5th INTERNATIONAL NORTH-AMERICAN CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAMEON-NA 2009

EDITED BY

Joseph Saur

And

Margaret Loper

AUGUST 26-28, 2009

GEORGIA TECH
ATLANTA, USA

A Publication of EUROSIS-ETI

Printed in Ghent, Belgium

5TH International North-American Conference

on

Intelligent Games and Simulation

ATLANTA, USA

AUGUST 26-28, 2009

Organized by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Ghent University	**GR@M**
UBISOFT	**Larian Studios**
GAME-PIPE	**The MOVES Institute**
Modelbenders	**HI-Rez Studios**

Hosted by

Georgia Tech

Atlanta, USA

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

Conference Chair
Joseph Saur, Georgia Tech, Atlanta, USA

Program Chair
Margaret Loper, Georgia Tech, Atlanta, USA

PROGRAMME COMMITTEE

Game Development Methodology
Track Chair: Licinio Roque, University of Coimbra, Coimbra, Portugal
Esteban Walter Gonzalez Clua, Universidade Federal Fluminense, Brasil
Gabriele D'Angelo, University of Bologna, Bologna, Italy
Oscar Mealha, University of Aveiro, Portugal
Jari Multisilta, University of Tampere, Finland
Ana Veloso, University of Aveiro, Portugal

Physics and Simulation

Graphics Simulation and Techniques
Stefano Ferretti, University of Bologna, Bologna, Italy
Yan Luo, National Institute of Standards and Technology, USA
Joerg Kienzle, McGill University, Montreal, Canada
Ian Marshall, Coventry University, Coventry, United Kingdom
Marco Roccetti, University of Bologna, Bologna, Italy

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

Rendering Techniques
Joern Loviscach, Fachhochschule Bielefeld, Bielefeld, Germany

PROGRAMME COMMITTEE

Artificial Intelligence

Artificial Intelligence and Simulation Tools for Game Design

Mokhtar Beldjehem, Ecole Polytechnique de Montreal, Montreal, Canada
Penny de Byl, Breda University of Applied Sciences, Breda, The Netherlands
Antonio J. Fernandez, Universidad de Malaga, Malaga, Spain
Gregory Paull, The MOVES Institute, Naval Postgraduate School, Monterey, USA
Oryal Tanir, Bell Canada, Montreal, Canada
Christian Thurau, Fraunhofer Institute, Schloss Birlinghoven, Germany

Learning & Adaptation

Christian Bauckage, Franhofer IAIS, Sankt Augustin, Germany
Christos Bouras, University of Patras, Patras, Greece
Adriano Joaquim de Oliveira Cruz, Univ. Federal de Rio de Janeiro, Rio de Janeiro, Brazil
Vinicius Fernandes dos Santos, Univ. Federal de Rio de Janeiro, Rio de Janeiro, Brazil
Andrzej Dzielinski, Warsaw University of Technology, Warsaw, Poland

Intelligent/Knowledgeable Agents

Nick Hawes, University of Birmingham, United Kingdom
Wenji Mao, Chinese Academy of Sciences, Beijing, P. R. China .
Scott Neal Reilly, Charles River Analytics, Cambridge, USA
Marco Remondino, University of Turin, Turin, Italy

Collaboration & Multi-agent Systems

Sophie Chabridon, Groupe des Ecoles de Telecommunications, Paris, France

Opponent Modelling

Ingo Steinhauser, Binary Illusions, Braunschweig, Germany

Peripheral

Psychology and Affective Computing

Bill Swartout, USC, Marina del Rey, USA

Artistic input to game and character design

Anton Eliens, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
Richard Wages, Nomads Lab, Koln, Germany

Storytelling and Natural Language Processing

Ali Arya, Carleton University, Ottawa, Canada
Jenny Brusk, Gotland University College, Gotland, Sweden
R. Michael Young, Liquid Narrative Group, North Carolina State University, Raleigh, USA
Clark Verbrugge, McGill University, Montreal, Canada

Modelling of Virtual Words

Rafael Bidarra, Delft University of Technology, Delft, The Netherlands
Ruck Thawonmas, Ritsumeikan University, Kusatsu, Shiga, Japan

PROGRAMME COMMITTEE

Online Gaming and Security Issues in Online Gaming
Pal Halvorsen, University of Oslo, Oslo, Norway
Andreas Petlund, University of Oslo, Oslo, Norway
Jouni Smed, University of Turku, Turku, Finland
Knut-Helge Vik, University of Oslo, Oslo, Norway

MMOG's
Chris Joslin, Carleton University, Ottawa, Canada
Michael J. Katchabaw, The University of Western Ontario, London, Canada
Alice Leung, BBN Technologies, Cambridge, USA
Mike Zyda, USC Viterbi School of Engineering, Marina del Rey, USA

Serious Gaming

Wargaming Aerospace Simulations, Board Games etc....
Roberto Beauclair, Institute for Pure and Applied Maths., Rio de Janiero, Brazil
Erol Gelenbe, Imperial College London, United Kingdom
Henry Lowood, Stanford University Libraries, Stanford, USA
Jaap van den Herik, Tilburg University, Tilburg, The Netherlands

Games for training
Michael J. Katchabaw, The University of Western Ontario, London, Canada
Gustavo Lyrio, IMPA, Rio de Janeiro, Brazil
Tony Manninen, University of Oulu, Oulu, Finland
Jens Mueller-Iden, Universitaet Muenster, Muenster, Germany
Maja Pivec, FH JOANNEUM, University of Applied Sciences, Graz, Austria
Martina Wilson, The Open University, Milton Keynes, United Kingdom

Games Applications in Education, Government, Health, Corporate, First Responders and Science
Paul Pivec, RaDiCAL
Daniela M. Romano, University of Sheffield, Sheffield, United Kingdom
Russell Shilling, Office of Naval Research, Arlington VA, USA

Games Interfaces - Playing outside the Box

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

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

Perceptual User Interfaces for Games
Tony Brooks, Aalborg University Esbjerg, Esbjerg, Denmark
Lachlan M. MacKinnon, University of Abertay, Dundee, United Kingdom

GAME'ON-NA 2009

All articles published in these Proceedings have been peer reviewed

EUROSIS-ETI Publications are ISI-Thomson and INSPEC referenced

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

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

Printed in Belgium by Reproduct NV, Ghent, Belgium
Final Cover Design by Grafisch Bedrijf Lammaing, Ostend, Belgium

EUROSIS-ETI Publication

ISBN: 978-90-77381-49-6
EAN: 978-90-77381-49-6

Preface

On behalf of Georgia Tech, the Georgia Tech Research Institute (of which I am a member), and the greater Georgia Tech gaming community, I'd like to welcome you to GAMEON-North America 2009, and to the fair city of Atlanta.

Over the past few years, the simulation and gaming industries have grown to include not just commercial video games, but flight simulators for training, imbedded team training environments, medical trainers. Each segment of this very large mix of hardwares and softwares has its own language, its own technologies, and its own (often very different) economic drivers. The value of conferences like this allow us, the ones who build the systems, lies in their ability to allow us to exchange thoughts and ideas, experiences and concerns, stories of what has worked in one venue (and which might be potentially migrated to another), and what has not. I know I look forward to listening to your presentations, and encourage the exchange of ideas over the next few days.

As well as the peer-reviewed papers, Game-On 'NA 2009 features a number of invited talks highlighting research done at Georgia Tech in the field of computer gaming. They are in order of presentation; “Adaptive Digital Media: Improvisation and Motivation” by Brian Magerko; “Handheld AR Games” by Blair McIntyre; “Research with a player-run virtual university in There.com” by Celia Pearce; and “The Role of AI, Storytelling, and Creativity in Entertainment” by Mark Riedl.

To finish of the research theme we will visit the Games Labs at Georgia Tech and Hi-Rez Studios.

Again, welcome to Atlanta, y'all!

Joseph Saur and Margaret Loper
Conference Chairs
Georgia Tech
Atlanta, USA

CONTENTS

CONTENTS

SCIENTIFIC PROGRAMME

GAME AI

VIRTUAL WORLD CREATION AND VISUALIZATION BY KNOWLEDGE-BASED MODELING

Jaime Zaragoza, Alma Verónica Martínez, Félix Ramos, Mario Siller, and Véronique Gaildrat
Centro de Investigación y de Estudios Avanzados, Unidad Guadalajara, México
Institut de Recherche en Informatique de Toulouse, France
email: {jzaragoz, vmartine, framos, msiller}@gdl.cinvestav.mx, veronique.gaildrat@irit.fr

KEYWORDS
Game Design, Game Engine Design, Game Environment Creation, Knowledge Bases, Distributed Systems

ABSTRACT

Virtual worlds can be used in a variety of areas, from entertainment to education, allowing us to see and interact virtual creatures and environments. The construction and correct visualization these worlds is a time and resource consuming task, which requires expertise in the modeling of 3D models and render engines. In this paper, we propose a method for creating and distributing the visual output of virtual environments, useful for any kind of simulations by means of knowledge assisted modeling.

INTRODUCTION

Nowadays computational technology allows creating rich, complex, and detailed simulations of Virtual Environments (VE). These virtual worlds can be used for severals propouses, from entertainment, such as movies or video games, to teaching, in the form of virtual trips or training sessions. The creatoin of these virtual worlds, where users interact with the entities dwelling inside these worlds, is a task which requires a multidisciplinary staff, from computer engineers, who design software that runs the simulations of the virtual environments, to artists, who create individual elements that appear in those environments. In addition, the whole process can take several hours or even years to be completed.

Several researches have proposed different methods to ease the task of creating virtual worlds, with the aim of obtaining visualizations that satisfy the user requirements, leaving the task of generating the models to graphic designers. Another method to create virtal worlds is declarative modeling (Gaildrat 2007). This is a process where users just describe the properties for the virtual world and the entities, and then let the modeler find the appropriate elements to be presented and their corresponding behaviors. The model is a coherent representation of the environment necessary to create a graphic representation of the virtual world. If such representation is in 3D, it is easier for final users to understand the events that occur in the environment.

RELATED WORK

Currently, there are not researches that deal with both the generation of the environment and the work of coherent distribution of the visualization. This lead us to present some related work in the creation of virtual environment models, as well as researches oriented towards the evolution of the visualization for the environment.

Some researches deal directly with the generation of any kind of virtual environments, using a variety of inputs, from everyday language to specialized haptic hardware. The first project is the WordsEye, a text-to-scene conversion system, developed by Bob Coyne and Richard Asproad at the AT&T laboratories which allows any user to generate a 3D scene, from a description written on natural language, using part-of-speech taggers and a statistical analyzesr to parse the entry, and then depictors (low level graphic representation) to compose the scene (Coyne and Sproat 2001). The second project is DEM^{2}ONS, a High Level Declarative Modeler for 3D Graphic Applications, designed by Ghassan Kwaiter, Véronique Gaildrat and René Caubet. It allows the user to easily construct 3D scenes in natural way and with a high level of abstraction. Composed by two parts: a modal interface and 3D scene modeler (Kwaiter et al. 1997). The last work is called CAPS or Constraint-based Automatic Placement System, developed by Ken Xu, Kame Stewart and Eugene Fiume (Xu 2002). It makes possible the modeling of big and complex scenarios, using a set of intuitive positioning restrictions that allow the manipulation of several objects simultaneously, while pseudo-physics are used to assure that the positioning is physically stable.

In the literature available for each of these projects, we found that none of them allow the user to make further modifications beyond reorganizing the scebe layout, and none of these researches include methods for self-evolution or simulations over the 3D environment. The input language, in most of them, uses specialized hardware or data, and, in the specific case of WordsEyes, it does not allow to include new concepts, due to its web-based nature. Also the distribution of the

visualization load is not approached. All of them make a local and centralized generation of the visual output. Some of these researches use web-based 3D standards, such as VRML97, while others process a render of a static shot of the scenario, limiting the type of environments, as well as the possibility of generating changes that look natural, avoiding a better evolution for the environment.
Recently, several approaches have been presented for supporting distributed rendering in cluster systems (González Morcillo et al. 2007). This kind of researches are not useful for us, because the output of the rendering process is an image, and there is a server in charge of composing the final image. In (Rangel-Kuoppa et al. 2003) is proposed a 3D rendering system that distributes rendering tasks across a multi-agent platform. The main idea of this system is the rendering of 3D individual objects in different computers. Thus, the agent, that generates the 3D visualization, takes and merges the information from the different buffers to produce a centralized visualization of the whole 3D Virtual Environment. In (Karonis Nicholas et al. 2003) is implemented a remote rendering system using a collaborative component and a high-resolution remote rendering component. But, this system is for near-real time viewing and later use; and the rendering task is distributed inside one cluster only. A system for distributed rendering of large and detailed virtual worlds was described in Chaudhuri et al. (2008). This system is a client-server implementation, where the processing of virtual worlds is distributed among the available servers. This system can be used for generating virtual worlds with fine detail at planetary scales. The capacity of the server limits the number of clients in the system.

VIRTUAL ENVIRONMENT MODELING

We use declarative modeling to create virtual environments. This methodology has three phases: the first one, *Description*, involves the user providing settings, entities and properties to be used to generate a model, as well as certain restrictions to be solved or satisfied. In this phase is defined the interaction language. We defined a language oriented towards composing descriptions of VEs, calle VEDEL (Zaragoza Rios 2006). VEDEL allows to compose and edit descriptions of VEs, directing the user's attention to entities and their properties, providing a structured method for the composition itself.
The second phase in declarative modeling is *Generation*, where the models are composed, validated and then presented. This phase solves the assignation of the properties stated by the user, as well as any kind of conflict that may appear during the generation. This can be achieved by different methods for solving constrained problems, being the Constrain Satisfaction Problem solving the most well-know. We focused on integrating knowledge exploitation on the solution of CSPs, as well as into the whole generation process, making these two tasks more easy and transparent, as well as solving implicit meanings for some concepts.
Finally, the *Insight* phase allows the user to decide which of the proposed solutions is the best one, or to make modifications over the proposed layout, in a way that the solution matches with the user's ideal.
Using this methodology we designed a modeler, which also implements the concept of knowledge exploitation, in the form of a knowledge base, an ontology, which helps in the model creation process by solving term ambiguity and concept value transforming.

VIRTUAL ENVIRONMENT EDITOR

The first part of our proposal corresponds to a Virtual Environment Editor (VEE) module, which take the task of receiving a user's input written in VEDEL definition, and then providing a solution for the user's statements. The input is received by a *lexical-syntactic parser*, which transforms the VEDEL entry into a data structure, organized according to the syntax rules for VEDEL, with entity type as the upper branches, and their properties as the lower leafs. The parsed entry is used then by a *model creator* to generate the model. The process begins with a *zero-state* model, which is a basic skeleton created with default values for the environment and entities, extracted from the knowledge base by an *inference function*, which makes accesses and queries the knowledge base, as well as making the necessary data type conversions between the data obtained and the modeler's needs, which are updated in the model with the values provided by the user for each of the entities' properties. The converted values are set to each entity in the model, with the exception of position and orientation, afterwards the model is sent to the *CSP solving algorithm* to set the layout and solve any spatial conflict that may arise.
The CSP solving algorithm works in two steps: first, it sets all the position values for each request made in the description. The corresponding values are computed using the knowledge base to obtain the ranges for the petition. The first entities to be assigned are those set to a specific place in relation with the environment or any *landmark* (a specially delimited area in the environment), such as `south` or `center`. These entities are called *pivots*, and are used to set the remaining of the entities' positions, using the ranges from the knowledge base and the values from the pivot entities to set their appropiate positions and orientations.
The next step is to find any possible conflict and then solving it. This is conducted through the CSP, which is defined as a tuple $CSP =< V, D, C >$, composed by a set of variables $V = \{X_1, X_2, \ldots X_n\}$, where $X_1, X_2, \ldots X_n$ belong to the set of entities in the environment, and $X_i = \{Position, Orientation, Scale\}$ $\forall X_i \in V$. The domains for each $v_i \in V$ are $D(X_i(P)) =$

$[-\infty,\infty]$, $D(X_i(O)) = [0,2\pi]$. Finally $D(X_i(S)) = [0,\infty]$, and the set of constraints is formed by the following equations:

$$(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2-(r_1-r_2) >= t_1 \quad (1)$$

$$\left(\frac{x}{p}^2\right)+\left(\frac{y}{q}^2\right)-2z <= t_2 \quad (2)$$

$$\left(\frac{x}{dx}^2\right)+\left(\frac{y}{dy}^2\right)+\left(\frac{z}{dz}^2\right) <= t_2 \quad (3)$$

Equation 2 is used to solve collisions. Thresholds t_1, t_2 and t_3 are set in the knowledge base, therefore, the strictness of the verification can be modified. This validation is carried out using *collision marks*, which are set for each entity. These marks consists of spheres that cover the entity's volume, and are retrieved along with the properties for the entity. The marks in an entity are tested against the marks wich belong to others entities, and if there are no collisions, that is, the result from equation 1 is bigger or equal to threshold t_1, for all the collision marks in all the entities, the collision validation has been passed.

If there is a collision, the CSP finds a new position for conflicting entities, and then, proceeds to verify the new position. This is carried out using equations 2 and 3, in combination with a series of *characteristic points* defined for each entity. As well as the collision marks, the values for equation 2 are stored in the knowledge base, and retrieved all the entity's properties.

The test for a position in which an entity makes reference to another is carried out using the characteristic points to evaluate function 2. If at least a number n ($n = 1$ by default) of characteristic points passes the test, this is, the result of equation 2 is less or equal to threshold t_2, the validation test has been passed and the position is valid. Equation 3 is used for absolute positioning in landmarks or the environment itself, or in specially cases such as: `over`, `inside`, and `against`. i.e., where the entities touch each other or are contained inside another. The validation is carried out similiar as equation 2. Any non-complying test leads to further modifications in the position of entities.

The CSP can perform verifications for local minimums or maximums, so finding a solution can be assured. It also records entity's past positions, in order to locate clusters of invalid or conflicting positions, and find another solution away from the cluster. Other conflict solving procedures are: the complete arrangement of the entities in the environment, and rotating conflicting or referenced elements. If the positioning tests are passed, the model is tagged as valid, and send to the final module in the modeler, the *output generator*. This module works over the Model-View Controller outline, sending any valid model obtained by the model creator, which is transformed by the MVC into a suitable output for the visualization and animation process, which is described in the following sections.

VIRTUAL ENVIRONMENT VISUALIZATION

This section describes the creation and visualization of the evolution in the virtual environment. The description used in the creation is received from the VEE, this is interpreted to identify the entities by the virtual environment. We considered a human avatar, which is the most complex entity, because, it can perform different animations that include all the parts of the avatar. we make use of skeletal animation based in H-anim (Group 2009). The purpose is to facilitate real-time management of each part of avatar's skeleton. We generated a real-time distributed visualization of 3D VEs using computational grids and peer-to-peer apporaches. Our architecture has the following components:

- Public knowledge base (KB): It's a set of ontologies based on OWL (Web Ontology Language) with information of VE (3D scenarios, avatars and 3D objects).
- Coordinator nodes: These are special users that manage the consistency of VE.
- User nodes: They are external entities that can perform actions into the VEs.

Figure 1 shows that our architecture is divided into different layers, in order to identify and reduce the dependence of upper and lower layers. The local processing layer is necessary to update the local interface and to maintain a minimum consistency into the VE. That is to say, when a user requests to execute an action, this is evaluated to verify its consistency. For example, all affected entities must exist and the actions must be valid in the entity before applying changes in their state.

When a user wants to perform an action (animation) on an avatar, it automatically generates a request. This request has the following elements: the required avatar, the action to perform, and the time stamp that indicates the order in which actions must be performed. For each avatar is managed a queue of requests. Once the actions have been ordered by the time stamp, these will be executed in a process based on a dynamic time slot (TS). In this slot are placed all the actions that can be executed in 60 milliseconds. The time is based in continuous movements (Maiche Marini 2002), for obtaining a real-time visualization.

Using a TS, it is possible to locally manage the workload. The TS is handled taking into account an increment in local time that increases or decreases the amount of actions to be executed in a process, generating a new state of the VE. Each process returns a result that is sent to the user.

VIRTUAL ENVIRONMENT DISTRIBUTION

One of the main challenges in a shared VE is to efficiently send update messages in a correct way to provide scalability, minimized the delivery delay, and to obtain

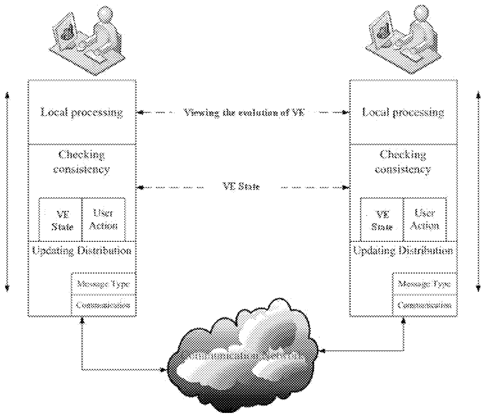
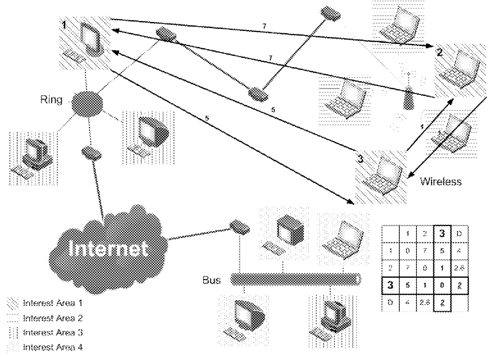


Figure 1: Architecture Peer-To-Peer



	1	2	3	D
1	0	7	5	4
2	7	0	1	2.6
3	5	1	0	2
D	4	2.6	2	

Figure 2: Choice of coordinator

a better reliability of the VE. The update message has as a result, a change of state in a virtual entity (avatar or 3D object). In our implementation, in order to carry out the distribution of messages among users, we take into account the following aspects: the organization of the nodes, the type of communication channel, and the involved communication protocols (see figure 2). These factors affect the delivery time of packages which is very important to ensure a real-time visualization of VEs. To identify the area of interest each avatar belongs, we consider two aspects: the vision area of the avatars and the constraints of vision of the VE. Constraints of vision of VE are given by the objects that restrict the range of vision of the avatars.

When several user exist in the VE, is necessary to check the consistency among all involved users. To do this, consistency messages are generated. These messages contain a description of the area of interest to which the user belongs. Consistency messages are distributed by the layer distribution of updates. This layer also is responsible for selecting a reliable means to deliver them to a responsible coordinator of an area of interest. All the coordinators assigned to an area of interest are in charge of arrangements for setting the correct state of the VE.

The use of areas of interest avoids sending and processing messages where the avatars can not perceive notorious changes or other avatars into the VE. Thus, having identified the areas of interest, it is necessary to choose a coordinator node for each area of interest. This choice is done taking into account the average distance between all nodes, and also considering the available bandwidth, in order to reduce the delay in message delivery and minimize loss of them. In a reliable connexion is possible to know the TTL field. This field contains the number of jumps needed for a packet to reach its destination. We define distance as the number of jumps that a packet has done.

Figure 2 also shows the computation of the average distance between all nodes in an area of interest. For example, node 3 has the shortest distance to all other nodes, so it is taken as the coordinator node.

The new state of VE is computed by taking into account two major agreements: the first one, among all involved users in an area of interest and its coordinator (new state of the area of interest) and the second one, among all involved coordinators in the VE (final state of VE). If the coordinator detects different times, an agreement is made between the nodes for establishing the correct time in the VE.

If a coordinator has a work overload, a new coordinator is selected. In this way, there may be more than one coordinator per area of interest. This ensures that each coordinator manages a balanced amount of users. Our proposal uses only a coordinator node to verify the consistency between users. If at any time the coordinator is not available, the nodes in the area of interest are capable to select a new coordinator among them.

RESULTS AND CONCLUSIONS

To our knowledge, htere are not current projects that offer the possibility of creating virtual scenarios that can also be used to run a simulation, or to let the virtual world to evolve by itself. We also propose a simple method for input the settings, entities and properties to be represented, without the need of special hardware, but adaptable enough to be extended to other input methods. Also, this method provides a structured format, to allow the user to focus in the content rather than the format of the description. Finally, the system is accessible enough to let users to adding new content, as well as modifying the constraints that will dictate the direction of the search for solutions.

We design our modeler to be extensible enough, by using the knowledge base, which allos changing the modeling process, as well as the results obtained. The outputs are also fully adaptable, thanks to the use of the MVC component.

On the downside, there must be an experienced user,

which will provide the first entities and environments, and the knowledge to process the users' requests for such elements. Also, the visualization will depend on the underlaying architecture and its rendering engine.
The graphical representation and evolution of the VE is based on a P2P architecture. The saturation in the client is inherently avoided due to the P2P communication scheme, and the overload of management of the coordinator. Thus, a P2P scheme is a convenient architecture to provide a better scalability for large scale VEs. The animation of the virtual entities is not based on pre-designed animations. The nodes are grouping in different interest areas, this minimizes the amount of messages in the network. The messages from VE are classified into subsets, a node receives messages from just one subset therefore this reduces the overload in each node.
We present as example the creation of two VEs: figure 3 (left) is the VE of a Maya game called "ULLAMA". The second example is a life simulation, figure 3, right, in which the user describes the environment to be represented and its inhabitants. The system, then, simulates everyday-life events.

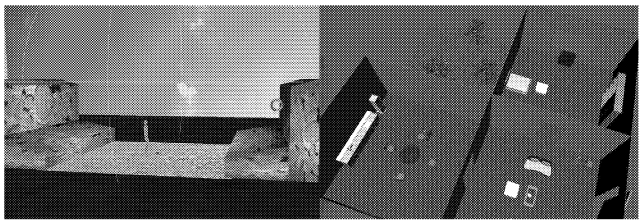


Figure 3: Examples of VEs: Maya Game Ullama and House environment

Acknowlegments

This research is partially supported by CoECyT-Jal Project No. 2008-05-97094, whilst authors J.A. Zaragoza and A.V. Martínez are supported by CONACYT grants 190965 and 47605.

REFERENCES

Chaudhuri S.; Horn D.; Hanrahan P.; and Koltun V., 2008. *Distributed Rendering of Virtual Worlds*. In *Technical Report CSTR 2008-02*. Computer Science Department, Stanford University.

Coyne B. and Sproat R., 2001. *WordsEye: An Automatic Text-to-Scene Conversion System*. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. AT&T Labs Research, 487–496.

Gaildrat V., 2007. *Declarative Modelling of Virtual Environment, Overview of issues and applications*. In *International Conference on Computer Graphics and Artificial Intelligence (3IA), Athènes, Grèce*. Laboratoire XLIM - Université de Limoges, vol. 10, 5–15.

González Morcillo C.; Weiss G.; Vallejo Fernández D.; Jiménez Linares L.; and Fernández Sorribes J.A., 2007. *3D Distributed Rendering and Optimization using Free Software. FLOSS International Conference*.

Group H.A.W., 2009. *H-Anim*. `http://www.h-anim.org/`. Last visited 2009.

Karonis Nicholas T.; Papka Michael E.; Binns J.; Bresnahan J.; Insley Joseph A.; Jones D.; and Link Joseph M., 2003. *High-Resolution Remote Rendering of Large Datasets in a Collaborative Environment*. *Future Gener Comput Syst*, 19, no. 6, 909–917. ISSN 0167-739X.

Kwaiter G.; Gaildrat V.; and Caubet R., 1997. *DEM^{2}ONS: A High Level Declarative Modeler for 3D Graphics Applications*. In *Proceedings of the International Conference on Imaging Science Systems and Technology, CISST'97*. 149–154.

Maiche Marini A., 2002. *Tiempo de reaccion al inicio del movimiento: Un Estudio sobre la Percepcion de Velocidad*. PhD perception, communication and time, Department of Educational Psychology, Universidad Autonoma de Barcelona, Barcelona.

Rangel-Kuoppa R.; Avilés-Cruz C.; and Mould D., 2003. *Distributed 3D Rendering System in a Multi-agent Platform*. In *ENC '03: Proceedings of the 4th Mexican International Conference on Computer Science*. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-1915-6, 168.

Xu K., 2002. *Constraint-based automatic placement for scene composition*. In *In Graphics Interface*. 25–34.

Zaragoza Rios J.A., 2006. *Representation and Exploitation of Knowledge for the Description Phase in Declarative Modeling of Virtual Environments*. Master's thesis, Centro de Investigación y de Estudio Avanzados del Intituto Politécnico Nacional, Unidad Guadalajara, Guadalajara, México.

BOT BUILDING STRATEGIES RELATED TO EDUCATIONAL METHODOLOGY

Clinton Rogers
Daniel Avila
Iren Valova
Department of Computer and Information Science
University of Massachusetts
285 Old Westport Rd., North Dartmouth,
MA 02747
E-mail: ivalova@umassd.edu

ABSTRACT

A popular place to employ artificial intelligence (AI) is the video game industry. The success of a game relies on its ability to challenge the player at a tolerable level, so naturally AI agents offer players the ability to choose the difficulty of the opponent they wish to play against. In addition, AI agents offer extended gameplay options for games that are normally multi-player only. These are only a few of the reasons why students engaged in game design must become aware of good AI algorithms and frameworks to employ in video games. The framework reported in this paper provides a basis to explore many different aspects of game AI and offers an opportunity for the students to develop measures for testing the overall effectiveness of introducing game bots.

INTRODUCTION

The video game industry is one of the most successful forms of entertainment on the market. Artificial intelligence plays a significant role in the growth of the gaming industry, and the growth of game content in general. The main reason for this comes from the need to offer controlled difficultly and competition in games. Due to the high demand for powerful, flexible AI in games, the need for generic AI frameworks is clearly manifested. In order to successfully create AI frameworks in video games, one must draw from many different fields in addition to thorough understanding of the game itself. Behaviors that are natural for humans, such as senses and reasoning, are not present in machines, and must therefore, be implemented. To simulate an intelligent human opponent in a video game requires a solid architecture that draws from the fundamentals of many different fields, including robotics and psychology, in addition to many of the common AI algorithms in existence. The game bot framework developed for this project is related closely to robotics paradigms, and draws from many different AI algorithms to create a power decision making structure.

The game bot framework must overcome similar challenges that the robots of today face. One such challenge is an effective haptic system. Sense of touch is very important in the human decision making process (especially in reactive behaviors). Therefore, game bots should employ an event driven system to simulate the reactive behavior of humans. Another challenge to overcome is simulating vision system. Sight is the primary sense of recognition in video games, as all games are displayed on a screen. Therefore, it is important to employ a vision system that is capable of simulating human vision to make realistic game bots. The third challenge that game bots share with robots is path-planning. Game environments can be even more complex than some real world environments, so having the ability to logically navigate the terrain is one of the most important behaviors to simulating a human opponent in a game. These three challenges, when successfully dealt with, provide the game bot with the information needed to develop a powerful decision making process.

It is important to establish a decision making process. Here are ten common methods, and three other less common, that are used to create a decision making process in game bots (Yildirim & Stene 2008): decisions trees, finite state machines, command hierarchies, manager task assignments, path finding/planning (e.g. A*), terrain analysis influence mapping, formations, flocking, emergent behavior, artificial neural networks (less common), genetic algorithms (less common), fuzzy logic (less common).

Of the thirteen listed, this project implements the following: finite state machines, path finding/terrain analysis, formations, flocking, and emergent behavior (reactive paradigm, described in later section).

INTRODUCTION OF THE GAME

The game that is hosting the game bots in our framework is Hyperion Wars, a game developed in C# utilizing XNA. Players pilot a space ship with the main goal to survive while defeating other opponents.

The game play is defined as a third person shooter, and offers the ability to freely travel the 3D world presented where there is no gravity. The environments are built from rectangular prisms, with the base building block being a cube. Each cube face, known as a Plane, has a flag that enables collision with environment objects. Several different weapons and items exists as power ups, which give the player special abilities when acquired through collision.

The game follows the client server model, where one person (does not have to be a player) hosts the server and the clients who are playing against each other connect to this server to play. Players can move forward, backward, left, and right. Additionally, the players can change the pitch of the ship and roll their ship. Players start with one weapon, but can obtain up to four different kinds. The only way to identify enemies is visual as there are no means to track players through the game HUD.

There are two reasons for choosing this game for our research. First, the framework in which Hyperion Wars was developed in is one of the few which ports to a game console (the Xbox 360). Secondly, Hyperion Wars is the optimal choice for game bots due to its lack of single player content. Several benefits are unveiled when introducing game bots to a multiplayer only game, such as offering a place for introductory players to start, allowing for filler positions in games that are not full, and giving players a reason to continue playing the game even if there is no one online. These benefits allow game design students to accurately develop game bot level difficulties. Additionally, game design students gain the means to develop quantitative measures to determine how much a game has improved with the introduction of game bots (e.g. surveys).

TERMINOLOGY

Game bot or bot, by the definition as used in this paper, is a non-player character that functions autonomously in place of a real player. Other pseudonyms that might be familiar include game agent and NPC (Yildirim & Stene 2008). Specifically, the game bots described henceforth are strictly opponents, they do not fall under the category of neutral or cooperative. Bots described in this paper will preserve fairness to the player by limiting the bot to exactly the resources a player has.

Quaternion: game bot positions are represented using two structures, the Vector3 for the x, y, and z of a point in the world, and the Quaternion for rotation in the world. Both of these structures are built into the XNA framework, making them ideal choices for our project. Quaternions are a rotation about a point, and have their own x, y, z, and rotation value w. The main reason for choosing quaternions is the ability to use spherical linear interpolation to produce smooth rotation animations, which is used in several of the bot behaviors (Shoemake 1985). Additionally, quaternions require less memory to maintain when compared to matrices, and XNA supports more operations with quaternions (Shoemake 1985).

Ray Shooting: specialized form of vision, in which a ray is shot out from a point in a direction with the intent of determining an intersection between the ray and objects [Aronov et al. 2008]. The first object the ray intersects becomes the visible object. For the game bots, our primary targets of intersection are be walls and players.

Game Tick: or tick, by the definition as used in this paper, is an iteration of the game loop inside the game code structure. Considering that a game is constantly running in a loop, then every one complete pass made through this loop is a game tick. For example, a game bot checks its status every game tick to see if it needs to do something new or different from what it is doing.

Position Polling: is used as a form of vision for the game bots. Each game bot will measure its distance from each opponent, and if the opponent is close enough, then they are visible to the game bot.

Field of Vision: what a player or bot can see. Field of vision is the most important piece of information for determining course of action because it is the only information that can be used to seek players.

METHODOLOGY

Research in the field of AI robotics has led to the creation of three distinct paradigms to implement a robot: hierarchical, reactive, and hybrid. Since the aim of a game bot is closely associated with that of a robot, these paradigms are considered as means of implementation of our AI bots.

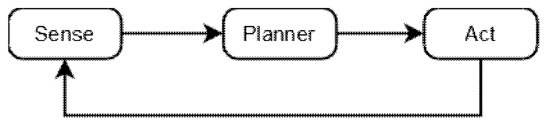


Figure 1 Hierarchical paradigm for bot development

The first paradigm, hierarchical, emphasizes a three stage process before the robot can do anything (Fig.1). The first state is to sense, which, to the bot, is the equivalent of data analysis of what it can "see" and what it "knows" about itself. The second stage is to plan, to decide what it wants to do based on what it sensed. The last state is to act, in which the plan that was created in stage 2 is enacted.

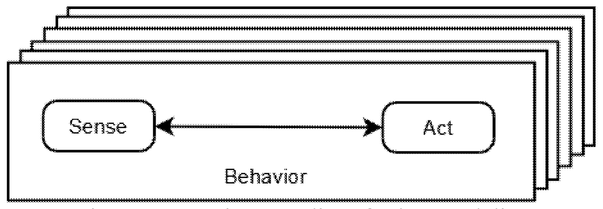


Figure 2 Reactive paradigm for bot modeling

The second paradigm, reactive, operates in a two stage process, i.e. sense then act (Fig.2). For each sensed value, an action is assigned, similar to instincts in an animal. When a cockroach, for example, senses light, it immediately runs away from it. Modeling these instinctive behaviors offers two benefits: the ability to run several behaviors concurrently, and the ability to use an event driven system to efficiently model bot behavior.

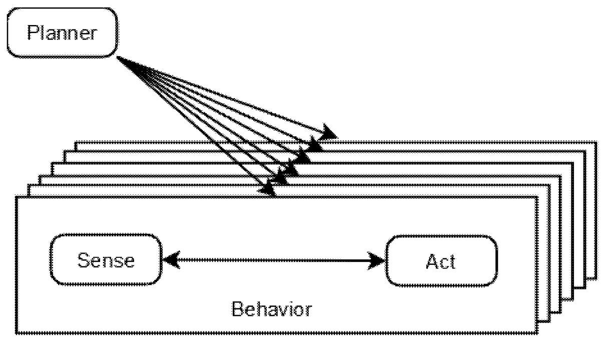


Figure 3 Hybrid paradigm for bot modeling

The final paradigm, the hybrid paradigm, is a combination of the previous two where there is a list of instinctual behaviors just like the reactive, but there is a planner running in parallel to plan out particular tasks (Fig.3). Since the planner is running parallel to the behaviors, the bot would be able to constantly make changes based on the developments in the game world immediately. For this reason, the hybrid paradigm is used in the implementation of the bots for this project.

Four behaviors define the actions a game bot will take in this game: Roam, Chase, Attack, and Boid/Flock.

The default behavior exhibited by the game bots is Roam. Roaming entails the bot fully navigating the environment that it exists in, with the intent of attacking players it sees along the way. Bots in Hyperion Wars cannot use traditional search algorithms such as like A* to find players, because the only means of detecting enemies is though a Field of Vision (FOV) (Rasmussen 2008). Therefore, a different approach to Roam is taken, called Informed Random Roam (IRR). A game bot using IRR will navigate the world by randomly picking destinations based on its current position. Additionally, while the game bot is still in the Roam state, it will search for opponents using its FOV.

The second behavior, Chase, uses a two-fold process. First, the bot must gain sight of the player, and secondly, it will attempt to follow the player. Maintaining the state Chase requires two conditions to be met. Once the bot has established visual contact of an opponent, the bot must maintain at least the correct distance to remain in the Chase state. Should the bot lose sight of its opponent, it will attempt to regain sight by investigating the most recent places the enemy occupied. If the bot does not re-establish visual contact with the opponent, the bot returns to the Roam (default) behavior. Fig.4 shows one way a bot could catch up to an opponent, by interpolating between points occupied by the enemy. The trade-off to having this degree of error, is that the bot can end up running into walls (Fig.4). It should be mentioned that methodologies that would allow the bot to follow a player no matter what (seeing through walls for example) would not preserve fairness to the player, as previously described, and thus are not considered.

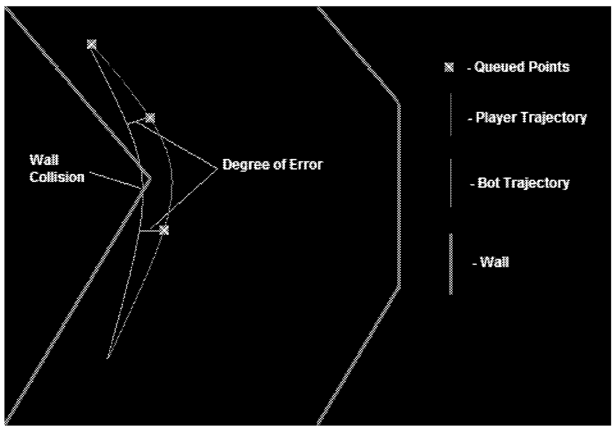
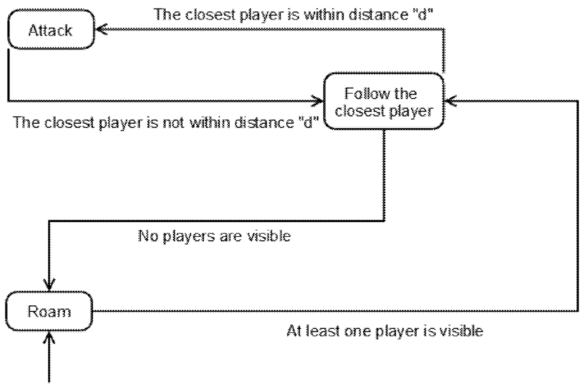
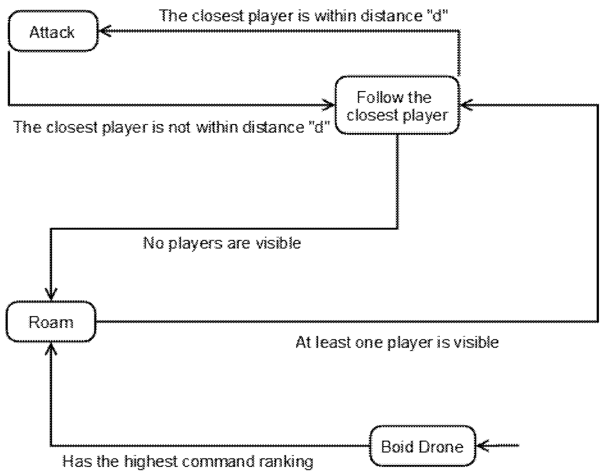


Figure 4 Chasing behavior of the bot

Attack is the follow-up state to Chase, in which the bot begins using whatever means available to damage the opponent it was previously Chasing. The transition from Chase to Attack occurs when the distance between the bot and the pursued is relatively small. Limiting the distance gives opponents who are not watching their backs less of a chance to counter attack and escape, while increasing the accuracy of each attack the bot uses. The methodology used for this paper is virtually identical to the Chase, with one added rule: if the bot is within a certain distance of the player, it may begin shooting. The distance between the bot and the player is calculated between the players (x_p, y_p, z_p), and bots (x_b, y_b, z_b) positions. Distance *d* is then calculated by the basic vector formula: sqrt(($x_p x_b)^2$+ ($y_p y_b)^2$+ ($z_p z_b)^2$). The bot then uses the following predicate to determine if it may attack: If *d* is less than some predetermined distance, then the bot may shoot.


Attack
The closest player is within distance "d"
Follow the closest player
The closest player is not within distance "d"
No players are visible
Roam
At least one player is visible
Figure 5 Association of states with behaviors

The concept of Boids (flocking or packs) is borrowed from nature where certain birds, fish, reptiles and mammals stay together in groups and move in group patterns (Chen, et al 2008). Boid bots move in packs, and their behaviors are coordinated through a leader bot. Leaders are determined by a command value and are established during group formation. Leaders follow the specification of our normal bots, and the rest of the bots in the pack will behave in a drone fashion, essentially imitating the leader. If the leader bot is defeated before the rest of the pack, one of the drone bots is promoted to leader and the pack reassembles around the new leader.


Attack
The closest player is within distance "d"
Follow the closest player
The closest player is not within distance "d"
No players are visible
Roam
At least one player is visible
Boid Drone
Has the highest command ranking
Figure 6 Boid state machine associations

These four behaviors are accessed through a state machine. The simple state machine controls which behaviors are employed and when. Fig.5 illustrates how each state is associated with a behavior, and how one state transitions to another. The boid state machine is identical to the simple state machine, but the behaviors change depending on whether the bot is a leader or drone, as shown in Fig. 6.

The information from the environment that drives the state machines is obtained through FOV.

Three methods to implement FOV are considered. The first is Position Polling, where the bot calculates the vector distance between the bot and all the players in the game. Based on this vector distance, only players that are within a certain distance parameter are then "visible" to the player. Given that there are 'p' players on the screen, it takes 'p' steps per game tick to calculate the vector distance from each player, essentially resulting in asymptotic complexity of O(p); or linear computational complexity.

The second method considered is bounding volumes. Bounding volumes can be viewed as invisible sphere (box, or another shape depending on the method used) that surrounds the object. A collision occurs when the Bounding Volume of one object is intersecting with the Bounding Volume of another object. The asymptotic complexity of collision detection using a convex bounding volume in the best case is O(nlog n), making it impractical when compared with Position Polling (Liu et al 2008).

In addition, typical collision detection methods employ Position Polling while detection is occurring, so extra computational time is required for recalculating the position of the Bounding Volume every game tick while using an algorithm similar to Position Polling (de Pascal 2006). For these reasons, the Bounding Volumes method is dismissed as a potential vision implementation.

The last method considered is that of Ray Shooting. Ray shooting is the concept of shooting invisible lasers in a certain field of vision. If any of them intersect with a player, the player is seen. As soon as the ray hits anything (wall, another bot, player, etc.), the ray terminates, and effectively prevents "seeing through walls" that the other two methods exhibited. Due to the computational complexity of the game environment, many intersection tests would have to be performed, and therefore it is a concern that the game bots would significantly decrease performance as more of them are introduced. For that reason, this method is not utilized.

Though Position Polling suffers from the possibility of seeing players through walls when the vision distance is too large, it offers the best real-time complexity for the purpose of building a game bot. Furthermore, it is possible to limit the "seeing through walls" by taking into consideration only the players in front of the bot, and making sure the bot is facing an open space when possible. Position Polling can be optimized to check a percentage of the opponents rather than the whole list. Since game ticks are quick (in some cases 30-60 per second), we can analyze a small amount of players rather than the whole list to lighten the load on the processing, without affecting the believability of the bots. Another benefit to Position Polling is memory consumption. One float is required per calculation to determine whether or not a bot is within range, making this algorithm efficient memory-wise. The final benefit is that once a game bot locks on a target, the algorithm can ignore all other enemies and continue to measure the distance for only its target. Since bots will be doing other things once they have noticed an enemy (like determining when to attack), reducing the amount of operations during game ticks is essential to fluid game play.

IMPLEMENTATION

The game bots implement Roam using IRR to navigate their way through the environment. Two approaches are considered in creating IRR: roaming by following a predetermined course (waypoints), and roaming by detecting the environment in real time. The resulting navigation system combines the two approaches, and the result is the Modified Waypoint System (MWS).

MWS is a navigation system in which the game bot creates waypoints in real-time, based on the center of nearby cubes (Hyperion Wars maps are made of cubes), and randomly picks one as its destination. When the game bot enters the Roam state, the first action for the game bot is to find and set the closet calculated center of all the cubes as its destination. This operation only happens whenever the bot transitions into the Roam state. Once the game bot has established its cube destination, it travels to that destination, incrementing its position each game tick and undergoes spherical linear interpolation each game tick until the game bot is facing its destination (Shoemake 1985). Once the game bot reaches the destination, it calculates the center of all nearby cubes, excluding the previously occupied cube, and continues moving through the environment. By excluding the previous cube in our destination calculation, game bots avoid getting stuck at the end of hallways, or backtracking to a position they just came from, and are therefore less likely to investigate cubes they have just recently visited.

The MWS offers several benefits over the traditional waypoint system. Game bots can adapt and roam any map they are placed in completely, and the paths they will follow will be similar to the behavior of a human player searching for enemies. The memory required to maintain the Roam behavior is independent of the size of the map, making it ideal for large maps. Calculating destinations is efficient and does not require searching, increasing the efficiency of the program. Lastly, the MWS introduces a great foundation for new game design students to expand and learn from, such as optimizing waypoint calculation or introducing optimized roaming strategies (in other words, not random).

Chase and Attack are identically implemented, except for the projectiles generated during the Attack state. Position Polling is used to determine how close opponents are, and if they fall with the correct distance, the bot will either Chase or Attack the opponent. In order to escape, the player must move out of the bots vision for several game ticks (depending on how long the bot has been tracking the opponent). Attack will revert to Chase and Chase will revert to Roam if the opponent can maintain a position outside of the predefined range.

Boid bots are an extension of the Simple bot. Boid bots have two special properties, that allow them fly in a pack, the offset and leader property. Whenever a flock is formed, one bot is assumed the leader and all others hold a reference to the leader. Flocks consist of two types of bots, the leader bot and flock bots. Leader bots follow the same specification as our simple bots; they roam the map looking for opponents, and will follow or engage enemies that move within a certain distance from the game bot. Flock bots position themselves at an offset from the leader, and will mimic all movement behaviors exhibited by the leader bot. Should a leader bot die before the pack is defeated, one of the flock bots will assume the role of the leader, and the group will reorganize as if it had just been formed. Flock bots will not pursue an enemy unless the leader bot does, but they will attack an enemy that gets close enough to them. An illustration of the game bot is provided in Fig.7.

CONCLUSION

This project features successful implementation of Finite State Machines, Path finding/Terrain Analysis (Way-Points), and Emergent behaviors AI methods in the creation of the game bot decision making process. Future users of the framework have a powerful tool that can be expanded upon to include new behaviors in addition to the already existing ones. Game design instructors can utilize the framework as powerful tool to introduce AI to prospective game developers, who can then easily expand upon the current implementation to learn how to develop effective decision making architectures.

In addition to creating powerful decision making architecture, the bots exhibit real-time navigation system that is not subject to the environment it is introduced into. Using the waypoint and vision system, game bots implemented with this framework are capable of gathering the necessary information to traverse 3D environments and combat players. Additionally, the algorithms that determine navigation are interchangeable, allowing for optimization depending on the environment.

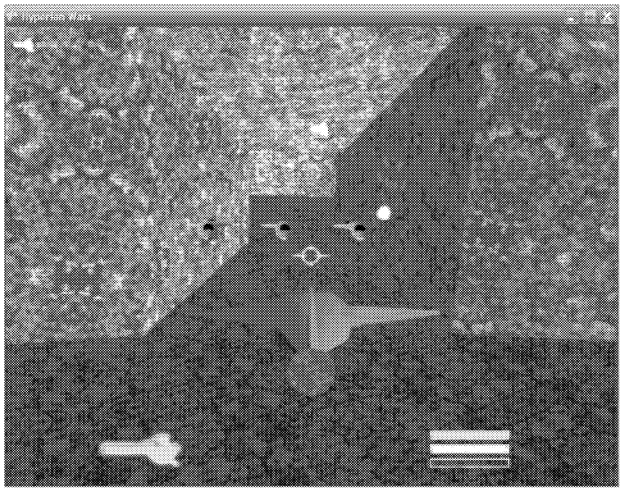


Figure 7 Flocking bot behavior

By using the hybrid paradigm as a baseline for bot construction, we are able to create a flexible game bot framework, which could easily be adapted and extended. Specifically, by building the bot with the intention of having a planner, we left room for the introduction of AI methods, such as genetic algorithms.

The ultimate goal of this project was to develop a framework for developing game AI. By developing a generic framework, we can evaluate the various implementations of traditional AI algorithms, robotic AI algorithms, and game specific AI algorithms through analytical case studies, and develop measures that can be used as standards for creating acceptable AI in video games.

REFERENCES

Arnov, B., deBerg, M., and Gray, C. (2008). Ray shooting and intersection searching amidst fat convex polyhedra in 3-space. *Computational Geometry: Theory and Applications* 41:68-76.

Chen, Y.W., Kobayashi, K., Kawabayashi, H., Huang, X. (2008). Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, 141-148.

Deep Blue (1997). http://www.research.ibm.com/deepblue/

de Pascale, M., Prattichizzo, D. (2006). A Framework for Bounded-Time Collision Detection in Haptic Interactions, VRST, 305-311.

Gibb, J. (2006). Backgammon Info, http://ezinearticles.com/

Liu, R., Zhang, H., Busby, J. (2008). Convex Hull Covering of Polygonal Scenes for Accurate Collision Detection in Games, Graphics Interface Conference, 203-210.

Medler, B. (2008). Views from Atop the Fence: Neutrality in Games, ACM SIGGRAPH symposium on Video games, 81-88.

Medler, B. (2008) Using conflict theory to model complex societal interactions, Conference on Future Play: Research, Play, Share.

Rasmussen, R. (2008). A Game Theory Approach to High-Level Strategic Planning in First Person Shooters, Interactive Entertainment.

Shoemake, K. (1985). Animating Rotation with Quaternion Curves, ACM SIGGRAPH, 245-254.

Yildirim, S., Stene, S. B. (2008). A survey on the need and use of AI in game agents, SpringSim '08: Proceedings of the 2008 Spring simulation multiconference, 124-131.

AUTHOR BIOGRAPHIES

CLINTON ROGERS is in his second year of pursuing Masters degree in Computer Science with University of Massachusetts Dartmouth. He graduated Summa Cum Laude with BS in Computer Science and is now following up on his education. His interests are in the area of robotics, human-computer interaction and intelligent learning systems.

DANIEL AVILA is currently pursuing BS degree in Computer Science with University of Massachusetts Dartmouth. While still at a junior level of his studies, he is involved in research on gaming, learning systems and neural networks. He is currently working on a novel clustering algorithm and exploring unsupervised learning algorithms for neural networks.

IREN VALOVA is currently an Associate Professor with the Computer Science Department, University of Massachusetts Dartmouth. She has extensive experience in the field of neural networks, learning algorithms, data mining and brain functionality modeling. Her research on learning systems led to work on gaming and bots. She is very active in pursuing development of game design education methodologies. Currently, she is involved in the design of interdisciplinary minor on game development following increasing student demand. Dr. Valova has authored more than 100 refereed publications in journals and conferences.

MODELLING AND GAMES DESIGN

A PROPOSITION OF PARTICLE SYSTEM-BASED TECHNIQUE FOR AUTOMATED TERRAIN SURFACE MODELING

Korneliusz Warszawski
Faculty of Electrical Engineering,
Computer Science and Telecommunications
University of Zielona Gora
ul. Podgorna 50
65-246 Zielona Gora, Poland
E-mail: k.warszawski@weit.uz.zgora.pl

Slawomir Nikiel
Institute of Control
and Computation Engineering
University of Zielona Gora
ul. Podgorna 50
65-246 Zielona Gora, Poland
E-mail: s.nikiel@issi.uz.zgora.pl

KEYWORDS
Terrain modeling, Particle systems, Virtual environments.

ABSTRACT

Automated methods for landscape modeling are the useful and time-efficient alternatives for manual terrain formation done by developers and virtual world builders. Considering efficient terrain generation we can use any combination of fractal-based algorithms, i.e., midpoint displacement, fault formation, iterated function systems. We can also use particle systems, presented in this article. We show an idea of a particle system that can be used in terrain modeling. We explain the system, its parameters and their influence on the final product, the virtual terrain.

INTRODUCTION

To achievement the realistic structure of landscape in open world games it is necessary for developers, artists and level designers to spend a lot of time manually forming polygon-nets. Alternatively, appealing visual level of terrain models can be obtained much faster by automated techniques. Several applications of procedural techniques have been used in edutainment systems with elements of virtual reality for military and civilian training simulations and as an element of environment modeling in cinematography and modern game development (Rickel and Johnson 1999; Bonk 2005).
Using particle systems in terrain surface modeling may become an efficient alternative for currently used methods. In addition, the proposed technique makes possible generation of complex height-field data, similar to mountains or island-like forms (Warszawski et al. 2008; Warszawski 2009).

RELATED WORKS

Particle systems were originally developed for computer graphics (Reeves 1983) as a fast method for real-time modeling of objects with irregular, dynamically changing and impossibly/hard to define surfaces like clouds, fire or explosion. It can be also used for modeling of vegetation like grass or cereals (Reeves and Blau 1985; Lander 1998). A proposition of parallel implementation was forwarded by (Sims 1990) and adopted by (McAllister 2000) and (Kolb et al. 2004) for their descriptions of particle systems.
The foundations of mostly generation techniques of terrain modeling are based on self-similarity fractal methods. The most traditional approach uses Madelbrot's Midpoint Displacement Method (MDM) and was thoroughly described by (Foumier et al. 1982). An opening height-field grid for the MDM algorithm has 2x2 resolution and in recursive subdivisions, the method increases its size. The landscape precision increases by calculation of height values of newly generated height-field nodes as averaged height of the neighbor points displaced by a random offset. Subdivision part of the algorithm attained several modifications except classical Square Subdivision. Mandelbrot and Musgrave proposed Hexagon Subdivision and Miller presents both Diamond-Square and Square-Square subdivisions algorithms. All proposed modifications give an alternative models for selection of neighbor points, but the idea of algorithm remains unchanged (Musgrave et al. 1989; Koh and Hearn 1992; Sala et al. 2002).
The technique for simulating of desert structures effected by wind and sand motions was described by (Benes and Roa 2004). The method used particles related to some amount of sand reloated by wind (saltation), rolled down the hill (creep) and moved indefinitely (suspension). Considering the saltation process, if one hill is reduced then the other one must be improved by the material transported between them. The creep process is similar to particle deposition and is used to smoothen the terrain.
Another different approach to terrain modelling uses Genetic Terrain Programming (Frade et al. 2009). The method primitives generated over height-field are: semisphere, step (fault), spectral synthesis, gaussian bell, plane or uniform random values. Starting population is chosen randomly from this primitives. In the reproduction phase, creation of new indiduals follows from one (mutation) or two (crossover and mutation) parents chosen from existing population. Authors noticed that the performance of their method can compete with those of classical methods only for low resolution heigh-fields.

TERRAIN REPRESENTATION

The data structure of memory stored landscape is a modified multilayered height-field organized as a two-dimensional array.

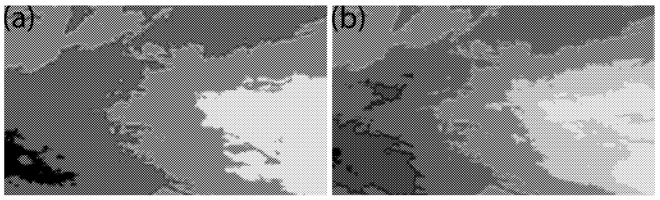
Each cell stores two pieces of information of a coordination point defined by rows and columns of the array. The first property is its altitude and the second is its vulnerability to the erosion process.
Information about topology and geological infrastructure can be imported from real data, but the terrain representation enables fast hardness map generation with help of one of classic height-field modeling technique, i.e., midpoint displacement, fault formation.
Unlike (Benes and Forsbach 2001) representation we treat the hardness property as a product of all elements that have influence for terrain materials at a given coordinates. When the hardness map is generated, the next step is to normalize all hardness values with (1) common normalization equation to the interval [0.0; 1.0].

$$H = (h - \min) / (\max - \min) \qquad (1)$$

Where (H) is a normalized value, (h) is the initial non-normalized data, (min) and (max) are respectively the minimal and maximal values of the entire unnormalized hardness map.
After normalization process we define the class of hardness for a current terrain layer. To do that, we limit all hardness values to the total number of materials which can exist simultaneously. Afterwards, we recalculate all values using the following equation.

$$C = \lfloor H * k \rfloor \qquad (2)$$

Where (C) is a calculated hardness, (H) is an initial normalized hardness value and (k) is the total number of materials which we called the class of the hardness map.



Figures 1: Sample of hardness map
(a) 3rd class, (b) 5th class

THE PARTICLE SYSTEM

The system we propose, is similar to the classical approach (Reeves 1983) that uses emitters to distribute the particles and to control their quality in a virtual environment. In our simulation we select main attributes for the emitter as: its position and orientation and size of the emission window, that determine the area where a new particle can be inserted into the system environment. The second element of our particle system is a collection of parameterized particles. Each particle has defined size, current position, directional and rotation angles, linear and rotation velocities.

TERRAIN MODELING

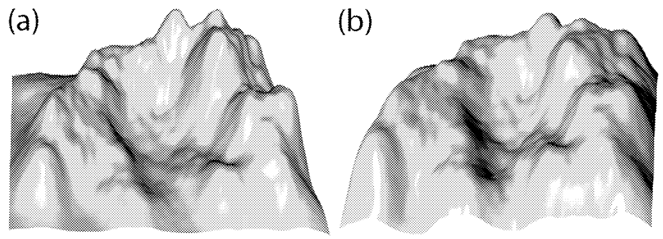
When data structure and particle system are defined, we can used them to model landscapes. We start with proper initialization of the entire system. At the beginning we must set the position for the emitter (or a set of the emitters). For a single emitter it is best to set its position over and in some distance to the central point of the multilayered height-field data. Next, it is important to set the size of the emission window. In general the whole height-field can be effected by algorithm with wide emission window. After setting up the emitters we choose parameters for particles in a collection, i.e., size of the particle define power of its influence for terrain modifications. Range of particle displacement (υ) in virtual space is it velocity. Direction of that displacement is based on both spherical angles (α) and (β).

$$\begin{aligned} x_{i+1} &= x_i + \upsilon * \sin(\alpha) * \sin(\beta) \\ y_{i+1} &= y_i + \upsilon * \cos(\beta) \\ z_{i+1} &= z_i + \upsilon * \sin(\alpha) * \cos(\beta) \end{aligned} \qquad (3)$$

When particle in its journey (3) from emitter window collides with the terrain surface it sets the central point of altitude modification on the related height-field. In that point, all modifications have maximal strength and are getting weaker with increasing distance from the collision point. When modification factor is assigned the negative value it means that the collided particle has no effect on the calculated point of the height-field. The modification factor is calculated by (4) equation for all height-field cells inside circle defined by the collision point and the size of the particle.

$$\Delta y_{x,z} = \sqrt{\lambda^2 - ((x_i - x_o)^2 + (z_i - z_o)^2)} \qquad (4)$$

Where ($\Delta y_{x,z}$) is modification factor of height value at calculating coordinate, (λ) is the radius of the influence zone (size) of a current particle, (x_o, z_o) are coordinates of collision point, (x_i, z_i) are currently calculated coordinate.
The size of the particle (λ) can be also treated as a surface regularity factor. With increasing this value the target landscape will be more regular.

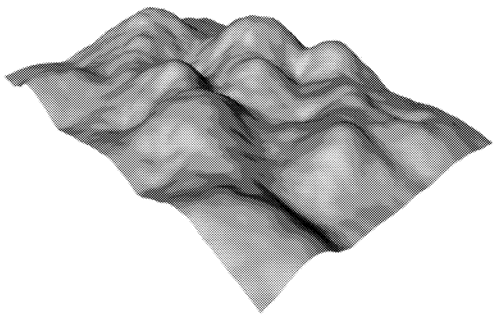


Figures 2: Regularity factor of the surface
(a) particle size at 10, (b) particle size at 15

The final step is to apply the modification factor and hardness value to the height-field. The result (5) as the altitude value is the product of these values.

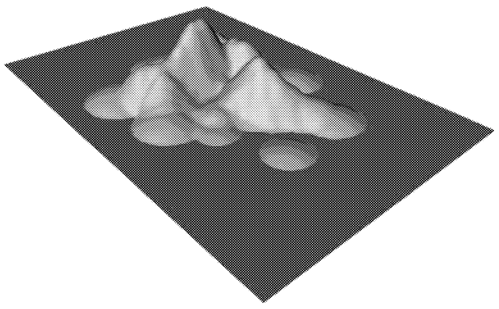
$$y^*_{x,z} = y_{x,z} + (H_{x,z} * \Delta y_{x,z}) \qquad (5)$$

Where ($y^*_{x,z}$) is the modified height value, ($y_{x,z}$) is the height value before modification, ($H_{x,z}$) is the hardness value and ($\Delta y_{x,z}$) is the modification factor.


Figures 3: Example of generated mountains

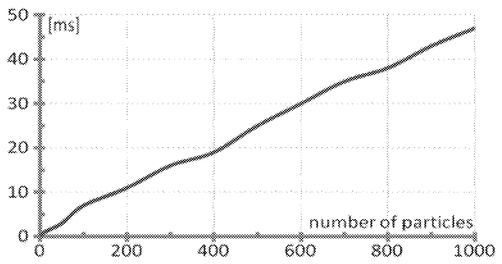
THE ISLAND MODIFICATION

To adapt the technique to model island-like structures it is important to restrict the area of height-field that can be enabled for modifications. We can do this by limiting the size of emission window. If the size becomes smaller less height-field cells can be modified by downfall of the particles. For our simulation we calculate that the size of emission window cannot exceeded 1/3 of the height-field area.


Figures 4: Example of generated island

CONCLUSIONS

The overall performance of the presented technique is almost independent from the height-field resolution. It depends mostly on the number of used particles (in a collection) and decreases proportional to their number.



Figures 5: Average performance graph calculated for height-field resolutions: 80x60, 160x120, 320x240 and 640x480

The method is open for parameterization, thus making it possible to adaptation to the satisfactory level of the modeling terrain surface characteristics.

Further research will be focused on the refinement of the particle-based algorithm to achieve the most realistic and fully automated method for non-spherical topology landscape modeling, like caves, canyons or cliffs.

REFERENCES

Benes, B. and R. Forsbach. 2001. "Layered data representation for visual simulation of terrain erosion." In *Proceedings of the 2001 Spring Conference on Computer Graphics*, 80-85.

Benes, B. and T. Roa. 2004. "Simulating desert scenery." In *Proceedings of the 2004 WSCG International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 17-22.

Bonk, C.J. and V.P. Dennen. 2005. "Massive multiplayer online gaming: A research framework for military training and education." Technical Report for Office of the Under Secretary of Defense for Personnel and Readiness.

Fournier, A., D. Fussell, and L. Carpenter. 1982. "Computer rendering of stochastic models." *Communications of the ACM*, Vol.25, No.6 (Jun), 371-384.

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

Koh, E-K. and D.D. Hearn. 1992. "Fast generation and surface structuring methods for terrain and other natural phenomena." *Computer Graphics Forum*, Vol.11, No.3, 169-180.

Kolb, A., L. Latta, and C. Rezk-Salama. 2004. "Hardware-based simulation and collision detection for large particle systems." In *Proceedings of the 2004 ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, 123-131.

Lander, J. 1998. "The ocean spray in your face." *Game Developer Magazine*, No.6 (Jun), 13-19.

McAllister, D.K. 2000. "The design of an API for particle systems." Technical Report, Department of Computer Science, University of North Carolina at Chapel Hill.

Musgrave, F.K., C.E. Kolb, and R.S. Mace. "The synthesis and rendering of eroded fractal terrains." *ACM SIGGRAPH Computer Graphics*, Vol.23, No.3, 41-50.

Reeves W.T. 1983. "Particle Systems - A technique for modelling a class of fuzzy objects." *Computer Graphics*, Vol.17, No.3, 359-375.

Reeves W.T., R. Blau. 1985. "Approximate and probabilistic algorithms for shading and rendering structured particle systems." *ACM SIGGRAPH Computer Graphics*, Vol.19, No.3, 313-322.

Rickel, J. and W.L. Johnson. 1999, "Virtual humans for team training in virtual reality." In *Proceedings of the 1999 World Conference on AI in Education*, 578-585.

Sala, N., S. Metzeltin, and M. Sala. 2002. "Applications of mathematics in the real world: Territory and landscape." In *Proceedings of the 2002 International Conference the Humanistic Renaissance in Mathematics Education*, 326-333.

Sims, K. 1990. "Particle animation and rendering using data parallel computation." In *Proceedings of the 1990 annual Conference on Computer Graphics and Interactive Techniques*, 405-413.

Warszawski, K., S. Nikiel, and T. Zawadzki. 2008. "Particle system for generation of terrain structures." *Przeglad Telekomunikacyjny i Wiadomosci Telekomunikacyjne*, No.6 (Jun), 860-862.

Warszawski, K. 2009. "Ground from smoke: Using particle systems for terrain modeling in C#." *Game Developer Magazine*, Vol.16, No.3 (Mar), 15-21.

CREATION OF VIRTUAL WORLDS THROUGH KNOWLEDGE-ASSISTED MODELING

Jaime Zaragoza, Véronique Gaildrat, and Félix Ramos
Centro de Investigación y de Estudios Avanzados, Unidad Guadalajara, México
Institut de Recherche en Informatique de Toulouse, France
email: {jzaragoz, framos}@gdl.cinvestav.mx, veronique.gaildrat@irit.fr

KEYWORDS
Game Design, Game Environment Creation, Knowledge Bases, Distributed Systems

ABSTRACT

The creation of virtual worlds is a complex task. Even experienced user may find difficult to create virtual representations for different needs (simulation of the real world, recreations of ancient worlds, fantasy worlds, teaching, etc). Through declarative modeling, an user can create a virtual scenario by simply stating some properties for the environment and the entities. In this paper, we present a novel approach for declarative modeling, whose main contribution is to use knowledge about the entities that conform the environment, in order to assist the creation and validation of the virtual world. This knowledge includes the necessary data to reduce in great measure the amount of processing needed to create the environment and the knowledge needed to allow the evolution of the environment in a dynamic scene. The main difference with other research is that the virtual world can be either static or dynamic, in the sense that the world can evolve.

INTRODUCTION

Virtual Reality has been used as a method for simulating entities in the real world in different fields of human expertise, such as medicine, construction, entertainment, and many others. This technology allows creating almost any kind of scenario or world, where a dynamic scene can be perfomed. Different approaches have been proposed to design and animate such virtual worlds. However, most of the time, these worlds are created by a full multidisciplinary staff composed of many artists, modelers and engineers, taking from a few weeks to years to complete a successful visualization of the desired environment. Even more, specialized tools are requiered, therefore the team needs special training and many practice hours to create an outcome of professional quality.

The problem can be split in two subproblems. The first one lies in creating the environment; the second one is describing the scene, that is how characters must evolve in the environment. In this paper we deal with the creation of the environment. The second subproblem was already tackled by us in (Ramos et al. 2002).

A method, which permits the creation of virtual scenarios, both simple and complex, is currently a topic of research. Declarative Modeling is one approach to reach this objective. In this research, Declarative Modeling must be understood as a methodology that creates a virtual scenario by means of declaring how we expect entities in the scenario must be arranged. A system based on Declarative Modeling must take as input this description and find at least one solution which satisfies the user requirements.

We propose the using knowledge about properties, objects and other entities requested by users to help the validation process of possible solutions found by the modeling system, resulting in a model is useful for manage all kind of animations required by the evolution of a scene. This research is part of the GeDa-3D project (Hugo et al. 2004), a distributed multi-agent architecture for 3D. GeDA-3D objective is to offer a complete tool to any unexperienced users.

DECLARATIVE MODELING

Declarative modeling is a recursive process which finds a solution to the model properties stated by the user. Applied to the creation of a scenario, the declarative modeling is a continuous process in which feedback coming from previously obtained solutions is used recursively, until the user is satisfied with the outcome solution.

The method is composed by three phases (Gaildrat 2007): *Description*, where the system receives the user's statements, written in a custom-tailored definition language for the modeling application; *Generation*, where the model is created using different methods to assure semantic consistency, the logic, and the positioning of the entities. One of the most common methods is solving a Constraint Satisfaction Problem or CSP.

We can define a Constraint Satisfaction Problem as a set of variables $X = \{X_1, X_2, \ldots X_n\}$, a domain D which indicates the possible values for each variable, and a set of constraints $C = \{C_1, C_2, \ldots C_n\}$ which specifies a subset of variables and the possible values for each of these variables. When each variable has been assigned

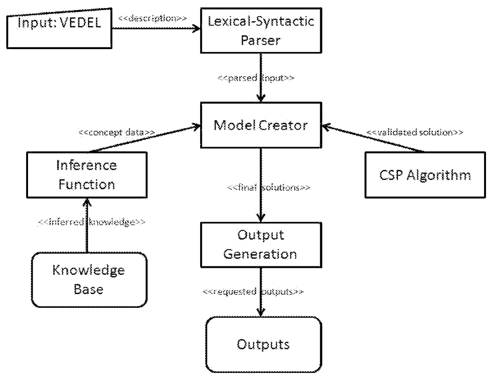


Figure 1: Modeler Architecture.

with a value from its domain, such that no constraint is violated, a solution has been found. For a given CSP, several solutions can exist, thus the CSP algorithm can provide several solutions (Russell and Norvig 2003).

Finally, the *Insight* phase involves presenting to the user solutions obtained, where the model can be modified.

A VIRTUAL ENVIRONMENT MODELER BASED ON KNOWLEDGE EXPLOITATION

The user expresses what should be placed in the virtual scenario, thus the system has the indications of the characteristics and positions of each element, which can be applied within certain fuzzy thresholds. For example, "`right`" will be always a specific area defined in relation to a referenced entity, whitout taking in consideration its orientation, size or position; and can be defined within a data structure that provides the parameters for obtaining such area. Such data structure can come in the form of a database. However, the necessary information should be provided with the inclusion of the relations between the concepts in evaluation, a more suitable solution is the knowledge bases. A knowledge base not only contains information of the concepts for any given domain, but also the relations between these concepts. It permits deriving new knowledge from data already present, expanding the scope of the knowledge base. The action of extrapolating new information from current knowledge is called inference, and it is a useful characteristic for validating concepts and properties, since the user can begin stating simple characteristics, which can be combined to infer complex knowledge.

With this idea, we design a declarative modeler aimed create virtual worlds from simple text descriptions, using knowledge base exploitation as the base of the modeling process. The modeler is composed by five modules, Lexical-Syntactic Parser, Model Creator, Inference Function, CSP Algorithm and Output Generator (figure 1).

The Lexical-Syntactic Parser receives the description written in a custom-defined language called VEDEL or Virtual Environment Description Language (Zaragoza-Rios 2006). VEDEL is like natural language, completely oriented to lead the description process, proving a structure to organize the idea for describing virtual worlds. VEDEL creates scenarios incrementally, by just adding objects or modification in specific sections, in order to extend the description. Some examples are presented in the `Current Work` section.

The Lexical-Syntactic Parser is basically a state machine, which extracts tokens from the description, and uses them to form a hierarchical data structure representing the information stated in the description. The hierarchy is organized with entity types as the upper branches, and the properties as their leaves. This module only verifies the correct composition of the description, and searches for non-valid characters, leaving semantics validation for the Model Creation module.

The Model Creator receives a parsed entry from the Lexical-Syntactic Parser, and proceeds to create a model, working in incremental steps, starting with a basic model with default values assigned to elements of the virtual environment, and then assigning the values requested by the user. The model creation finishes when all properties have been assign and these properties do not violate constraints.

First, the modeler obtains information about the environment. Any request for information is handled through the Inference Function component, which accesses directly to the knowledge base. The inference function obtains information, formats the data for the modeling tasks and validates requests made in the description.

The information gathered from the knowledge base about the environment is used to construct virtual world's rules and assign data type values to its properties. If the environment has some special constructions or elements, the modeler creates the necessary entries in the model to satisfy these indications. Landmarks such as walls, doors, specific areas or furniture in the environment, are all created this way. Landmarks correspond to implicit information about the environment and have not explicit visual representations, the second sort of descriptions corresponds to objects that must be instantiated, placed, and represented as individual elements in the environment.

Once the environment has been set, the modeler continues with the entities that will dwell the scenario. Each of these entities is created using basic representations instantiated with default values from the knowledge base. Such representations are called *avatars*. Each request is validated first through the avatar, to verify wheter the entity can represent the property requested. Then, the Inference Function validates the values to be assigned are correct, or correspond to the property in question. Any necessary conversion is carried out through the knowledge base.

Once all entities have been created and their properties instantiated with the values specified in the description, the modeler proceeds to position all of them in the correct location, according to the description statements. Entities are first placed using a default position stored in the knowledge base. Parameters corresponding to the position concept are obtained from the knowledge base, and then instantiated with entities' values. The referenced entity can be any other entity in the environment, the environment itself, or a landmark. Concept parameters can be defined with fuzzy parameters, achieving certain randomness to allow the generation of different models through the process.

After every entity has been positioned, the modeler sends the model's current state to the CSP component. This component validates that there are no collisions between entities and that the position of each of them corresponds to the statements in the description. To accomplish these tasks, each entity has a series of collision tags, as well as characteristic point marks. These specialized tags are stored in the knowledge base, and are instantiated when the entity is created. When the modeler changes the entity's position, these tags are also updated.

The collision tags are defined as a series of spheres that wrap the entity's geometry (figure 2), and are used as the primary constraint to be satisfied by the CSP Algorithm. To carry out the collision verification process, the system first verifies if the Euclidean distance between any given pair of entities is smaller than the longest diagonal corresponding to a box formed by the entities' dimensions. If true, the distance equation for two spheres is evaluated for all the possible pairs of collision tags between the two entities in conflict. This must be satisfied for any pair of collision tags corresponding to different entities, $C_1 = (x_1, y_1, z_1, r_1)$ and $C_2 = (x_2, y_2, z_2, r_2)$, the following relations must be kept:

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 - (r_1 - r_2) >= t \quad (1)$$

If some pair fails this evaluation, that is, the result of the evaluation is less than a threshold ($t \in \Re, 1 > t > 0$) set by the system administrator, a collision is detected and the model must be modified. Some considerations are verified prior to any change: If an entity is a "*pivot*" (it is fixed to an absolute position, i.e. *north* or *east*), the size of the entities, and the relationships between them. First, the relations between elements is explored, since some possible positioning concepts require that two elements come in contact, such as "`against`", "`inside`" or "`over`", thereby in these cases the collision will be dismissed and other validations will take place. Otherwise, if an entity in conflict makes reference to another, this must be moved first. If the model configuration cannot be validated, the referenced entity will be then moved. Otherwise, if both entities make reference to a third one, the CSP Algorithm relocates both entities in

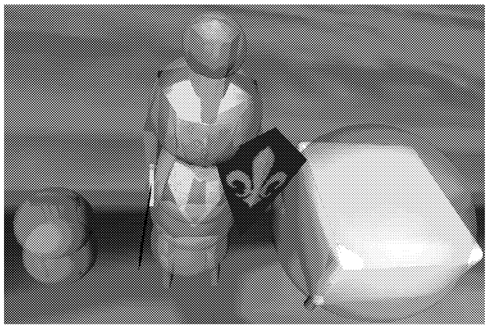


Figure 2: Collision tags examples.

a new valid position, making use of a technique similar to one the used with L-Systems (Prusinkiewicz and Lindenmayer 1990). Finally the size is considered to decide which entity should be moved first. Smaller entities have a better chance to be moved into a valid position than large elements, therefore the system will firsly move the smaller entities in conflict. Pivot entities are moved at last, and only if the current model requires such changes to obtain a solution.

Collision verification is conducted until no more conflicts are found. If the current model falls into a local minimum, or cannot find a solution, the system restores a partial previous solution, in which less conflicts were found; then it proceeds to create new solutions. If the restored solution is the initial model sent by the Model Creator, the system verifies wheather any possible entity has been repositioned, and restarts. If there are no more entities to move, the system informs that cannot find a solution.

Once it was verified that there are no collisions among entities, positioning validation is carried out, using characteristic points marks, which indicate the most prominent points in the entity's geometry. Positioning validation is conducted over all entities, in order to assure that the current position satisfy description statements. In order to successfully locate an entity, the following validation must be satisfied: at least one of its characteristic points must be located inside the volume of an ellipsoid or a paraboloid, instantiated with the parameters corresponding to the referenced entity's data, and the concept values. That is, for any given entity, at least one of its characteristic point ($G_i = (x, y, z)$) is in the surface or inside a validation volume $E = (dx, dy, dz)$, for ellipsoids (3), or $P = (p, q)$, for a paraboloid (2), instantiated with the values from the referenced entity. We can represent this as:

$$\left(\frac{x}{p}^2\right) + \left(\frac{y}{q}^2\right) - 2z <= 0 \qquad (2)$$

$$\left(\frac{x}{dx}^2\right) + \left(\frac{y}{dy}^2\right) + \left(\frac{z}{dz}^2\right) - 1 <= 0 \qquad (3)$$

The parameters for both equations are stored in the

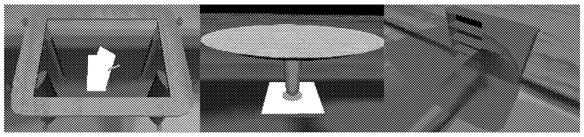


Figure 3: Characteristic points and positioning volumes examples.

knowledge base, either as part of the positioning concept, or as information for the entity in reference, which can also include the quantity or the list of characteristic points that need to be inside the volume in order to validate the position. Relative positions are always validated through paraboloid volumes, for which the parameters p and q correspond to the referenced entity's size values, and z is a value set by the system administrator.

Spatial relations such as "`against`", "`inside`" or "`over`", are validated using ellipsoidal volumes, which as only one parameter smaller than 1, and located over the entity's surfaces. For positions that require a volume inside an entity, or a location in a given area such as landmarks, the ellipsoidal volume is instantiated with the entity or landmark values.

If the positioning validation for an entity fails, a new position for this entity is computed. Before the new position is set, the system queries a list of previous positions for that particular entity. If the neighborhood of these previous positions forms a cluster, then a new position outside that cluster is set, and the validation is carried out again. If the new position is not valid, or it cannot be set outside the cluster, the system cannot find a new position for the entity and returns to a previous model state. As with collision verification, if the previous model is the starting set, the description requested cannot be solved.

Each entity in the model stores the list of elements which share some relation. In that way, each entity can follow the movements done in the model, and modify its own parameters by consequence

Once each entity has passed both validations, the model is sent to the final component, the Output Generator. This component uses a Model-View Controller (MVC) method to generate the necessary output data structures and files, then the underlying architecture can perform the actions necessary to create the visualization of the virtual environment, and conduct the process that will activate the simulation. This output employs templates that receive the model's data structure, and through the MVC are instantiated with the values obtained through the modeling process. This method allows the easy modification of the output parameters, to permit users to add or modify the information sent to the underlying architecture, which can be also a 3D viewer. The outputs and their types, are set in the knowledge base.

CURRENT WORK

We have developed a prototype of the modeler in the Java language, to allow multi-platform capabilities. The knowledge base has been defined using the Protégé framework, and was created using the OWL Standard (McGuinness and van Harmelen 2004), selected due to its extensibility over the Internet. Next, we present some examples obtained using the lastest build of the modeler, which uses the X3D standard and a X3D-compliant viewer.

Figures 4 and 5 show some examples obtained using the current modeler and its X3D output.

CONCLUSIONS

We can make reference to three researches that deal with declarative modeling of virtual scenarios: WordsEyes, citepref:COYNE01, developed at the AT&T Laboratories, is the closest project to ours. It is an automatic text-to-scene converter, which works with part-of-the-speech markers and statistical analyzers. The system is web-based, so the user just has to enter to project's website, write a short description, and submit it to obtain a static visualization of the created scenario. The second related research is project DEM^{2}ONS (Le Roux et al. 2000), which is a multimodal system composed by a modal interface and a 3D modeler. It allows simple interactions, that is, the modification of the non-dynamic elements presented in the scenario. Finally, CAPS (Xu 2002) is a constraint-based system, oriented to solve the placement of objects inside a scenario. It uses pseudo-physics and a specialized input language to position up

```
[ENV]
 LivingRoom.
[/ENV]
[ACTOR]
 ManSuit John.
 YoungWoman Sarah.
[/ACTOR]
[OBJECT]
[/OBJECT]
```

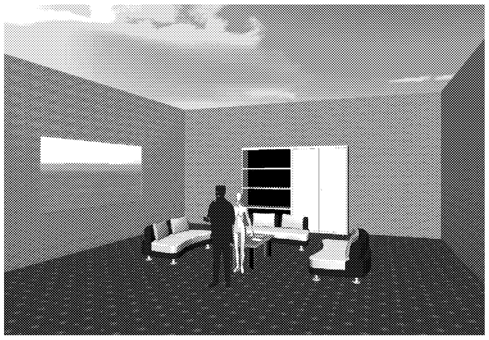


Figure 4: VEE Example 1.

```
[ENV]
 house.
[/ENV]
[ACTOR]
 Knight, anywhere Kitchen.
 YoungWoman, anywhere Garden.
 woman, anywhere Livingroom
[/ACTOR]
[OBJECT]
 Chair, front woman0.
 Puff, front bed0.
[/OBJECT]
```

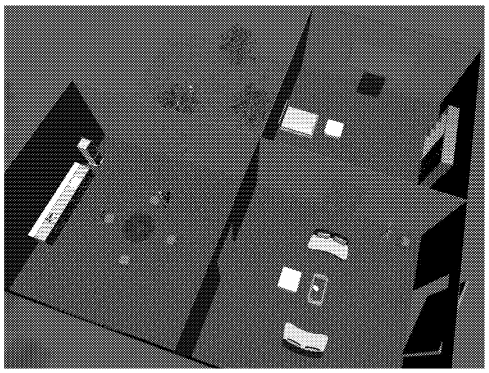


Figure 5: VEE Example 2.

to hundreds of elements in a scenario. To our knowledge, these researches do not provide further methods to allow scenario self-evolution, nor provide any information that allow to simulate a scene on the model.

In this article, we describe our novel approach for declarative modeling using knowledge bases to assist the process of model generation. We have implemented a prototype to test our proposal. So far, we have created different scenarios with low generation times using several objects. Also we have shown how easy is the modification of parameters and concepts, achieving significantly differences between solutions obtained through several iterations of the modeler. The result shows the transparency in the modeling process. We use the GeDA-3D architecture to represent a dynamic scene, whose scenario was created using our declarative modeling. Such dynamic properties are not present in any of the referenced researches.

The main drawback of this research is that knowledge bases must contain all concepts, as well as a 3D object database. However once this has been fulfilled the creation of 3D scenarios is available to final users, achieving the main objective of this research.

Future work includes the development of a user-friendly tool to manage knowledge and the 3D object database extension, as well as dealing with specific cases that may need special modeling methods.

Acknowledgments

This research is partially supported by CoECyT-Jal project No. 2008-05-97094. The authors would also like to thank the National Council on Science and Technology, CONACyT, for providing the PhD Scholarship number 1910965.

REFERENCES

Gaildrat V., 2007. *Declarative Modelling of Virtual Environment, Overview of issues and applications.* In *International Conference on Computer Graphics and Artificial Intelligence (3IA), Athènes, Grèce.* Laboratoire XLIM - Université de Limoges, vol. 10, 5–15.

Hugo P.; Zúñiga F.; and Ramos F., 2004. *A Platform to Design and Run Dynamic Virtual Environment. International Conference on Cyber Worlds, Tokyo Japan*, 18–20.

Le Roux O.; Gaildrat V.; and Caubet R., 2000. *Design of a new constraints solvers for 3D declarative modeling.* In *International Conference on Computer Graphics and Artificial Intelligence (3IA), Limoges, 03/05/200-04/05/200.* 75–87.

McGuinness D.L. and van Harmelen F., 2004. *OWL Web Ontology Language Overview.* W3c recommendation, World Wide Web Consortium. Http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Prusinkiewicz P. and Lindenmayer A., 1990. *The algorithmic beauty of plants.* Springer-Verlag New York, Inc.

Ramos F.; Zúñiga F.; and Piza H.I., 2002. *A 3D-Space Platform for Distributed Applications Management. International Symposium and School on Advanced Distributed Systems 2002 Guadalajara, Jal, México.*

Russell S.J. and Norvig P., 2003. *Artificial Intelligence: A Modern Approach.* Pearson Education.

Xu K., 2002. *Constraint-based automatic placement for scene composition.* In *In Graphics Interface.* 25–34.

Zaragoza-Rios J.A., 2006. *Representation and Exploitation of Knowledge for the Description Phase in Declarative Modeling of Virtual Environments.* Master's thesis, Centro de Investigación y de Estudio Avanzados del Instituto Politécnico Nacional, Unidad Guadalajara, Guadalajara, México.

TOWARDS AUTOMATED FEATURE SELECTION IN REAL-TIME STRATEGY GAMES

Kurt Weissgerber, Brett Borghetti, Gary Lamont, Michael Mendenhall

Air Force Institute of Technology*
Graduate School of Engineering
2950 Hobson Way
Wright Patterson AFB, OH 45433-7765, USA

kurt.weissgerber; brett.borghetti; gary.lamont; michael.mendenhall@afit.edu

July 30, 2009

KEYWORDS
Real-time Strategy, Feature Selection, Classification, Clustering, Optimization

ABSTRACT

Since the underlying goal of an Artificial Intelligence agent in a real time strategy game (RTS) is to defeat the enemy, the correctness of any state representation can be determined by the agent's ability to quickly and correctly evaluate the likelihood the agent will win using that representation. Feature selection methods are techniques for finding the important parameters in a representation. We present two feature selection methods for representing a state in a RTS and show how they can generate classification models highly predictive of whether the agent will win or lose, long before the game is over.

INTRODUCTION

When designing an artificial intelligence player (agent) for a real time strategy game (RTS), one of the design decisions is how to internally represent the state of the game. State representation is important because it informs the decision-making process of the agent. In an on-line decision-making setting, the size of the state space should be reasonable so that an agent can make a search through state-action pairs to find one with a high expected value (where the value of a state is a function of the likelihood of winning from that state). Armed with this capability, the agent can take the action and then repeat the process, hopefully until it wins the game.

While keeping the state representation small is important for computational tractability, the abstract representation must express information highly correlated to value in the real state. In other words, if a state representation is useful, the expected value calculation for a representation of a state should closely match the value of the real state in the game.

A common method of representing a state in a computationally tractable way while maintaining a good approximation of the original reward value is feature selection. In feature selection, the goal is to find a good subset of key parameters that represent a larger set of parameters (i.e. the state of the game) so that the subset can be used to enable tractable computations.

This work explores feature selection in the RTS domain. Our goal is to develop a computational method for finding a subset of features which can be used to predict the agent's expected reward with high probability. We use two approaches to search through feature space, one deterministic (depth-first search with backtracking) and one stochastic (simulated annealing).

The next section discusses related work in state representation and feature selection. We then present a mathematical formulation of the feature selection problem for the RTS domain and describe the details of the two search methods used. Our data collection description and method of testing are followed by results of experiments. Finally, we conclude and discuss areas of future work, and explain how this research could be applied to an RTS agent.

RELATED WORK

Current Artificial Intelligence (AI) approaches vary in how they define a state. The Case Based Reasoning approach in (Ontanon et al. 2007) uses a vector of thirty-five features to define a state. Then, the case in

*The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

their database which closely matches the current state is selected for execution. Because each case is used to decide an entire strategy, many actions could be taken based on one state, which reduces the required computation time of the agent.

Dynamic scripting (Spronck et al. 2006) reasons over the values of rules and inserts new actions into an AI script at each turn. This was initially implemented in a role-playing game, then extended to an RTS architecture by (Kok 2008). One of the identified problems is the agent is not allowed to have knowledge of the environment when determining its current state; instead, it is limited to knowing the rule values. Kok believes giving the agent some knowledge of the environment would be useful.

Much research has been done in the areas of feature selection and classification. One taxonomy is given in (Jain et al. 1999), another in (Dash and Liu. 1997). Many researchers approach the feature selection/classification topic as two separate problems: first, choose the correct features, then determine how to classify using them. We believe an embedded approach as outlined in (Blum and Langley 1997), where feature selection/classification is done simultaneously, will be better for our problem.

PROBLEM DESCRIPTION

In the RTS domain there are a large number of features available to any agent. We need to determine which of these features are "important." For a feature to be considered important, its value at some time should be indicative of the result of the game. For example, one feature in the game could be the amount of money possessed by a player. If the amount of money a player has is important, then a state where we observed a high value for this feature (much higher than the opponents) would cause us to predict the player would win. If this relationship is true for a high percentage of the samples observed then this is an important feature. However, if high values for this feature are observed in both winning and losing game outcomes then it is not an important feature and should not be used in a state representation.

The previous discussion is a simple example; in reality, the game outcome is dependent on a large number of features, but there should be some features which do not significantly affect the outcome. Our goal is to find a subset of all the features available to an agent whose values are highly predictive of game outcome. To determine whether a feature is important, we build classification models based on a subset and use them to classify states as winning or losing.

In our experiment, the agent had perfect information about the entire game state. Features were the difference between the value of the agent's feature and the opponent's feature: if an agent has six infantry troops and his opponent has four, the value of the "infantry troop" feature is two.

The first part of the problem is to choose the correct features. The second part is to determine the combination of values for these features which lead to winning or losing states. Say we pick three features: infantry units, tanks and jet fighters. If we have a state where the value of this triple is (6, 4, 2), then we can say the agent is in a good position to win, he outnumbers the opponent in every unit category. However, if the value is (6, 2, -5) the prediction becomes harder. Perhaps tanks are unable to attack planes, and planes are slightly more powerful than infantry units.

If we pick a subset of five features and restrict the values to integers in the interval [-10, 10], then there are $21^5 = 4,084,101$ possible combinations. Predicting a value for each one of these combinations is computationally expensive and unnecessary, most of these states are close enough together they will result in the same outcome. Therefore, to find appropriate combinations we pick some of the state samples as representative of winning/losing states. Outcome predictions are assigned to samples based on their proximity to one of the chosen samples; if closer to a winning sample then predict a win; closer to a losing sample, predict a loss. We refer to the chosen samples as "centers" for the rest of this paper.

In most feature selection/classification problems, the two problems are solved individually. First, a set of features is chosen, then a set of centers. We believe the optimum feature set is determined by the optimum center set and vice versa. As a result, the problems should be solved at the same time, as a single optimization problem. Once we pick a feature, we also pick centers which best generalize the feature.

This optimization problem is, formally, given a data matrix X of dimensions $|N| \times |M|$, where N is the set of features and M is the set of samples (each sample is labeled as winning or losing), output a set of features $n \subset N$ and centers $m \subset M$, where m will contain $|m|/2$ winning centers and $|m|/2$ losing centers (a winning and losing center associated with each feature). The classification model defined by n and m will take some sample x, where x contains only the features in n, and determine to which of the centers in m it is closest. The classifier is a zero/one prediction function $f(x)$, where its output is zero if it finds x is closer to a losing center and one if it is closer to a winning center. For ease of notation, $f^*(x)$ will be used to refer to the actual winning/losing value of sample x.

The optimization problem is to maximize the function:

$$G(X) = \sum_{x=0}^{X} g(x) = \begin{cases} 1 & f(x) = f^*(x) \\ 0 & f(x) \neq f^*(x) \end{cases} \tag{1}$$

The value $G(X)/|X|$ would be the prediction accuracy of the classifier. A solution to this problem can be thought of as a binary string of length $|N| + |M|$. The first $|N|$ bits represent the features; if there is a one in a feature's position, then that feature is in the solution. A zero means it is not. Similarly, the last $|M|$ bits represent the samples; a one means the corresponding sample is a center, a zero means it is not. A visualization of this idea is in Figure 1.

F_1	F_2	...	F_N	C_1	C_2	...	C_M

Figure 1: Solution format. F_n is a feature, C_m is a center

All the possible solutions to this problem are $O(2^{|N|} \times 2^{|M|})$, it is NP-Complete. Like all problems in NP-Space, the optimal solution can not be found due to computational complexity. Instead, we can either reduce the complexity of the problem by abstracting away some of the details to obtain an approximate solution, or we can pursue a stochastic algorithm which finds a solution to the original problem which is not guaranteed optimal.

We develop an instance of each method and then compare the results found.

Deterministic Search

In a deterministic solution, we search the entire solution space of the problem to find the optimal answer. There are many algorithms to do so, including Breadth First Search, Depth First Search, Best First SearchCormen et al. (2001).

As already discussed, the possible solutions of this problem are of size $O(2^{|N|} \times 2^{|M)|}$. If we limit the size of n and m the solution space is reduced to $O(|N|^{|n|} \times |M|^{|m|})$. Additionally, if we pair every feature with a winning and losing sample, so when we choose a feature we also choose the winning/losing centers which best take advantage of this feature, we force the relationship $|m| = 2 \times |n|$. We pass the number of features to put in a solution, $|n|$, to the search as a parameter k. This further reduces the solution space to $O(N^k)$, which is deterministically computable when N and k are small.

To link features with samples, we computed a Bhattacharyya coefficient (BC) (Thacker et al. 1998) for each feature to determine how well it separated the two classes of data (winning/losing). The BC is a heuristic for determining whether the game will be won or lost using only a single feature. We pick the centers corresponding with the features as the median sample of the respective distribution.

The BC is calculated by taking a histogram of all the data, and then determining the probability of a sample falling in a bin for both classes. The two probabilities for each bin are multiplied together and summed over the entire histogram. Formally, this is:

$$BC = \sum_{i=1}^{I} P(W_i) \times P(L_i) \tag{2}$$

I is the number of bins in the histogram, W_i is the set of winning samples, L_i is the set of losing samples and $P()$ is the probability of the samples being in the bin. Figure 2 is a visualization of this idea.

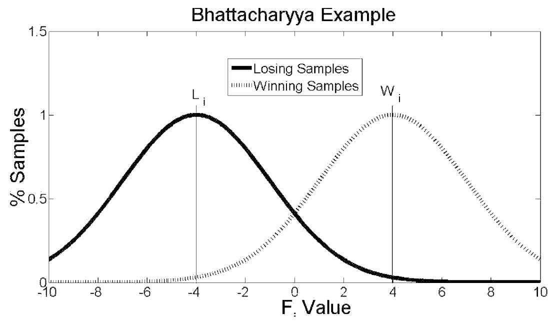


Figure 2: A visualization of the Bhattacharyya coefficient (BC) on feature F_i. The two curves are distributions over the winning and losing samples. The BC is a number between one and zero, expressing the amount of "overlap" of the two distributions; zero represents no overlap, while one represents complete overlap. On this graph, it is the space bounded by both curves. To pair a feature with a winning and losing center, we take the sample at the median of the respective distributions, symbolized by the lines W_i and L_i. We have expressed the win/loss samples for feature F_i as gaussian distributions, but the BC works for any type of distribution.

In any search, we need to define some general constructs. First, our goal is to construct a full solution which maximizes our objective function (the "fitness" of a solution). At each level, we add a triple (one feature and two centers) to a partial solution. We have a possible solution when our partial solution contains k triples, we test its fitness with equation 1 and save it if better than the best solution found so far. We terminate when we have explored all possible solutions.

Since k is determined by the user, we can use a Depth First Search with Backtracking (DFS-BT) (Lewis and Denenberg 1991) to generate and test all the possible solutions. The DFS-BT is more memory-efficient than a Breadth First or Best First Search because it only has to maintain its current level of the search tree and the best solution found so far. Computing the value of the objective function takes time $O(k^2 \times |M|)$: we select each sample in X and test its distance to every center in m, based on all the features in n. We compute this for each solution. A DFS of this solution space takes time $O(|N|^k)$: we construct each solution in at most k steps, which is a constant in this problem. Therefore, the entire search runs in $O(k^2 \times |N|^k \times |M|)$. In this domain, we can compute this for values $k \leq 4$ in a reasonable amount of time. A sample search tree is shown in Figure 3.

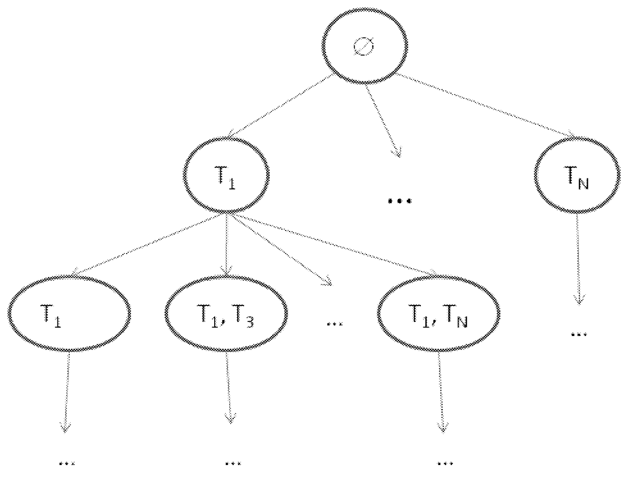


Figure 3: A Depth First Search Tree. At each level, a new feature/center triple is added to the solution (T_i). The search continues to level k.

The BC is a heuristic in the search. We order the feature/center triples based on this heuristic and allow the search to greedily pick some number l of these features, and then do a DFS-BT to pick the next $k - l$ features. This allows us to search to a lower depth in the solution tree. Figure 4 shows the potential gain from this modification. The complexity of the problem changes to $O(k^2 \times |N|^{k-l} \times |M|)$ and the new limitation of our search is $k \leq l + 4$.

We vary the values of l and k and determine the best classification models. A complete list of the parameter combinations tested are in Table 1.

Stochastic Search

As an alternative to reducing the solution space, we also use a stochastic local search algorithm, simulated annealing (SA), to solve the problem (de Castro 2006). Simulated annealing is used as a representative of all stochastic algorithms. Instead of adding to a partial solution at each step SA searches the solution space. At each step, a new solution in the neighborhood of the current solution is generated and its fitness is compared to the current solution. If the new fitness is better, the new solution is accepted and the process repeats. SA also allows for solutions with lower fitness values to be accepted with some probability P based on a user defined initial temperature T_0 and cooling parameter α; the temperature at step i is a function of the initial temperature and the cooling parameter. At the beginning of the search the temperature is high so the probability of accepting a solution with a lower fitness is higher. As the search progresses, the temperature falls and more and more frequently only higher fitness solutions are accepted. This is a method of dealing with the exploration vs. exploitation problem of any search. SA is biased towards exploration at the beginning, then leads to exploitation at the end. The algorithm terminates when the temperature approaches zero.

In our SA implementation, a solution is represented just as in the problem description, visualized in Figure 1. The initial solution is randomly generated. At each step, we swap out a feature and a winning and losing sample, an explanation of this swap is in Figure 5. The

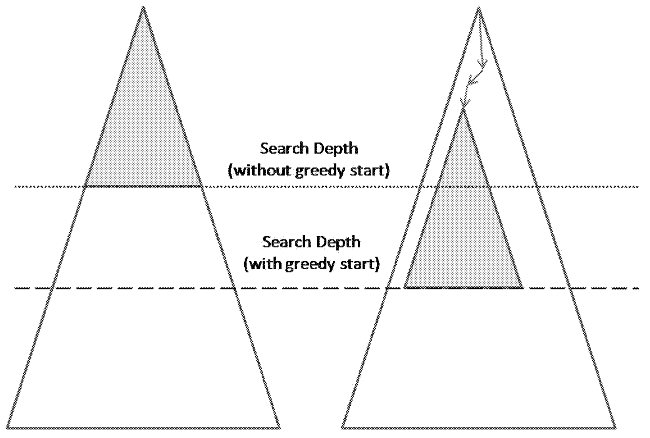


Figure 4: The increased space searchable when a greedy search portion is included

k	1	2	3	4	5	6	7	8
l	0	0..1	0..2	0..3	1..4	2..5	3..6	4..7

Table 1: Parameter combinations tested for deterministic search. Step-size for l, the number of levels to do a greedy search, was 1. The remainder of the levels are filled out by DFS-BT (DFS-BT done for the last $k - l$ levels).

solution at each iteration differs by at most one feature and two centers from the solution at the iteration before. Like in the deterministic search, we force $|n| = 2 \times |m|$, but we do not pair features with samples. This allows us to possibly find better samples to use as centers based on our feature choices.

$$s = F_1..F_N \mid S_1 ...... S_M$$

$$x = 0110 \mid 10010110$$

$$z_1 = 0101 \mid 10100101$$

$$z_2 = 1001 \mid 01101001$$

Figure 5: Proximity in solution space. s is a general solution, the first $|N|$ numbers are features, the next $|M|$ are samples. x is a possible solution, there are four features and eight samples in this example. The first four samples are winning, the last four are losing. z_1 is a possible nearby solution, one sample and two centers have been swapped. z_2 is not a nearby solution, two samples and four centers have been swapped out.

Temperature was decayed using a geometric regression, where $T_{i+1} = T_i \times \alpha$. The initial solution was randomly chosen from all possible solutions. Since SA is stochastic we ran the SA 100 times for each set of parameter values and took the best solution found. A complete list of the parameter combinations tested are in Table 2.

$\|n\|$	T_0	α
2..8	100	0.5
5	25..175	0.5
5	100	0.2..0.9

Table 2: Parameter combinations tested during SA. Step-size for $|n|$ was 1, T_0 was 25, α was 0.1

DATA

We implemented our algorithms on data from a RTS platform called Bos Wars (Beerten et al. 2008). Bos Wars is an open source RTS developed as a no-cost alternative to commercial RTS games. It has a "dynamic, rate-based economy", making it somewhat different than most other RTS games. Energy(money) and magma(fuel) are consumed at a rate based on the number of units/buildings a player owns. As the size of the player's army increases, more resources must be allocated to sustaining infrastructure. Additionally, Bos Wars has no "tech-tree", so all unit and building types can be created at the beginning of any game.

There are three scripted AIs packaged with the development version of the game, blitz, tankrush and rush. Blitz creates as many buildings/units as possible in the hopes of overwhelming the opponent. Tankrush tries to create tanks as quickly as possible, using a strong unit to beat the weaker units normally created at the beginning of a game. Rush creates as many units as quickly as it can and attacks as soon as possible in order to catch the enemy off guard.

There are eight maps packaged with the game. In most maps starting conditions for both players are similar. Each player has the same amount and access to resources and starts with the same number and type of units. We used three different two-player maps: two had similar starting conditions, one had a line of cannons (defensive buildings) for one player.

Additionally, there are three different difficulty levels for the game: Easy, Normal and Hard. Changing the difficulty level allows the AI to execute its script faster, so it progresses farther in its strategy in a given time period during a Hard game than a Normal game, Normal progresses further than Easy.

To collect data, we modified the source code so it would take a snapshot of the game at intervals of five seconds and output the feature values to a text file. Each snapshot consists of 30 different statistics[1]. We also created delta values for all the features based on the snapshot taken 25 seconds before, so there were 60 features to choose from.

Altogether, we ran 81 games. For the three maps, we ran three runs for each combination of AI (tankrush v rush, tankrush v blitz, rush v blitz) at each difficulty level, so every map has 27 game traces.

We assumed win/loss prediction is easier the closer one gets to the end of the game. We examined game states in the third quarter, the ones starting after 50% of the game had elapsed and before 75% of the game had

[1]Energy Rate; Magma Rate; Stored Energy; Stored Magma; Energy Capacity; Magma Capacity; Unit Limit; Building Limit; Total Units; Total Buildings; Total Razings; Total Kills; Engineers; Assault Units; Grenadiers; Bazoo; Dorcoz; Medics; Rocket Tanks; Tanks; Harvesters; Training Camps; Vehicle Factories; Gun Turrets; Big Gun Turrets; Cameras; Vaults; Magma Pumps; Power Plants; Nuclear Power Plants

elapsed. Our shortest game was about ten minutes long, while the longest was more than forty minutes. Predictions ranged from samples 2.5 minutes from the end of the game to 20 minutes from the end of the game.

METHOD

Our goal was to test the quality of the solutions found using the two different search methods. Our initial data set had 60 features and 4500 samples. To ensure an unbiased performance comparison between the two, we split our samples into two portions. The first portion, the Training Set, was used to train the algorithms. It contained two thirds of the data. The second portion, the Testing Set, was used to test the five best classification models generated by each algorithm.

To prevent bias of the algorithms to a particular data set, we used K-fold cross validation to determine the best classification models, where $K = 3$. We used 2/3 of the Training Set data to train the model, then used the 1/3 of the Training Set held out to pick the best model for comparison with the other algorithm. The partitions of the sets were chosen randomly from the Training Set, so the BC values for the features were not the same over all the sets. As a result, the models generated by the deterministic search were different in each fold.

Finally, we took the top ten classification models (top five from each algorithm), evaluated their performance on the Testing Set, and compared the results.

RESULTS

For the deterministic search, here are the features selected by the top five performing classifiers on the Training Set, along with the accuracies on the training data (based on their specific set):

- One: Gun Turrets; Change in Total Buildings; Vehicle Factories; Change in Tanks; Total Razings[2] - 91.0%
- Two: Power Plants; Change in Total Buildings; Change in Tanks; Total Razings - 90.5%
- Three: Gun Turrets; Vehicle Factories; Change in Tanks; Total Razings - 90.0%
- Four: Assault Troops; Energy Capacity; Vehicle Factories; Total Razings - 89.7%
- Five: Vehicle Factories; Change in Tanks; Total Razings - 89.6%

[2]Opponent's buildings destroyed

For the stochastic search, the features selected by the top five performing classifiers and their accuracies on the training data (based on their specific set) were:

- One: Magma Rate; Total Kills; Total Units; Engineers; Change in Gun Turrets - 86.6%
- Two: Change in Medics; Magma Rate; Change in Tanks; Assault Troops; Change in Power Plants - 85.8%
- Three: Change in Gun Turrets; Magma Rate; Assault Units; Big Gun Turrets; Change in Total Units - 82.1%
- Four: Grenadiers; Change in Harvesters; Assault Troops; Change in Big Gun Turrets; Nuclear Power Plants; Magma Capacity; Change in Magma Capacity - 81.7%
- Five: Magma Capacity; Big Gun Turrets; Change in Total Razings; Change in Building Limit; Change in Power Plants - 81.7%

The parameter values and accuracy on the Testing Set for the deterministic search are in Table 3. The stochastic values and accuracy are in Table 4.

Set	$\|n\|$	l	k	accuracy
Two	4	0	4	93.6
One	5	1	5	92.5
Four	4	0	4	92.1
Three	5	1	4	90.4
Five	3	0	3	89.0

Table 3: Top five results from deterministic algorithm. Ordered based on the classification model performance on the Testing Set; the accuracy shown is for the Testing Set. A greedy search was conducted to level l, then a DFS to level k. The number of features in the solution is $|n| = k$.

Set	$\|n\|$	T_0	α	accuracy
One	5	100	0.6	88.7
Four	7	100	0.5	87.1
Two	5	125	0.5	86.9
Three	5	100	0.8	82.9
Five	5	75	0.5	80.0

Table 4: Top five results from stochastic algorithm. Ordered based on the classification model performance on the Testing Set; the accuracy shown is for the Testing Set. $|n|$ is the number of features in the solution, T_0 is the initial temperature and α is the cooling parameter.

Additionally, we wanted to look at the effectiveness of our center selections. For each center, we determined how many samples it was closest to and how many of these samples it correctly classified. These results are in Tables 5 - 8.

Center	Right	Wrong	Total	% Correct
W1	33	3	36	91.7
W2	141	12	153	92.2
W3	169	50	219	77.2
W4	42	2	44	95.5
L1	531	5	536	99.1
L2	55	22	77	71.4
L3	251	0	251	100
L4	147	0	147	100
Total	1369	94	1463	93.6

Table 5: Center accuracy for DFS Two

Center	Right	Wrong	Total	% Correct
W1	107	22	129	82.9
W2	68	11	79	86.1
W3	130	47	177	73.4
W4	45	7	52	86.5
W5	11	4	15	73.3
L1	0	0	0	n/a
L2	62	19	79	76.5
L3	673	0	673	100
L4	109	0	109	100
L5	148	0	148	100
Total	1353	110	1463	92.5

Table 6: Center accuracy for DFS One

Center	Right	Wrong	Total	% Correct
W1	136	95	231	58.9
W2	46	0	46	100
W3	83	0	83	100
W4	53	0	53	100
W5	28	11	39	71.8
L1	183	0	183	100
L2	136	0	136	100
L3	86	31	117	73.5
L4	452	0	452	100
L5	95	28	123	77.2
Total	1298	165	1463	88.7

Table 7: Center accuracy for SA One

Center	Right	Wrong	Total	% Correct
W1	28	0	28	100
W2	76	83	159	47.8
W3	28	0	28	100
W4	79	0	79	100
W5	133	0	133	100
W6	14	0	14	100
W7	11	0	11	100
L1	233	70	303	76.9
L2	274	1	275	99.6
L3	27	0	27	100
L4	102	35	137	74.5
L5	61	0	61	100
L6	118	0	118	100
L7	90	0	90	100
Total	1274	189	1463	87.1

Table 8: Center accuracy for SA Four

CONCLUSION

In all but one case, the deterministic search attained accuracies $> 90\%$. The stochastic search attained accuracies $> 85\%$, except for one case. These are high accuracies, it is possible to generate good classification models using the search techniques outlined above.

The BC heuristic used in the greedy jump start of the deterministic search was ineffective. Three of the top five solutions did not use the greedy portion of the search; the two that did used only the first feature. However, using BC as a pairing mechanism to create feature/center triples was effective as demonstrated by the high center accuracies.

The features selected by all the deterministic solutions were similar and the features selected by all the stochastic solutions were also similar. However, there was little overlap between the two sets. The stochastic solutions did have lower accuracies, but does a better job of exploring the solution space because it is less restricted. The stochastic algorithm can choose any samples to serve as centers, while the deterministic algorithm is limited to those determined by the BC heuristic.

The center accuracies show both algorithms' ability to find good centers based on the features selected. There was only one center (in DFS One) which was completely unused; this shows potential overlap in the centers chosen for some of the features using the BC heuristic. However, all the other choices show good variation across the data; almost all were closer to more than twenty-five samples. There were some centers with poor classification accuracy; this may reflect overlap in the data samples.

Overall, the two search algorithms perform well. Both appear to be promising methods for feature selection, but the computation complexity of the deterministic solution makes it less attractive than increasing the effectiveness of the stochastic solution. Complete stats on computation time are not presented here, however, 100 iterations of the SA algorithm with eight features in the solution took 3 minutes, while a DFS to a depth of four took > 1 hour.

FUTURE WORK

A better heuristic is needed to jump start the deterministic solution. This would increase the value of the greedy search portion, and would lead to solutions containing more features. Additionally, instead of restricting features to one set of centers, allowing for more freedom in center choice could lead to better solutions.

The choice of a center in the stochastic solution should be used to decrease the number of centers which can be chosen from the samples. As in the deterministic solution, the feature selected should inform the choice of centers and vice versa. We should also look at the other features which are in the solution and think about their values in any selected center.

Additionally, instead of randomly choosing a feature/centers to swap out of the stochastic solution at each step, we should choose based on some value added aspect. If features and centers which did not significantly contribute to the solution's fitness are swapped out first, this should lead to better solutions.

Finally, our goal is to apply the generated solution (classification model) to an RTS agent. The method in this paper finds points which represent winning/losing middle game states (centers). The features selected can be used to reduce the actions considered by an agent; only actions which can affect important feature values should be taken.

For example, if the classification model generated contains the infantry unit feature and the agent is at a state where this feature value is closer to a losing center then it should take actions to increase the value of this feature. It could build more infantry units or the infrastructure to produce them, destroy the enemy's infantry units/infrastructure, build defensive emplacements which are effective against infantry attacks, etc. With a reduced set of possible actions, along with known good states, this problem becomes tractable.

The end goal of our research is to generate such an agent.

REFERENCES

Beerten F.; Salmon J.; Taulelle L.; Loeffler F.; Mistry N.; and Penfold T., 2008. *Bos Wars.* Open Source Software. URL `http://www.boswars.org/`.

Blum A.L. and Langley P., 1997. *Selection of relevant features and examples in machine learning. Artificial Intelligence*, Volume 97, no. 1-2, pp 245–271. ISSN 0004-3702. doi:http://dx.doi.org/10.1016/S0004-3702(97)00063-5.

Cormen T.; Leiserson C.; Rivest R.; and Stein C., 2001. *Introduction to Algorithms.* The MIT Press.

Dash M. and Liu. H., 1997. *Feature Selection for Classification. Intelligent Data Analysis 1*, Volume 2, pp 131–156.

de Castro L.N., 2006. *Fundamentals of Natural Computing.* Chapman and Hall/CRC.

Jain A.; Murty M.; and Flynn P., 1999. *Data Clustering: A Review. ACM Computing Surveys*, Volume 31, no. 3, pp 264–322.

Kok E., 2008. *Adaptive Reinforcement Learning Agents in RTS Games.* Master's thesis, University Utrecht, The Netherlands.

Lewis H.R. and Denenberg L., 1991. *Data Structures & Their Algorithms.* Addison-Wesley.

Ontanon S.; Mishra K.; Sugandh N.; and Ram A., 2007. *Case-based Planning and Execution for Real-time Strategy Games.* In *ICCBR '07: Proceedings of the 7th International Conference on Case-Based Reasoning.* pp 164–178.

Spronck P.; Ponsen M.; Sprinkhuizen-Kuyper I.; and Postma E., 2006. *Adaptive Game AI with Dynamic Scripting. Machine Learning*, Volume 63, pp 217–248.

Thacker N.A.; Aherne F.J.; and Rockett P.I., 1998. *The Bhattacharyya Metric as an Absolute Similarity Measure for Frequency Coded Data. Kybernetika*, Volume 34, no. 4, pp 363–368.

MACHINATIONS: ELEMENTAL FEEDBACK STRUCTURES FOR GAME DESIGN

Joris Dormans
Hogeschool van Amsterdam
Weesperzijde 190
1097DZ Amsterdam, The Netherlands
E-mail: j.dormans@hva.nl

KEYWORDS
Game design, game structure, feedback loops, design patterns.

ABSTRACT

This paper presents a structural model that can describe the gameplay mechanisms found in the great majority of games. In particular, different types of feedback loops are suggested as the elemental structural patterns that give rise to interesting gameplay. Any formal method of describing games must be able to express feedback loops and their relations to be of any use in designing games. Using a formal notation based on Petri nets and the concept of internal game economies, different types of feedback loops are investigated and discussed.

INTRODUCTION

There are many ways to look at games. They are entertainment devices geared towards pleasurable interaction on one hand, and they are cultural media contributing to and reflecting on contemporary culture on the other. At the same time, games consist of complex rule systems that model fictional environments and facilitate player agency in those environments. This allows us to approach games in many different ways. Without suggesting that one approach is better than the other, this paper treats games as complex state machines: interactive devices that can be in many different states, and which current state affects the transition to a new state. In particular, this paper focuses on the role feedback loops play in games and sets out to identify the most elemental types of feedback that can occur in games.

It is an approach that focuses on game systems and neglects players. This is a move that can and has been critiqued: without players games would be, quite literary, meaningless. The formal rule systems of games are subject to constant change and reinterpretation. A formal approach always runs the risk of turning a blind eye towards this dynamic and important dimension of games (see Malaby 2007). However, a game designer first and foremost builds games systems. It this system that codifies the player's possible interaction and generates individual game experiences. The aim of this paper is to understand the elemental structures that contribute to quality gameplay and that ultimately facilitates the expressive and dynamic nature of games.

This is a structural approach to game systems; it focuses on the structure of game systems and patterns that might be found in these structures. It is not what I call a formal approach, as it lacks the mathematical rigor required to deserve such label. Yet, in it intents to do exactly what many other would loosely call a formal approach: to provide a common, abstract and precise language that can be used to increase our understanding of games. It is not the first attempt in this respect. The call for such a formal or structural language has been expressed before with mixed results (LeBlanc 1999, Church 1999, Kreimeier 2002, Grünvogel 2005, Koster 2005, Bura 2006). To date, none of these have been so successful that they have become an industry or academic standard. I feel that is largely due to the fact that they tend either to be too mathematical for the diverse population of game designers and scholars, or were not explored or presented with enough detail. Most importantly, for a framework like this to be of any success, it requires the designers to make an investment by learning the paradigm. Only an obvious return justifies this investment: using the framework should improve design and speed up the design process.

Many of the concepts proposed in these works have found their way into the approach presented in this paper. But I also drew inspiration from fields as diverse as linguistics, semiotics, the science of complexity and modern control theory. This paper expands the ideas presented in an earlier paper (Dormans 2008) which used a similar approach based on UML diagrams. The response on that work, as well as my work with students and my own experience as a game designer, inspired me to focus more closely on feedback patterns in games and their relation with the flow of the game, which I understand here as a particular progression through different game states that is the result of playing a game.

When one sets out to model anything as complex as games, the problem always is that a model can never do justice to the true complexity of that what it one tries to model. This is true for most models, and the best models succeed in stripping down the complexity of the original by leaving out, or abstracting away, many important details. This is certainly the case with the models I present here. However, any model is a tool that can help us understand and work with complex systems. The model presented in this paper certainly is such a tool. To be able to use the model to the best effect understanding the concepts that informed the creation of the model is required. As any model, this model only facilitates understanding; it never is a substitute for it.

This paper starts by investigating games as state machines in the first two sections. It discusses state machine diagrams and

Petri nets as possible methods for modeling games. In the next two sections it explores the important structural notion of feedback loops and devise a diagram language based on Petri nets to express feedback structures in games. An example of how understanding of a game's feedback structure can be utilized to improve its design is next. Finally, a small number of elemental feedback patterns that can be found in many games are presented.

I am not the first author to build a set of game design patterns. Staffan Björk and Jussi Holopainen (2005) do exactly that. Their collection of over two-hundred patterns has a broader scope than what I present here, but describe games mostly from the outside. In contrast, this approach tries to construct game design patterns inside-out. It takes theoretical concepts concerning state machines as its starting point, and tries to identify patterns in the structure of games from there.

The notation for feedback structures developed in this paper is adapted from the interactive diagrams that were developed for my research into feedback structures. These diagrams, an online tool to create these diagrams as well as an extended list and more thorough description of the patterns presented here can be found on the website that accompanies this paper: www.jorisdormans.nl/machinations/wiki.

GAMES AS STATE MACHINES

A game can be understood as a state machine: there is an initial state or condition and actions of the player (and often the game, too) can bring about new states until an end state is reached (Grünvogel 2005). In the case of many single-player video games either the player wins or the game ends prematurely. The game's state usually reflects the player's location, the location of other players, allies and enemies, and the current distribution of vital game resources. From a game's state the player progress towards a goal can be read. State machines can be diagrammed. In these diagrams circles represent states and arrows represent transitions between states. Often these transitions are marked with labels that indicate what brings the transition about. For example, figure 1 represents a state machine of a fairly straightforward and relatively simple, generic adventure game.

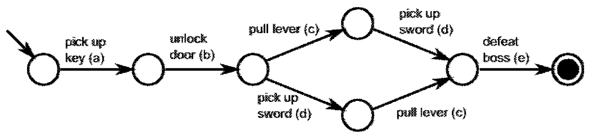


Figure 1: Adventure game state machine

Many things have been omitted from this diagram. For example, the way the player moves through the world has been left out, which is no trivial aspect in an action adventure game with a strong emphasis on exploring (as is the case with most *Legend of Zelda* games). Still, we can easily abstract it away from this diagram as movement does not seem to bring any relevant changes to the game state (other than the requirement of being in a certain location to be able to execute a particular action).

The question is whether or not a formal representation of a game is of any use. Looking at the diagram this game does not look complex at all. The possible set of different trajectories trough the state machine is very limited. The only possibilities are 'abcde' and 'abdce'. This game is a machine that cannot produce any other result. It is, to use Jesper Juul's categories, a game of progression, and not a game of emergence (Juul 2005). To be fair, most adventure games have much larger set of states and player actions that trigger state transitions. There might be side quests for the player to follow, or even optional paths that lack the symmetry of the two branches in figure 1. A game like this might grow in complexity very fast (see for example figure 2), but still the possible trajectories remains ultimately finite. Yet this is what a lot of games have done in the past.

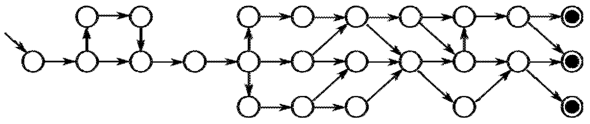


Figure 2: A more complex game state machine, but one that still produces a finite set of possibilities

To really change the character of the output of the machine, a structural change needs to be made to the set up of the game. One possibility is to create recursion in the diagram. Recursion simply means that a transition takes you back to a previous state, allowing you to loop through the diagram in different and more varied ways. Chomsky has shown that including recursion in a state machine the set of possible results quickly becomes infinite (1957: 18-25). For example we could make the supply of keys and locked doors in our previous game endless allowing the player to loop back indefinitely (see figure 3). The possible set of results is now {abcde, abdce, ababcde, ababdce, abababcde, abababdce, ..., etc.}.

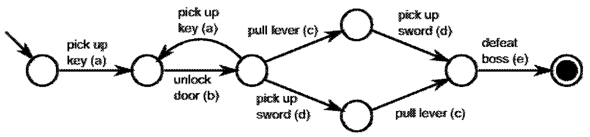


Figure 3: an adventure game with recursion

Of course, this has little meaning in the context of the game, unless the number of keys the player collected somehow affects his chance of defeating the end boss. In the later case we might want to consider creating different states for each number of keys the player has collected but this would create an infinite number of states, which is impossible to diagram. In a real implementation of a game, the number of keys would be stored in a variable, but state machine diagrams have no method of representing variables. This is problematic as the state of many games is best expressed using variables like these.

Consider the board game *Risk*. In this game the state of the game can be expressed by each individual player's current possession of lands, armies and cards. The number of different distribution of lands, armies and cards over different players is, for all practical means, too large to be diagrammed in a useful manner using classic state machine diagrams. Even if, we abstract away the location of countries on the board and reduce the state to the number of lands, armies, and cards in the player's possession, the number of possibilities is still too large for a classic state machine diagram. It is impossible to model such a game using a classical state machine diagram; games might be state machines, they are rarely finite.

A DIFFERENT LOOK AT GAME STATES

Game states are usually much better expressed using a mix of variables and states. Not only allows such a mixture to model the large number of states encountered in most games, it also shifts attention towards the transitions between the states, which corresponds to user actions. If we take *Risk* again as our example we can construct a diagram for this game with only four states and seven transitions (figure 4) in which each transition affects the number of lands, armies and cards.

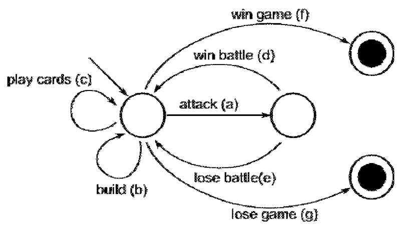


Figure 4: a state diagram for *Risk*

The diagram shows a lot of recursion, and as a result an infinite number of different paths through the state machine are possible. A diagram that focuses on transitions is clearly more capable to capture the nature of games and the varied sessions of play. However, the diagram omits important rules and mechanics. For example, in *Risk* you can only play cards if you have a valid set of three, the number of armies you gain from a building action depends on the number of lands you control, and the chances of reaching victory in a battle is affected by the number of armies you have. It is possible to write down these rules in or next to the diagram, although this will do little to make the diagram more accessible to most game designers.

Petri nets are an alternative modeling technique suited for game machines (*cf.* Bura 2006). Petri nets work with a system of nodes and transitions. A particular type of nodes: places, can hold a number of tokens. In a Petri-net a place can never be connected directly to another place, instead a place must be connected to a transition, and a transition must be connected to a place. In a classic Petri net places are represented as empty circles, transitions are represented as squares and tokens are represented as solid circles. In a Petri-net tokens flow from place to place; the distribution of tokens over spaces represents the current state of the Petri net (see figure 5). This way the number of states a Petri net can express is infinitely larger than non-looping, finite state machine diagrams. Petri nets put much more focus on the transitions and have a natural way of representing integer values through the distribution of tokens over the places in the network. However, Petri nets can be somewhat difficult to read, as the transitions are often identified using names and are defined with formal mathematical definitions.

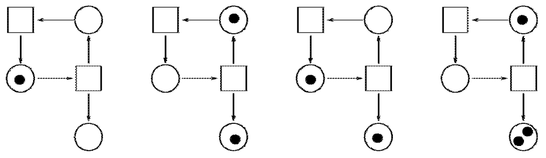


Figure 5: Four iterations of the same Petri net showing movement of tokens through the network

GAME ECONOMY

One way to enhance the readability of Petri nets is to reduce the number of possible transitions to a small set of basic operations that still allows us to represent game mechanics. Although this would still entail an abstraction of a game's true logic, I argue it is possible to come up with a set that is able to express a game's characteristic flow, and therefore can be used to create a useful model for the game. This set is based on the idea that all games can be understood in terms of their internal economy.

According to literature, most, if not all, games have an internal economy and this economy plays a vital role in its emergent behavior (Adams & Rollings 2007). A game's economic system is dominated by the flow of resources. In games resources can be anything: from money and property in *Monopoly*, via ammo and health in first person shooters, to experience points and equipment in role playing games. Even more abstract aspects of games, such as player skill level and strategic position can be modeled through the use of resources. Once we have identified a game's most important resources we can look at how these resources are produced, consumed and how they interact. In the case of *Risk*, we might think of lands, armies and cards as resources, where both lands and cards can be used to produce armies, and armies can be risked to gain lands and cards.

Adams and Rollings identify four basic economic functions for games: sources, drains, converters and traders (ibid.: 331-340). Sources create resources, drains destroy resources. Converters replace one type of resource for another, where as traders allow the exchange of resources between players or game elements. These economic functions set up a network of economic transactions that determine the flow of a game. I found that, together with a concept for pools, or places where resources can gather, these economic functions are indeed the essential operations that can represent most game mechanics. Of these structures, sources and drains are the most elemental. It can be easily shown that a converter can be constructed from a combination of a source and a drain, whereas a trader can be created from a set of interlinked pools. Figure 6 explains the diagrammatic language I use to express these elements. This language is loosely based on Petri nets. It incorporates the ideas of places and tokens. It specifies a number of base transitions that represent the elementary operations required to build an internal economy. In addition, special links represent communication of a pool's status. These can affect the settings of a particular operation. A special set of indicators can be used to mark different types of unpredictability (see feedback signatures below).

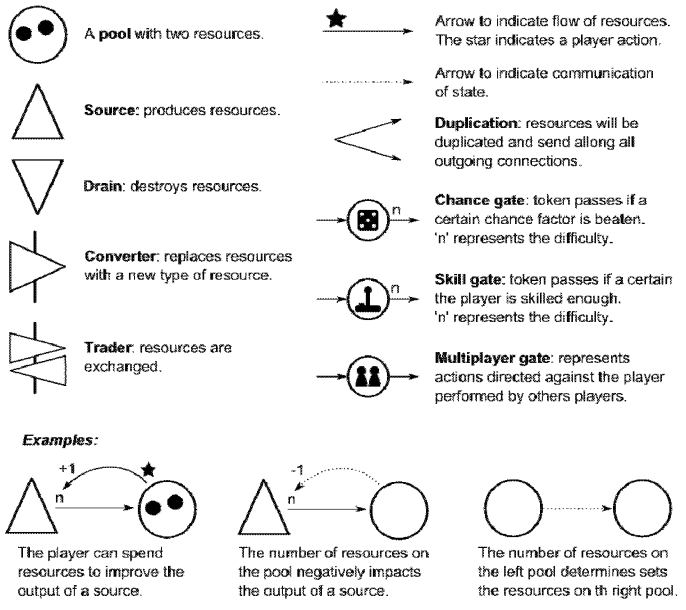

A pool with two resources.
Source: produces resources.
Drain: destroys resources.
Converter: replaces resources with a new type of resource.
Trader: resources are exchanged.
Arrow to indicate flow of resources. The star indicates a player action.
Arrow to indicate communication of state.
Duplication: resources will be duplicated and send allong all outgoing connections.
Chance gate: token passes if a certain chance factor is beaten. 'n' represents the difficulty.
Skill gate: token passes if a certain the player is skilled enough. 'n' represents the difficulty.
Multiplayer gate: represents actions directed against the player performed by others players.
Examples:
+1
n
-1
n
The player can spend resources to improve the output of a source.
The number of resources on the pool negatively impacts the output of a source.
The number of resources on the left pool determines sets the resources on th right pool.

Figure 6: 'Game economy' diagrams

Figure 7 is a diagram of the internal economy of *Risk*. As you can see in this diagram, unlike Petri nets, it is possible to directly connect multiple transitions. The different colors denote different mechanics are structural features of the game. The black elements and connections represent *Risk's* main mechanic of risking armies in battle to gain land. The light green connections and elements in the middle of the diagram indicate the building mechanism which in turn takes the number of lands as an input. The dark grey elements on the bottom indicate the bonus armies gained from capturing continents. The dark grey elements on the top represent the card mechanism in risk: a successful attack will get the player a card. Particular sets of three card will get the player more armies as well. The random knot indicates that this mechanism is subject to the randomness generated by the drawing of a card. The light grey elements on the top and the right indicate the effects generated through multiplayer dynamics: loss of armies and lands, which is informed by the number of lands and continents the player has, (for reasons of clarity a similar connections from armies to the multiplayer dynamic mechanism is omitted). In the game of *Risk* the main resources are easily identifiable: armies, lands and cards are represented by actual playing pieces, positions on the board and playing cards. However, sometimes resources can be more abstract. To stay with the example of *Risk*, strategic positions can be seen as another resource in the game, as can be player skill. In a games like *Go*, *Chess* or *Checkers*, strategic position is often more advantageous than the number of playing pieces under your control. Likewise, in a platform game, the avatar's altitude can be vital resource in reaching the end of a level or gaining advantage over his enemies. The use of abstract resources can be vital in understanding a game. In *Boulder Dash*, for example, a level's relative instability is an important factor. One might say that the player is constantly converting movement and collected diamonds into more instability. Should the instability exceed the player's skill level, he loses. Abstract resources can be modeled just like other resources. To give an example, jumping in a platform game can act like a source of the abstract resource altitude. While at other moments in the game, altitude might be converted into victory points or spent to reduce risk of difficult actions.

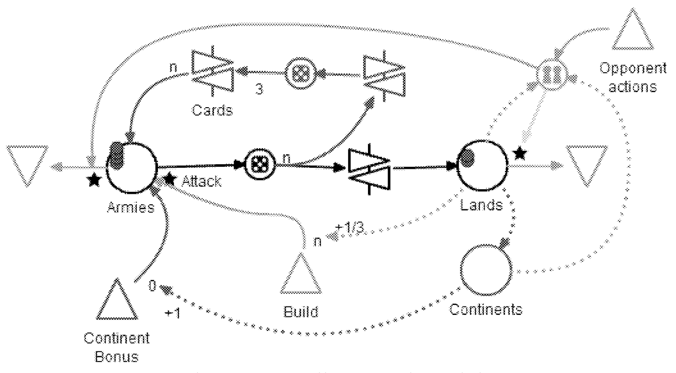


Figure 7: A diagram for Risk

FEEDBACK

Just as recursion is an important structural characteristic of state machines that increases the number of sequences a state machine can produce, feedback is an equivalent structural characteristic of a game's economy. Here feedback has nothing to do with giving the player information about the game or it state, rather feedback is understood in its original meaning, where the output of a process feeds back into the same process often strengthening the process further. A classic example of feedback in games can be found in *Monopoly* where money is spend to buy property which in turn generates more money, with which the player can buy more property, etcetera. The concept of feedback comes from classic control theory and has been introduced to the game design community by Marc LeBlanc (1999).

As is the case in classic control theory (DiStefano III, et. al. 1967, Andrei 2005), Marc LeBlanc distinguishes between two types of feedback: positive and negative feedback. Positive feedback strengthens itself and destabilizes a system. Positive feedback occurs whenever a small deviation will create a stronger deviation, which creates a stronger deviation in turn, as is the case with the *Monopoly* example above. Positive feedback can be applied to positive game effects but also to negative game effects, as is the case with loosing pieces in *Chess*, which increases the chances of loosing more piece, etcetera. LeBlanc suggests that positive feedback drives the game to a conclusion and magnifies early successes (LeBlanc 1999, see also Salen & Zimmerman 2003: 224-225). Negative feedback is the opposite of positive feedback. It stabilizes a game by diminishing differences between players, by applying a penalty to the player who has done something that takes him closer to his goal and winning the game, or by giving advantages to the trailing players. LeBlanc points out that in most multiplayer games that allow direct interaction some sort of negative feedback is already in place, as most sensible players will target the leader more than any other player. As one might expect negative feedback can prolong a game and magnifies late successes (*ibid.*).

Control theory, in almost all cases, strives for negative feedback while avoiding positive feedback, as it aims to create stable systems. A large part of control theory concerns itself with determining and optimizing the stability of the system. For games the situation is, of course, very different. Positive feedback loops are much more frequent in games because, in general, designers understand that players do not want to play a game that drags on forever. Yet, negative feedback is also wanted, as most games with only positive feedback will seem too random to many players as they will unable to catch the player who took an early lead. *Monopoly* is a good example of this effect as an early, lucky break is an accurate prediction of who will win in the end. And despite the lack of negative feedback the game still seems to drag on forever.

Marc LeBlanc observations have been picked up by influential game designers and theorists. It features prominently in the work of Katie Salen and Eric Zimmerman (2003), Ernest Adams and Andrew Rollings (2007), and Tracey Fullerton (2008). They use it as a promising, analytical lens for game design. In an earlier paper I have discussed feedback in relation to emergence in games (Dormans 2008) following suggestions by Jochen Fromm (2005) who states that true emergence can only occur in systems with multiple feedback loops. From these discussions it becomes clear that feedback goes a long way in explaining the flow of a game; many games can be characterized by the particular set up of feedback loops. But, to my knowledge, none of these discussions have attempted to expand on LeBlanc's original idea. So far, no-one looked beyond positive and negative feedback in any detail or tried to identify the most elemental patterns of feedback in games. This is what I intend to do here.

FEEDBACK SIGNATURES

The beauty of the diagrams I propose is that they are very effective in capturing feedback loops. Feedback is realized by a closed flow of resources and/or state connections. In figure 7, there are four feedback loops clearly visible. The first feedback loop involves the capture of lands and the positive effect this has on the number of armies you can build, with which you can capture more lands. In the diagram the loop is closed by green building mechanism. The second feedback loop involves the cards (the red mechanic), which are rewarded for winning lands, and which can be converted into more armies once a set of three cards is collected. The third feedback loop is formed by the blue mechanics representing the capture of continents. Finally the purple, multiplayer dynamic mechanism constitute the fourth feedback loop.

The first three feedback loops are positive: more lands or cards will lead to more armies which will lead to more lands and cards. Yet they are not the same. The feedback of cards is much slower that the feedback of lands, but at the same time the feedback of the cards is also much stronger. Feedback from the capturing continents operates fast and strongly. These are important characteristics of the feedback loops that have a big impact on the dynamics of the game. Players are more willing to risk an attack when it is likely that the next card they will get completes a valuable set: it does not improve their chances of winning a battle but it will increase the reward if they do. Likewise the chance of capturing a continent can inspire a player to take more risk than he should. Only identifying all feedback loops as positive is not enough to explain these gameplay effects.

Table 1 lists seven characteristics that are used to describe the signature of a feedback loop. At a first glance some of these characteristics seem overlapping, but they are not: It is easy to confuse positive feedback with constructive feedback and negative feedback with destructive feedback. However, positive destructive feedback exists as is the case with loosing pieces in a game of *Chess*. Likewise, the board game *Power Grid* employs a mechanism in which the game leaders have to invest more resources to build up and fuel their network of power plants: negative constructive feedback.

Table 1: Characteristics of feedback

Characte-ristic	**Value**	**Description**
Type	Positive	Enhances differences, destabilizes the game.
	Negative	Dampens differences, stabilzes/balances a game.
Effect	Constructive	Operates on a game effect that helps you win.
	Destructive	Operates on a game effect that will make you loose.
Return	High	The net gain is high.
	Low	The net gain is low.
	Insufficient	The gain does not outweigh the investment (net gain is negative).
Investment	High	Many resources must be invested before the feedback is activated.
	Low	Few resources must be invested before the feedback is activated.
Speed	Fast	The effects of the feedback are fast or immediate.
	Slow	The effects of the feedback take time or several iteration to activate or kick in.
Range	Short	The feedback operates directly over a few steps.
	Long	The feedback operates indirectly over many steps.
Durability	None	The feedback works only once.
	Limited	The feedback works only over a short period of time.
	Extended	The feedback works over a long period.
	Permanent	The effects are permanent.

The strength of a feedback loop is an informal indication of its impact on the game. Strength cannot be attributes to a single characteristic; it is the result of several. For example, permanent feedback with a little return can have a strong effect on the game.

In many games the characteristics of feedback are affected by outside factors such as chance, skill and social interaction (see Table 2). Feedback in a multiplayer game that allows direct player interaction like *Risk* can change over time. As LeBlanc already pointed out, it often is negative feedback as players act stronger, or even conspire against, the leader. At the same time, it can also be positive as in certain circumstances it can be beneficial to pray on the weaker player. In other cases random chance can affect the nature of the feedback as is the case in many board games that involve dice.

Table 2: Determinability

Deterministic	Given a certain game state, the feedback will always act the same.
Random	The feedback depends on random factors. The randomness can affect the feedback's speed and/or return, or the possibility of feedback occurring at all. Or the return might be infrequent. Random feedback is difficult for the player to assess, and increases the chance of deadlocks.
Multiplayer dynamics	The type, strength, and/or game effect of the feedback are affected by the direct interaction between players.
Meta-dynamics	The type, strength, and/or game effect of the feedback are affected by the strategic interaction between players.
Player skill	The type, strength, and/or game effect of the feedback are affected by the player's manual skill in executing the action.

The skill of player in performing a particular task can also be a decisive factor in the nature of feedback, as is the case for many computer games. For example, in a shooter game there often exists a feedback loop between the ammunition a player invests to defeat opponents and the ammunition these opponents drop when they are killed. The player's skill is an important factor in this feedback as a skillful player will waste less ammunition; his investment is lower than that of a less skillful player. Here player skill is a factor on the operational or tactical level of the game. Games of chance, tactical skill, or games that involve only deterministic feedback, a whole set of strategic skills can be quite decisive for the outcome. However, that is a result of a players understanding of game's feedback structures as a whole, and as such it is not an element that can or needs to be modeled within the structure.

Games that feature only deterministic feedback, can still show surprising emergent behavior and unexpected outcomes. In fact, it is my conviction that a well-designed game is build on only a handful feedback loops and relies on chance, multiplayer dynamic, and skill only when it needs to and refrains from using randomness as an easy source of uncertainty.

Combining the characteristics of feedback with its determinability it is possible to describe a signature for each feedback loop and a feedback profile for different games. While a profile like this can be very helpful in identifying the nature of feedback in a game, it does little to reveal the interaction between different feedback loops. This is where diagrams, such as figure 7, excel. Many of the characteristics of feedback loops described above can be read from the diagrams. The effect of the feedback is directly related to the constructive or destructive nature of the feedback loop, whereas return and investment depends on the number of resources involved. Range can be read from the number of elements involved in the feedback loop, speed from the number of iterations required to activate the feedback. Slightly more difficult to read are the return and the type of feedback, but this is possible, too. In the diagram for *Risk* I have already included a symbol to mark chance factors; this is extended with symbol for multiplayer dynamics, meta-dynamics and player skill (see table 2). The type of feedback (positive or negative) is perhaps the most difficult to read from a static representation, and requires careful inspection of the diagram, but this is possible, too. The plus symbols in the diagrams in the paper do not indicate positive feedback, only that there is positive correlation between the number of resources in the pool and the value it is affecting, which can induce negative or positive feedback.

FEEDBACK ANALYSIS

An analysis of a game's feedback loops can be used to identify structural strengths and flaws in its design. To create interesting and varied gameplay feedback is an important tool, and most successful games incorporate one or several feedback loops. Structural flaws, or 'bad smells' in analogy to software engineering, are constructions that are best avoided. If we take *Risk* again as our example, we can identify one of its problems from play experience: building as often as you can is an effective, almost dominant, strategy. In fact, the game has a rule that disallows players to build for more than three turns in a row to counter this strategy. Inspection of the feedback structure of the game suggests another way of resolving the problem. Attacking feeds into a triple positive feedback structure (lands, cards and continents), which is a strength of it its design, but apparently the feedback is not effective enough. Adjusting the feedback of lands will help only a little as building is part of the same feedback loop and will probably strengthen the unwanted behavior. Either the feedback through cards or the feedback through continents needs to be improved. The card feedback loop involves two random factors: success of attack and the blind draw of the card itself. This makes the feedback unpredictable and very hard for the player to assess. In general, involving too much randomness in the same loop is best avoided, especially when this randomness affects different steps in the loop. It is very hard to balance and predict the feedback of such a loop, so reducing the randomness, for example by allowing the winner a pick of three open cards, will help a lot.

Alternatively the feedback through the capture of continents can be improved. The problem with this feedback is that is strong, permanent, direct and fast: it is very obvious and will inspire strong reaction by opposing players, in other words it acts as a red flag. Combined with a relative high investment, it is a difficult strategy, but one that is very rewarding if it succeeds. The strength and the obviousness of the feedback which invites a strong negative feedback create a feedback loop that is too crude: it is either on and going strong or it is off. Either the player succeeds in taking and keeping a continent and has a very good shot at winning, or players quickly take the continent away from the player. By making the feedback less strong, and perhaps increase the number of continents (or rather regions) for players to conquer, a far more subtle feedback loop is created that will pay-out more often without unbalancing the game too much.

ELEMENTAL FEEDBACK PATTERNS

Looking at feedback structures in games, many recurrent patterns emerge. Below is a short list of patterns with some examples. Some patterns are diagrammed as well, although often there are multiple ways to implement the pattern. These descriptions are informal, they are presented here as a sample of what feedback patterns can be found in games. Extended descriptions that follow more closely the format for design patterns used in software engineering (Gamma et. al. 1995), and game design pattern libraries inspired by those patterns (Kreimeier 2002, Björk and Holopainen 2005), including interactive diagrams and multiple sample implementations can be found on the website that accompanies this research.

Dynamic Engine – A resource called energy is produced by a source and can be spent to improve its flow. This constitutes constructive permanent positive feedback (see figure 8). *Settlers of Catan* at its heart has a dynamic engine that is affected by some randomness: randomly selected tiles produce resources for players that have villages and cities next to the tile. Building villages improves the chances of player to get resources, while upgrading villages to cities increase the resource output of a tile when they are selected.

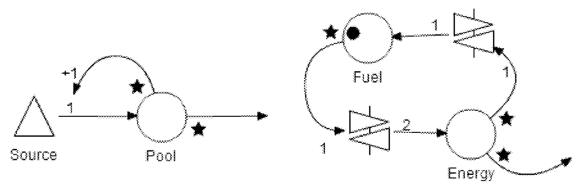


Figure 8: Dynamic Engine (left) and Converter Engine (right)

Converter Engine – If a player can change one type of resource (energy) into another (fuel) and than change it back into energy to generate a surplus of energy the game includes a converter engine (see figure 8). *Power Grid* is an example of a converter engine. In this game players spend money to buy fuel and burn fuel to make money. The surplus is used to invest in better power plants, among other things. The risk of feedback engines is the chance of deadlock, if both resources dries up the engine dies out and cannot be revived. Consider combining a feedback engine with a weak static engine to prevent deadlocks (as is the case in *Power Grid*). A converter engine offers more opportunities to create positive feedback and is therefore well suited as part of the engine building; a higher level pattern in which players compete to build efficient economic engines. Most real-time strategy games follow a complex engine building pattern with destructive feedback between the players, not unlike *Risk* (*cf.* Salen & Zimmerman 2003: 222).

Playing Style Reinforcement – Slow, positive, constructive feedback on player's actions that also have another game effect causes the player's avatar or units to develop over time. As the actions themselves feed back into this mechanism the avatar or units specialize over time, getting better in a particular task. As long as there are multiple viable strategies and specializations, the player's avatar or the units will, over time, reflect the player's preferences and playing style, often this mechanic employs experience points as a resource (see figure 9). Playing Style Reinforcement is a common pattern that can be encountered in most games that include 'role-playing elements'.

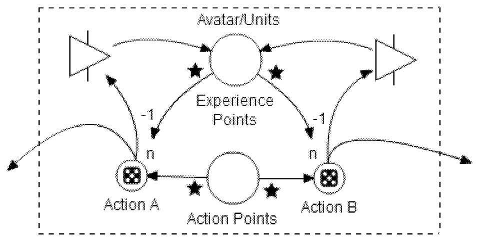


Figure 9: Playing Style Reinforcement

Escalating Complications– The basis of many action games is confronting the player with a task that keeps growing more difficult as the player's actions to complete a goal also applies feedback to the skill needed to complete the task, reducing the effectiveness of the skill (see figure 10). A classic example can be found in *Space Invaders* where destroying an alien makes the others move slightly faster, increasing the difficulty of destroying the remaining aliens.

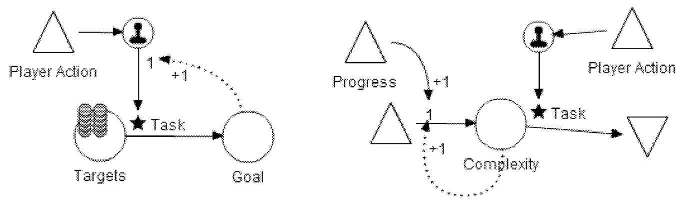


Figure 10: Escalating Complications (left) and Escalating Complexity (right)

Escalating Complexity – Another basic set-up for action games is introducing a system in which the positive feedback to the game complexity is automatic and steady, while the player actions work to reduce complexity. As the game progresses complexity will be created increasingly faster (see figure 10). This way, skilled players can manage the difficulty longer than players with less skill. Games like this remain balanced for a while and then quickly spin out of control. *Tetris* is an excellent example of this pattern. When the positive feedback mechanism is unsteady or unpredictable, the pace of the game can vary a lot.

CONCLUSION

There are some limitations to the use of these diagrams. The idea of internal economy is suited better to some games. In particular, it works very well for board games. In games where economy is more abstract these diagram can be difficult to abstract, and many games can be diagrammed in multiple ways depending on the analyst's focus. Still, feedback loops can go a long way in explaining the flow of a game. Gameplay can be related to more characteristics of feedback loops than positive and negative feedback only. In addition, the delicate interaction between multiple feedback loops must be taken into account to get a complete picture of the dynamics of games as (infinite) state machines. The list of feedback patterns presented in this paper, and on the accompanying website is neither definitive nor complete. For now, my research into feedback structures is still ongoing. Currently, patterns are still being harvested from analysis of existing games, games under production, and by exploring theoretical possibilities suggested by the framework. Future effort is aimed at collecting more patterns and establishing methods and practices for using the patterns to improve the design process of new games. To this end I will be looking at correlating particular game design patterns with specific game design goals in order to provide more grip on the elusive nature of gameplay and its (serious) application.

It is important to note that although examples were illustrated in both words and diagrams, there are many different ways of implementing these patterns. A pattern description is not a prescription of a particular implementation. These patterns have many different ways in which they can be combined, and each individual game provides its own particular opportunities for interesting combinations. Although, I would personally lean towards the simplest possible implementation, as it is usually my objective to create complex, rather than complicated, games. It is best to consider this framework as a set of building blocks (pools, resources and economic functions) that can be used to build an infinite number of different structures some of which are recurrent patterns that can be used to analyze existing games and explore new concepts.

ACKNOWLEDGEMENTS

I would like to thank Stéphane Bura for being a great inspiration for the work presented here. Not only did he urge me to explore the diagrams I developed further using an interactive tool, his work, comments, and our discussions set me on the right track to discover some of the key concepts presented here. Furthermore I would like to thank Jacob Brunekreef for reading and commenting on a previous draft of this paper. Finally, I am grateful to the Hogeschool van Amsterdam for supporting my PhD research and allowing me to test this work with game design students.

REFERENCES

Adams, Ernest, & Rollings, Andrew. (2007). *Fundamentals of Game Design*. Upper Saddle River: Pearson Education, Inc.

Andrei, Neculai (2005) "Modern Control Theory: A historical perspective". Retrieved May 24, 2009, from http://prodlogsys.ici.ro/camo/neculai/history.pdf

Björk, S. & Holopainen, J. (2005) *Patterns in Game Design.* Boston: Charles River Media.

Bura, Stéphane (2006) "A Game Grammar". Retrieved May 24, 2009, from http://www.stephanebura.com/diagrams/

Church, Doug (1999) "Formal Abstract Design Tools" on Gamasutra. Retrieved May 24, 2009, from http://www.gamasutra.com/features/19990716/design_tools_01.htm

Chomsky, Noam (1957) *Syntactic Structures*. The Hague, Mouton Publishers.

DiStefano III, Joseph J., Stubberud. Allen R. & Williams, Ivan J. (1967) *Theory and Problems of Feedback and Control Systems*. New York, McGraw-Hill.

Dormans, Joris (2008) "Visualizing Game Mechanics and Emergent Gameplay". Paper presented at the Meaningful Play Conference, East Lansing, Michigan. Retrieved May 24, 2009, from http://www.meaningfulplay.msu.edu/program.php?presentation=40&type=paper

Fromm, Jochen. (2005) Types and Forms of Emergence. Retrieved September 8, 2008, from http://arxiv.org/abs/nlin.AO/0506028

Fullerton, Tracy (2008) *Game Design Workshop: A Playcentric Approach to Creating Innovative Games, 2nd Edition*. Morgan Kaufman.

Gamma, Erich, Helm, Richard, Johnson, Ralph & Vlissides, John (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, Addison Wesley.

Grünvogel, Stefan M. (2005) "Formal Models and Game Design". On GameStudies.org. Retrieved May 25, 2009, from http://gamestudies.org/0501/gruenvogel/

Juul, Jesper. (2005) *Half-Real, Video Games between Real Rules and Fictional Worlds*. Cambridge: The MIT Press.

Koster, Raph (2005) "A Grammar of Gameplay: game atoms: can games be diagrammed?" Presentation at the Game Developers Congres 2005. Retrieved September 8, 2008, from http://www.theoryoffun.com/grammar/gdc2005.htm

Kreimeier, Bernd (2002) "The Case For Game Design Patterns". Paper on Gamasutra. Retrieved May 25, 2009 from http://www.gamasutra.com/features/20020313/kreimeier_01.htm

LeBlanc, Marc (1999) "Formal Design Tools: Feedback Systems and the Dramatic Structure of Completion", presentation at the Game Developers Conference. Retrieved May 24, 2009, from http://algorithmancy.8kindsoffun.com/cgdc99.ppt.

Malaby, Thomas M. (2007) "Beyond Play: A New Approach to Games". *Games and Culture* 2007. No 2, 95-113.

Salen, Katie.& Zimmerman, Eric. (2003) *Rules of Play: Game Design Fundamentals*. Cambridge: The MIT Press.

AUTHOR BIBLIOGRAPHY

JORIS DORMANS is a lecturer, PhD student, and game designer based in Amsterdam, Netherlands. He has a Masters Degree in cultural studies but does not shun programming in C++ or teaching Action Script. His research papers and game related projects can be found online: www.jorisdormans.nl.

BOARD GAMES

On feature discovery process in board games

Rafał Łopatka
Institute of Control and Industrial Electronics
Warsaw University of Technology
email: `lopatkar@isep.pw.edu.pl`

Vasik Rajlich
Computer chess software developer
2007, 2008, 2009 World Computer Chess Champion[*]
`www.rybkachess.com`
email: `vrajlich@gmail.com`

KEYWORDS
Two person zero-sum perfect information games, computer game playing, feature discovery, chess-like games

ABSTRACT

In this paper we focus on application of statistics and basic probabilistic notion of Hellinger distance - a probabilistic measure of distance between two probability distributions - to a feature discovery process in context of two person zero-sum perfect information board games, in particular chess. A hypothesis aimed at short cutting feature discovery process is proposed and the result of its statistical verification is given. With the hypothesis proposed and it's practical implementation in the domain of chess we want to emphasize that statistics can be a meaningful approach to a feature discovery solution.

INTRODUCTION

Playing games by computer programs has become a very important subfield of artificial intelligence. Nowadays, it is hard to find a board game having no software implementation. Along with simply a great interest in the gaming field, observations lead to a number of paradigms, concepts and heuristics being developed. Following Pell (1993) we distinguish three main classes of approaches to a computer game-playing: knowledge engineering, database enumeration and machine learning. Each of the classes is characterized in short below.

- Knowledge engineering covers doubtless the most popular approach to a computer game playing, i.e. the game tree search technique and all it's modifications. Within this technique an evaluation function is applied to the leaves of a game tree which is constructed in order to make a decision on the optimal move. A knowledge-based search is another technique inside the knowledge engineering class. It is similar to the game tree search technique, but it differs is the way the optimal decision is sought, i.e. by reaching some intermediate subgoals, counter to game tree search where an ultimate goal is the subject of interest.
- Database enumeration is simply about gathering data on particular positions and best actions to take in context of a game. This is least ambitious technique among of all enumerated here, but very efficient on the other hand.
- Machine learning class of approaches is linked with two subclasses: learning from experts and unsupervised learning. The first of subclasses relies on some domain expert (human or computer agent) which is provider of the domain knowledge, the second one relies on self-play with no external support of a domain expert.

Our attention is focused on the knowledge engineering class of approaches, the game tree search technique in particular. One of the main practical problems in development of an expert system within that technique is representation of a domain specific knowledge. In most cases of a two person zero-sum perfect information games some form of an evaluation function is employed. The evaluation function must be able to assess a single state of the game (chess board, for example). With a domain specific knowledge incorporated inside the evaluation function the expert system is able to rank available options and make optimal decision in the sense of possessed evaluation function. The last few decades have shown that an evaluation function for the game tree search technique can be constructed in various forms and knowledge can be cumulated inside the evaluation functions themselves and in many ways. For example, neural networks have been successfully applied by Thrun (1995), Gherrity (1993), Baxter et al. (2000), Fogel et al. (2004) to learn evaluation functions for chess

[*]for championship tournament reports, results and other details on `Rybka` chess engine see `www.rybkachess.com`

and other domains. In Gherrity (1993) a set of general features applicable to many board games is used with a neural network trained to approximate position scores. The system is general in its application area, but the learning process advances slowly. The opposite takes place in Baxter et al. (2000) where many hundreds of features combined with a neural network allow to obtain a significant progress in playing strength of the system in a relatively very short time. A somewhat different solution from neural networks has been described in Fawcett (1993) with the system able to produce features from some basic domain specification is presented. The system named Zenith implemented in Prolog stores both, knowledge and generated domain features in the form of rules and clauses. That type of knowledge representation is very convenient for a human to read and analyse. See Bratko (2000) for a simple Prolog example of chess endgame knowledge database including some strictly defined rules and Morales (1994) for a more sophisticated set of rules induced within the rule induction framework. Evaluation functions based on features of that type are much easier to explore than neural networks. Zenith proved its efficiency with respect to two domains: the game of Othello and with the telecommunication networks. In Pell (1993) a new paradigm of learning is proposed, however, some elements of this paradigm are common with the concepts above, i.e. generation of evaluation function and the game tree search. The generation process is based (like in Fawcett (1993)) on subject domain specification.

Common among the systems mentioned above is the usage of features which allow to abstract chess (or any other subject domain) into the space of attributes easy to learn for the neural network or any other evaluation function approximator. Features are one of the most essential components of the computer game-playing system within a knowledge engineering approach. In our paper we intend to point to a different approach, a statistical based approach, to a feature discovery problem.

PRELIMINARIES AND ASSUMPTIONS

Throughout the paper we discuss the feature discovery problem in context of symmetric chess-like class of games with the game of chess as its representative member.

Symmetric chess-like games

An extensive definition of the symmetric chess-like games is given in Pell (1993). Here we restrict ourselves to adopt a short definition covering the main properties of these types of games and an illustrative example.

Definition 1 *A game is called a symmetric chess-like game (an SCL game) if*

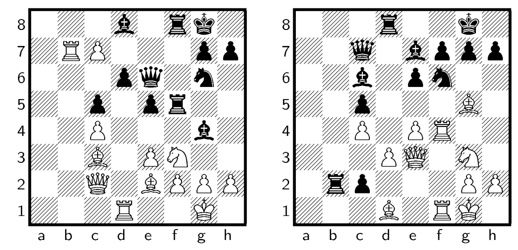


Table 1: Chess boards illustrating the definition of a SCL game. Left board is white to move, right one is black to move.

- *it is possible to present the entire set of rules for a given game from the perspective of one player only,*
- *exists some board symmetry $S(\cdot)$ such that for any board position p the relation $p = C(S(p))$ holds, where $C(\cdot)$ is a transformation interchanging ownership of the pieces and move turn.*

Example 1 *Consider chess positions shown in the table 1. In light of the definition 1 both positions are identical since in chess rules of the game are the same for both players and the symmetry transformation S is reflection with respect to the axis between 4th and 5th rank of the board. After application of the reflection to an arbitrary position of the presented boards in the table 1, changing colors of all pieces and changing side to move one obtains the other board.*

Different from traditional chess examples of SCL games might be: Giveaway Chess, Shogi, Chinese Chess, Checkers, Tic Tac Toe, Go-Moku. In general, Pell (1993) shows that every finite two player perfect information game G has its equivalent in an SCL games class G_{SCL}, where game equivalence means G and G_{SCL} have the same game trees and adjective finite means in the course of play only finite number of positions is visited.

The Hellinger distance

Let $\mathbb{P}$ and $\mathbb{Q}$ denote two probability measures on a measurable space Ω with σ-algebra $\mathcal{F}$. Let λ be a measure on $(\Omega, \mathcal{F})$ such that $\mathbb{P}$ and $\mathbb{Q}$ are absolutely continuous with respect to λ, with corresponding density functions p and q (for example, λ can be taken to be $(\mathbb{P} + \mathbb{Q})/2$ or can be the Lebesgue measure).

Definition 2 *The Hellinger distance between $\mathbb{P}$ and $\mathbb{Q}$ on a continuous measurable space $(\Omega, \mathcal{F})$ is defined as Basseville (1989), Birge (1985), Gibbs and Su (2002)*

$$\begin{aligned} H(\mathbb{P}, \mathbb{Q}) &:= \left[\frac{1}{2}\int_\Omega \left(\sqrt{\frac{d\mathbb{P}}{d\lambda}} - \sqrt{\frac{d\mathbb{Q}}{d\lambda}}\right)^2 d\lambda\right]^{1/2} \\ &= \left[\frac{1}{2}\int_\Omega (\sqrt{p} - \sqrt{q})^2\, d\lambda\right]^{1/2} \qquad (1) \end{aligned}$$

The square roots of densities $\sqrt{p}$ and $\sqrt{q}$ belong to the Hilbert space of square integrable functions L^2 Pollard (2003). This definition does not depend on the choice of the measure λ Gibbs and Su (2002), Pollard (2003). Hellinger distance can be used to estimate the distances between two probability measures independent of the parameters Sengar et al. (2008).
For a countable space Ω, measures $\mathbb{P}$ and $\mathbb{Q}$ on $(\Omega, \mathcal{F})$ are N - tuples $(p_1, p_2, \ldots, p_N)$ and $(q_1, q_2, \ldots, q_N)$, respectively, satisfying following conditions: $p_i \geq 0$, $q_i \geq 0$, $\sum_i p_i = 1$ and $\sum_i q_i = 1$.

Definition 3 *The Hellinger distance between measures $\mathbb{P}$ and $\mathbb{Q}$ on a discrete measurable space $(\Omega, \mathcal{F})$ is defined as Fannes and Spincemaille (2001), Gibbs and Su (2002), Salkind (2007), Sengar et al. (2008)*

$$H(\mathbb{P}, \mathbb{Q}) := \left[\frac{1}{2} \sum_{i=1}^{N} \left(\sqrt{p_i} - \sqrt{q_i} \right)^2 \right]^{1/2} \tag{2}$$

In some papers (Gibbs and Su (2002), Salkind (2007), Zolotarev (1984)) the factor of $\frac{1}{2}$ in definitions 1 and 2 is omitted. We consider definition containing this factor, as it normalizes the range of values taken by the distance. Some sources Borovkov (1998), Diaconis and Zabell (1982) define the Hellinger distance as the square of H. Defined by formulas (1) and (2), the Hellinger distance is a metric, while H^2 is not a metric, since it doesn't satisfy the triangle inequality.

Regardless of the fact that throughout the paper we utilize discrete Hellinger distance defined by formula (2), any other distance measure is applicable instead of Hellinger distance if it takes values from some bounded interval which can be normalized to the interval $[0, 1]$. Within such normalized intervals specification of significantly different distributions is obvious, counter to the all unbounded distance measures. However, application of unbounded distance measures is possible in the procedures specified below for the price of lower quality of separation.

PROBLEM STATEMENT

The feature discovery problem discussed in this paper is stated as follows.

Given an initial class K of objects and its distribution ρ_K, which characterizes a certain random process associated with each element of this class, the goal is to propose two sets $L, M \subset K$ different (suggestive) enough in some aspect to enable at least a single feature discovery.

For many years, clustering tasks of this kind have been solved using various methods, taking advantage of the features of objects from the set K. After specification of the value of all features, the set K is usually divided in accordance with the calculations performed using the values ascribed to individual features. In this manner, we obtain the criteria of assignment of elements of the set K to each of the classes L and M.

The main problem with solving the above stated problem is hidden at the stage the feature's determination from objects from the set K is made. Depending upon the complexity of the problem, for which clustering of the set K is conducted, the task of specification of the features of elements of this set may turn out either to be trivial or very complex. It is necessary to keep in mind that the set of features assigned to the elements of the set K cannot be a free one, it must ensure sufficient separation of the two new classes L and M. The fact that not every set of features is appropriate to obtain a successful solution of the clustering task, makes it all the more difficult.

When the problem is complex, specification of the features of positions by a person, who has limited experience in a given field, to which the elements of the set K belong, often becomes too difficult, making it necessary to seek for the assistance of an expert. Experts often refuse to participate in undertakings of this kind as they are burdensome. An additional reason, which is quite significant, is that a decisive majority of experts in fields characterized by a great degree of complexity of the problem are often unable to formulate precisely the reasons, for which they believe a given feature to be significant, when it is successfully identified. It can be seen very well in the case of chess. We often hear that a given position on the chessboard is good or bad, but the justification of such statement can be the so-called "presentiment" or "intuition", and in the best case, these statements are characterized by a low level of precision. Unfortunately, none of these concepts and statements can be effectively applied to solve the problem of clustering.

We will show below that the Hellinger distance (as a representative member of the entire class of probabilistic distance measures), apart from its other applications, can be useful during the clustering process, where set K can be described using distribution in accordance with the classical probability theory. It makes our work substantially easier, taking into account the fact that it is a simpler task to specify the features of two exemplary sets, which are different, than to create the features without having a warranty that they are sufficiently good to provide a division of elements from the set K into classes L and M.

GAME TREE SEARCH AS A RANDOM PROCESS

In Baum and Smith (1997) an interesting probability based approach to a game tree search has been proposed.

The authors suggest to model errors of some evaluation function used for scoring leaf nodes of the search tree in the form of distribution. The evaluation function error distribution (which has to be created empirically) tells us how likely a score of the given node can change by some given value after deeper search. To shortly illustrate the concept consider the real world example given in figure 1 and a few remarks on it that follow:

1. For KRNPkr class of chess positions Rybka's evaluation function does very well. For some given position taken out of this class its initial assessment has a nearly 75% probability to remain the same after deeper search and only 2.3% probability that deeper search makes the initial assessment worse.

2. Although `Rybka` does very well for this class of positions, there is still something to improve. It is easy to see that for relative error value of 500 there exist some spike in the subject distribution. This fact implies there is some subclass of KRNPkr class which could be assessed more accurately and, delving further, if such a subclass exists then it is likely that some property makes its elements (positions) different from the rest of the KRNPkr class members. With distribution we are able to identify very easily both sets without the necessity of knowing the difference between them. The difference is to be identified next, by examination of both sets leading in this way to a discovery of some new feature. This is a very primitive outline of the solutions we present in the further part of the paper.

Since the game tree search procedures like alpha - beta and its variations are deterministic an important question is still to be answered: is the game tree search process really random? We think it is due to two main reasons besides theoretical justification such as given in Baum and Smith (1997):

- Result of the game tree search in a real world chess engine depends much on hash contents, i.e. depends on recently performed searches by the engine. In practice two computers of the same hardware configuration can produce different scores for the same position after performing search with identical parameters.

- Both, hardware and software configuration of the computer running the engine affect the game tree search process. It is easy to imagine that the chess engine coexists with some other process consuming considerable computer resources much.

Further throughout the paper we assume that the random process is a game tree search process. On the other hand, to maintain the general context of the proposed solutions we continue using the term 'random process' to prevent ourselves from excluding any other random processes which could set the ground for making distributions.

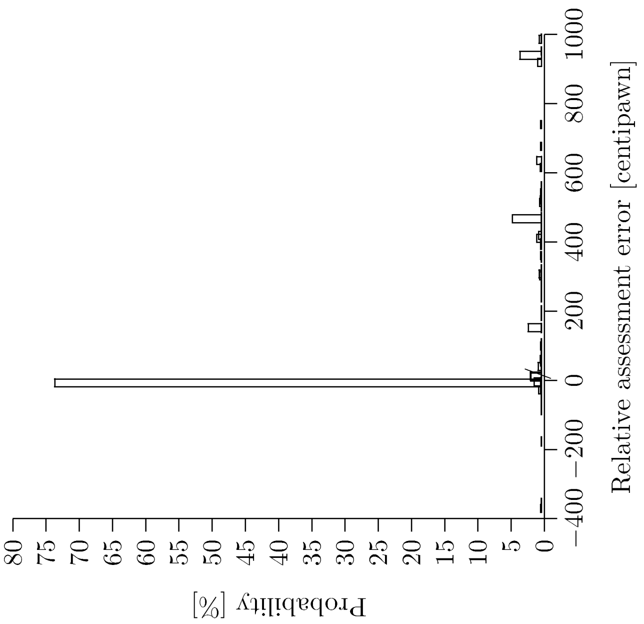


Figure 1: Distribution of assessment errors for the class KRNPkr (white: king + rook + knight + pawn vs black: king + rook). The distribution is constructed with 2000 samples produced by `Rybka 2.2n2` chess engine.

SOLUTION

We propose two solutions to the problem with a short discussion of their applicability. Both solutions according to a review paper by Kotsiantis et al. (2006) fall into a 'feature construction' category, although there is a little difference which makes our approach unique. We don't build our features on the basis of the features already discovered directly. Instead, in our approach, an evaluation function's properties manifested via assessment error distributions are exploited to produce good training data for feature discovery (construction).

Generating training data with Hellinger distance

Let us start with the following definition.

Definition 4 *We will refer to sets $\overline{M}$ and $\overline{L}$, for which the conditions specified below are met, as different sets within the set K in the context of Hellinger distance*

1. $\overline{M} \subseteq M \subset K$, $\overline{L} \subseteq L \subset K$.

2. $H(\rho_K, \rho_{\overline{L}}) \leq \alpha$, $H(\rho_K, \rho_{\overline{M}}) \geq \beta$.

3. $\alpha < \beta$.

where $\rho_{\overline{L}}$ i $\rho_{\overline{M}}$ are distributions associated with sets $\overline{L}, \overline{M} \subset K$.

A short remark is necessary with regard to the manner of construction of sets $\overline{L}$ and $\overline{M}$ in accordance with requirement 2 of definition 4. For the fixed values of α, β in compliance with requirement 3, the process of construction of both sets takes place in accordance with the following procedure.

Procedure 1 *Creation of the set X different from the given set K in context of the Hellinger distance.*

1. *Set the desired cardinality γ of the set $X = \emptyset$.*
2. *Draw element $k \in K$ and we perform the corresponding random experiment for it.*
3. *If $H(\rho_{X \cup k}, \rho_K) \leq \alpha$ ($H(\rho_{X \cup k}, \rho_K) \geq \beta$), we add element k to set X. Otherwise, go back to step 2.*
4. *If $|X| < \gamma$, then go to step 2.*
5. *Set X is the sought set.*

In fact the procedure 1 would be solution to the problem formulated in previous section if the Hellinger distance can be proved to be successful in generation of sets containing objects of different properties. To examine if the Hellinger distance has such a property and due to a statistical nature of the process associated with considered domain objects a statistical test is used in order to verify the hypothesis below.

The experiment, which is to show the usability of Hellinger distance for the clustering process, is conducted as follows

Procedure 2 *Let us have*

- *set K and distribution ρ_K assigned to it,*
- *a random process associated with the elements of set K,*
- *any clustering method based upon the features of the elements of the clustered set.*

1. *Using procedure 1 we establish sets $\overline{L}$ and $\overline{M}$ for the specified values of α, β, $\alpha < \beta$ and cardinalities of both sets.*
2. *We conduct the clustering of set $\overline{L} \cup \overline{M}$ using any method taking advantage of the features of elements belonging to set $\overline{L} \cup \overline{M}$.*
3. *We summarize the results obtained by checking whether the selected classical method allowed for differentiation of at least one cluster containing only (or mostly) the elements of set $\overline{L}$ or set $\overline{M}$.*

The existence of at least one such cluster can be treated as a confirmation of the hypothesis that the Hellinger distance may successfully suggest to the person comparing the elements of sets $\overline{L}$ and $\overline{M}$ a criterion of division without the necessity of a priori identification of any features. Formally the successful experiment is defined as follows.

Definition 5 *A single experiment, which constitutes a statistical test, will be considered to be successfully completed, if among the clusters received using the classical clustering method, we obtain a single cluster (or more), containing most of the positions from one of the sets $\overline{L}$, $\overline{M}$. In this case, "most" means the ratio of the number of elements originating from the two sets being 2 to 1 or greater, assuming that the number of elements of the dominant set in the cluster is not lesser than 10% of the cardinality of this set.*

Having established the criterion of success or failure of a single experiment in this manner, we can commence the statistical test χ^2. It is assumed that the cardinality of both sets $\overline{L}$ and $\overline{M}$ is 50. The values of parameters α and β varied in order to obtain the results for the widest possible spectrum of cases. For clustering using the features of positions from sets $\overline{L}$ and $\overline{M}$, we will use two algorithms: K-means and fuzzy K-means. More than ten experiments conducted first suggested the following hypothesis:

With probability of 75%, the Hellinger distance allows for the suggestion sets $\overline{M} \subset M \subset K$, $\overline{L} \subset L \subset K$ such that at least one subset of set $\overline{L}$ and $\overline{M}$ exists, allowing for identification of a condition making it possible to divide the set K into two parts.

In other words, the hypothesis says that it is reasonable (with the probability of 3/4) to apply the Hellinger distance to produce two suggestive sets and analyze their content to discover some feature making them different. Application of Hellinger distance for the generation of such sets makes the discovery more likely than exploring entire class K.

The calculations conducted for all 126 attempts (a single attempt is described by the definition 5) performed within the framework of the χ^2 test confirmed the hypothesis in 96%. In most applications of statistics for technical purposes, exceeding of the threshold of 95% in the χ^2 test is considered to be a sufficiently good result.

Generating training data with threshold

The process described below to identify chess features is similar to what is used in a commercial chess program Rybka. The process is less strict than the solution presented so far since it is heavily based on the domain's expert knowledge and sometimes even intuition.

Procedure 3 *On a regular basis, the following basic steps are carried out:*

1. *Start with some large group of positions, and assign to each position some type of automatically generated score. Mainly it is the difference in score calculated for two presumed depths.*
2. *Identify those positions in this group which have anomalous scores (i.e. the positions having score changed above some initially fixed threshold after deeper search).*
3. *Browse the anomalous position group and try to identify chess features which appear to be overrepresented in this group.*
4. *Confirm that the proposed features do in fact correlate with the generated scores.*
5. *Apply the extracted features to Rybka and confirm that program performance is improved.*

In theory, it is possible to skip steps 1-4 of the procedure 3 and devise chess features based purely on domain's expert understanding of the problem (and sometimes in practice this will do), but quite often the ability to browse groups of anomalous positions is a very useful source of ideas. The ability to confirm the speculated correlations is also a useful form of feedback.

It is interesting to consider the main differences between currently considered process and the process explained previously. During Rybka development process it is desired to have a sample of positions to browse which are as anomalous as possible, and no attention is focused on the control group. The approach previously presented differs from the currently described version in two ways:

- two groups (anomalous and control) are maintained at all times,
- the anomalous group is "mixed" - it contains many normal (non-anomalous) positions and only differs from the control group in the aggregate statistical sense.

At first glance, having a mixed anomalous group would seem to be counterproductive. From a domain expert point of view, only the anomalous positions are interesting. Non-anomalous positions in the anomalous group only serve as noise and make the feature extraction more difficult. The two approaches can be reconciled by the realization that the human domain expert has an intuitive, finely calibrated sense of the "normal" for his domain and can function quite well in the absence of strictly correct control information. The chess master doesn't need to see normal chess positions to identify unusual features in anomalous ones. Likewise, a medical doctor does not require a normal healthy patient in order to diagnose the sick one.

Discussion

The two methods for feature discovery described in this paper are essentially equivalent, if we consider that humans with strong domain expertise can neglect control data and simply focus directly on the anomalies. It is also possible to point out some more similarities. Both methods utilize evaluation function error distributions. The method based on the Hellineger distance does it explicitly while the threshold based method makes it implicitly. Collecting positions with the score change above some fixed threshold is equivalent to drawing positions from some evaluation function error distribution (which in case of the procedure 3 is unknown and there is no need to have it). Both methods relay on domain expert knowledge, but in the case of the Hellinger distance based method it is possible to eliminate an expert assistance in order to discover features. Some initial experiments with CN2 rule induction algorithm Clark and Niblett (1989) confirmed it is possible to discover sound features with this method having no support from a domain expert. The subject domain was king vs king and pawn chess endgame. The CN2 algorithm has been run on the training data obtained according to the procedure 1 and produced rules which could be transformed into a feature satisfying criterion stated in definition 5 (the feature discovered was relative position of the pawn with respect to its enemy king measured in ranks). This fact can not be easily skipped since it makes the Hellinger distance method a promising option for the research of an autonomous learning system. Its advantage with respect to a threshold based method is generation of two sets having at least one property making both sets different. Counter to a human expert, in a machine learning framework a single set for identification of new features is not enough. For the enablement of an autonomous system to discovering features and exploring the subject domain automatically two sets are necessary to make system aware what is normal and what is abnormal.

REAL WORLD APPLICATION AND EXPERIMENTS

The technique described with procedure 3 has been successfully applied for feature discovery in a chess program Rybka and has proved to be effective. Features generated in this manner (after refinement by a domain expert) allowed to obtain an expert system playing at master level. Here are some facts:

- Each feature gives an average raise of playing strength of the program by 0.5 Elo rating point.
- Not every feature discovered is finally incorporated into evaluation function. The ratio of rejected to accepted features is about 1:1. For example, `Rybka` undervalues the bishop pair in open positions. Browsing positions obtained according to the

procedure 3 shows this pretty clearly. Several formulations of this have been tried, all without success.

- The total number of features doubles every 2 or 3 years.
- Sometimes to discover a good feature just one position is enough. Typically browsing $50-100$ positions is the case.

The Hellinger distance based method has been successfully verified on some chess endgames, and one of those experiments is described below.

Example 2 *In this example, on the basis of procedure 2, we will conduct an experiment confirming the effectiveness of Hellinger distance as a tool supporting the work of expert in the clustering process. Set K will be the set of all chess positions type* `KRNPkrnp`*. Distribution ρ_K is established experimentally, the random process associated with each position is based upon specification of the difference in assessment of the position while searching it to the depth of 10 and 15 plies. To verify the quality of the division suggested by Hellinger distance, we will use the EM clustering method (expectation maximization). Acting in accordance with procedure 2, we obtain:*

1. *It is assumed that $\alpha = 0.2$, $\beta = 0.9$ and the cardinalities of both sets is $\gamma = 50$. We obtain sets $\overline{L}$ and $\overline{M}$ of position type* `KRNPkrnp` *such that $H(\rho_K, \rho_{\overline{L}}) \leq 0.2$, $H(\rho_K, \rho_{\overline{M}}) \geq 0.9$.*
2. *As a result of clustering performed for set $\overline{L} \cup \overline{M}$, we obtain 10 clusters. The elements of one of these clusters have been presented in table 2.*
3. *In order to answer the question whether the Hellinger distance has turned out to be an efficient tool supporting the expert in the clustering process, let us take into account the following facts:*
    - *In set $\overline{L}$, consisting of 50 elements, 20 have at least one pawn in front of the promotion line (line 2 for black pieces and line 7 for white pieces). Even a novice player will easily notice that a good feature differentiating the positions belonging to class K is* **having at least one pawn of any color before promotion line**.
    - *In the entire set $\overline{L} \cup \overline{M}$, there are 28 elements characterized by this feature.*
    - *Among the 10 clusters obtained as a result of application of algorithm EM, one is worth noting, for which all elements (presented in table 2) are characterized by the feature specified above.*

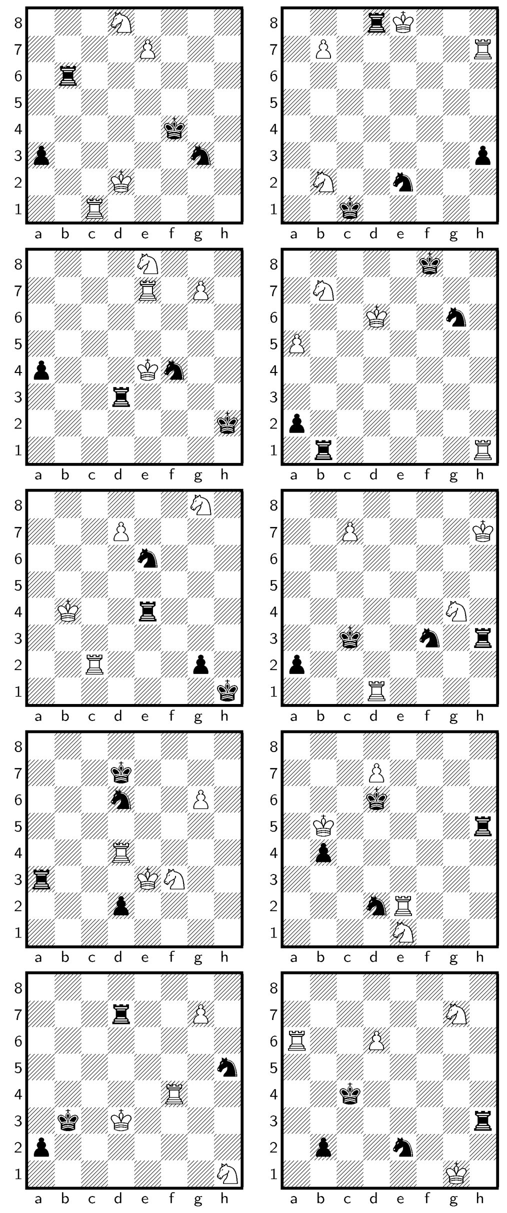


Table 2: One of the clusters obtained using the EM method. In each position, the move is to be made by white pieces. A typical feature of these positions is the presence of at least one pawn in front of the promotion line.

Thus we can conclude that the experiment was successful. On the basis of the set $\overline{L}$ it is possible to capture one of the features, which may serve as a basis for division of set K into two classes L and M, and clustering conducted using one of the classical methods confirmed the appropriateness of this choice, identifying the cluster presented in table 2. In this cluster, 8 out of 10 positions belong to set $\overline{L}$.

SUMMARY

In this paper we have presented two similar approaches to a feature discovery problem in board games with sort discussion of their main properties. Both methods can be beneficial for practitioners aiming at the creation of the game playing software. Selection of the method for a real world application depends on quality of the domain expert. For highly skilled experts the method based on procedure 3 is recommended. It is general and relies entirely on an expert's knowledge. The chess program Rybka is an example of its efficiency. On the other hand, the feature discovery method based on the Hellinger distance can be an attractive option for those who cannot cooperate well with a trained and experienced expert. It is much more labor-intensive solution than the threshold based method, but greatly relaxes demands in context of an expert's skills and proficiency.

REFERENCES

Basseville M., 1989. *Distance Measures for Signal Processing and Pattern Recognition. Signal Processing*, 18, no. 4, 349–369.

Baum E.B. and Smith W.D., 1997. *A Bayesian Approach to Relevance in Game Playing. NEC Research Institute, 4 Independence Way Princeton NJ 08540.*

Baxter J.; Tridgell A.; and Weaver L., 2000. *Learning to Play Chess Using Temporal Differences. Machine Learning*, 40, no. 3, 243–263. URL `citeseer.ist.psu.edu/baxter00learning.html`.

Birge L., 1985. *Non-Asymptotic Minimax Risk for Hellinger Balls. Probability and Mathematical Statistics*, 5, no. 1, 21–29.

Borovkov A.A., 1998. *Mathematical Statistics*. Gordon and Breach Science Publishers.

Bratko I., 2000. *Prolog Programming for Artificial Intelligence*. Addison Wesley. ISBN 0201403757.

Clark P. and Niblett T., 1989. *The CN2 Induction Algorithm*. In *Machine Learning*. 261–283.

Diaconis P. and Zabell S.L., 1982. *Updating Subjective Probability. Journal of the American Statistical Association*, 77, no. 380, 822–830.

Fannes M. and Spincemaille P., 2001. *The Mutual Affinity of Random Measures. eprint arXiv:math-ph/0112034v1.*

Fawcett T.E., 1993. *Feature discovery for problem solving systems*. Ph.D. thesis, Amherst, MA, USA.

Fogel D.B.; Hays T.J.; Hahn S.L.; and Quon J., 2004. *A self-learning evolutionary chess program. Proceeding of the IEEE*, 92, no. 12, 1947–1954.

Gherrity M., 1993. *A game-learning machine*. Ph.D. thesis, La Jolla, CA, USA. URL `citeseer.ist.psu.edu/47161.html`.

Gibbs A.L. and Su F.E., 2002. *On Choosing and Bounding Probability Metrics. International Statistical Review*, 70, no. 3, 419–435.

Kotsiantis S.B.; Kanellopoulos D.; and Pintelas P.E., 2006. *Data Preprocessing for Supervised Leaning. International Journal of Computer Science*, 1, no. 2, 111–117.

Morales E.M., 1994. *Learning Patterns for Playing Strategies*. URL `http://w3.mor.itesm.mx/~emorales/Papers/icca.ps`.

Pell B., 1993. *Strategy Generation and Evaluation for Meta-Game Playing*. URL `citeseer.ist.psu.edu/pell93strategy.html`.

Pollard D., 2003. *Asymptopia*. book in progress. URL `www.stat.yale.edu/~pollard`.

Salkind N., 2007. *Encyclopedia of Measurement and Statistics*. Thousend Oaks (CA): Sage.

Sengar H.; Wang H.; Wijesekera D.; and Jajodia S., 2008. *Detecting VoIP Floods Using the Hellinger Distance. IEEE Transactions on Parallel and Distributed Systems*, 19, no. 6, 794–805.

Thrun S., 1995. *Learning To Play the Game of Chess*. In *Advances in Neural Information Processing Systems 7*. The MIT Press, 1069–1076.

Zolotarev V.M., 1984. *Probability Metrics. Theory of Probability and its Applications*, 28, no. 2, 278–302.

FRACTAL TERRITORY GAME

Siao-Fan Siao, Lo-Wei Lee and Wen-Kai Tai
Department of Computer Science and Information Engineering
National Dong Hwa University
1, Sec. 2, Da Hsueh Rd., Shou-Feng,Hualien, 974,
Taiwan, Republic of China.
E-mail: {windiscovery|roywfsh}@gmail.com, wktai@mail.ndhu.edu.tw

KEYWORDS
Fractal, Board Game, Fractal Territory Game, Abstract Board Game

ABSTRACT

We devise a novel abstract board game, fractal territory game, using the concept of subdivision from fractal board game. In our game, two players place a game piece by turns in each round to occupy square territories. As all the four vertices of a square territory are occupied, players can get the scores based on the number of vertices they have occupied respectively. During a game, the player with disadvantage is allowed to subdivide the game board in a way that the subdivision of the quadtree does. The game ends if one of the termination conditions is satisfied. The continuity of lines is utilized to provide the infinite recursive subdivision for generating the sub-boards. The subdivision mechanism is used to balance the dominance of the leading player so that there are opportunities for players with disadvantages to catch up, thus provides more enjoyable gameplay. Also, we formulate the score calculation to allow electronic implementation.

INTRODUCTION

The concept of fractal has been widely applied. In Fractal board games (Browne 2006), the authors proved that fractal will increase the complexity of a game and further applied the concept of fractal on board games. Due to the fact that fractal needs infinite recursive subdivision, a continuous space is required to implement the concept. However, since a board must be discrete, it is impossible to implement a genuine frac-tal game. In Fractal board games, the authors applied the fractal concept on the Potential Y (Browne 2003) to devise an imaginary game called 'Fractal Py.' In 'Fractal Py,' the author used a continuous board to provide the space for the infinite recursive subdivision required. Though a board can be subdi-vided into infinite parts theoretically, points are used in prac-tice to quantify the area during the process of implementation; hence the limited levels of subdivision. Furthermore, if 'Fractal Py' is implemented on the infinite level, the scores of players in infinite sub-games must be calculated; though 'Fractal Py' can be assigned to a specific level of subdivision, how to determine the number of levels and the size of a board can be very challenging: the complexity of a game will become very low if the number of levels is set to a small value; on the other hand, if the number of levels is set too large, it is too compli-cate for players to determine their moves. To determine a fa-vorable level of subdivision is a dilemma.

In this paper, we devise a fractal-oriented territory-occupation game, called Territory Game (TG). In the beginning, the black player and white player will place a game piece at a vertex on the square board in turn. After four vertices of the square are occupied, the scores, for the black and white players respec-tively, will be calculated according to the number of their game pieces placed. It is clear that TG will results in a draw no matter how players place the game pieces. To im-prove the gameplay, we extend TG into Fractal Territory Game (FTG). In FTG, we recursively subdivide the board into sub-boards; when players place game pieces on the board, the game pieces are also placed on the corresponding sub-boards. This changes the fairness of game and increase the complexity. In FTG, the player with lower score is permitted to subdivide the board, and dominates the game. This not only remedied that leading players in traditional abstract board games will dominate until the game is over but also makes the game more interesting. Also, since the timing of subdivision is determined by players, they can decide the size of the board and level of subdivision.

We conclude the contributions of this paper as follows:

- A novel abstract board game is devised, utilizing the continuity of lines to provide infinite recursive subdivi-sion for sub-boards generation.
- A mechanism to balance the dominance of players in the abstract board game so that there are opportunities for players with disadvantages to catch up, increasing the gameplay.
- Formulation of the score calculation enables the elec-tronic implementation of a fractal game.

Due to the infinite recursive subdivision, we cannot prove the fairness of the proposed game yet. To visualize the whole game scenario is another challenge, since the limited resolution of monitor is no match with the growing size of the board.

In the following, we present related work of fractal games in section 2. Section 3 describes the rules and scores calculation of the proposed game in detail. Section 4 discusses the pros and cons of applying subdivision on abstract board games, and whether the fairness will alter or not after applying fractal on games. Finally, we conclude this paper and point out possible future work in section 5.

RELATED WORK

The proposed game FTG is a kind of abstract strategy games in board game categories. The following paragraphs will introduce: (1) the classifications of board games according to the chance elements, (2) the definitions and pros and cons of abstract strategy games, (3) the definitions and the characteristics of fractal-like games, and (4) the fractal game which currently is not implemented yet in practice, fractal Py (Browne).

Board game

Board games are the games that can be played on desks or any kind of flat planes. Board games can be classified according to chance elements.
Chance elements are uncertain events in games. They can be classified into two categories: the outcome uncertainty and the state uncertainty.
The outcome uncertainty is to embed the possibility elements in games, which makes players unable to predict the results accurately. The results of throwing a dice and the results of randomly delivering cards are two demonstrations of the out-come uncertainty. Games with the outcome uncertainty are called stochastic games. For example, in Monopoly, players cannot predict the outcomes of throwing a dice accurately. In this kind of games, luck usually plays an important role in increasing entertainments. On the contrary, the games without chance elements of the outcome uncertainty are called deter-ministic games. For example, Chess, Five in a row, Xiangqi, GO, Hex and Contract Bridge are all deterministic games.
The state uncertainty arises from the reason that players do not know their opponents' information, such as the locations and cards held in hands of their opponents'. Due to the lack of information, players have to make guesses or estimations. The games with the state uncertainty are called partial information games. For example, in Contract Bridge, players do not know exactly what cards their opponents may have. They have to bid and analyze to determine their strategy. This kind of games usually requires players to adopt certain strategies, to perform estimation and make inference. In the contrast, the games without the state uncertainty are called full information games. Chess, Five in a row, Xiangqi, Go, Hex, Monopoly, and Backgammon are some examples.

Abstract strategy game

Abstract strategy games are the games without chance ele-ments, with full information and certainty. Their rules are usually simple, and the outcomes are purely depending on players' intelligence instead of luck. A good abstract strategy game is an attractive competition of logic and intelligence, since the players have to do their best to compete throughout the game. The players usually have same advantages on the game board in the beginning. Once a player begins to domi-nate, this advantage often lasts to the end of the game, leaving subordinate players little chance. Due to this fact, player who starts to take the lead and makes no mistake in the rest of the game shall triumph.
We propose a mechanism of game boards subdivision to ease disadvantages of subordinate players, sparing them chances to reverse the situations. Furthermore, players can decide the appearance of boards, make the game more enjoyable.
There is another characteristic of abstract strategy games: the playing-first player has advantage usually. Take GO as an example, the mechanism called "Komi" is applied (extra points are added to the score of the playing-second player.) to balance the vantage. In FTG, allowing the score-losing players to subdivide the board is also to make equal between players.

Fractal-like game

A fractal-like game is the game in which players play on each level of sub-boards subdivided by fractal rules according to the same rules of the original game. Furthermore, after recur-sive subdivision, each level of sub-boards has self-similarity (Weisstein 1999). When playing fractal-like games, players will place their game pieces both on the original game board and on the corresponding recursively subdivided sub-boards. Therefore, after placing game pieces, the players must consid-er their next moves on the original board and on every addi-tional recursively subdivided sub-board simultaneously. This increases both the difficulty and the complexity of games. However, since fractal games require infinite subdivision, and fractal-like games are processing on discrete space of boards, the boards can only be subdivided to a certain degree. For example, Hex games (Meyers) are proceeding on every sub-board, thus players will place game pieces on the original board and on each level of sub-boards at the same time. As a result, players may not form their tactics merely on the first level of sub-boards, which greatly increases the complexity of games.

Fractal Py

In the field of fractal board games, fractal Py has not been implemented yet. Fractal Py is an extension from the game called Poten-tial Y with the subdivision to any level. Potential Y is a kind of transformation from the connection game "Y". Fractal Py uses area to provide infinite recursive subdivision. Although any number of levels can be assigned in fractal Py, the level of recursive subdivision and the size of boards must be deter-mined in the beginning of the game. However, if the level of recursive subdivision is set too small, the complexity of games will become low. On the other hand, it will be extremely dif-ficult for players to devise fine strategies if the level is set too large. In FTG, the players can dynamically determine the fa-vorable level of recursive subdivision and the size of boards.

FRACTAL TERRITORY GAME

In the following, we will introduce the rules of FTG. Next, we will explain the scoring formula of FTG in detail, for the score computation of each sub-board subdivided during the game.

Game rule

Players
The black player and the white player, using black game pieces and white game pieces respectively.

Rules
I. In the beginning of a game, players of both sides have

to set the target score and the gap score.

II. In each round, players of both sides take turns to place a game piece of their own color at an unoccupied vertex (cross-point of the board lines).

III. When the four vertices of a square are all occupied by game pieces, the players of both sides can calculate their scores according to the number of their game pieces on the vertices. In addition, the initial game board and every subdivided square sub-board must be taken into consideration when players are calculating their scores.

IV. During the game, the player with lower score can choose to subdivide the game board in beginning of his turn.

V. (Termination conditions) The game will be terminated when:

   A. All vertices of the game board are occupied with game pieces, or
   B. Any player reaches the target score, or
   C. The difference of scores between the players of both sides is larger than or equal to the gap score.

Game board

Initially, the game board is 3x3, i.e. there are 3x3 vertices. In the first level, there is one 3x3 board, and in the second level, there are four 2x2 boards (2x2 cells), as shown in Figure 5. The board will dynamically change according to whether players subdivide the cells or not. Figure 5(b) demonstrates the level of the game board using the quadtree structure.

Board subdivision

Subdividing one cell generates four (2x2) next level cells (smaller squares), as shown in Figure 6.

Winning condition

When one of the termination conditions of the game is satisfied, the player with higher scores is the winner.

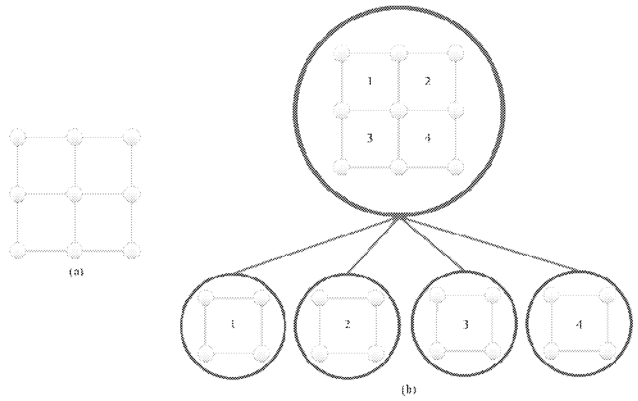


Figure 5: (a) The game board is 3x3 initially. (b) The level demonstration of the initial game board by using the quadtree structure: the root level (first level) is the 3x3 board and four 2x2 boards (quadrants) are in the second level.

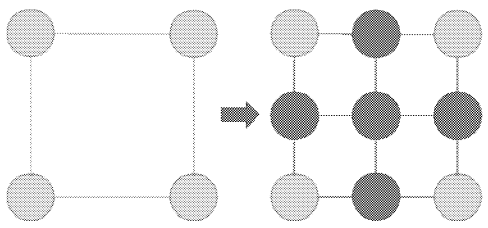


Figure 6: Subdivide a cell to generate four smaller square cells (sub-boards).

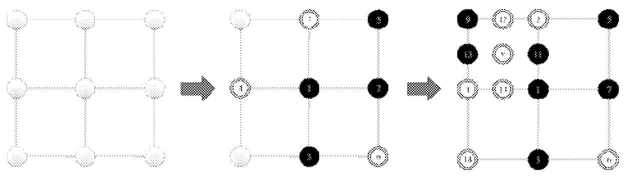


Figure 7: In the beginning of the game, the game board is 3x3. The numbers represent the ordering to place the game pieces; When the white player is in his 8th turn, he subdivides the cell at upper left corner.

Samples of playing

In Figure 7, the players take turns to place game pieces. When the white player (2nd-playing) is in his 8th turn, he must perform subdivision (the cell on the upper left corner is subdivided,) and then place his game piece. After the black player (first-playing) finished his 13th turn, all vertices on the game board are occupied with game pieces. Obviously, the game terminates because one of the termination conditions is satisfied. In the end, there is a tie for the players of both sides (black 18 – white 18.) Figure 8 demonstrates the game board after the game is terminated by using quadtree structure. We can calculate the scores of both players on the square of each node in the quadtree according to Rule III.

Score calculation

After the vertices of a square are all occupied, the players of both sides can get scores according to the numbers of their game pieces on the vertices. The scores of both players must be calculated on each subdivided square. We conclude several formulas to facilitate the score calculation of the board.

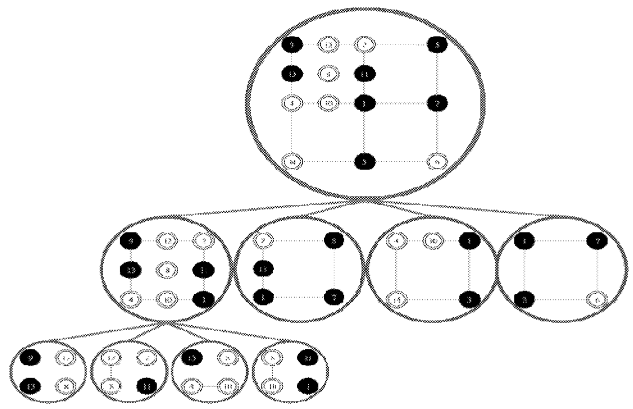


Figure 8: In the sample of playing, we use the quadtree structure to describe the game board. The scores of both

players will be calculated on the square of each node in the quadtree.

We can represent the game board by the quadtree structure. Let the board size be (n+1)x(n+1), and the root level (first level) be 0 and $h=\log_2 n$ is the height of the quadtree. We have the following formulas to calculate the scores for both players:

$$\mathrm{SumOfScore}(c) = \sum_{i=0}^{n}\sum_{j=0}^{n} S(i,j) \times \mathrm{Color}(i,j,c) \tag{1}$$

$$\mathrm{ScoreOfPiece}(i,j) = \sum_{l=0}^{h} S_{\mathrm{Index}}(i,j,l) \times \mathrm{Window}(i,j,l) \tag{2}$$

$$\mathrm{Window}(i,j,l) = \begin{cases} 1, if\ i\ mod\ 2^{h-l} = 0, j\ mod\ 2^{h-l} = 0 \\ 0, \qquad otherwise \end{cases} \tag{3}$$

$$S_{\mathrm{Index}}(i,j,l) = \sum_{o_i=0}^{1}\sum_{o_j=0}^{1}\left|\prod_{\delta_i=0}^{1}\prod_{\delta_j=0}^{1} \mathrm{Piece}[i+(o_i-\delta_i)\times 2^{h-l}][j+(o_j-\delta_j) \times 2^{h-l}]\right| \tag{4}$$

$$\mathrm{Piece}[i][j] = \begin{cases} 1, & white \\ -1, & black \\ 0, & otherwise \end{cases} \tag{5}$$

$$\mathrm{Color}(i,j,c) = \begin{cases} 1, & if\ Piece[i][j]\ is\ equal\ to\ c \\ 0, & otherwise \end{cases} \tag{6}$$

In equation (5), Piece[i][j] stands for the game pieces of both players on the game board. In equation (2), ScoreOfPiece(i,j) represents the earned scores of Piece[i][j] on each level when a game piece is placed on Piece[i][j]. In equation (2), "l" stands for the level of the quadtree. c represents the colors of both players: "1" stands for the white; "-1" stands for the black. In equation (4), $S_{\mathrm{Index}}(i, j, l)$ denotes if the four vertices of a square are all occupied by game pieces. In equation (3), Window(i, j, l) denotes the group of levels of sub-boards where the game piece of Piece[i][j] belong. In equation (1), SumOfScore(c) denotes the summation of the scores, for each player, that every game piece can earn.
Let us take an example to explain the formulas of score calculation. Figure 9 stands for the game process. In Figure 9, the numbers of game pieces denote the ordering to place these game pieces. The numbers in the left and the top sides of the figure are used to mark the coordinates of a game board. The number in the top-right corner of a game piece denotes the score that the game piece currently earns. Additionally, the game board has been subdivided into three levels; there are total: one 5x5 board, four 3x3 boards (bold-lined), and 16 2x2 boards (thin-lined). For example, after the game piece was placed by the white player in his first turn (Piece[0][2]), according to Window(i, j, l), we can determine that the game piece locates both in the 2x2 (thin-lined) board and in the 3x3 (bold-lined) board. From the 2x2 board, the game piece earns:
$|1 \times 1 \times 0 \times 0| + |0 \times 1 \times 0 \times 0| + |(-1) \times 1 \times 1 \times 1| + |0 \times (-1) \times 0 \times 1| = 1$

and from the 3x3 board, the game piece earns:
$|1 \times (-1) \times 0 \times 0| + |(-1) \times 1 \times 0 \times 0| + |(-1) \times 1 \times 1 \times (-1)| + |(-1) \times (-1) \times (-1) \times 1| = 2$
Hence, the position, Piece[0][2], can earn three points in total.

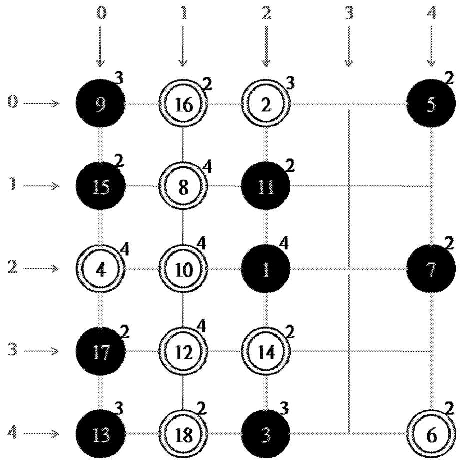


Figure 9: An example of score calculation for all game pieces on the 5x5 board.

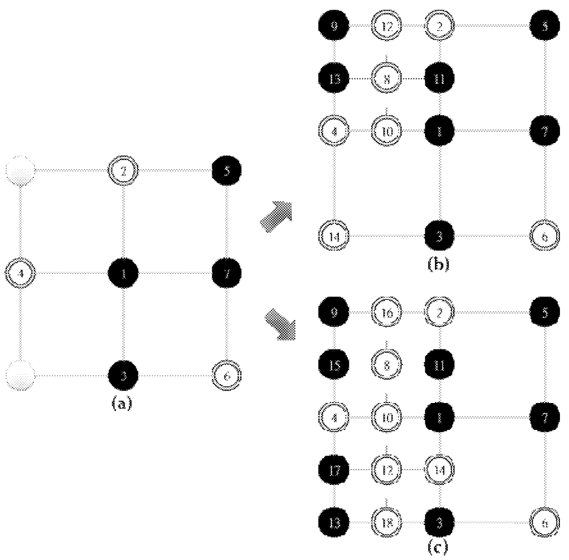


Figure 10: Advantage of Board Subdivision. (a) The playing-first player (black) is temporarily with advantage at his 7th turn (black 6 – white 2). (b) The game is a draw (black 18 – white 18), after one subdivision taken by the playing-second player (white) at his 8th turn. (c) An alternative play which shows the result (black 25 – white 27) after twice subdivision by the playing-second player (white) at his 8th and 12th turns respectively.

DISCUSSION

We discuss about (1) the benefits of using board subdivision, (2) an alternative way to play fractal games, and (3) the enjoyableness and the fairness of games.

Advantage of board subdivision

After subdivided, several sub-boards (cells) will be generated. By the rules, placing a game piece on the board means placing game pieces on all the corresponding sub-boards. Hence, players may take the chance of board subdivisions to make their own game pieces occupy more vertices on the sub-boards. That is, the players have to keep all the sub-boards in mind when placing their game pieces. This increases the interest and complexity of the game.

For FTG, the player with lower score has the right to subdivide the boards. The player's game pieces on the board may occupy the territory of both original board and the sub-boards. This provides the player with disadvantage a chance to tie or even lead the game, which increases the enjoyableness of the game.

As Figure 10(a) shows, both players are doing their best from 1st to 6th turns in order to win the right to subdivide the game board. However, in the 7th turn, wherever the playing-first player places his game piece, his scores will be higher, leaving his opponent chance to subdivide the board. In Figure 10(b) , the playing-second player chooses to subdivide the square at the upper left corner because he has two game pieces in the square territory and his opponent has only one game piece there. The playing-first player then picks the unoccupied vertex, where he can get the highest scores at the time, to place his game piece, while the rest of the unoccupied vertices earn the same scores. Finally, after the playing-first player finishes his 13th turn, there is no unoccupied vertex left, hence, the end of the game. This results in a draw (black 18 – white 18.) In Figure 10(c), we demonstrates an alternative play which the playing-second player subdivides the square at the upper left corner in 8th turn, and subdivides the square in the lower left corner in 12th turn. From the 8th turn to the 18th turn, both of the players adopt the best strategies to play the game. After the playing-second player finishes 18th turn, there is no unoccupied vertex left, hence the the playing-second player wins (black 25 – white 27.)

Gameplay

The termination condition B (when any player's scores reach the target scores) and C (when the difference between two players' scores reaches the gap scores) provide players more possibilities to think about different winning strategies, which further enhance the gameplay. Of course, players can make an agreement to change the game rule about the chance of game board subdivision such as throwing a dice or drawing a card. FTG will then transform into a stochastic game, providing alternative ways to play.

Fairness

Herik et al. provided a definition for the fairness of a game: if a game is a draw game and the possibilities for two players to make mistakes are equal, it is a fair game (Herik et al. 2002). However, it is difficult to mathematically examine if the possibilities for two players to make mistakes are equal; this is due to the fact that whenever a new strategy is created, the calculation of the possibilities of making mistakes will be different, which will further influence the fairness. Hence, it is difficult to mathematically prove the fairness. On the contrary, it is much easier and plausible for us to prove the unfairness. According to the definitions of the fairness and unfairness of Connect6 by Wu et al. (Wu et al. 2005):

- Definite unfairness: A game is definitely unfair if it has been proved that one player wins the game definitely. For example, Go-Moku is unfair because the playing-first player definitely wins.
- Monotonic unfairness: A game is monotonically unfair if it has been proved that one player does not win the game, but it has not been proved for the other player. For example, for Connect(k,p,p) or Connect(m,n,k,p,p), White side cannot win, based on the so-called strategy-stealing argument. Thus, Connect(6,1,1), Connect(7,1,1) and Connect(6,2,2) are all monotonically unfair, since Black side has not been proved to win or tie in these games.
- Empirical unfairness: A game is empirically unfair if most players, in particular professionals, have claimed that the game favours some player. For example, Go-Moku is empirically unfair, since most professionals claimed that playing-first player would win.
- Potential fairness: A game is considered potentially fair if it has not yet been shown or claimed to be definitely unfair, monotonically unfair, or empirically unfair.

According to the definition of the potential fairness, if a game is potentially fair currently, does not mean that the game will be potentially fair in the future. Nevertheless, the longer a game can remain potentially fair, the higher possibility that it becomes a fair game. Since FTG has yet been proved as definite unfairness, monotonic unfairness, or empirical unfairness, FTG currently remains as a potentially fair game based on the above reasons.

CONCLUSION

We have introduced a fractal board game which players can dynamically make decisions to subdivide the game board so that the leading player is replaceable improving the complexity and enjoyableness of the game.

In the future, we will prove the fairness of FTG. We would design an online version of FTG, which allows more players to play via the internet. Data can be collected then, to see if FTG is empirically unfair. Also, we may apply Strategy-Stealing argument to prove if the monotonic unfairness exists in this game. After the game board keeps being subdivided, it may grow larger and larger. To visualize the complete game board is another problem to solve.

REFERENCES

Browne, C. 2006. "Fractal board games", Computers & Graphics, Vol. 30, Issue 1, 126-133.

Browne, C. "Py: Potential Y", http://members.optusnet.com.au/cyberite/py/py-1.htm

Weisstein, E. 1999. "Concise encyclopedia of mathematics". Boca Raton: CRC Press.

Meyers, S. "Quadrant Hex", http://home.fuse.net/swmeyers/quadrant.htm

Herik, H. J. van den; J.W.H.M. Uiterwijk; and J.V. Rijswijck. 2002. "Games solved: Now and in the future." *Artificial Intelligence*, Vol. 134, 277-311.

Wu, I-C.; D.-Y. Huang; and H.-C. Chang. 2005. "Connect6", ICGA Journal, Vol. 28, No. 4, 234-241.

SERIOUS GAMING

LIFESIM: SOFTWARE FOR HEALTH SCIENCE

Charles C Earl and Daniel Fu
Stottler Henke Associates, Inc.
Suite 360, 951 Mariners Island Blvd
San Mateo, CA 94404
E-mail: {earl,fu}@stottlerhenke.com

Isobel Contento and Pam Koch
Health and Behavioral Studies
Teachers College, Columbia University
525 West 120th Street, New York, NY 10027
E-mail: {contento,pkoch}@exchange.tc.columbia.edu

Ana Islas
E-mail: ana.islas@gmail.com

Erin Hoffman
E-mail: erin.n.hoffman@gmail.com

Angela Calabrese Barton
College of Education
Michigan State University
Erickson Hall, East Lansing, MI 48824
E-mail: acb@msu.edu

KEYWORDS
Serious Gaming, AI, Design, Methodology

ABSTRACT

Stottler Henke Associates, in collaboration with Teachers College Columbia University, is building an instructional software game called LifeSim that provides middle school students motivations, skills and sense of personal agency about health and nutritional science needed to make healthy diet choices in their day to day life. The software is being developed from the content and activities of a middle school science module, Choice, Control & Change (C3) that is part of the Linking Food and the Environment (LiFE) Curriculum Series. The LifeSim uses aspects of social gaming such as avatar role-playing, chat interactions, minigames, and a badge reward system. In particular, LifeSim makes extensive use of intelligent agents that adapt the play to the experience, skill, and eating behaviors of the player. This paper discusses the design of LifeSim and methodology that will be used to test it. (*The project described was supported by Award Number R44RR019780 from the National Center For Research Resources. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Center For Research Resources or the National Institutes of Health.*)

GOALS AND BACKGROUND OF LIFESIM

Our main goal is to equip students with the knowledge, motivation, skills and competence or sense of personal agency about health and nutrition that will enable them to make healthy diet and exercise choices in an environment where the choices are complex. We employ a computer game format to achieve this. The need for this software is created by the widespread and mature use of gaming technologies for educational purposes on the one hand and the need for instruction that specifically addresses the growing crisis in child nutrition on the other (Ogden et al. 2008).

The Health Crisis

Although the rates of childhood overweight and obesity seemed to have plateaued in recent years, the overall prevalence remains at an all time high, at 31.9% (Ogden et al. 2008). The link between obesity and other serious health problems such as stroke, heart disease, many forms of cancer, and depression has been established. These developments are linked in part to dramatic changes in the food and activity environment: the increase in serving sizes, the presence of soft drinks in schools, the prevalence of sedentary lifestyles. Although environmental changes are definitely called for, educational interventions remain a crucial part of the solution. If children can acquire the knowledge, motivation, skills and sense of personal agency needed to make healthy choices for food and activity regardless of the environment, they stand to improve their short and long-term quality of life.

The LiFE Curriculum

The Linking Food and the Environment (LiFE) Curriculum Series, developed at Teachers College Columbia University is an inquiry-based science and nutrition education program for 5th—7th grades that provides a needed holistic understanding of food, nutrition and the environment. The LiFE Curriculum Series has been developed through Science Education Partnership Awards (SEPA), a program of the National Center for Research Resources of the NIH. LifeSim is being developed from the content and activities of the Choice, Control & Change (C3) module of the LiFE Curriculum Series. The process of developing C3 used a systematic theory-based procedural model for designing nutrition education interventions (Contento 2007). The question students study throughout the C3 module is: ***How can we use scientific evidence to help us make healthy food and activity choices?*** C3 is divided into five units, each with its own driving question. UNIT 1: QUESTIONING OUR CHOICES introduces the curriculum by having the students really think about what choices they have pertaining to what they eat and how active they are. UNIT 2: BODIES IN MOTION introduces key concepts about the body and energy and explores the question: ***How can we make sure that we get the right amount of energy to help our bodies do what we want them to do?*** UNIT 3: MOVING TOWARD HEALTH teaches students to examine their own eating and activity behaviors as they study the question:

How can we use personal data to help us make healthful food and activity choices? UNIT 4: BODY SCIENCE gives students a deeper understanding of the science behind the C3 goals as they investigate the question: ***Why are healthful food and activity choices important for our bodies?*** UNIT 5: MAINTAINING COMPETENCE provides students with the understandings and skills to maintain a healthy lifestyle as they explore the question: ***How can I maintain my skills as a competent eater and mover?*** The interactions of the LifeSim game are structured to teach the knowledge and skills present in these units.

PRIOR WORK IN GAMES FOR HEALTH

There have been several efforts to develop games that teach nutrition or promote health. Most of these efforts have involved the development of games using the role-playing and adventure game motifs. Squire's Quest! (SQ) (Cullen et al. 2005), developed by the Children's Nutrition Research Center of Houston's Baylor College of Medicine, is an adventure game set in a medieval kingdom. The player's goal is to help defeat creatures who are destroying the kingdom's fruit and vegetable crops. To do this they must prepare healthy meals for the king and queen, set and meet personal goals to eat more fruit, juices, and vegetables (FJV) at home and school, answer questions about the nutritional content of various food, and battle vegetable-destroying adversaries. Intervention group students on average increased their consumption of FJV by 1 per day.

Escape from Diab (Thompson et al. 2007) was also developed by Children's Nutrition Research Center with software development done principally by ArchImage. Escape from Diab focuses on a behavioral objective of getting the player to set and achieve goals related to diet (e.g. consume 3-5 servings of fruit per day) and exercise (e.g. 60 minutes of physical activity per day). The player finds themselves in a world in which the children are permitted to eat junk food all day and exercise is discouraged. The player has to alter behavior of other characters in Diab by leading by example by adopting diet and exercise goals. The player leads a group of other children in an escape from Diab.

While Escape from Diab provides support for encouraging players to *increase* consumption of healthy foods and amount of exercise; support for *decreasing* unhealthy behaviors, such as the amount of sweetened beverages consumed, is not explicitly present. In LifeSim, we have incorporated goals aimed at decreasing consumption of foods known to adversely impact health. While Escape from Diab is adaptive in the sense that diet and exercise goals are tailored to the self-reported behavior and profile of the student; there is no notion of a student behavioral model which could be used to further customize instruction. LifeSim uses existing Stottler Henke Associates artificial intelligence technology to both model the student's attitude and knowledge state and to adapt the activities presented to the student.

Hungry Red Planet (HRP) is another role-playing game which provides a comprehensive and scientifically accurate computer based learning environment for nutrition education. Children (ages nine and above) make menu choices for a simulated colony on Mars. An evaluation showed significant improvement in the students' ability to use food labels and the Food Pyramid.

The motif of virtual companion management and role-playing we believe opens up prospect for interaction which can extend the strengths of successful games for health, and also provide an experience that is engaging and exciting in its own right.

Multiplayer interaction will allow us to leverage this finding. For example, students could add restaurants to a shared section of the game world, and win points on the basis of: 1) how the meals impact the health of their peers creatures; 2) how their peers evaluate those menus in terms of health and taste.

LifeSim draws upon several modes of interaction that have enabled success in the HRP and SQ games, as well as feature that have not been heretofore examined in the context of health promoting games. These are: 1) support for interactions that mirror those of the real world (e.g. guiding the player on how to ask for more fruits as a lunch snack); 2) deriving the content and activities in the game from an existing, successful curriculum; and 3) correlating game reward structures and activities with the adoption of healthy behaviors. To implement this system, we will use virtual companion care as our central game motif, while drawing upon world-building motifs and multiple-player interaction.

LIFESIM DESIGN

We adopt the game motif of *virtual companion care,* involving a set of virtual creatures that perform tasks and complete adventures. Specifically, the story thread is that the player has been selected to be the camp counselor for creatures that exist in a faraway world.

There are four reasons behind the selection of this motif. First, findings from a preliminary investigation indicated that players were interested in having characters—people or "creatures" that they could identify with; and students and teachers reported that it would be useful to be able to shift between both individual and "city-level" view of the game. Because maintaining the health and happiness of the virtual creatures is a central goal of the game, the C3 content is relevant to success and can be explored in depth as the player progresses though the game. Second, monitoring the health of creatures is one central aspect of the game, thus integrating the C3 skill of personal diet monitoring could be reinforced. Students must be aware of both the health of the creatures that they manage, as well as that of the others at the camp. Third, the motif of virtual companion care is tremendously popular with students close in age to our target group as evidenced by sales of similar games to this age group (e.g. Animal Crossing) and in the popularity of social games such as NeoPets. We develop wrinkles upon this theme (e.g., the player is a camp counselor for creatures in age and behavior mirroring that of slightly younger humans) that are specifically tied to the mostly 5th—7th graders that we are targeting.

The creatures that attend the camp are designed to be in the 8-10 year old age range (younger than the students that will be involved in the study and our target audience) and exhibit the behaviors of human children in that age range. Although the physical appearance of the creatures is "vaguely humanoid", their physiology generally mirrors that of humans, including some of the diet related diseases that can befall them. Effects of diet upon health manifest faster in the creatures than in humans.

The creature campers are not passive beings. For example, the player can suggest what they should eat at the camp cafeteria, but must have an appreciation of their taste preferences and eating habits to make inroads in their behavior. Merely determining a balanced diet is not enough—the player has to confront one of the central themes of C3 notion of diet change.

The types of food available to the creatures closely resemble those available on Earth both in appearance and nutritional content. The character that the player assumes in the game is that of a 13 year old (towards the higher end of the age range of our target audience). Thus, role-playing the character comes with an appeal of "coolness". There are multiple goals that the player pursues during the game. They have to keep the "creature campers" that are in their group healthy and happy; there are a number of activities in which the student's "campers" can compete with other creatures in the camp; and the player can solve puzzles involving topics related to C3 curriculum and thus accumulate points. Points allow the player to accumulate "virtual money" that allows them to buy things for their campers, and eventually allows them to add progressively more interesting things to the world. Hand-in-hand with points is the notion of "creature admiration": the more the students are liked and respected by the creatures, more creature campers will join their groups. Having more campers in one's group enables a player to undertake more projects. Improving the health and admiration of the campers also allows the player to advance in responsibilities. The goal of the game is to progress to the title of Camp Sensei–a title bestowed upon the most expert of the camp counselors.

Winning challenges allows the creatures to get points and gives the player more "world money" for getting items for the creature. The healthier and happier the creature is, the more likely it is to do better in challenges. As the creatures become content, the "fame" of the student increases, thereby drawing other creatures to be part of the group.

The player begins with resources to purchase food for their creature campers. However, the player has to find the balance between the physiological preferences of the creature, their existing eating habits and the nutritional value of the food that is available. Reflecting real world choices, the player initially has to purchase and recommend to the creature what to eat. That is the creature camp cafeteria and adjoining restaurants and snack shops offer a variety of food options that the creature campers must negotiate if they are to be healthy enough to participate in the various camp activities. As the player acquires more points, they are able to use their resources to construct places in the world. For example, build a gym for the creature and its friends to work out, or start a single restaurant for the creatures.

One issue that the game must address is linking the behavior, physiology, and health of creatures to the student understands of their own health and the impact of diet upon it. There are four ways in which we address this. First, the game will make the point that "your creature needs YOU to be healthy in order to be healthy." Students will be asked about the food they would have selected, will be given health goals as takeaways that they can work on offline (e.g. drink at least five cups of water today). Second, discussion materials will be developed for teachers that draw connections between the health and behavior of the creature and the player. These will be presented by and discussed with the teacher at least twice during the course of four baseline sessions in the school. Third, the game will include puzzles related to C3 curriculum–for example energy balance—that the student will have to take to win additional resources and get items for their creature. Finally, there will be parent-student activities during the course of the formative and summative evaluations of LifeSim. Pre-activities for these will require parent and student to bring in a favorite recipe and food log for the previous day to the session.

LIFESIM ARCHITECTURE

There are three software products created by Stottler Henke for simulation and serious games (SimVentive (Fu and Ludwig 2007), SimBionic (Fu and Houlette 2002)) and intelligent training system (FlexiTrainer (Ramachandran et al. 2004)) development. These tools will be leveraged for LifeSim. **SimVentive** is a comprehensive software toolkit that gives simulation and serious games designers the means to create training scenarios and to define the behavior of characters and objects that populate those scenarios. Using a visual approach, SimVentive supports the rapid development of user interfaces, game objects, and behavior to create realistic single-player and multi-player training systems. It explicitly includes three different authoring modes which enable developers to create a serious game from scratch while still providing a way for instructors and subject matter experts to customize a deployed game. Authoring assistants check scenarios for errors and provide wizards to help with authoring tasks. SimVentive also encourages the reuse of game content with tools for building and managing libraries of scenario modules.

SimVentive incorporates its earlier predecessor, SimBionic, for supplying game behavior logic. **SimBionic** is a behavior authoring tool which enables an instructor or tactical expert to create entity behavior using a graphical "drag and drop" interface to quickly build complex behavior. Once behaviors have been created, they can be easily modified and re-used for other scenarios. SimBionic behaviors are implemented as Behavior Transition Networks comprised of states and transitions. SimBionic extends the usual notion of finite state machines by making it possible for states to refer to other Behavior Transition Networks hierarchically, to define modular behaviors that can be combined powerfully. SimBionic software also provides four extensions that increase the power and expressiveness of the basic engine: global and local variables interrupt transitions, "blackboards" for sharing knowledge among Behavior Transition Networks, and polymorphic indexing for run-time selection of behaviors.

FlexiTrainer (Ramachandran et al. 2004) is our student modeling and instructional behavior toolkit. It provides significant flexibility in Intelligent Tutoring System (ITS) development. Each agent responds to a set of instructional goals when some pre-conditions are met. The instructional planner coordinates these agents, selecting the optimal agent to satisfy a goal based on the pre-conditions, the tutor's state, and the student's state.

We intend to leverage these three core tools to build LifeSim. Each is composed of an authoring tool plus a deployable "engine" to make LifeSim operational. There are three types of applications areas: student, instructor, and developer.

Student Applications: These are versions of the already-existing SimVentive player which will be adapted to run inside of a social network platform – we are currently using the Ning platform, but other systems such as Facebook could be supported. This system loads scenario files from the Backend and enables students to play the LifeSim game and later upload their results to the Backend for teacher viewing. This application interprets creature data, and enables students to interact with each other. Interactions include minigames, adding material to the creature's world, and visiting the creatures of other players.

Teacher Applications: We anticipate the use of applications for customization and student performance review. It will enable the teacher to customize simple elements of scenarios.

Developer Applications: The primary development applications are our three core tools plus art asset and media generation tools such as Photoshop or Maya.

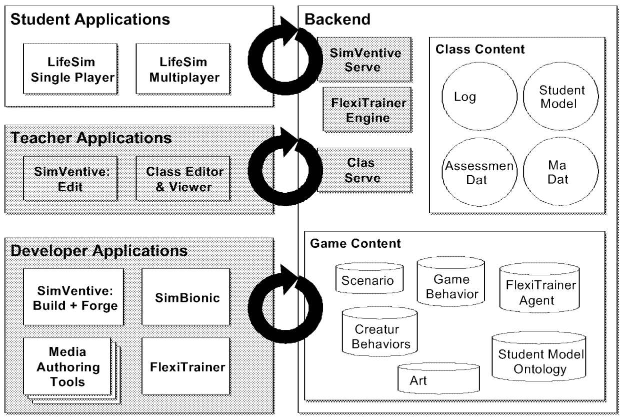


Figure 1: LifeSim Architecture

EVALUATING LIFESIM

We will conduct the evaluation of the LifeSim software in three steps. In the first step, an initial software prototype will be developed. A pilot study with two classes will be conducted following the release of this version. This will include qualitative data including observations, questionnaires, class discussion, and interviews with participating teachers and students. The aim of this pilot is to improve the structure and flow of the game. Based upon the feedback and experiences of this initial pilot, a second release of the LifeSim software will be developed. In the second step, Release 2 of the LifeSim software will be formatively evaluated involving 10 classes. A mixture of process and outcome measurements will be used in this step, including focus group discussions, surveys, and interviews.

Pilot, LiFESim Release 1

We will recruit two classes to take part in this step of the evaluation. In this step, and in steps 2 and 3, to have the students play the game in a controlled condition, where we can assure complete exposure to all phases of the game, we will have students play the game as an entire class, with each student having a computer with the game. In all steps of the evaluation, a member of the LifeSim team will be in the classroom to answer questions and to solve any software issues. This person will allow the student to move through the game at their own pace; while also encouraging students to complete all aspects of the game.

In this step of the evaluation, we will collect three sources of data. First, while the classes are playing the game, we will have three observers in the classroom who will take detailed field notes on how the students are interacting with the game, what excites the students, what challenges the students have, conversations between the observers and students and conversations between students. Second, we will have questionnaires for both the students and the teachers with written questions that detail what they like about the game and why; how they seem the game relating to their own food and activity choices, and their ability to navigate our current food and activity environment; and suggestions for how to improve the game. We will also have a full class discussion to have the students share more on the above categories. Finally, we will interview the teachers and a subset of students to get more detailed information on the above topics. The students who will be interviewed will be pre-selected so that the observers can pay careful attention to these students as they play the game to ask detailed questions about the student's experience with the game. We will code these data on emerging themes, develop summary reports on what we have learned, and review these data in order for the game developers and content experts to revise the game based on what was learned in this pilot.

We are currently developing a preliminary release of LifeSim that will be evaluated with a small group of 10 students to inform the pilot study. This preliminary release is being constructed with the Metaplace social gaming platform and will be evaluated August 18 and 19.

Formative Evaluation

This step of the evaluation will include a process instrument that will be completed by all students in 10 classes after they complete the LifeSim game which will ask the students multiple-choice and open-ended questions about what they liked, what they would change, what was easy or challenging, and how they felt the game related to their own food and activity choices. Additionally, the students will complete a pre and post questionnaire that was developed to evaluate the impact of C3 on students. This questionnaire includes questions on conceptual understanding. This instrument collects data in several areas. First, students' conceptual understandings on the scientific evidence for why to make healthful food choices, skills on how to make healthy food choices, and ability to evaluate their diet to determine if it is meeting the C3 behavioral goals. The survey will also measure students' skills on how to plan for and implement small gradual changes that will lead to maintenance of more healthful habits.

The survey will evaluate mediators of behavior change including beliefs, perceived benefits, perceived barriers, self-efficacy, and competence in navigating today's physical activity and food environment. Third, we want to determine if LifeSim has any positive impact on the student's eating behavior–in particular do the students' diets move toward meeting the C3 behavioral goals.

The data from the process and outcome evaluations will be analyzed to enable us to prepare Release 3 of LifeSim for the outcome evaluation and to modify the evaluation instrument for the summative evaluation. We will analyze the process evaluation, summarize the comments from the users and refine the LifeSim game based on the comments. We will also use T-test analysis to compare the pre and post surveys for statistical changes. We will use these data to determine of the game needs to be modified to make is stronger in enabling the students to transfer what they learn in the game to their personal choices. We will also modify the student outcome instrument based on what we learn from the data analysis.

Summative Evaluation

The main goal of this third step of the evaluation is to conduct a summative evaluation to determine how LiFESim impacts students, using a pre-post, intervention-control design. The students in both intervention and control groups will complete the final student outcome instrument developed in step 2 pre and post. The intervention group will receive the LiFESim game. The control group will play another computer game such as a math game or a game on another health topic. After completing the game (7-10 episodes), students will complete the post-survey. We will use ANCOVA to compare the post-test between the intervention and comparison groups with the pre-test scores as co-variants.

The sample size was calculated based upon analysis of the C3 student outcome evaluation data. Because the C3 data suggests that the curriculum was particularly effective in reducing consumption of high fat high sugar foods, we have chosen these behaviors for the basis of our sample size calculations. C3 resulted in a reduction of .9 day per week in sweetened beverage consumption with meals and .4 with snacks, and .6 days per week reduction in consumption of packaged snacks. We estimate that in order to be able to detect a .5 day per week average reduction in these behaviors with the LiFESim game, we need a sample of 407 to achieve 90% power and a sample size of 304 to achieve 80% power. We also estimated, based on our previous work, that we would have 20 students in each class with complete data, or nonparticipation/dropout rate of 25% of each class. Thus, our summative evaluation will have 20 intervention and 20 control classes. We anticipate approximately five classes from each school, and thus about four schools in each condition. We will recruit eight total schools with our Teachers College partners in association with the Middle School Science Coordinator for the New York City Department of Education. We will place the schools into four pairs matched on overall academic performance (standardized literacy and math test scores), percentage of students who qualify for free or reduced lunch, ethnicity, and general school tone. One school from each pair will be randomly assigned to intervention and the other school to control condition. These It anticipated that the students will be about 50% each of boys and girls, and predominantly African-American and Latino.

Statistical design for the analysis is an ANCOVA model. The analysis will be performed as the post-test data with regression adjustment for covariates measured at baseline, thereby including time-related information without modeling time explicitly in the analysis. Covariates are added to the model to reduce confounding, to improve the precision of the estimate of the intervention effect. This analysis will allow us to detect how the LiFESim game impacted students when compared to the control students. The primary hypothesis of the study is that there will be differences between the intervention group and the control group in terms of the targeted behaviors. We also hypothesize that LiFESim will also impact the mediating variables of conceptual understandings—scientific evidence for why to make healthful choices, beliefs, self-efficacy, competence in navigating today's physical activity and food environment and behavioral intention. In addition to statistical testing of hypotheses, we will estimate intervention effect sizes in terms of absolute differences and ratios with appropriate confidence intervals. We will also examine correlation between mediating variables and behaviors.

REFERENCES

Contento IR. Nutrition education: Linking research, theory, and practice. Sudbury, MA: Jones and Bartlett Publishers, Inc.; 2007.

Cullen KW, Watson K, Baranowski T, Baranowski JH, Zakeri I. Squire's Quest: Intervention changes occurred at lunch and snack meals. Appetite. 2005, 45(2): 148-51.

Fu, D, and R Houlette. 2002. Putting AI in Entertainment: An AI Authoring Tool for Simulation and Games. IEEE Intelligent Systems.

Fu, D, and J Ludwig. 2007. *"Game AI Solutions for Serious Games,"* I/ITSEC Conference Tutorial.

Cynthia L. Ogden, PhD; Margaret D. Carroll, MSPH; Katherine M. Flegal, PhD High Body Mass Index for Age Among US Children and Adolescents, 2003-2006 JAMA. 2008;299(20):2401-2405..

Ramachandran, S, E Remolina, and D Fu. 2004. FlexiTrainer: A Visual Authoring Framework for Case-based Intelligent Tutoring Systems. In Proceedings of the Seventh International Conference on Intelligent Tutoring, 848-850.

Thompson, D.J., Baranowski, T., Buday, R., Baranowski, J., Juliano, M., Frazior, M., Wilsdon, J., Jago, R. 2007. In pursuit of change: Youth response to intensive goal setting embedded in a serious video game. Journal of Diabetes Science and Technology. 1(6):907-917.

BIOGRAPHY

CHARLES EARL was born in Atlanta, Georgia and received his doctoral degree in Computer Science from the University of Chicago in 1998. Charles is currently a project manager at Stottler Henke Associates.

REAL-TIME WARFARE SIMULATION GOES WEB 2.0

Gustavo Henrique Soares de Oliveira Lyrio
Roberto de Beauclair Seixas

Institute of Pure and Applied Mathematics – IMPA
Estrada Dona Castorina 110, Rio de Janeiro, RJ, Brasil 22460-320
e-mail: glyrio,rbs@impa.br

Computer Graphics Technology Group – TECGRAF
Pontifcia Universidade Catlica do Rio de Janeiro – PUC-Rio
Rua Marqus de So Vicente 255, Rio de Janeiro, RJ, Brasil 22453-900
e-mail: glyrio,rbs@tecgraf.puc-rio.br

KEYWORDS

War gaming, serious gaming, real time tactics, WEB 2.0

ABSTRACT

This work presents the authors' experience in the process of developing a massive real time multi-player for military tactical online training simulations, and how some web 2.0 applications provide helpful tools to overtake common problems found during this process.

INTRODUCTION

As long as organized conflicts are known in the history of humanity, there are also war gaming. These games were used for training or entertainment.

Although as popular as table-top games, it has taken to tactical war games a little longer to be available for computers. Its large number of units and the advanced set of rules demanded processing power beyond the capabilities of hardware available at the moment. Also, the most established set of rules were made for turn-based war games so, it took a while to translate these rules in real time and also to release the first real-time tactics (RTT) (Adams 2007) computer game.

Real-time tactics is a sub-genre of tactical war games. It is differentiated from real-time strategy (the most popular) due to the lack of resource micro-management and base building, the greater importance of individual units (Adams 2007) and the focus on battlefield tactics.

With the evolution of the hardware resources and the popularization of the Internet, war games became available in online paradigm, where a player create a server game with its own world and other few players connect to this server and play together. It didn't take too much time to the players to demand a unique persistent world, where all the players could play together. The Massive Multi-player Online games (MMO) were born.

About ten years ago, a new way to use the internet was arising. Called "Web 2.0", this second generation of web applications facilitates communication, secure information sharing, interoperability and collaboration on the World Wide Web.

In the last twelve years, we have been working with the Brazilian Marine Corps, in the development of their training system. In the beginning, the goal was to simulate an Amphibian Assault Operation. With the expansion of the system to cover other Brazilian Marines' operations, like riverine operations, amphibian incursions and peace operations (military operations other then war), came the demand for new technologies to provide more details.

Then, it came the idea of joining the concepts and technology used among Web 2.0, MMO and tactical war games to develop a Massive Multi-player Online Real Time Tactics (MMORTT) training tool adding the necessary portion of reality to reach the training goals.

The goal of this paper is to present the adversities found in development of a MMORTT military training system and to show how some Web 2.0 applications could take part in this system providing useful solutions.

COMMON ADVERSITIES FOUND IN DEVELOPMENT PROCESS

During the development process three major adversities where identified: real terrain and 3D units modeling, development of a specific communication system and keep track of units actions.

3D Real world and units modeling

Differently from an entertainment game, the user will be expecting a real world. The training should be done in some place that really exists in a known region of our world. Otherwise the system will lack of reality. That is a huge problem to developers because the system will demand 3D modeling of all the training regions with its characteristics (mountains, rivers, lagoons, swamps and vegetation, for instance).

Real terrain demands real edification, bridges, and roads, which will be another problem. Again the designers will have a lot of work to do. Also you will need units that user can identify as real military units.

- **What we have learned**: Real world means real terrain, real scructures and real units 3D modeling;

Communication system

An additional problem arises from the fact that you are training a group of persons together, not entertaining isolated players. Since the results will be evaluated at a team level those persons, of course, need to interact with it each other. In the real world it would be done by radio communication, with a lot of features like static, interference and electronic warfare, so we need to simulate a radio communication system.

- **What we have learned**: Real world also means real communications with all real features and real problems;

Units line of actions tracking

The last problem, which we have discovered with some field experience, is that after the training is done, a debriefing is made and the users has to discuss about the actions each one took during the game. If it was a long game (more that one day, for instance), the large amount of information and the high level of interaction between users, makes it hard to determine why a specific action has been taken in a specific moment. It would be easier if each users had something like a journal to write down, and explain their line of actions. Such procedure is know on ships as the captain's diary.

- **What we have learned**: Discuss the game result is more important then the game itself;

WEB 2.0 SOLUTIONS

In the previous session we identified three major problems in the process of developing a MMORTT for military training. We next present three solutions provided by Web 2.0 applications that will make the developer's life easy when dealing with these problems.

Terrain and 3D unit 3D modeling with Google Earth API

The google earth API was chosen to handle this issue due to the fact that it alone, handles all the 3D rendering, dismissing all the design work.

In the middle of 2008 the Google company has made available for developers an API for including Google Earth system inside a web page. The benefits from this API are notorious. The developer uses javascript to gain control of the Google Earth plugin and its capability to display a 3D model of the whole world with no cost. Also it is possible to personalize the system with images, 3D models and XML/KML files.

Google has also provided Google SketchUp, a 3D software tool that enables to draw and place models in real world coordinates using Google Earth API. Users are encouraged to make their models available in Google Earth Warehouse, madding huge the probability to someone already has modeled that bridge in your terrain or even that military unit you need.

Below we will show some of the benefits that Google Earth API can bring to the development process of a MMORTT.

Aerial Reconnaissance
Aerial reconnaissance is a system where the user fly through a real terrain and try to detect the positioning of enemy forces. This task showed to be complex due to the necessity of builds a 3D model of every area of training and of each military unit. Terrain modeling can be a real headache when working with huge terrains. It demands a modeling team with designers and developers. Rendering algorithms with culling, LOD and other techniques. Would not be amazing if all that could come ready to use in a single API? Well, it became possible with Google Earth. The Figure 1 shows the reconnaissance system developed using a raster image of a UH helicopter cockpit (with transparent glass) above the Google Earth API. It took about 30 minutes to develop such application.

3D Visualization
The Brazilian Marines instructors complained about the difficulty to understand the line of sight and communication obstructions using a 2D map. They requested a way to really see the game instead of little marks on a map. In addition, a 3D view always makes a game more impressive. Again the same problems discussed above in the Aerial Reconnaissance session came back. Fortunately Google API has resolved it for us providing a tool called Google SketchUp.

Google SketchUp is a tool that allows its users to create

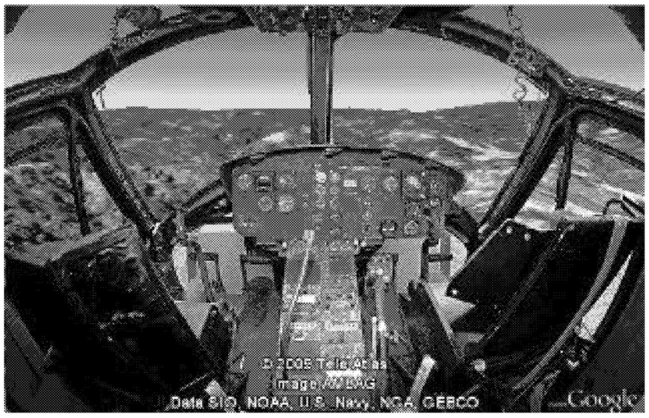


Figure 1: The aerial reconnaissance system using Google Earth API to display an raster image with transparent helicopter cockpit above the Google Earth terrain

3D models and made it available on the Internet. This models are fully supported by the Google Earth API (Figure 2).

Figure 2: 3D visualization showing some Google SketchUp models (downloaded) placed over a real terrain (Grand Canyon). The terrain model is provided by Google Earth API

Communications using Team Speak SDK

Our simulation communications are done with three real radios installed on each simulation room: one for tactical issues, one for logistic requests and one for request fire support. Although the system considers electronic warfare in unit-unit and units-user communications, the communication between two users is done directly by those radios and doesn't pass through the system. The idea is to replace those radios by software using TeamSpeak server and developing an interference module bringing the effects of electronic warfare to the user-user communications also.

TeamSpeak is a software that enables people to speak with each other over the Internet. It consists of both client and server versions. The server acts as a host to multiple client connections, capable of handling literally thousands of simultaneous users. A relevant feature the TeamSpeak present is the ability to create channels. A channel works exactly as a radio channel does. One user creates a channel and may turn it public or protect it with a password. Other users connected on the server may then connect to this channel (providing the password if it has one). TeamSpeak became popular with the Counter-Strike game as a "radio" system used by the players to manage their teams. Another possible solution is to use Skype.

Unit journals through Twitter or Blog

Tracking a combat unit progression over the simulation is important to allow the instructor to make a debriefing and points out to the players which actions went wrong, what should be improved or even to show a great maneuver. However, it has been a hard task. The system needs to be able to store each action taken by each unit. This kind of storage demands a lot of work to detach what was the action line of a particular unit.

Twitter is a free social networking and micro-blogging service that enables its users to send and read other users' text-based posts of up to 140 characters, which makes it fast to read. The posts are known as tweets and are displayed on the user's profile page and delivered to other users who have subscribed to them (known as **followers**). Users can send and receive tweets via the Twitter website, Short Message Service (SMS) or external applications.

A proposal to solve our problem is to generate a twitter account for each unit, where the unit by itself, through the twitter API, informs which actions were taken in a scenario when looking for a goal.

For a debriefing process instructors can subscribe as followers of any unit and follow the unit progression with the 3D visualization tool, and consulting the unit tweets to determine why an unexpected action was taken. This process increases the capability of identifying wrong actions taken by the officers or even registering outstanding unexpected maneuvers that should be shared and discussed with other officers.

Recognized potential in Web 2.0 tools

The United States Army and Air Force saw potential in Twitter and other Web 2.0 tools. Recently, the news has reported that these tools have been used by the military to communicate with people that uses the Internet as alternative source of information to the press media in Afghanistan. One of the goals is to oppose the Taliban advertising.

RESULTS

Using the Google Earth API we were able to develop a 3D visualization and a 3D aerial reconnaissance tool, solving problems that have persisted in the project for many years with no more than 150 lines of codes each. In addition, the visual including 3D models, water movement and sun light effects caused a good impression to the users. The integration with the whole system was also easy, since it was only necessary to call a new function that builds an html page with Google Earth Javascript inside. The whole development process took no more than one hour.

The process of developing radio communication by software using TeamSpeak was slightly more complicated due to the necessity of acquiring data for each type of radio used by the Brazilian Marine Corps and to implement interference and electronic warfare over the TeamSpeak SDK. The idea was also well received by the officers and is currently in a process of evaluation and homologation.

Using Twitter as a track tool for the units was praised by the instructors as an alternative to an email system that has been font of problems and missed units' information for a long time. The tool is also in evaluation. As occurs with the radios, replacing something that has been used by twelve years requires some time to settle down as the new way to do the job.

DISCUSSION

Unfortunately Google Earth API still presents some issues that prevents its use in a training application. The first one is the lack of a Google Earth plugin for Linux. The Brazilian Marine Corps is on a process of free software adoption and would be bad to demand lots of Windows or Mac licenses to support the system. Other issue that demands attention is the time that Google Earth spends to render. In a commercial game it would be not acceptable to wait the rendering process looking the world slowly appear in front of your eyes. Additionally, there are some flickering problems during the render process. An example is when the camera gets a huge height related to the ground.

However, no one can deny that Google Earth API has a promising future. Google Team is working hard to solve these issues and to implement new features to keep the API up to date with the Google Earth software version. The last huge improvement was the ability to go "underwater" available from version 5. This feature inspired us to start working in the 3D representations of submarines and combat divers.

After all, we must consider that Google Earth API is just a one year old API and the contributions that it has made are huge. We believe that in a few years Google Earth API will be "the direction" to follow when doing real terrain rendering.

REFERENCES

Adams D., 2007. *The State of the RTS.*

Graham P., 2006. *Web 2.0.*

Montenegro R.S.R.S.A., 2009. *Objects Visualization in Digital Terrains Using Adaptive View-Dependent Techniques.*

O'Reilly T., 2006. *What is Web 2.0.*

O'Reilly T., 2007. *Web 2.0 Compact Definition: Trying Again.*

Seixas G.L.R., 2004. *Riverine Operations: Modeling and Simulation.*

T. B., 2003. *Strategy Game Programming With Directx 9.0.* Wordware Publishing, Inc.

BIOGRAPHY

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

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

Increasing P2P Gameplay Performance Utilizing I3P

Jeremy Kackley
School of Computing
University of Southern Mississippi
Hattiesburg, MS

Jean Gourd
Dept. of Computer Science
Louisiana Tech University
Ruston, LA

Matthew Gambrell
Programmer
Amaze Entertainment
Austin, TX

{jeremy.kackley, jean.gourd, mgambrell}@gmail.com

Abstract

I3P (Intelligent Peer-to-Peer Protocol) is a peer-to-peer (P2P) protocol developed to assist in the development of low-latency online multiplayer games by providing a minimal but sufficient set of tools appropriate to the task and amenable to study. In this paper, we review the protocol and discuss its utilization in a sample game. We extend our previous work in I3P [16] and provide further analysis, particularly regarding its impact on latency.

1 Introduction

Multiplayer online games (MOGs) are a relatively new but rapidly growing field of entertainment. Peer-to-peer (P2P) networking is also relatively new yet is becoming extremely prevalent. We aim to apply some of the methodologies of P2P networking [18] to the existing infrastructure on which MOGs are built; by this we mean doing such things as pushing tasks currently performed by centralized servers or nodes–which often act as points of failure–in clusters onto the actual players' machines. There are obvious benefits to this such as decreased setup and operational costs. Furthermore, P2P based networks avoid a single point of failure and scale extremely well [20]; such approaches can often result in less server load and lower latencies. Typically, the more peers involved, the more likely it is that a good connection can be made; the less the server is asked to perform, the more users can be supported.

In order to push operations onto peers in a MOG system one has to first partition the player base into subgroups. The reason for this is that every player in the entire game "world" will not be interacting simultaneously; thus, it makes sense to partition them into groups either spatially or associatively. These various partitions can then be handled by subservers, or in our case, by a combination of subservers and peers. This has been covered in other works [3, 4, 9, 15, 17, 21].

Once partitioned, tasks that can be pushed onto clients must be identified. Typical tasks that need to be performed include authentication, communication of perceptive data, validation of received messages, and maintenance and updating of each client's state by the server. These tasks could be subdivided further, but this gives a rough overview of the kinds of operations that continuously occur.

We ultimately decided to push onto the peers the tasks that involve the communication of perceptive data and the validation of received messages. These two steps consist of the clients communicating intent to a server after which the server communicates results to the clients. It seems obvious that the clients could communicate amongst themselves and determine their own state, only occasionally communicating that state back to the server. There is a great deal of detail that one can go into when discussing this concept [11], but it relies on one underlying principle: one needs a very reliable way for peers to communicate with each other. With that in mind, we further realize that the more information that is transmitted, the more likely it is for some of that information to be dropped.

This paper is structured as follows: in section 2, we review I3P. We introduce its integration in a simple MOG in section 3 and further discuss optimization techniques. We discuss performance results in section 4 and conclude, identifying potential future work, in section 5.

2 Intelligent P2P Protocol (I3P)

I3P aims to greedily use all available bandwith to ensure connectivity and reliability, minimizing latency at the cost of bandwidth [16]. It is designed with efficiency in mind, allowing peers to communicate with each other in an economical manner and internally managing non-receipt of transmissions. The core idea of the protocol is that each peer will maintain an *idea* of what it thinks each other peer in its peer group knows about and will automatically forward messages to peers it thinks have fallen behind. It does this by maintaining a matrix of numbers indicating the last message received by each peer from every other peer.

By recruiting each peer to forward messages to all the others, the key notion is for each peer to keep track of which messages all the others have received. This will enable the peer to preemptively update the others with messages it believes them to be missing. Under some circumstances, the need for protocol negotiation will therefore be reduced, thus improving latency. I3P can reliably broadcast a player's intentions to a group of peers with whom s/he is communicating.

Logically, the communicated matrix of numbers corresponds to several *update* messages, which is defined as a message sent by peer A to peer B that informs B that A has received all the messages up to a given number, N, from another given peer, C. In this sense, the update message is sent in a matrix syntactically indicating the latest message peer A has received from each peer in the network.

Suppose there are two signals which peer Z can transmit:

1. Send message M from queue X to peer Y
2. Send "Z has received X up to N" to peer Y

The first signal merely conveys a sequence-numbered user message, which may either have originated on peer Z or on another peer (and is being forwarded by peer Z).

The second signal is a means of indicating which messages peer Z has received. After receiving this signal, peer Y would no longer need to preserve messages N or older from peer X on peer Z's behalf. Once peer Y no longer needs to preserve a message from X for anyone else, it is safe to discard it.

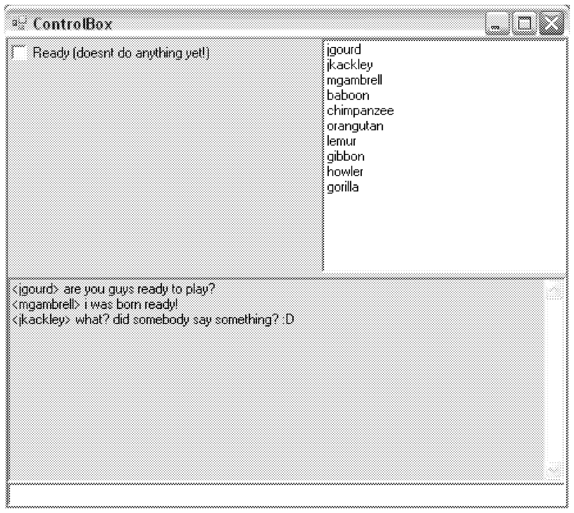
These two signals are all the tools we need for the protocol. Crafting the optimal signal to send at any given point in time remains the challenge. Optimization is also necessary to meet our design goals, as the most unintelligent choices, such as sending the same signal repeatedly, will result in a broken protocol.

The reader can refer to our previous work for a more in depth explanation of this protocol, several illustrative examples, as well as a mathematical analysis of its performance and its use in a simulation [16].

3 I3P in a MOG Setting

In our previous work [16], we simulated the protocol in order to produce initial results and provide a rudimentary performance analysis of I3P. This yielded some interesting data, but we felt we needed a more realistic test. Thus, we extended our previous work by implementing the protocol as part of a multiplayer game built on the XNA Framework [10] in order to measure the performance of the protocol in a real environment. The rest of this paper deals with the details of this implementation and the results of that testing.

The completeness of the I3P implementation was confirmed by its integration into a four-player game–which



jgourd
jkackley
mgambrell
baboon
chimpanzee
orangutan
lemur
gibbon
howler
gorilla

<jgourd> are you guys ready to play?
<mgambrell> i was born ready!
<jkackley> what? did somebody say something? :D

Figure 1: Initial game state illustrating numerous networked players

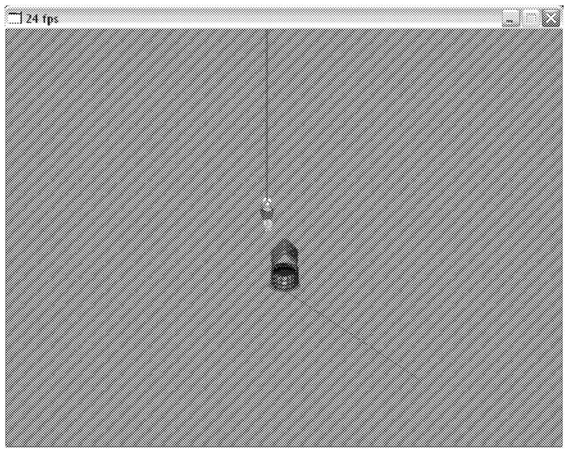


Figure 2: Attempting to cooperatively place the character's feet on top of the well roof

we call *Monster Fishing*–that requires the players to maneuver a cursor onto a target. The simple communication needs of four closely-interacting players resemble the needs of a small group in a MOG setting. Figures 1 through 3 illustrate the game, where the combined efforts of the players are used to guide the character's feet directly on the top of the roof of the well. The cursor is the average of each player's input. Each player's contribution to the motion of the cursor is random and confusing. A player cannot easily play his part without discerning the response of the cursor to his controls; but each player is trying to discern this simultaneously. Success in this game requires a degree of emergent comprehension of the other players' minds–such emergence is impossible if the latencies between players are too great. Therefore, the low latency features of I3P are indeed helpful.

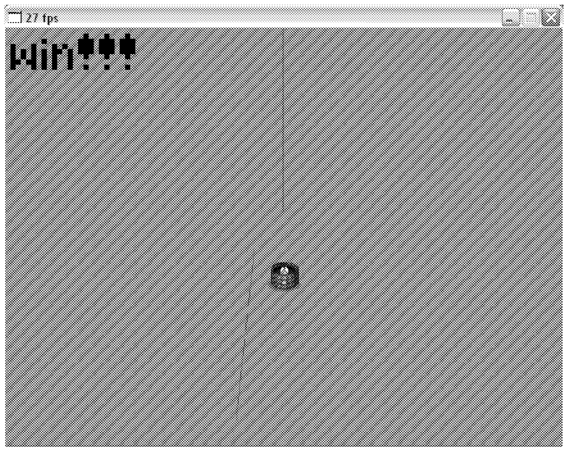


Figure 3: A successful use of combined efforts

In the game, each client uses I3P to broadcast status updates to its peers. The player's individual cursor position is constantly relayed, as well as a flag indicating whether the client thinks the cursor position is in the winning area. When a client believes that the cursor has been in the winning area for eight seconds, it claims that a win has occurred. This message is broadcast via I3P as well; whenever a client discovers that a peer has won, it puts itself into the win state.

Bootstrapping a peer's participation in I3P requires some out-of-band signaling. The new peer needs to discover which sequence number for each peer to advertise its awareness of. In the game, clients broadcast out-of-band to a set list of IP addresses in order to establish their presence in the game. Once this is satisfactorily arranged, the new peer may send update messages to the peer group so that the other peers can begin to integrate the new peer into their broadcasts. I3P subsequently takes care of the rest.

3.1 Network Coding

Network coding (see [13] for a complete review) can be applied to improve efficiency under many conditions. If sources A and B are each trying to send a message to sinks 1 and 2, we may encounter a situation where we can regard peer X in the middle as wanting to behave as a router and finding that it has to decide which sink to relay messages to. In two units of time the peer could send A, B to sink 1; or it could send A, B to sink 2; or it could send A to sinks 1 and 2. In all of these cases, X has been unable to completely relay all the information at its disposal. However, network coding could be employed to send $A + B$ to each sink, also costing two time units.

If each sink had already received messages A and B, then the relayed $A + B$ transmission would be superfluous under any circumstances. If a sink failed to receive either A or B, then the receipt of $A + B$ would permit the inference of whichever message had been lost. This would seem a straightforward improvement. But, if a certain sink had not yet directly received any messages, then its receipt of a relayed $A + B$ will be a disappointment since it represents entirely neither A nor B. Suppose, after receiving $A+B$ at time $t = 0$, that the sink in question is then destined to receive the direct transmissions of A at $t = 1$, followed by B at $t = 2$. The network coding would increase the effective latency of A (which might have been relayed instead of $A + B$) from 0 to 1, while decreasing the effective latency of B from 2 to 1 by making it available before the direct transmission of B is even received. Thus all information is received at $t = 1$, versus $t = 2$ without network coding. If B had been chosen for relay then we would end up with all information received at $t = 1$.

Under these conditions, network coding would improve the total latency while leveling it out a bit, which may or may not be desirable. On the other hand, in a catastrophic situation where, for example, all direct messages are lost, the sink would have been better off had it been relayed something useful. Determining where network coding is advantageous requires detailed analysis of its impact on latency.

4 Results

We ran several instances of the *Monster Fishing* game as earlier described in order to test I3P and have obtained interesting performance results. The first test case–which we simply call game 1–is illustrated in Table 1. Given this simple network of four players, we immediately notice that no packet loss occurred. This makes sense, as we had a rather small test case and all clients possessed high-bandwidth Internet connections. On average, 41.79% of all received packages were relayed by other peers as defined in the I3P protocol. The average latency–defined as the average time difference of the receipt of a sequence and when it was first relayed by another peer–was calculated to be approximately 0.018 seconds.

A second instance of the game–played at a different time when the Internet is typically busy–yielded quite different results as illustrated in Table 2. In this case, client `jkackley` had a particularly horrible satellite connection and consequently only received a small fraction of the total packets. Client `jgourd` experienced packet loss, but the use of I3P suggests that every one of the sequences received via a relay was useful. On average, 21.38% of sequences received were via some relay, and the average latency for all clients was approximately 0.157 seconds.

Table 1: I3P performance results in game 1

User	Packets Relayed/ Packets Received	Average Latency (s)	Packets Lost
baboon	125/312	0.0311342592592593	0
jgourd	150/334	0.0325	0
jkackley	247/416	0.0	0
mgambrell	71/311	0.0101232394366197	0
Averages	41.79%	0.01843937467396975	0

Table 2: I3P performance results in game 2

User	Packets Relayed/ Packets Received	Average Latency (s)	Packets Lost
baboon	5/2839	0.43125	0
jgourd	121/2726	0.0100694444444444	31
jkackley	21/26	0.0	0
mgambrell	4/2624	0.1875	0
Averages	21.38%	0.1572048611111111	7.75

5 Conclusion and Future Directions

The major premise of this protocol relies on an algorithm for signal selection. While we briefly discussed some of the caveats of signal selection in our previous work [16], we have not as yet formulated any substantial algorithms for signal selection and plan to explore this in future work. Our signal selection and concatenation algorithms are primitive in their current forms. An obvious next step is to develop solid, parameterizable algorithms and derive optimal values. The simplicity of the core signals will aid in modeling and simulation.

A parallel effort in developing a more intricate state definition, with more record-keeping and performance variables, will provide a toolbox upon which the algorithms can be built. This could even facilitate the real-time selection of entirely different, simpler algorithms, each optimized for different situations, which would be activated as the situation demanded.

Although we have provided a start with the integration of I3P into a rather simple MOG, we must continue to develop the setup and maintenance procedures necessary for making I3P more useful in games. Particularly, newly-entering peers will confuse everyone else by their sudden appearance unless an additional mechanism is added to prepare existing peers for the discontinuity. We are investigating several techniques as introduced in [1, 2, 6, 19, 23, 24, 25]. Furthermore, we plan to implement I3P as the protocol in other MOGs to produce comparative results in addition to more exhaustively comparing I3P to existing P2P protocols.

Implementing network coding, as discussed in section 3.1, as part of I3P may yield interesting results, particularly with respect to optimistic performance and increased protocol efficiency. Investigating potential benefits of such techniques and integrating them is left for future work.

Lastly, I3P was originally designed as a result of the exploration of ways in which groups of peers could continuously communicate intention and maintain state among themselves; we found the existing protocols to be lacking in this respect. Now that we have developed the protocol into a functional state, it would benefit us to revisit the original problem: how do we take the tasks performed by subservers in a MOG setting and allow peers to perform these tasks? This would yield insight into the true effectiveness of this protocol and might even lead to further refinements of it, as well as potentially leading to a framework for moving these tasks onto sets of peers.

References

[1] Ailixier Aikebaier, Naohiro Hayashibara, Tomoya Enokido, and Makoto Takizawa. A distributed co-

ordination protocol for a heterogeneous group of peer processes. *aina*, 0:565–572, 2007.

[2] S. Alda. Component-based self-adaptability in peer-to-peer architectures. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 33–35, Washington, DC, USA, 2004. IEEE Computer Society.

[3] Michele Amoretti, Francesco Zanichelli, and Gianni Conte. Performance evaluation of advanced routing algorithms for unstructured peer-to-peer networks. In *valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 50, New York, NY, USA, 2006. ACM Press.

[4] S. Caltagirone, M. Keys, B. Schlief, and M. J. Willshire. Architecture for a massively multiplayer online role playing game engine. *Journal of Computing Sciences in Colleges*, 18(2):105–116, 2002.

[5] Chris Chambers, Wu chang Feng, and Wu chi Feng. Towards public server mmos. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 3, New York, NY, USA, 2006. ACM Press.

[6] Alvin Chen and Richard R. Muntz. Peer clustering: a hybrid approach to distributed virtual environments. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 11, New York, NY, USA, 2006. ACM Press.

[7] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza. Locality aware dynamic load management for massively multiplayer games. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 289–300, New York, NY, USA, 2005. ACM Press.

[8] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. An empirical evaluation of tcp performance in online games. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 5, New York, NY, USA, 2006. ACM Press.

[9] Kuan-Ta Chen and Chin-Laung Lei. Network game design: hints and implications of player interaction. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 17, New York, NY, USA, 2006. ACM Press.

[10] Microsoft Corporation. Xna framework, 2007. http://msdn2.microsoft.com.

[11] Keiichi Endo, Minoru Kawahara, and Yutaka Takahashi. A proposal of encoded computations for distributed massively multiplayer online services. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 72, New York, NY, USA, 2006. ACM Press.

[12] W-C Feng, F. Chang, W. Feng, and J. Walpole. Provisioning on-line games: a traffic analysis of a busy counter-strike server. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 151–156, New York, NY, USA, 2002. ACM Press.

[13] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. Network coding: an instant primer. *SIGCOMM Comput. Commun. Rev.*, 36(1):63–68, 2006.

[14] Tobias Fritsch, Benjamin Voigt, and Jochen Schiller. Distribution of online hardcore player behavior: (how hardcore are you?). In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 16, New York, NY, USA, 2006. ACM Press.

[15] Thorsten Hampel, Thomas Bopp, and Robert Hinn. A peer-to-peer architecture for massive multiplayer online games. In *NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, page 48, New York, NY, USA, 2006. ACM Press.

[16] J. Kackley, M. Gambrell, and J. Gourd. I3p: A protocol for increasing reliability and responsiveness in massively multiplayer games. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 12(2):142–149, 2008.

[17] J. Kim, J. Choi, D. Chang, T. Kwon, Y. Choi, and E. Yuk. Traffic characteristics of a massively multiplayer online role playing game. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8, New York, NY, USA, 2005. ACM Press.

[18] Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. On the topologies formed by selfish peers. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 133–142, New York, NY, USA, 2006. ACM Press.

[19] Yoshio Nakajima, Kenichi Watanabe, Naohiro Hayashibara, Makoto Takizawa, Tomoya Enokido, and S. Misbah Deen. Satisfiability and trustworthiness of peers in peer-to-peer overlay networks. *ares*, 0:42–49, 2006.

[20] X. Ren, K. Li, R. Li, and L. Yang. An improved trust model in p2p. In *2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*, pages 76–81, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[21] Abdennour El Rhalibi and Madjid Merabti. Agents-based modeling for a peer-to-peer mmog architecture. *Comput. Entertain.*, 3(2):3–3, 2005.

[22] Abdennour El Rhalibi, Madjid Merabti, and Yuanyuan Shen. Aoim in peer-to-peer multiplayer online games. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, page 71, New York, NY, USA, 2006. ACM Press.

[23] Jiuyang Tang, Weiming Zhang, Weidong Xiao, Daquan Tang, and Junfeng Song. Self-organizing service-oriented peer communities. *aict-iciw*, 0:99, 2006.

[24] Kenichi Watanabe, Yoshio Nakajima, Naohiro Hayashibara, Tomoya Enokido, and Makoto Takizawa. Confidence-based trustworthiness of acquaintance peers in peer-to-peer overlay networks. *icdcsw*, 0:30, 2007.

[25] Kenichi Watanabe, Yoshio Nakajima, Naohiro Hayashibara, Tomoya Enokido, Makoto Takizawa, and S. Misbah Deen. Trustworthiness of peers based on access control in peer-to-peer overlay networks. *icdcsw*, 0:74, 2006.

JEREMY KACKLEY is a doctoral candidate at the University of Southern Mississippi in Hattiesburg, MS. His primary areas of research include intelligent agents, distributed computing, and software engineering.

AUTHOR LISTING

AUTHOR LISTING