Broadcast Algorithm for Inter-vehicular Communication", in Proc. of Wireless Personal Multimedia Communication (WPMC'05), Aalborg, DK, Sep 2005.

M. Guo, M. H. Ammar, E. W. Zegura, "V3: a vehicle-to-vehicle live video streaming architecture", in *Proc. of 3rd IEEE International Conference on Pervasive Computing and Communications*, Kauai, HI, Mar. 2005.

IEEE WLAN MAC, IEEE 802.11g, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4GHz Band.

IEEE 802.11g, IEEE 802.11g, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Amendment 4: Further Higher Data Rate Extension in the 2.4GHz Band.

G. Korkmax, E. Ekici, F. Ozguner, U. Ozguner, "Urban Multi-hop Bradcast Protocol for Inter-vehicle Communication Sustems", in Proc. of 1st ACM Workshop on Vehicular Ad-hoc Networks (VANET'04), Philadelphia, PA, Oct 2004.

NS-2, *The Network Simulator NS-2*, 2007. Available: http://www.isi.edu/nsnam/ns/

C. E. Palazzi, S. Ferretti, S. Cacciaguerra, M. Roccetti, "Interactivity-Loss Avoidance in Event Delivery Synchronization for Mirrored Game Architectures", IEEE Transactions on Multimedia, IEEE Signal Processing Society, vol. 8, no. 4, pp. 847-879, Aug 2006.

C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "How Do You Quickly Choreograph Inter-Vehicular Communications? A Fast Vehicle-to-Vehicle Multi-Hop Broadcast Algorithm, Explained", in *Proc. of IEEE International Workshop on Networking Issues in Multimedia Entertainment* (CCNC/NIME 2007), Las Vegas, NV, USA, Jan 2007.

C.E. Palazzi, M. Roccetti, S. Ferretti, G. Pau M. Gerla, "Online Games on Wheels: Fast Game Event Delivery in Vehicular Ad-hoc Networks", in *Proceedings of the 3rd International Workshop on Vehicle-to-Vehicle Communications 2007 (V2VCOM 2007) - IEEE Intelligent Vehicles Symposium 2007*, IEEE Computer Society, Istanbul (Turkey), June 2007.

L. Pantel, L. C. Wolf, "On the Impact of Delay on Real-Time Multiplayer Games," *in Proc. of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Miami, FL, USA, pp. 23-29 May 2002.

W. Piekarski, B. Thomas, "ARQuake: the outdoors augmented reality system" *CACM*, 45 (1), 36–38, 2002.

EDUCATION

CLIMA FUTURA @ VU[*]– COMMUNICATING (UNCONVENIENT) SCIENCE

Anton Eliëns, Marek van de Watering, Hugo Huurdeman, Winoe Bhikharie
Intelligent Multimedia Group, VU University Amsterdam
eliens@cs.vu.nl rvdwate@few.vu.nl hchuurde@few.vu.nl svbhikha@few.vu.nl

Haroen Lemmers, Pier Vellinga[†]
Climate Centre, VU University, Amsterdam
lemh@geo.vu.nl p.vellinga@falw.vu.nl

KEYWORDS
climate change, science communication, model-based simulation, serious games

ABSTRACT

In this paper we introduce Clima Futura, a game about climate change. The primary aim of Clima Futura is to gain experience with the parameters affecting climate change and to give access to climate change related research in a playful manner. The concept for the game has been developed as a submission for the yearly Dutch contest for the communication of science. In this paper we will give an overview of the scientific background of the game, the overall design of the game, and our approach for realizing the game, deploying a modular architecture which allows for extending the game with minigames contributed by the community of players.

INTRODUCTION

Over the last couple of years, climate change has come into the focus of public attention. Moved by television images of dislocated people in far-away countries, ice bears threatened by the corruption of their native environment, tsunami waves flooding the third world, and hurricanes destroying urban areas, the general public is becoming worried by what Al Gore has so aptly characterized as *an inconvenient truth*: the climate is changing and human affluence may be the prime cause. In response to the *pathos* of the media, many civil groups do an appeal on the responsibility of individual citizens and start campains for an *ethos* of climate-correct behavior, by saving on energy-consumption or driving CO2-friendly cars. In the media, such campains are either advocated or critized by authorities from public government, and experts from a multitude of sciences, with conflicting opinions. As a result, the general audience, initially with genuine concern about the state of our world, gets confused and looses interest. And more worrisome, the adolescents, looking at the serious way adults express their confusion and ignorance, take distance and may decide that the *climate issue* is not of their concern.

At the Climate Centre of the VU University Amsterdam, we are not happy to observe that *pathos* and *ethos* overtake the public debate, and we actively wish to participate in the public debate bringing our multidisciplinary scientific background into play. Moreover, since we *borrow the earth from our children*, as the old Indian saying goes, which Al Gore again brought to our attention, we feel that we must take an active interest in bringing the *climate issue* to the attention of the youth, in a form that is appropriate. From this background, we engaged in developing Clima Futura, a multi-disciplinary undertaking, bringing together climate experts from a variety of backgrounds with multimedia/game development researchers. 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. Given the state of our knowledge, the science of climate change itself may be characterized as a somewhat unconvenient science, and as such an interesting challenge to present by means of a game.

structure The structure of this paper is as follows. First, we will briefly discuss general issues of game design. Then we will describe the context, the science communication contest, the process of developing the concept for the game, and the actual design of Clima Futura. Before discussing the overall architecture of the game, we will characterize our *game event description format*, developed to allow for collaborative design, involving participants from a wide variety of disciplines. We will then outline the technical properties of our proposed game architecture, which accomodates extensions for special interest groups as well as contributions from the community of players.

[*]www.climafutura.nl
[†]also at University Wageningen

GAME PLAY, SIMULATION AND EXPLORATION

Games are increasingly becoming a vital instrument in achieving educational goals, ranging from language learning games, to games for learning ICT service management skills, based on actual business process simulations, Eliens & Chang (2007). In reflecting on the epistemological value of game playing, we may observe following Klabbers (2006), that the game player enters a *magic circle* akin to a complex social system, where *actors*, *rules*, and *resources* are combined in intricate (game) configurations:

game as social system

actors	**rule(s)**	**resource(s)**
players	events	game space
roles	evaluation	situation
goals	facilitator(s)	context

An often heard criticism on educational games is, unfortunately, that, despite the good intentions of the makers, they do not get the target audience involved, or put in other words, are quite boring. This criticism, as we will argue later, also holds for many of the climate games developed so far, and the question is how can we avoid this pitfall, and present the impact of climate change and the various ways we can mitigate or adapt to the potential threats of global warming in an entertaining way, that involves the player not only intellectually but also on a more emotional level? Put differently, what game elements can we offer to involve the player and still adequately represent the climate issue?

Looking at the games discussed in *Playing Games with the Climate*[1], we see primarily games that either focus on (overly simplified) climate prediction models (*logos*), or games that challenge the player how to become climate-correct (*ethos*). In our approach, we not only aim to include (well-founded) *logos* and *ethos* oriented game-playing, but also wish to promote an understanding of the *pathos* surrounding climate change, where we observe that the models taken as a reference are often gross simplifications and from a scientific perspective not adequate! To this end we will, as an extra ingredient, include interactive video as an essential element in game playing. This approach effectively combines a turn-based game-play loop, with a simulation-loop based on one or more climate reference models, with in addition exploratory cycles, activated by game events, which allow the player to explore the argumentative issues in the rethorics of climate change, facilitated by a large collection of interactive videos in combination with minigames.

[1]www.worldchanging.com/archives/003603.html

BUILDING THE TEAM

The *Academische Jaarprijs*[2] (yearly national Dutch prize for scientific communication) is a contest for bringing high-standing scientific research under the attention of the general public, including the younger generations! The VU University decided to submit their internationally well-renowned climate research[3], e.g. Kabat et al. (2005), as a candidate for the prize.

Looking for adequate means to communicate our scientific insights to the general audience, it took not long before the idea of a *game* came up. Both senior and junior staff of all relevant faculties were assembled to discuss the plan of a game, and an inventory was made of what games existed, followed by brainstorm sessions in which initial ideas were proposed.

Games we looked at included: *Planet Green*[4], offering ways to explore climate-correct behavior, the *ThinkQuest*[5] climate game, checking your knowledge for basic climate-related facts, the British *Climate Change Hero*[6] game, meant to improve the players knowledge about climate change factors, the German *Climate Simulator*[7], which allows for experimentation with climate change based on a simulation model, and the BBC game *Climate Challenge*[8], where the player must take decisions to tackle climate change and yet stay popular. But none of these games seemed to be satisfactory as a basis for our game, although each of them provided some inspiration, one way or another.

When we came accross a serious game in an altogether different domain, we nevertheless did find the inspiration we were looking for. In the ground-breaking *Peacemaker*[9] game, we found an example of how to translate a serious issue into a turn-based game, which covers both political and social issues, and with appealing visuals, not sacrificing the seriousness of the topic. By presenting real-time events using video and (short) text, Peacemaker offers a choice between the points of view of the various parties involved, as a means of creating the awareness needed for further political action. With Peacemaker as an example after which to model our climate game, we started working on the design of a turn-based game, allowing the player to manipulate parameters of climate change over a period of time, against the background of a climate simulation model, and offering the opportunity to explore climate-related issues and opinions, using interactive video or by playing minigames. Clima Futura was born!

[2]www.academischejaarprijs.nl
[3]www.climatecentre.vu.nl
[4]planetgreengame.com
[5]library.thinkquest.org/5721/climategame.html
[6]www.devon.gov.uk/index/environment/climatechange
[7]www.deutsches-museum.de/dmznt/climate/climategame
[8]www.bbc.co.uk/sn/hottopics/climatechange
[9]www.peacemakergame.com

CONCEPT – CLIMA FUTURA

The Clima Futura game is targeted at an audience in the age of 12-26. Primary goals are to create involvement with the *climate issue*, and to provide information by allowing the player to explore cause and effect relations, using models based on scientific research in a continuously evolving field of knowledge.
Clima Futura is a turn-based game, with 20 rounds spanning a 100-year period. In each turn, the player has the option to set parameters for the climate simulation model. The game is centered around the so-called *climate star*, which gives a subdivision of topics in climate research, as indicated below.

- climate strategies – (1) emission reduction, (2) adaptation
- climate systems – (3) feedback monitoring, (4) investment in research, (5) climate response
- energy and CO2 – (6) investment in efficiency, (7) investment in green technology, (8) governement rules
- regional development – (9) campain for awareness, (10) securing food and water
- adaptation measures – (11) public space, (12) water management, (13) use of natural resources
- international relations – (14) CO2 emission trade, (15) European negotiations, (16) international convenants

Of the topics mentioned, not all may immediately be represented in the simulation model underlying Clima Futura, but may only be addressed in exploratory interactive video. The *climate star* is actually used by the VU Climate centre as an organizational framework to bring together researchers from the various disciplines, and in the Clima Futura game it is in addition also used as a *toolkit* to present the options in manipulating the climate simulation model to the player.
The result parameters of the climate simulation model are for the player visible in the values for *People*, *Profit* and *Planet*, which may be characterized as:

- *People* – How is the policy judged by the people?
- *Profit* – What is the influence on the (national) economy?
- *Planet* – What are the effects for the environment?

A generally acknowledged uncertainty within climate research surrounds the notion of *climate sensitivity*, that is the extent to which the climate and climate change is actually dependent on human activity. In practice, the actual assessment of climate sensitivity may determine whether either a choice for mitigation or adaptation is more viable.
In the Clima Futura game we choose for using *climate sensitivity* as as a parameter for setting the level of difficulty of the game play, where difficulty increases with the value for climate sensitivity.

To give an example of game play, we let the player start in 2007, the year the IPCC[10] (Intergovernmental Panel on Climate Change) report was published. In each subsequent round, the player may choose to undertake action. For example, when the player decides to enforce restrictions on CO2 emissions, s/he may choose option (1) in the climate star, which can be reached through *climate strategies*. The result will then be visible, after some period of (game) time, in either one of the result parameters, *People*, *Profit*, and *Planet*.
The climate simulation model[11] underlying Clima Futura is primarily based on the Climber 2.0 model, which is used for scientific simulations of climate change, based on the division between land and sea, the density of vegetation, sea temperature, and the amount of CO2. Economic costs and benefits of climate policy options are calculated by means of an integrated assessment model coupled to the climate model. Additionally, an alternative model, the MERGE[12] model is used, which gives a flexible means to explore a wide range of contentious issues: costs of abatement, damages from climate change, valuation and discounting. MERGE contains submodels governing domestic and international economy, energy-related and non-energy related emissions of greenhouse gases, as well as market and non-market damages due to global climate change.
As an aside, the choice of models[13] is in itself a controversial scientific issue, as testified by J. D. Mahlman's article on the rethorics of climate change *science versus non-science*[14], discusssing *why climate models are imperfect and why they are crucial anyway.*

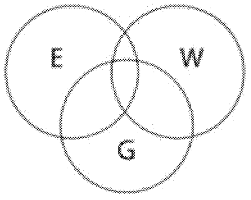


Fig 1. Game play, Simulation, Exploration

In summary, see fig. 1, 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 – by means of interactive video (E)

Each of the three elements is essentially cyclic in nature, and may give rise to *game events*. For example, game events may arise from taking turns after 5-year periods, due to alarming situations in the climate simulation, such as danger of flooding an urban area, or accidental

[10]www.ipcc.ch
[11]en.wikipedia.org/wiki/Climate_model
[12]www.stanford.edu/group/MERGE
[13]www.grida.no/climate/ipcc_tar/wg1/308.htm
[14]www.gfdl.noaa.gov/~gth/web_page/article/aree_page1.html

access to confidential information in the exploration of video material. 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. Clima Futura also features *advisors* that may be consulted, to gain information about any of the topics of the *climate star*.

GAME DESCRIPTION FORMS

Having decided on the general structure and elements of the Clima Futura game, a turn-based game loop, a climate-model driven simulation, exploratory video, and mini-games, the problem is how to connect these elements in a meaningfull way, and design a coherent collection of game events. This problem is further aggravated by the need to find a way to design in a collaborative fashion, necessitated by the sheer amount of disciplines and people involved.

To enable collaborative design we developed a game event description format, which standardizes the way game events are to be described, and for which we also developed an online form, structured as outlined below:

- name of event – give a meaningful name
- event-id – for administrators only
- type – (generic/specific) game/model/video
- cause – game play/simulation/exploration
- feedback/information – give a logical description
- player actions – indicate all (logical) player options
- description of visuals – for feedback, information and player options
- additional information – give a url with references to additional informatin and visuals
- relates to event(s) – give id's or descriptions of related events

Before enforcing the game event description format, our ideas about the design of Clima Futura were gathered in a collection of narratives and brief descriptions, in what we called the *Clima Futura Design Bible*. Using the standardized game event description format, we hope to arrive at a more uniform way of describing the narratives, the perspectives from which these narratives can be experienced, the challenges or problems a player must solve, the resources available to the player, such as capital, knowledge and political power, the rewards, possibly using bonus credits for succesfully playing a mini-game, as well as the visuals, which will where possible be derived from the collection of videos we have available.

For the elaboration of the design, we are developing storyboards, which characterize in a visual way the major (dramatic) elements of narratives, structured using a subdivision in:

scenario(s)

- context – general setting, situation
- problem – event(s) to occur, problem to solve
- S-R situation(s) – stimulus/response (one or more)
- climax – action must be taken
- resolution – find solution or result

Although the actual workflow that we will deploy during development is at the moment of writing not clear, we will strive for developing templates that allow for a quick realization of the designs captured by the game event and minigame description format(s), along with the storyboards for visual design.

A MODULAR ARCHITECTURE

In the beginning, we envisioned the realization of our climate game as a first-person perspective role-playing game in a 3D immersive environment as for example supported by the Half Life 2 SDK, with which we gained experience in creating a *search the hidden treasure*[15] game in a detailed 3D virtual replica of our faculty. However, we soon realized that the use of such a development platform, would require far too much work, given the complexity of our design. So, instead of totally giving up on immersion, we decided to use *flash* video[16], indeed as a poor-man's substitute for real 3D immersion, which, using *flash*[17] interactive animations, has as an additional benefit that it can be used to play games online, in a web browser. Together with the Flex 2 SDK[18], which recently became open source, *flash* offers a rich internet application (RIA) toolkit, that is sufficiently versatile for creating (online) games, that require, in relation to console games or highly realistic narrative games like Half Life, a comparatively moderate development effort. To allow for component-wise development, we choose for a modular architecture, with four basic modules and three (variants) of integration modules, as indicated below, in fig 5.

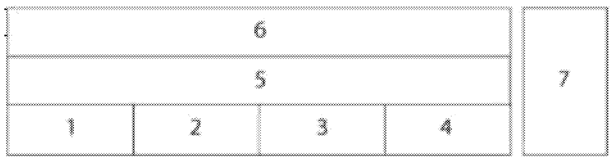


Fig 2. Clima Futura Architecture

1. climate model(s) - action script module(s)
2. game play interaction - event-handler per game event

[15]www.cs.vu.nl/~eliens/game
[16]www.adobe.com/products/flash/video
[17]ww.adobe.com/devnet/flash
[18]www.adobe.com/products/flex/sdk

3. video content module - video fragment(s) and interaction overlays
4. minigame(s) - flash module(s) with actionscript interface
5. Clima Futura - integration of modules 1-4, plus server-side ranking
6. adapted versions – educational, commercial
7. multi-user version –with server-side support

In addition, we would like to develop a facility that allows players not only to submit their own video material, but also to build or modify their own minigames, which might then be included in the collection of minigames provided by Clima Futura.
For the actual production, we will use additional components, including *game physics*[19], a *relation browser*[20], and an *earth*[21] component. In particular, both physics and in-game building facilities seemed to have contributed to a great extent to the popularity of Second Life, Eliens et al. (2007). In creating *digital dossiers*[22] for contenporary art, we have deployed concept graphs, that is a relation browser, to give access to highly-related rich media information about art in an immersive manner. Finally, given the topic of Clima Futura, being able to visualize models of the surface of the earth seems to be more than appropriate. It is interesting to note that our technology also allows for the use of *flash* movies directly by invoking the *youtube* API[23] as a web service, which means that we could, in principle, build minigames around the evergrowing collection of *youtube*, or similar providers.
Providing flexible access to collections of video(s) to support arguments concerning controversial issues has been explored in, among others Vox Populi[24].
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. Although a research topic in itself, we would like to develop a *video content module* (3), that provides flexible access to the collection of video(s), and is media driven to the extent that video-material can be added later, with proper annotation. Together with in-game minigame building facilities, it would be in the spirit of a participatory culture, to provide annotation facilities to the player(s) of Clima Futura as well, to comment on the relevance and status of the video material,

[19]www.fisixengine.com
[20]http://der-mo.net/relationBrowser
[21]www.flashearth.com
[22]www.few.vu.nl/∼dossier05
[23]www.youtube.com/dev
[24]homepages.cwi.nl/∼media/demo/IWA/

CONCLUSIONS

To present the concept of Clima Futura, we decided to have three central presenters (anchors) and an expert-panel (choir), that may comment on detailed scientific or technical issues. The presentation, stressed the multi-disciplinary approach.
Although it too early to look back, we may on reflection ask attention for another potential pitfall, that endangers any educational game, once aptly expressed by Sartre in his criticism of *l'esprit de serieux*. Indeed, we may become too serious! In concluding our account of the design and development of Clima Futura, we may refer to an *ontology of humour*, Dormann et al. (2007), that may be taken as a guideline to avoid the common pitfall of *serious games*. In brief, Dormann et al. (2007) distinguishes between three theories of humour, that each denote a particular function of humour: *relief theory*, which explains humour as a reduction of stress, *superiority theory*, which asserts that humour has a social function, as a means to enforce the norm of a group or culture, and *incongruity theory*, which relates humour to the discovery of hidden meanings. We leave it to the imagination of the reader to establish in what way the various types of humour may be put to effect in the *climate issue*!

Acknowledgement(s) We thank all (other) members of the Clima Futura Team[25]: Frans Berkhout, Peter van Bodegom, Merlijn Draaisma, Alex Halsema, Thijs Louisse, Anne Martens, Karlien Meulenaars, Elia Orru, Frans-Jan Parmentier, Pieter Pauw, Rob Schuddeboom, Charlotte Spliethoff for their enhusiasm and effort, and Paulis Klerk (Harlequin), and Suzanne Waldau (Ex'Machina), for there suggestions and support in the realization of Clima Futura.

REFERENCES

Dormann C., Barr P. and Biddle R. (2007), Humour Theory and Videogames: Laughter in the Slaughter, In *Proc. of the 2006 ACM SIGGRAPH symposium on Videogames*

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. (2007), VU @ Second Life – creating a (virtual) community of learners, In *Proc. EUROMEDIA 2007*

Kabat P., van Vierssen W., Veraart J. Vellinga P., Aerts J. (2005). Climate proofing the Netherlands, Nature 438, pp. 283-284

Klabbers J.H.G. (2006), *The Magic Circle: principles of Gaming and Simulation*, Sense Publishers

[25]www.climafutura.nl/team/

MASHUPS IN SECOND LIFE @ VU

Anton Eliëns
FEW
VU University
Amsterdam
eliens@cs.vu.nl

Frans Feldberg
FEWEB
VU University
Amsterdam
jfeldberg@feweb.vu.nl

Elly Konijn
FSW
VU University
Amsterdam
ea.konijn@fsw.vu.nl

Egon Compter
Communicatie
VU University
Amsterdam
e.compter@dienst.vu.nl

KEYWORDS
Second Life, Web Services, Mashups, Virtual Economy, Serious Games, User Tracking

ABSTRACT

In this paper we explore how to enhance our presence in Second Life by utilizing Web Services in meaningful compositions (mashups). After discussing the technical requirements that must be met, we discuss possible applications of mashups in Second Life, including serious games, and delineate a behavioral model that allows for tracking the behavior of visitors of our world. Taking our requirements analysis and envisioned design, which essentially includes the 3D nature of Second Life worlds, as a starting point, the paper provides an overview of research and development(s) that may contribute to the realization of meaningful mashups and serious games in Second Life.

INTRODUCTION

Second Life seems to be overtaking the world. In the whole range of cummunity-building platforms, Second Life stands out as an immersive 3D world with an almost stunning adoption, by both individuals, companies and institutions, followed attentively by the Press. Not entirely without an understanding of the value of press coverage, the VU University Amsterdam decided to create presence in Second Life, by creating a virtual campus, to realize a (virtual) community of learners, Eliens et al. (2007). And, indeed, we succeeded in being the first university in The Netherlands with presence in Second Life and, as hoped, this was covered in the 8 o'clock nation-wide TV news.
More substantial than getting into a nation-wide television broadcast, however, is our aim to communicate our institutional goals, *creating a community of learners*, by creating a virtual campus in Second Life, offering *an information portal* as well as *a meeting point*, in a media platform that is widely adopted by our target community. Virtual presence in Second Life, obviously, is not enough. The relatively long history of virtual worlds has shown that lack of interesting content and functionality easily leads to boredom, desinterest, and hence *churn*, users dropping off. As a consequence, there is a need for sustainable functionality, that both motivates people to come back and participate, and, otherwise why choose Second Life, makes essential use of the 3D immersive environment offered by Second Life. In this paper, we will explore how to use web services in meaningful compositions or mashups to enhance our presence in Second Life, and create a community where visitors actively participate in both education and research,

structure The structure of this paper is as follows. First, we will briefly describe the construction of our virtual campus and discuss the likely success factors of Second Life. and indicate how to use Second Life as a platform for interaction and serious games. We will investigate what technological support is needed to create mashups in Second Life and, after that, whether Second Life offers the functionality needed to incorporate web services. Further, we will characterize applications based on web services that make essential use of the 3D environment and fit within the virtual economy as it exists in Second Life, in particular serious games and (corporate) awareness systems. And, finally, in before giving our conclusions, we will sketch a behavioral model to characterize user interaction in Second Life, that may contribute to the realization of meaningful mashups and serious games in Second Life.

VU @ SECOND LIFE

What has been characterized as a shift of culture, from a media consumer culture to a participatory culture, Jenkins (2006), where users also actively contribute content, is for our institution one of the decisive reasons to create a presence in Second Life, to build a virtual platform that may embody our so-called *community of learners*, where both staff and students cooperate in contributing content, content related to our sciences, that is. In December 2006, we discussed the idea of creating presence in Second Life. Our initial targets were to build a first prototype, to explore content creation in Second Life, to create tutorials for further content creation, and to analyze technical requirements and opportunities for deployment in education and research. Two and

a half months later, we were online, with a virtual campus, that contains a lecture room, a telehub from which teleports are possible to other places in the building, billboards containing snapshots of our university's website from which the visitors can access the actual website, as well as a botanical garden mimicking the VU Hortus, and even a white-walled experimentation room suggesting a 'real' scientific laboratory. All building and scripting were done by a group of four students, from all faculties involved, with a weekly walkthrough in our 'builders-meeting' to re-assess our goals and solve technical and design issues. The overall style is realistic, although not in all detail. Most important was to create a visual impression of resemblance and to offer the opportunity to present relevant infomation in easily accessible, yet immersive, ways. Cf. Bolter & Grusin (2000).
As we argue in Eliens et al. (2007), the surprising success and appeal of Second Life may be attributed to an optimal combination of avatar modification options, gesture animations, in-game construction tools, and facilities for communication and social networking, such as chatting and instant messaging. Incorporating elements of community formation, and very likely also the built-in physics and the inclusion of elementary economic principles, seem to be the prime distinguishing factors responsible for the success of Second Life. In addition, the possibility of recording collaborative enacted stories, using built-in *machinima*[1] certainly may contribute to its appeal.
The goal of this paper is to explore the use of web services for, among others, the creation of serious games, and to provide a behavioral model that allows us to give an interpreation to users' behavior patterns, and may perhaps even help to guide users' behavior in Second Life, by providing appropriate recommendations.

WEB SERVICES & MASHUPS

By now the phrase *Web 2.0* as well as applications representing it, such as Flickr and YouTube, are well established, and enjoyed by a wide community. Each day new items are added to the growing list of mashups[2], and the number of web services that constitute the building blocks of mashups also shows a steady growth. Mashups seem to be the easy way to start up a company, since the technology is relatively easy and, making use of appropriate services, initial investment costs can be low. Cf. Shanahan (2007).
What Web 2.0 stands for, from a technical perspective, is succinctly expressed in Dorai's Learnlog[3] *XML Is The Fabric Of Web 2.0 Applications*:

- client side is AJAX (Asynch. Javascript and XML)
- server application typically exposes data through XML
- the interaction model is web services
- mashups combine multiple webservices

And eventhough many alternative representations, such as JSON[4] (Javascript Object Notation) are increasingly being used, all in all XML may be regarded as the *interlingua* of the Web 2.0.
Before taking a closer look at the communication protocol(s) underlying Web 2.0 and de-construct the tight link of AJAX to HTML in-page formatting, it is worthwhile, following Shanahan (2007), to give an overview of a selected number of services, that may be used to create mashups:

services

- google – code.google.com/
- yahoo – developer.yahoo.com/
- del.icio.us – del.icio.us/help/api/
- flickr – www.flickr.com/services/
- bbc – www0.rdthdo.bbc.co.uk/services/
- youtube – www.youtube.com/dev

Although mashups featuring google maps seem to be the dominant mashup type, other services such as offered by del.ici.us, Flickr and BBC might prove to be more worthwhile for 'serious' applications. For example, for developing e-commerce applications Amazon[5] offers services for *product operations*, such as item search and similarity lookup, *remote shopping carts*, to create and manage purchase collections, *customer content*, to access information contributed by customers, and *third party listings*, to find related resellers. It is important to note that many of these services, as for example the *shopping cart* services, may be used independently of the commercial offerings of Amazon!
Most of the service providers and services mentioned above are accessible using a choice of protocols, including WSDL, SOAP, XML-RPC and the REST protocol. The REST protocol seems to be most widespread and as we will discuss in the next section, it seems to be tho most appropriate protocol in Second Life.
REST stands for *Representational State Transfer*. In essence, the REST protocol uses the url as a command-line for stateless RPC invocations, which allows for services to be executed by typing in the address box of a web browser. A great tutorial about the REST protocol can be found in Joe Gregorio's column[6]: *The Restful Web*. As fully explained in Van der Vlist et al. (2007), the phrases *representation*, *state* and *transfer*, respectively, stand for:

REST

- representation – encoding in a particular format
- state – data encapsulated in an object

[1]www.machinima.org
[2]www.programmableweb.com/mashuplist/
[3]dorai.wordpress.com/tag/mashups/
[4]www.json.org/
[5]aws.amazon.com
[6]www.xml.com/pub/a/2004/12/01/restful-web.html

- transfer – using HTTP methods

In practice, the use of REST means that the state associated with a resource or service must be managed by the client. Together with mechanisms such as content-negotiation and URL-rewriting, REST provides a simple, yet powerful method to invoke services using HTTP requests.

The Web 2.0 offers a lively arena for consumers and developers alike, with a multitude of blogs discussing the future of the web. For example, in Dion Hinchcliffe rebuttal[7] of Jeffrey Zeldman's *Web 3.0 Web 1.0 = Web 2.0* blog, entitled *Is Web 2.0 Entering "The Trough of Disillusionment"?* it is suggested that *our services could even be more powerful* by creating *semantic mashups*[8].

To conclude this brief overview of web services and mashups we wish to give another quote from Dorai's Learnlog, this time from Jon Udell, in his blog on his move to Microsoft:

> *The most powerful mashups don't just mix code and data, they mix cultures.*

which provides a challenge that trancends all issues of mere technological correctness.

INFRASTRUCTURE

Second Life offers an advanced scripting language with a C-like syntax and an extensive library of built-in functionality. Although is has support for objects, LSL (the Linden Scripting Language) is not object-oriented. Cf. Eliens (2000). Scripts in Second Life are server-based, that is all scripts are executed at the server, to allow sharing between visitors. Characteristic for LSL are the notions of *state* and *eventhandler*, which react to events in the environments.

Among the built-in functions there are functions to connect to a (web) server, and obtain a response, in particular (with reference to their wiki page):

built-in(s)

- request – wiki.secondlife.com/wiki/LlHTTPRequest
- escape – wiki.secondlife.com/wiki/LlEscapeURL
- response – wiki.secondlife.com/wiki/Http_response

Other functions to connect to the world include *sensors*, for example to detect the presence of (visitors') avatars, and chat and instant messaging functions to communicate with other avatars using scripts. In addition, LSL offers functions to control the behavior and appearance of objects, including functions to make objects react to physical laws, to apply force to objects, to activate objects attached to an avatar (as for example phantom Mario sprites), and functions to animate textures, that can be used to present slide shows in Second Life.

[7]web2.sys-con.com/read/172417.htm

[8]www.web2journal.com/read/361294.htm

On the Mashable[9] *Social Networking News* site a brief overview is given of the use of web services in Second Life, entitled *Second Life + Web 2.0 = Virtual World Mashups.* To access Second Life from outside-in (that is from a web browser), so-called *slurls* may be used, for example to reach VU[10] @ Second Life, and all slurls listed in del.icio.us under *slurlmarker*[11] may be used, also to activate in-world teleporting using scraping techniques.

As remarked in the *hackdiary*[12] by Matt Biddulph, Second Life (currently) lacks the ability to parse XML or JSON, so the best way to incorporate web services is to set up a web server with adequate resources. As Matt Biddulph indicates, to access *flickr* photographs for a particular user (avatar), a web server may contain the following resources:

web server

- /seen?user=SomeAvatar – records the presence of SomeAvatar
- /touched?user=SomeAvatar – invokes flickr API with users tag
- /set_tag?user=SomeAvatar&tag=FavoriteTag – records SomeAvatar's favourite tag

For example, in response to a 'touch' event, invoking *touch* results in consulting the database for the user's tag and asking the Flickr API for a random photo with that tag. It then returns a string containing the url for a particular photograph. LSL functions used in this application include *sensors*, to check for presence, *listen* functions, to respond to spoken commands, and *touch* events, for the physical interface. In addition to supporting strings and lists, LSL provides a perl-like split function to convert a string into a list of strings, thus allowing for processing multiple items in response to a server request.

Another example of using web services in Second Life is writing blogs[13] from within Second Life using the BlogHUD[14] developed by Koz Farina who also is reported to have found a flash hack that allows for reading RSS feeds.

The RSS display uses the ability to stream Quicktime video in Second Life, and again the mashup is not created in Second Life but by appropriate server support. In a similar vein we may incorporate live streaming video[15], for example by using WireCast[16] to capture and organize live camera input, possibly together with screen output of other applications such as *powerpoint*, which must then be sent to a streaming server supporting

[9]mashable.com/2006/05/30/second-life-web-20-virtual-world-mashups/

[10]slurl.com/secondlife/VU%20University%20NL/29/151

[11]del.icio.us/tag/slurlmarker

[12]www.hackdiary.com/archives/000085.html

[13]nwn.blogs.com/nwn/2006/10/really_simple_s.html

[14]bloghud.com/

[15]blogs.electricsheepcompany.com/chris/?p=206

[16]www.varasoftware.com/products/wirecast/

Quicktime, such as Apple's Darwin[17], which may then be accessed from Second Life to texture a display object. Finally, as another *Web 2.0 to Web 3D* phenomenon, announced in New World Notes[18], we may mention the used of Twitter[19] messages, that allow residents to send and receive message about ongoing activities. A similar service is reported to exist for *jaiku*[20] messages.

VIRTUAL ECONOMY

Mashups on the Web are interesting representatives of what one may call a *virtual economy*, with a business-model that is not grounded in traditional *production* and *trade* values, but rather consists of value-added services with an indirect, albeit substantial, financial spin-off, due to recommendations and referrals. The basic mechanisms in a recommender economy are, according to Kassel et al. (2007):

- cross sale – users who bought A also bought B
- up sale – if you buy A and B together ...

Where the principles underlying this virtual economy have definitely proven their value in first (ordinary) life economy, what are the chances that these principles are also valid in Second Life?
According to the media companies selling their services to assist the creation of presence in Second Life, there are plenty *New Media Opportunities In The Online World Second Life*[21], to a possibly even greater extent, as they boldly claim, as in what they call *the predessor of Second Life, the World Wide Web.*
To assess the role web services, including semantic web services, may play in Second Life, it seems worthwhile to investigate to what extent web services can be deployed to deliver more traditional media, such as *digital TV*. To support the business model of digital TV, which in outline may be summarized as *providing additional information*, *game playing* and *video on demand*, with an appropriate payment scheme, Daskalova & Atanasova (2007) argue in favor of the use of a SOA (Service Oriented Architecture), to allow for a unified, well-maintainable approach in managing collections of audio-visual objects. Such services would include meta-data annotation, water-marking for intellectual property protection, and search facilities for the end-user.
With respect to the application of web services in Second Life, however, a far more modest aim, it seems that nevertheless the business model associated with the delivery of media items through digital TV channels may profitably be used in Second Life, and also the idea of wrapping media items in web services has in some way an immediate appeal.

[17]developer.apple.com/opensource/server/streaming/
[18]nwn.blogs.com/nwn/2007/03/post_1.html
[19]twitter.com/
[20]devku.org/docs
[21]www.youtube.com/watch?v=8NOHRJB9uyI

Leaving the economic issues aside we will briefly consider two applications that we envisage to realize within the virtual campus of VU @ Second Life. The first application is Clima Futura[22], a game meant to give information about climate change.
Technical issues in realizing Clima Futura in Second Life are support for ranking, as well as meta-information with respect to locations where relevant information can be found, which may be realized with the techniques indicated previously. Another issue is giving flexible access to video material related to specific topics in climate change.
The other application we wish to discuss is PANORAMA, Vyas et al. (2007). In developing PANORAMA we proceeded from the assumption that people somehow like to have a feel of what is going on in the workspace, although not in any detail, and also like to see items of personal interest, including (their own) birth-announcements and sport troffees.
Embedding PANORAMA in Second Life would allow us to observe, in more detail than in a previous user study, the behavior of users, that is, to be more precise, the proximity to particular objects of interest, the duration of their presence, and, using the mechanisms of recommendation, their interest in related items. Technically, such monitoring can be achieved using the sensors and listeners described before. To make sense of such data, however, we need some model that allows for an interpretation that is more meaningful than the mere registration of presence.

TRACKING INTERACTION

Our virtual campus in Second Life already allows for performing simple statistics, by recording the presence of users at particular spots in the virtual world, using sensors and listeners installed in 3D objects. Since the LSL script-based counters appear to be rather volatile, tracking data are sent to a web server and stored in a database. This mechanism can easily be extended to a more encompassing form of user tracking, recording for a particular user not only presence at particular spots, but also the duration of presence, the actual proximity to objects, and the proximity to other users, as well as explicitly spoken comments or actions such as the donation of (Linden) money. For example, observing that a user spends a particular amount of time and gives a rating r, we may apply this rating to all features of the item, which will indirectly influence the rating of items with similar features.
This does of course not explain nor how ratings come into existence, nor what features are considered relevant, or even how guided tours should be generated. However, as we have demonstrated in Ballegooij & Eliens (2001), based on a rudimentary tagging scheme, we may in

[22]www.climafutura.nl

response to a query generate a guided tour taking the topographical constraints of the virtual world into account, for example to make a user familiar with the (virtual replica of the) actual workspace. It seems that this approach can be generalized to one that uses alternative descriptive methods, as long as they support feature-based information retrieval[23].

Obviously, both user tracking and recommendations may be fruitfully used in the realization of serious (corporate) games, as well as to support exploratory activity in non-serious games and (corporate) awareness systems.

CONCLUSIONS

Based on an overview of research and development(s) in web technologies and our assessment of the technical facilities offered by the Second Life platform, we may conclude that there are ample opportunities to incorporate web services and mashups in Second Life. Our intended applications, moreover, covering (corporate) game playing as well as a system for promoting (corporate) social awareness, indicate that there are clear motivations to deploy web services in Second Life, both for tracking users' behavior and for providing additional information based on recommendations that may be derived from taking record of users' behavior patterns. Although we have sketched a first behavioral model that allows to assign meaning to behavior and interaction, it is clear that this model must be further refined and that we need to gain experience in developing mashups in virtual space to arrive at effective and meaningful compositions of web services, supporting the realization of (serious) games, in Second Life.

Acknowledgement(s) We (again) thank the students involved, Viola van Alphen (FEWEB), Tom Bestebreurtje (FEW), Elbert-Jan Hennipman (FEW), and last but not least Bjorn de Boer (FSW), for their enthusiasm, creativity, and of course their hard work which led to an amazing result.

REFERENCES

Atanasova T., Nern H.J., Dziech A. (2007), Framework Approach for Search and Meta-Data Handling of AV Objects in Digital TV Cycles, Workshop on Digital Television, Proc. EUROMEDIA 2007, Delft, Netherlands

Ballegooij A. van and Eliens A. (2001), *Navigation by Query in Virtual Worlds*, In: *Proc. Web3D 2001 Conference*, Paderborn, Germany, 19-22 Feb 2001

Bolter J.D and Grusin R. (2000), *Remediation – Understanding New Media*, MIT Press

Daskalova H. and Atanasova T. (2007), Web Services and Tools for their Composition considering Aspects of Digital TV Workflow, Workshop on Digital Television, Proc. EUROMEDIA 2007, Delft, Netherlands

Eliens A. (2000), *Principles of Object-Oriented Software Development*, Addison-Wesley Longman, 2nd edn.

Eliens A. and Chang T. (2007), *Let's be serious – ICT is not a (simple) game*, In:*Prc. FUBUTEC 2007*, April 2007, Delft

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

Jenkins H. (2006), *Confronting the Challenges of Participatory Culture: Media Education for the 21th Century*, White Paper, MIT MediaLab

Kassel S., Schumann C-A. and Tittman C. (2007), Intelligent Advertisement for E-Commerce, In *Proc. EUROMEDIA 2007*, Delft, Netherlands

Nern H.J., Dziech A., Dimtchev and Jesdinsky (2007), Modules for an Integrated System Approach for Advanced Processing of AV Objects in Digital TV Workflow, Workshop on Digital Television, Proc. EUROMEDIA 2007, Delft, Netherlands

Rymaszewski M., Au W.J., Wallace M., Winters C., Ondrejka C., Batstone- Cunningham B. (2007). *Second Life – the official guide*, Wiley

Shanahan F. (2007), *Amazan.com Mashups*, Wiley Publishing Inc.

Van der Vlist E,, Ayers D., Bruchez E.,, Fawcett J. and Vernett A. (2007). *Professional Web 2.0 Programming*, Wiley Publishing Inc.

Vyas D., van de Watering M., Eliens A., van der Veer G. (2007), *Engineering Social Awareness in Work Environments*, accepted for *HCI International 2007*, 22-27 July, Beijing, China

[23]www.cs.vu.nl/∼eliens/research/rif.html

TEACHING AI CONCEPTS BY USING CASUAL GAMES: A CASE STUDY

Cesar Tadeu Pozzer
Santa Maria Federal University (UFSM)
Centro de Tecnologia
Santa Maria, RS, Brazil
pozzer@inf.ufsm.br

Börje Karlsson
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
ICAD/IGAMES/VisionLab
Rio de Janeiro, RJ, Brazil
borje@inf.puc-rio.br

KEYWORDS
Games in education, artificial intelligence, computer science.

ABSTRACT

Nowadays it is not uncommon for computer games to be used as tools to help introduce basic computer science concepts. In this paper we argue that games could also be used in more advanced subjects. We propose a new approach where applications can be easily developed to play games. In our case, games are used as support to teach programming and AI techniques, among other areas. A case study is described, showing the overall principle and the design of an environment we developed on which students will work in building game player software that incorporates specific AI algorithms to solve a given problem. Our testbed is a popular gem-swapping puzzler.

INTRODUCTION

Digital games have a huge appeal towards students, especially CS students, who generally spend a large part of their entertainment time playing games. Games can attract user attention in many different ways. Some people enjoy playing games; some prefer designing and developing them rather than playing games; others even like to watch other people playing games. Many researchers claim some kind of consensus status that games can motivate students, provide positive experiences, and might be a good incentive to learning (Swedy et al. 2005). Digital games are especially popular in coursework around designing and implementing games or courses that usually focus on playing games to learn different specific subjects, from chemistry to history.

Other approaches, similar to the design and development one, but with a different target, talk about using games as support in teaching basic computer science concepts. One major example being the Reality and Programming Together program from the Rochester Institute of Technology, that uses game development in early undergraduate computer science education courses. The RAPT program uses games as an application area on top of a traditional CS curriculum in three initial courses that teach core topics in computer science, focusing in real programming and games as complex software (Bayliss and Strout 2006).

But few of these studies on the use of games as a learning tool try to really understand why using games works. More specifically, most of the studies on using games to teach CS subjects are motivated by the need to improve student interest. Increasing motivation is definitely a valid concern that has to be addressed as it is key to effective learning. But motivation needs to be sustained through feedback, reflection and active involvement in order for learning to take place (Garris et al. 2002). Digital games could have a much bigger role in learning than just as a motivational tool.

Recent research has tried to analyze why and how games can be helpful in different learning contexts from the learning sciences perspective. Also, studies have been published that track student performance through time trying to identify if their performance was improved by the usage of games in the curriculum (Bayliss 2007), and show promising results.

Mayo in (Mayo 2007) claims that video games can teach science and engineering better than regular lectures as they have the potential to address many systemic deficiencies of the traditional teaching methods. Mayo describes this potential as derived from five reasons:
I – Massive reach – games are already hugely popular and affect players throughout their lives;
II – Effective learning paradigms – games present support for different learning precepts from learning science;
III – Enhanced brain chemistry – research has indicated that dopamine release occurs during video game play, suggesting that it may be that video games are able to chemically “prepare” the brain for learning;
IV – Time on task – people spent a great deal of time playing digital games, so compelling video games that could also deliver educational content might raise time in learning tasks;
V – Learning outcomes data – initial studies comparing video game teaching effectiveness to the classic lecture show positive improvements, typically 30% or more.

Mayo concludes that games are better than a regular lecture for learning and shows data suggesting that designing games also increases student performance in science subjects. This finding is also supported by a survey (Hake 1998) that shows that courses with some degree of “interactive engagement” have better student performance, which arguably could also be said for CS and other engineering subjects.

As shown above, using digital games in education by itself is not a new idea, but most of the time the focus is on introductory or basic concepts. In this paper we propose a new approach: using games as support to motivate more advanced students by creating programs that play games. But not as the traditional AI approach of solving a game like chess, checkers, or go. Our approach uses popular available casual games as base for a framework where students write the game playing software completely detached from the game. Our goal here is to describe the developed framework, how to use it, and share our experience from the case study of its application in two undergrad courses at one institution and one graduate course at another institution.

In the next section, we present some related work, our proposed approach, and the chosen casual game for the case study. Then, a generic framework for building game players, some implementation details, and how to use it during a course are described. Finally, we present some results of our case study, some remarks, and comments on future work.

RELATED WORK AND PROPOSED APPROACH

As previously stated, using digital games in education by itself is not a new topic. Most of the existing work focuses on teaching game development, using games to teach a specific subject - primarily school level subjects, or teaching basic CS concept by using games as a motivation - ex: (Lemmon et al. 2007). Very few try to explore how digital games could support teaching of higher level computer science subjects.

One of the best known works in using games to teach CS is the RAPT program (Bayliss 2007, Bayliss and Stout 2006), and it already shows improvements in students who went through the program. Bayliss examines in detail the use of game development in early undergraduate Computer Science education. The RAPT experience shows that students do have a strong interest in using games and find the program materials subjectively more satisfying than regular ones. In its courses, student assignments can be either playing or coding games (Bayliss and Strout 06).

Moskal et al. describes yet another experience of using games in basic CS and provides some statistics collected during two years showing that students achieved better retention and attitude towards computer science. Students matching the "drop out" profile, but had basic courses using games to teach concepts in the curriculum, scored significantly higher than students who did not go through this kind of course (Moskal et al. 2004). One other major point of this work is the claim that game design helps stimulate students to develop human centered design skills and how to work in an iterative development process, both skills that are very important in their professional life.

A little different from these two works is the work by Kelleher and Pausch in using a storytelling version of Alice to raise interest and motivation in CS for high school students, especially female students. Kelleher and Paucsh (2007) also say that students often find their first courses boring or uninspiring and that typical assignments fail to engage many students. By using 3D movies and storytelling, they intended to attract and motivate students. Their storytelling approach increases the academic success and retention of at-risk college students and girls using Storytelling Alice were more motivated to program.

None of these approaches deal with using games in an upper-level course in the CS curriculum, the area that our approach focuses. To the best of the authors' knowledge, Youngblood (2007) is the only one currently addressing gamedev in an upper-level class. Youngblood uses what he calls Game Segments – full real-world-like code of a game with a clear missing element – in a higher level AI course and also defined principles for designing them. He applies games as homework assignments, using each to test specific AI techniques; while our approach uses a clear structure for the student to develop in; and leaves only the design of the solution up to the students, allowing them to choose and combine different techniques, or focus on optimizing one.

A computer game comprises a number of distinct modules like physics, graphics, user input, graphical interfaces, AI, networking, among others. When building a game, designer and programmer must consider all these parameters to define appropriate data structures, languages and so on. If one is interested in studying just the AI, for example, being able to focus on the AI issues would be more productive. Although the Game Segments approach minimizes these other influences, it still leaves different code bases for the students to learn, and allows the students to change the game part of the game segments. In our case, instead of full source games, or game segments, casual games are used as a base.

Our focus is on using game AI challenges to help students learn AI concepts, programming techniques, and possibly image segmentation and pattern matching. By using finished games we take advantage of their gameplay and avoid possible visual quality discrepancies (as when students with different artistic skills create a game). Also, there is no need to learn different code bases before getting each assignment done. Even though being able to create new visuals and sounds helps exercise creativity, this should not be the focus in an AI course. Lastly, using a finished game provides a common testbed for development and judgment of the assignment completion.

There are several challenges to be faced by the students when developing code for a game player. Some of them are: understanding the game logic; make the program recognize the game environments and the pieces/characters to be moved around; design heuristics to select plays; timing and interaction issues with the GUI; performance vs. play quality. But the benefits really help focus on the AI part. Also, our approach eases the process of creating competitions among the different implementations, what - our experience shows - is a great factor in motivating students.

BEJEWELED

The selected casual game for our study was Bejeweled by PopCap Games (2007), a very popular gem-swapping puzzler where one must swap adjacent gems to align sets of 3 or more gems of the same color in the vertical or horizontal axis. There are 7 different gem colors and shapes, as shown in Figure 1. Whenever a valid (winning) set is formed, such gems are removed, the upper gems are shifted downward to fill the created hole, and random gems are inserted in the upper row. To swap gems, one must click once over each gem or drag a gem with the mouse cursor, or click in a neighboring gem. The game runs on a web browser.

The more gems removed at a given time, the bigger is the score. Combos and cascades award extra points. The game has a gem meter bar that increases as gems are removed. When this meter if full, an increasing bonus is awarded. Also, as one moves up the levels, the winning set is worth more and more points. The game can be played both in simple and timed modes. In timed mode, the gem meter decreases continuously as time goes by, with a speed proportional to the current bonus level. The greater the current bonus, the greater the bar reduction.

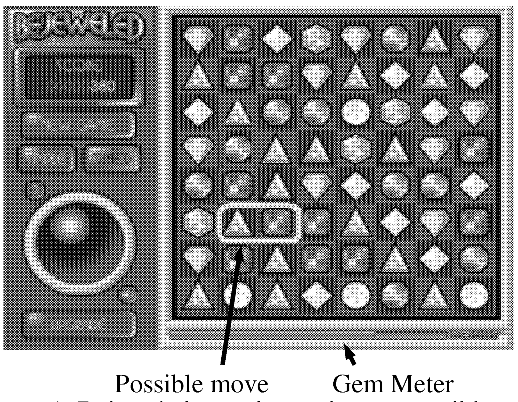


Figure 1: Bejeweled snapshot and some possible moves.

PROPOSED FRAMEWORK

Our approach to create a virtual player is similar to the way a human being plays the game. The player analyzes the screen image, selects a play, and uses the mouse to execute the play. With this approach, theoretically, virtual players can be built for any kind of games.

In order to accomplish these requirements, the game playing program must be capable of performing at least the following operations: I - Capture the OS (MS-Windows) screen - desktop; II - Analyze and segment the captured image; III - Locate the game position in the screen; IV - Locate individual game pieces; V - Define playing strategies and heuristics; VI - Move the mouse cursor programmatically; VII - Control action timing; and VIII - Send mouse events for clicking on the game pieces. These requirements can be translated into different modules and low-level implemented native components that communicate with the OS to get: screen colors; screen size; mouse position; screen contents; and press and release mouse buttons.

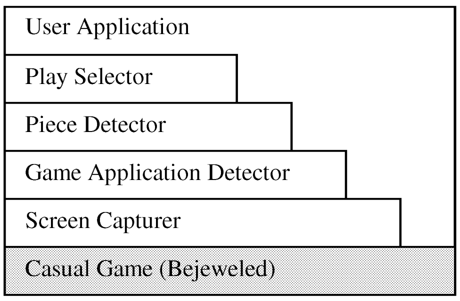


Figure 2: Framework Modules.

To reach the proposed goal, we built a hierarchy of modules as shown in Figure 2. In the bottom of the hierarchy there is the casual game (in this case, Bejeweled) and in the top is the user application (game player). In the middle are modules for processing and executing plays. In this architecture, the user application can communicate with all other modules. In fact, one can see all these layers as a single application.

Screen Capturer is a module responsible for capturing the desktop image (content). It may use native resources responsible for capturing the desktop being viewed. **Game Application Detector** is necessary because the game does not necessarily runs on full screen. **Piece detector** is the first module more specific to the Bejeweled game and detects a gem's type inside the game. Every gem may assume one position of a predetermined set of positions in a grid. When implementing the Piece Detector in other games, like Tetris, this step may become much harder. **Play Selector** is the main component and comprises the main AI algorithms. Students using this framework will mainly work on the Play Selector and possibly in the Piece Detector, depending on the assignment at hand.

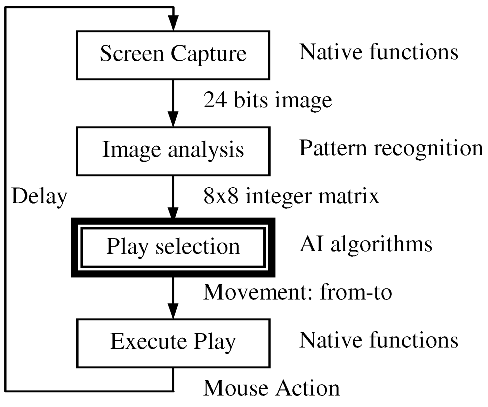


Figure 3: Player execution. Highlighted module is the AI

To verify the feasibility of our solution, we have implemented a virtual player that comprises the above mentioned modules. An executable demo version of a player and the API code can be downloaded from the web (Pozzer and Karlsson 2007). Instead of creating a video to demonstrate the player in action, we invite everyone to download the demo and try it. Figure 3 shows the execution loop of the game player framework. The role of the teacher (advisor) is to decide which modules would be available for students for building their players. At least the Play Selector module should be under responsibility of the students.

The Piece Detector can be implemented in different ways. It can be as simple as matching a pattern of colors (our approach) or by using pattern matching technique, image processing, or segmentation. Screen Capturer and Execute Play use functions from the operating system. These functions are wrapped in an API and library that are made available to the students. Each of the necessary steps can be implemented in C, C++, or Java. It is up to the students to decide which approach to use taking into account: performance, programming resources, and previous language knowledge.

The implemented framework can easily be modified in order to accommodate other similar games with fairly simple changes in the provided modules.

CASE STUDY RESULTS

This framework was used three times until now (2006 - 2007), in three different courses in both graduate and undergraduate computer science courses; once in an undergraduate course at UFSM and twice in a graduate course at UnicenP. The assignment students receive is to implement an intelligent Bejeweled player. Some aspects of the implementation are also questioned in a test during the course. Near the end of each course, a competition was held to evaluate the implemented players.

Depending on the level of difficulty, students may construct all the hierarchy or just some high level modules like the user application and play selection. How students are going to implement the Play Selector is up to them. They can use heuristics, search, state machines or even brute force, or any combination of them. Also, students have to handle all the timing issues in sending events, as well as possible interruption during game flow (like the bonus message).

Different criteria can be used to try to measure performance, at least for ranking purposes in the course competition. Some possible criteria are: I - Reaching an N point bonus in less time; II - Score is valid until the "*no more moves*" message is displayed for the first time; III - Greatest score within a given time limit. Games as Bejeweled are time based, so the faster the player, the bigger is the reward. Because of this, a fast algorithm can have a big advantage over a slower one. But shorter times do not necessarily mean best scores. So, one can tailor the ranking criteria to make the students try different approaches. In order to tip the balance toward more complex algorithms, we decided to adopt the bonus criteria as it is more complex and open to different AI techniques.

During the application of this project in all three courses, we could notice that students were excited and motivated by the approach. Also, a "competition spirit" was also always present because of the final ranking. During the player development, lots of e-mail were exchanged between the students about scores already reached and other groups were motivated to improve their players. Some players even managed to reach more than 1,000,000 points, and kept improving long after the final ranking.

CONCLUDING REMARKS AND FUTURE WORK

A lot has been written about using games for learning purposes, but most of the available research and literature is on coursework around designing and implementing games or courses that usually focus on playing games to learn different specific subjects, from chemistry to history, to math. When applied to computer science concepts, most approaches focus only on teaching basic concepts in the CS curriculum.

We propose a different approach; to use popular casual games as a base to teach more advanced CS courses. In our case study, we focused on AI. In order to accomplish this, we implemented a framework to allow the development of game playing software. This way the students are freed from unnecessary detail and can focus just in the AI itself.

During the case study it was possible to clearly notice the students' competition spirit, and that the students were having fun with the project. It was clear that it was a highly positive experience that stimulated their interest for the subject. We showed that, in conjunction with an otherwise already practical course, using a casual game as base for teaching AI techniques in a somewhat large project provides an immersive learning experience. As in Youngblood's approach, student performance and questions provided good insights and caused a great deal of information exchange.

As future work, we will start working on more elaborate game player projects. There are many possibilities. Tetris is a good initial choice as pieces can have different shapes and a more elaborate algorithm for detection is required. Moreover, play selection is quite more complex as pieces may be rotated and placed in different locations. We'll also try the same approach with a fight game, where image analysis and segmentation will be much more important.

REFERENCES

Bayliss, J. D. 2007. "The Effects of Games in CS1-CS3". *Journal of Game Development*, vol 2, Issue 2 (Feb 2007), 7-18.

Bayliss, J. D., and Strout, S. 2006. "Games as a Flavor of CS1". In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, Houston, USA, 500-504.

Garris, R., Ahlers, R. and Driskell, J. 2002. "Games, Motivation and Learning: A Research and Practice Model". *Simulation and Gaming*, 33: 441-467.

Hake, R. 1998. "Interactive-engagement vs. Traditional Methods: A 6,000-student Survey of Mechanics Test Data for Introductory Physics Courses". *American Journal of Physics*, 66 (1), 47-62.

Kelleher, C., and Pausch, R. 2007. "Using Storytelling to Motivate Programming". *Communications of the ACM*, July 2007, volume 50, number 7, ACM Press, 58-64.

Lemmon, C., Bidwell, N. J., Hooper, M., Gaskett, C., Holdsworth, J., and Musumeci, P. 2007. "Creativity in the Cane Fields: Motivating and Engaging IT Students Through Games". In *Proceedings of the Microsoft Academic Days on Game Development in Computer Science Education 2007*, 43-47.

Mayo, M. J. 2007. "Games for Science and Engineering Education". *Communications of the ACM*, July 2007, volume 50, number 7, ACM Press, 30-35.

Moskal, B., Lurie, D., and Cooper, S. 2004. "Evaluating the Effectiveness of a New Instructional Approach". In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, ACM press, New York, 75-79.

Popcap Games. 2007. "Bejeweled". [online] Available from: http://www.popcap.com/games/bejeweled [Accessed 14 August 2007]

Pozzer, C. T. and Karlsson, B. 2007. "GamePlayer API". [online] Available from: http://www.inf.ufsm.br/~pozzer/gameplayer/

Swedy, E., Delaet, M., Slaterry, M. C., and Kuffner, J. 2005. "Computer Game and CS Education: Why and How". In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 256-257.

Youngblood, G. M. 2007. "Engaging Students in Advanced Computer Science Education Using Game Segments". *Journal of Game Development*, vol 2, Issue 2 (Feb 2007), 33-46.

EGO: AN E-GAMES ORCHESTRATION PLATFORM

Davide Rossi
Elisa Turrini
Dipartimento di Scienze dell'Informazione
Università di Bologna
Mura A. Zamboni, 7 - 40127 Bologna
Italy
E-mail: {rossi,turrini}@cs.unibo.it

KEYWORDS
Serious gaming, e-learning, process modeling.

ABSTRACT

Simulation-based learning games proved their effectiveness as a support to regular or e-learning courses, especially for those related to economics. Authoring and delivering multi-user simulation-based games, however, requires a lot of effort, even when dealing with a limited number of users. In this paper we present EGO, a platform for the authoring and the Web delivery of multiuser e-learning games that has been used for economic simulations. In EGO two games model cooperate: the interaction model and the environment model; the former is described by using a process modeling language, the latter by using domain specific tools, in the case of economic simulations we used a spreadsheet-based model.

Introduction

The idea of using simulation games for learning is not new (Gredler 2003). But, of the many existing proposals to support game development in this context, most are based on elementary facilities, typically on scriptable environments that clearly show their limitations when dealing with multi-user gaming. Multi-user games add a new dimension of concern: interaction management (notice that we do not refer to massive on-line multi user games in this context, a games typology that introduces further concerns). Multi-user games can be turn-based games or can be concurrent games with synchronization points or a mixing of these two models. Lots of different interaction patterns can take place during the gameplay and the management of players coordination by using a scripting language is a very complex and error-prone task, even when specific coordination tools (libraries, language extensions) are provided. The reason for this inherent complexity is that, in these games, two distinct models have to cooperate: the game environment model (that, in the case of an economics game, deals with economic indexes and their relationships) and the game interaction model. By providing a method that allows to separately address them, we can obtain better separation of concerns, reducing code tangling and promoting its reuse. A clear advantage of this approach can be appreciated when focusing on the game development team: the developer in charge of the economic model (an advanced model that requires specific advanced knowledge of economics) has probably little confidence with tools for players coordination; while a member of the team that has a good knowledge of interaction modeling has probably not the skill required for environment modeling. In this paper we propose a dual model approach to simulation games for e-learning. Specifically, the interaction modeling makes use of a process modeling language and the environment model can use specific domain-dependent tools. In the case of economic simulation games like the one we focus on in this article, a spreadsheet-based model is assumed. We also present a Web-delivery platform that enacts both models in order to govern gameplay. This platform is based on rich internet application techniques for the interface management and on a process enactment engine for interaction management.

Interaction modeling

Interactions in EGO are driven by a process specification. We use for this task EPML (Rossi and Turrini 2007), a graphical, executable, process modeling language. EPML has a high expressive power associated to a rather simple and intuitive notation. The main advantage of EPML in the context of this work, however, is that it has an accompaining enactment engine that is easily composable inside complex software architectures with moderate effort. The enactment engine, in fact, has to interact on the one side with the Web application for the definition of the interfaces that have to be dispatched to the user and, on the other side, with the spreadsheet to govern the interactions between the players and the game environment model. The interactions of the players and the gaming system in EGO progress by using a sequence of interfaces. The interaction model defines this sequence in function of the stage of the game and of the interaction pattern among users at the current stage

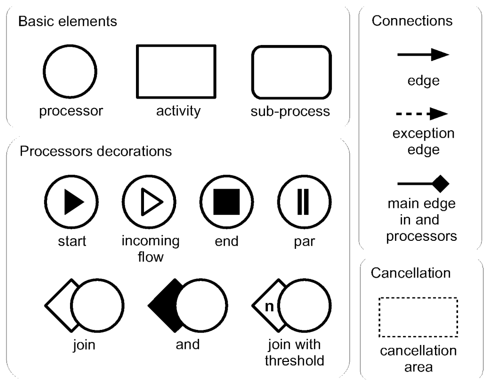


Figure 1: EPML elements

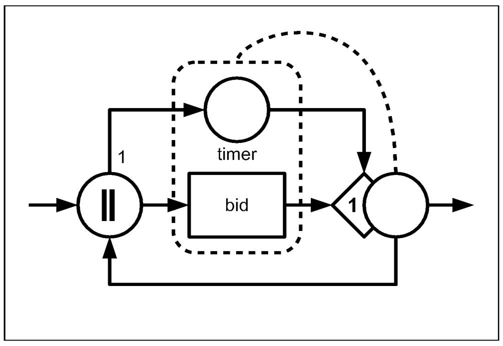


Figure 2: An auction in EPML

(turn-based, concurrent playing, mixed models). Notice that in the same game several interaction patterns are possible so the language has to able to model all of them with ease.

A detailed description of EPML is outside the scope of this paper, the interested reader can refer to (Rossi and Turrini 2007), in the next few lines we will give only a very brief survey of it. EPML is a graphic language, a process model (also called process specification) is obtained by composing the elements depicted in figure 1 in a free direct graph structure.

Specific elements can be used to control constructs like choice, concurrent execution, synchronization, etc. In a EPML specification control-flow elements (the processors, depicted as circles) can activate tasks (or activities, depicted by rectangles). Activities are computational elements that, in the case of gameplay coordination, correspond to the actions taken by the players. Given the player interaction-interface association we discussed before, activating a task corresponds to activating an interface for a specified user. This mechanism is very similar to what happens in most workflow management systems: actors are associated to a set of work items. At any time actors can take in charge work items and later notify the system when their work is over (potentially communicating the outcome of the work). In EGO the interfaces are the work items: a player completes a work item by interacting with an interface. The main difference with respect to workflow systems is that in EGO players can have only one active interface at any given moment whereas in a workflow system an actor can have several (or none) associated work items. The run time system is designed so that each time a new activity (interface) is assigned to a player the previously assigned one (if any) automatically completes. If players have no assigned interfaces a default "wait" interface is assigned to them. This insures that the correct semantics (exactly one interface per player at any time) is guaranteed. Consider now the example in figure 2: this is the modeling of a portion of a game in which an auction takes place.

We use this simple example to give a flavor of what modeling game processes with EPML looks like. The initial par processor creates as much concurrent flows as the number of registered players plus one additional flow that activates the `timer` processor. Each of the former flows activate one `bid` activity (i.e. the corresponding player is presented with a bidding interface). The ending join processor is activated as soon as either one player place its bid or the timer is expired. By analyzing its incoming flow the join processor decides to return to the par processor (in the case a player submitted a bid) or to continue the game with the following steps (when the timer expires). The dashed area connected to the join processor is used to cancel the timer (if still active) and the other player interfaces when the processor is activated. Notice that the cancellation is not really required since the join processor ignores all the flows that come after the first one that activates it; since the activation of a new interface for a given player automatically causes the termination of the existing one, the correct semantics of the system is preserved. It is however a good idea to explicitly use cancellations to keep the model as generic as possible since the specific behavior we just described applies only to the EGO platform.

EGO architecture

The run-time architecture of EGO is based on Java Enterprise Edition. The game delivery system is a Web application hosted in an JEE-compliant application server (only the Web tier is used, so a stand-alone servlet container like Tomacat or Jetty can be used). The behavior of the application is driven by the process enactment engine that, in its turn, interfaces with the economic model described by a spreadsheet. The system has been developed so that when users connect to the game URL, after an identification step, the interface for the current game step is visualized (since the game can be a long-running economic simulation, game time can be several days, so it is normal for users to log in several times to com-

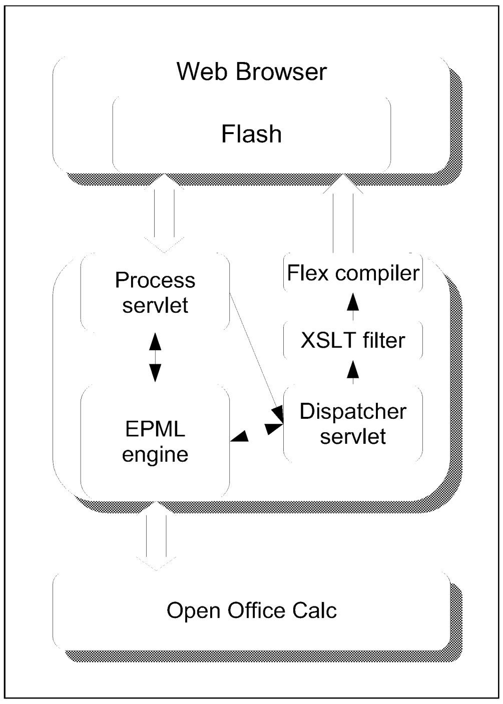
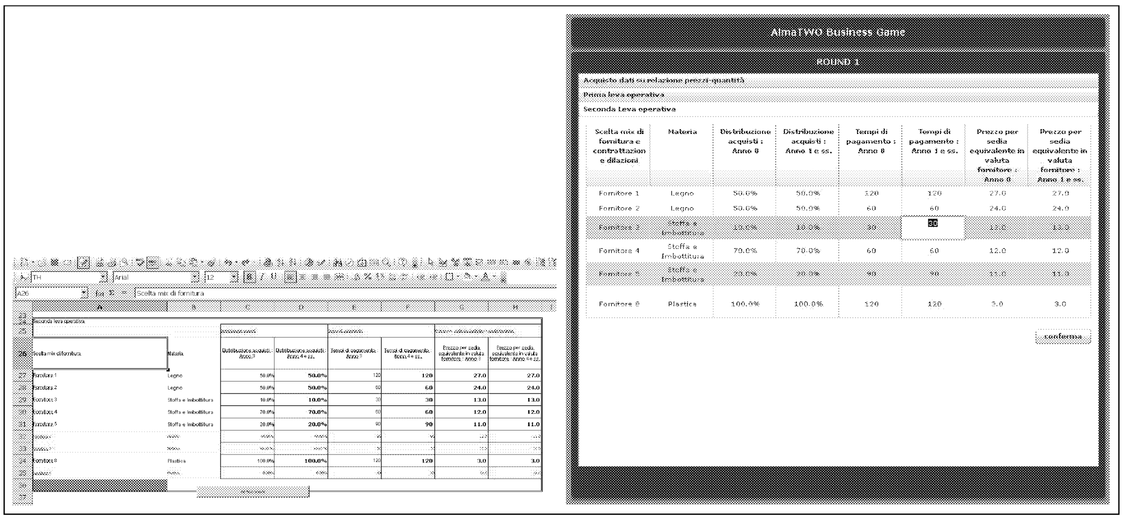


Figure 3: The overall delivery architecture of EGO - only major component are shown

plete a single game). The interfaces use rich internet applications technologies to allow a rich gameplay and to asynchronously check with the back-end if a new interface has to be loaded (the current interface of a player can change independently from its interactions with it, for example when that player is taking too much time to complete its interaction or when its interface has to change from a wait-you-turn one to a playing one because it is now the player's turn). The overall delivery architecture of the system is depicted in figure 3.
The sequence of invocations associated to a user request is as follows. When the user completes its interaction with an interface the interface sends an HTTP post request (containing the data relative to the user input formatted as XML) that is handled by a `Process` servlet that acts as a controller. The `Process` servlet retrieves the input data from the request and the user information from the session data and calls `Play`. `Play` interfaces with the engine: it creates an `end activity` event (containing the user and the input data) and notifies it to the engine. The engine enacts the process, typically activating a processor, i.e. a process logic component that interacts with the spreadsheet by using the UNO/Java library. The behavior of the processor is, in general, to put users data into specific cells of the spreadsheet. The control returns to `Play` that delegate the reply to another servlet, `Dispatcher`. `Dispatcher` queries the engine to know the activity (if any) that is now associated to the player. The activity descriptor holds references to a portion of the spreadsheet that contains relevant data that should be used to populate the new interface. `Dispatcher` creates a new XML document containing the interface description and the spreadsheet data and returns it as an HTTP reply. This replay is then intercepted by a filter. The task of the filter is to build a rich interface by using the information contained in the XML document. Filters can use different technologies for creating the interface. By using XSL transformations a plethora of different XML-based technologies can be used. We built interfaces using HTML+XForms, MXML (the Adobe Flex source format) and LZX (the OpenLaszlo source format). Flex and OpenLaszlo are used to create Flash-based interfaces; OpenLaszlo can, from the same source, create both Flash-based and Dynamic HTML-based interfaces. In the case of the e-game we focus to in this paper (the economic simulation) we used a Flash-based interface generated with Adobe Flex. The filter then transforms the incoming XML document in a MXML one which is fed to the run-time Flex compiler that generates the resulting Flash movie.
EGO also includes Web-based management tools for creating new games, adding players, managing existing games and so on. The details about them, however, are outside the scope of this paper.

Authoring EGO games

The authoring process of EGO-based games is composed by three steps: interaction modeling (using EPML), environment modeling (the economic model created using a spreadsheet) and the definition of the relationships among gaming interfaces and elements in the environment model. In order to ease this last step we used a metadata approach for the environment model. The basic idea is that each gaming interface can be associated to a specific rectangular zone of the spreadsheet. Inside this rectangle metadata (in form of styling information associated to the cells) can be used to define partially or *in toto* the information that have to appear in the gaming interfaces that are related to the information contained in the spreadsheet. Consider the screenshot on the left of figure 4: the rectangular area delimited by a double-line border contains the information (cell values) and the meta-information (cell styles) that have to appear in the gaming interface. Specifically, elements associated to the `output` style are displayed as they appear in the spreadsheet in the corresponding Web interface; elements associated to the `input` style (that appear with a darker - yellow - background in the spreadsheet) appear as input field in the Web interface, and so on.
The screenshot on the right of the same figure depicts the Flash interface automatically generated, using Adobe Flex, from the specification given in the spreadsheet.
When the game includes interfaces that are very different in graphical appearance and in purpose, different XSL transformation sheet can be used; the interface de-



AlmaTWO Business Game

ROUND 1

Acquisto dati su relazione prezzi-quantità

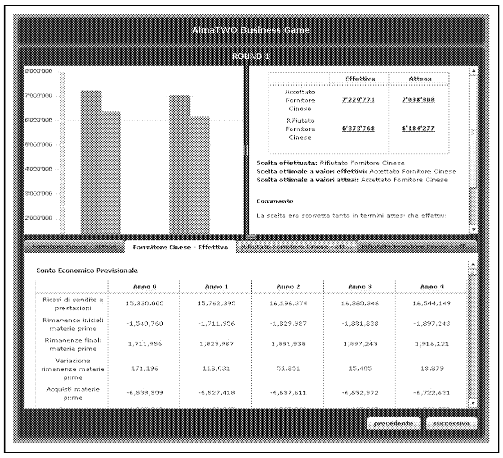
Prima leva operativa

Seconda Leva operativa

Scelta mix di fornitura e contrattazione dilazioni	Materia	Distribuzione acquisti : Anno 0	Distribuzione acquisti : Anno 1 e ss.	Tempi di pagamento : Anno 0	Tempi di pagamento : Anno 1 e ss.	Prezzo per sedia equivalente in valuta fornitore : Anno 0	Prezzo per sedia equivalente in valuta fornitore : Anno 1 e ss.
Fornitore 1	Legno	50.0%	50.0%	120	120	27.0	27.0
Fornitore 2	Legno	50.0%	50.0%	60	60	24.0	24.0
Fornitore 3	Stoffa e Imbottitura	10.0%	10.0%	30	30	13.0	13.0
Fornitore 4	Stoffa e Imbottitura	70.0%	70.0%	60	60	12.0	12.0
Fornitore 5	Stoffa e Imbottitura	20.0%	20.0%	90	90	11.0	11.0
Fornitore 8	Plastica	100.0%	100.0%	120	120	3.0	3.0

conferma

Figure 4: A portion of the spreadsheet and its corresponding interface



AlmaTWO Business Game

ROUND 1

	Effettiva	Attesa
Accettato Fornitore Cinese	7'229'771	7'038'308
Rifiutato Fornitore Cinese	6'373'768	6'184'277

Scelta effettuata: Rifiutato Fornitore Cinese
Scelta ottimale a valori effettivi: Accettato Fornitore Cinese
Scelta ottimale a valori attesi: Accettato Fornitore Cinese

Commento

La scelta era scorretta tanto in termini attesi che effettivi

Fornitore Cinese - Effettivo

Conto Economico Previsionale

	Anno 0	Anno 1	Anno 2	Anno 3	Anno 4
Ricavi di vendite e prestazioni	15,330,000	15,762,395	16,196,374	16,360,346	16,544,149
Rimanenze iniziali materie prime	-1,540,760	-1,711,956	-1,829,987	-1,881,838	-1,897,243
Rimanenze finali materie prime	1,711,956	1,829,987	1,881,838	1,897,243	1,916,121
Variazione rimanenze materie prime	171,196	118,031	51,851	15,405	18,879
Acquisti materie prime	-6,538,309	-6,527,418	-6,637,611	-6,652,972	-6,722,631

precedente successivo

Figure 5: An interface generated with a specific XST sheet

picted in figure 5, for example, is generated by a different XSL sheet with respect to the one used for the previous example.

By using the aforementioned technique the environment model is overloaded with concerns related to the structure of the game, in which the relations between this model and the gaming interfaces appear in an explicit way. This is reasonable, it is still in fact possible to develop the economic model with no relation to the game structure and later add some sheets to the spreadsheet (as it is actually the case with the game we took the screenshot from) that address the correspondence between elements in the model (by simply referencing them from the new sheets) and gaming interfaces. This let the experts in environment modeling concentrate on the inner workings of their models allowing other members of the team, in a later stage, to complement their models with the relationships between them and gaming interfaces. Our experience shows, however, that the first approach enormously eases game development and cuts its times considerably. We argue that, even when using different tools for environment modeling, similar techniques should used if possible.

Related works

Before describing the works related to our proposal, it is worth noticing that e-games have impact on different research areas that span from artificial intelligence to e-learning. It should also be noted that the literature provides very few proposals that can directly be compared to EGO, in this section we try to point out the systems that share common traits with it and could enjoy reciprocal influence.

Among the plethora of papers that investigate e-games from different perspectives, the work most similar to our is the Extensible Graphical Game Generator (EGGG) (Orwant 2000). EGGG allows users to create almost any kind of two-dimensional game with a minimum of programming effort. Designers can describe a game using an high-level language. The game description is in turn rendered by the EGGG engine that employs a set of heuristics that embody the similarities between games and then generates the program (that is the computer game ready to be played) by composing reusable software components. Both EGGG and EGO have the goal of speed up the implementation of new games; the main difference between the two is that EGO does not use heuristics to interpret the game description and then requires the game description to be precise and non-ambiguous. The drawback is that designers have to be fluent in EPML (the expertise required depends on the game complexity); the advantage is that the range of

games that can be implemented is wider and not limited to two-dimensional games only.
Another interesting e-game authoring tool is presented in (Moreno-Ger et al. 2007). Authors state that the development of good *educative* games requires the active participation of field experts (such as teachers) that often do not have a deep formation in computer technologies. The collaboration between those experts and game developers is realized by means of a *document driven* approach: the game can be described using a Domain Specific Language (DSL) in order to facilitate the expert's task. Those documents are then fed to an engine that generates and executes the game. As DSL has to be kept reasonably simple, it could hardly permit the development of generic games: for this reason the genre and functionality of the games have been restricted, and focus on one single, well-defined genre (Adventure Games).
There exist also other game building tools that can be used by people with do not have any programming experience (see, for example (Cook 2005)), but, they have been designed to create video games only. On the contrary EGO does not impose strict limitations on the genre of the game (albeit it is best suited to non-realtime games), but can be used to implement many kind of game.
To conclude this section, we cannot ignore the growing interest in e-games as tools to support (e-)learning. Nonetheless, the vast majority of studies, such as for example (Magnussen et al. 2003, Arnseth 2006, Alsagoff 2005), focus on psychology and pedagogical aspects of e-games, on the maximization of their learning potential, and in case of multi-players game also on the interaction dynamics among players.

Conclusions

The EGO platform we presented in this paper offers several advantages for both the authoring and the delivery of e-learning simulation games. Games in EGO are based on two main models: the interaction model, described using the EPML process modeling language, and the game environment model that, in the case of economic simulations, can be described using a spreadsheet. In this case the authoring process can take advantage by the use of styles as meta-data to describe the relationships between sheet elements and interface elements. As a delivery platform EGO can easily accommodate different Rich Internet Applications techniques since all the communications between the back-end and the front-end take place by using XML. Using asynchronous AJAX-like techniques the interfaces can be kept consistent with the game status as stored in the server. These techniques, however, are based on HTTP polling so they are not feasible for games in which a strictly coherent view of the game status is required for all players (but this is not often a requirement for e-learning simulations).
In our experience EGO revealed itself as a valuable tool for authoring and delivering simulation-based learning games. The development of the economics-based simulation game discussed in the paper revealed that the promise of a clear separation between the different models participating in the gameplay can be fulfilled, allowing a team of experts in different fields (interaction modeling, user interface design and business economics) to focus on their respective abilities and yet to cooperate beneficially.

REFERENCES

Alsagoff Z.A., 2005. *The Challenges & Potential of Educational Gaming in Higher Education.* In *Proc. of the Second International Conference on eLearning for Knowledge-Based Society.*

Arnseth H.C., 2006. *Learning to Play or Playing to Learn - A Critical Account of the Models of Communication Informing Educational Research on Computer Gameplay. International journal of computer game research*, 6, no. 1.

Cook B., 2005. *Game Building Tools.* http://www.apple.com/games/articles/2005/08/gamebuildingtools/. Accessed November 2007.

Gredler M.E., 2003. *Games and simulations and their relationships to learning.* In D. Jonassen (Ed.), *Handbook of research for educational communications and technology*, Lawrence Erlbaum Associates, Mahwah, NJ, USA. 571–581.

Magnussen R.; Misfeldt M.; and Buch T., 2003. *Participatory design and opposing interests in development of educational computer games.* In *Proc. of the Digital Games Research Conference.*

Moreno-Ger P.; Sierra J.; Martínez-Ortiz I.; and Fernández-Manjón B., 2007. *A documental approach to adventure game development. Science of Computer Programming*, 67, no. 1, 3–31.

Orwant J., 2000. *EGGG: automated programming for game generation. IBM System Journal*, 39, no. 3,4, 782–794.

Rossi D. and Turrini E., 2007. *EPML: Executable Process Modeling Language.* Tech. Rep. UBLCS-2007-22, Department of Computer Science, University of Bologna.

SERIOUS GAMING

GAMING TECHNOLOGY IN CULTURAL HERITAGE SYSTEMS

Tim Horz Albert Pritzkau Christof Rezk-Salama Severin S. Todt Andreas Kolb

Computer Graphics Group
University of Siegen
Hoelderlinstr. 3
57076 Siegen, Germany

{tim.horz, albert.pritzkau}@student.uni-siegen.de
{rezk, todt, kolb}@fb12.uni-siegen.de

KEYWORDS
Serious Gaming, Methodology, Rendering

ABSTRACT

This paper describes the design and implementation of an interactive walk-through of a reconstructed German stronghold, the Dillenburg. The application is currently in use at the local museum. Applying technologies and algorithms primarily used for the development of realistic 3D computer games, we present a system that can be categorized as a *Serious Gaming* environment, a term that has recently come into existence. More specifically, it can be called a *Cultural Heritage Game*. With this paper we want to give a programmer's view on the topic of interactive cultural heritage systems.

INTRODUCTION

In recent years the markets for high-quality 3D computer games and commodity graphics hardware have grown immensely. Every new generation of graphics cards allows for the next generation of 3D games, which in turn demands the following generation of graphics hardware. During this development, the amount of innovative techniques and algorithms needed for a realistic real-time rendering of virtual environments keeps growing.
The technology and specific know-how gained from years of computer game development, not limited to 3D games, can be used for creating applications *outside* a gaming environment. Recently, this notion has been labeled as *Serious Gaming* (Blackman, 2005; Raybourn and Bos, 2005). As can be seen below, numerous projects show the major interest in cultural heritage applications, all of them using game technology in some way or another. Those and future interdisciplinary projects of that kind can be called *Cultural Heritage Games.*
Our *Cultural Heritage Game* presents an information and learning experience that helps to revive a part of history. As a cultural heritage undertaking funded by a non-profit organization and implemented by undergraduate students, our project stands outside the usual scope of business models and profit interests.
Throughout this paper we discuss aspects concerning real-time graphics programming applied to the 3D model extracted from a previous cultural heritage reconstruction project.

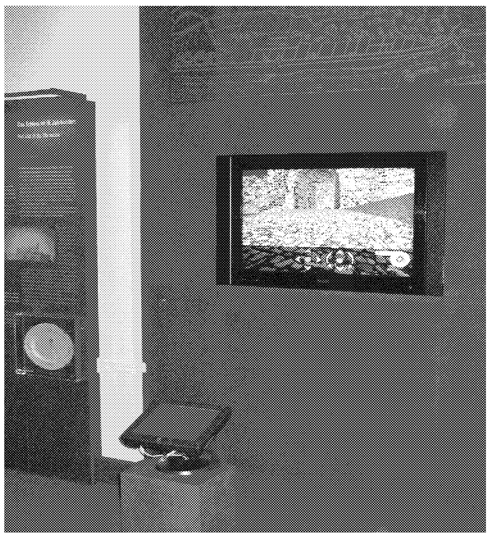


Figure 1: The Installation at the Museum

RELATED WORK

Generally, our work can be categorized as a computer aided cultural heritage project. Numerous comparable projects exist throughout Europe, some of which inspired our design decisions in different ways. Most notably, the virtual walk installation at *Trento* (Conti et al. 2006), the guided tour at *Sagalassos* (Pollefeys et al. 2001) and the (web-based) visit to the *Piazza dei Miracoli* (Carrozzino et al. 2005) gave us insight into cultural heritage projects. Those projects provide some good concepts and ideas that inspired some of our solutions. Content related aspects of the projects were

not adapted, since our work is focused on the application of game technology to existing 3D cultural heritage content.
The starting point to our work was, in fact, a previous project, aiming at the virtual reconstruction of the long-gone German stronghold Dillenburg (Todt et al. 2007). The project yielded a detailed 3D model used to generate an off-line rendered documentary movie of the stronghold that was totally destroyed in 1760. The movie and some rendered flights around individual buildings can be viewed by visitors of the *Museum Wilhelmsturm*, located at the archaeological site of the Dillenburg stronghold. The multimedia installation is integrated seamlessly into the exhibition. Visitors can browse a DVD containing the available renderings using a touchscreen interface (see Figure 1). The ambitious project was inspired and funded by the local cultural heritage association (the *Museumsverein Dillenburg*). From this project, we use the reconstructed model to implement an interactive walk-through. For human-computer interaction we take advantage of the already present touchscreen interface and use it as the input device for the entire simulation.

THE VIRTUAL MUSEUM

When creating a virtual museum application, most design decisions are determined by the setting, audience and objective of the corresponding exhibition.
In general, the heterogeneous audience of a museum, ranging from school classes to interested senior citizens, implies many options the system has to offer. For example, children might get uninterested if the simulation does not provide the visual quality that they are used to from modern computer games, while seniors might reject the notion of interactivity and prefer watching the original narrated movie rather than exploring the stronghold themselves. Furthermore, a good and easy means to navigate through the virtual environment is necessary, as well as an information system to present the most important and interesting facts about the site. Some of these aspects can be addressed by finding similar problems in computer game development, while others need a more tailored solution.

IMPLEMENTATION

Within the scope of this kind of project, it is not feasible to implement a custom game-engine from scratch. Since licensing a commercial game-engine was also not an option, we decided to combine various free and/or open-source software and adapted it to our needs. For the most important piece, the graphics engine, we chose the *Ogre*-3D-engine, after analyzing different open-source competitors. The *Ogre* engine convinced us with its ease-of-use, its extensibility and, equally important, the very active community that contributes to a wiki-portal and a Q&A-forum. Through an extension called *OgreNewt* we have simple access to the functionality of the free *Newton* physics engine. Similarly, sound support is added via *FMOD*, and keyboard and mouse input is handled by the *Open Input System (OIS)*. Direct access to the touchscreen is provided by the manufacturer *Elo*. As an embedded HTML-rendering component we use TerraInformatica's *HTMLayout*. Another plugin called *WMvideo-Plugin*, created and supported by the *Ogre* community, enables the system to show movies on arbitrary surfaces. Figure 3 sketches the collaboration of all these APIs.

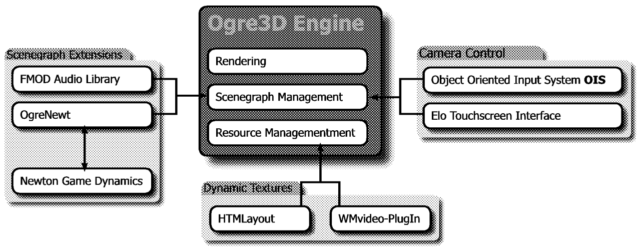


Figure 3: System Overview

Rendering

Nowadays, products that use real-time three-dimensional rendering, most notably computer games, have to meet a simple but challenging criterion: the consumers have to feel the rendering looks *good*, i.e. convincing and realistic. Being a subjective factor as it may, the perception of quality is the driving force in computer graphics technology, since visual quality is a selling point for each generation of graphics hardware. Reversely, consumers get used to better and better quality. If a product does not meet the visual standards the users have grown accustomed to, it may be disregarded completely. This notion can be especially observed with an important audience of cultural heritage sites: children and young adults. Any approach to a virtual museum has to meet this quality challenge. Therefore, we chose to apply a state-of-the art rendering technique to ensure the good looks of our simulation. When designing the system we also took advantage of high-end computer graphics hardware, an *nVidia GeForce 8800 GTX* with 756 MB of video memory.
The preceding reconstruction project mentioned above yielded a complex model of the Dillenburg stronghold, completely textured with decal maps, normal maps and gloss maps. An 8-minute flight around and inside the model was rendered off-line, using various global illumination and compositing techniques. With our interactive version we try to offer an equally impressive rendering of the same model in real-time. Due to the complexity of the Dillenburg model we use an environment map

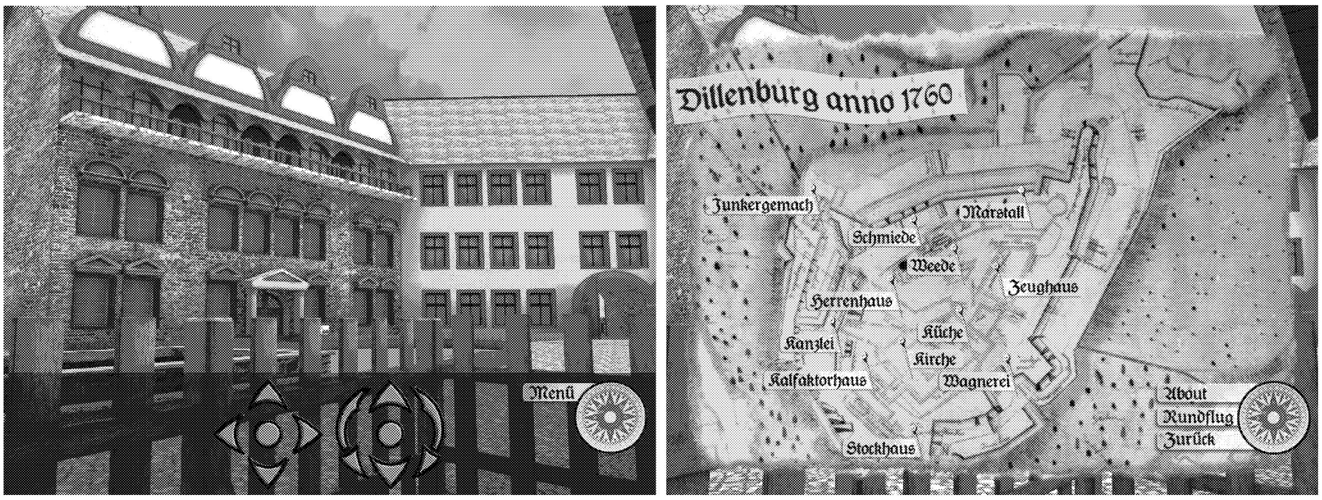


Figure 2: left: the Manor-house inside the Stronghold, right: Overview Map with Teleportation Points

(representing the sky) to light the entire scene, rather than using one or various point-light sources. However, traditional (even HDR) environment maps cannot be used for a complete and realistic real-time lighting of an object, as they only represent the light information for perfectly reflective surfaces. *Prefiltered environment maps* provide a powerful lighting and rendering approach to compete the challenge (Kautz and McCool 2000). This technique approximates glossy reflection with a set of cube maps generated from the original environment maps. Using ray casting techniques, the diffuse part of the lighting is precalculated and stored in an irradiance map. Likewise, a set of maps corresponding to different predefined specular lobes are generated. In the application, the information is combined on a per pixel basis using programmable graphics hardware as follows:

$$I = K_d \cdot I_{irr} + K_s \cdot ((1-g) \cdot I_{specLow} + g \cdot I_{specHigh}) \quad (1)$$

In Equation 1 I is the final lighting for the pixel. K_d simply denotes the color value determined by the decal texture. I_{irr} corresponds to the diffuse term in the Phong illumination model and is determined by a texture lookup inside the irradiance environment map. The factor K_s represents the shininess of the material, passed to the graphics hardware as a uniform parameter. The specular term is approximated by the linear interpolation of two predefined specular lobes, denoted as $I_{specLow}$ and $I_{specHigh}$. After experimenting with some values, we chose $I_{specLow} = 20$ and $I_{specHigh} = \infty$, as they yield the best visual results. The interpolation value g is looked up inside the gloss-texture and is equivalent to the specular exponent of the Phong model. Note that to avoid a blueish touch on non-reflective surfaces (caused by the blue sky), we desaturate the irradiance value beforehand.

Realistic shadowing is achieved by a combination of two types of precalculated (baked) shadow-maps. On the one hand traditional baked shadows are used. On the other hand we generated ambient occlusion maps to improve realism and plasticity of the objects (Landis, 2002; Zhukov et al., 1998). As the original model had been created with *Autodesk*$^{®}$ *Maya*$^{®}$, we used the baking technique provided by this modeling software which allows textures containing lighting information to be created fast and easy. After the rendering process both the ambient occlusion and the shadow maps were combined into one texture. To ensure a high quality we decided to use one grayscale 1024x1024 pixel texture per building or wing of a building. With a total of 130 textures, the lightmaps for the entire reconstruction consume 130 MB of video memory. The shadowing information is combined with Equation 1 as follows:

$$I_{shadowed} = s \cdot I \quad (2)$$

Here, the factor s denotes a lookup inside the combined shadow and ambient occlusion map. Figure 4 shows the different textures used for rendering on the most prominent building of the site, the manor-house. Another screenhsot showing the same building inside the simulation can be seen in Figure 2, left.

Interaction

The creators of the *Sagalassos* project argue that offering a completely free walk through a virtual museum is not desirable, as users might not get to see all the important buildings or even get bored. Therefore, they only offer virtual guided tours to their reconstruction. Our system however allows the visitor to walk freely through the environment, while we address the inherent problems as well.

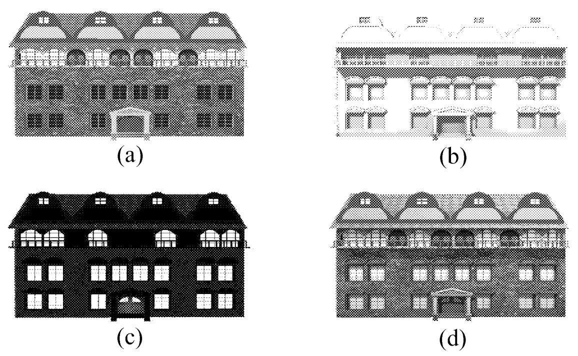


Figure 4: (a) Decal Map, (b), Combined Ambient Occlusion and Shadow Map, (c) Reflectivity (Gloss) Map, (d) Combined Result Illuminated by Prefiltered Environment Maps

An examination considering different input tools (e.g. keyboard and mouse, joystick, or a gamepad) led us to the decision to rely on the touch-screen monitor which was already part of the configuration. In contrast to other options as used in computer games it associates the visual experience and the navigational task in a manner best suited especially for the untrained user.
As for navigating through the virtual environment, we considered several strategies to account for visitors with gaming experience and unexperienced users alike. Finally, we devised two ways of interaction. Inspired by the keyboard-and-mouse input known from games, the camera can be moved by clicking on arrow symbols that control lateral and longitudinal motion as well as yaw and pitch, that way allowing complete freedom. This method of interaction is clearly targeted at more experienced users. Navigation is only limited by collision geometry, which is a sparse 3D model used for the collision detection by *OgreNewt* and the *Newton* physics engine.
The other method is based on a list of predefined viewpoints stored for each building. When the user points at a certain location on the screen, a camera-to-world ray and its intersection with the geometry is computed. If it hits a building, the system checks the associated list of predefined viewpoints and navigates to the one closest to the current position. Should the ray hit the ground, the camera simply moves to the indicated position. This approach allows the user to directly specify the building they want to examine, without getting distracted by the task of getting there on their own.

Supplemental Material

In any real museum, visitors need additional information, usually in the form of information plaques. A virtual environment has to offer some kind of information system as well.

The creators of the *DentroTrento* project describe a hotspot system that marks areas of interest inside their simulation. Likewise, our system allows content creators to place information panels at arbitrary world positions, but instead of only rendering some sort of hotspot icon the information is brought across directly inside the simulation: the information panels display freely configurable HTML content. That way users can easily obtain further information on the area they are looking at without changing into a different view (e.g. a menu).
This is achieved by an HTML rendering extension to the *Ogre* 3D engine we developed, using *TerraInformatica*'s free *HTMLayout* engine. The extension interprets HTML content and renders the result to a texture, which is then applied to the face of an information display inside the scene. The content can be handled in the same way known from any other browser, so when the visitor points to a hyperlink by touching the screen, the system opens the referred page and shows it on the information display. We also implemented basic support for augmenting the HTML content by video files. Note that those HTML-signposts have only one associated predefined viewpoint, which represents the optimal camera position needed to read and interact with the HTML content. When a user points to such a signpost, the mechanism described above will move the camera to that optimal position.

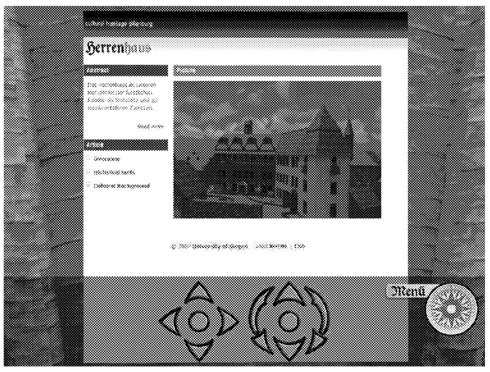


Figure 5: An HTML-signpost

RESULTS AND CONCLUSION

It is important to find new ways of applying the specific know-how from computer games to different applications and to provide highest rendering quality and state-of-the-art interaction techniques to the field of *Serious Gaming*. Using a variety of 3D game related algorithms, techniques and third party software we developed a system that resembles basic structures of a computer game.
We solved the problems inherent in cultural heritage simulations by applying concepts that exist in countless computer games, e.g. the overview map and the arrow-

based navigation. Innovative techniques and ideas have been implemented within the project, most notably the HTML-signposts and the viewpoint-based interaction system.

However, contrary to a computer game development project, our work required less time and fewer resources than a current state-of-the-art computer game, since only a small part of a game's functionality had to be implemented. For example, the system does not offer "opponents", so no artificial intelligence is necessary. Scalability was also not an issue, since the application is used on only one target machine, which was a high-end desktop PC with the most powerful graphics hardware available at the time. We could harness all the power available without ever having to implement options for toggling to a low-detail version of the model or a stripped-down rendering process. In that regard, our project is more related to game console development rather than to a PC game. Furthermore, the lack of typical tasks like level, character, and sound design sets our project somewhat apart from a "normal" 3D game.

FUTURE WORK

Our system can be adapted and improved to facilitate other reconstructed cultural heritage sites. With *Ogre*'s many import/export tools it is possible to convert data from a wide range of modeling software, while our modularized design facilitates the adaptation to other specific needs.

However, even with only the current Dillenburg model in mind the application in its current state holds still room for improvement. It would especially benefit from some "eye-candy" such as particle and animation systems for effects like torches, fountains or movable doors. The *Ogre* engine supports all this in a very straightforward manner, so after incorporating those features into the system it would all become a question of content creation. This next step would also benefit from the use of more recent developments, most notably geometry shaders.

Further down the road the system could be extended to support predefined animated guided tours (comparable to the "virtual guide" used by the *Sagalassos* project), selectable on the overview map. Furthermore, the HTML-signposts have a lot of potential, since the (now static) HTML-pages could just as well be generated by some kind of CMS, maintained by the museum staff and scientists. Added to that, combining the sound-system with the signposts would make it possible to have a narrator read the information back to the user, thus further enhancing the experience for the audience.

Another useful addition could be internationalization support, allowing translated versions of both written and audible text sources inside the virtual museum, just like usually offered by many of its real-world counterparts.

Acknowledgements

Our thanks go to the *Museumsverein Dillenburg* which funded parts of our project.
Special thanks to Matthias Hoffmann, Jeong Il-Sin, Jan Kunze, Michael Schantin, Jens Schlemper, Sascha Schlude, and Tim Wenclawiak,the members of the preceding project which yielded the final Dillenburg model. Their results were the very basis to our work.

REFERENCES

Blackman S., 2005. *Serious games...and less! SIGGRAPH Comput Graph*, 39, no. 1, 12–16.

Carrozzino M.; Brogi A.; Tecchia F.; and Bergamasco M., 2005. *The 3D interactive visit to Piazza dei Miracoli, Italy.* In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology.* ACM Press, New York, NY, USA, 192–195.

Conti G.; Piffer S.; Girardi G.; de Amicis R.; and Ucelli G., 2006. *DentroTrento: a virtual walk across history.* In A. Celentano (Ed.), *AVI.* ACM Press, 318–321.

Kautz J. and McCool M., 2000. *Approximation of Glossy Reflection with Prefiltered Environment Maps.* In *Proc. Graphics Interface.* 119–126.

Landis H., 2002. *Production-Ready Global Illumination.* In *ACM SIGGRAPH Course Notes 16.* 87–102.

Pollefeys M.; Gool L.V.; Akkermans I.; Becker D.D.; and Demuynck K., 2001. *A Guided Tour to Virtual Sagalassos.*

Raybourn E.M. and Bos N., 2005. *Design and evaluation challenges of serious games.* In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems.* ACM Press, New York, NY, USA, 2049–2050.

Todt S.; Rezk-Salama C.; Horz T.; Pritzkau A.; and Kolb A., 2007. *An Interactive Exploration of the Virtual Stronghold Dillenburg.* In *Proc. Eurographics Cultural Heritage.* 213–218.

Zhukov S.; Iones A.; and Kronin G., 1998. *An Ambient Light Illumination Model.* In *Proc. Eurographics Rendering Workshop.* 45–56.

AIBO as a needs-based companion dog

José M. Blanco Calvo, Dragoş Datcu and Léon J. M. Rothkrantz
Man-Machine Interaction Group
Delft University of Technology
Mekelweg 4, 2628CD Delft,
The Netherlands
E-mail: L.J.M.Rothkrantz@tudelft.nl

KEYWORDS
AIBO-robot, human-computer interaction, companion robot, needs-based, personality modeling, nPME model.

ABSTRACT

In this paper we describe the architecture that allows the modeling of an emotionally intelligent robotic companion dog. We chose to implement this architecture for AIBO, which is a dog-like autonomous robot, developed by Sony. AIBO was developed as an entertainment robot and its 'mind' is programmable. The focus of this paper is on creating a complex personality model that provides a realistic needs-oriented behavior for robotic companion dogs. The personality model is an extension of the nPME model and implies that needs play the most important role and influence the behavior of the AIBO robot in every situation. Personality, Mood and Emotions are three layers that, in combination with the needs, make AIBO show a real dog-like behavior. The existing personality models need to be modified, adaptated and improved so as to handle the unknown and dynamic environment in which AIBO performs. So, as part of the system design, the importance of incoming events has been considered. A modification in this architecture modeling the function of amygdale has been introduced to speed up the resolution of some situations. The implemented architecture is also designed to be suitable for future extensions in the model. The prototype has been tested in different scenarios using an AIBO robot. An extensive user study has been carried on involving several students and workers to test the realism, emotional responses and event coherency in robot's behavior.

INTRODUCTION

The societal relevance of intelligent robots is increasing nowadays covering a wide range of applications: from entertainment robots such as conversational partners, soccer players, or companion dogs, to robots that provide independent living support for elderly users in basic activities such as mobility or household maintenance (Heerink et al. 2006). Security tasks can also be performed by surveillance robots like AIBO watchdogs (Yang et al. 2006). Studies conducted by Masahiro Fujita prove that AIBO seems to be a good partner for users, having a positive effect on their emotional state (Fujita 2004). Robots have shown their utility even in medical domains by aiding the diagnosis and therapy of diseases as autism (Scassellati 2005). According to Fujita the current implementation of AIBO software provided by Sony uses behaviors that come from a "manually designed database" (Arkin et al. 2003). The next steps would be a learning-based systems or evolving systems able to create new behaviors through the continuous interaction with both human beings and environment. Mature systems would play also an important role in the future of entertainment robotics.

Figure 1: AIBO robot as a companion dog

This project focuses on the idea of improving interaction between man and machine (Figure 1) through the creation of a biological inspired model of human mind. This artificial brain must act coherently with the environment, drawing conclusions according to the different situations that can arise during a day. To achieve this, it is necessary to create a personality model capable of reasoning about a certain situation to perform complex actions in every context. So there must be a whole coherence between events, actions, and emotions.
The situation awareness process would include a reasoning system that combines the different personality parameters (PME) with the levels of certain needs (n). Triggered by incoming events it should be able to perform a set of actions to satisfy the needs, in the similar way a real dog would do.
This paper is organized as follows: first we will introduce the original nPME model (Dobai and Rothkrantz 2005); then our extension will be explained; next section will describe the architecture implemented to test the model; and finally, a summary of the evaluation method and results in the user study will be provided.

THE nPME MODEL

So far, the design of virtual humans, agents or game characters used mostly two categories of personality models: PME models that take into consideration personality (P), mood (M) and emotions (E) and PE models that are based solely on personality (P) and emotions (E). The work of (Ksirsagar and Magnenat-Thalmann 2002) described an layered based architecture for modeling personality, moods and emotions.
nPME model combines Personality, Mood, Emotions and also 'needs' to establish a complex behaviour model. For the nPME model we started with a layered PME architecture assumes the mood is seen as an intermediate layer between personality and emotions and therefore is influenced by both. Needs are used to let AIBO act independently in a unpredictable and changing environment (nPME). The inclusion of the needs modifies the role of the mood, and now mood is not directly influenced by emotions or personality, but indirectly as a consequence of the evaluation of the goals, preferences and standards in regards with the events that occur in the environment.

Concepts

Personality.
The most important factor in the nPME model is the personality parameter itself. Among the existing psychological models of personality the most widely accepted is the Five Factor Model that describes and measures the personality using five broad dimensions: *Openness*, *Conscientiousness*, *Extraversion*, *Agreeableness* and *Neuroticism*. This model is also known as the OCEAN model (Beaumont 2003). Every personality can be classified by those parameters, although those parameters with their respective values cannot define a whole human personality. According to Paul Costa Jr. (Costa and McCrae, 1992) these parameters can experiment "nuanced" changes during a person's life. However two assumptions are done over the basis that our goal is to apply the model to a robotic dog: first, those parameters are enough for define the dog-like personality. Secondly, we assume that those parameters are kept constant because their variations along a robot life are imperceptible. According to this, the personality will not change during the life of the robot dog.

Needs.
Primary emotions are considered to be driven by the basic needs of surveillance and social affiliations (de Sousa 2003). Moreover, according to Ekman the expression and recognition of those simpler emotions are universal (Ekman and Friesen 1989). Therefore the theoretical basis for the use of needs has been proved. Needs are the basic engine that run emotions and these can be modeled equally for different subjects. The model of needs is based on the theory introduced by Maslow (Maslow 1970). Needs are organized in a pyramid (Figure 2): the need that is at the bottom of the pyramid is the first to be satisfied. On the other side, the one which is on the top will only be satisfied if all the rest under it have already been satisfied.

This model is powerful enough to allow for the development of a complex personality and it is also simple enough to implement. The pyramid of needs that Maslow introduced as a base for motivation theory contains 5 categories of needs: physiological, safety, love and belonging, self esteem and self actualization. The first 4 categories of needs have been introduced by Maslow as "*deficit needs*" while the last category of needs is known as "*being needs*". In the nPME model only "deficit needs" are taken into consideration. Self actualization needs are seen as growing (a continuous driving force in a very long term) and Maslow states that not everyone ultimately seeks self-actualization (Maslow 1970).

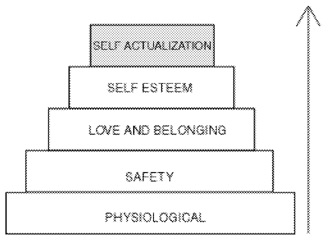


Figure 2: Maslow Pyramid of Needs

According to the autonomy shown by current robots, these need would not be met ever so it can be skipped for a robot model. In practice the different categories of needs are mapped into AIBO specific needs (e.g. physiological needs are represented by the battery level, etc.). Needs have priorities and act as thermometers in the sense that once critical values have been reached by different categories of needs, depending on their priorities, they lead to different structures of goals, preferences and standards (e.g. once the battery reaches a critical value of 10 % a new goal with the highest priority is being generated that states: recharge, etc.) (Dobai and Rothkrantz 2005).

Mood.
Mood is a conscious and prolonged state of mind that directly controls the emotions. While emotions are instantaneous, mood is constant for longer time spans. Without mood, the link between personality that has the sense of "eternal", and the emotional expressions that are instantaneous, vanishes. In our approach, the mood is based on a model which basically applies a projection on a two dimensional space having valence and arousal as axes, as proposed by Lang (Bartneck 2002). Valence refers to whether the mood is positive or negative and arousal refers to the intensity of the mood.

Emotional States and Expressions.
An emotional state is a particular state of mind that is reflected visually by means of an emotional expression. To model this, the emotion categories proposed by Orthony, Clore and Collins, commonly known as the OCC model (Ortony et al. 1988) are primarily used. The model categorizes 22 different emotions based on positive or negative reactions to events, actions and objects. However, not all the possible emotions are universally expressed so a restricted set should be used. According to Ekman and Friesen (Ekman and Friesen 1989) there are six universally

expressed recognizable emotions: happiness, sadness, fear, anger, surprise and disgust. In our work we make use eventually of these emotions to model the emotional response of the robot.

Goals, Preferences and Standards.
The goals, preferences and standard parameters are set dynamically based on needs and personality. Every time changes occur in the parameters related to needs, the goals, preferences and standards are also updated (Fujita 2004). By goals we understand tasks orientated objectives that are SMART (simple, measurable, acceptable, realistic, time frame) (e.g.: aibo find ball). By preference we understand the tendency associated to aspects of objects (e.g. like/dislike of ball). By standards we understand approval/disapproval regarding actions of agents. The standards can focus on the selft agent or other agents (e.g. approval of being touched on back).

Modification of nPME model

Due to the number of variables in the original nPME model (4 needs, 5 traits of personality, 2 axis for mood, 6 emotions with different intensities, and goals, preferences and standards) the complexity of the implementation arises considerably. That is the reason that some assumptions have to imposed. These are taken under the constraint of ensuring the final efficiency of the behavior.

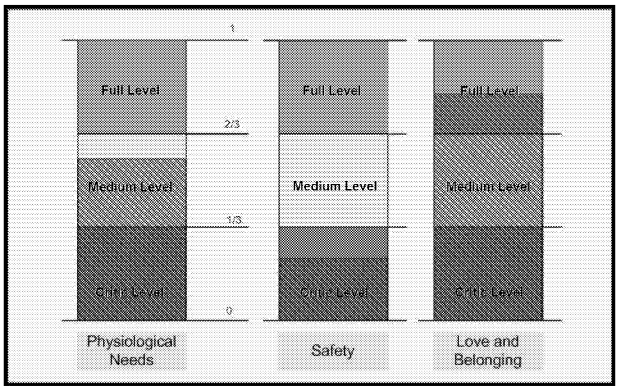


Figure 3: The bucket model of the behavioral needs

The five traits in the personality parameter have been removed, under the assumption that personality can be seen as a continuous underlying line of reacting and making choices when facing different situations. This means that two different personalities may show some differences in the emotional responses and different ways of acting, while facing the same situation.
In this case we have created 4 fixed personalities named with the most prominent feature of their behavior: *friendly*, *unfriendly*, *aggressive* and *dissobedient*. The needs have been reduced to three: *physiological*, *safety* and *love&belonging*. This has been done because the fourth need has not been proved to exist in a real dog. Needs are the main forces that influence the behavior. They can be seen as buckets (Figure 3) with certain levels of water. Accordingly, the water is dropping during time, and when the level is too low, some actions must be performed to fill the bucket (satisfy the need). We set the satisfaction of needs as the main criteria to satisfy in any situation by always keeping the priority of each need as explained in Maslow Pyramid of Needs model.
The model of mood by Lang (Lang 1995) with two axes (*valence* and *arousal*) has been simplified to only one axis meaning the intensity of the mood (arousal). Valence is to be translated in the value of the arousal. If its value is higher than 0, it means positive valence, otherwise being considered negative valence. The goals, preferences and standards have been removed as intermediates. The combination of need to satisfy and personality stereotype, and that of mood in the context of incoming events, will give the appropriate emotional response.

DESIGN CONCEPT FOR COMPANION AIBO

A modular architecture

Figure 4 illustrates the whole architecture that integrates the personality model described above and the division of logical parts into system components. It has 4 different modules to control the connection, incoming events, scripts to execute, the reasoning process and one more that is the central control unit.
The architecture has been designed to allow for easy integration of the exisisting modules. The centralized architecture allows the incoming events to be automatically processed by the Central Control Unit - CCU and the events module through the connection control to the reasoning module. The most important modules are the CCU and the reasoning module. The former controls all the system, the latter is the module where the extended nPME model has been implemented.

Reasoning module.
All the personality components of the nPME model are created and managed in the reasoning module in combination with incoming events coming from Events module through Central Control Module.
The reasoning unit manages all the components of the nPME model and, in the presence of contextual events, it generates a proper response to the CCU. The reasoning unit has two main components that are called inference engines(Figure 5).
The first inference engine initially checks if there is any important incoming event. In that case it will adjust the values of the needs to make them more representative given the current situation. Then the engine rechecks the values of the needs and it chooses the one that is assigned with the highest priority. The selected need is the one that must be satisfied immediately.
The second inference engine makes use of a complete set of different rules for each predefined personality. That means that a certain personality can hold totally different ways to satisfy a need than the rest of the personalities.
Considering the appropriate set of rules defined, the need to satisfy, the value of the mood and the knowledge of the previous scripts (that are already executed), the second engine takes the decision about which is the next script to execute and which emotion will combine with it.

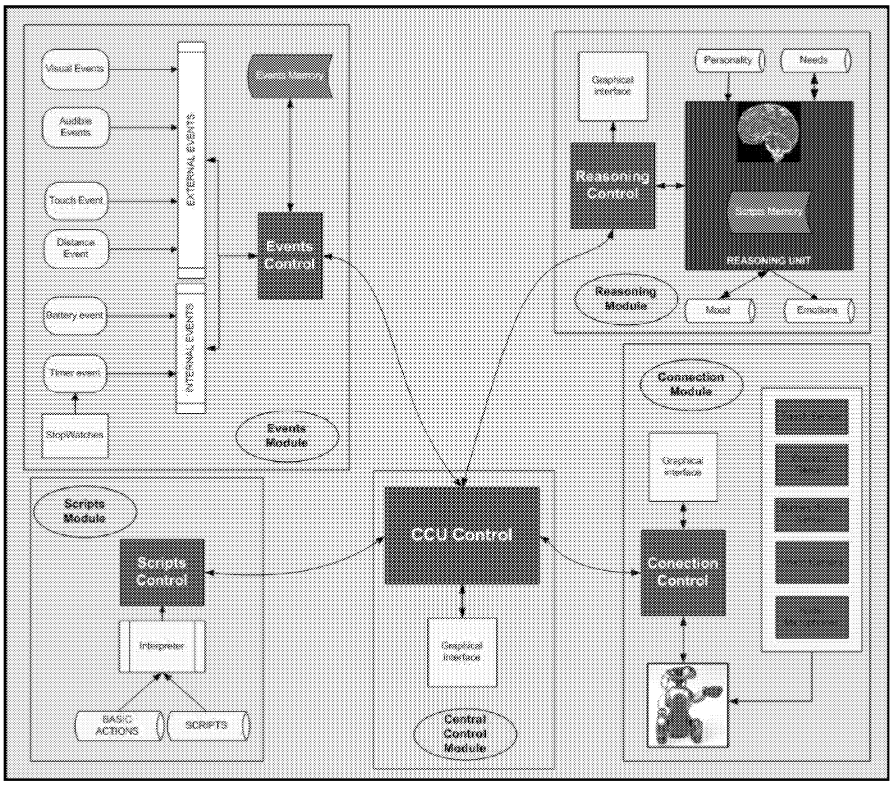


Figure 4: The architecture of the system

Once the reasoning algorithm has finished, the reasoning control sends the information related to the script and the emotion to the CCU so as to be executed by AIBO server. At the same time, the values of the needs, the mood, and emotions are updated according to the chosen script that will subsequently be ready for the next iteration.

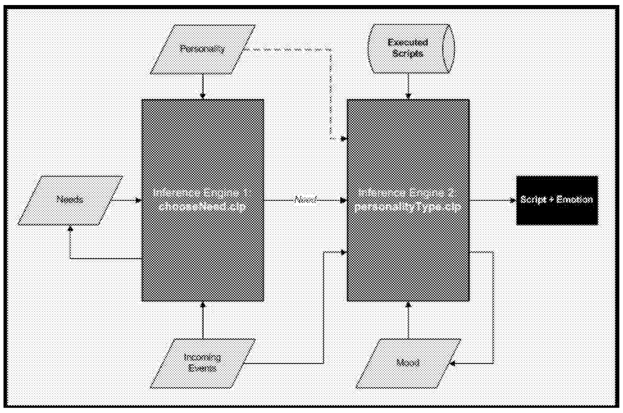

Personality
Executed Scripts
Needs
Inference Engine 1 chooseNeed.clp
Need
Inference Engine 2 personalityType.clp
Script + Emotion
Incoming Events
Mood

Figure 5: The inference engines of the reasoning system

Central Control Unit - CCU.
The Central Control Unit manipulates what every module of the framework. On one hand, it supervise the information transfer from one module to the another. On the other hand, it manages the main time line of the system, meaning that it supplies execution commands to every module at certain times. In a real-time system that functionality is very useful because different modules perform in conformance with their own time line (new events may occur while the robot is already performing a set of actions, etc). The central control makes possible to avoid mistakes if, for example, one module requires some information that is still being processed by another module.
While the robot is in operation, the CCU is listening all the time to the incoming events. When a new event occurs, it passes the information to the reasoning module. This will take the appropriate actions and will provide back an action script to the CCU. Every time the reasoning module generates a script, the CCU translates it into sets of URBI commands using the scripts module. As soon as a script has been translated, the CCU sends the information to the connection module. This will send the right commands to AIBO server, and so makes it to perform certain actions. Once AIBO robot has finished to perform the actions, the CCU obtains the control again and performs another iteration, meaning that the process continues again with the reasoning module.

Development perspective

The modular architecture we have introduced in this paper is based on an existing platform that was developed during the URBI Project at ENSTA Laboratories. URBI (Universal Robotic Body Interface - www.urbiforge.com) is a scripted language designed to operate over a client-server architecture in order to remotely control a robot. URBI is released under the GNU General Public License. Our present prototype runs the server on AIBO (the original URBI server) and the client on a PC (developed by us). The AIBO mind support is resident on the PC. AIBO robot is in charge of sending raw data information from the sensors and of executing complex commands written in the URBI scripting language. Our long term goal is to have both the client and the server be running on the AIBO robot and so to have a perfectly autonomous robot.

EVALUATION AND RESULTS

A user study has been done to test the correctness of the prototype in terms of behavior. The study expected to evaluate the realism of the prototype, the level of coherence for the events-actions-emotions, the continuity of the behavior and the personality designs (Figure 6). For the experiment four different scenarios were developed. AIBO had to face different situations and its goal was to satisfy its most priority need. The performance was measured based on a comprehensive analysis of the reaction of the robot dog, obtained from a set of different video recordings in which up to 24 potential users (students of the university and independent workers) have participated. The people were divided in two groups, each group having to act specific roles, according to different particular personalities (friendly and unfriendly).
The results of the analysis suggest that, even if there have been noticed some discrepancies between the reactions of the robot dog and a real companion dog, the behavior was mostly coherent and well-adapted to each of the contexts. Further work is to be carried on for accurately examining the impact of improper robot reactions given the context and the personality. The movements were mostly natural and continuous for a robotic dog and the emotion responses were mostly clear and well adapted to the situations.

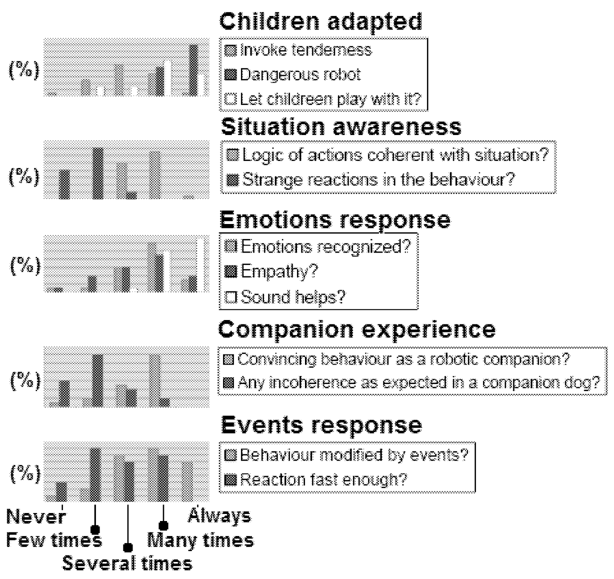


Figure 6: The results for the evaluation of the nPME model for AIBO robot

CONCLUSIONS

For creating an autonomous robot that acts coherently in an unpredictable and changing environment, a main force that drives its behavior must be designed: this force is the satisfaction of the needs. The needs have to be considered to be combinated with the events that define the environment and always together with the emotion response that defines the robot attitude towards the situation. The emotion responses are directly related to the election of the personality and the level of mood. Having all this information, the system is able to choose from a set of actions to be performed by the AIBO robot. As proved by the user study, our approach gave very positive results when comparing the behavior of the AIBO robot dog with the behavior of a real companion dog.

REFERENCES

Arkin, R.C.; M. Fujita; T. Takagi; R. Hasegawa. 2003. “An Ethological and Emotional Basis for Human Robot Interaction”. In proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, No.3(Mar), 191-201.

Bartneck, C. 2002. “Integrating the OCC Model of Emotions in Embodied Characters”, Workshop on Virtual Conversational Characters.

Beaumont, R. L. 2003. “Five Factor Constellations and Popular Personality Types”, Psychology 106.

Costa, P.T. and R.R. McCrae. 1992. “Normal personality assessment in clinical practice: The NEO personality inventory”. Psychological Assessment.

Dobai, I.; L. Rothkrantz; C. van der Mast. 2005. "Personality model for a companion AIBO", ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ISBN 1-59593-110-4, ACM, Broadway New York, 438-441.

Ekman, P. and Friesen, W.V. 1989. “The argument and Evidence About Universals in Facial Expressions of Emotion”. In Handbook of Social Psychophysiology. New York: John Wiley and Sons, Ltd.

Fujita, M. 2004. “On activating Human Communication with Pet-Type Robot AIBO”. In Proceedings of the IEEE, Vol. 92, No. 11, 1804-1813.

Heerink M.; B. Kröse; V. Evers and B. Wielinga. 2006. “The Influence of a Robot’s Social Abilities on Acceptance by Elderly Users”. Amsterdam, The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006 - ROMAN 2006, ISBN: 1-4244-0565-3.

Ksirsagar, S. and N. Magnenat-Thalmann. 2002. “A Multilayer Personality Model”, In Proceedings of 2nd International Symposium on Smart Graphics, ISBN: 1-58113-555-6, 107-115.

Lang, P. J. 1995. “The Emotion Probe: studies of motivation and attention, A study in the Neuroscience of Love and Hate”. Hillside, NJ: Lawrence Erlbaum Associates, Publishers.

Maslow, A.H. 1970. “Motivation and Personality”, 2nd. ed., New York, Harper & Row.

Ortony, A.; G. L. Clore and A. Collins. 1988. “The Cognitive Structure of Emotions”, Cambridge University Press.

Scassellati, B. 2005. “Using social robots to study abnormal social development”. In proceedings of the 5th International Workshop on Epigenetic Robotics. ISBN 91-974741-4-2.

de Sousa, R. 2003. "Emotion", Stanford Encyclopedia of Philosophy (Spring 2003 Edition), Edward N. Zalta (ed.), http://plato.stanford.edu/archives/spr2003/entries/emotion.

Yang, Z; B. T. Hau and L.J.M. Rothkrantz. 2006. "AIBO as a watchdog". Game-On 2006, Eurosis, No.7(Dec), 74-78.

www.urbiforge.com

AGENT BASED VIRTUAL TUTORSHIP AND E-LEARNING TECHNIQUES APPLIED TO A BUSINESS GAME BUILT ON SYSTEM DYNAMICS

Marco Remondino
Department of Computer Science
University of Turin
Cso Svizzera 185,
Italy
E-mail: remond@di.unito.it

KEYWORDS
Reinforcement Learning, Action Selection, E-Learning, software agents, Enterprise Simulation.

ABSTRACT

An advanced Business Game is presented in the paper, built on the methodology of System Dynamics. It can be used for cognitive learning and knowledge transmission in schools and Universities; it allows the learners to take decisions at each time step, after which it calculates the corresponding results, showing them according to the principles of double entry accounting.
An agent based framework is then discussed, which constitutes a form of virtual tutorship for the learners. The agents act as a decision support system for the decisions to be taken, and can explain some cause/effect relations. The agents themselves learn how the model work by practicing it, through some reinforcement learning techniques.

INTRODUCTION

Business Games (BG) can be considered a sort of role playing games, caracterized by a managerial context. The players usually face some situations typical for enterprise management and must take various core decisions, mainly about marketing, logistics, production, research and developement politics and so on. A very interesting feature of business games is that they can be employed as a teaching instrument; the students can learn some important concepts about enterprise management, by trying them on the field, instead of just studying them on books. This is reguarded as "learning by doing" concept.

For this reason, in the following discussion, the players will be defined as “learners”. The main didactic goals for BGs are to refine the decision capacities of the learners when facing situations of uncertainty, and above all their ability to take managerial decisions when there is a tradeoff between risk and profit. Besides, through a BG, some advanced managerial techniques can be teached, and so can be the interaction among the different enterprise functions.

The BG presented here implies that the learners must be organized in teams; each of them represents the manager of a single area (production, sales, research & development, manufacturing and so on) and they must coordinate themselves in order to achieve the best possible result. This is ment to promote a collective knowledge, that is valuable in the real world.

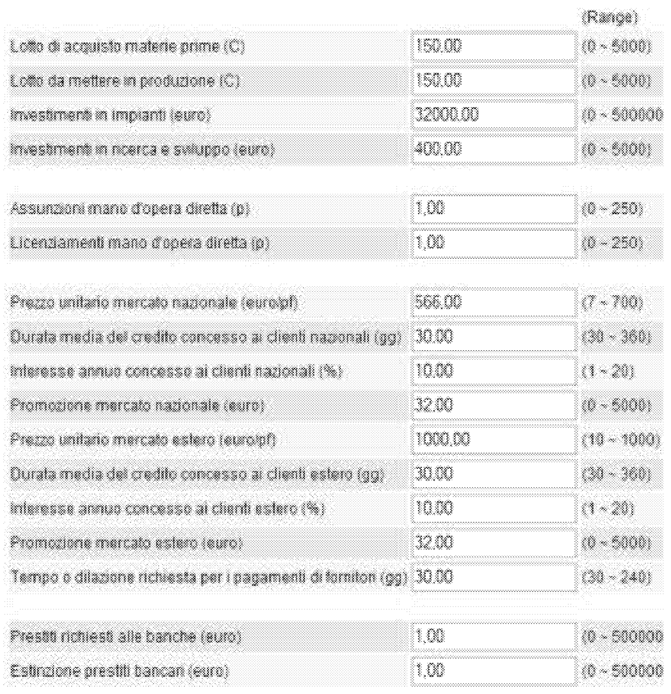
The presented BG is built on the System Dynamics methodology (Forrester, 1961); this means that the mechanisms of the game are based on finite differences equations and curves defining the main parameters of the game itself.

An agent based framework is applied in the form of virtual tutoring system for the learners; the intelligent agents learn through a trial and error technique, based on Reinforcement Learning paradigms, by practicing the system. After this trial period, they form a model of how the simulation works (in particular of the cause/effect relations among the decisions and the observed results) and then they can be used as a decision support system for the human learners, during the game play.

A WEB BASED BUSINESS GAME

An existing simulation framework is described in this paragraph, used as a teaching support in some University courses at the Department of Computer Science, University of Turin, Italy; this will be the one to which the agent based paradigms described before will be applied, in order to obtain a virtual tutorship and a decision support system for the learners (the users). The simulation framework (http://e-lab.di.unito.it/SimulazioneAziendale), developed by prof. Gianpiero Bussolin and Dr. Marco Remondino at the University of Turin, Department of Computer Science, has been conceived as a teaching platform, used for transmitting such concepts as “double-entry accounting”, and the way in which the decisions taken in a real enterprise affect the synthetic results, at the end of each period (month). The model, for now, is just in Italian, but a translation in English will soon be available.

In this model, the users have to take a number of core decisions at the beginning of every month; the system, based on Forrester’s System Dynamics, generates a set of reports, typical for Management and Enterprise analysis. The users, by reading these reports, can track down the influence of the single decision – or even better the aggregate effects coming from two or more decisions – on the synthetic results, representing the monthly performance of the whole enterprise.



		(Range)
Lotto di acquisto materie prime (C)	150.00	(0 ~ 5000)
Lotto da mettere in produzione (C)	150.00	(0 ~ 5000)
Investimenti in impianti (euro)	32000.00	(0 ~ 500000
Investimenti in ricerca e sviluppo (euro)	400.00	(0 ~ 5000)
Assunzioni mano d'opera diretta (p)	1,00	(0 ~ 250)
Licenziamenti mano d'opera diretta (p)	1,00	(0 ~ 250)
Prezzo unitario mercato nazionale (euro/pf)	566.00	(7 ~ 700)
Durata media del credito concesso ai clienti nazionali (gg)	30.00	(30 ~ 360)
Interesse annuo concesso ai clienti nazionali (%)	10.00	(1 ~ 20)
Promozione mercato nazionale (euro)	32.00	(0 ~ 5000)
Prezzo unitario mercato estero (euro/pf)	1000.00	(10 ~ 1000)
Durata media del credito concesso ai clienti estero (gg)	30.00	(30 ~ 360)
Interesse annuo concesso ai clienti estero (%)	10.00	(1 ~ 20)
Promozione mercato estero (euro)	32.00	(0 ~ 5000)
Tempo o dilazione richiesta per i pagamenti di fornitori (gg)	30.00	(30 ~ 240)
Prestiti richiesti alle banche (euro)	1,00	(0 ~ 500000
Estinzione prestiti bancari (euro)	1,00	(0 ~ 500000

Figure 1: a form for monthly decisions

Agents can have many roles in such a system; first of all, reactive agents can be used as a part of the system, in order to simulate customers or suppliers. These agents should be very simple, just reacting to some market curve. On the other hand, reactive agents could also be the production implants, with the possibility of being programmed by the users in some way, and then adapting themselves to the number of pieces to be produced, and so on. This kind of interactivity would make the model more realistic.
Cognitive agents may have different and more important roles in this kind of models used for e-learning. After a training period on the model itself, using the reinforcement learning methods discussed above, an agent can compute some strategies to be used to make profit in the simulation. That said, this agent could then be used both as a decision support system for human users – since it could foresee some results, based on its acquired experience – and as a virtual tutor, explaining the relations among certain variables (decisions) and the achieved results. This could help the learners to understand the cause/effect links.

The Inner Structure of the Model

The model is built using a structure based on the theory of System Dynamics.
The model itself is considered as an artefact, an interface between the *internal structure* (implemented in Java) and the *external environment*, i.e.: the physical one, in which the system itself is used by the learners, i.e.: the final users of the model.
There are six main subsystems, mutually connected, in the simulated enterprise: production, finance, emplants, research and development, marketing and sales. Some of these subsystems are divided into other subsystems, if needed (e.g.: national sales and sales to the rest of the World).

The model is a dynamic system and the temporal walkthrough in the system has been converted into a set of differential equations and laws that can generate the walkthrough itself. This description consists into a constant relation between the system status in a generic time T and the status after a brief time interval "delta T" (DT).
Two are the main variable types in the model: the stock type and the flow type (or rate). The latters are used to recalculate the formers after each DT.

Many of these flows are generated by the "actions" of the learners, i.e.: their decisions, in order to modify the states of the system. Not all the stats are modifyied by external actions, though. There exist some inner actions and regulations that can be considered as "internal impicit decisions" performed by the system, used to normalize the levels. The choice of the configuration and balance among the external decisions and implicit decisions identifies the nature and type of knowledge that has to be transferred to the learner in a direct od indirect way.
The external decisions are those that make it possible for the individual learners to know the object of their studies, since the object is directly "acted upon" by them. This kind of actions are simply referred to as "decisions", since they can be carried on by the learners. The other kind of decisions are those that make it possible to keep the system "alive" even when the learners (for a lack of knowledge) has not been able to lead the system.
The enterprise is part of a bigger external environment (or space) with which it continuously interacts. This environment is configured by some other sub-systems, like the banking system (able to supply the financial means for the developing of new technologies, new products and the enterprise itself), the market system (where the demand is generated in the form of orders for the enterprise), the technology system (that determines what kinds of technologies are available at a certain time step), the suppliers system and customers system (respectively simulating those sides) and the workforce system (determing the average wages, the work supply on the market and so on).

The equations in the model are in the form of:

$$SFi = SSi + (RIi - ROi) * DT \qquad (1)$$

Where SFi at the first member is the i-th Stock Variable at the end of a DT, while the SSi on the right is the same variable at the beginning of the DT. RIi and ROi are respectively the Input Rate and Output Rate relative to the i-th stock variable.
The variation is then depicted as a difference among the Input Rate and Outup Rate during the considered DT; this is summed to the previous stock value, to calculate the new one. It's important to notice that the algebric difference amond the two rates is to be weighted by the time in which that rates applied.
The units of measurement in the system directly derive from the above equation. The time is measured in *months* and the stocks are measured in *units*. The rates are then units/month and DT is again measured in months.

DT is a very brief time period; for simplicity, in the model is's set to be 1/100 of a month.

TWO AGENT BASED PARADIGMS

The term agent, deriving from the Latin *agens*, identifies someone (or something) who acts; the same word can also be used to define a mean through which some action is made or caused. In Nwana (1996) the studies on software agents have been divided into two main strands: the first one, starting in 1977, is based on the studies on Distributed Artificial Intelligence and gave origin to the cognitive agents, endowed with inner symbolic models. The second one, whose origins are to be found in the 90s, focuses on a wide range of agents, defined as reactive. They do not have any internal representation of their environment and the emphasis not on the way in which they decide what to do, but simply on when to act and what action to choose, basing on the stimuli from the environment. Reactive agents simply retrieve pre-set behaviors similar to reflexes without maintaining any internal state.
Both paradigms, though quite different, have some peculiar features that make them suitable for some given situations; the main problem with a purely cognitive agent, when dealing with real-time systems, is reaction time. For simple, well known situations, reasoning may not be required at all. In some real-time domains, minimizing the latency between changes in world state and reactions is important and so reactive agents can be successfully employed. On the other end, cognitive agents should be used when artificial learning or reasoning is concerned.

Reactive (sub-symbolic) Agents

This kind of agents may be regarded as any simple (not structured) software entities which interact among them and with the environment. A multi-agent context of this kind allows the emergency of complex behavior and self-organization, with no definition of a formal rule a priori. Apparent intelligent behavior is a product of the interaction among agents and environment, and of the interaction among many simple behaviors. It can be really hard to describe the real world under every aspect: some fundamental macro-actions can thus be defined on single agents, which allow cooperation with the environment and with other agents. The concept of Multi Agent System for Social Simulations is thus introduced: the single agents have a very simple structure. Only few details and actions are described for the entities: the behavior of the whole system is a consequence of those of the single agents, but it's not necessarily the sum of them. This can bring to unpredictable results, when the simulated system is studied.
In some situations, effective results can be obtained just by building simple, sub-symbolic agents, whose behavior is randomly determined or is built by applying fixed pre defined reaction rules; this is the case, for instance, of *Heatbugs*, one of the canonical Swarm demonstrations (www.swarm.org):

"*It's an example of how simple agents acting only on local information can produce complex global behavior. As we read on Swarm main site, each agent in this model is a heatbug. The world has a spatial property, heat, which diffuses and evaporates over time. In this picture, green dots represent heatbugs, brighter red represents warmer spots of the world. Each heatbug puts out a small amount of heat, and also has a certain ideal temperature it wants to be. The system itself is a simple time stepped model: each time step, the heatbug looks moves to a nearby spot that will make it happier and then puts out a bit of heat. One heatbug by itself can't be warm enough, so over time they tend to cluster together for warmth*"

Cognitive (symbolic) Agents

These agents' behavior is goal-directed and reasons-based; i.e. is intentional action. The agent bases its goal-adoption, its preferences and decisions, and its actions on its Beliefs. In [8] we read that a software cognitive agent should feature the following properties:

- *autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- *social ability*: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- *reactivity*: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative." The Wooldridge and Jennings definition, in addition to spelling out autonomy, sensing and acting, allows for a broad, but finite, range of environments. They further add a communications requirement.

In general, cognitive agents have some sort of behavioral pattern, that can be described through modal logic, equations, evolutionary algorithms and so forth. In order to make the agents able to learn and improve themselves, some reinforcement algorithms can be embedded into their structure.

ACTION SELECTION AND REINFORCEMENT LEARNING FOR THE AGENTS

The action selection problem at time t+1, along with the goal selection, at a macro (aggregate) level, are central topics, when the agents must "learn" how the models works, by experimenting on it and being able to act as a decision support system for human users.
For action selection we do not simply mean the problem of choosing which action to take at a micro level (agent level), but also which one, among the possible goals, to select. The first level of detail is typical for reactive agents; in fact they don't have any goals, if not those imposed by the external environment. The cognitive agents feature both levels of detail. In this work, the action selection problem is crucial

for the formal definition of the involved agents, especially when they are employed as a decision support system, thus requiring some learning ability.
The goals could be combined to form higher level objectives or, on the contrary, be incompatible among them. In order to decide which actions to take, it's necessary to evaluate the utility for each of them; specific Reinforcement Learning (RL) algorithms are used for this purpose. These transform quantitative data (the payoff) in behavioural patterns for the agents.
An agent endowed with some RL algorithm, when in a particular state of the world (x), makes an action (a) and gets a payoff (r), calculated by a reward function based on the consequences of the action itself. Through a trial & error mechanism the agent learns what are the actions that maximise this numerical value. An effective RL algorithm is the one called Q-learning [Watkins, 1992]: an agent "lives" in a world modelled as a Markovian Decision Process (MDP), that's a set of states X, in which some actions from the set A can be done. For a state x, belonging to X, and an action a, belonging to A, there exist a probability function $P_{xa}(y)$, determining the transaction probability towards a new state y. At the same time, for each couple of possible x and a, there exist another probability function $P_{xa}(r)$, determining the payoff – or reward – r, generated by the action. We obviously have that:

$$\sum_y P_{xa}(y) = 1 \tag{2}$$

And that:

$$\sum_r P_{xa}(r) = 1 \tag{3}$$

The Q-learning algorithm builds the so called Q-values, Q(x,a), by considering not only the payoff of a singular action, but also the expected discounted sum of future payoffs obtained by taking action a from state x and following an optimal policy thereafter. So we have that in each time-step t:

$$R = r(t) + \lambda * r(t+1) + \lambda^2 * r(t+2) + ... \tag{4}$$

where $\lambda \leq 1$ is the discount factor.

By defining policy (π) a set of action rules, given a state, we have that by following that policy, the total discounted reward at time t equals the previous formula with E(r) instead of r, that's its expected value defined as:

$$E(r) = \sum_y r(x,a) * P_{xa}(y) \tag{5}$$

Besides acting as a decision support system for human learners experiencing with the model, artificial agents can be used to supervise the decision taken by learners, in order to interpret them in a cognitive way. For example, in the previously mentioned enterprise accounting model, some users could immediately pursue an high profit, while others could be concerned first with the expansion of their enterprise on new markets. Others could choose to improve industrial plants, while others could want to differentiate production and invest on research & development and marketing. All these decisions are complex, since they are determined by the combination of many different variables. Sometimes the learners won't even realize that they are pursuing a strategy instead of another one, and they often won't foresee what the selected strategy could bring.
That's where the motivational model is employed; this is based on a function called wellbeing, where the intensity of the motivation to take a certain decision comes from the combination of two factors: internal drive and in a limited way, some external stimuli. We have that:

$$M_i = D_i + w_i \tag{6}$$

In order to give more relevance to the first term, an activation threshold can be used, such as:

$$\text{if } D_i \leq L_i \Rightarrow M_i = 0$$
$$\text{if } D_i > L_i \Rightarrow M_i = D_i + w_i \tag{7}$$

The wellbeing variable is calculated as the difference between the highest possible value and the sum of the motivational drives, weighted by a factor α, which represent in a cognitive way the personality of the human agent (the weight that the learner gives to the single decisions). We have that:

$$WB = WB\_\max - \sum_i \alpha_i * D_i \tag{8}$$

The agents can also constitute some parts of the model itself; in the considered enterprise accounting model, some reactive agents can form the supply chain, or the warehouses, or even the competitors operating on the same market.

When dealing with reactive agents, the action selection problem is to be found at a macro (aggregate) level, i.e.: population level. If reactive agents are the competitors of human learners in the simulated world, they could have a fixed rule of behavior over time. Some evolutionary algorithms could be embedded in the agents, so that the best players on the market could merge, to form some other artificial players with an even better behavior.
In this way it's possible to start with a population of agents with a random behavior, facing the standard decisions in the model, and select – through the various "generations" – the best ones. So it's not the single agent that selects his behavior by updating its own policy (that remains the same, being the agent a reactive one), but the population that evolves over time, through the mechanism of reproduction and mutation.
This is an approach often used when the rules of the environment are given and the main task is to observe some emerging aggregate behavior arising from simple entities, i.e.: reactive agents. Since these agents does not feature a goal based – pro-active – behavior, the way they act tends to

be deeply dependent on the choices made by the designer. In order to design flexible systems, the aggregate behavior (at population level, i.e.: macro level) can be made self-adaptive through the implementation of an evolutionary algorithm (EA). In this case the agents will have a wired random behavior at the beginning, and evolve according to the environment in which they act, through a selection mechanism.

EA derive from observations of biological evolution. Genetic Algorithms (GA) [Holland, 1975] are inspired by Darwin's theory of evolution, often explained as "survival of the fittest": individuals are modelled as strings of binary digits and are the encode for the solution to some problem. The first generation of individuals is often created randomly, and then some fitness rules are given (i.e. better solutions for a particular problem), in order to select the fittest entities. The selected ones will survive, while the others will be killed; during the next step, a crossover between some of the fittest entities occurs, thus creating new individuals, directly derived from the best ones of the previous generation. Again, the fitness check is operated, thus selecting the ones that give better solutions to the given problem, and so on. In order to insert a random variable in the genetic paradigm, that's something crucial in the real world, a probability of mutation is given; this means that from one generation to the next one, one or more bits of some strings can change randomly. This creates totally new individuals, thus not leaving us only with the direct derivatives of the very first generation. GA have proven to be effective problem solvers, especially for multi-parameter function optimization, when a near optimum result is enough and the real optimum is not needed.

CONCLUSION AND FUTURE DIRECTIONS

A cognitive business game has been presented in this paper, used to form learners in the Universities and schools. The structure of the model is built on the theory of System Dynamics, by using the concept of stocks and rates, and considering the variations of the stocks as the difference among the input and output rates, multiplied by a delta T, a very short time interval.

The inner structure of the model has been briefly described in the paper, along the main sub-systems tied to form the whole.

The users of the system (called "learners") must take decisions at each time step, after which the system calculates the corresponding results, showing them according to the principles of double entry accounting.

Some agent based paradigms are then described as a future development for the system itself. The agent based framework will constitute a form of virtual tutorship for the learners. The agents act as a decision support system for the decisions to be taken, and can explain some cause/effect relations. The agents themselves learn how the model work by practicing it, through some reinforcement learning techniques, and are then able to assist the learners in the decision process.

ACKNOWLEDGMENTS

I would like to gratefully acknowledge the key support of *prof. Gianpiero Bussolin*, who originally designed and participated to the implementation of the Enterprise Simulator used as the starting point for many ideas described in this paper. I would also like to thank e-business LAB and in particular *prof. Marco Pironti* (University of Turin, Italy) for his precious support. I'd also like to aknowledge *the managerial board of Fontazione CRT*. Last but not least a big thanks to *Stefano Gabutti*, for his precious help and ongoing support.

REFERENCES

Singh, H. (2003): *Building Effective Blended Learning Programs*, in Issue of Educational Technology, Volume 43, Number 6, Pages 51-54.

Simon, H. A. (1996): *The Sciences of the Artificial*, (third ed.). Cambridge, MA, MIT Press

Nwana, H.S. (1996): *Software Agents: an Overview*, in Knowledge Engineering Review, Vol. 11, N. 3, pp.1-40, Cambridge University Press.

Sutton, R. S. (1998): *Reinforcement Learning: an Introduction*, MIT press

Watkins, C. J. C. H., Dayan P. (1992): *Q-Learning*, Sprinter edt.

Gadanho S. C. (2003): *Learning behavior-selection by emotions and cognition in a multi-goal robot task.* The Journal of Machine Learning Research. Volume 4 Pages: 385 – 412. MIT Press Cambridge, MA, USA.

Holland J.H. (1975): *Adaptation in natural and artificial system*, Ann Arbor, The University of Michigan Press

Woolridge, M., and Jennings, N. (1995): *Intelligent agents: Theory and practice.* Knowledge Engineering Review 10(2). pp. 115-152

BIOGRAPHY

MARCO REMONDINO got his **Master Degree in Economics** at the beginning of 2001, with *110/110 cum Laude et Mentione*. Some months later, he started a **PhD in Computer Science**, during which he delved into theoretical studies about the structure of different software agents, namely the reactive and deliberative (BDI) ones. He completed his PhD in January 2005.

After that, he was awarded a **two-year research scholarship** from ISI Foundation, for the *Lagrange Project on Complex Systems*, during which he applied agent based paradigms to design several models in different scientific fields, namely Game Theory, Biology, Economics and Enterprise Simulation.

At present, he holds a **post-DOC research fellowship** from University of Turin, Department of Computer Science.

His main research interests are Agent Based Modelling, different paradigms for agents and their integration, computer simulation, Social Systems, emergent properties for Complex Systems, Social Networks, Action Selection, agent learning through Neural Networks, Genetic Algorithms, Classifier Systems, paradigms of Reinforcement Learning, Data Mining, validation of models and E-Learning.

AUTHOR LISTING

AUTHOR LISTING