16TH INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2015

EDITED BY

Sander Bakkes

and

Frank Nack

December 2-4, 2015

University of Amsterdam
Amsterdam
THE NETHERLANDS

A Publication of EUROSIS-ETI

16TH International Conference

on

Intelligent Games and Simulation

AMSTERDAM, THE NETHERLANDS

DECEMBER 2 - 4, 2015

Organised by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Binary Illusions

Ghent University

Larian Studios

University of Skövde

University of Amsterdam

Amsterdam, The Netherlands

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITOR

General Conference Chair

Sander Bakkes
Universiteit van Amsterdam, The Netherlands

Local Programme Committee

Frank Nack, Universiteit van Amsterdam, The Netherlands
Anja van der Hulst, TNO, The Netherlands,
Anton Eliëns, Vrije Universiteit Amsterdam, The Netherlands
Pieter Spronck, Tilburg University, Tilburg, The Netherlands

INTERNATIONAL PROGRAMME COMMITTEE

Game Development Methodology

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

Physics and Simulation

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

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

Rendering Techniques
Joern Loviscach, Hochschule Bremen, Bremen, Germany

Artificial Intelligence

Artificial Intelligence and Simulation Tools for Game Design
Stephane Assadourian, UBISOFT, Montreal, Canada
Flavio Soares Correa da Silva, USP, Sao Paulo, Brazil
Antonio J. Fernandez, Universidad de Malaga, Malaga, Spain
Marc-Philippe Huget, University of Savoie, Le-Bourget-du-Lac, France
Joseph Kehoe, Institute of Technology Carlow, Carlow, Ireland
David Moffat, Glasgow Caledonian University, Glasgow, United Kingdom
Sam Redfern, National University of Ireland, Galway, Ireland
Miguel Tsai, Ling Tung University, Taichung, Taiwan

Learning & Adaptation
Christian Bauckage, University of Bonn, Sankt Augustin, Germany
Christos Bouras, University of Patras, Patras, Greece
Adriano Joaquim de Oliveira Cruz, Univ. Federal de Rio de Janeiro, Rio de Janeiro, Brazil
Andrzej Dzielinski, Warsaw University of Technology, Warsaw, Poland
Maja Pivec, FH JOANNEUM, University of Applied Sciences, Graz, Austria

INTERNATIONAL PROGRAMME COMMITTEE

Intelligent/Knowledgeable Agents
Nick Hawes, University of Birmingham, United Kingdom
Wenji Mao, Chinese Academy of Sciences, Beijing, China P.R.

Collaboration & Multi-agent Systems
Sophie Chabridon, Groupe des Ecoles de Telecommunications, Paris, France

Opponent Modelling
Pieter Spronck, University of Maastricht, Maastricht, The Netherlands
Ingo Steinhauser, Binary Illusions, Braunschweig, Germany

Peripheral

Psychology, Affective Computing and Emotional Gaming

Myriam Abramson, US Naval Research Laboratory, USA
Eva Hudlicka, Psychometrix Associates, Blacksburg, 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

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

Online Gaming and Security Issues in Online Gaming

Marco Furini, University of Modena and Reggio Emiliano, Modena, Italy
Pal Halvorsen, University of Oslo, Oslo, Norway
Andreas Petlund, University of Oslo, Oslo, Norway
Jouni Smed, University of Turku, Turku, Finland
Knut-Helge Vik, University of Oslo, Oslo, Norway

MMOG's

Michael J. Katchabaw, The University of Western Ontario, London, Canada
Jens Mueller-Iden, University of Munster, Munster, Germany

Serious Gaming

Wargaming Aerospace Simulations, Board Games etc....
Roberto Beauclair, Institute for Pure and Applied Maths., Rio de Janeiro, Brazil
Joseph M. Saur, Georgia Tech Research Institute, Atlanta, USA
Jaap van den Herik, Tilburg University, Tilburg, The Netherlands

Games for training
Michael J. Katchabaw, The University of Western Ontario, London, Canada
Jens Mueller-Iden, Universitaet Muenster, Muenster, Germany

INTERNATIONAL PROGRAMME COMMITTEE

Games Applications in Education, Government, Health, Corporate, First Responders and Science
Laura Lima, Universidade Federal de Juiz de Fora, Minas Gerais, Brazil
Russell Shilling, Office of Naval Research, Arlington VA, USA

Games Interfaces - Playing outside the Box

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

GAME ON®
2015

All articles published in these Proceedings have been peer reviewed

EUROSIS-ETI Publications are ISI-Thomson and IET referenced

GAMEON proceedings are indexed on SCOPUS and Elsevier Engineering Village

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

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

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

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

EUROSIS-ETI Publication

ISBN: 978-9077381-91-5
EAN: 978-9077381-91-5

Preface

Dear Participants,

It gives me the greatest pleasure to welcome you to the interdisciplinary Game Studies programme at the University of Amsterdam, and to the 16th edition of GAMEON, which we are hosting between 2nd and 4th of December 2015.

We are most pleased to host you, dear conference participants, and learning from your specific approach to game research, were it focused on game design, education, training and simulation, artificial intelligence, or procedural and online gaming.

Moreover, we are exited to host two excellent keynote speakers, namely Prof. Georgios Yannakis – who will talk on affective computing and player experience modelling in relation to (automated) game design, and Dr. Anja van der Hulst – who will talk on serious gaming for tactical and strategic decision making. Also, we are happy that the programme is supplemented with workshops that will be provided by renowned game scholars Prof. dr. Anton Eliëns and Dr. Pieter Spronck.

Finally, we are exited to host the co-alligned *Game Studies Symposium* on Friday afternoon, which will host keynote speaker Prof. Georgios Yannakis, and which will further explore the topic of affective game experiences.

I would like to express my gratitude to all those who have contributed to this event: firstly to those who have submitted papers, and will present them over the next couple of days and; of course, to Philippe Geril who is the driving force behind GAMEON; and to the programme committe who have reviewed papers, and contributed to organising this event.

I hope that you will have a great time in the city of Amsterdam, and will find the conference interesting and inspiring.

Amsterdam, December 2015

Sander Bakkes
GAMEON'2015 General Conference Chair

CONTENTS

CONTENTS

GAME AI

INTELLIGENT AGENTS

PROCEDURAL PROGRAMMING

ONLINE GAMING

SCIENTIFIC PROGRAMME

GAME DESIGN

INTERACTIVITY IN COMPUTER GAMES

Barbaros Bostan
Department of Information Systems
Yeditepe University
Inonu Mah. Kayisdagi Cad
34755 Atasehir/Istanbul
Turkey
E-mail: bbostan@yeditepe.edu.tr

Gokhan Sahin
Department of Information Systems
Yeditepe University
Inonu Mah. Kayisdagi Cad
34755 Atasehir/Istanbul
Turkey
E-mail: sahin@yeditepe.edu.tr

Mehmet Can Uney
Department of Game Design
Bahcesehir University
Kemeralti Caddesi
Karaoglan Sokagi No: 24/a
Turkey
E-mail: mehmetcan@gmail.com

KEYWORDS
Interactivity, interaction, computer games

ABSTRACT

In this paper we attempt to clarify the popular term 'interactivity' within the context of computer games by classifying it into three categories: personal interactivity, social interactivity and environmental interactivity. Each category of interactivity contains five subcategories and success stories from popular computer games are given for each section to highlight the depth and quality of interaction the selected games provide.

INTRODUCTION

Interactivity is a popular catchword and selling strategy for new media but everyone has their own idea about what interactivity is. Is it simply a computer system that responds to user's actions? Is it the ability of the user to control and modify messages? And what is so interactive about computer games? According to Lievrouw and Livingstone (2002) interactivity is not unique to new media but is generally considered to be a central characteristic of it. Researchers often focus on the ambiguity of the concept, conceptualization difficulties, and overuse of the term, because various fields defined interactivity from different perspectives and associated it with different terms: synchronicity, control, rapidity and speed, participation, choice variety, directionality, hypertextuality, connectedness, experience and responsiveness (Rafaeli and Ariel, 2007).

Lievrouw & Livingstone (2002) defined three forms of interactions for emerging communication systems: user-to-user interactivity, user-to-documents interactivity and user-to-system interactivity. According to the level of receiver control on the messages (low and high) and the direction of communication (one-way and two-way), user-to-user interactions are classified into four groups: monologue, feedback, responsive dialogue and mutual discourse. Based upon the level of receiver control on the messages (low and high) and the nature of the audience (active and passive), user-to-documents interactions are classified into four groups: packaged content, content-on-demand, content exchange and co-created content. According to the center of control (human and computer) and the nature of the interface (apparent and transparent), user-to-system interactions are classified into four groups: computer-based interaction, human-based interaction, adaptive interaction and flow. An ideal interactive system, according to these models, takes the form of mutual discourse, co-created content and flow.

User-to-user interaction, which is mediated through one or more of the five senses, primarily composes of verbal and non-verbal communication forms such as gestures, poses, facial expressions, etc. From a computer gaming perspective, it takes two different forms: player-to-player interactions experienced in multi-player environments and the information exchange between the player and a synthetic agent or a non-player character (NPC). Player-to-player interactions, which enable communicators to have more control over their experience beyond the constraints of time and geography, can be classified as mutual discourse where the sender and receiver roles become nearly indistinguishable. On the other hand, player-to-NPC interactions in computer games can take different forms such as monologue, feedback or responsive dialogue. The interaction level depends on the believability and the AI capabilities of the agent. Although it may seem strange at first, players may attribute human characteristics to virtual agents. The more believable and realistic these characters and their behaviors are, the more human-like the communication process becomes.

In terms of user-to-content interactivity, traditional computer games are usually shipped to online or brick and mortar stores in the form of packaged content. Online gaming platforms like Games for Windows or Xbox Live provide content-on-demand, allowing users to purchase games or game add-ons, to gain and keep track of their achievements (gamerscore) in order to display their progress and prowess to the community. Content exchange can be experienced in computer games that provide special tools or scripting languages to their players. Players can use these tools or scripting languages to modify the existing game or to create new chunks of content, both of which are usually shared on community websites. Modifications change the gameplay process by altering the game mechanics, the virtual environment, the appearance and behavior of 3D

objects and virtual characters. And finally, interactive storytelling refers to gaming environments where real-time feedback collected from the players is used by the game engine to continuously modify the content as it is being delivered. Although the player is passively providing a feedback of his/her preferences and play style while actively playing the game, these systems can be classified as being closest to co-created content.

The study of user-to-system interactivity or Human-Computer Interaction (HCI), which aims to improve the interactions between users and computers, is not only concerned with hardware and software but is also the intersection of several fields of study such as psychology, sociology, cognitive science, human factors, interface design, etc. Today's computer games can be classified as computer-based interaction where the player makes his/her selections from the presented information. The interfaces used to interact with a computer game (personal computers, gaming consoles, gamepads, etc.) are usually not transparent but apparent. Interface transparency can be experienced in immersive 3D environments where head-mounted displays, sensing gloves or other specialized equipment are used to block the sensations from the real world and to help the user focus on the sensations of the virtual. But it can also argued that, regardless of the transparency of the interface, our minds won't really want to do the work of separating media from reality if the media image is pleasant or motivating at a deep psychological level (Castranova, 2007). This is a reasonable explanation since some players loose themselves in the gaming environment and experience the flow state described as activities that provide a sense of discovery, a creative feeling of transporting the person into a new reality or previously undreamed-of states of consciousness (Csikszentmihalyi, 1990).

Sellers (2006) brought a brand new perspective to our notion of interactivity by defining four different levels of it that lead from perceptual, cognitive and psychosocial processes. Perceptual and physical interactivity comprises of the reactions humans give to physical stimuli, especially to bright colors, flashes, moving images, rhythmic or explosive sounds, and to specific proportions in form and color. Game players are also attracted by these pleasurable stimuli and they seek variety in these sensuous impressions. Short-term cognitive interactivity incorporates tasks that involve short-term memory and emotional focus. They combine together to form longer-term goals where planning and strategy come into play to define long-term cognitive interactivity. In this regard, computer gaming can be defined as a series of short-term cognitive interactions to reach a longer-term cognitive goal: the completion of the game or the mission of the protagonist. In a computer game, micro choices are moment-to-moment choices of a player; the way these micro-choices fuse as a long-term strategy defines the macro level of a choice (Salen and Zimmerman, 2003). And social interactivity is both an internal and an interpersonal process that becomes more important if players have persistent identities and are able to affect the game state together.

INTERACTIVITY IN COMPUTER GAMES

In a gaming environment, although it is being overshadowed by the attributes of the technology or characteristics of the medium, interactivity is a product of the computer mediated communication process and an outcome of player actions. According to Rafaeli (1988), interactivity is an under-defined concept that has little consensus on its meaning but it is not located in the features of the medium or user perceptions but in the relatedness of transmitted messages with previous exchanges of information where sender and receiver roles become interchangeable. In this sense, messages transmitted by a computer game are related with the previous exchanges of information or the former actions of the player. The complexity of this relatedness varies from simply keeping track of game scores to interactive storytelling systems that analyze player actions throughout the game for a customized experience. Our first definition based on Rafaeli (1988) is:

Game playing is a retrospective experience where the player's previous actions or choices have an impact on his/her future.

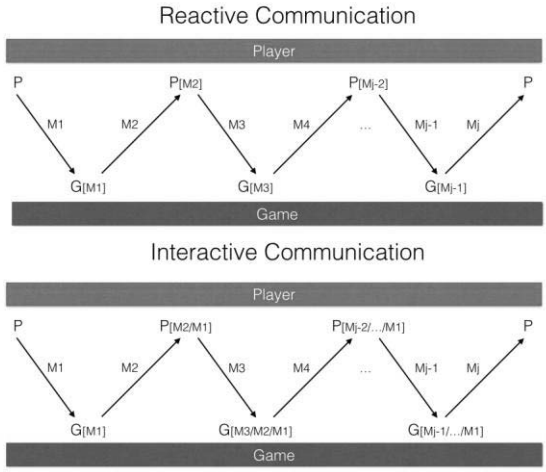


Figure 1: Interactivity model adapted from Rafaeli (1988)

Reactive communication occurs when the game is responding to every message or input send by the player but the system is not keeping track of the history of messages and simply reacting to the last message or input provided by the player. Interactive communication occurs when the game records the history of player actions or choices and responds to the player based on this knowledge.

According to Steuer (1992), interactivity in virtual environments is composed of three elements: speed, mapping and range. Speed is the response time of the virtual world; range represents the number of attributes that can be manipulated by the user; and mapping is a function of the types of controllers used to interact with the mediated environment. Computer games aim to provide instant feedback to player actions but the response time usually depends on the configuration of the personal computer if it is a PC game and on the bandwidth of player's Internet

connection if it is an online game. The mapping or the controllers used to interact with the game ranges from standard keyboard or mouse for PC games and gamepads for consoles to specialized input devices such as joysticks, racing wheels or wireless remote pistols. It should be noted that speed and mapping are constant variables for console players. The range of attributes that can be manipulated by the player greatly varies, from the movement/rotation of simple objects in a Tetris game to massive multi-player gaming environments with thousands of objects and players to interact with. Speed and mapping are the technological variables of interactivity here but range is the key towards a meaningful definition of interaction in computer games. Our second assumption based on Steuer (1992) is:

The range of variables that can be manipulated and customized by the user determine the depth of interaction.

Every player interacts with something in a computer game but what is the range of variables that can be manipulated by the player? How do we classify these interactions?

Personal Interactivity

Computer games aim of creating artificial sensory information similar to the stimuli human senses detect and interpret in the real world so that the player will have a sense of being physically in a virtual world. This is personal or physical interaction, or the interaction with the second self. The player needs an avatar or a physical representation in the virtual world which is usually the protagonist of a story. This character or entity may have gestures, facial expressions and movements which can be classified as kinesics, which is body motion communication. He/she will have dialogue options and/or voice as a means of verbal communication. He/she will also have various equipment (clothing, armor, weapons, etc.) and abilities (skills, spells, feats, etc.) to play the game. Therefore, physical interaction can be analyzed in five sections: avatar, kinesics, verbal communication, equipment and abilities. Schultze and Leahy (2009) also defined the technological features of the second self as a body, possessions, animations, a profile, a camera and modes of communication. Given below are examples from popular computer games that offer rich interaction opportunities for personal interactivity.

Avatar: The *Elder Scrolls* series offered extensive customization options for the representation of your character. Besides common changes such as skin tone, hair color, hair model, eye color, etc. you can also adjust your character's brow, chin, cheekbone, nose, jaw and mouth with various sliders. There are hundreds of forum posts and YouTube videos on how to make your character look like celebrities. The *Dragon Age* series also provide similar customization options when you create your character.

Kinesics: The *Witcher Wild Hunt* used a motion capture studio, with the help of master swordsman Maciej Kwiatkowski, to practice and capture the protagonist's movements and all of the other combat animations for the game. Even the sex scenes in the game are constructed from over 16 hours of motion capture data. Realistic kinesics make the protagonist of the game more believable for the player.

Verbal Communication: The *Baldur's Gate* series offered well written dialogue options for the characters that captivate the player. The player can also select his/her character's voice from a preset of voices. Voice acting for the protagonist of the game is a major issue in game design because he/she will be the voice of the player in the virtual world.

Equipment: The *Diablo* series offered hundreds of different equipment throughout each game but with the material salvaging and item crafting options there are virtually an infinite number of items that the player can use. The legendary loot is the most captivating feature of the game. Similarly, the *Kingdoms of Amalur: The Reckoning* offer blacksmithing and sagecrafting skills for item customization. Sagecrafting skill allows the player to craft gems from shards found throughout Amalur, socket them into equipment, and combine lower quality shards into higher quality shards.

Abilities: Games based on the famous Advanced Dungeons & Dragons tabletop role-playing game (RPG) offer a complex and rules heavy system with different classes that come with specific abilities, skills, feats and spells but the player can also customize his/her own class. *Baldur's Gate* and *Icewind Dale* series are the most prominent examples.

Social Interactivity

The player is rarely alone in the virtual world and he/she constantly interacts with other characters whether they are artificial or real. Some of the artificial characters simply populate the environment but others are merchants or key characters of the story. There are also support characters called henchmen that travel with the player and help him/her. Characters, artificial or real, also build factions or guilds in the game where they gather, share information and resources, help eachother. And of course there are enemies scattered across the virtual world which oppose the player. Bartle (2004) also classified the inhabitants of the virtual world as characters, non-player characters and monsters. He also identified the roles of non-player characters as: to buy, sell, and make stuff; to provide services; to guard places; to get killed for loot; to dispense quests, to supply background information, and to do stuff for players. Given below are examples from popular computer games that offer rich interaction opportunities for social interactivity.

Non-Player Characters (NPCs): *The Mass Effect* series have various NPCs, each with a personality of his/her own. Players are inclined to form an emotional bond with these characters who fight side by side with him/her. This emotional bond, or the empathy between players and characters, creates a more powerful narrative effect

(Freeman, 2004). Online tribute videos dedicated to various NPCs are indicators of this emotional bond.

Guilds/Factions: Guilds in the *World of Warcraft* offer many benefits including free items, opportunities for groups, access to trade skill masters, quest items, and greatly enhances the gameplay experience. Players can meet friends, share adventures, and find people to protect themselves. There are also guilds in single player games. For example, *The Elder Scrolls IV: Oblivion* was the Elder Scrolls title to employ the conventional four factions, which were called "Guilds". These factions included the Thieves Guild, the Mages Guild, the Fighters Guild, and the Dark Brotherhood.

Henchmen: Henchmen of the *Guild Wars* are computer-controlled adventurers who are always ready to travel with the player. Their skills, attributes, and levels will change as the player progresses. They count as party members and thus, claim their share of the party's loot and experience.

Other Players: The *MapleStory* is a free-to-play massively multiplayer online role-playing game where players can interact with others in many ways, such as through chatting, trading and playing minigames. Throughout the first six years eight million accounts have been registered and more than twenty million characters were created.

Enemies: The Nemesis system of the *Shadows of Mordor* provides randomly named enemies in Sauron's Army that are generated uniquely with each play of the game. Each enemy has their own personality and will rise or fall within their social structure as the game progresses. They are affected by the player's actions and they remember the player if they have encountered him/her before.

Environmental Interactivity

The player does not only interacts with virtual or real characters but also effects his/her environment. The virtual world consists of objects ranging from a spoon to a skyscraper that can be manipulated or effected by the user. There are movable objects, destructible objects, constructive objects and upgradable objects in a virtual environment. The interactions with these objects conform to the laws of physics defined by the designers. The environment may also interact with itself regardless of player actions but may affect the player's gameplaying experience. Given below are examples from popular computer games that offer rich interaction opportunities for these categories.

World Physics: The *Star Wars: The Force Unleashed* series used the Digital Molecular Matter engine for realistic object and environment behavior, Euphoria Engine for intelligent characters responses to physics, and Havok engine for rag-doll physics and collision detection. The result is a delicate balance between realistic and entertaining physics. The physics engine also affects the manipulation of virtual objects, such as moving or destroying them.

Movable Objects: The Wizard character of the action adventure game *Trine 2* can magically move many things with his Levitation ability and conjure a plank or a floating platform into the world. Conjured boxes can be used for weight or for jumping on. By placing a plank on top of a box, one can levitate the box when standing on top to reach places one could not originally. Moving objects can be used for solving puzzles, protecting characters and/or defeating enemies.

Destructible Objects*: Red Faction* series are first-person shooters where all environments are fully destructible which means that every single building or structure in the game can be destroyed. Destroying objects is not only a visual effect but an integral part of the gameplay, meaning that the player could use explosives to dig holes through cave walls or blow holes in walls.

Constructive Objects: Gameplay of *Minecraft* involves players interacting with the game world by placing and breaking various types of blocks in a 3D environment. Other activities in the game include exploration, resource gathering, crafting, and combat.

Upgradable Objects: Sid Meier's Colonization is a a turn-based strategy game themed on the early European colonization of the New World. When the player founds a colony it already has the most basic structures, but improving them and constructing new ones can greatly further the player's cause in the New World. For example, a fort is a substantial improvement over the stockade and a fortress is an upgraded fort or an armory can be upgraded into a magazine and a maganize can then be upgraded into an arsenal.

CONCLUSION

Interactivity in computer games is defined by the number of possible actions at a given time and the number of variables that can be manipulated by the user but these actions or manipulations are related with eachother and earlier player choices. The realism, playability, and believability of these player actions/manipulations determine the depth of interaction. It should be noted that the player should also feel himself/herself in the world mediated by the computer and should believe that he/she exists there (presence) and the player should also feel a sense of control over the events thinking that he/she can affect the virtual world. So, our definition of interaction in computer games is:

Interactivity is the extent to which a player can modify his/her second self, the characters and the objects that constitute the virtual world in a retrospective way so that the past interactions/choices will affect his/her future.

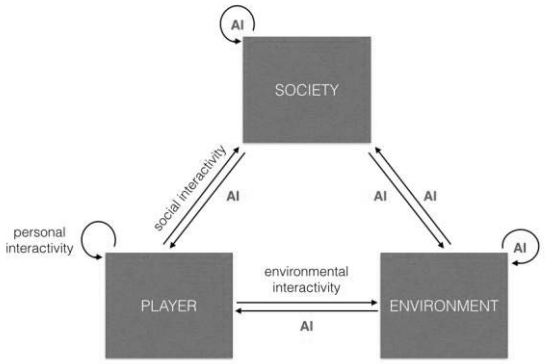


Figure 2: Interactivity in (single-player) computer games

It is important to note that the three interaction dimensions mentioned in this study are not necessarily correlated in the characterization of interactivity. They can be considered as independent dimensions/vectors that may or may not be applied to a computer game. The popular mobile game *Candy Crush* does not provide any form of personal or social interactivity but focuses on environmental interactivity. In this regard, role playing games require special attention because all three forms of interactivity are provided with correlations between them. Your actions/choices as the protagonist of the game affect the society, the environment and your second self.

The interactive nature of computer games has been a cliché among researchers but there is a need for classification of player interactions in computer games. What do the players interact with? How do they interact? What is the range of variables that can be manipulated by the player? In this paper, we attempt to answer these questions by defining three categories of player interactions: personal interaction, social interaction and environmental interaction. Five subcategories for each further explained the nature of 'interactivity' with examples from popular computer games that offer rich interaction opportunities.

REFERENCES

Bartle, R. A. (2004). Designing Virtual Worlds. Berkeley, CA: New Riders.

Castranova, E. (2007). *Exodus to the Virtual World: How online fun is changing reality.* New York: Palgrave Macmillan.

Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience.* Harper Perennial.

Freeman, D. (2004) Creating Emotion in Games: the Craft and Art of Emotioneering. Berkeley, CA: New Riders.

Lievrouw, L. and Livingstone, S. (Eds.). (2002). *Handbook of New Media: Social Shaping and Social Consequences.* London: Sage.

Rafaeli, S. (1988). Interactivity: From New Media to Communication, in: Hawkins, R.P., Wiemann, J.M., and Pingree, S. (Eds.) *Advancing Communication Science: Merging Mass and Interpersonal Processes.* Newbury Park, CA: Sage, pp.110-134.

Rafaeli, S. and Ariel, Y. (2007). Assessing Interactivity in Computer-Mediated Research, in: Joinson, A.N., McKenna, K.Y.A., Postmes, T., and Rieps U.D. (Eds.). *The Oxford Handbook of Internet Psychology*. Oxford University Press, pp. 71-88.

Salen, K. and Zimmerman, E. (2003). *Rules of Play: Game Design Fundamentals*, Cambridge, MA: The MIT Press.

Schultze, U. and Leahy, M.M. (2009). The Avatar-Self Relationship: Enacting Presence in Second Life. International Conference on Information Systems Phoenix, AZ: Association of Information Systems

Sellers, M. (2006). Designing the experience of interactive play, in: Vorderer, P., Bryant, J. (Eds.). *Playing video games: Motives, responses, and consequences.* Mahwah, NJ: Lawrence Erlbaum, pp. 9-22.

Steuer, J. (1992). Defining virtual reality: Dimensions determining telepresence. *Journal of Communication 42* (4), 73-93.

Comparative Analysis on Game Design Pattern Collections

Tapani N. Liukkonen, Olli I. Heimo and Tuomas Mäkilä
Technology Research Center

University of Turku
20014 Turun yliopisto
Finland
Email: {taneli, olli.heimo, tusuma}@utu.fi

Jouni Smed

Department of Information Technology
University of Turku
20014 Turun yliopisto
Finland
Email: jouni.smed@utu.fi

KEYWORDS

Game design patterns.

ABSTRACT

Varying methods to support the game design process exist. Researchers of game design have worked on formalizing the experience-based craft by creating game design patterns. However, these patterns are incompatible in their creation, presentation, and usage. They also have strayed away from their original roots. In this paper, we suggest a taxonomy of game design patterns based on their usage during the game design and implementation. Our proposal is based on the literature research on the history of the game design patterns, and analysis of the existing game design patterns and collections of them.

INTRODUCTION

Gaming industry has become one of the largest branches of the entertainment industry, sales worth being over 15 billion dollars in the USA alone during year 2014 (ESA, 2015). Budgets for the large game productions regularly go over 100 million dollars, which makes their production a high-risk endeavor. In the market, the shelf time of a game is short and competition for sales is fierce. During this time, the game competes with experiences aimed to generate gratification in players. These experiences are created by the video game designer.

The art of video game design is a young discipline which has drawn many of its tools from the related fields that are also partly participating in the design and implementation process of the game artifact, e.g. graphical artists with their tools and background knowledge of their craft. Many of the other disciplines participating in this creative process have more formal and mature means of relaying their ideas from person and group to another, e.g. programmers have derived their design tools and methods from the applicable domains of software engineering.

The game designer is responsible for creating the overall structure of the game, planning the player's progression through the game, activities available during the game, and the experience the player has during the gameplay. Designers' means to achieve these goals are the rules, mechanics, and story. (Smith, 2012) For game designers, movie industry and other entertainment branches have been the source of common tools such as storyboards and scripts, but the tools used in these other branches have been developed for creating passive forms of media. Games are inherently interactive, and thus require elements that the passive media like movies do not offer.

Alongside the growth and maturation of the gaming industry, also the path to become a game designer has been under a change. Schools and institutions on different levels of the education system offer courses and diploma studies for aspiring game designers. Traditionally, game designers learned their craft through personal experience (playing games and analyzing existing games) and experimentation (designing games).

These changes are also reflected on the published game design related literature. In 1984, book by Chris Crawford started the still on-going trend of the game design literature which concentrates on sharing of the wisdom of the experts in the form of experiences, guidelines and taxonomies (e.g. Rollings & Morris, 2003; Rollings & Adams, 2003; Zimmerman & Salen, 2004; Barwood & Falstein, 2006; Schell, 2008). Crawford's book also serves as an example of game design knowledge moving to more formal direction from the roots of self-learning and expertise gained in practice. A decade after Crawford, Costikyan (1994; updated 2002) started work on shared vocabulary for game designers as he saw this as a basic requirement for the analysis and understanding what kind of game design choices work and makes games interesting.

The developmental branch for the game design patterns was started by Kreimeier (2002a), and Björk and Holopainen (2002). Game design patterns are game design tools, but they can also be used as tool for communication between different parties, documentation of the game design, and as a source for automatic code generation from the pattern model in some cases.

In this paper we continue from the work of Almeida et al. (2013), who reviewed systematically the tools and methods developed for game design, by expanding the branch

containing the game design patterns. In the next section we shortly describe the method used to find the key literature and how the material was analyzed. The following section represents the birth and history of game design patterns. After that, we present the current state of the field in a form of a simplified taxonomy and describe its contents to the readers. Then, in the final section we discuss the shortcomings, strengths, and future directions of game design patterns. We conclude the paper with a call for the game design pattern language, which could unify the field, further advancing its maturation as a usable design tool.

METHODOLOGY

Methodology behind this work is a literature review in the form of a systematic mapping of the literature (Grant et al., 2009). The systematic mapping was done by searching databases (e.g. ACM, IEEE, and Web of Science) with the search string "game design pattern". Literature was also searched from game industry sites (e.g. Gamasutra), industry events (e.g. Game Developers Conference) and from known game-related academic conferences (e.g. GameOn, International Workshop on Games and Software Engineering (GAS) at International Conference on Software Engineering (ICSE), and Foundations of Digital Games).

From these sources the relevant articles were identified, based on their content. Content of the articles had to describe game design patterns, either relating to general gameplay or some specific are of game design. The results were further complemented by following the references in these papers. Also, a freeform search were conducted on the popular web search engine Google and its academic counterpart Google Scholar to find out about other potential sources, especially from the practical side of the game design world. This left us with 52 relevant sources for the game design patterns.

The collected material was then analyzed by going through their content. From the content their references to earlier work done on the field was noted with their stated goals and the discipline from where the work originated from. This information was then synthesized to the historical timeline and family tree of game design patterns, and the game design patterns presented in these studies were categorized based on their descriptions and features to form our simplified taxonomy of game design patterns. Other observations relevant to the history, evolution and creation of the game design patters done during this process are stated at the relevants parts of this paper.

BIRTH AND EVOLUTION OF GAME DESIGN PATTERNS

The idea of design patterns came to light in the year 1977 when Alexander et al. published their series of books concerning the architectural design. Especially the volume called *A Pattern Language: Towns, Buildings, Construction* (Alexander et al., 1977) which described 253 "good design practices" that conveyed the common wisdom gathered from the field of architecture has been influential also on other domains. Design patterns were applied on software engineering by Beck and Cunningham (1987), but were popularized on the 1990s when Gamma et al. published their influential *Design Patterns: Elements of Reusable Object-Oriented Software* (1994). Since then design patterns have been applied on a diverse set of aspects in the field of software engineering. These uses range from general software design to games (e.g. Chowdhury and Katchabaw, 2012).

Video game design is a relatively young discipline, and its methods and tools have been drawing influence from other similar fields such as books, movies, and web design. Since the field started to become more formalized, and recognized its own need for maturation, the tools and methods have been surveyed several times (e.g. Kreimeier, 2003; Lindley, 2007; Neil, 2012; Almeida, 2013).

The maturation itself begun by the call for shared and critical language by Costikyan in 1994 (1994; updated 2002). Church (1999) continued this by calling for *"Formal Abstract Design Tools"*. Taxonomies (e.g. Lindley, 2003), game design patterns (e.g. Kreimeier, 2002), frameworks (e.g. LeBlanc, 2004) and ontologies (e.g. Zagal, 2005) soon followed this call. This development has also led to several taxonomies and models which represent different aspects of game design.

One of these is Lindley & Sennersten's (2007) meta-model describing and interrelating different approaches and methodologies for game design. This meta-model has five levels on which different design methodologies are placed depending on their maturity level. Meta-model is presented on the following Figure 1.

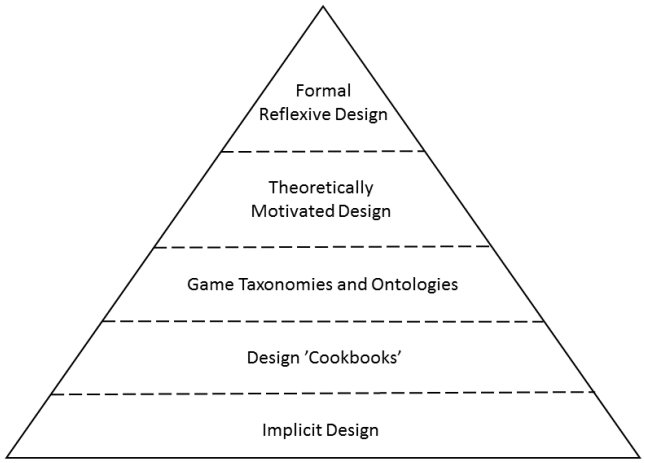


Figure 1. Game Design Meta-Model

On the lowest level of this model are the game design practices based on Implicit Design. At this level the game designs are based on the experience of the designers, which is cumulated from past experiences and from bits of knowledge shared by the peers (e.g. Crawford, 1984). On the second level, the experience is shared in the forms of guidelines, checklists and rules. These collections of knowledge are called Design Cookbooks and they are the simplest form of organization for the cumulative knowledge

present on the craft. (e.g. "The 400 rules project" by Falstein & Barwood, 2002)

Third level includes ontologies, taxonomies and game design patterns which describe game elements and design concepts used in games in a structured manner. But they do not explain why specific design choices are made. This is a requirement for the fourth level, where design tools have to be able to explain why the design choices were made and how effective they might be compared to other choices. The fifth stage is meant for design tools and ideas that potentially allow the designers to create experimental and unique games exploring the nature of games as a medium (Lindley & Sennersten, 2007). In this paper, we concentrate on the third level, and especially on the game design patterns.

In 2002 Bernd Kreimeier called for more formal design methods, namely game design patterns, at the Game Developer Conference 2002 roundtable event in March 2002 (2002a) and in a Gamasutra article on the same month (2002b). These patterns have their roots in Alexander et al.'s (1977) architectural design patterns, and Kreimeier called his patterns "Alexandrian patterns", which were noted to be *"simple collections of reusable solutions to solve recurring problems"*.

He defined game design patterns as follows: *"In a nutshell, patterns are simply conventions for describing and documenting recurring design decisions within a given context, be it game design or software engineering"*. He also classified his patterns as content patterns, differentiating them from the software engineering patterns which are used to describe the structure of the software, not the content.

Simultaneously and independently to the work of Kreimeier, Björk and Holopainen (2002) had started their work on game design patterns, and held the first workshop on them at Computer Games & Digital Cultures 2002 conference in the beginning of June 2002 (personal correspondence with Holopainen, 27.8.2015). To Björk and Holopainen the patterns were *"commonly recurring parts of the design of a game that concern gameplay"* (2004).

As a historical side note, the first identified usage of Kreimeier's pattern style was Ekström (2002) who used them to sketch out multiplayer design patterns for his personal massively multiplayer online game (MMOG) -project. This project has been dormant since 2002.

Kreimeier, Holopainen, and Björk combined their efforts, and held a game design pattern lecture at the GDC 2003 (Kreimeier et al., 2003). Since this, Kreimeier has been concentrating on other topics related to his work as a programmer and software engineer in game industry. From these early steps the work continued, and new authors diversified game design patterns from general gameplay patterns to new directions. This development will be described in the following chapter where different game design pattern collections are presented and classified.

TAXONOMY OF GAME DESIGN PATTERNS

We group the separate game design pattern collections to three main groups, based on the analysis of their descriptions and where they fit on the simplified model of game design process. In this paper, the game design process has been simplified to three aspects, *Guidelines for Intent* which sets the limits and goals of the game development process which steers the Guidelines for Design. *Guidelines for Design*, in turn, provides the groundwork for the activities and experiences offered to the player. *Guidelines for Implementation* contains the game design patterns that have the closest resemblance to the Alexandrian patters as they describe the audiovisual and concrete story related assets (e.g. level design, dialogues, and non-player characters) that converts the design to artefact.

These three aspects are not separate from each other, instead they influence each other during the game design and implementation further complicating the overall process. In the following chapters we keep them separate from each other to simplify the taxonomy.

Guidelines for Intent

These patterns act as guides for the design work and steer the decisions specific to certain types of games and goals that the game design process has.

Patterns belonging to the groups **Design Intent** and **Dark Patterns** (Zagal et al., 2013) have many common features. Both contain intents that game developers want to achieve with the game, be it motivating them in learning a new skill (Kiili, 2010; Kelle, 2012; Dormann et al., 2013), use serious games for education (Plass & Homer, 2009; Hyunh-Kim-Bang et al., 2010; Plass et al., 2010; Ibrahim et al., 2011), or change their behavior in some way (Holopainen & Björk, 2008; Lewis et al., 2012; Lewis, 2013; Ašeriškis & Damaševičius, 2014). Patterns belonging to the Dark Patterns are considered to be ethically problematic. They are defined by Zagal et al. (2013) to be *"a pattern used intentionally by a game creator to cause negative experiences for players which are against their best interest and likely to happen without their consent."* Also, game project might have **Economical patterns** (Zagal et al., 2013) to guide the economic aspects of game design.

Guidelines for Design

Game Type specific patterns are mostly relevant with some types of games, which has led to the development of highly specialized **genre-specific patterns**. Representative of these are the patterns for stealth games (Hu, 2014) that concentrate solely on the aspects of how to create games to use stealth as a game mechanic. Cermak-Sassenrath has been working on patterns that are derived from popular games from the 1980's (Cermak-Sassenrath 2012a, 2012b, 2012c). Also, some platform specific patterns have been developed, e.g. for mobile games (Davidsson et al., 2004).

Another interesting group of specialized patterns are for role playing games (RPGs) by Kirk et al. (2006). As the name states, these patterns describe how to create RPG type of games. Their roots are in the board games, but are applicable also to the computer RPGs. Interesting aspect of these patterns is that they do not share the otherwise common roots to Kreimeier, Björk and Holopainen. Instead they are derived straight from the works of Gamma et al. (1994), as their main creator Kirk has background in programming and software engineering.

Game(Play) design patterns were originally described to be collections of shared design vocabulary. Aim of the shared vocabulary is to enable the designers to communicate efficiently with each other and document their experience in written format for other game designers. Patterns also make it possible to analyze existing games using this same vocabulary, even if games were not designed by using them. (Kreimeier, 2002a; Björk et al., 2003)

Kreimeier created the first published patterns on 2002 (2002a). In this seminal work he describes seven patterns concerning general game design, e.g. Predictable Consequences. Björk et al. published their work shortly after Kreimeier (2003), in which they state that they had so far found over 200 game design patterns. Currently this game design pattern wiki contains 536 patterns and their descriptions (Björk, 2015).

Guidelines for Implementation

Patterns in this group are essentially level design patterns. In this group the patterns range from the scale of the individual objects sized from small rock to massive open worlds (Level Design Patterns, 2015).

Environment patterns are closest to the original Alexandrian patterns as they concern the architectural features of the game's graphical visualization of its world (Hullett et al., 2010; Dahlskog et al., 2012; Dahlskog et al., 2015). There are also sub patterns that aim to guide the player's movement (Milam et al., 2010; Lannigan, 2014) and/or attention (Milam et al., 2012) to certain directions on the game world with these features.

Patterns for Assets are about creating the artefacts that are part of the game world that make the game world and story more life-like, e.g. sounds (Alves & Roque, 2010; Sound Design in Games, 2010), or artefacts that the player can use e.g. weapons (Giusti et al., 2012), and potentially other objects like vehicles (Level Design Patterns, 2015).

Interaction patterns differ from other two subgroups in the sense that they are guidelines to design and implement non-audiovisual elements of the game world that the players experience. These patterns are applicable to quests (Smith et al., 2011), conflicts (Lankoski and & Björk, 2007; Lankoski and & Björk, 2008), dialogue (Brusk & Björk, 2009), NPCs (Lankoski & Björk, 2007; Rivera et al., 2012), behavior (Pellens et al., 2008), social networks in the game (Lankoski and & Björk, 2007), social interaction between player and NPCs or other players (Bergström et al., 2010; Reichart & Bruegge, 2014; Reichart & Bruegge, 2015), cooperation/collaboration (Rocha et al., 2008; Seif El-Nasr et al., 2010; Reuter et al., 2014) and AIs (Treanor et al., 2015).

Simplified Taxonomy of Game Design Patterns

In this simplified taxonomy, patterns in higher level are grouped according to their role in the game design and implementation process. Inside this broad grouping, there are seven subgroups, based on the aspects of the game design they are related to. This categorization is visualized in Figure 2.

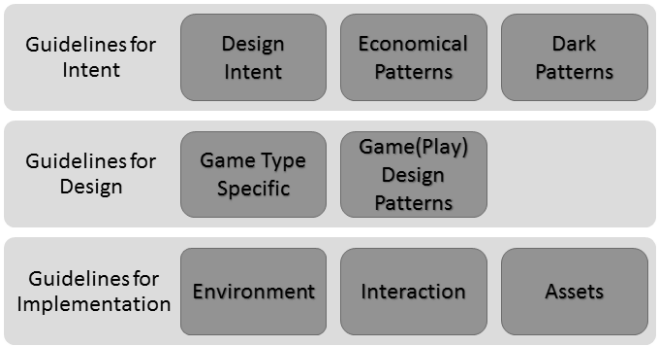


Figure 2: Categorization of game design pattern collections

Game(Play) Design pattern group, which originally started the development of these patterns is the most difficult to break in to smaller groups. Even the original authors have adopted different classifications for the patterns depending on their usage cases (more on these at Björk (2015).

DISCUSSION

In this part we discuss some problems related to the game design patterns that were identified during the search and analysis phase. These problems relate to the fragmentation of the patterns, their current usage, and their maturity in general.

It is cumbersome to go to more detailed level and to properly form subgroups for all the available individual game design patterns from different game design pattern collections. The reason for this is that the authors of these collections do refer to each other's work but create their own separate collections which use their own conventions to describe patterns. This creates a highly fragmented field, leading to overlapping patterns, incompatible naming and, more importantly, incompatible pattern templates and creation styles – (ranging from Alexander et al.'s (1977) notation to Kreimeier (2002a), or Björk & Holopainen (2003), UML based (Ašeriškis & Damaševičius, 2014), or some modified form of Alexander et al. (Björk & Holopainen, 2003) or Björk & Holopainen (Hu, 2014)).

Game design patterns have been criticized notably by Folmer (2006), McGee (2007) and Dormans (2013). All of these authors have critiqued game design patterns for including the term "pattern", but deviating from the Alexander et al.'s (1977) and Gamma et al.'s (1994) problem - solution pairing

principle, in which the pattern is known solution to a known problem. For this reason, Folmer compares the game design patterns to heuristics, and Dormans to design vocabularies and taxonomies. In the words of McGee, *"This expands the original Alexander usage from just problem-solution pairs to include patterns that are less precise or that support creative experimentation"*. In defense of patterns supporting creative experimentation, Alexandrian patterns were not meant to be strict rules, instead they are guidelines to be used when designers encounter a problem.

As a reaction to respond to this criticism, game design pattern creators could look back into the software design patterns where their own roots are. Research of the software design patterns include specific section devoted to the pattern writing and creation (which was noted by Reichart & Bruegge, 2015), e.g. Meszaros & Doble (1998) and Wellhausen & Fießer (2011).

To properly address these problems, game design patterns community should move towards the creation of game design pattern language. Some parts of it have already done it, the most comprehensive of them being the sound design for games (Alves & Roque, 2010). In a smaller scale, Cerman-Sassenrath (2012b) has done the same with "old school action games". In this move, the previous work done by the software design patterns community could be a helpful source. Another example is also the unification attempt started by Zavcer et al. (2014) in the field of patterns for serious games design.

In the issue of maturity, a game design pattern language and unified creation templates for the patterns could be a beginning for the move towards the fourth level of the Lindley & Sennersten's (2007) maturity model, Theoretically Motivated Design. Game pattern language would make it easier to create comprehensive theoretical background and tests for the effectivity of the existing patterns. In this sense, Milam & Seif El-Nasr (2010) and Milam et al. (2012) have already begun this work by testing the effectivity of their patterns.

CONCLUSION

In this paper, we looked at the development history of the game design patterns and their current state. Our methodology was the analysis of the material found on the systematic mapping of the literature. We observed how game design patterns originated from software design patterns, and diversified to cover various aspects of the overall game design.

Also, we observed how this diversification also led to a fragmentation as authors on the different subfields modified the pattern templates to their own needs and created patterns that unnecessarily replicate patterns from other pattern collections and are incompatible with them.

Based on our analysis of the game design patterns and game design pattern collections, we recognized the need for unification of game design patterns. Specifically this field of research and practice as whole could benefit from a common unified game design patterns language to amend the problems created by the current state of fragmentation.

From the material we constructed the simplified taxonomy of game design patterns. On the top level, this taxonomy categorizes the patterns based on their role on the game design process. These top level categories are Guidelines for Intent, Guidelines for Design and Guidelines for Implementation.

Inside these levels, the patterns are categorized based on which aspect of the game design they influence during the game design process. In the Guidelines for Intent, Design Intent, Economical and Dark patterns steer the game design to specific use when the game is created with other motives than pure entertainment.

Guidelines for Design contains the main patterns used in the game design process. Game(Play) Design patterns describe the core mechanics, and goals of the game, while Game Type specific patterns are used to supplement them when game designers' goals is to make a game for specific genre.

Patterns belonging to the Guidelines for Implementation are used when game designers are creating the content for the game world. The audiovisual and story related elements are created with the of the Environment, Interaction and Assets patterns. From the material we constructed the simplified taxonomy of game design patterns. On the top level, this taxonomy categorizes the patterns based on their role on the game design process. Inside this level, the patterns are categorized based on which aspect of the game design they influence during the game design process.

The future work on this matter requires more concentration on assorting the second level pattern taxonomy to more fine graded groups. Currently some of these groups contain diverse assortment of patterns, which might benefit from clearer categorization.

ACKNOWLEDGEMENT

This article was done as a part of the Gamified Solutions in Healthcare research project. The project is conducted by University of Turku and Turku University of Applied Sciences together with partners Puuha Group, GoodLife Technology, City of Turku and Attendo. The project is funded by Tekes – the Finnish Funding Agency for Innovation.

REFERENCES

Alexander, C., Ishikawa, S. and Silverstein, M. (1977). A Pattern Language: Towns, Buildings, Construction. Vol. 2. Oxford University Press, 1977.

Almeida, M.S.O. and Silva, F.S.C. (2013). A Systematic Review of Game Design Methods and Tools. Lecture Notes in Computer Science 8215. Springer, Berlin and Heidelberg, pp.17-29, 2013, http://dx.doi.org/10.1007/978-3-642-41106-9_3

Beck, K. and Cunningham, W. (1987). Using Pattern Languages for Object-Oriented Programs. Technical Report CR-87-43, Tektronix, Inc., September 17, 1987. Presented at the OOPSLA'87 workshop on Specification and Design for Object-Oriented Programming.

Björk, S. and Holopainen, J. (2002). Computer Game Design Patterns. One-day workshop Computer Games & Digital Cultures 2002 conference, Tampere, Finland.

Björk, S., & Holopainen, J. (2004). Patterns in game design (game development series). 1. ed., Charles River Media, December 2004.

Chowdhury, M. I., & Katchabaw, M. (2012). Improving software quality through design patterns: a case study of adaptive games and auto dynamic difficulty. In the Proceedings of the GAMEON'2012, Eds. Antonio J.Fernandez-Leiva, Carlos Cotta Porras and Raul Lara Cabrera. November 14-16, 2012, University of Malaga, Malaga, Spain. ISBN 978-90-77381-74-8.

Church, D. (1999). Formal Abstract Design Tools. Gamasutra (July 1999), http://www.gamasutra.com/view/feature/3357/formal abstract design tools.php

Costikyan, G. (1994). I have no words and I must design. nteractive Fantasy# 2. British roleplaying journal.

Costikyan, G. (2002). I have no words & I must design: Toward a critical vocabulary for games. In F. Mdiyrd (Ed.), Proceedings of the Computer Games and Digital Cultures Conference (pp. 9-33). Tampere, Finland: Tampere University Press.

Crawford, C. (1982). The art of computer game design. Vancouver, WA: Washington State University. Available at http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html

Dormans, J. (2013). Making design patterns work. In Proceedings of the Second Workshop on Design Patterns in Games, DPG '13, in association with Foundations of Digital Games, FDG'13, Chania, Greece, 2013.

ESA (2015). Essential Facts about the Computer and Video Game Industry. Report, available at: http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf

Falstein, N., Barwood, H. (2002). More of the 400: Discovering Design Rules. Presentation at GDC 2002, available at http://www.gdconf.com/archives/2002/hal_barwood.ppt

Gamma, E., Vlissides, J., Johnson, R., and Helm, R. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Grant, M. J., & Booth, A. (2009). A typology of reviews: an analysis of 14 review types and associated methodologies. Health Information & Libraries Journal, Vol. 26, Iss. 2, pp. 91-108.

Hunicke, R., LeBlanc, M., and Zubek, R. (2004). MDA: A formal approach to game design and game research. In: Proceedings of the AAAI 2004 Workshop on Challenges.

Kreimeier, B. (2002a). The Case for Game Design Patterns. Gamasutra.com, March 13, 2002. Available at: http://www.gamasutra.com/view/feature/132649/the_case_for_game_design_patterns.php

Kreimeier, B. (2002b). Content Patterns in Game Design. GDC 2002, March 19-23, Roundtable. Available at: http://www.onearrow.org/game/pattern/

Kreimeier, B. (2003). Game Design Methods: A 2003 Survey. Gamasutra, 2003. Available at: http://www.gamasutra.com/view/feature/131301/game_design_methods_a_2003_survey.php?print=1

Kreimeier, B.; Holopainen, J. and Björk S. (2003). Game Design Patterns. Lecture Notes from GDC 2003 (March 6).

Lindley, C. A. (2003). Game Taxonomies: A High Level Framework for Game Analysis and Design. Gamasutra. Available at: http://www.gamasutra.com/features/20031003/lindley_01.shtml

Lindley, C.A., and Sennersten, C.C. (2007). An Innovation-Oriented Game Design Meta-Model Integrating Industry, Research and Artistic Design Practices. In O. Leino, H. Wirman & A. Fernandez (Eds.), Extending Experiences. Structures, Analysis and Design of Computer Game Player Experiences (pp. 250-271). Rovaniemi, Finland: Lapland University Press.

Neil, K. (2012). Game Design Tools: Time to Evaluate. Proceedings of DiGRA Nordic 2012 Conference: Local and Global Games in Culture and Society (2012).

Meszaros, G., and Doble, J. (1998). A pattern language for pattern writing. In R. C. Martin, D. Riehl, & F. Buschmann (Eds.), Pattern languages of program design, Vol. 3, pp. 529–574. Reading, MA: Addison-Wesley.

Rollings, A., & Morris, D. (2003). Game architecture and design: a new edition. New Riders.

Rollings, A., & Adams, E. (2003). Andrew Rollings and Ernest Adams on game design. New Riders.

Salen, K., & Zimmerman, E. (2004). Rules of play: Game design fundamentals. MIT press.

Schell, J. (2008). The Art of Game Design: A book of lenses. CRC Press.

Smith, G. (2012). Expressive Design Tools: Procedural Content Generation for Game Designers (Ph.D. thesis). University of California, Santa Cruz.

Zagal, J., Mateas, M., Fernandez-Vara, C., Hochhalter, B., and Lichti, N. (2005). Towards an Ontological Language for Game Analysis. Proceedings of the DiGRA 2005 Conference: Changing Views - Worlds in Play, Vancouver, BC, Canada.

Zavcer, G., Mayr, S., and Petta, P. (2014). Design Pattern Canvas: Towards Co-Creation of Unified Serious Game Design Patterns. In 6th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), 2014, 9-12 Sept. 2014.

Wellhausen, T., & Fießer, A. (2011). How to write a pattern. European Conference on Pattern Languages of Programs, EuroPloP '11.

Pattern references

Alves, V., & Roque, L. (2010). A pattern language for sound design in games. In Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound (p. 12). ACM.

Ašeriškis, D., & Damaševičius, R. (2014). Gamification Patterns for Gamification Applications. Procedia Computer Science, 39, 83-90.

Bergström, K., Björk, S., & Lundgren, S. (2010). Exploring aesthetical gameplay design patterns: camaraderie in four games. In Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments (pp. 17-24). ACM.

Björk, S. (2015). Gameplay design patterns collection, available at http://129.16.157.67:1337/mediawiki-1.22.0/index.php/Category:Patterns . Checked at 15.2015.

Björk, S. and Holopainen, J. (2005). Games and Design Patterns. In The Game Design Reader: A Rules of Play Anthology, Salen, K. & Zimmerman, E. (Eds). ISBN 0-262-19536-4. MIT Press.

Björk, S. and Holopainen, J. (2004) Patterns in Game Design. Charles River Media. ISBN 1-58450-354-8.

Björk, S., Lundgren, S. and Holopainen, J. (2003) Game Design Patterns. In Copier, M. & Raessens, J. (Eds.) (2003) Level Up - Proceedings of Digital Games Research Conference 2003, Utrecht, The Netherlands, 4-6 November 2003.

Brusk, J. and Björk, S. (2009). Gameplay Design Patterns for Game Dialogues. Paper presentation at DiGRA 2009: Breaking New Ground: Innovation in Games, Play, Practice and Theory. London, UK.

Cermak-Sassenrath, D. (2012a). Experiences with design patterns for old school action games. In Proceedings of the 8th Australasian Conference on Interactive Entertainment: Playing the System (p. 14). ACM.

Cermak-Sassenrath, D. (2012b). A Design Pattern Language for Old school Action Games. In Proceedings of The 2nd International Conference on DESIGN AND MODELING IN SCIENCE, EDUCATION, AND TECHNOLOGY: DeMset 2012. (pp. 194-9)

Cermak-Sassenrath, D. (2012c). Designing Games with Patterns. In: Colab Journal: Creative Technologies. Vol. 3, special issue on interactivity, 2012.

Dahlskog, S. and Togelius, J. (2012). Patterns and procedural content generation: revisiting Mario in world 1 level 1. In Workshop Proceedings of the 7th International Conference on the Foundations of Digital Games.

Dahlskog, S., Togelius, J., & Björk, S. (2015). Patterns, dungeons and generators. In Proceedings of the 10th Conference on the Foundations of Digital Games

Davidsson, O., Peitz, J., & Björk, S. (2004). Game design patterns for mobile games. Project report to Nokia Research Center, Finland.

Dormann, C., Whitson, J. R., & Neuvians, M. (2013). Once More with Feeling Game Design Patterns for Learning in the Affective Domain. Games and Culture, 1555412013496892.

Ekström, Olkof (2002). Multiplayer Design Patterns. Available at https://www.abc.se/~m10383/Haven/General/Multiplayer_Design_Patterns.html

Folmer, E. (2006). Usability patterns in games. Future Play.

Giusti, R., Hullett, K., & Whitehead, J. (2012). Weapon design patterns in shooter games. In Proceedings of the First Workshop on Design Patterns in Games (p. 3). ACM.

Holopainen, J. & Björk, S. (2008). Gameplay Design Patterns for Motivation. Paper presentation at ISAGA 2008, Kaunas, Lithuania.

Hu, M. (2014). Game Design Patterns for Designing Stealth Computer Games.

Hullett, K., & Whitehead, J. (2010). Design patterns in FPS levels. In proceedings of the Fifth International Conference on the Foundations of Digital Games (pp. 78-85). ACM.

Huynh-Kim-Bang, B., Wisdom, J., & Labat, J. M. (2010). Design patterns in serious games: A blue print for combining fun and learning. Project SE-SG, available at http://seriousgames. lip6. fr/DesignPatterns.

Ibrahim, A., Vela, F. G., Sánchez, J. L. G., & Zea, N. P. (2011). Playability design pattern in educational video game. In Proceedings of the 5th European Conference on Games Based Learning (pp. 282-290). Academic Conferences Limited.

Kelle, S. (2012). Game design patterns for learning. Shaker Verlag, Aachen.

Kiili, K. (2010). Call for learning-game design patterns. Chapter in a book Educational Games: Design, Learning, and Applications, pp. 299-311. Editors: Frej Edvardsen and Halsten Kulle. Nova Publishers, 2010. ISBN: 978-1-60876-692-5

Kirk III, W. J., Cantrell, M. R., & Holmes, M. (2006). Design Patterns of Successful Role-Playing Games.

Kreimeier, B. (2002a). The Case for Game Design Patterns. Gamasutra.com, March 13, 2002. Available at: http://www.gamasutra.com/view/feature/132649/the_case_for_game_design_patterns.php

Lankoski, P., & Björk, S. (2007). Gameplay Design Patterns for Believable Non-Player Characters. Paper presentation at DiGRA 2007, Tokyo, Japan.

Lankoski, P., & Björk, S. (2007). Gameplay Design Patterns for Social Networks and Conflicts. Paper Presentation at Computer Game Design and Technology Workshop, John Moores University, Liverpool.

Lankoski, P., & Björk, S. (2008). Character-Driven Game Design: Characters, Conflicts, and Gameplay. Paper presentation at GDTW, Sixth International Conference in Game Design and Technology, 2008.

Lannigan, R. (2014). Developing Player Movement Design Patterns in Multiplayer Video Games. In the Proceedings of the GAMEON'2014, Ed. Patrick Dickinson. September 9 - 11, 2014, University of Lincoln, Lincoln, United Kingdom. ISBN 978-90-77381-85-4.

Level Design Patterns (2015). Available at: https://ldp.soe.ucsc.edu/doku.php?id=start

Lewis, C., Wardrip-Fruin, N. and Whitehead, J. (2012). Motivational game design patterns of 'ville games. In Proceedings of the International Conference on the Foundations of Digital Games (FDG '12). ACM, New York, NY, USA, pp. 172-179. DOI=10.1145/2282338.2282373. Available at: http://doi.acm.org/10.1145/2282338.2282373

Lewis, Chris. (2013). Motivational Design Patterns. UC Santa Cruz: Computer Science. Doctoral thesis, retrieved from: http://escholarship.org/uc/item/30j4200s

McGee, K. (2007). Patterns and computer game design innovation. In IE '07: Proceedings of the 4th Australasian conference on Interactive entertainment. Melbourne, Australia, RMIT University, 2007, pp. 1-8.

Milam, D. and Seif El-Nasr, M. (2010). Design Patterns to Guide Player Movement in 3D Games. Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games, 2010.

Milam, D., Bartram, L., & El-Nasr, M. S. (2012). Design patterns of focused attention. In Proceedings of the First Workshop on Design Patterns in Games (p. 5).

Pellens, B., De Troyer, O., & Kleinermann, F. (2008). CoDePA: A conceptual design pattern approach to model behavior for X3D worlds. In Proceedings of the 13th International Symposium on 3D Web Technology

Plass, J. L., & Homer, B. D. (2009). Educational game design pattern candidates. Journal of Research in Science Teaching, Vol. 44, Iss. 1, pp. 133-153.

Plass, J. L., Homer, B., & Perlin, K. (2010). Research on design patterns for effective educational games. In Game Developers Conference 2010.

Reichart, B., & Bruegge, B. (2014). Social interaction patterns for learning in serious games. In Proceedings of the 19th European Conference on Pattern Languages of Programs (p. 22). ACM.

Reichart, B., & Bruegge, B. (2015). Serious Game Patterns for Social Interactions. In Proceedings of the IADIS Multiconference on Computer Science and Information Systems (MCCSIS) 2015, Game and Entertainment Technologies 2015, Eds. Katherine Blashki and Yingcai Xiao. July 21 - 24, Las Palmas de Gran Canaria, Spain. ISBN: 978-989-8533-38-8.

Reuter, C., Wendel, V., Göbel, S., and Steinmetz, R. (2014). Game Design Patterns for Collaborative Player Interactions. DiGRA, 2014.

Rivera, G., Hullett, K., & Whitehead, J. (2012). Enemy NPC design patterns in shooter games. In Proceedings of the First Workshop on Design Patterns in Games (p. 6). ACM.

Rocha, J. B., Mascarenhas, S., & Prada, R. (2008). Game mechanics for cooperative games. ZON Digital Games 2008, pp. 72-80.

Seif El-Nasr, M., Aghabeigi, B., Milam, D., Erfani, M., Lameman, B., Maygoli, H., & Mah, S. (2010). Understanding and evaluating cooperative games. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 253-262). ACM.

Smith, G., Anderson, R., Kopleck, B., Lindblad, Z., Scott, L., Wardell, A., Whitehead, J., & Mateas, M. (2011). Situating quests: Design patterns for quest and level design in role-playing games. In Interactive Storytelling (pp. 326-329). Springer Berlin Heidelberg.

Sound Design in Games (2010). Website. Audio Mostly 2011 Conference, September 7-9, 2011. Available at: http://www.soundingames.com/index.php?title=Main_Page

Treanor, M., Zook, A., Eladhari, M. P., Togelius, J., Smith, G., Cook, M., Thompson, T., Magerko, B., Levine, J., & Smith, A. (2015). AI-Based Game Design Patterns. In 10th International Conference on the Foundations of Digital Games, 2015.

Zagal, J.P., Björk, S. and Lewis, C. (2013). Dark Patterns in the Design of Games. Foundations of Digital Games 2013, May 14-17, 2013, Crete, Greece.

TRANSLATING A MODERN FILM AND TV SCREENING ROOM TO AN INTEGRATED GAME ENGINE PRODUCTION ENVIRONMENT

Oliver Engels and Robert Grigg
International Game Architecture and Design (IGAD)
NHTV Breda University of Applied Sciences
4800 DX Breda, The Netherlands
E-mail: 120010@nhtv.nl

KEYWORDS
Screening Room, Methodology, Game Engine, Tools, Quality Assurance

ABSTRACT

This research paper presents a Screening Room tool, inspired from the Film and TV industry, that is then applied to the game industry. The work involved the creation and development of several in-game prototypes of a Screening Room tool within the game engine Unity 5. The iterative development involved usability testing and benchmarking against an example of current game development quality assurance processes that use BugZilla. The prototype resulting from the third iteration was then tested within a commercial games company CodeGlue and it was found that an overall time saving of approximately 500% was made whilst usability being arguably better for bug finding staff whilst similar to more complex for the bug fixing staff.

INTRODUCTION

This paper explores the research question of: *Would the features and functionality of a Film and TV Screening Room (SR) tool be beneficial for the game production process if applied to a modern game engine and its pipeline?*
The art pipeline within game development sometimes rely on software used in the Film and TV industry; the software is called Screening Room (SR) or Reviewing Room (RV). A SR can handle tasks such as the review, feedback, and possible version selection of certain assets. Such SR tools include TACTIC *(Technology Southpaw, 2005)*, Shotgun Software *(Autodesk, 2013)*, FTrack *(FTrack Inc, 2015)*, and Atlassian Software *(Atlassian Pty Ltd, 2015)* amongst others. This paper explores whether a number of features from SR type tools could be better utilized in a game development pipeline.
To research this question the following two hypotheses are established:
H1. Hypothesis One: *Integrating a film-style Screening Room into a modern game engine improves productivity in a game production pipeline.*
H2. Hypothesis Two: *The use of a game engine integrated Screening Room tool is easier to use over existing methods and tools that it replaces.*
A Proof of Concept will be generated in Unity 5 *(Unity Technology, 2015)* engine as this engine currently holds 45%[1] of the market share. Also a large number of testers are used to Unity 5 *(ibid.)*, setting the focus more on the tool than on the engine. As for the scope of this research, this paper and experiment will focus on the Quality Assurance (QA) processes within the game development pipeline.

SCREENING ROOM (SR)

Figure 1: SR - Shotgun Software Screening Room *(Autodesk, 2013)*.

Within the industry there is no real difference between a SR or a RV. Shotgun Software *(Autodesk, 2013)* uses the name SR for their tool while other software uses the naming of RV for their tools. The rooms allow users to view, compare, and select various assets. These assets can be selected to become part of the final product. Next to this, it also allows users to generate annotations on the assets to quickly spot problems within a given scene or particular asset *(see Figure 1)*. This feedback will then be sent to the user responsible for the asset to quickly notify and solve a given problem. This approach makes it possible for each team member to make annotations and review each others work. Usually this is done

[1] This number from the current Unity 5 website: *https://unity3d.com/public-relations (Accessed: January, 2015).*

through a specific department that would give feedback on shots and cuts from a movie while also handling feedback from clients or publishers *(Dunlop, 2014)*. The use of a SR aims to increase product quality by making the production pipeline more efficient through improvement of ease-of-use, enhancing communication throughout the production team and with associated stakeholders.
In addition, if just before a major release a problem is found within a newer version of the asset, the development team has the tools to quickly switch to an older version if that fixes the problem. Shotgun Software *(Autodesk, 2013)* also makes it possible for development teams to view, compare, and choose a different versions of the product through a side-by-side comparison.

SCREENING ROOM TODAY

Currently SR tools are used within the production pipeline of professional teams around the world to produce their products. Shotgun Software uses an app called *Revolver*[2] to review, annotate, and compare different cuts and/or screens from a movie directly from their desktop.
Today, Shotgun Software *(ibid.)* is arguably the leading package for Film and Games. Within the Games industry *Electronic Arts (Arts, 1982)*, *Blizzard (Blizzard, 1994)*, and *Microsoft Games Studios (Microsoft, 2002)* use this tool to streamline their production pipeline. This SR technology, however, is not utilized across all assets of the Games Development pipeline. Currently the game industry uses Shotgun Software *(Autodesk, 2013)* to publish, review, and comment on each other's work. It helps artists to focus on the art instead of the management part of process. Models are automatically placed in the right map, together with the textures, animation, and other elements connected to the asset. Artists can view the levels in which assets are placed and will be notified by activities within Shotgun. Next to this Shotgun can open all popular tools that are used within the art pipeline, including but not limited to: *Adobe Photoshop (Systems, 2014)*, *Autodesk Maya (Autodesk, 2015b)*, *Autodesk 3ds Max (Autodesk, 2015a)* and *Houdini (Software, 2015)* making Shotgun a very powerful tool for the games industry.

QA PIPELINE PROCESS

R. Dunlop describes QA as *"The publish time process through which you technically check assets to ensure their integrity and suitability for further progression through the pipeline." (Dunlop, 2014)*. The QA process can occur both internally and externally. In-house QA processes are often inter-departmental where the development team sends the build of the game to the QA department. This department then checks the game for problems and documents the problems in a log reporting program. Examples of these programs include: *Bugzilla (Contributors, 1998)*, *Mantis (Contributors, 2002)*, or *Request Tracker (Vincent, 1995)* which are open-source and *Atlassian Jira (Atlassian Pty Ltd, 2015)* or *YouTrack (JetBrains, 2014)* which are commercial products. The second quality check is sometimes done by the manufacturer of the platform/marketplace within the final development phase of the product. The manufacturer wants to ensure quality to prevent it from reflecting badly on their hardware platform [3] *(Dunlop, 2014)*.

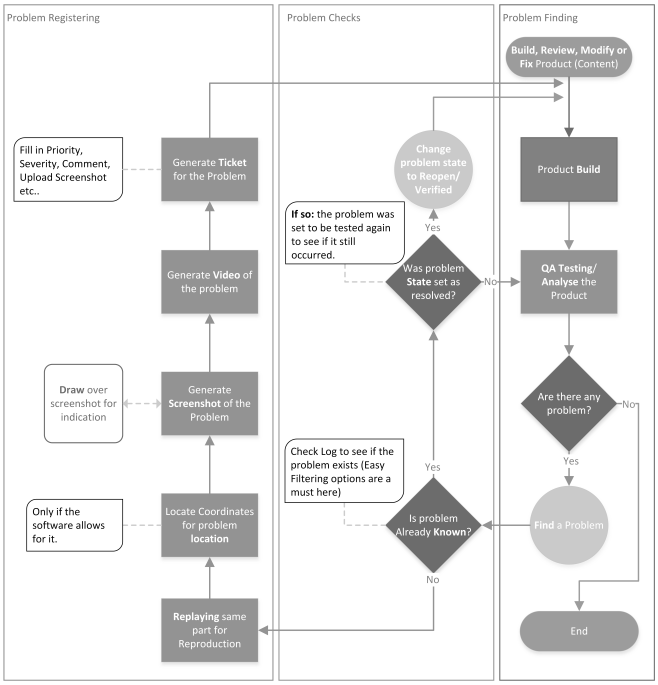


Figure 2: QA - Problem Life-Cycle and decision process.

Each bug report needs to be written down in a manner decided by the company. Some of these reports include details such as: *priority*, *severity*, *department*, *status*, *hardware*, *summary*, and a *screenshot* which often includes an *annotation drawn over the screenshot*. The priority of a problem is set by combinations of these components. Sometimes the QA staff need to play the same part of the level multiple times to detect the *Severity* of a bug. This then changes the priority of the bug. A game breaking bug can still have a low priority if the reproduction rate of the bug is very low.
Sometimes a whole department is dedicated to the finding of and registering of problems. Game-play needs to be recorded to show the problem manifests itself. Screenshots are generated and drawn upon to give the Developer more information and locations need to be registered to indicate the position of the problem. The more data the Developer has the easier the problem can be found and resolved. All this information, however, needs to be gathered and correctly registered. If not the gathered information would be useless. Managing

[2]Shotgun Software has partnered with Tweak RV to integrate their software with Shotgun's Software. The result is a standalone tool that is connected to the Shotgun framework.

[3]They test on more aspects within their tests, including but not limiting to *Game Stoppers*, *Overall durability/runtime testing*, *Button Assignment*, and other regulations.

this amount of data and the setup for all the recorded bugs is a large and important undertaking.

EXPERIMENT

Given the results from the survey of current SR tools within the industry a Proof of Concept will be constructed for this experiment using the Unity 5 *(Unity Technology, 2015)* engine and editor as the environment. Because of time constraints this experiment will focus on the QA process within the game development pipeline. A benchmark will be created using existing traditional game development QA tool BugZilla *(Contributors, 1998)* in order to perform usability tests for the developed tool against this.

Figure 3: Proof of Concept - Level test issues highlighted to guide experiment participant.

The tests will be performed by letting participants use the tool and perform tasks that required the use of various features available within the tool. The participants will be asked to fill out a survey to collect more data in regard to user experience. To keep the tests as accurate as possible each participant will be given the same set of problems. The problems will be highlighted so that they can be quickly spotted by test participants so that the focus is more on the tool than the game problem *(see Figure 3)*.
Each user test will be allowed to verbally convey and point out potential problems[4]. The final test will be done in the same manner within a professional game company called Codeglue *(De jong and Sibrandi, 2000)*, located in the heart of Rotterdam, The Netherlands.

RESULTS

In regards to hypothesis H1 it was found to be supported with data showing a time saving of 46% over that of traditional tool use. Within the bug fixing part of the tool, the difference is immense with it being over 500%. If we take a closer look at these results we immediately see that the tool has gained a higher score than the original software. Users are more

[4]This type of testing is also called the RITE Method, Rapid Iterative Testing and Evaluation. It was defined by Michael Medlock, Dennis Wixon, Bill Fulton, Mark Terrano and Ramon Romero *(Medlock et al., 2005)*.

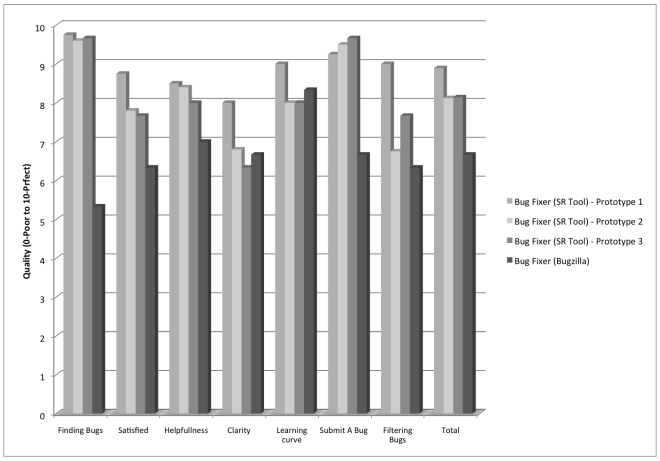


Figure 4: Bug fixing: Usability comparison of SR Tool pro-totypes and BugZilla.

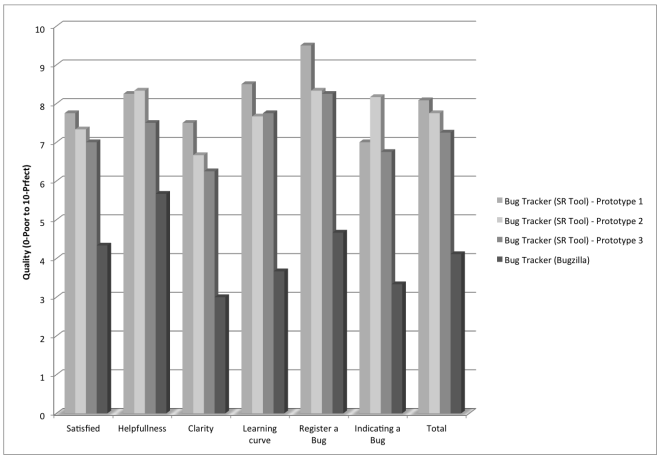


Figure 5: Bug finding: Usability comparison of SR Tool prototypes and BugZilla.

motivated to perform a given task which would benefit the project as well.

In regards to hypothesis H2 it was found to be neither completely supported or otherwise. Users rated the tool on the same level as the old bug tracking software. The learning curve for bug trackers was a significant amount lower than that of the bug fixers. This is due to the bug trackers being mostly working inside the game world, while bug fixers worked mostly within the more complicated tool interface.

When we look at Figure 6 we see that the time spent doing certain tasks was much higher in bug fixing than in bug tracking. The score for the SR engine tool was *1.83* higher than traditional tools, within the bug tracking part it was even higher with a score difference of *3.43*. In Figure 4 the score difference between traditional tools and the SR engine tool are not that significant. Nevertheless, participants where quicker with fixing a problem than with registering one.

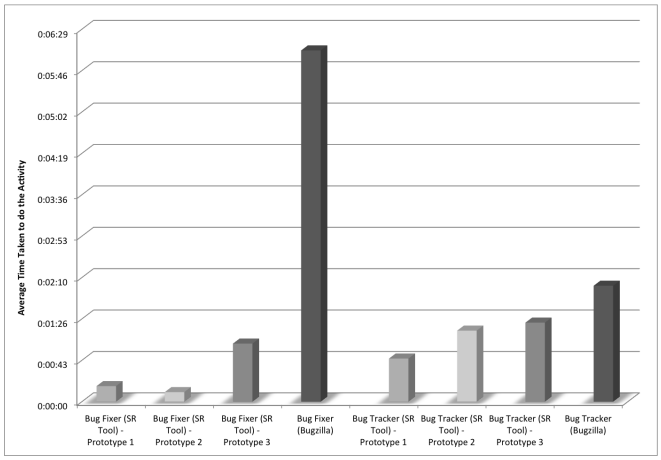


Figure 6: Unity 5 SR Tool - Shows time spent doing the tasks.

DISCUSSION

SR Tool - Prototype 1.

The first SR Tool prototype included the base functionality of the tool. Within this version there was only the possibility for the user to generate a marker within the game world and specify a *comment*, *priority*, *type* and *name* for that annotation. The bug fixer was able to look through all previously recorded bugs with the use of a *ToDo* list, visualize the bugs inside of the game engine and with the press of one button move to the location of the bug.

Participants found that the registering and finding of bugs worked exceptionally well. The user could focus on the task at hand which was the fixing of the problem. The participants also found that the learning curve was comparable to the BugZilla *(Contributors, 1998)* tool.

The SR Tool allowed users to focus on the bug itself instead of looking around the game world to find the location of the bug, which is common in traditional bug tracking software. Within BugZilla *(ibid.)* participants would have to read the entire documentation of a given bug to understand a problem properly. The SR Tool allowed a user to understand a problem immediately as it is visually recorded. Most of the time the title of a bug report and the location it referred to proved to be sufficient in conveying the problem.

All participants felt displeased about filling in the bug report form both within BugZilla *(ibid.)* and the SR Tool prototype. When given the bug report of the participants to the Developer, they would find that bug reports from BugZilla *(ibid.)* were rather difficult to solve compared to the SR Tool. Forgetting to add a visual reference to the bug report in BugZilla *(ibid.)* could turn into a problematic situation when the description itself is not properly specified. Although this behavior is discouraged, if a poorly written bug report were to be added to the SR Tool, the results would be far less problematic. Another improvement that comes with the SR Tool is that the time necessary to fill out a bug report is considerably less as it is no longer necessary to specify exact locations and visual references.

SR Tool - Prototype 2.

The second prototype of the SR Tool included a new set of features. Some of these changes included, but were not limited to: *Only showing bugs that are visible within the game world itself*, *adding a timestamp to each bug placement*, and *the option to filter bugs by different categories*. The interface was recreated to make it more understandable. Participants were given the option to disable all irrelevant information through various drop-down menus.

During a second test it became apparent that users preferred a filter option for the *ToDo* list as well. Not having the ability to edit an existing marker was also regarded to be an issue. An additional remark was the possibility to select multiple objects and attach them to an annotation. It was suggested to generate a sphere that would indicate the radius of the annotation, making it possible to select all assets within that radius. From this the fixer had an accurate selection of possible objects that could have caused the problem.

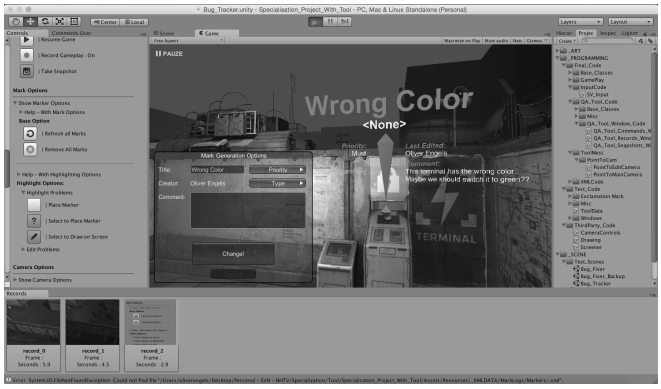


Figure 7: Unity 5 SR Tool (Prototype 3) - Shows the place-ment of an annotation.

Figure 8: Unity 5 SR Tool (Prototype 3) - Shows the record-ing of an annotated problem.

Prototype 3: Codeglue.

The third prototype included considerable changes compared to the first two prototypes. *The user interface was revised once more*, *an option was provided to edit existing annotations*, *the file manager was improved upon*, *an option for selecting multiple annotations was created*, and *the option for re-positioning or relocating an object to a selected annotation was added*. The final version of the SR Tool prototype can be seen in *Figures 7* and *8*.
Figure 5 clearly shows that the quality in bug reporting becomes lower as more features are added to the SR Tool. Figure 6 provides a similar result where more time is taken to document a bug once the software becomes more complex. The main concern with these results is that this version made it difficult for users to separate the bug-fixing section from the bug-tracking section. The control-scheme for this tool was also different from the engines native control scheme which should have been avoided. The same widget should be used to move, scale, and rotate the objects within the tool as participants did not want to learn a new set of controls.

CONCLUSION

The research in this paper has shown that there is a real demand for an in engine SR tool. Some of the functionality would be better suited outside the engine, however, most of it can be added inside. Generating annotations within a game world gives Developers instant access to the location of the error as well additional features which can be added to the annotation. Compared to BugZilla *(Contributors, 1998)* the Unity 5 *(Unity Technology, 2015)* version of the tool already was graded *1.5* higher for bug fixing and *3* higher for bug tracking.
SR tools like Shotgun Software *(Autodesk, 2013)* and FTrack *(FTrack Inc, 2015)* are perfect to optimize the development pipeline of the Art department within game development. A different type of tool, however, is needed to *Fix Gameplay Issues*, *In-game Problem Reporting*, *In-game Problem Finding*, *Client Feedback*, and *Saving time registering in engine problems* amongst others. Generating annotations and adding a number of different assets to these annotations within a game engine already resolved a large portion of these problems.
Research has also shown that the tool is not only beneficial for big companies but also for small Indie game companies that want a tool to indicate problems for later fixing. Indie game developers have said that they were going to use the SR Tool to generate their problem lists instead of writing it down on paper. Indie game developers also saw no need to fill in the large number of options in traditional bug reporting software, they simply want to place an annotation, connect objects, write a simple annotations and go on with development. When the time comes they can look through the annotations and instantly see what the issue was on that given location.

REFERENCES

Autodesk (2014). "Autodesk to Acquire Shotgun Questions and Answers". In: *Autodesk to Acquire Shotgun - Questions and Answers*, p. 7.

Dunlop, Renee (2014). *Production Pipeline Fundamentals for Film and Games*. Ed. by Renee Dunlop. 1st ed. Burlington: Taylor & Francis. ISBN: 9780415812290.

Highsmith, Jim and Alistair Cockburn (2001). "Development : The Business of Innovation". In: *Science* 34.9, pp. 120–123. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=947100.

Ilopis, Noel (2004). "Optimizing the Content Pipeline". In: *Game Developer Magazine, April 2004* April, pp. 36 – 44. URL: http://www.convexhull.com/articles/gdmag_content_pipeline.pdf.

Medlock, Michael C. et al. (2005). "The rapid iterative test and evaluation method: Better products in less time". In: *Cost-Justifying Usability*, pp. 489–517. DOI: 10.1016/B978-012095811-5/50017-1.

WEB REFERENCES

Arts, Electronic (1982). *Electronic Arts (EA)*. URL: http://www.ea.com/.

Atlassian Pty Ltd (2015). *Atlassian*. URL: https://www.atlassian.com/.

Autodesk (2013). *Shotgun Software, Inc.* URL: https://www.shotgunsoftware.com/.

– (2015a). *Autodesk 3ds Max*. URL: http://www.autodesk.com/products/3ds-max/overview.

– (2015b). *Autodesk Maya*. URL: http://www.autodesk.com/products/maya/overview.

Blizzard, Activision (1994). *Blizzard Entertainment*. URL: http://eu.blizzard.com/en-gb/.

Contributors (2002). *Mantis*. URL: http://www.mantisbt.org/.

Contributors, Ndividual bugzilla.org (1998). *Buzilla.org*. URL: https://www.bugzilla.org/.

De jong, Peter and Maurice Sibrandi (2000). *Codeglue*. Rotterdam. URL: http://codeglue.tumblr.com/.

FTrack Inc (2015). *FTrack Inc.* Stockholm, Sweden. URL: https://www.ftrack.com/.

JetBrains (2014). *YouTrack*. URL: https://www.jetbrains.com/youtrack/.

Microsoft (2002). *Microsoft Game Studios*. URL: http://www.microsoftstudios.com/.

Software, Side Effects (2015). *Houdini*. URL: http://www.sidefx.com/.

Systems, Adobe (2014). *Adobe Photoshop*. URL: http://www.adobe.com/products/photoshop.html.

Technology Southpaw (2005). *Southpaw Technology TACTIC*. URL: http://www.southpawtech.com/.

Unity Technology (2015). *Unity 5*. URL: https://unity3d.com.

Vincent, Jesse (1995). *Request Tracker*. URL: https://www.bestpractical.com/rt/.

INTERACTIVE EDUCATIONAL GAME DESIGN

SERIOUS GAME CREATION IN TEACHING CONTENT

Merikki Lappi and Esa Lappi
Paivola School of Mathematics
Päivölän opisto
Päivöläntie 52
37770 Tarttila
Finland
E-mail: merikki.lappi@paivola.fi

KEYWORDS
Serious gaming, educational gaming, game creation, modeling, cooperative learning, inquiry-based learning.

ABSTRACT
In Päivölä School of Mathematics a group of high school students learned modeling and subject matter considering a developing country, or a collapsed state, by creating an educational serious game. The students played the game as decision makers, with a variety of parameters and analyses the results. The original idea was given to the students by the international data farming community before the academic year 2013-14 and the result was a Humanitarian Assistance and Global warming -simulator. Over the following years different students have continued the work with a variety of smaller models including one with a historical topic.

The serious gaming project was conducted in cooperation with the international data farming community, and the results were published in Scythe – Proceedings and Bulletin of the International Data Farming Community, Issues 15-17. Three student groups were awarded in the Finnish Contest for Young Scientist, two in 2014 and one in 2015.

All sub-projects could be used for at least rapid scenario prototyping and the students learned causalities while creating models to a developing country game. As a result, we have continued to use serious game creation as a teaching method for talented young students.

SERIOUS GAMING PROJECT AS A TEACHING METHOD

The game building as teaching method can be used as a tool to apply inquire-based approaches to learning. This project combined all three genres of inquiry-based learning: project-based, problem-based and learning through design. (Barron and Darling-Hammond 2010) The pedagogical framework of this project is 21st century competences (Dumont and Instance 2010) and CSSC learning (de Corte 2010). The students construct their knowledge from a variety of sources when acquiring information on the chosen topic. They are self-regulated, to some extent, because they are able to choose their role in the process. The problem is real and all the factors are tied to reality and the project is highly collaborative.

To build a serious game and a simulation requires modelling and simulation skills. Our choice of theoretical framework and method of simulation in these games was data farming (Horne 1999, Horne 2010 and Horne et al. 2014). Data farming incorporates modelling and simulation, high performance computing, and design of experiments. It is applicable to examining questions with a large number of alternative scenarios. The process for modeling and simulation is relatively straight forward making it suitable for teaching purposes. (Figure1)

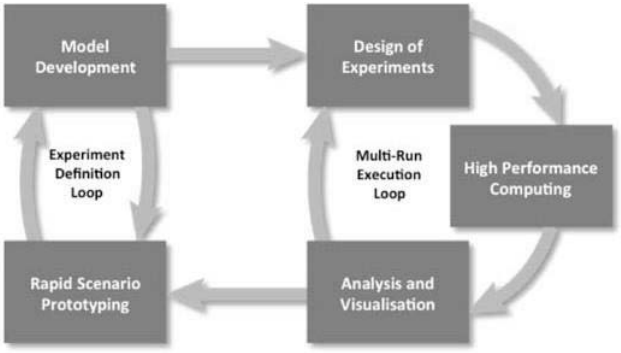


Figure 1. Data farming loop of loops from Horne et al. 2014.

In the Gameon 2013 an ongoing serious gaming project was presented, in which game creation was to be used in teaching. (Lappi 2013). Game development has been widely used in teaching software engineering and mathematics; see e.g. Cagiltay 2007, but we have not succeeded in finding teaching experiments of creating game programs as a method for teaching subjects that are not related to programming.

FIRST SET OF MODELS - THE HUMANITARIAN ASSISTANCE AND GLOBAL WARMING - SIMULATOR

The game creation project started in June 2013 during the What if –workshop 26 in Washington, when the idea of model was presented by Ted Meyer and Rachael Jonassen in form of a board game. We saw the game as an opportunity to develop a serious game project to teach students the phenomena and problematics around a collapsed state. From a brain storming session with a variety of possible sub-models (Meyer T. et al. 2013) the students chose the models they wanted to create with some guidance as to which ones were the most important.

The original plan was presented in Gameon 2013 and is visualized in figure 2. During the GAMEON 2013 the serious gaming project had just started. The connection

between a humanitarian simulation model and the subject matter included in the Finnish curriculum was established and reported. The first steps in modeling had been taken and the first encouraging results were available.

The learning objectives were modeling and the structures and functioning of a community in a humanitarian assistance situation in relation to natural phenomena such as the weather. Even though each group or pair of students focused on one aspect, the interaction between phenomena was illustrated in the parameter exchange between the subprojects as well as during all the presentations and briefings during the process. The learning results per se were never measured but the feed-back from the students gave indication of a much deeper insight and closer connection to the target country.

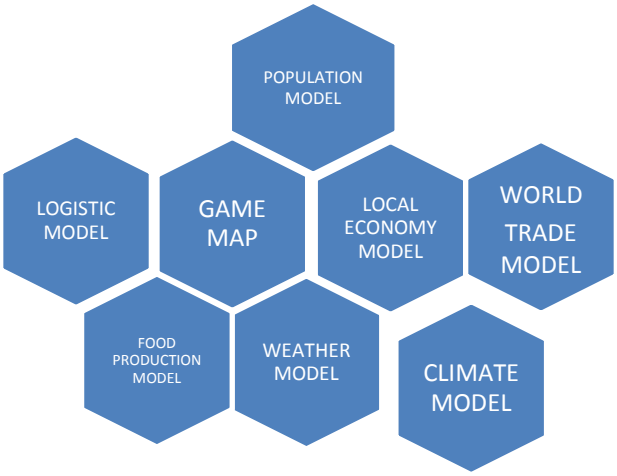


Figure 2: The model structure according to the original plan in October 2013 (Lappi & Lappi, Gameon 2013)

The results of the serious game creation projects were published in Scythe, Proceedings and Bulletin of the International Data Farming Community (Lappi et al. 2014, Lappi et al. 2015a and Lappi et al. 2015b). The papers concentrated mainly on the simulation game model and use of model, not on the serious game creation as teaching method, which is discussed in this paper.

The students taking part in this project were Finnish senior secondary school students who study in Päivölä School of Mathematics, a boarding school for mathematically talented students. It is a two year program, where senior secondary school studies are accelerated from three years to two years. The program is also enriched with more mathematics and computer science than is customary in a Finnish senior secondary education. An important part of the studies is involvement in a science project or several and introduction to academic writing in both Finnish and English. The Päivölä study program is described by (Hornyák 2011) and a short presentation of teaching computing science in Päivölä was published in ESTA 2009. (Lappi 2009).

In this gaming project several school subjects were included. The students were expected to learn content in fields previously unfamiliar to them. The content was, however, relevant to the senior secondary school curriculum. (Finnish National Board of Education 2003). The students deepened their knowledge in geophysics and geography, Finnish history, social studies, mathematical modeling and software engineering. They learned paper writing, English language and teamwork.

The serious gaming project was open to Päivölä alumni who contributed with their own interests but even with programming skills helping those less advanced in programming.

The groups decided to work on the models for population, logistics, local economy, world trade, food production, weather and climate. The game map – the game engine – was to be made by one team and later on the design of experiments was later made a subproject of its own.

During the project, the student teams developed their projects and communicated with the other teams. After research on the subject matter they built a network of the models. The structure evolved to interaction design as shown in figure 3 (from Lappi et al. 2014a).

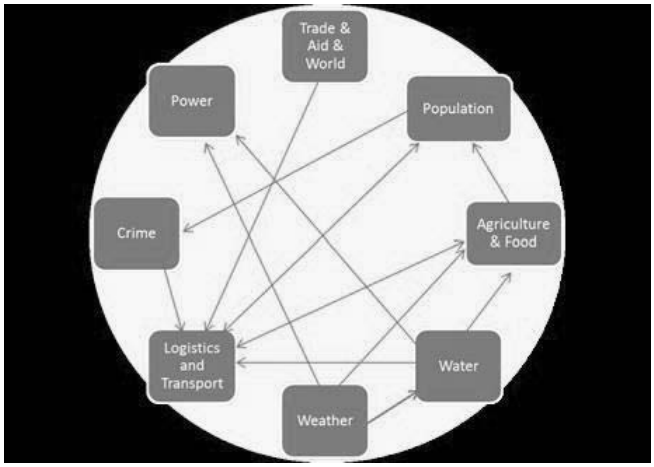


Figure 3. Game Simulation interactions as the project was finished. Picture from Scythe 15.

The students played the game as decision makers with a variety of parameters and analyze the results.

The project took half a year with an intensity that corresponds to approximately one month of full time research. All the steps of scientific work were taken: choosing a topic, learning about methodology, planning the research, programming the model, testing, presenting the work and the results for others and writing a paper in Finnish and in English.

The research seminar was held in presence of many former students of Paivola School some of whom are PhDs in the field and they gave constructive feedback.

The timeline of the first serious gaming project is presented in table 1.

Table 1: Timeline of the student game creation project

	Timeline	
	Students	Teachers
June 2013		Planning of the serious gaming process during What if workshop 26
October 2013	Choosing the topic, literature studies, project planning	Introduction to modeling and scientific writing
November 2013	Model development. Seminar on the preliminary models	Arranging the seminar, helping with model building and literature search
December 2013	Coding each sub model / game engine	Discussions, problem solving, moral support
January 2014	Merging the models to the game engine, design on the experiments, taking part in a what if workshop 27 (Helsinki, separate venue Päivölä)	Taking part in the workshop, helping with the problems in writing the papers within the dead-line.
February 2014	Dead-line of the paper for the Finnish Contest for Young Scientists.	
March 2014	Writing the English version for the Scythe	Assisting with the final versions and writing the project presentation for Scythe. Submitting the papers.

Even though the subject matter expertise gained by an individual student for his/her own project was from a relatively narrow field, the communication within the project and the seminars forced them to communicate and consider the other students' topics.

As a teaching method one advantage is that for evaluation purposes the models show whether the students have understood the subject matter. The model – however simplified – shows the causalities between the different factors and any misunderstandings are usually obvious and are removed from the final model. The models show that the students learned to understand concepts of cause and effect and get insight to the conditions of living for people in a different time or place. Part of the serious gaming project was the data farming workshop. There have been 29 data farming workshops since 1999 under different names. During the workshops data farmers gather together to study real world problems with data farming methods and also improve the methods. Usually work shop teams publish their work in Scythe, Proceedings and Bulletin of the International Data Farming Community. Päivölä students were welcome to join with their own teams.

As a result the following student papers were published in Scythe 15 January 2014 Workshop 27.

3A Imberg, Juhani & Toivanen, Pihla: Game engine of the HAGW-simulator.

3B Kauppila, Mirjam: Design of data farming experiment used in the HAGW-simulator.

3C Jantunen, Jonna & Niemi, Esa: Water model of the HAGW-simulator

3D Pouru, Asser & Nurro, Niko & Palmu, Allan: Logistics and transport of the HAGW-simulator.

3E Ahlskog Nicklas & Ijäs, Matias & Vilppula, Riku: Power production model of the HAGW-simulator

3F Hirvola, Tatu & Voutilainen, Kaisa: Crime in HAGWsimulator.

3G Hannikainen, Jaakko & Räty Roosa & Shakespeare Cliona: Agriculture model of the HAGW-simulator.

3H Ijas, Juuso & Pöyri, Saku: Trade and humanitarian assistance.

3I Tofferi, Julia: Weather model of the HAGWsimulator

3J Lunnikivi, Henri (Päivölä Alumni; Tampere University of Technology): Population model.

One example of interactions that are simplified by choices explained in the report of the crime model is presented in figure 4.

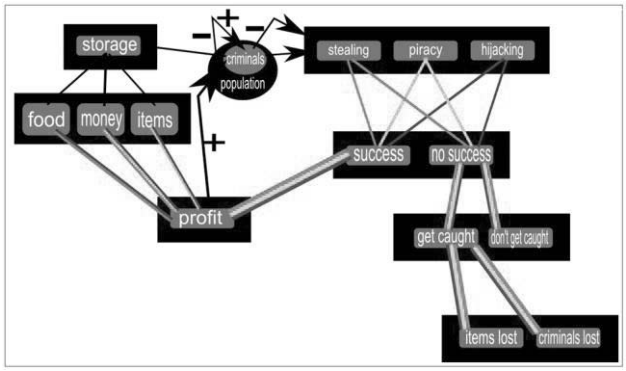


Figure 4: Crime model interactions (Hirvola and Voutilainen 2014)

The interaction between the different models lead to natural cooperation between the students especially in this first set even though the crime model and the global economy model were not connected to the game engine but functioned as stand-alone models.

As we live in boarding school with the students, we were able to observe the students and get feedback during this first project. This informal information supported our view that serious game creation can be a useful teaching method. Thus we decided to continue the project the following year.

THE SECOND SET OF MODELS

In school year 2014-2015 the yearly science project was not one large project but the students chose their individual topics. Serious gaming was one option, and three of the topics were linked to game creation.

Table 2: Timeline of the student game creation projects school year 2014-2015

	Timeline	
	Students	Teachers
October 2014	Start of work, literature studies, project planning, What if workshop in Jefferson US with a separate venue in Päivölä Finland. Both first and second year students participate.	Lessons of modelling and scientific writing in general. What if workshop in Jefferson with two teachers in the US and one in Finland
November 2014	Model structure development. Seminar	Arranging the Päivölä student seminar, supporting model building and literature search
December 2014	Coding sub models for new topics	Discussion and feedback.
January 2015	Reporting the projects, writing the paper for Finnish Contest for Young Scientists.	Supporting the writing process.
February 2015		Planning for the What if workshop in Finland
March 2015	The what if workshop. Main topic model development and learning simulation Student first year students	Present project in workshop, giving the research questions
April 2015	writing the texts in English for scythe	help with academic writing, comment the texts

The second set of models produced the following papers for Scythe 16 (Workshop 28 Jefferson, Maryland, USA & Päivölä Finland):

Team 4A: Lunnikivi, Vivian & Tuukkanen, Aaro & Häihälä, Eero & Virtanen, Maisa: Crime Simulator

Team 4B: Herring, Jan-Kristian & Qianyue Jin: Great Famine Simulator

Team 4C: Hokkanen, Joel & Mustonen, Vili & Alvinen, Markus & Kääriäinen, Kaisla: Recycling Simulator

THE THIRD SET OF MODELS

What if workshop 29 was held in Riihimäki, Finland and fourteen first year students in five different projects took part in the workshop in our own venue in Päivölä. Two of the projects were building on the first set of models and three were entirely new models about health care issues. The one week long workshop resulted in five papers. All of the participants were first year students.

Papers in Scythe 17 (Workshop 29 Finland)

Team 3A1Allred, Samuel& Karppila Hannes Malaria in Finland

3A2: Tenhunen, Anssi. Thomasson Janica & Moilanen, Henrik. Using Datafarming to Determine the Most Effective Combination of Malaria Prevention Methods

3B: Häihälä, Eero. Kääriäinen, Kaisla. Virtanen, Maisa & Klemola, Julius Effects of Infrastructure Optimization in the Trade of Sub-Saharan Africa

3C: Hämäläinen, Aleksanteri. Huuskonen, Petteri & Virtanen, Lauri. Modelling Placement of Health Care Facilities

3D: Valkama, Eero& Kumpulainen, Iiro Water Storage for Agriculture - Model

3E: Ikäheimonen, Siiri. Kotimäki, Julia & Salonen, Anniina. What MMR Vaccination Rate Is Needed to Stop an Out bursting Measles Epidemic?

DISCUSSION

The student projects achieved their goals, but the games were too difficult to play if the player was not familiar with the code. That means that the games are not yet usable in teaching other students about the content matter. Because it would be motivating to create games that are useful even after the paper is published, a development phase in our serious gaming projects would be to make the models more transparent and easier to use.

The serious game creation as teaching method is extremely time consuming. Only the added benefits of learning modelling and academic writing both in English and in the students' mother tongue – in this case Finnish - make it possible to use this method in teaching.

With just a few case studies no statistical conclusions can be drawn, but we consider the method successful in our multidisciplinary boarding school environment. However, there are challenges in using the method in other environments. We estimate the methodology to be too heavy for every day teaching in normal student groups. No single

subject has enough time in the Finnish Curriculum for this type of multidisciplinary method. Furthermore the student groups that took part in the projects consisted of talented students, who all had experience in computing science.

More scaffolding would have made the project faster and easier but possibly also less realistic considering the goals in learning how to do research. Not only the pretty parts but also the frustrating side of it. The group work means that the students can use their strengths – and avoid some parts of the process, if they so desire. This meant that not everyone was or was expected to be a very adept programmer.

One measure of success is the Finnish Contest for Young Scientist, where three groups were awarded, two in 2014 and one in 2015 (www.tukoke.fi) and the authors of this paper were given a teachers' award in 2015.

REFERENCES

Barron B. & Darling-Hammond L. 2010 Prospects and challenges for inquiry-based approaches to learning. In *Dumont, H. Istans, D. Benavides, F. (Eds) The Nature of Learning*. Centre of Educational Research. OECD. 199-226

Cagiltay N. E. 2007 "Teaching software engineering by means of computer-game development: Challenges and opportunities" *British Journal of Educational Technology* Volume 38, Issue 3, 405–415.

de Corte, E. 2010 Historical development in the understanding of learning. In *Dumont, H. Istans, D. Benavides, F. (Eds) The Nature of Learning*. Centre of Educational Research. OECD. 35-68

Dumont H. & Instance D. 2010 Analyzing and designing learning environments for the 21st century. In *Dumont, H. Istans, D. Benavides, F. (Eds) The Nature of Learning*. Centre of Educational Research. OECD. 19-34

Hirvola T. and Voutilainen K. 2014 Crime Model for the "Humanitarian Assistance and Global Warming" Simulator. In. *Horne G. & Mayer T (Eds.) Scythe, Proceedings and Bulletin of the International Data Farming Community, Issue 15 Workshop 27*

Horne, G. 1999 Maneuver Warfare Distillations: Essence Not Verisimilitude. In *Proceedings of the 1999 Winter Simulation Conference*

Horne, G. 2010. Summary of Data Farming. in Proceedings 4th International Sandis Workshop. Ed. Hämäläinen, J. Defence Forces Technical Research Centre Publications 23. 8-9

Horne G et. al. 2014 STO-TR-MSG-088 - Data Farming in Support of NATO. Final Report of Task Group MSG-088Nato >Science and Technology Organisation.

Hornyák B. 2011. "The mathematics programme of the Päivölä School" In International Horizons of Talent Support, I J.G. Gyori (Ed.) Hungary, 58-68

Lappi, M. 2009. "Cooperation Between a school with High Ability Students and a Technology Company." (Presentation) *In Proceedings of 7th International Conference on Education and Information Systems, Technologies and Applications.* Orlando. http://www.iiis.org/CDs2009/CD2009SCI/EISTA2009/PapersPdf/E720SE.pdf

Lappi M et al 2014 Developing a Data Farmable Humanitarian Assistance and Global Warming Simulator In. *Horne G. & Mayer T (Eds.) Scythe, Proceedings and Bulletin of the International Data Farming Community, Issue 15 Workshop 27* http://download.meyercraft.net/Scythe15-IWW27-V2o.pdf

Lappi, M. Lappi, E. & Hjorth, J (2015a): Applying Data Farming Process to Decentralized Project Work In *Horne G. & Mayer T (Eds.) Scythe, Proceedings and Bulletin of the International Data Farming Community, Issue 16, Workshop 28* http://download.meyercraft.net/Scythe16-IDFW28-V2.pdf

Lappi, M. & Lappi E. (2015b) Using Data Farming for Humanitarian Assistance and Global Warming. In *Horne G. & Mayer T (Eds.) Scythe, Proceedings and Bulletin of the International Data Farming Community, Issue 17, Workshop 29* http://download.meyercraft.net/Scythe17-IDFW29-Full-Good.pdf

Meyer, T. Lappi, M. Bouchard, A. Lee, C. Abdi, A. G. Ansardi, K. Estes, J. Faulkenberry, C. Jonassen, R. & Tolone, B. Climate Change and Humanitarian Assistance: challenges and complications. In *Horne G. & Mayer T (Eds.) Scythe, Proceedings and Bulletin of the International Data Farming Community, Issue 14, Workshop 26.* http://download.meyercraft.net/Scythe14-Workshop26-V1.pdf

Finnish National Board of Education. *National Core Curriculum for General Upper Secondary Schools*. Regulation 33/011/2003

WEB REFERENCES

www.tukoke.fi (web pages of the Finnish Contest for Young Scientists)

AUTHOR BIOGRAPHY

MERIKKI LAPPI was born in 1970. She studied first Geophysics in Helsinki University and got her B.A in Nordic languages 2004. She has been working in the Geophysics department of Helsinki University during the years 1990 and 1991, worked as part time simulation assistant in a fire protection engineering company Turvallisuusarviointi TA Oy from 1993 to 1997. She has been working as full time teacher in upper secondary school level first in Maunulan yhteiskoulu (junior and upper secondary school) 1996-1997 and in Päivölän kansanopiston matematiikkalinja (Päivölä school of mathematics) and Valkeakoski Upper secondary school since 1997. She has been the leader of mathematics program since 1999. Merikki Lappi was given Nokia educational award year 2002, Academy of Finland teachers' award of Science competition Viksu 2005, and Technology Industries of Finland Centennial Foundation award for successful mathematics teachers 2007 and Vaisala sponsored teachers' award of Finnish Contest for Young Scientists 2015.

XIMPEL IN EDUCATION - INSPIRING CREATIVITY THROUGH STORYTELLING AND GAMEPLAY

S.V. Bhikharie and A. Eliëns
Business Web & Media
Department of Computer Science
Faculty of Sciences, VU University
De Boelelaan 1081
1081 HV Amsterdam
The Netherlands
E-mail: svbhikha@few.vu.nl & eliens@cs.vu.nl

KEYWORDS
ximpel, education, interactive video

ABSTRACT

The XIMPEL framework has been successfully applied in multiple educational settings (university courses, workshops for high school students and educators) to enable users to create interactive media productions. In three steps (define a story graph, prepare media items and configure the XIMPEL playlist and application) users are guided in creating their productions. By providing thematical constraints users are challenged to find a personal approach towards the creation of their production, using storytelling and/or gameplay as a practical means to apply a theme into their work. When comparing XIMPEL to (commercial) game engines for usage within education, the biggest strengths of XIMPEL are the relative ease of use and fast prototyping capabilities, while maintaining rich interaction possibilities. Guided by ongoing technological advances, a HTML5 based version of XIMPEL is in active development, with the intention of making the framework even more accessible for both desktop and mobile platforms along with more users, targeting increasingly educational applications as developed for our courses in serious games.

INTRODUCTION

Since its inception in 2007, the XIMPEL framework for interactive media has been applied in multiple educational settings, namely in workshops and university courses. The workshops are given within the time span of 2 hours in which the participants create a short interactive video using the XIMPEL player. The participants for these workshops have varied from high school students and university students to educators and colleagues at conferences.

The university courses where XIMPEL is used are currently given at VU University Amsterdam and University of Twente. These courses are 4 or 8 weeks long and students have multiple deliverables they must present to their fellow students, including an interactive media production using the XIMPEL framework. For both workshops and university courses, we have defined guidelines in the form of three steps to help users create their XIMPEL productions:

1. Define a story graph
2. Prepare media items
3. Configure the XIMPEL playlist and application

structure The structure of this paper is as follows. We will first discuss the three steps that guide users in creating a XIMPEL application. We will then explain how thematical constraints can help shape a narrative structure, followed by a comparison of XIMPEL with game engines to point out its strenghts, closing off with future research and development.

DEFINING A STORY GRAPH

A story graph is a directed graph that specifies the navigational structure for a story. If a node in a graph has two or more branches, it is a choice within the story. In the context of an interactive production, these choices form the basic building blocks.

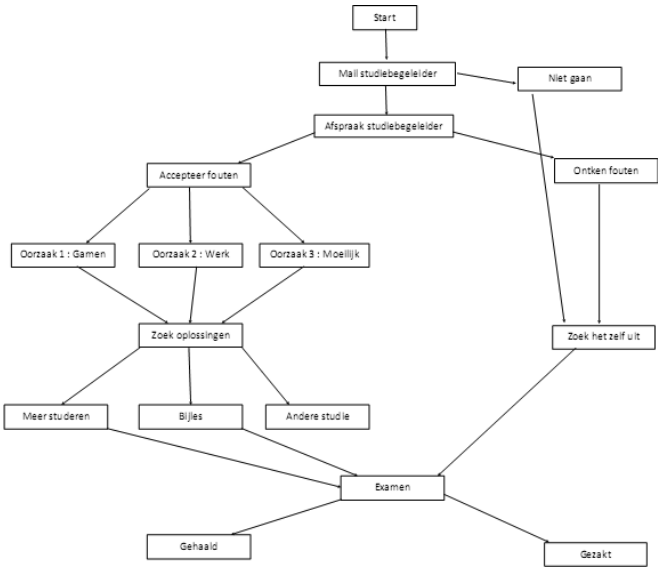


Figure 1: Story graph with multiple branching choices

By adding branches to a story graph, the number of nodes grows exponentially. To reduce the number of nodes, it can be useful to have multiple nodes leading to the same node.

Cycles within a graph can be useful if you want to enforce a certain choice: you will keep looping (within a select number of nodes) until you make a choice that leads you out of the cycle, forcing the player to explore its possibilities in order to advance.

End nodes (nodes without outgoing branches) can be used to give closure to a certain set of made choices. Due to its non-linear nature, a graph can contain more than one end node, which can increase the replay value and in turn increase the player's involvement.

PREPARING MEDIA ITEMS

When preparing media items for usage within a XIMPEL production, the Internet provides a vast amount of content, which can be found and/or edited with relative ease. At the other end of the spectrum, creating your own content is a viable option as well. Whether this is done with a high end video camera, video capturing software/hardware or the built-in camera of a smart phone, the advantage is that the user has more control over the produced output. Either way, the nodes defined in the story graph act as a rough checklist for the media items that will need to be produced.

The XIMPEL player has built-in support for three media types: picture, video and YouTube. The picture type is a static image, with support for jpeg and (transparent) png files. The video type supports h264 mp4 (recommended) and flv (legacy) videos. It should be noted that the picture and video media types can be used for both offline and online XIMPEL applications. As a rule of thumb for online usage, it is recommended to keep the file sizes for these media types as small as possible, since this content will need to be downloaded progressively through your web browser.

When using larger video files, the YouTube type is a suitable alternative for using videos within XIMPEL. These videos are remotely hosted on the YouTube servers and directly streamed into the XIMPEL player.

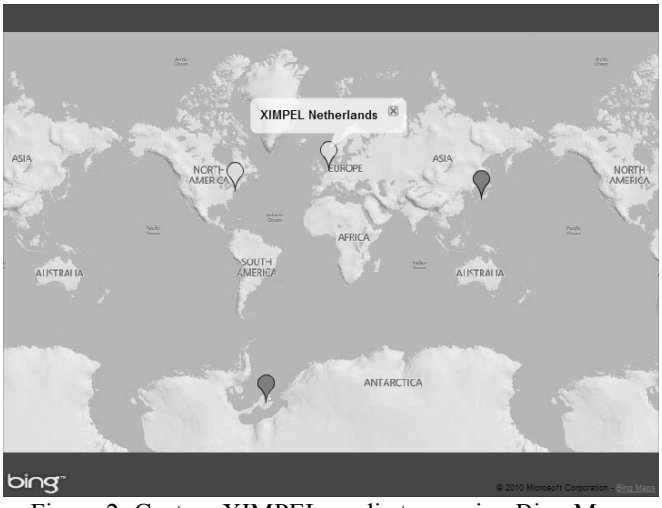


Figure 2: Custom XIMPEL media type using Bing Maps

Apart from the built-in media types, it is also possible to add custom media types, like audio, geographical maps or even minigames. This requires some additional work in the form of programming an extension for the XIMPEL player, but there are no further restrictions on how a media type should look or behave, greatly increasing the expressivity and possible interactions.

CONFIGURING THE PLAYLIST AND APPLICATION

To get started with the XIMPEL playlist, a package containing the XIMPEL application must first be downloaded from the XIMPEL download section[1]. This package is available as a basic and advanced version. With the basic package, the user can immediately get started with creating a playlist with an already compiled application. The advanced package contains a library and source code for the XIMPEL application and allows for customizations and extensions.

```
<ximpel>
    <subject id="0">
        <description>
            Make a choice
        </description>
        <media>
            <picture file="1.png">
                <canvas>
                    <overlay x="190" y="140"
                     width="120" height="420"
                     leadsto="1a"/>
                    <overlay x="970" y="140"
                     width="120" height="420"
                     leadsto="1b"/>
                </canvas>
            </picture>
        </media>
    </subject>
    <subject id="1a">
        <description>XIMPEL</description>
        <media>
            <youtube id="aUAjicsqBjE"/>
        </media>
    </subject>
    <subject id="1b">
        <description>Fast</description>
        <media>
            <video file="fast.mp4"/>
        </media>
    </subject>
</ximpel>
```

Figure 3: XIMPEL playlist where the first subject is linked to the other subjects using overlays

The playlist is specified as an XML file that is loaded by the XIMPEL player. Using the story graph as blueprint, the basic structure can be created. In figure 3 a basic playlist with three subjects is displayed.

1http://ximpel.net/downloads/

Subjects are the main building blocks within a playlist and contain a list of one or more media items. They can be linked with overlays, which are visual cues that are placed on top of a media type. When you click on an overlay, the associated subject is loaded.

Figure 4: Rendered output of the first subject of the playlist with two rectangular semi-transparent overlays

Overlays can have different shapes (rectangular or elliptical) and can be filled with either a (transparent) color or an image as background, in combination with a snippet of text. The XIMPEL documentation section[2] provides a more exhaustive list of options for building overlays and playlists.

Apart from the playlist, the application itself can also be configured using a configuration XML file. Some of the options include changing the title, authors and splash image for the application, allowing users to give a finishing touch to their XIMPEL application.

THEMATICAL CONSTRAINTS

Even when presented with the functional means to create a XIMPEL application, users must first be challenged and subsequently inspired before they are able to tell their (interactive) story. By providing thematical constraints, the users are forced to think about what story they want to tell.

Some examples of themes we have used in the past are a guided tour through the university, a tour of Amsterdam, environmental issues, ethical frameworks, mathematical games and dealing with feedback. Although we do not strictly enforce the themes, most users tend to choose the themes as a starting point and give these themes their own spin, often resulting in surprising, funny, endearing and even serious results.

By systematically applying (thematical) constraints in a narrative and providing a proper feedback loop, the interaction can be greatly improved by creating an engaging, game-like experience. In the case of an ethical framework as constraint, as described in Bhikharie & Eliens (2013), the player is confronted with and recognizes moral dilemmas through the narrative and must actively engage them by making choices and seeing the consequences of these choices, which is discussed in Sicart (2013).

2http://www.ximpel.net/documentation

COMPARISON WITH GAME ENGINES

XIMPEL goes hand in hand with the term *poor man's immersion*, meaning that it makes use of immersive realism of videos and images as a poor man's substitute for interactive 3D immersion, as can be experienced in for example games and virtual reality applications. We coined this term in reference to the climate game described in Eliens et al. (2007).

The trade-off being made is directly related to the cost of development. A 3D game engine like Unity or Valve's Source requires a relatively big investment in time and knowledge to get started, let alone to create a world populated with 3D models and program the necessary game logic and behaviours.

From our own experiences in developing a masterclass game development for high school students using the Source engine, as described in Eliens & Bhikharie (2006), we have seen that to make a 3D engine accessible for users with no prior experience, a decent amount of preparation is required.

In our case, we made a modified non-violent version of the Half-Life 2 multiplayer mode, together with a partial virtual recreation of the VU University as an introductory playground to get acquainted with the 3D game world and engine. We furthermore prepared a template level with the level editor of the Source SDK (Hammer), along with instructions how to get started with simple level editing and creating custom textures for surfaces.

Although the students were able to produce their own levels with relative ease and the produced work was certainly visually creative, it was limited in its interaction, only allowing you to walk around.

With XIMPEL we wanted to create a media framework that is easy to use and allows to quickly create a working prototype, while still allowing rich interaction.

By using the XML standard as foundation for the playlist, we specified a human-readable format that can be easily understood and authored, without the need for complex tools. Furthermore, since the playlist is the minimal amount of input needed to create a XIMPEL application, getting an application up and running can be accomplished in a matter of minutes.

Using a story graph as a blueprint for interaction, the application can reach its full potential by transforming the graph into a full-flegded XIMPEL playlist. By using custom media types and applying thematical constraints, it is possible to further enrich the interactions.

FUTURE RESEARCH AND DEVELOPMENT

When we started with the development of the XIMPEL framework in 2007, we chose for the Adobe Flex SDK as our main implementation platform, allowing us to programmatically create Flash SWF files with built-in

support for the (then popular) FLV and MP4 video formats with no extra cost and enabling usage of XIMPEL applications both offline and online, giving us a user base of everyone who uses a desktop computer with a Flash Player plugin enabled web browser. Over the years, powerful mobile devices like smart phones and tablets have become more prominent. Since these platforms lack support for the Flash Player, we felt the need to look at potential alternative technologies that allows us to deploy the XIMPEL framework onto these platforms as well, in addition to the current Adobe Flex implementation.

The most effective technology available that caters to both desktop and mobile devices is HTML5 (in combination with JavaScript). Over the years, HTML5 has matured in both its specifications and browser implementations. Especially mobile devices have adopted it as one of their default technologies for their browsers, independent of the software stack (Android, iOS, Windows etc.) that it might run. By developing a HTML5 based version of XIMPEL, we have the potential to reach everyone using a modern and up-to-date web browser, independent of their mobile or desktop platform.

Furthermore, a HTML5 based XIMPEL offers possibilities to directly connect with JavaScript based applications and APIs, allowing for new interaction possibilities. These developments also lead us to target increasingly educational applications as developed for our courses in serious games.

REFERENCES

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

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

Eliëns A., Huurdeman H., van de Watering M., Bhikharie S.V. (2008), *XIMPEL Interactive Video -- between narrative(s) and game play*, In Proc. GAME-ON 08, Valencia, Spain

Eliens A (2012), *Serious games in a social context*, In Proc. GAMEON'2012, Malaga, Spain

Bhikharie S.V. & Eliens A. (2013), *XIMPEL for Ethical Frameworks*, In Proc. GAMEON'2013, Brussels, Belgium

Sicart M. (2013), *Beyond Choices: The Design of Ethical Gameplay*, MIT Press, Cambridge, Massachusetts

SOME REFLECTIONS ON BOLOGNESE FOOD: A DIGITAL PERSPECTIVE, WITH A LOT OF FUN

Marco Roccetti Silvia Colombini Marco Zanichelli

Department of Computer Science and Engineering
Alma Mater Studiorum - University of Bologna
Mura A. Zamboni,7 – 40127 Bologna, Italy
E-mail: {marco.roccetti}@unibo.it

KEYWORDS

Bolognese Food, Well-being, Gastronomic Heritage, Interaction design, Gaming, Human Computer Interaction

ABSTRACT

In this short paper we describe a study of interaction design developed to celebrate the excellence of the Bolognese gastronomic heritage that has culminated with the release of the "Manifattura del Gusto" (The Manufacturing of Taste). The "Manifattura del Gusto" is comprised of three different multimedia interactive exhbits that narrate the history of three different world-famous Bolognese food delicacies, respectively: *tortellino*, *tagliatella* and *mortadella*. These three exhibits are permanently installed at the *Bologna City Museum*, based in the Pepoli Palace, in Bologna, Italy. From a technical viewpoint, gaming and interaction techniques, textile sensors, holograms, motion sensing devices, Arduino controllers and even stretching cloth fabrics have been all put to good use to offer to visitors the possibilities of understanding that tacit knowledge which is always hidden behind the process of preparation and production of a delicious food.

INTRODUCTION, MOTIVATIONS AND CONCEPT

From the kitchens of the city of Bologna in Italy, the masterpieces of the Bolognese gastronomy (*tortellino*, *tagliatella* and *mortadella*) have arrived onto the tables of every home all over the western world, often influencing the culinary habits of places and people living thousands of miles far from Bologna. Tortellino, tagliatella and mortadella, indeed, all belong to a centuries-old Bolognese gastronomic heritage, that directly links to its delicious land products and to the specific ability of the Bolognese people to prepare a delicacy to eat. In particular, the Bolognese tortellino is a small, belly-button-shaped dumpling, traditionally prepared in chicken broth. The tagliatella (or better tajadela, in our Bolognese slang) is a long and flat ribbon, made with egg pasta (similar in shape to the *fettuccina romana*), and served with a classic Bolognese meat sauce. Finally, the mortadella is a typical Bolognese sausage, made of finely hashed pork meat and flavoured with spices, including black pepper, for example. This is not the full story as the techniques of preparation and conservation of these delicious foods have evolved and grown into an important food production industry, making the region where Bologna is placed one of the most relevant industrial district for food preparation and conservation in the world. To summarize the current sentiment on how this kind of food is perceived in Bologna, it is enough to remind that it is essentially a mix of human ingenuity, technology innovation and business skills; with a bit of emotions, too - everyone should always have in mind, in fact, that tortellino, tagliatella and mortadella are both a delicacy to consume but also a cultural pastime worth preserving (Rosner, Roccetti and Marfia, 2014).

Along this line of sense, we have been struggling from years with the problem of how to communicate using computer technologies the great amount of tacit knowledge which is behind the typical dishes of our city and how to demonstrate that all these delicacies inform of the uniqueness of our territory, as well as of the soul and spirit of generations of family members in Bologna, who have preserved and passed on these pieces of tradition (Osterlund *et al.*, 2015; Roccetti, Marfia and Zanichelli, 2010). With this in view, we have conceived, developed and then installed at the *Bologna City Museum*, based in the ancient Pepoli Palace, in Bologna, Italy, three different interactive multimedia exhibits that narrate a story made of memory, hospitality, gratification, love and creativity, as each of those emotional ingredients is comprised in the recipes of those dishes and is able to transform each meal into a kind of "pleasant ritual". We do not have to forget, in fact, that like the Proust's madeleine, a taste of certain dishes has the power to bring us back to the time we lost, where each dish holds and passes on traditions that tell the world the history and the process of civilization experienced by many individuals, or even by an entire people (Salomoni *et al*., 2015). With this in mind, a common goal of our artefacts has been that to try to recreate suggestions that could transform a museal experience into the digital opportunity of feeling as if we were all together; all guests eating while sitting at the same table, in peace and joy.

On this view, the three interactive exhibits have been arranged along a path where, in turn, the concepts of: i) the tradition behind a given dish preparation, ii) the joy of eating

together, and iii) the life cycle of the food production process were illustrated through the use of three different interactive machines. The remainder of this paper proceeds as follows. The next Section describes and illustrates the three different exhbits we designed and developed to honour the Bolognese food, while the Conclusion Section terminates this short paper.

INTERACTIVE MULTIMEDIA EXHIBITS

While the correct degree of pervasiveness of digital technologies in cultural practices is still a matter of study and controversy, it is somewhat unconfutable now that the use of computer technologies applied to cultural venues should be in some sense ludic, able to provide fun and also a mild form of entertainment to visitors. We have already shared in a previous paper our vision of technology in these terms, as an additional intermediary; as the lens, indeed, through which people can enjoy not only a memory preservation experience, but also be subjected to a pleasant condition where great stories become more comprehensible and understood, or even reveal unexpected information that only the eyes of a digital system are able to make explicit (Roccetti *et al*., 2014; Roccetti *et al*., 2013). Along this line and within the context of a more general project termed "*La Manifattura del Gusto*" (The Manufacturing of taste), we developed three different interactive multimedia exhibits, termed: *Tortellino Memories*, *Symposium Tajadela* and *Mortadella Machine* with the explicit aim of addressing the prominent issues of tradition (Tortellino Memories), conviviality (Symposium Tajadela) and food production and preparation ability (Mortadella Machine). We illustrate and discuss each of them below, in isolation. Before this, it is important to mention that each multimedia exhibit we developed is equipped with a user interface based on a table cloth fabric. This particular choice in the design of the interaction was adopted based on the consideration that a growing number of people is rediscovering, or newly embracing, the joy of having a family-style meal, where the table is well laid with all its traditional decorations.

Tortellino Memories

The narrative path we developed begins with the stories of the three selected foods: tortellino, tagliatella and mortadella, respectively. To this aim, we designed and built a large and transparent totem incorporating a monitor (Figure 1), where the essence of the stories of each dish is evoked, using the computerized graphical style of the *title sequences* inspired to the world-famous American graphic designer, Saul Bass (Figure 2; Bass, 1959). Under the monitor, three panels clothed with fabrics resembling a cloth table (and enriched with specific textile sensors) can be touched on by the visitor to activate the storytelling (Figure 3).

Symposium Tajdela

This exhbit allows visitors to enjoy the experience of sitting at a table for having a meal, while joining a special guest. This is made with a table, incorporating a monitor where our three different dishes are served (in a digital form). While a dish is served, a special guest comes on stage under the form of a hologram appearing in front of the diner. Again the three different holograms can be activated by touching on special buttons, clothed with fabrics resembling a cloth table (and enriched with specific textile sensors). This exhibit is shown in Figure 4, while an example of interaction is provided in Figure 5.

Figure 1. **Tortellino Memories**: the exhibit at the museum

Figure 2. Example of the adopted graphical style

Figure 2. **Tortellino Memories**: interaction

Figure 4. **Symposium Tajadela**: the exhibit at the museum

Mortadella Machine

In this case, the exhibit we designed aims at presenting tangible histories of the production of the Bolognese food, under the form of a *memory game*. In particular, we built a huge and square box representing a *hypothetical* food production machinery (Figure 6). The special and magical feature of this machinery is that it is equipped with a front panel, made again of a cloth fabric, whose characteristic is that it can be stretched and almost penetrated (Figure 7). Behind this panel (and inside the machinery), we have set a Kinect device, a projector, a tailor-made vending machine and an Arduino controller. The flow of our memory game is as follows. Upon activation of the Mortadella machine, the six different and subsequent steps of the production process of the mortadella are shown, as projected over six different spots of the stretching panel. Then the machinery turns off and simply asks the visitor to touch (or better penetrate, like in Figure 7) the panel, repeating the same sequence of steps through which the mortadella is prepared. Obviously, with each new game match, a player is presented with a different positioning over the panel of the six different steps for the preparation of the mortadella and with a limited amount of time to complete the game (e.g., 30 seconds). Only if the visitor wins (i.e., s/he perfectly retraces the steps, identifying each correct position over the panel, within the time deadline), then the machinery dispenses a museum gadget to the lucky winner. Currently, the dispensed gadged is a museum pocketbook shaped like a mortadella slice.

Figure 5. **Symposium Tajadela**: interaction

Figure 6. **Mortadella Machine**: the exhibit at the museum

Figure 7. **Mortadella machine**: interaction with the stretching fabric

CONCLUSIONS

We here described three interactive multimedia exhibits that provide honour to the food prepared in the city of Bologna, allowing people to have also fun while interacting. It is our persuasion that projects like ours may highlight new opportunities for sustaining that special and usually tacit knowledge which is behind the preparation of a dish and whose tradition we might want to preserve. Hopefully, this kind of initiatives, supported by a delicate form of digital entertainment technologies, may reveal more about the history and collective memory of a gastronomic cultural practice than would a classic cooking lesson alone.

REFERENCES

Bass, S. 1959. "Creativity in visual communication". *Creativity, New York: Hastings House*, 121-142.

Osterlund, C.; Bjorn, P.; Dourish, P.; Harper, R. and Rosner, D. K. 2015. "Sociomateriality and design ". In *Proc. the ACM Conference on Computer Supported Cooperative Work, CSCW*.

Ferretti, S. and Roccetti, M. 2005. "Fast delivery of game events with an optimistic synchronization mechanism in massive multiplayer online games". In Proc. *ACM International Conference Proceeding Series 265*, 405-412

Roccetti, M.; Marfia, G.; Roversi Monaco, F.; Varni, A. and Zanichelli, M. 2014. "Telling the story: An interactive multimedia exhibit narrating the 900 years of the Alma Mater". In *Proceedings of 20th European Concurrent Engineering Conference 2014, ECEC 2014 - 10th Future Business Technology Conference, FUBUTEC 2014*.

Roccetti, M.; Marfia, G. and Bertuccioli, C. 2014. "Day and night at the museum: Intangible computer interfaces for public exhibitions. *Multimedia Tools and Applications*, 63(3), 1131-1157.

Roccetti, M.; Marfia, G.; Varni, A. and Zanichelli, M. 2013. "How to Outreach the External World from a Museum: The Case of the Marsili's Spirit App". In *Proc. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*.

Roccetti, M.; Marfia, G. and Zanichelli, M. 2010. "The art and craft of making the Tortellino: playing with a digital gesture recognizer for preparing pasta culinary recipes". *Computers in Entertainment*, 8(4), ACM.

Rosner, D. K.; Roccetti, M. and Marfia, G. 2014. "The Digitization of Cultural Practices". *Communications of the ACM*, 57(6), 82-87, ACM.

Salomoni, P.; Prandi, C.; Roccetti, M.; Nisi, V. and Nunes, J. N.. 2015. "Crowdsourcing Urban Accessibility: Some Preliminary Experiences with Results". In *Proc. 2015 CHItaly, 11th Edition of the Biannual Conference of the Italian SIGCHI Chapter, Rome, ACM SIGCHI.*

ACKNOWLEDGEMENTS

We are indebted towards those who allowed us to develop and install our exhibits. We wish to thank: prof. F. A. Roversi Monaco for hosting our artefacts at his museum, prof. A. Varni for being our historical consultant, and finally Loop Multimedia for its essential support.

BIOGRAPHIES

M. ROCCETTI is a Professor of Computer Science at the University of Bologna. His most recent research has focused on the goal of combining modern computer techniques and technologies with cultural practices.

S. COLOMBINI since many decades has gained an international experience as a senior copywriter, serving for many international advertising agencies, including TBWA, LINTAS and BBDO.

M. ZANICHELLI is an interaction designer and a senior art director with an international and broad professional experience. He has served for many international advertising agencies, including Leo Burnett.

VIRTUAL GAMING TRAINING ENVIRONMENTS

VIRTUAL REALITY SITUATIONAL LANGUAGE TRAINER FOR SECOND LANGUAGE: DESIGN & EVALUATION

Timo Korkalainen, Juho Pääkylä, Tapani N. Liukkonen, Lauri Järvenpää and Tuomas Mäkilä
Technology Research Center
University of Turku

20014 Turun yliopisto
Finland
Email: {ttkork, jujapa, taneli, lauri.jarvenpaa, tusuma}@utu.fi

Yrjö Lappalainen

University of Tampere
School of Information Sciences
Tampere Research Center for Information and Media
33014 Tampereen yliopisto
Finland
Email: yrjo.lappalainen@uta.fi

Heli Kamppari

Brahea Centre
University of Turku

20014 Turun yliopisto
Finland
Email: helkam@utu.fi

KEYWORDS
L2, education, VR, gamification, UX, learning environment

ABSTRACT

Use of virtual learning environments has been studied for years, but no common guidelines for evaluating them for education exist. The Developing Virtual Learning for Finnish project proposed a set of such guidelines in context of teaching a second language to immigrants. To aid the definition work and test the guidelines, a demonstrative application was created with focus on introducing the students to everyday situations and to the use of spoken language. The main objective of the game design was to create a friendly and comfortable environment, where the users would feel safe to experiment with the new language. To evaluate both the game and the initial guidelines, user experience research was conducted on actual immigrants learning Finnish as a second language. According to the questionnaire results, the game fulfilled the primary objectives and the evaluation criteria provided an insight into the usefulness of the game for learning.

INTRODUCTION

The Developing Virtual Learning for Finnish project set to develop guidelines for evaluating virtual learning experiences. In order to define and test guidelines, a demonstrative application was developed. The aim of the application was to study how to simulate simple real world situations in a virtual setting in order to let the students learning Finnish as second language on the levels A2 to B1 of the Common European Framework of Reference for Languages (CEFR) to improve their ability to cope with the lingual challenges of daily life. Afterwards user testing was conducted to evaluate what the members of the target audience thought about such an approach to learning.

The background chapter summarizes the educational context and goals of the application and discloses contemporary approaches to evaluating gamified learning experiences. The chapter on game implementation describes how the application was developed and why certain design approaches were selected. Chapters Research Method and User Experience Testing Results outline the testing procedures and the results, while the final chapter summarizes the key points found in the study and based on them presents views on the future development of virtual reality based learning of a second language.

BACKGROUND

The use of virtual worlds as part of teaching has been both researched and practiced for several years. One of the common approaches is to use open virtual worlds, such as Second Life, to encourage cooperation and communication between learners and teachers from different parts of the real world. (Lappalainen, 2015, p. 42) However, virtual worlds offer another approach: simulation of real world situations in a safe and private environment. The students can comfortably experiment with different approaches to the same situation without the fear of being judged for mistakes. This encourages the learning of new language skills and promotes the culture around the language itself. (Kiili, 2004)

Second Language Pedagogy

In this study the focus was on immigrant students learning Finnish as second language. More advanced learners were excluded due to previous studies showing that they would not benefit as much. (Lehtonen et al., 2015, p. 26). For the students in the focus group it is vital to lower the barrier and get initial encouragement for using the language more in their everyday lives, as it is the only way to get comfortable with it in spoken situations and enhance the rate and success of learning.

Traditional methods of teaching a second language are mostly centered on the official written form. This leaves the learners in the blind with casual everyday encounters, where they either do not understand the spoken language used by the natives or use expressions that generate other than desired reaction. Neither tend

the traditional classroom methods to provide a comfortable approach to learn the customs of behavior in the natural face to face encounters. Such moments can cause stress and embarrassment both before and after the situation and thus greatly affect the student's motivation and commitment towards the learning process. (Jauregi, 2012; Canto, 2013)

Virtual reality allows simulating real world situations with no other real human being involved than the players themselves. As all the virtual characters' reactions are controlled programmatically, they express only the emotions desired and never get frustrated no matter how many wrong answers the learner gives. Virtual conversations also make it possible to create interactive characters that comply with the requirements of pedagogical objectives.

Game Design

Designing and creating game is a multidisciplinary task, especially when games are combined with pedagogical goals. It is not enough to just entertain the users, but to also educate them in a way that can be measured according to the standards employed in the traditional ways of teaching. For some students, the interactive nature of games that encourages experimenting with causation, can even be the most effective way of learning. (Gee, 2008; Viinikkala et al., 2014)

While the game still has to entertain enough to keep the players immersed in the story, the students have to recognize the progress they are making with the actual subject of learning. The games also should not present different story paths just to keep the students playing longer, but also to concretize how the different ways of behavior are responded in real life. Thus all aspects of the game should be pre-evaluated on both how they affect the quality of the game experience and how they can advance the learning process. (Ondrejka, 2008)

Vital to educational games is the attitude the players form towards the experience: if they play reluctantly and just because it is part of the classroom work, the learning results suffer. In contrast, when the learning is almost a side effect to the gaming and the players enjoy the experience, they both learn more and the memory trace becomes more profound. (Luckin & Fraser, 2011) Thus the game design needs to be balanced between providing enough pedagogical context and an experience that is memorable by itself.

In order to immerse the player into the virtual world, no additional visual avatar was added to represent the player. Instead the game is viewed from the direct first person perspective of the player. Thus the students are encouraged to feel like being in the situation themselves, and not just controlling an artificial person.

User Experience Research with Educational Games

Many of the traditional software user experience testing methods are also suited for evaluating games. User testing methods can be divided into two categories: 1) attitudinal, for measuring the subjects' attitude towards the application and 2) behavioral, for measuring how the application is used. While many techniques exist for behavioral usability testing, attitudinal testing is mostly conducted using questionnaire methods. (Pagulayan et al., 2007, p. 21 - 27)

In the context of educational games with a specific target group, it is natural to focus the user testing on the same audience. Thus the need for large random sample testing is minimal and results are achieved with a smaller number of selected test subjects. In educational games the emphasis in user testing is also more on whether the players learned the required information, not necessarily on fine tuning the gaming experience. It is also important to measure the attitudinal aspects as they too affect the learning results.

GAME IMPLEMENTATION

The primary objective of the game design was to create an environment, where the players would feel safe and comfortable, allowing them to focus on the content and forget any of the fears the real world situations impose. The game was chosen to use the classical adventure game mechanics with focus on the use of the Finnish language in interactions with the virtual characters. To make the story believable and yet interesting, much emphasis was placed on the authenticity and natural feel of the dialogue. This was also done to ensure that everything the players would learn was accurate and usable in the verbal encounters in the real world.

Story

The story of the game is set inside a restaurant car of a train. The story was written by pedagogy professionals with the help of the game designers and developers. The writers struggled at first with the interactive nature of video games, but after every iteration of the prototype their understanding on the medium grew. An important role in improving the scriptwriting process and allowing easy iteration, was the use of the articy:draft 2 (Nevigo, 2015), an application designed for writing interactive game scripts. It allowed the writers to visualize the flow of the script in a more comprehensible format, as seen in Figure 1.

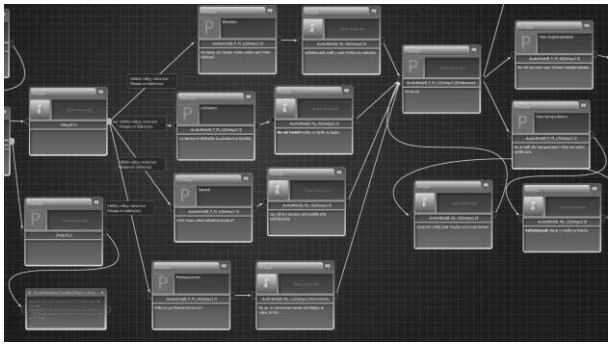


Figure 1: Game Script Visualization (3D Suomi, 2015)

It also supports testing out the discussions in the tool itself and thus eased the process of learning to write interactive game scripts.

A challenge in the script writing was to find the balance between making the dialogue and the player's options both understandable and challenging enough at the same time. Both extremes, too easy or complex dialogue, would cause learner to lose interest. If the players would not understand what to do next, they would again lose interest.

Art Design

In order to help create the wanted safe and comfortable environment, the art design of the game was simple and cartoon like. For example, the restaurant car presented in Figure 2, was closely modelled after the design of a real restaurant car used in Finland.

Figure 2: Restaurant car and conductor in 3D Suomi (3D Suomi, 2015)

Model of the car was simplified, but yet giving clear cues of aspects relating closely to real life elements. All the materials were using bright and friendly colors. This was done to clearly distance the surroundings from the real world. The cartoon like style was continued in the characters which were caricaturized versions of people with different backgrounds. The simple style also helped to keep the players' focus in the important parts of the scene that related to the story.

By design the game environment provides a limited number of objects and persons to interact with and the discussions have a limited number of paths to follow. Additionally the cartoon like style was used to convey neutral visual message to the players with varying cultural backgrounds. The visual style also helped in allocating resources effectively and in improving the cost-effectiveness.

The audio design for the application continued the same simplified style: only quiet yet constant rattling of the train wheels and rain pouring down outside were used to create an atmosphere that would still be enough to distance the player from their actual environment. The studio recorded dialogue was made loud enough to be clearly heard over the ambient sounds. The different dialects gave the characters a personal touch, as none of them would use quite the same type of language.

Technology

Modern game development tools allow fast prototyping and creation of complex virtual reality experiences with less focus on the technology, but on the content itself. The demonstrative game for the project was developed using the cross-platform game engine Unity 4 (Unity Technologies, 2015). The engine was chosen due to it providing necessary features for the project and the developers having prior experience in using Unity in mixed reality applications in the context of learning environments.

An important tool for the development was the plug-in Dialogue System (Pixel Crushers, 2015) for Unity 4. This plug-in has an option to directly import dialogue from articy:draft (Nevigo, 2015), the script writing tool used in the project. This made it easy for the developers to import any changes in the dialogue into the game without additional work, allowing fast iteration.

While the combination of tools used are not the only option, a readily available development platform is a requirement for cost-effective development of educational games. Many modern game engines, such as the Unreal Engine 4 (Epic Games, 2015), are available for productions where the focus must be on the content and not in the technology.

RESEARCH METHOD

As the goal of the project was to define guidelines for evaluating virtual learning experiences, it was seen necessary to form and test some of the proposed guidelines by practically designing, developing and finally conducting user experience testing on a real working demonstrative product. After development and initial testing, a large scale user experience testing was conducted in several of the partner institutions.

Study Setup

User experience testing for the application was conducted in a classroom setting with multiple subjects testing the game on different desktop computers at the same time. Few subjects also tested in pairs in order to find out what kind of a difference that would make to the experience. A teacher was available in the classroom at all times for help, in case someone had technical difficulties. The subjects began the test with the game already running in its initial state and left it running once completed. The subjects were reminded that the test was anonymous and that the testing was about the game and how well it suited the educational purposes, not about their personal performance or knowledge.

Questionnaires

After completing the game the subjects were presented with a questionnaire of 60 questions formed roughly on the basis of the Game Experience Questionnaire (GEQ) (IJsselsteijn, 2007) and System Usability Scale questionnaire (SUS) (Brooke, 1996). The main questionnaire was in Finnish, but additionally the questions and options for answering were available in several languages to make sure all subjects understood everything. Additionally some of the subjects were interviewed after the questionnaire.

The answer options were based on agreeing or disagreeing on a scale from 1 to 7, where 1 was total disagreement and 7 total agreement.

Participants

Altogether 147 completed questionnaires were collected and analyzed. Of all the respondents 58.3% were women and 41.7% men, the majority (53.1%) being from 25 to 34 years old of age. Most (54.4%) of the subjects had some form of a degree and already used Finnish (74.0%) on a daily basis on some level, with English being the second most used language (49.3%). The most common reasons for immigration were family (57.1%) and education (16.3%) related ones.

Almost all of all the respondents (80 % of 112 answers) have been assessed by their teachers on levels either A2.2 or B1.1, which are the median scores in the 9 step scale of the Finnish version (National Board of Education) of the 6 step scale of CEFR - and also form the main focus group of our user experience testing.

Most of the subjects used information technology on a daily basis (6.06), but on the contrary most had not much experience in playing video games (3.46).

USER EXPERIENCE TESTING RESULTS

According to the user experience testing the demonstrative application performed well in general and was especially thanked by the subjects for presenting the everyday situations in an approachable manner. The possibility of comparing the spoken dialogue to the written subtitles was also well received and was seen as one of the key points of the experience. In the interviews one of the subjects especially noted that "my listening skills in Finnish need a lot of work, but my reading comprehension is better, so having the text of the spoken conversation shown on the screen while I could listen to the conversation was very helpful to me".

The goal of comfortable and safe experience seems to have been achieved as the majority of the subjects (average agreement 5.14, on scale from 1 to 7) reported feeling happy after playing the game. Many also felt the experience had given them courage to act in real life situations (5.10) and especially felt it had been helpful in practicing to understand spoken language (6.17).

The visual style of the game was one of the most dividing topics and while the answers were diverse, most would have opted for a more realistic approach for the characters (4.42). While one of the interviewed subjects called the visuals "primitive", other told that "they looked funny and made me smile". Still, most had felt the game was immersive (4.96), and especially that the virtual characters' dialogue had seemed authentic (5.67).

When asked what they would want to see added to the game, the most common answer among the subjects was "more content": a longer story, more characters and more things to do. Also, more freedom to choose what to do and in which order, was among the more common answers.

The questionnaire reveals that the game performed well technically and was easy to use (5.42). However, the game also wasn't seen very challenging (3.65) and most didn't need help in understanding the language used in the game (2.94). The game mostly seen as a refreshing approach to language education and thought to be a fun way to learn Finnish (5.89).

CONCLUSIONS

The aim of the demonstrative virtual language learning game experience developed within the project was to help in defining the guidelines for evaluating virtual learning environments. The initial questionnaire used to evaluate the application worked well and gave a clear indication that such experiences are effective method for both learning especially spoken second language and getting the students accustomed to everyday situations in the new culture.

Especially the design decisions aimed for creating a safe and friendly virtual environment for the students seems to have succeeded according to the results of the user testing. However, the game was possibly made too easy for the target audience, as it did not provide much challenge for most of the test subjects. This could be solved by having more content and by making the players face increasingly challenging situations as they progress in the game.

As many of the players reported to have been immersed in the game, even as it was played using a traditional desktop computer, it appears the decisions not to visualize the player's character and not to mimic photorealism, were correct. However, employing writers experienced in creating interactive content would likely have produced a more compelling story.

Virtual learning applications can be developed with limited resources while still allowing an experience that is both good enough for its purpose and most likely feasible commercially. An interesting development for the field is the advancement of affordable virtual reality headsets that allow even more immersive learning experiences than contemporary desktop computers.

Another game changing advancement in virtual learning technology will be the introduction of robust enough speech recognition algorithms that can accurately recognize even non-native speakers. This way the clumsy text based interface can be replaced with all audio user interaction methods that both support learning objectives and allow even higher levels of immersion. While some such applications exist already, like the Danish Simulator platform (Dansksimulatoren, 2015), it will take time before such tools are widely available for little spoken languages like Finnish.

ACKNOWLEDGEMENTS

This article was done as a part of the Developing Virtual Learning for Finnish project funded by European Social Fund. The project was conducted as collaboration between University of Turku's Technology Research Center and Brahea Centre, University of Tampere's School of Information Sciences, Tampere Adult Education Centre, and University of Jyväskylä's Language Centre, together with partners Axxell Ltd., Heuristica Ltd., PragmatIQ Ltd. and Ubiikki Ltd. We thank the VR-Group Ltd for the train car references and likeness.

REFERENCES

Canto, S., Jauregi, K. & van den Bergh, H. (2013). Integrating cross-cultural interaction through video-communication and virtual worlds in foreign language teaching programs: is there an added value? ReCALL: The Journal of EUROCALL, Vol. 25, Iss. 1, pp. 105–121.

Brooke, J. (1996). SUS: a 'quick and dirty' usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.), Usability Evaluation in Industry. London: Taylor and Francis.

Gee, J. P. (2008). "Learning and Games." The Ecology of Games: Connecting Youth, Games, and Learning. Edited by Katie Salen. The John D. and Catherine T. MacArthur Foundation Series on Digital Media and Learning. Cambridge, MA: The MIT Press, 2008, pp. 21–40.

IJsselsteijn, W.A., de Kort, Y.A.W., Poels, K., Jurgelionis, A., and Belotti, F. (2007). Characterising and Measuring User Experiences, ACE 2007 International Conference on Advances in Computer Entertainment Technology, Workshop 'Methods for Evaluating Games - How to measure Usability and User Experience in Games' (Salzburg, Austria, 13-15 June 2007).

Jauregi, K., de Graaff, R., van den Bergh, H. & Kriz, M. (2012). Native non-native speaker interactions through video-web communication, a clue for enhancing motivation. Computer Assisted Language Learning Journal, Vol. 25, Iss. 1, pp. 1–19.

Kiili, K. (2005). Digital game-based learning: Towards an experiential gaming model, The Internet and Higher Education, Vo. 8, Iss. 1, 1st Quarter 2005, pp. 13-24, ISSN 1096-7516

Lappalainen, Y. (2015). Avoimien virtuaaliympäristöjen opetuskäytön mahdollisuuksia. In Y. Lappalainen, M. Poikolainen & H. Trapp H. (Ed.) Tila haltuun! Suosituksia virtuaalisen suomen opiskelun toteuttamiseen. Turun yliopiston Brahea-keskuksen julkaisuja 6. University of Turku, Turku, Finland.

Lehtonen, T., Lakkala, M., Eloranta, J. & Rasila, M. (2015). Pedagoginen perusta kielenoppimisessa. In Y. Lappalainen, M. Poikolainen & H. Trapp H. (Ed.) Tila haltuun! Suosituksia virtuaalisen suomen opiskelun toteuttamiseen. Turun yliopiston Brahea-keskuksen julkaisuja 6. University of Turku, Turku, Finland.

Luckin, R. and Fraser, D. S., (2011). Limitless or pointless? An evaluation of augmented reality technology in the school and home. International Journal of Technology Enhanced Learning, Vol. 3, Iss. 5 (August 2011), pp. 510-524.

Ondrejka, C. (2008). Education Unleashed: Participatory Culture, Education, and Innovation in Second Life. The Ecology of Games: Connecting Youth, Games, and Learning. Edited by Katie Salen. The John D. and Catherine T. MacArthur Foundation Series on Digital Media and Learning. Cambridge, MA: The MIT Press, 2008. pp. 229–252.

Pagulayan, R., Keeker, K., Fuller, T., Wixon, D., Romero, R. & Gunn, D. (2007). User-centered Design in Games, Human-Computer Interaction Handbook. 2nd Edition

Pöyhönen, S., Tarnanen, M., Vehviläinen, E., Virtanen, A. & Pihlaja, L. 2010. Osallisena Suomessa: kehittämissuunnitelma maahanmuuttajien kotoutumisen edistämiseksi. University of Jyväskylä, Jyväskylä. Finland

Storhammar, M.-T. (1993). Ulkomaalaisopettajien opetuspuheen piirteitä. In L. Löfman, L. Kurki-Suonio, S. Pellinen & J. Lehtonen (Ed.) The competent intercultural communicator. AFinLA Yearbook 1993, pp. 79–97. Available at http://www.afinla.fi/sites/afinla.fi/files/1993Storhammar.pdf

Viinikkala, L., Leskinen, O.-P., Heimo, O., Korkalainen, T., Mäkilä, T., Helle, S., Pönni, V., Arimaa, J.-P., Saukko, F., Pääkylä, J., Jokela, S. & Lehtonen, T. (2014). The Luostarinmäki Adventure – An Augmented Reality Game in an Open Air Museum. NODEM 2014 – Engaging Spaces – Interpretation, Design and Digital Strategies.

WEB REFERENCES

Dansksimulatoren (2015). Dansksimulatoren. Available at http://www.dansksimulatoren.dk/

Epic Games (2015). Unreal Engine 4. Available at https://www.unrealengine.com/what-is-unreal-engine-4

National Board of Education (2015). Kielitaidon tasojen kuvausasteikko. Available at http://www.edu.fi/download/119698_taitotasot.pdf

Nevigo (2015). articy:draft 2. Available at http://www.nevigo.com/en/articydraft/overview/

Pixel Crushers (2015). Dialogue System. Available at: http://www.pixelcrushers.com/dialogue-system/

Unity Technologies (2015). Unity. Available at https://unity3d.com/

Evaluation of a Virtual Training Environment for Aggression De-escalation

Tibor Bosse[1], Charlotte Gerritsen[2], and Jeroen de Man[1]

[1]Department of Computer Science
VU University Amsterdam
Amsterdam, the Netherlands
t.bosse@vu.nl, j.de.man@vu.nl

[2]Netherlands Institute for the Study of
Crime and Law Enforcement
Amsterdam, the Netherlands
cgerritsen@nscr.nl

KEYWORDS
simulation-based training, aggression de-escalation, dialogue system, evaluation.

ABSTRACT

Public transport employees are confronted with aggressive behavior on a regular basis. As such encounters can have serious consequences, employees need to be well prepared, so that they know how to deal with incidents of aggression. The current paper describes an ongoing endeavor that is aimed at the development and evaluation of a simulation-based training environment for public transport employees, by which they can practice their verbal aggression de-escalation skills during face-to-face conversations. A prototype of the training environment is presented, as well as an experiment to evaluate the environment in several steps. The results indicate that the prototype is evaluated positively with respect to user satisfaction, whereas there is room for improvement with respect to learning effectiveness.

INTRODUCTION

People working in the public sector often have to deal with aggressive behavior. According to a national safety investigation in the Netherlands in 2011, almost 60% of the employees is confronted with unwanted behavior on a daily basis (Abraham et al., 2011). This behavior can include verbal or physical aggressive behavior, but also sexual assault or discrimination.

The municipal public transport operator in Amsterdam (GVB) is one of the organizations of which the employees have to face aggressive behavior on a regular basis. In 2014, the GVB reported 443 incidents of aggressive behavior against employees (GVB, 2012). Only a small part of this number relates to physical incidents, but verbal forms of aggression can be perceived as unwanted as well. Typical examples of incidents are situations where travelers insult a bus driver, or intimidate a tram conductor to get a free ride. Such confrontations may have a range of serious consequences for employees, including reduced work pleasure, decreased work performance, sick leave, various mental health symptoms and even post-traumatic stress disorder (PTSD).

To better prepare them for these incidents, companies like the GVB offer their employees resilience training. Such training is typically performed in a group setting based on role-play, where employees learn to communicate with aggressive clients in a de-escalating manner. Although this form of training has shown to be successful, it is quite expensive with respect to both money and time. Furthermore, the training is not always easy to control or repeat systematically.

As a complementary approach, we propose the use of simulation-based training of aggression de-escalation. This is in line with a number of recent initiatives that show promising results regarding the possibility to train social and communicative skills based on simulated environments involving virtual humans (Bruijnes et al., 2015; Hays et al., 2012; Kim et al., 2009; Vaassen and Wauters, 2012). The main idea of the current system is that public transport employees can practice their aggression de-escalation skills by engaging in conversations with aggressive virtual travelers. By designing the scenarios in such a way that the virtual characters calm down if they are being approached correctly, but become more aggressive if they are being treated inappropriately, trainees will receive immediate feedback on their performance. By using such a system, employees have the ability to practice their aggression de-escalation skills in a cost-effective, personalized and systematic manner.

In this paper, a prototype of such a training environment is presented, which has been developed in collaboration with the public transport company GVB. In addition, an experiment is described that has been performed to evaluate different aspects of the training environment.

LEARNING GOALS

To design an effective training tool, a first question to be asked is what should be the learning goals of the system. For the current context, these learning goals are similar to the ones used in the real world training of the public transport company, and are related to the development of *emotional intelligence*: employees should be able to recognize the emotional state of the (virtual) conversation partner, and choose the communication style that suits this emotional state.

More specifically, when it comes to aggressive behavior, it is important that employees learn to recognize the nature of the aggression. Here, two main categories can be distinguished: aggression can be either *emotional* (or *reactive*) or *instrumental* (or *proactive*) (Dodge, 1990). One of the key

differences between these two types is the absence or presence of anger (Miller and Lyna, 2006).

In case of emotional aggression, the aggressive behavior typically is caused by an angry reaction to a negative event that frustrates a person's desires, cf. the *frustration-aggression hypothesis* (Berkowitz, 1978). Such a person is likely to be angry with respect to whatever stopped him from achieving his goal. By a carry-over effect, the anger can be transferred to new situations as well (Angie et al., 2011). Examples in the public transport domain are people getting angry because the tram is late while they have to attend an important meeting, or because they want to enter the tram while carrying food or drinks that are not allowed. When dealing with an emotional aggressor, *supportive* behavior from the de-escalator is required, for example by ignoring the conflict-seeking behavior, calmly making contact with the aggressor, actively listening to what he has to say, showing empathy, and suggesting solutions to his problems.

In contrast, in case of instrumental aggression, the aggressive behavior is only used 'instrumentally', to achieve a certain predetermined goal. Such behavior is not a direct response to a negative event and is less strongly related to heavy emotions. A well-known example of this type of aggression in the domain of public transport involves someone who wants to travel without paying for his ticket. This type of aggression often starts with an attempt to persuade the conversation partner, e.g. "Oh, I forgot my wallet, can I just come along for two stops?" or "Hi honey, I don't have to pay for a short ride, do I?". Often, in case the employee does not give the aggressor what he wants, the aggressive behavior will reveal itself through more threatening remarks like "I know where you live", "I will be back tomorrow with my friends", or "I will be waiting for you at the end of your shift".

A possible basis for this behavior can be found in the social learning theory, which states that if a person has used aggression to achieve a goal in the past, and if this behavior was successful, then by operant conditioning (s)he will be likely to follow the same behavioral pattern in the future. So, the behavior is learned through positive reinforcement (Bandura, 1963). Hence, to de-escalate instrumental aggressive behavior, a *directive* type of intervention is assumed to be most effective. It is necessary to show the aggressor that there is a limit to how far he can pursue his aggressive behavior, and to make him aware of the consequences of this behavior.

To conclude, the presented training environment will be centered around two main learning goals, namely 1) *recognizing* the type of aggression of the conversation partner (i.e., emotional or instrumental), and 2) selecting the appropriate *communication style* towards the conversation partner (i.e., supportive or directive).

To assess the type of aggression, employees need to carefully observe the verbal and non-verbal behavior of the aggressive individual. In general, reactive aggressors will show more arousal (e.g., flushed face, emotional speech) than proactive aggressors. Also, the context should be taken into account (e.g., someone who just finds out that he lost his ticket will be more emotional that someone who knew this all along, and just tries to intimidate the tram driver to ride for free).

TRAINING ENVIRONMENT

In Bosse et al. (2014), a global overview is presented of the simulation-based training environment that is being developed within the current project. The environment consists of two main components, namely a *virtual reality environment* and a *training agent*. The virtual reality environment has the form of a 3D graphical environment that simulates a particular context in the real world (e.g., the interior of a tram including travelers),[1] with which the user can interact based on a dialogue system. The training agent is an intelligent virtual tutor that monitors the behavior of the trainee and generates personalized support. Two types of support are used, namely run-time modifications of the scenario to adjust its difficulty level to the trainee's performance (scaffolding) (Bosse et al., 2015), and personalized feedback on the trainee's performance in terms of after-session hints (Bosse and Provoost, 2015). To evaluate the overall training environment in a systematic manner, the current paper focuses exclusively on the virtual reality component (and the underlying dialogue system); evaluation of the training agent is left for future research.

The virtual reality environment is based on the InterACT software,[2] developed by the company IC3D Media.[3] InterACT is a software platform that has been specifically designed for simulation-based training of interpersonal skills. Unlike most existing software, it focuses on smaller situations, with high realism and detailed interactions with virtual characters. True-to-life animations and photo-realistic characters are used to immerse the player in the game. An example screenshot of a training scenario for the public transport domain is shown in Figure 1. In this example, the user plays the role of a tram conductor that has the task of calming down an aggressive virtual traveler.

To enable users to engage in a conversation with an emotional conversational agent (ECA), a dialogue system based on conversation trees is used. The system assumes that a dialogue consists of a sequence of spoken sentences that follow a turn-taking protocol. That is, first the ECA says something (e.g. "I forgot my public transport card. You probably don't mind if I ride for free?"). After that, the user can respond, followed by a response from the ECA, and so on. In InterACT, these dialogues are represented by conversation trees, where vertices are either atomic ECA behaviours or decision nodes (enabling the user to determine a response), and the edges are transitions between nodes.

[1] Although the focus of this paper is on public transport, in principle the approach can be applied to any domain involving aggressive behavior in face-to-face conversations.

[2] http://www.interact-training.nl/.

[3] http://ic3dmedia.com/.

The atomic ECA behaviors consist of pre-generated fragments of speech, synchronised with facial expressions and possibly extended with gestures. Scenario developers can generate their own fragments using a motion sensing input device such as the Microsoft Kinect camera and a commercial software package FaceShift.[4] As the recorded fragments are independent from a particular avatar, they can be projected on arbitrary characters.

Each decision node is implemented as a multiple choice menu. Via such a menu, the user has the ability to choose between multiple sentences. Hence, the emphasis of the current system is on the verbal aspects of aggression de-escalation. In the system used for the current study, three options are available with every decision node. These options have been created in such a way that one of them is clearly *supportive*, another one is clearly *directive*, and the third option is neutral. Here, the supportive and directive option relate to the communication styles explained earlier. Figure 1 illustrates how these three options can be instantiated in terms of concrete sentences (in this case: A=neutral, B=directive, C=supportive).

For the current evaluation study, a number of scenarios have been developed, in collaboration with (and approved by) domain experts of the public transport company. To be precise, the scenarios address 9 different situations in which a conflict may arise, such as 'traveler is not allowed to take hot coffee on board' and 'tram arrives 10 minutes late'. Moreover, for each scenario three variants have been written: two variants in which the virtual character shows emotional aggression, and one in which it shows instrumental aggression.

The contents of the scenarios (i.e., the conversation fragments) have been recorded with the help of professional trainers of the public transport company. Each of the 9x3 scenarios has been recorded with a female trainer and with a male trainer, with a specific focus on showing emotional behaviors. Hence, in total a set of 54 scenarios (9x3x2) has been created. The scenarios have been set up in such a way that if the user takes the appropriate communication style, the character calms down and conflict is resolved; however, if the user takes an inappropriate communication style, the situation will escalate. On average, a scenario lasts about 3 to 4 interactions (i.e., both the user and the virtual character speak 3-4 sentences before the scenario ends).

METHOD

This section describes the experiment that was conducted to investigate the impact of the virtual training environment on the user experience as well as the performance of potential end-users from the public transport domain.

Participants

Initially, 30 people were selected to participate in the experiment. All participants were employees of the public transport company (in particular: professional tram conductors and tram drivers). Among these participants, initially 15 were allocated to the training group and 15 to the control group, based on their availability. However, after this allocation had been made, 6 participants withdrew from the study. This resulted in a training group of 14 participants and a control group of 10 participants. Within the training group, 8 participants were male and 6 were female. The average age in this group was 42,7 (σ = 13.1). Within the control group, 5 participants were male and 5 were female. The average age in this group was 48,3 (σ = 9.9).

Experimental Design and Procedure

For the experiment, a pre-test post-test design has been used, where the pre-test and the post-test were separated by a period of 4 weeks. At the start of the pre-test, all participants (in both groups) filled out an informed consent form and

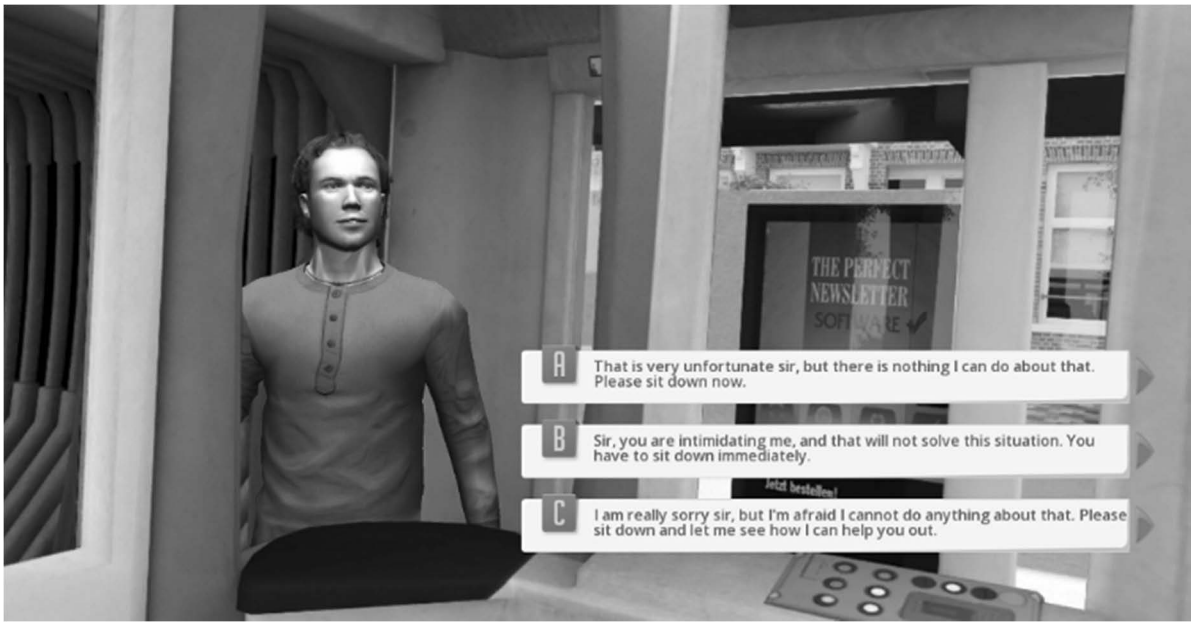


Figure 1: Example screenshot of a training scenario

[4] http://www.faceshift.com/.

provided their personal data. This, as well as all other data gathered in this experiment, was collected anonymously. After that, they made the pre-test, which had the form of a written exam that had been developed in advance by instructors of the public transport company. The exam was composed of 7 multiple choice questions with 4 options each and 5 open questions, which were designed to be representative for the learning goals of the training environment. All closed questions consisted of a particular context description, similar (but not identical) to the ones used in the virtual training (e.g., "a traveler enters the tram and shouts to you that he refuses to pay for his ticket because your tram is much too late"), followed by four alternative responses of which the participant should select the most appropriate one. The open questions were more general, but also related to the learning goals (e.g., "how can you recognize emotional aggression of a traveler?").The post-test was also made by all participants. This test also had the form of a written exam; it had the exact same structure as the pre-test, only the contents of the scenarios and questions were slightly modified to prevent a learning effect (e.g., by changing some properties of the main character, or by rephrasing the multiple choice answers).

In the period between the pre-test and the post-test, the training group performed 4 training sessions, in which they worked with the software for about 30 minutes. More details about these sessions is provided in the next sub-section. The control group did not participate in these training sessions.[5]

After the last training session, the participants in the training group filled out a usability questionnaire. This questionnaire consisted of 13 statements about which the participants had to express their opinion on a 5-point Likert scale. The questionnaire was inspired by Witmer and Singer (1998), and included statements about issues such as user experience, presence, and perceived effectiveness. In the end, the statements were grouped into 4 categories, namely *content*, *interaction*, *emotional*, and *effect*, to obtain an average score on these aspects. The *content* category contained statements about the perceived realism of the scenarios and the characters (e.g., 'the virtual characters showed believable behavior'). The *interaction* category contained statements about how natural it was to interact with the characters (e.g., 'I felt that my answers had an influence in the behavior of the virtual characters'). The *emotional* category addressed the perceived sense of presence of the participants (e.g., 'during training I felt engaged in the scenarios'). Finally, the *effect* category contained statements asking the participants for their opinion about the effectiveness of the training (e.g., 'I think this type of training is a useful addition to real world training').

Training Sessions

All training sessions were executed in a computer room at the public transport company. At the start of a session, participants received a document with instructions about how to work with the training software. They were instructed to solve each virtual scenario to the best of their ability by identifying the type of aggression they observed during the conversation and by selecting the appropriate response in the multiple choice menu. After having read the instructions, they could start the training software.

Upon launching the software, the start menu shown in Figure 2 was displayed. In the upper part of the menu, participants had to input their personal ID and gender. Below that, they could select which scenarios they wanted to run. As can be seen, there were 4 training sessions (corresponding the 4 weeks of the training), each of which consisted of 10 scenarios. The sets of scenarios were chosen in such a way that they were representative for the types of incidents encountered on the job (for instance, they contained more male aggressors than female aggressors, and more cases of emotional aggression than instrumental aggression). All 40 scenarios were slightly different from each other; hence, no scenario was presented more than once. The order in which the scenarios were offered was determined randomly.

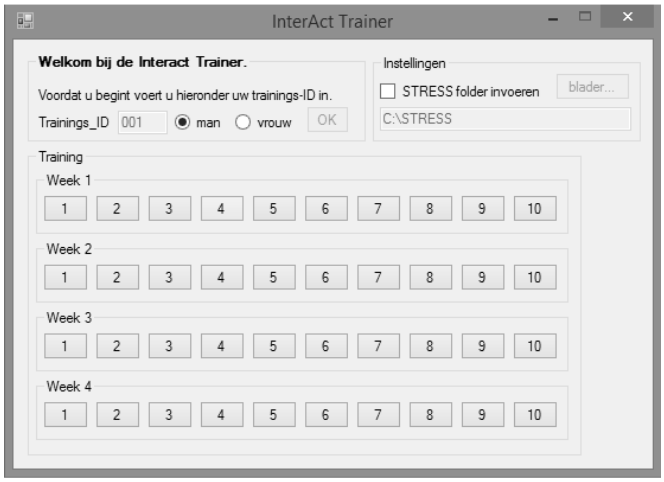


Figure 2: Start menu of the training software (in Dutch)

At the end of each scenario, participants had to indicate whether they thought the aggressive behavior shown by the virtual character was emotional or instrumental. All choices they made in the multiple choice menu were logged, as well as the time it took them to play a scenario.

Variables

The variables that were measured during the study were selected in such a way that they could roughly be related to the training evaluation model by Kirkpatrick (1994). This model distinguishes four levels on which training programs can be evaluated, namely *satisfaction* ('did the participants enjoy/appreciate the training?'), *learning* ('was there an increase in knowledge/skills during training?'), *impact* ('did the participants change their behavior on the job as a result of the training?'), and *results* ('did the training positively affect the organization?'). In the current study, the emphasis is on the first two levels (satisfaction and learning), where the evaluation of *learning* can be further divided into two sub-questions, namely 'did the participants' performance within the training environment improve over time?' and 'did the training result in an increased performance in a different environment that involves the same skills?'. Below,

[5] Note, however, that the participants in both groups did continue their regular work activities in the meantime.

we will refer to these two aspects of learning by *learning during training* and *transfer of learning*, respectively.

Based on this categorization, we can relate the different levels of evaluation to measurable variables in the following way. To evaluate *satisfaction*, the results of the usability questionnaires filled out by the training group (i.e., the answers given to the Likert questions) were analyzed.

To evaluate *learning during training*, the behavior of the participants of the training group during the training sessions was analyzed. In particular, we measured their performance in terms of *identification* (i.e., how well are they able to recognize the type of aggression of the virtual characters?) and *response* (i.e., how well are they able to provide the appropriate responses to the aggressive behavior). As both measures were applied to emotional as well as instrumental aggression separately, this resulted in 4 scores (2x2) to evaluate learning during training. By observing the change of these scores over the four weeks of the experiment, we could evaluate whether the participants improved over time.

To evaluate *transfer of learning*, the written exams made during the pre- and post-test were used. Here, by comparing the scores for the pre-test with the scores for the post-test (in a within-subjects analysis), we could investigate whether the participants' knowledge had improved. Additionally, by comparing the improvement of the training group to that of the control group (in a between-subjects analysis), we could investigate whether the training had an added value over the regular work activities. In this analysis, the independent variable was the condition (i.e., training or no training), and the dependent variable was the change in score between the pre- and post-test. The scores for the pre- and post-tests were obtained by having an instructor of the public transport company grade all exams.

Finally, note that besides for evaluation, the performance of the participants in the different tests (the pen-and-paper exams and the simulation-based training) could be used for assessment purposes as well. That is, by observing the behavior of their employees during the study, the public transport company could gain more insight in how they act in various situations that are representative for real world incidents.

RESULTS

In the following sections, the results obtained during this evaluation study are presented as described above. That is, the first part shows the *satisfaction* of the participants using the training software, while the section thereafter present their *learning during training*. The final section shows the results on the pen-and-paper exams with regard to the *transfer of learning*.

Satisfaction

The experimental group completed a questionnaire asking about their opinion on the training sessions. The answers to these questions are grouped in four categories as explained above; interaction, content, emotional and effect. The scores (on a scale from -2 up to 2) are shown in Figure 3.

The first category, *content*, contained questions regarding the scenarios and virtual characters. With an average score of 0.5 the results were mainly positive, however there were critical remarks as can be seen by the rather larger standard deviation of 0.66. Similar results are found for the second category, *interaction,* and are, with an average score of 0.4 and a standard deviation of 0.71, again mainly positive with some negatives.The worst results are found on the category asking about the *emotional* aspects of the training. This entailed questions about their personal involvement in the scenario or whether they got frightened by the aggression of the virtual characters. With an average score of -0.3 the results do not look promising, however again the standard deviation is rather large (0.73) indicating some positive results as well.The last category contained questions to their personal belief whether such a training has an *effect*. For example, if they think they improved in their interaction with travelers or if they believe such a training is useful addition to the current role-play scenarios. Overall, responses to these questions were positive (average 0.7), with almost no negative scores across the participants (standard deviation 0.52).

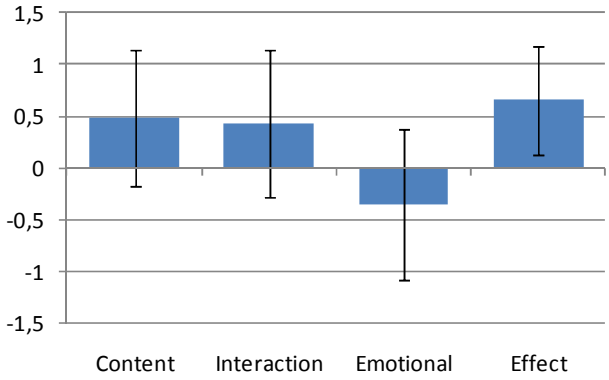


Figure 3: Average scores on satisfaction

Learning during Training

The experimental group underwent 4 weekly training sessions, each of which consisted of 10 scenarios. For each scenario they had to identify the type of aggression as well as respond correctly to de-escalate the situation. Figure 4 shows both the percentage of correctly identified aggression types as well as the correct responses split into instrumental (i) and emotional (e) aggression per week. Higher scores are better, where scores of 0.5 for identification and 0.34 for response would be expected with random answers.

Firstly, none of the measurements show an increase over time, indicating there is no real increase in performance over these 4 weeks. But, when taken a closer look, it can be seen that participants were able to identify emotional aggression correctly approximately half the time and subsequently responded well half of the time. However, instrumental aggression was identified correctly more often, while the response on these situations was the worst of all. This is confirmed by paired t-tests as well; the difference between the identification of and response to instrumental aggression is significant ($t(46) = 7.37$, $p < 0.001$), while for emotional aggression this is not the case ($t(46) = 1.45$, $p = 0.153$). Furthermore, instrumental aggression was identified

correctly more often ($t(46) = 5.87$, $p < 0.001$), while the response to emotional aggression was significantly better ($t(46) = 3.45$, $p = 0.001$).

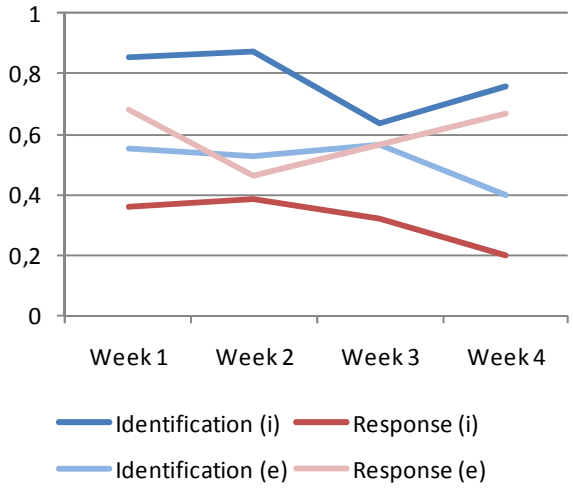


Figure 4: Average scores for learning during training

Transfer of Learning

The results of the pen-and-paper tests of both the experimental and control group are shown in Figure 5. The total score is subdivided in a score for the open questions (blue color, max. 10 points) and multiple choice questions (red color, max. 7 points). The error bars represent the standard deviation of the total score.

It can be seen that both groups performed better on the post-test as confirmed by a t-test with $t(23) = 2.64$, $p = 0.014$ for the experimental group and $t(18) = 3.31$, $p = 0.004$ for the control group. On closer examination, it turns out that there is no significant change in the score on the multiple choice questions, but the higher scores are due to better answers on the open questions. The important question is whether the experimental group experienced a greater increase than the control group, which unfortunately does not show in this data ($t(21) = 0.06$, $p = 0.950$).

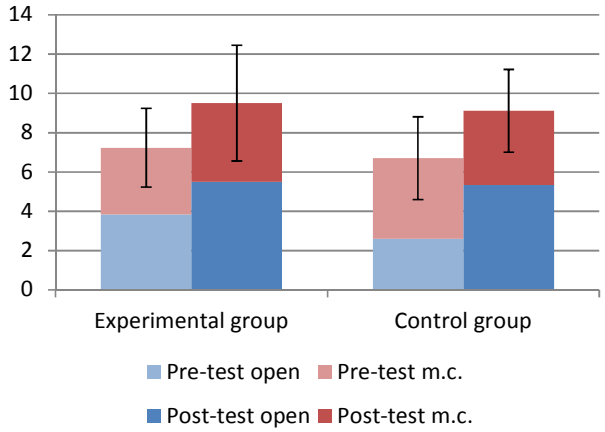


Figure 5: Average results for transfer of learning

DISCUSSION

Firstly, did the virtual training help improve the participants in the experimental group more than those that did not use the training? Unfortunately not, but nevertheless this research helps us in understanding why. Considering the difference between the pre- and post-test, the improvement in both groups was mainly due to a better score on the open questions. As they did the same tests, this improvement could be due to the second test being a bit easier, in which case there would be no 'real' improvement in either group. Another potential explanation could be that merely being part of the experiment already made the participants reflect on the topic of aggression de-escalation during the 4 weeks of the experiment (even if not all of them participated in the training sessions). This might explain why both groups obtained a better score in the open questions.

On the other hand, participants did not seem capable of translating this increased understanding of aggression de-escalation (as measured with the open questions) to correct decisions on how to act in concrete situations (as measured with the closed questions). This is in line with the results retrieved from the training sessions, which did not show any improvement in performance during the scenarios.

Then, what do these results learn us? Throughout the training, participants identified instrumental aggression quite well, but did not respond accordingly. As the correct response for instrumental aggression is very direct, it might be that they preferred a more 'polite' or 'friendly' answer. Another explanation for the lack of improvement might be that each of the participants already has a set way of responding and has difficulty in changing this 'default' approach. This option is backed by the data as well; by looking at the scores of those participants with more than 2 years of experience in comparison with the others, there is no difference in test scores as well (pre-test $t(20) = -0.72$, $p = 0.480$; post-test $t(20) = -1.09$, $p = 0.288$).

From this, the conclusion might be drawn that on your own, it is difficult to learn correct responses for the different types of aggression. Then, it would be important to provide timely feedback such that a trainee understands the mistake and is able to improve on it. Providing such feedback was not yet implemented in this training software, but is being developed (Bosse and Provoost, 2014).

It should be considered as well that there might be a more fundamental problem, such as a flaw in the pedagogical approach or simply a lack of motivation from the participants. However, from the subjective evaluation, it is clear that participants do see the usefulness of such a training and already experience a belief of improvement due to it. Nonetheless, it is important to consider methods to improve the emotional involvement of trainees during the various scenarios as this was shown to be insufficient and could potentially affect the learning as well. Improvements can be made by for example changing from a standard desktop screen to a head-mounted display or increasing the intensity of the aggression shown by the virtual agents. All in all, we believe the potential of such a virtual training is supported by these results.

CONCLUSION

The current paper introduced a prototype of a simulation-based training environment that enables public transport employees to practice their verbal aggression de-escalation skills during face-to-face conversations. The design of the system is centered around two learning goals, namely the ability to recognize the type of aggression (emotional or instrumental) and the ability to select the most appropriate communication style for the observed aggression type (supportive or directive).

The prototype was evaluated by means of an experiment in which 24 employees of the public transport company of Amsterdam participated. The results indicate that with respect to user satisfaction, participants were moderately positive about the content of the virtual scenarios and the mechanisms to interact with the characters. Also, they were very positive about the potential of the system as an effective learning tool. The only category for which their opinion was below neutral involved their perceived sense of (emotional) engagement and presence.

Regarding the performance during training, no significant improvement was found, which might be explained by the fact that this particular task is difficult to learn without specific feedback. In line with these results, also no transfer of learning was found to a similar task on paper (in particular, to the closed questions, where participants had to indicate how they would behave in fictional scenarios). In contrast, participants in the training group did show an improved performance regarding the open questions of this paper task, but this improvement was not significantly larger than that of the control group.

Finally, an interesting side effect was that the training environment also proved useful as an assessment tool. For instance, it allowed us to conclude that the current group of participants is significantly better in identifying proactive aggression than reactive aggression, but at the same time has significantly more difficulties in dealing with proactive aggression than with reactive aggression.

Inspired by the current findings, our future research will concentrate on two main aspects. First, we will try to incorporate additional elements in the training with the aim to enhance users' emotional engagement and presence. Examples of such elements are more extreme aggressive behavior of the virtual characters (e.g., louder voice volume, more threatening facial expressions and utterances), the use of immersive technology like head-mounted displays, and the use of mechanisms to introduce a 'simulated threat', e.g., based on air blast devices or electric surges. Secondly, we will integrate our previously developed modules for learner feedback (Bosse and Provoost, 2014) within the system, to explore whether this has a positive impact on learning effectiveness.

ACKNOWLEDGEMENTS

This research was supported by funding from the National Initiative Brain and Cognition, coordinated by the Netherlands Organization for Scientific Research (NWO), under grant agreement No. 056-25-013. The authors wish to thank Maria ter Beek, Ernst van der Horst, Aly Lubberink, Jermaine Ravenberg, Caroline de Ridder, Petra ter Weeme, and all participants in the study for their contribution to the project.

REFERENCES

Abraham, M., Flight, S., and Roorda, W. 2011. "Agressie en geweld tegen werknemers met een publieke taak". Research for the program 'Veilige Publieke Taak 2007 - 2009 – 2011'. Amsterdam: DSP.

Angie, A.D., Connelly, S., Waples, E.P., and Kligyte, V. 2011. "The influence of discrete emotions on judgment and decision-making: A meta-analytic review". Cognition & Emotion, 25(8), 1393-1422.

Bandura, A. 1963. "Social learning and personality development". New York: Holt, Rinehart, and Winston.

Berkowitz, L. (1978). "Whatever Happened to the Frustration-Aggression Hypothesis?" American Behavioral Scientist, 21, 691-708.

Bosse, T., Gerritsen, C., and Man, J. de. 2014. "Agent-Based Simulation as a Tool for the Design of a Virtual Training Environment". In: Proceedings of the 14th International Conference on Intelligent Agent Technology, IAT'14. IEEE Computer Society Press, pp. 40-47.

Bosse, T., Gerritsen, C., Man, J. de., and Tolmeijer, S. 2015. "Adaptive Training for Aggression De-escalation". In: C.J. Headleand et al. (eds.), Proceedings of ALIA 2014, CCIS 519, Springer Verlag, pp. 80-93.

Bosse, T. and Provoost, S. 2014. "Towards Aggression De-escalation Training with Virtual Agents: A Computational Model". In: Proceedings of the 6th International Conference on Human-Computer Interaction, HCI'14. Springer Verlag, pp. 375-387.

Bosse, T. and Provoost, S. 2015. "Integrating Conversation Trees and Cognitive Models within an ECA for Aggression De-escalation Training". In: Proceedings of PRIMA 2015, Springer Verlag, in press.

Bruijnes, M., Linssen, J.M., op den Akker, H.J.A., Theune, M., Wapperom, S., Broekema, C., and Heylen, D.K.J. 2015. "Social Behaviour in Police Interviews: Relating Data to Theories". In: Conflict and Multimodal Communication, Springer Verlag, pp. 317-347.

Dodge, K.A. 1990. "The structure and function of reactive and proactive aggression". In D. Pepler and H. Rubin, (eds.), The development and treatment of childhood aggression (pp. 201-218). Hillsdale, NJ: Erlbaum.

GVB Amsterdam. 2012. "Incidenten overzicht 2002 tot heden". Technical Report.

Hays, M., Campbell, J., Trimmer, M., Poore, J., Webb, A., Stark, C., and King, T. 2012. "Can Role-Play with Virtual Humans Teach Interpersonal Skills?". In Interservice/Industry Training, Simulation and Education Conference (I/ITSEC).

Kim, J., Hill, R.W., Durlach, P., Lane, H.C., Forbell, E., Core, C., Marsella, S. Pynadath, D. and Hart, J. 2009. "BiLAT: A game-based environment for practicing negotiation in a cultural context". International Journal of Artificial Intelligence in Education, vol. 19, issue 3, pp. 289-308.

Kirkpatrick, D.L. and Kirkpatrick, J.D. 1994. "Evaluating Training Programs". Berrett-Koehler Publishers.

Miller, J.D. and Lyna, D.R. 2006. "Reactive and proactive aggression: Similarities and differences". Personality and Individual Differences, 41(8), 1469-1480.

Vaassen, F. and Wauters, J. 2012. "deLearyous: Training interpersonal communication skills using unconstrained text input". In: Proceedings of ECGBL. pp. 505–513.

Witmer, B.G. and Singer, M.J. 1998. "Measuring presence in virtual environments: a presence questionnaire". Presence: Teleoperators and Virtual Environments 7, pp. 225-240.

GAME AI

STARCRAFT II BUILD ITEM SELECTION WITH SEMANTIC NETS

Andreas Stiegler
Stuttgart Media University
Nobelstraße 10
70569 Stuttgart, Germany
E-mail: stiegler@hdm-stuttgart.de

Keshav Dahal
University of the West of Scotland
Paisley Campus
PA1 2BE Paisley, Scotland
E-mail: keshav.dahal@uws.ac.uk

Johannes Maucher
Stuttgart Media University
Nobelstraße 10
70569 Stuttgart, Germany
E-mail: maucher@hdm-stuttgart.de

Daniel Livingstone
Glasgow School of Art
Digital Design Studio
G3 6RQ Glasgow, Scotland
E-mail: d.livingstone@gsa.ac.uk

KEYWORDS
RTS, Reasoning, Semantic Structures, StarCraft II

ABSTRACT

We are proposing to express both the dynamic world state and domain knowledge in a single semantic net for a player-level AI playing the RTS game StarCraft II. The AI uses a utility system and queries on the semantic net to solve high-level planning problems. We explain the different components of the AI and illustrate the reasoning process through the example of selecting which unit to produce, given a specific game state. To do this, we introduce counters-relations, mapping a typical RTS game mechanic on relations in semantic structures, incorporating data derived from test games and tailored scenarios to measure how well unit classes counter each other. Finally, we discuss the issue of precise Micro-Management, which is not yet covered fully by the presented approach for a player-level AI. We give an outlook on related reasoning systems for squads and individual units and how they could interact with the semantics-based planner.

INTRODUCTION

The Real-Time Strategy games are an interesting application for game-AI research offering rich challenges to both human and AI players was proposed by Buro (Buro 2003; Buro 2004) and later on revisited by many researchers, such as (Yannakakis 2012). They usually require significant strategic and tactical planning as well as multi-tasking. There are many Real-Time Strategy games each with their individual problems and game mechanics. Some focus on building and maintaining complex economies, while others skip any kind of economy altogether and instead focus on tactical unit control in combat.

Among them, we chose StarCraft II as a platform for various reasons. StarCraft II is tailored towards eSport (Browder 2011) and offers a competitive setting and unit balancing. Our research targets cooperation between human and AI players in StarCraft II. In this scenario, players have to agree on the overall strategy (often called a "Build"), as well as tactical parameters like attack directions and special weapon timings (for example when to use Nuclear Weapons). Defining a strategy typically happens outside of an actual game round (the "Meta Game"), where communication can also be important (Rabin 1994).

We chose StarCraft II as a platform for our experiment due to its focus on eSport. The game design of StarCraft II avoids dominant strategies. If such a strategy would exist, chances are that the AI and the human player would just follow such a dominant strategy leading to "accidental cooperation" without any communication or agreement at all.

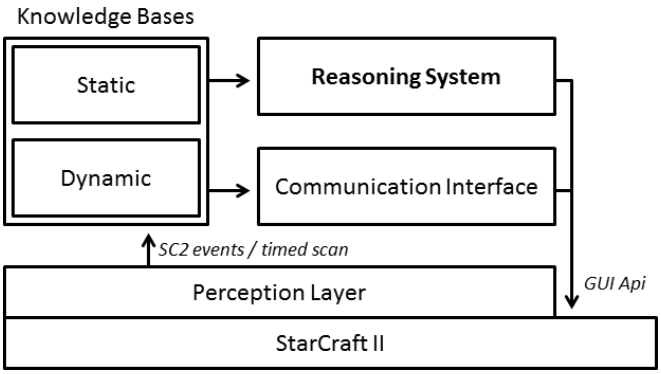
Further, StarCraft II is a typical example of the Real-Time Strategy genre and includes most of the typical challenges: a simple economy, complex tactical relations between multiple unit types, a technology tree to advance in and micro-management during battle. Its predecessor, StarCraft, is also a commonly used platform for AI research in Strategy games, including fields such as unit-oriented reasoning via approaches like goal-driven autonomy (Weber 2010), micromanagement of individual units (Synnaeve 2011), pathfinding aside from the famous A* (Hagelbäck 2012), exploration (Togelius 2010) and high-level planning (Churchill 2011). While different in some details, StarCraft II shares many similarities to StarCraft.

Being an adversarial game, one way to measure the performance of an AI in StarCraft II is to measure its capability to win the game through the means of the game mechanics. Yet, when interacting with a human player in a cooperative scenario, it also becomes important how effective and enjoyable cooperation with the AI is. This is similar to the "play to win or play to be fun" (Yildirim 2008; Laird 2001) problem in game AI, where "artificial stupidity" (Lidén 2003) becomes an important factor to be entertaining and believable (McGee 2010). We chose the competitive performance of AI-Human teams as a metric for evaluation, excluding the aspect of enjoyable communication for now.

We will present a bot that uses semantic nets as its key data structure to represent domain knowledge and memory. We will illustrate the reasoning process by demonstrating how the bot solves a high-level-planning problem: To decide which unit to build. For this task, counters are a core concept in RTS games.

BOT ARCHITECTURE

The Bot's architecture consists of 5 core pieces: The Perception Layer, the Static and Dynamic Knowledge Base, the Reasoning System and a Communication Interface.



Figures 1: Architecture Overview. Feeding commands into StarCraft II uses the same interface that human players would interact with through the GUI.

The Perception Layer is an abstraction of the StarCraft II API gathering information and forwarding events to the other AI layers. The Perception Layer only forwards information that a human player would be able to perceive in a given situation, too. It preserves fog-of-war, for example.

Both knowledge bases form a connected semantic net, which serves as the key data structure for the bot's reasoning. The Static Knowledge is given at game start and represents domain knowledge of the game, such as the starting health and damage values of different unit classes. The Static Knowledge Base never changes during a game round. The Dynamic Knowledge, in contrast, is allocated by the bot at runtime and stores information on the current gameplay situation, such as the current health of a specific unit. The Dynamic Knowledge spans relations towards the Static Knowledge base. A specific unit in play, for example, is represented in Dynamic Knowledge but will have "is-a" relations towards Static Knowledge nodes, such as the unit class to which the unit belongs to. More information on the structure of the Knowledge Bases can be found in (Stiegler 2013a; Stiegler 2014) In a typical game round of StarCraft II, the semantic structures peak at ~500 nodes with ~4500 relations, of which ~65% consist of nodes and relations from the Static Knowledge Base, with the rest being dynamically allocated through the AI in the Dynamic Knowledge Base.

The Communication Interface in a separate module offering an interface towards the human ally. It connects to the Knowledge Bases, as collaborative strategies can change aspects of the game mechanics. If both allies, for example, agree to split air and ground defense between them, then this has to be represented in the Dynamic Knowledge Base by altering relations. The Communication Interface also connects to the Reasoning System, as communication is bidirectional, meaning that the AI can initiate communication with the human on its own or perform counter offers. It can be argued that communication is an integral part of a cooperative RTS and as such, the Communication Interface should have been wired in more deeply. We chose its rather lose connection to the remaining architecture layers, as we expect it will undergo the most development and changes in the future. Currently, it only supports simple symbolic communication, but utilizing a more complex NLP solution seems very attractive, as all of the bot's knowledge is already based on semantic structures, a common knowledge representation for NLP systems (Waltz 2014). More information on the current version of the Communication Interface can be found in (Stiegler 2013b).

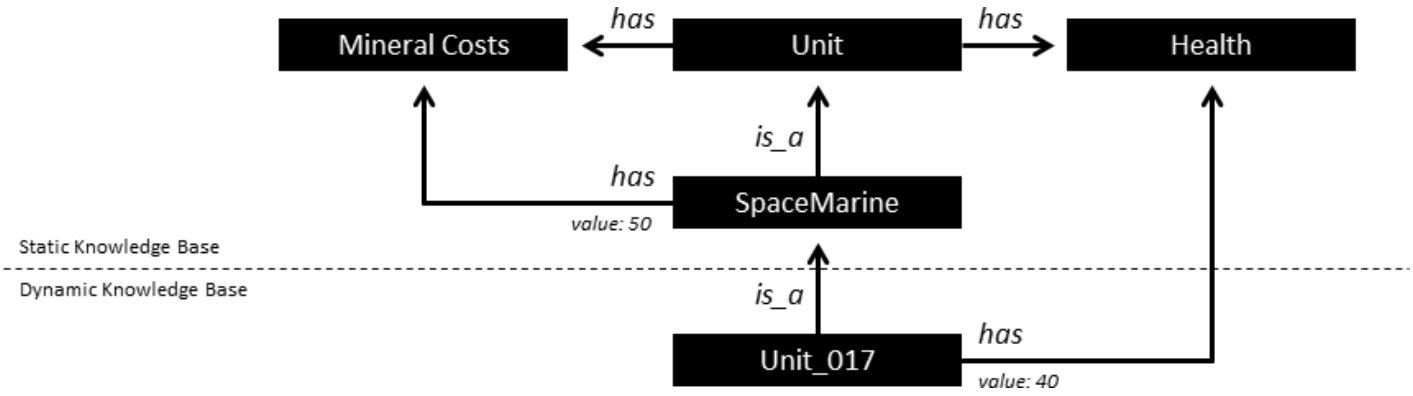
At the core of the bot is the Reasoning System. The Reasoning System consists of a collection of algorithms and callbacks, working on the two knowledge bases. It is a utility system selecting from the whole action space in given intervals. Each utility function is composed of operations on the semantic structure. These operations are described through semantic structures themselves, which are again part of the Static Knowledge Base, allowing the bot to switch between different sets of utility functions at runtime.

Further, the Reasoning System periodically calls refinement operations on the Dynamic Knowledge Base. As the game continues, the Dynamic Knowledge Base tends to become vast and fragmented, leading to perceivable time lags in AI reasoning or greatly reduced performance. Some of the refinement operations try to clean the Dynamic Knowledge Base up, for example by truncating relations from dead unit nodes or by abstracting squads of units into squad nodes. Other refinement operations are just functions on the semantic structure altering parameters of the reasoning system, such as switching between utility functions or modifying dynamic knowledge to compensate for uncertainty, such as the army predictors trying to predict what the opponent's army is composed of based on past observations (Stiegler 2014).

Most of the processing time of the reasoning system (>70% in average) is spent in search and inference operations on the semantic structure, whereas the actual action selection is just a simple utility system. Goals are implicitly encoded into the Static Knowledge Base. The Reasoning System is designed more like a search engine, inspired by Orkin (Orkin 2012). In this paper, we will describe how basic reasoning tasks, such as selecting which unit to build, are covered by a semantic structure.

SEMANTIC UNDERSTANDING OF GAME MECHANICS

The Dynamic and Static Knowledge Bases are interlinked semantic nets, where the Dynamic Knowledge Base can contain relations to nodes of the Static Knowledge Base. The dynamic and static nature of the semantic net is transparent, meaning that at any point of time during a game session, the Reasoning System refers to the Knowledge Bases as just a single semantic net, with no obvious boundaries between the two categories. The structure of the semantic net is a simple one, just consisting of nodes, relations and attributes of relations. An example of a Space Marine unit instance linking to the respective nodes describing it is shown in Figure 2.



Figures 2: Semantic structure excerpt illustrating the links between the Dynamic and Static Knowledge base for a Space Marine unit. Weapons, damage, abilities, movement etc are not shown.

Key idea behind this approach is to store as much information about the domain - in case of an RTS about the game mechanics - in a data structure accessible to the Reasoning Systems. Our hypothesis is, that this will allow an AI to better react on dynamic changes to game mechanics, such as a cooperative scenario, where one player decides to take care of anti air units, largely altering the game mechanics for the remaining player, as they no longer have to care about this aspect of the build. To pursue this goal, we avoid hardcoding values into the scripts that the Reasoning System utilizes, but instead replace them with queries on the semantic structure. Further, we expect that having a semantic representation of the game state from an AIs perspective will help it forming statements in bidirectional communication with human allies.

REASONING SYSTEM

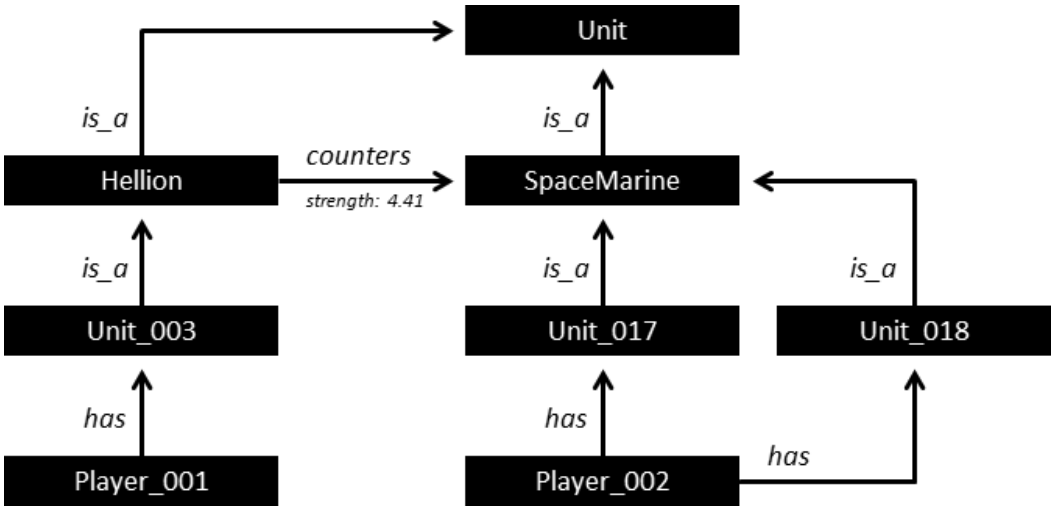
At its heart, the Reasoning System deployed in the prototype is a simple utility system, where each utility function is a chain of operations on the semantic structure. In StarCraft II, a dominant aspect of the gameplay is about countering enemy units. A counter to a unit is either explicit through game mechanics, such as the flamethrower of an Hellion dealing bonus damage against light units such as infantry. Further, counters can also be implicit, for example a Siege Tank having superior range and a blast radius, rendering it effective against groups of smaller units, such as infantry squads. Obviously, counters also depend on the resource costs of the respective units, as an AI will wants to use as few resources as possible to destroy the most hostile units.

To select units to build, the Reasoning System runs over the units expected to be in the hostile army. These consist of directly observed units as well as opponent modelling predicting which units might be fielded by opponents, depending on past observation. For opponent modelling, army predictors are used as described in (Stiegler 2014).

The utility of building a specific unit class then becomes the sum of all hostile units, weighted by their respective counter strength. Figure 3 shows an excerpt of the semantic structure with the basic nodes and relations relevant for counter-based unit selections. Player 1 controls one Hellion, whereas Player 2 controls two Space Marines. To decide which unit to build next, the Reasoning System of Player 1 would iterate over all available unit class nodes (such as the node labeled Hellion), for which research requirements are met. Research requirements are expressed with similar requires relations, not shown in Figure 3 for the sake of simplicity. For each unit class node, the Reasoning System will check if there is a path to unit nodes owned by hostile players that include counters relations. In this case, it would find that there are two such relations, one for the path from Hellion to Unit_017 and one for the path from Hellion to Unit_018. The final utility of building a Hellion would be 2*4.41=8.82. Note that, although the semantic net forms a directed graph, this path-finding operation interprets each relation as bidirectional. For pathfinding, a simple implementation of A* is used, close to the implementation in (Millington 2012). This exploits that both Static and Dynamic Knowledge are expressed in one single, coherent data structure, as a path can be found just through domain knowledge (for example that a Hellion is_a Unit), or through world state nodes (for example that producing a Hellion requires a certain Factory unit instance to be present).

The A* is slightly modified, allowing it to pass which relations are allowed to be used in order to find a path. In this example, only is_a and counters relations were allowed. If no counters relation, but a path to the opponent units is found, the utility per unit value defaults to 1.0, in all other cases, the utility per unit becomes 0. Utility scores are not normalized, so an arbitrary high value could be achieved. Yet, the action space of the AI is split into categories, so that build actions don't have to directly compete with other actions, such as attack or defense goals.

As units have different construction costs, their utility has to be normalized in regard to the effort to produce them. The implementation normalizes all utilities in respect to Minerals cost of 100. As Hellions cost just 100 Minerals, their utility stays at 2*4.41 * (100/100)=8.82, while the utility of building a Space Marine with a Minerals cost of 50 would become 2*1.0*(100/50)=4.0. The second resource used in StarCraft II, Vespene Gas, is mapped on minerals, where the mapping factor depends on the current gathering rate, the current stock and the chosen overall strategy.



Figures 3: Semantic structure excerpt illustrating simple counter relations.

COUNTERS RELATIONS

The above example used an implicit counter, as Hellions deal bonus damage against Space Marines. Yet, many counters are more complex. A sniper unit like a Ghost, for example, has very high counter values for units which have less health than one sniper shot deals damage. If so, a unit can be killed with a single shot, preventing the enemy from firing back. This renders a Ghost more effective against units with low base health or wounded units. Such a situation can also be expressed in semantic structures, for example by spanning a counters relation between the Ghost unit class node and a specific unit node, such as Unit_017 in the above example, instead of counters relations between unit classes. As each unit node encodes its own health value through a relation, these special cases can be resolved and respective relations can be added in a graph refinement step. Similar relations can be constructed for units that are grouped closely and splash damage dealt by siege weaponry such as Siege Tanks.

More complex unit interactions, like special abilities, require further constraints. The Banshee bomber, for example, can cloak itself, rendering it invisible unless an opponent fields a Detector. As invisible units cannot be shot at, this becomes a very powerful counter to many unit classes and situations. These more complex interactions are currently encoded by additional, tailored attributes to counter relations, but might be replaced by an extended version of the semantic structure, introducing new families of nodes dealing with timing. Note that typically, special abilities closely link to micromanagement, while still having a large impact on high-level strategies.

An important aspect of counters relations is their strength attribute. They are derived from a series of test games, in which predefined groups of units march at each other, utilizing different formations and unit states, such as research abilities or energy levels. Each test run records how much damage a specific unit inflicted per hostile unit class during its lifetime. Hostile units spawn as long as the test unit is still alive. A counters-value of 2.0 expresses that a unit kills (or deals damage equal to the health of) two of the countering units before it dies. These tests are automated and run 100 times per formation, setup and unit type and the arithmetic average of each combat result is used. Figure 4 shows some counters-values for units countering the Space Marine, resulting from a test run of 6300 tests. Note that the Space Marine countering itself is not exactly 1.0. This happens due to different positioning and the built-in pathfinding of StarCraft II, sometimes allowing a Space Marine to fire an extra volley at an enemy without being shot at. A threshold is introduced to avoid adding counters relations with values close to 1.0, unnecessarily enlarging the semantic net. The current threshold is 0.1.

Source	Target	Counters Value	Comment
Space Marine	Space Marine	1.01	
Reaper	Space Marine	1.72	
Marauder	Space Marine	3.90	
Ghost	Space Marine	2.88	(no snipe)
Ghost	Space Marine	41.62	(with snipe)
Hellion	Space Marine	4.41	(no upgrade)
Hellion	Space Marine	5.70	(with upgrade)
Hellbat	Space Marine	5.11	

Figure 4: Some exemplare Counters values as derived from the automated test scenario.

DISCUSSION AND OUTLOOK

We have illustrated how semantic structures can be used to express domain knowledge (Static Knowledge) and the current game state (Dynamic Knowledge) with a coherent data structure. Further, we described how a simple reasoning algorithm, such as a utility system, can be combined with queries on the semantic structure to access the knowledge bases and do a complex decision, such as deciding which unit to build for a given game state.

While the current algorithm can select which units to build and similar high-level planning actions such as selecting an expansion, there are still many open questions in regard to Micro Management. Current plans are to adopt Goal Driven Autonomy (Weber 2010) for squads and single units. In the long run, our hypothesis is that Micro Management could also be covered by a vastly extended version of the Dynamic Knowledge Base and queries on the semantic structures originating from a simple decision algorithm such as a utility system for each single unit. Yet, such a semantic structures would grow significantly, a first experiment using hierarchical state machines for squads showed relation counts to be around 15 times higher if spatial relations, such as distance and formations are encoded. As the complexity of the operations on the semantic structure, such as the utility function described above, scales with relation counts, the computational effort for the AI would skyrocket, in particular as a lot more of these operations would be called if each single unit would deploy such a reasoning system. This might limit our approach to high-level planning, whereas Micro Management is better served with other algorithms.

The time it takes to construct the respective units, although important for gameplay, is not taken into account for calculating the utility of training a unit. We plan to cover this in a future development iteration by extending the utility system with a temporal planner.

REFERENCES

Browder, D. 2011. "The Game Design of STARCRAFT II: Designing an E-Sport". Talk at *Game Developers Conference*, http://www.gdcvault.com/play/1014488/The-Game-Design-of-STARCRAFT, accessed September 15th 2015.

Buro, M. and Furtak, T. 2003. "RTS games as test-bed for real-time AI research". In *Proceedings of the 7th Joint Conference on Information Science* (JCIS 2003), 481-484.

Buro, M. 2004. "Call for AI Research in RTS Games". In *AAAI Workshop on Challenges in Game AI*, 139-141.

Churchill D. and Buro M. 2011. "Build order optimization in StarCraft". In *Proceedings of AIIDE*, 14-19.

Hagelbäck, J. 2012. "Potential-field based navigation in starcraft". In *Computational Intelligence and Games* (CIG), 2012. 388-393.

Laird, J. E. and Duchi, J. C. 2001. "Creating human-like synthetic characters with multiple skill levels: a case study using the Soar Quakebot". *Ann Arbor*, 1001.

Lidén, L. 2002. "Artificial Stupidity: The Art of Intentional Mistakes". In Rabin, S. (ed.) *AI Game Programming Wisdom*, 41-48.

McGee, K. and Abraham, A. T. 2010. "Real-time team-mate AI in games: A definition, survey, & critique". In *proceedings of the Fifth International Conference on the Foundations of Digital Games*, 124-131.

Millington, I. and Funge, J. 2012. "Artificial intelligence for games". 215ff.

Orkin, J. 2012. "Data-Driven Digital Actors," Keynote at *Conference on Computational Intelligence and Games* (CIG), 2012.

Rabin, M. 1994. "A model of pre-game communication". In *Journal of Economic Theory*, 63(2), 370-391.

Synnaeve, G. and Bessiere, P. 2011. "A Bayesian model for RTS units control applied to StarCraft". In *Computational Intelligence and Games* (CIG), 190-196.

Yildirim, S. and Stene, S. B. 2008. "A survey on the need and use of AI in game agents". In *Proceedings of the 2008 Spring simulation multiconference*, 124-131.

Stiegler, A. and Livingstone, D. J. 2013. "AI and human player cooperation in RTS games". In *Proceedings of the 8th International Conference on the Foundations of Digital Games*, 449-450.

Stiegler, A. and Livingstone, D. J. 2013. "Cooperative AI in Real-Time Strategy Games". In *Proceedings of the GameOn Conference 2013*, 45-51.

Stiegler, A. and Livingstone, D. J. 2014. "Semantic Structures for RTS Army Prediction". In *Proceedings of the GameOn Conference 2014*, 65-69.

Togelius, J.; Preuss, M.; Beume, N.; Wessing, S.; Hagelback, J.; and Yannakakis, G. 2010. "Multiobjective exploration of the StarCraft map space". In *IEEE Symposium on Computational Intelligence and Games*, 265-272.

Waltz, D. L. 2014. "Semantic Structures (RLE Linguistics B: Grammar): Advances in Natural Language Processing (Vol. 23)". Routledge.

Weber, B.G.; Mateas, M.; and Jhala, A. 2010. "Applying goal-driven autonomy to StarCraft". In *Proceedings of the Sixth Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Yannakakis, G. N. 2012. "Game AI revisited". In *Proceedings of the 9th conference on Computing Frontiers*, 285-292.

DEVELOPING TRAINABLE BOTS FOR A MOBILE GAME OF TENNIS

Maxim Mozgovoy[1], Akane Yamada[1], and Iskander Umarov[2]

[1]The University of Aizu
Tsuruga, Ikki-machi, Aizuwakamatsu
Fukushima, 965-8580 Japan
{mozgovoy, m5191102}@u-aizu.ac.jp

[2]TruSoft Int'l, Inc.
204 37th Ave. N #133
St. Petersburg, FL 33704 USA
umarov@trusoft.com

KEYWORDS
Case-based reasoning, learning by observation, behavior capture.

ABSTRACT

While behavior patterns of AI-controlled videogame characters are typically designed manually, self-learning AI systems possess unique attractive features. They can be used to create distinct character personalities of different skill levels, and "train your own character" can be a major user-end feature of a game. In this paper we present our attempt to develop a self-learning AI system for a mobile game of tennis. We show that our AI agents can learn behavior patterns from user actions, and play accordingly in similar game situations, exhibiting playing skills comparable to skills of a trainer.

INTRODUCTION

According to the report by International Data Corporation, online multiplayer games already surpassed single-player games in terms of consumer spending and player commitment (Ward 2015). The competition among such games is high, so game designers have to introduce innovative gameplay elements to stay on the market.

Typically, in online multiplayer games people compete both with other people, and with AI-controlled bots, so the quality of AI system has a significant impact on the overall success of a product. For our ongoing project of an online mobile tennis game, we decided to concentrate the effors on the AI system that implements the elements, suggested in (Umarov and Mozgovoy 2014):

1. Complex, non-repetitive behavior of AI bots.
2. Distinct personalities of AI bots, exhibiting a variety of skill levels and playing styles.
3. "Train your own character" mode as an element of gameplay.

To achieve these goals, we employ a learning by observation-based approach to AI design, outlined in (Mozgovoy and Umarov 2011). As a result, we are planning to obtain AI agents that can learn from human players, and exhibit human-like behavior, comparable in terms of style and skill level to behavior of their trainers. Our current system shows promising results, and is already able to learn from human actions and play accordingly.

TENNIS GAME ENGINE

The game of tennis relies on a custom-designed game engine, developed with Unity3D (see Figure 1).

Figures 1: Tennis Game Engine

Each player in the game can perform actions of two types: run to the specified location and shoot the ball into the specified point on the court. The game physics is accurate (Lopukhov 2015), so, for example, if a player tries to send the ball coming at a high speed to a nearby point on the opponent's side of a court, a backspin shot will be performed, so the ball will fly high over the net. Since we want to favor tactical gameplay rather than arcade, the game engine will automatically steer the players towards optimal ball receiving points. Thus, a player only has to care about own character location while the ball is moving towards the opponent, and about the best target for the next shot.

AI DESIGN PRINCIPLES

The AI subsystem can operate in two distinct modes. In learning mode, it observes the actions of the specified player, and stores them in its knowledgebase. In acting mode, the AI subsystem uses its knowledgebase to retrieve the most suitable action for a given game world state upon request from the game engine.

AI knowledgebase is represented with a set of three graphs, where individual vertices correspond to different game situations, and edges correspond to player actions. The difference between the graphs is in the list of attributes used to identify a single game situation. In other words, each graph describes desired agent behavior in form of a finite state machine at more or less accurate levels of abstraction. Our current system setup is described in Table 1.

In learning mode, each player action triggers insertion of a new (Game situation, action) pair into each of three graphs. If a game situation with the same attributes already exists in a certain graph, the action will be connected to the existing game situation node.

Table 1: Attributes of a Game Situation

Graph	Attributes
most accurate (level 0)	Game state (serve / hit / receive / etc.)
	Player position*
	Ball target point*
	Player's intended shot target*
	Player movement destination point*
	Opponent position*
	Opponent intended shot target*
	Ball position*
average accuracy (level 1)	Game state (serve / hit / receive / etc.)
	Player position*
	Ball target point*
	Player movement destination point*
	Opponent position*
	Opponent intended shot target*
	Ball position**
least accurate (level 2)	Game state (serve / hit / receive / etc.)
	Player position**
	Ball target point**
	Opponent position**
*	*(x, y); inside a 10×10 grid*
**	*(x, y); inside a 5×5 grid*

Table 2: Sequence of Knowledgebase Queries

Query	Graph level	Require action adjacency	Extended range search on attributes
1	0	Yes	—
2	0	No	—
3	1	Yes	—
4	1	No	—
6	2	Yes	—
7	2	Yes	Player, opponent, and ball coordinates
8	2	No	—
9	2	No	Player, opponent, and ball coordinates

In acting mode, the AI subsystem performs a number of queries to the graphs in order to find the best possible action in the given game situation. We start with the most strict query to find a perfect match for the given game situation in the most accurate graph, and if no relevant actions are found, we relax search conditions.

Currently there are three ways to relax a query: a) search in a graph of a higher abstraction level; b) instead of a perfect match require a match within a given value range — this is helpful for numerical attributes, such as player coordinates; c) do not require that two subsequent actions are also adjacent in the game graph (and thus do not follow the same game strategy). We cannot rely on assumption that AI always finds a perfect match in the knowledgebase for any possible game situation, so certain capabilities for approximate search are necessary.

A fragment of an actual level 0 graph of a trained agent is shown in Figure 2. The sequence of search actions we currently use in acting mode is provided in Table 2.

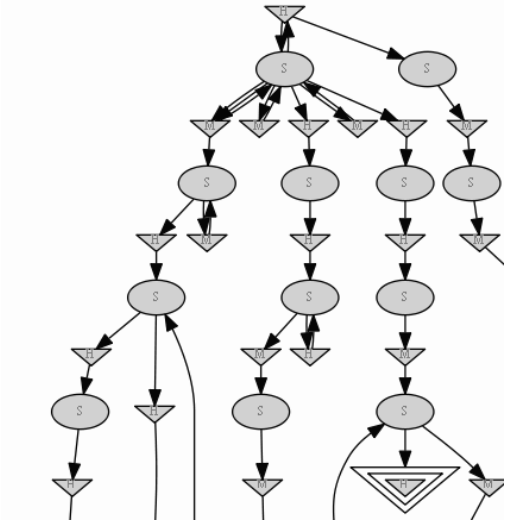


Figure 2: A Fragment of a Game Graph (Visualized with AT&T GraphViz)

EXPERIMENTAL SETUP

One of the goals of fine-tuning graph attributes and query conditions is to improve the overall playing experience for the users. In particular, we believe that the AI system should exhibit satisfactory behavior after 6-10 minutes of observations. Our experiments show that in average a player performs approximately 100 actions per minute (auto-steering movements, performed by the game engine, are also classified as actions), so a typical training session includes 600-1000 actions.

As a preliminary test of our system, we ran several game sessions between three players, as described in Table 3. An individual match ends when seven points are scored by either party. Obviously, the most skillful player is C, and the key component of his playing style is a winning strategic placement of a game character.

Next, we tested how the obtained AI agents play against each other by playing three matches for each pair of opponents. In all experiments the agents showed performance comparable to their trainers: A beats B with a score ranging from 7:0 to 7:3, and C beats A with the same outcome. Furthermore, C was able to beat B with a score 7:1 in all three test matches. This outcome is expected, since A is more skilled than B, but our training data did not contain games between B and C. While these results cannot yet prove that the obtained agents actually preserve acting style of human players, they show that the relative skill level of the opponents was preserved.

Table 3: Game Sessions

Session	Duration, sec	Players	Outcome
1	56	A vs. B	7:0
2	66	A vs. B	7:1
3	80	A vs. B	7:2
4	81	A vs. B	7:2
5	83	A vs. B	7:2
6	57	A vs. C	2:7
7	53	A vs. C	0:7
8	47	A vs. C	0:7

CONCLUSION

Game AI is a special topic for academic research, since computer games set such unusual requirements for AI systems as human-likeness, unpredictability, skill level adjustments, and *fun*, a core of any entertainment. Self-learning AI systems have a potential to address these challenges, since they can directly learn from human opponents, and thus exhibit distinct human-like behaviors of different skill levels. Surveys show that gamers actually prefer such AI opponents (Soni and Hingston 2008).

In this paper, we briefly discussed our work-in-progress AI systems for the game of tennis. While our AI did not reach a production-quality stage yet, it demonstrated the ability to learn and repeat behavioral patterns of human players

REFERENCES

Lopukhov, A. 2015. “Realistic Ball Motion Model for a Tennis Videogame”. *Proceedings of International Workshop on Applications in Information Technology (IWAIT), pp. 81-83*.

Mozgovoy, M. and Umarov, I. 2011. “Behavior Capture with Acting Graph: a Knowledgebase for a Game AI System.” *Lecture Notes in Computer Science*, vol. 7108, pp. 68-77.

Soni, B. and Hingston, P. 2008. “Bots Trained to Play Like a Human are More Fun”. *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 363-369.

Umarov, I. and Mozgovoy, M. 2014. “Creating Believable and Effective AI Agents for Games and Simulations: Reviews and Case Study”. *Contemporary Advancements in Information Technology Development in Dynamic Environments*, pp. 33-57.

Ward, L. 2015. “Gaming Spotlight, 1H15: How Online Multiplayer Is a Game Changer.” Document #256831, International Data Corporation.

A SIMPLE HYBRID ALGORITHM FOR IMPROVING TEAM SPORT AI

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

KEYWORDS

Adaptive AI, Fuzzy Logic, N-gram prediction, Team Sports Games.

ABSTRACT

In the very popular genre of team sports games defeating the opposing AI is the main focus of the gameplay experience. However the overall quality of these games is significantly damaged because, in a lot of cases, the opposition is prone to mistakes or vulnerable to exploitation. This paper introduces an AI system which overcomes this failing through the addition of simple adaptive learning and prediction algorithms to a basic ice hockey defence. The paper shows that improvements can be made to the gameplay experience without overly increasing the implementation complexity of the system or negatively affecting its performance. The created defensive system detects patterns in the offensive tactics used against it and changes elements of its reaction accordingly; effectively adapting to attempted exploitation of repeated tactics. This is achieved using a fuzzy inference system that tracks player movement, which greatly improves variation of defender positioning, alongside an N-gram pattern recognition-based algorithm that predicts the next action of the attacking player. Analysis of implementation complexity and execution overhead shows that these techniques are not prohibitively expensive in either respect, and are therefore appropriate for use in games.

INTRODUCTION

Artificial intelligence (AI) and video games have always been intrinsically linked. From providing very basic control of enemy characters in the early days of arcade games (the classic example being the ghosts in *Pac-Man* (Namco 1980), to the creation of complex systems that model the behaviour of realistic human characters in more recent titles.

As the games industry has grown over the years, so too have players' expectations of the perceived level of intelligence and realism exhibited by the enemies they now face. Some in the industry have gone as far as saying that "high-quality game AI has become an important selling point of computer games in recent years" (Tan, Tan and Tay 2011). Conversely, publishing a game that features obviously bad or broken AI is now a sure-fire way to draw harsh criticism from both consumers and the media.

Whereas some effort has been taken to address the imbalance between graphic fidelity and NPC 'intelligence' in RPG and the like, the same effort does not appear to have been taken when designing team sport games. It is still the case that once the player has established a specific strategy to defeat the opposition, this tactic will always work and the opposition are unable to learn or adapt to counter these moves. This significantly limits the replayability and shelf life of the game.

There are many existing AI algorithms capable of incorporating an element of learning/prediction that would be appropriate (Millington 2006). However, the method does not necessarily need to be complex. Decision trees and FSM-based systems form effective frameworks that can be adapted and augmented in various ways to exhibit the desired characteristics. The key issue then is selecting the techniques that are most appropriate in terms of code and implementation complexity; while also achieving the desired adaptive effect in the given situation.

ADAPTIVE AI

The goal of this project was to create a defensive system that would detect the use of repeated tactics, and react to this attempted exploitation in a behavioural way. For this purpose, adaptive AI can be defined as any algorithm which takes relevant data from the player's actions and changes the behaviour of the AI system in an appropriate way. In this game-specific context there is no definite solution due to the constantly changing nature of the desired gameplay; the goal is to simply improve the AI system in a way that allows it to be less rigid.

METHODOLOGY

The first step of developing the simple, adaptive AI was to build a basic ice hockey defence simulation. This system acts as a framework upon which the desired features are implemented separately to allow for comparison of results. It also enables access to necessary input data for the desired algorithms as well as application of their outputs to the defending agents. All adaptive functionality is built on top of this baseline application. Additionally, all data acquisition and processing will occur during execution (online adaption) so that the performance of each algorithm can be analysed. The ice hockey game was built using the Unity3D engine (Unity Technologies 2013).

A decision tree was implemented for the base framework as there is only really one state that the defenders can be in; defending. This ensures that the decision tree is kept fairly short and easy to visualise and maintain. A FSM would make more sense in a full hockey simulation where both teams can take possession of the puck, and where players can be on the ice or sitting on the bench.

A basic ice hockey defence was designed and implemented. It features two defending players whose movement and actions are controlled by a decision tree. Two

attacking players have also been created, and can either be controlled directly by the player or via a choice of three scripted offensive plays that have been created to control them automatically. When the attacking player who possesses the puck enters the defensive zone (he must be the first attacking player to do so, otherwise it would be called offside), the decision tree was activated.

Trigger boxes are used to detect when the attacking players have entered the defensive end. A vector is drawn from the puck carrier to the goal and this vector is used to position one of the defending players between him and the net. If a second attacker (a passing target for the puck carrier) is also detected, a vector is drawn between the two attackers and used to position one of the defenders between them. These processes can be seen in Figure 1.

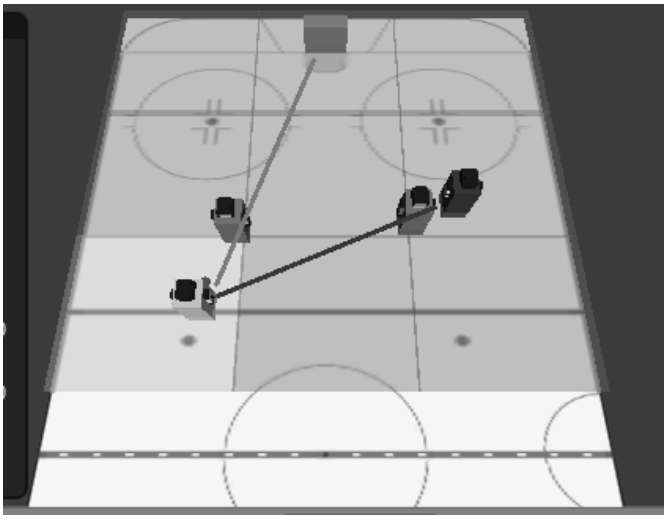


Figure 1: Detection Zones and Position Vectors in Created Application.

The leaves of the decision tree are "block goal" and "take action", which represent actions that the defending players can take. It is at these stages of the defensive process that the adaptive functionality is implemented.

Defender Positioning

The first adaptive aspect of the AI is in the positioning of the defender who challenges the puck carrier. A crucial element of this positioning is how far away from the puck carrier the defender will stay. This "offset" is a scalar value that is applied to the position vector of the defending player to move them closer to or further away from the puck carrier. This value is determined by a Fuzzy Inference System (FIS).

There are two inputs to the FIS. The first is the magnitude of the vector drawn from the puck carrier to the goal (scaled to between 0 and 1) i.e. the current distance from the net. Logically the closer an attacker is to the goal, the more likely they are to shoot. The second input is the "event heat" of the current detection zone (the shaded square area in Figure 1). The event heat is measured separately for each zone, again as a value between 0 and 1. Every time a shot or pass occurs in a zone, a weighted value is added to that zone's heat and another value subtracted from that of other zones (the exact values for which have been reached through an iterative process of trial and error). In this way, repeated actions in the same zone raise its heat very quickly, and the heat of other zones will reduce more slowly. This allows the system to adapt very quickly to repetition, while maintaining some memory of previous choices.

Fuzzy Inference System

The initial rule-base for the FIS took the inputs detailed above and combined them as shown in Table 1, the outputs relating to the amount of offset. The rules were initially set up symmetrically, balanced equally between both inputs and the outputs.

Table 1: Initial Rules of the FIS

	Distance		
Event Heat	Low	Medium	High
High	Low	Low	Medium
Medium	Low	Medium	High
Low	Medium	High	High

Though this seemed like a sensible approach, when applied to the simulation the resulting behaviour appeared rather unresponsive. The puck carrier was allowed far too much space when near the goal and in high-scoring zones that were not very close the net. To balance the defence in a more aggressive way, the rules of the system were changed to those in Table 2.

Table 2: Final Rules of the FIS.

	Distance		
Event Heat	Low	Medium	High
High	Low	Low	Low
Medium	Low	Medium	Medium
Low	Low	Medium	High

As shown, when the distance to the goal is low and the event heat of the current zone is high, the offset output by the system is low. However, the offset should also be low whenever the event heat is high or the distance is low.

The fuzzy set for the input 'Distance' is shown in Figure 2. The fuzzy set for 'Event Heat' is the same. The domain of each membership function was reached through experimentation both in MATLAB (MathWorks 2014) and using the application. Triangular shapes were chosen for the sake of simplicity; ensuring that fuzzifying the input values was as efficient as possible. The output fuzzy set 'Offset' (Figure 3) also has triangular membership functions. In addition to this, they do not overlap to simplify the defuzzification process.

The FIS uses the centroid method of defuzzification to generate a crisp numerical value from the fuzzy output that is calculated. Though far from the simplest method of defuzzification, the centroid method is one that gives the most variation of output values (Nurcahyo, Shamsuddin and Alias 2003). Since lack of variation is generally the problem with crisp rule-based implementations, it made sense to select the centroid method for this reason

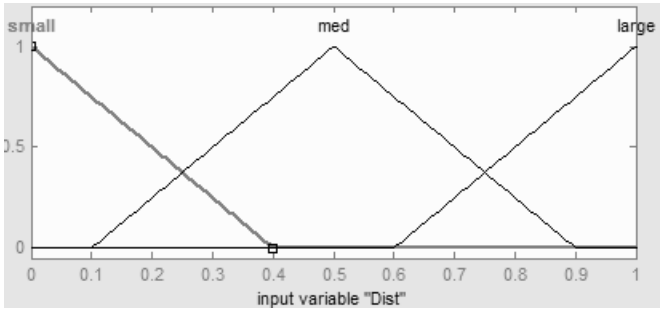


Figure 2: Input Fuzzy Set 'Distance'

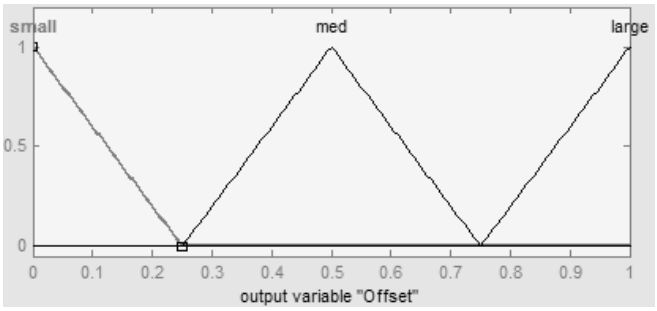


Figure 3: Output Fuzzy Set 'Offset'

Due to the shape of the membership functions, the implemented defuzzification gives an output offset value that is between 0.09 and 0.92. This value is then applied to the defender's offset vector via scalar multiplication, allowing for the gap between the two players to adapt to the given inputs.

Action Prediction

Defender positioning is only one component of an adaptive defence. The defender also has to decide whether the attacker is going to shoot or pass and act accordingly. The defensive end has been split in to 9 detection zones, shown in Figure 4. More than one detection zone can be active at any given time. During play each detection zone stores the number of shots and passes that have occurred and then stores the previous ten events as an ordered string of Char objects.

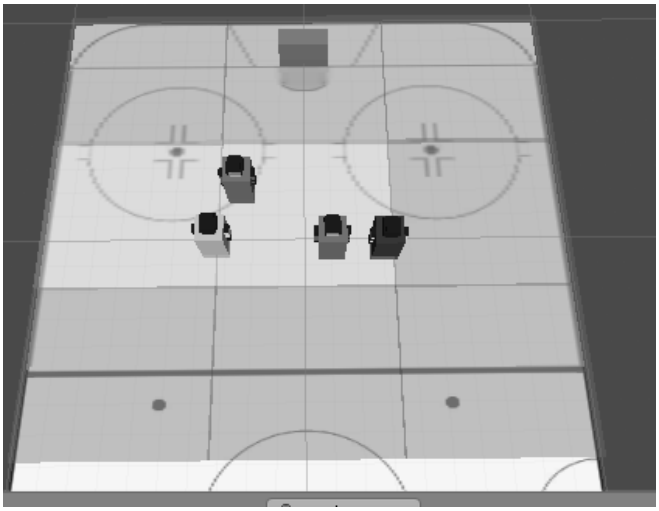


Figure 4: Detection Zones in the Defensive End.

N-gram Pattern Recognition

The action prediction implemented in the application is a 3-gram string-matching algorithm (Muise. et al. 2009). A shot is stored as an 'S', and a pass as a 'P'. An example event history from the completed application is shown in Figure 5. When an event is likely to occur (event heat > 0.5), the history of the current zone will be analysed. In this 3-gram method, the algorithm takes the last 2 events (P and S, in this case) and searches the history for instances of this pattern. The event that follows this pattern most often (another pass, in the above case) is then chosen to be the next predicted action. However, if the previous two actions have not occurred in that order before, the algorithm will be unable to

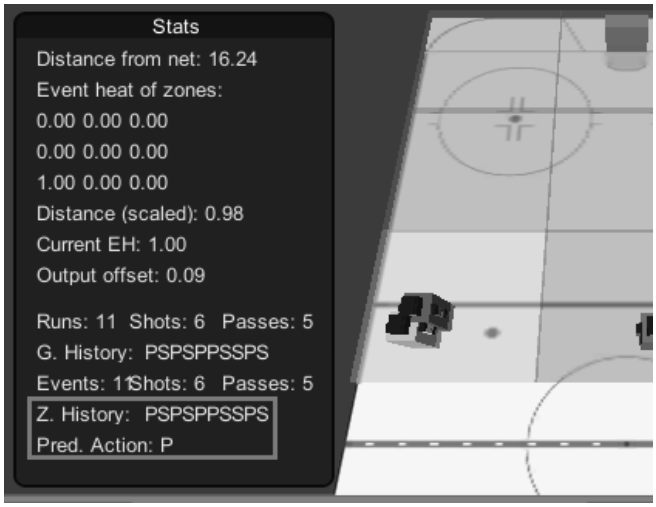


Figure 5: Event History of the Current Active Zone.

predict what the next action will be. For this reason, a final piece of work was carried out in this area.

Probability and N-gram Combination

A combined action prediction system was created. It simply carries out the previously described n-gram string-matching, and if no event can be predicted by that method, reverts to a probabilistic approach. When the event heat of the current active area is above 0.5, the total number of passes and shots for that area is compared, and the one with the larger volume is predicted to be the next action chosen by the player. If they are equal, the defence will play safe and predict a shot.

EVALUATION

Defender Positioning

Presented below are the graphed results of the FIS controller compared to a simple Crisp Rule Based System using the same rule base. One run with no event heat present (Figure 6), one with medium event heat present in the first zone (Figure 7) and a final run with high event heat in zone 1 (Figure 8) have been simulated. It should be noted that the sudden drops shown at the start of the second and third graphs are due to the puck carrier first entering the defensive zone.

As expected, the output from the crisp rule-based system is distinctly rigid and unvaried. The FIS offset values show a much greater degree of variance, resulting in a greater range of output values. Both systems seem to react to the combination of inputs in similar ways, suggesting that they both allow for a similar degree of adaption; though this is to be expected since they make use of the same rule set.

Action Prediction

Each of the three methods of action prediction (probabilistic, pattern recognition and combination) have been presented with a set of defined historical data. Table 3, shows the action predicted by each when exposed to the given string of event history. 'S' denotes a shot, 'P' a pass and an 'E' signifies that no action could be predicted. Though the pattern recognition method could handle this in a number of ways (default to predicting a shot, carry out previous chosen action, choose one at random, etc.), this

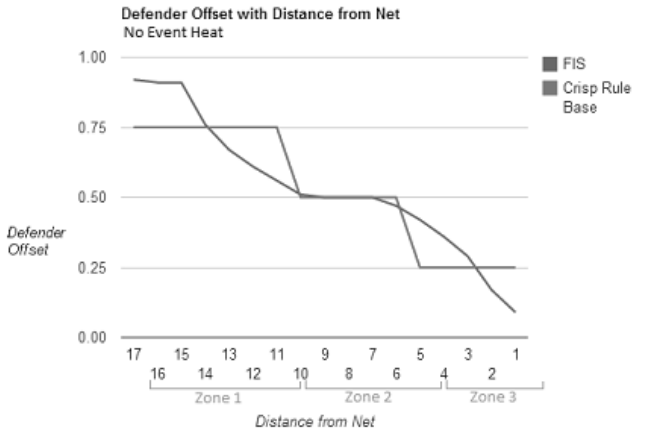


Figure 6: Results with no Event Heat and Decreasing Distance

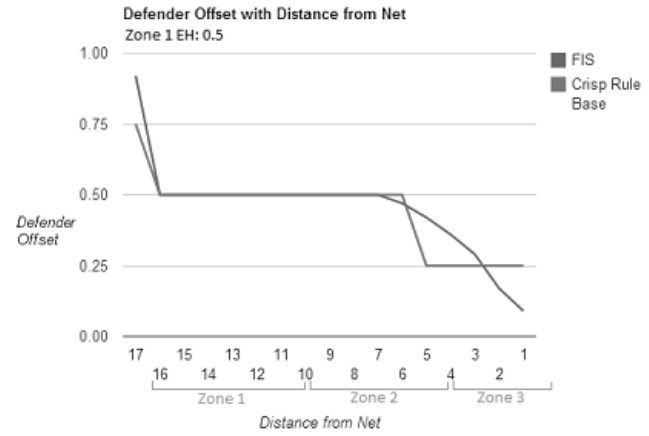


Figure 7: Results with Medium Event Heat in the First Zone

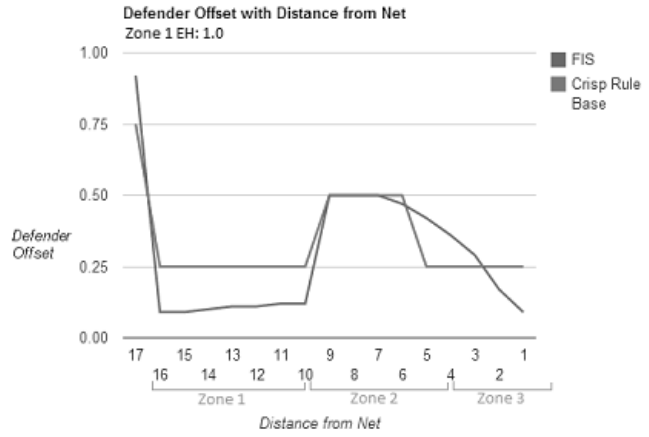


Figure 8: Results with High Event Heat in Zone 1

Table 3: Predicted Actions When Exposed to Historical Data

History	Desc.	Prob.	Pattern	Comb.
SSSSSSSSS	10 S	S	S	S
SSSSSSPPP	7S, 3P	S	P	P
SPSPSPSPSP	5S, 5P	S	S	S
SSPSSPSSPS	7S, 3P	S	S	S
PSSSPSPSPS	6S, 4P	S	P	P
PPSPPSPPSS	6P, 4S	P	E	P
SSSSSSSSSP	9S, 1P	S	E	S

would not be a meaningful prediction and therefore has not been implemented for these results.

Complexity

The complexity of each adaptive algorithm is displayed in Table 4. For comparison, the basic decision tree-based defence logic is included at the top.

Table 4: Complexity of Implemented Techniques

Technique	Lines of code	CPU time (ms)
Decision tree defence	281	between 0.02 and 0.05
NC - Crisp rule-based	99	0.01
NC - FIS	438	0.01
AP - Probability	4	0.01
AP - Pattern Recog.	85	0.03
AP - Combination	89	0.03

DISCUSSION

The main issue with evaluating gameplay systems such as those created is that there is often no real scientific way of measuring success in this context. While it is easy to test whether the defender is close to the puck carrier when they should be, it is much harder to say how effective a prediction algorithm is when the next action the player will take cannot be known. For this reason, there is bound to be an inherent level of subjectivity in any analysis of most created game AI systems. That said, much effort has been made to avoid bias both in the implementation of each algorithm and now in their discussion.

The overall results of the project (in terms of what has been created) are generally positive. Multiple methods of creating each desired feature have been researched, designed, implemented and tested. A robust application has been created, which showcases different adaptive AI algorithms in a relevant team sports context.

As shown in the graphs for Defender Positioning, there is a clear difference in the amount of output variation given by the FIS in comparison with a Crisp Rule-Based System. The FIS produces a greater array of different offset values than the crisp rule base does, and covers a wider range of the given domain. Though the range of values given by the rule-based system is somewhat due to implementation choice, there is no way around the lack of variation that results from using such a system. For this reason, if variation in terms of resulting values is desired when implementing a numerical control system of this kind, the FIS is clearly the better choice. The increased variation given by the FIS will make it appear to react in a more natural, realistic way.

Another thing worth noting is that the created system is far too precise in nature, even with the stated increase in variation provided by the FIS. In a sport as fast-paced as ice hockey, it is entirely unnatural for defending players to always be exactly in the right position. For this reason, it would be a good idea to add some kind of constrained random adjustment to the blocking defender's position vector. This would serve to ensure that some shots would actually make it past them.

As for the Action Prediction the purely probabilistic method has the distinct advantage of always being able to return a meaningful prediction, as long as at least one event has occurred in the current zone. This means that it is rapid to detect when an action is being repeated. It does not need to wait for sufficient history data to analyse properly. The n-gram method relies on sufficient history data being present

for it to find any form of pattern. When repetition cannot be found, the purely pattern-based prediction is unable to give a relevant output. While this can easily be fixed by having it default to predicting a shot, it is still a definite weakness of the approach. However, combining the two techniques so that probability is used when a pattern cannot be found results in a very robust system that has the strengths of both approaches, with no obvious drawbacks.

Complexity

As Table 4 shows, the crisp rule-based system required only 99 lines of code to implement, whereas the FIS resulted in over 400. While neither system has been compressed aggressively in terms of code (readability and clarity to a new observer have been emphasised during development), this is a marked increase in implementation complexity and therefore time. The fact that the FIS is substantially longer than the entire baseline decision tree defence logic suggests that it may only be worth implementing if numerical variation is highly important in the game situation. If not, a simple rule-based approach may be entirely appropriate.

However, as evidenced in Table 4, there is no marked increase in performance overhead when a FIS is used instead of a crisp rule set. This is due to the fact that both systems essentially boil down to a set of logical AND operations; with the FIS requiring simple and efficient calculations on either side of these rules. There are no expensive memory operations, so performance overhead is minimal and not of concern for current generation hardware.

Unsurprisingly, the probabilistic action prediction is incredibly easy to implement. A simple comparison of two numbers has almost no implementation cost at all, as shown in Table 4. Though somewhat more complex in concept, the created n-gram prediction only seems costly to implement due to its comparison with the probability approach (85 lines of code vs 4). In context, even the combination of both to create a fairly robust and effective system is far less costly than the original decision tree-based defender control.

Performance-wise, there is a small but noticeable increase in the CPU time used by the pattern recognition. However, the resultant CPU usage is still incredibly small and most definitely not an issue on any form of modern processor.

Future Work

While it is clear that an adaptive system has been created that functions correctly, there has been no evaluation of whether players would actually *enjoy* tackling it. The next step in developing such a system would therefore be to carry out some form of survey-based analysis of whether the average player feels the created system is fair, balanced and actually effective at what it aims to do.

In order to carry out the above evaluation, it would be a good idea to extend the created application into a fully-fledged team-based ice hockey simulation. This would involve making minor changes to how the defence currently functions, as well as implementing some form of attacking AI as well. It is likely that this would be a significant undertaking, but one that is entirely necessary to fully evaluate the performance of the defensive system in a proper context.

A logical extension would be to try to port the created adaptive system to other, similar team sport games. Though the gameplay mechanics of other sports like football and basketball are very different, the core concepts of defending (effective positioning, shot blocking) are completely transferable. In this way the AI performance of not just ice hockey, but all similar team sports games could be improved to provide a better experience for the paying customer.

CONCLUSIONS

In conclusion, the development of this project has shown that intelligent and believable behaviour can be modelled with a combination of fairly simplistic techniques. With graphical improvements in games becoming less and less noticeable with each new generation of hardware, it would seem then that creating better and more engaging game AI is of the utmost importance. There is clearly room for improvement in the games industry's approach to AI development as a whole; this project alone demonstrates that existing rigid systems can be improved without massive costs in development or impacts to performance.

REFERENCES

Millington, I. 2006. *Artificial Intelligence for Games.* Morgan Kaufmann, San Francisco, C.A.

Muise, C. et al. 2009. "Exploiting N-gram analysis to predict operator sequences." In: *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*.

Namco *Pac-Man*. 1980. [arcade]. Arcade. Namco.

Nurcahyo, G. Shamsuddin, S. and Alias, R. 2003. "Selection of Defuzzification Method to Obtain Crisp Value for rRpresenting Uncertain Data in a Modified Sweep Algorithm." *Journal of Computer Science & Technology*. 3(2). 22-28.

Tan, C., Tan, K. and Tay, A. 2011. "Dynamic Game Difficulty Scaling Using Adaptive Behavior-based AI." *IEEE Transactions on Computational Intelligence and AI in Games.* 3(4), 289-301.

WEB REFERENCES

MathWorks. 2014. *What is Sugeno-type fuzzy inference?* [online]. Available from: http://www.mathworks.co.uk/help/fuzzy/what-is-sugeno-type-fuzzy-inference.html [Accessed 30 April 2014].

Unity Technologies. 2013. *Unity*. [software]. Version 4.3.2. Available online, from: https://unity3d.com/unity/download/archive

BIOGRAPHIES

DAVID KING is a lecturer in Maths and Artificial Intelligence at Abertay University teaching on the Computer Games Technology and Computer Games Application Development Programmes.

DAVID EDWARDS Graduated with First Class Honours in Computer Games Technology from Abertay University in 2014 and is now a programmer for Gameplay, AI & User Experience.

INTELLIGENT AGENTS

PROPOSING AN INTELLIGENT AGENT FOR THE FOUR-SIDED DOMINOES GAME USING THE EXPECTIMINIMAX ALGORITHM

Endrews Silva
Marly Costa
Nirvana Antonio
Cicero Costa Filho
Technological and Information Center
Federal University of Amazonas
Av. General Rodrigo Otavio Jordão Ramos, 3000
Aleixo, Manaus, AM, Brazil. 69077-000
E-mail: cffcfilho@gmail.com

KEYWORDS
Four-sided Dominoes Game; Imperfect Information; Non-Deterministic; *Expectiminimax*.

ABSTRACT

This paper presents an intelligent agent to play the four-sided dominoes game, typically played in Amazonas state, Brazil. In this work, a version of the double-6 dominoes game is used, played by four players, forming two pairs. This game is of imperfect information and is non-deterministic, if we consider having pieces for any valid side as a chance event. The proposed agent is based on the *expectiminimax* technique. The methodology used for finding the best choice includes a depth-limited search with phase-related search, and a chance model based on the hidden pieces. We propose and test four strategies with different depths in the phases of the match, and tested each one against a pair playing with a basic strategy and a best strategy obtained by a genetic algorithm. In 10 simulations of 5,000 matches, the best strategy of this study obtained 72.04% of wins against a basic strategy and 58.34 % against the best strategy obtained by a genetic algorithm.

INTRODUCTION

The game of dominoes is comprised of rectangular "stones" divided into two halves. Each half has engraved dots, representing numbers, varying between 0 and N. Depending on *N* value, there are different versions of the game of dominoes. In Brazil, the most popular version of dominoes has two ends and *N=6*, resulting in 28 stones. Therefore, the stone with high numeration has six dots engraved on each half.

Dominoes is a non-deterministic game of imperfect information. This study uses probabilistic inferences to choose the best move. The need for probabilistic inference makes the task of developing an intelligent agent for dominoes games a complex one and suggests using the *expectiminimax* algorithm to determine the best move, (Michie, 1966) over the *minimax* (Neumann and Morgenstern 1944). Adding chance nodes that correspond to probabilistic events in the tree search implies an increase in its complexity by a factor multiplier O (b^d), where *b* is the number of chance events in each move and *d* is the search depth used to explore of the entire tree.

According to Russel and Norvig (2003), game trees are very complex, preventing the *minimax* and the *expectiminimax* algorithms from exploring the maximum depth of the game search tree (from the root node to the leaf nodes), when looking for the best move; therefore, some authors suggest dividing the game into phases and proceeding with a search into each phase. This technique is called a phase-related search (Smed and Hakonen 2006).

This study aims to develop an intelligent agent for the four-sided dominoes game and has the following secondary objectives:

- Proposing an intelligent agent for choosing the best move in a four-sided dominoes game using the expectiminimax algorithm;
- Using the phase-related search technique to explore the search tree of the four-sided dominoes game and evaluate the effect of changing the search depth in each phase over the number of wins of the *expectiminimax* algorithm;
- Proposing a probabilistic modeling of the chance events for the four-sided dominoes game.

.

RELATED WORKS

In the literature, some authors propose intelligent agents for the two-sided dominoes game. Developed for the 2-sided dominoes game, a study by Garza (2006) compares the performance of six game strategies against the basic strategy. The author proposes a static evaluation function with some terms. Each term incorporates the objectives of one strategy; nevertheless, the author provides few details about each term of the evaluation function. The strategies proposed in the study are not effective when tested against the basic strategy. The best result is 54% of wins of the selfish strategy against the basic strategy.

The study of Cruz et al. (2013) also addresses the two-sided dominoes game, obtaining better results than Garza (2006). The authors propose an evaluation function that incorporates a probabilistic calculus, where a Boltzmann exploration is employed, with the temperature of the Boltzmann equation related to the number of stones in the table. Of the four strategies proposed by the authors, one that aims to block the adversary's game obtains the best result: 65% of wins against the basic strategy.

Concerning the four-sided dominoes game, the study of Antonio et al. (2013) describes a detailed evaluation function for choosing the best move. The evaluation function terms represent two types of information. One term refers to the number of points obtained by a player's move. The other terms refers to the game state, based on the stones already placed on the game table. A different coefficient precedes each term and determines its relevance. A genetic algorithm optimizes the value of these coefficients in order to obtain a maximum number of wins. The authors propose different evaluation functions differing from one another by the number of terms and obtaining the best results, 69.18% of wins against the basic strategy, with a maximum number of terms of the evaluation function.

DOMINOES

The dominoes game version of this study comprises 28 stones. People in Amazonas state, in northern Brazil, usually play this game. The rules of this game are listed below. In addition, detailed rules can be seen at http://4-always.blogspot.com.br/2009/03/manaus-terra-do-domino.html. These rules assume two pairs playing the game:

1. Initially, each player receives seven stones. Partners of the same pair place one in front of the other. Figure 1 illustrates the four ends of a dominoes game and the player's positions on the table. Player P_0 is partners with player P_2, and players P_1 and P_3 are opponents;
2. A game match is comprised of several rounds. The objective of each player pair is to place all the stones in their hand onto the table;
3. In the first round of the game, the player with the double-six stone, 6-6, places it on the table. The following moves occur counterclockwise. Each player tries to connect a stone to one of the ends of a stone on the table with the same number of dots.
4. From the second round onwards the player who ends the previous round starts the second round with any double-N stone;
5. Each pair scores a single count. A match ends when one of the pairs reaches a count of 200 points at the end of a round;
6. In a move, a player can add 5, 10, …50 points to the count of its pair. This sequence shows the ways of accumulating points:
   - P_1: After a player's turn, if the sum of the dots in all four game ends is a multiple of 5, the player adds this multiple to the count of its pair. Figure 1 illustrates a situation where a player adds 10 points to the count of their pair;
   - P_2: When a player does not have stones to add to any of the four game ends, the adversary pair adds 20 points to its count. Figure 2 illustrates a game situation where the player shown does not have stones to move in any game end;
   - P_3: When a player, before moving a stone, declares that no players will move any stone after their move, and this actually occurs, they add 50 points to the count of their pair. This move is traditionally called a "rooster";
   - P_4: In the end of a round, if a player finishes the round with a double-N stone, they add 20 points to the count of their pair;
   - P_5: If a player finishes a round, the sum of dots in the stones owned by the adversary pair is rounded down to the nearest multiple of 5 and added to his pair count. Traditionally players call these points "garage". Figure 3 shows, a hypothetical scenario with the stones held by the adversary pair, when a round finishes. As there are 14 dots, the garage is 10 (the result of rounding down 14 to 10, the nearest multiple of 5).

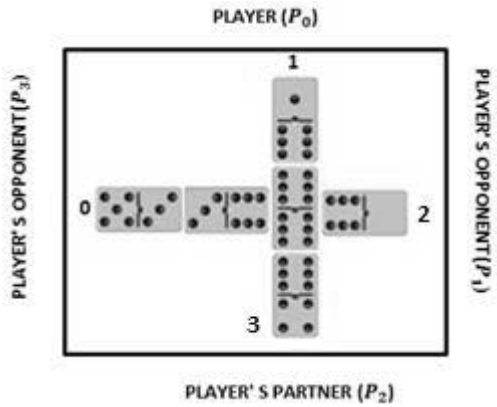


Figure 1: Example of a game situation where, after a player takes a turn, 10 points are added to the count of their pair.

Figure 2: Example of a game situation showing stones in the hand of a player, when there are no stones to add to any of the ends of the game.

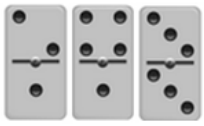


Figure 3: Example of a garage totaling 10 points.

METHODOLOGY

The depth search algorithm used in this study for the four-sided dominoes game uses a limited *expectiminimax* search. In this study, each search depth depends on the game phase. This study proposes dividing the dominoes game into three phases. In each phase, the search depth assumes a constant value.

The *expectiminimax* tree search for the four-sided dominoes game has three types of nodes: MAX, MIN and CHANCE. Figure 4 shows an example of a search tree with a depth of 2. A MAX node corresponds to one of the pair of players who uses the *expectiminimax* algorithm.

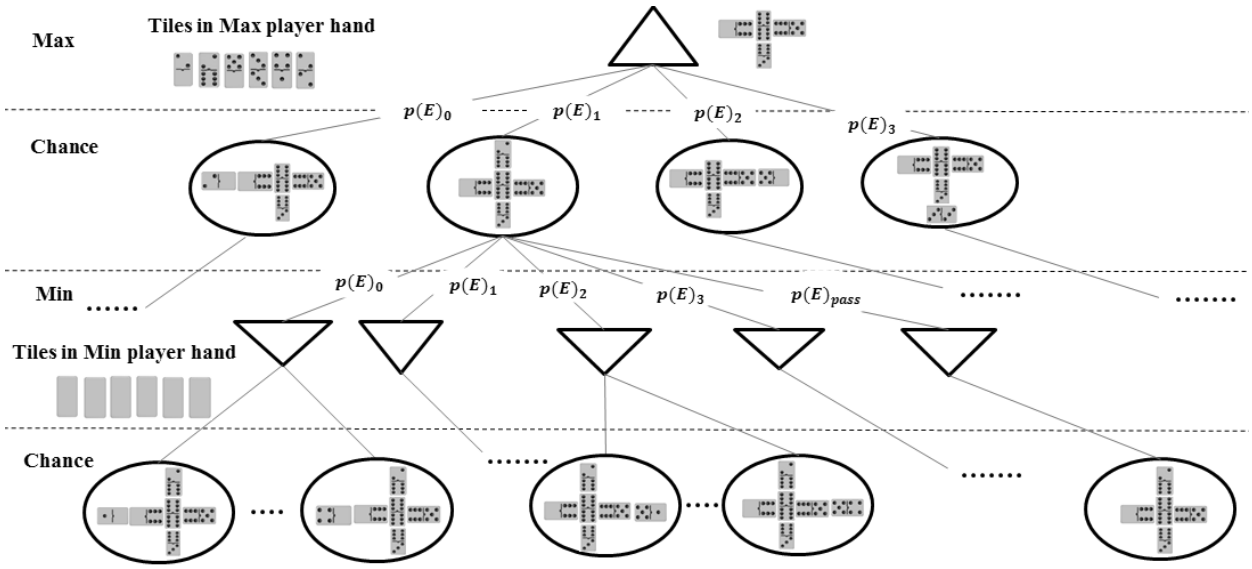


Figure 4: Search tree for the four-sided dominoes game with a depth of 2. $\boldsymbol{p(E)_0}$, $\boldsymbol{p(E)_1}$, $\boldsymbol{p(E)_2}$ and $\boldsymbol{p(E)_3}$ represent the probabilities of player "MAX" or "MIN" adding a stone to a game end 0, 1, 2 and 3. $\boldsymbol{p(E)_{no-stone}}$ represents the probability of players "MAX" or "MIN" having no stone to add to any end of the game.

In Figure 4, the upper triangle corresponds to a MAX player. A MIN node corresponds to one of the players of the adversary pair (the pair that does not use the *expectiminimax* algorithm). In Figure 4 the inverted triangles represent the MIN player. The circles represent a CHANCE node; A CHANCE node has several possible moves that are represented by probabilities: $p(E)_0$ – add a stone in to a game end 0; $p(E)_1$- add a stone in to a game end 1; $p(E)_2$- add a stone in to a game end 2; $p(E)_3$- move a stone in the game end 3; $p(E)_{no\ stone}$ – have no stone to add to any end of the game. Algorithm 1 shows the *expectiminimax* algorithm used for exploring this tree in the four-sided dominoes game.

Algorithm 1: *expectiminimax* algorithm for the four-sided dominoes game.

```
Function Expectiminimax (node, depth, State)
   1) IF depth=0
   2)  return ExpectPairScore-OpponentPairScore
   3) ELSE IF node is MIN node
   4)   BestScore ← + ∞
   5) FOR EACH possible action
   6)   Score=Expectiminimax(Chance,depth-1)
   7)   BestScore←min(BestScore,Score)
   8)   Undo move
   9) ElSE IF node is Max node
   10)  BestScore ← - ∞
   11) FOR EACH possible action
   12)  Score=Expectiminimax(Chance,depth-1)
   13)  BestScore←max(BestScore,Score)
   14)  Undo move
   15)ElSE node is a CHANCE event
   16)  BestScore ← 0
   17) FOR EACH available end of the table
   18)  Score=Expectiminimax(Child,depth-1)
   19)  BestScore ←BestScore+(Prob(i)*Score
   20) Return Bestscore
```

As in the original *expectiminimax* algorithm, the one just shown for the four-sided dominoes game is a recursive procedure that uses two auxiliary procedures: the move generator and the static evaluation. The *expectiminimax* algorithm must be provided with three parameters: an identification of the player who will take a turn (node), the depth search (depth) and the current configuration of the game (state).

Complexity of four-sided dominoes game tree

The number of leaf nodes usually measures the complexity of a game tree. For an imperfect information game, this number depends on the ramification factor of the tree (number of possible moves available for each player), the search depth and the number of options of a CHANCE node. For the four-sided dominoes game, the ramification factor does not present a linear behavior, as in chess and Stratego games, because its calculation depends on the number of stones owned by the players. It is possible to fix an upper limit to the ramification factor, considering the following worst-case scenario: in the second move of the game, the player who performs this move owns all stones with numeration equal to six in one of the sides (num_6) . This corresponds to a maximum value of the ramification factor, which is equal to six. Figure 5 shows these six possible moves.

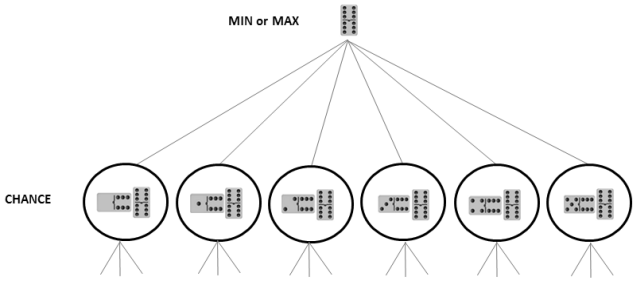


Figure 5: Maximum value for the ramification factor: a hypothetical situation where, in the second move of the game, the player who performs the move owns all stones with numeration $\boldsymbol{num_6}$

As stated before, the number of options of CHANCE nodes is equal to 5. Figure 6 shows these options. The last parameter needed to calculate the four-sided dominoes game

tree complexity is the search depth. This depth is equal to the maximum number of rounds of the game, which is 25.

According to Hauk (2004), equation (1) calculates the game tree complexity, C.

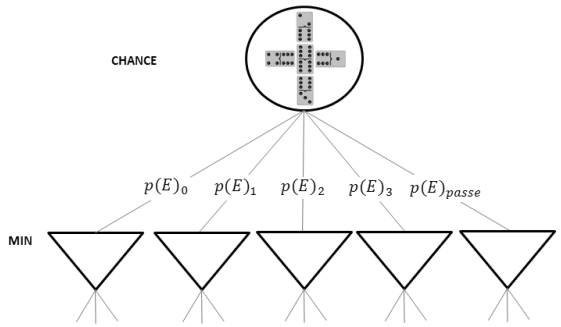


Figure 6: Ramification factor for the CHANCE nodes.

$$C = O\ ((RF)^p \times (CN)^p) \quad (1)$$

Where:

RF – Ramification factor of a MAX or MIN node - 6;
CN – Number of options of a CHANCE node -5;
p – Search depth -25.

Using the values previously determined results for complexity C:

$$C = O(6^{25} \times 5^{25}) \approx O(10^{36})$$

This is the maximum number of leaves that can exist in the dominoes game tree. For comparison, Table 1 shows the game complexity of some other games (ARTS, 2010). As shown, the four-sided dominoes game complexity is only greater than the game Connect-Four.

Table 1: Game tree complexity of some games (ARTS, 2010).

Game	**Game tree complexity**
Trail	10^{50}
Connect-Four	10^{21}
Othello	10^{58}
Chess	10^{123}
Chinese Checkers	10^{150}
Stratego	10^{535}
Go	10^{360}

Probabilistic modeling of CHANCE nodes

According to Walpole et al. (2011), if a sample space has N elements, with each one with the same probability, they all have the same probability *1/N*, and the probability of an event *A*, with *p* elements, is given by equation (2).

$$p = \frac{n}{N} \quad (2)$$

Calculating the probabilities $p(E)_i, 1 \leq i \leq 5$, associated with a CHANCE node requires a different calculation for the pair that uses the *expectiminimax* algorithm and for the adversary pair.

For the MAX node (a player of the pair who uses the expectiminimax algorithm), the calculus is very simple. As the stones are known by the algorithm, $p(E)_i = 1$ or $p(E)_i = 0$. For MIN node (a player of the adversary pair); nevertheless, the calculus is more complex and is explained in the sequence.

First, calculating the denominator of equation (2), N is explained. In this study, the stones not known by MAX are named hidden stones. Set_h comprises the set of hidden stones. The number of elements of Set_h is S_h. The sample space is formed by all hidden stone permutations: $S_h!$. Each permutation is an element of the sample space.

Second, calculating the numerator of equation (2), *n* is explained. The numeration of the game end *i* is num_i. The maximum number of stones with num_i, *t* owned by MIN player is given by equation (3).

$$t = minimum(S_1, S_{num_i}) \quad (3)$$

Where:
S_1 - number of stones owned by player MIN.
S_{num_i}– number of stones with numeration num_i in one of the sides. .

The number *n* in this study is calculated as a sum of T_k terms, where $1 < k < t$. T_k represents the number of elements of the sample space where player MIN owns *k* stones with numeration num_i in one of the sides. As shown below, for exemplifying how T_k is determined, this study assumes that:

P_0 – MAX player;
P_1 – MIN player;
P_2 –P_0 partner;
P_3 - P_1 partner;
$Set_h = \{p_{11}, p_{12}, p_{13}, p_{AB}, p_{CD}, p_{EF}, p_{GH}\}$ (7 stones);
$num_i = 1$;
Stones with numeration $num_i - p_{11}, p_{12}, p_{13}$, ($S_{num_i}$ =3);
3 stones of Set_h held by P_1 (S_1=*3*);
2 stones of Set_h held by P_2 (S_2=*2*);
2 stones of Set_h held by P_3 (S_3=*2*)

As S_{num_i} =3 and S_1=*3*, from equation (3), *t*=3. So, for calculating *n* we must add three terms: T_1, T_2 and T_3. Figure 7 shows some elements of the sample space with 1, 2 and 3 stones with numeration num_i owned by P_1. Before showing the expressions for T_1, T_2 and T_3, the following definitions are provided:

$$\alpha = S_h - S_{num_i} \quad (4)$$

$$\beta_1 = S_1 - 1, \beta_2 = S_1 - 2 \ldots \beta_k = S_1 - k \quad (5)$$

Where:

$k \in \{1, 2, \ldots, t\}$.

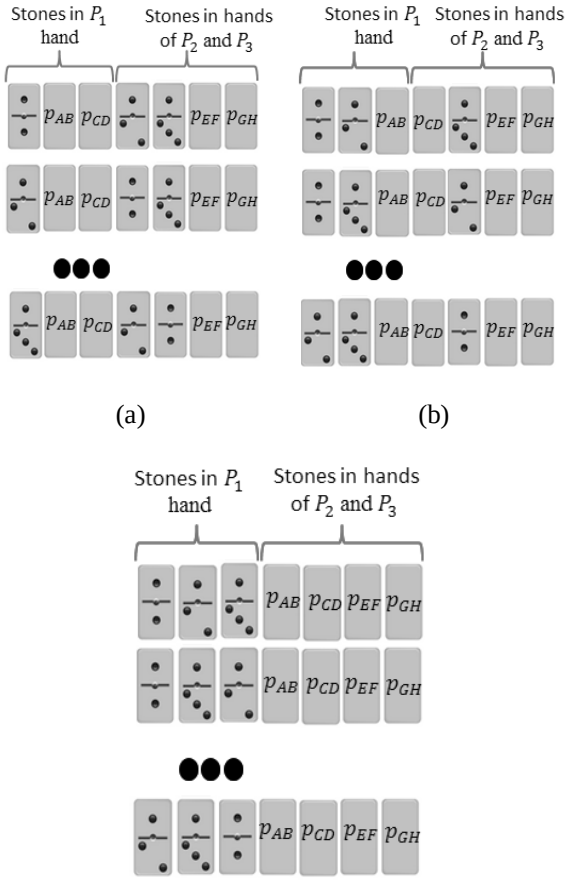


Figure 7: Elements of sample space: (a) examples of elements with 1 stone with numeration num_i owned by P_1; (b) examples of elements with 2 stones with numeration num_i owned by P_1; (c) examples of elements with 3 stones with numeration num_i owned by P_1.

This study calculates T_1, T_2 and T_3 using the following equations:

$$T_1 = A_{S_{num_i},1} \times C_{S_1,1} \times A_{\alpha,\beta_1} \times (S_h - S_1)! \quad (6)$$

$$T_2 = A_{S_{num_i},2} \times C_{S_1,2} \times A_{\alpha,\beta_1} \times (S_h - S_1)! \quad (7)$$

$$T_3 = A_{S_{num_i},3} \times C_{S_1,3} \times A_{\alpha,\beta_1} \times (S_h - S_1)! \quad (8)$$

Where:

$A_{j,k}$ – Number of arrangements of j elements in groups of k elements.

$C_{j,k}$ – Number of combinations of j elements in groups of k elements.

In a general way:

$$T_k = A_{S_{num_i},t} \times C_{S_1,k} \times A_{\alpha,\beta_k} \times (S_h - S_1)! \quad (10)$$

For the game situation assumed in this study, and using equations (6), (7) and (8), the calculus of T_1, T_2, T_3, n and $p(E)_i$ results: $T_1 = A_{3,1} \times C_{3,1} \times A_{4,2} \times 4! = 2592$, $T_2 = A_{3,2} \times C_{3,2} \times A_{4,1} \times 4! = 1728$ and $T_3 = A_{3,3} \times C_{3,3} \times A_{4,0} \times 4! = 144$. n= T_1+ T_2 + T_3 = 4464. The value of n=7!=5040 and $p(E)_i = \frac{4464}{5040} = 0{,}89$.

Game strategies

The depth search of the *expectiminimax* algorithm is limited according to the number of phases into which a game round is divided. In this study the four-sided dominoes round is divided into three phases: initial, middle, and final.

Aiming to define the size of each phase, this study conducts the following experiment: 5,000 matches between pairs using the basic strategy, resulting in 10,282 rounds. In each round, the number of moves was measured. Figure 8 shows the results of this experiment.

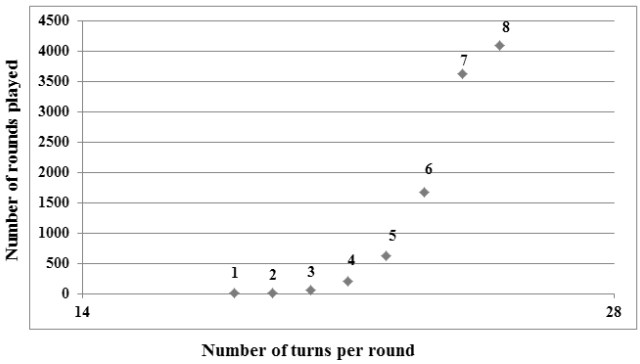


Figure 8: Number of rounds versus number of moves/round in an experiment with 5,000 matches (10282 rounds).

In Figure 8 the vertical axis represents the number of rounds, while the horizontal axis represents the number of moves/round. Point 1, for example, shows that only 3 rounds finished with 18 moves, while point 8 shows that 4,103 rounds finished with 25 moves.

Briesemeister (2009) estimated the number of moves/round of the game OnTop, simulating 1,400 matches. A mean value of 31.36 moves is reported.

The mean value of moves of a game can be calculated using equation (11).

$$\bar{p} = \frac{p_1 x_1 + p_2 x_2 + \cdots + p_n x_n}{p_1 + p_2 + \cdots + p_n} = \frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} p_i} \quad (11)$$

Where:

x_i – quantity of moves/round;

p_i – number of rounds with x_i moves in the experiment.

Applying the point values of Figure 8 in equation (11) results in a mean value of moves of 24.03. Therefore, there are approximately 24 moves/round. This study calculates the number of moves in each phase of the four-sided dominoes game dividing this value by 3, as shown in Table 2.

This study tests different search depths in each phase of a round and proposes four strategies, depending on values of these depths. Table 3 shows these strategies.

The first and second strategies use low values for depth search in the first and middle phases, while the third and fourth strategies use higher values for depth searches in these two phases. In all strategies, the depth search of the last phase was fixed in a maximum value of 9. For depth searches above 10, this study verifies that the *expectiminimax* algorithm the algorithm gets stuck in recursion.

Table 2: Phases for a match round of the four-sided dominoes game proposed in this study.

Match phase	Round
Initial	1~8
Middle	9~16
Final	>16

Table 3: Strategies defined for the four-sided dominoes game, varying the search depth in each round.

Strategies for *expectiminimax* agent	Depth search in round phase		
	Initial	Middle	Final
First	5	8	9
Second	8	5	9
Third	9	9	9
Fourth	10	10	9

RESULTS AND DISCUSSION

All the experiments were done on a laptop with an Intel Core i5 @ 2.5 GHz processor, with a Windows operating system 8.1. The simulations used the Eclipe IDE. This study divides the results into four groups.

The first group of results, with the aim of evaluating the number of wins of the *expectiminimax* algorithm against the basic strategy, proposes two simple experiments. These experiments change the depth search of initial and middle phase, while fixing the depth search of final phase. Both experiments performed 5,000 matches.

In the first experiment, the depth search of the initial and final phases were fixed at 5 and 9, respectively, and the length of middle phase ranged from 1 to 7. Figure 9 shows the number of wins of the pair that uses the *expectiminimax* algorithm (vertical axis) based on depth search of the middle phase (horizontal axis). The best result for the pair that uses the *expectiminimax* algorithm is 2985 wins. This occurs for a depth search of 7 in the middle phase.

In the second experiment, the depth search of the middle and final phases were fixed at 5 and 9, respectively, and the depth search of the initial phase ranged from 4 to 7. Figure 10 shows the number of wins for the pair that used the *expectiminimax* algorithm (vertical axis) based on depth search of the initial phase (horizontal axis). The best result for the pair that used the expectiminimax algorithm was 2,985 wins. This occurs for a depth search of 7 in the initial phase.

These two experiments suggest that the number of wins of a pair that uses the *expectiminimax* algorithm against a basic strategy is proportional to the depth search of the initial and middle phases.

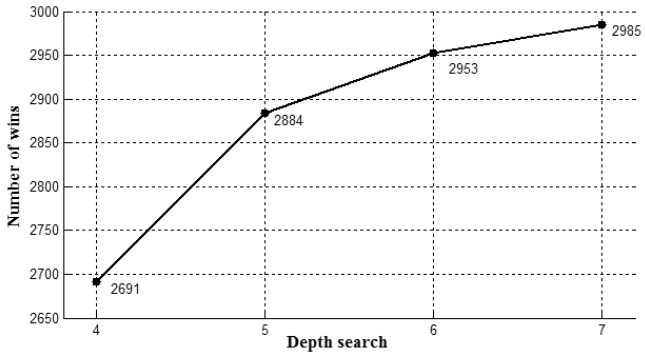


Figure 9: Number of wins for a pair using the *expectiminimax* algorithm against a basic strategy, with the following phase search depths: initial = 5; middle = 4, .., 7; final = 9.

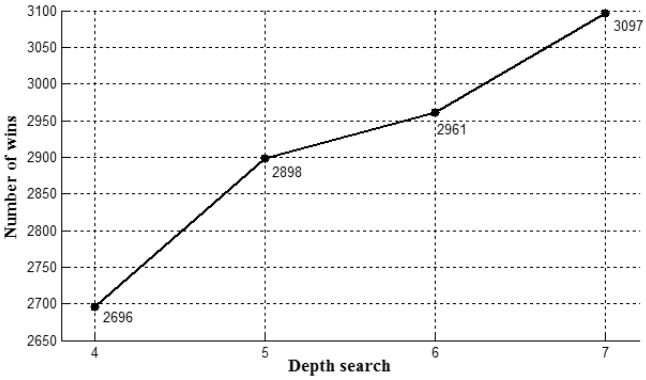


Figure 10: Number of wins for a pair using the *expectiminimax* algorithm against a basic strategy, with the following phase search depths: initial = 4…7; middle = 5; final = 9.

The second group of results evaluates the performance of the strategies defined in Table 3 against a basic strategy and against the best strategy defined by Antonio et al. (2013), hereafter-called strategy GA. Each test uses 10 simulations with 5,000 matches. The mean number of wins with the corresponding standard deviation are registered. This study proposes using the t-Student hypothesis test to evaluate the statistical significance of the results. The value of each t-Student test, T, is compared with a critical value t_c. The null hypothesis, $H_0 = 2{,}500$ wins, is rejected if $T \geq tc$ or $T \leq -tc$. A significance level of 99% is employed, with 9 freedom degrees. This result in a critical value $tc = 2.81$. Table 4 shows the results obtained in this step.

According to Table 4, all the strategies defined previously in this study obtained statistically significant results against the basic strategy, with mean victory values above 2,500. The fourth strategy obtained the largest number of wins and the lowest standard deviation, followed by strategy 3. Only strategies 3 and 4 obtained statistically significant results against the GA strategy. Again, the fourth strategy achieved the highest number of wins and the lowest standard deviation, followed by strategy 3.

In the third group of results, the aim is to evaluate the performance of each strategy playing against the other. Table 5 shows the results. Each test uses 10 simulations with 5000 matches, and the mean number of wins with the corresponding standard deviation are registered. Strategy 4 obtained statistically significant results against the other strategies. Perhaps, one possible reason is that this strategy uses deeper searches than other strategies in all four-sided dominoes game phases defined in Table 2.

Strategy 3 presents statistically significant results when playing against strategies 1 and 2. Strategy 2 provided a

statistically significant result only against strategy 1. Strategy 1 did not obtain statistically significant results against other strategies. Perhaps, the possible reason is that this strategy uses shallower searches than other strategies in all four-sided dominoes game phases defined in Table 2.

Table 4: Performance of each strategy defined in this study against the basic strategy and against the GA strategy (Antonio et al., 2013): mean number of wins in 10 simulations with 5,000 matches.

Strategies	Number of wins f		H_0: $\bar{f}$ = 2500
	$\bar{f}$	σ	T
First x Basic	3116.1	8.53	228
Second x Basic	3225.6	7.23	317
Third x Basic	3535.6	7.12	460
Fourth x Basic	3597.1	5.36	647
First x Ga	2331.6	5.21	-102.19
Second x Ga	2375.8	5.35	-73.41
Third x Ga	2795.1	5.92	157.76
Fourth x Ga	2909.3	5.08	254.87

Table 5: Performance of each strategy defined in this study playing against the other: mean number of wins in 10 simulations with 5,000 matches.

Strategies	Number of wins f		H_0: $\bar{f}$ = 2500
	$\bar{f}$	σ	T
First x Second	2378.6	5.58	-68.78
First x Third	2324.8	6.39	-86.69
First x Fourth	1952.2	7.24	-239.31
Second x First	2589.1	8.77	32.11
Second x Third	2383.8	8.22	-44.72
Second x Fourth	2101.8	8.93	-141.02
Third x First	2629.5	9.59	42.68
Third x Second	2591.8	7.18	40.45
Third x Fourth	2323.1	10.25	-54.60
Fourth x First	3017.7	8.45	193.82
Fourth x Second	2864.6	7.07	162.98
Fourth x Third	2645.6	5.87	78.40

In the fourth group of results, the aim is to evaluate the best performance (maximum number of wins) of each strategy defined in this study against the basic strategy and against the GA strategy. A total of 10 simulations with 5,000 matches were conducted and the maximum number of wins was registered. Table 6 presents the results. The statistical significance of results were evaluated using a Chi-square test, with a significance level of 99% and with 1 degree of freedom, resulting in a critical value of 10.83 ($\chi^2 = 10.83$).

Table 6**:** Performance of each strategy defined in this study against the basic strategy and against the GA strategy (Antonio et al., 2013): maximum number of wins in 10 simulations of 5,000 matches.

Strategies			Number of wins		χ^2 (chi-square)
Pair 1	x	Pair 2	Pair 1	Pair 2	Pair 1 x Pair 2
First	x	Basic	3127	1873	314.50
Second	x	Basic	3235	1765	432.18
Third	x	Basic	3543	1457	870.28
Fourth	x	Basic	3602	1398	971.52
First	x	GA	2342	2658	19.97
Second	x	GA	2381	2619	11.33
Third	x	GA	2805	2195	74.42
Fourth	x	GA	2917	2083	139.11

Table 6 shows that all strategies achieved statistically significant results against the basic strategy. The best result is 72.04% of wins. This result is higher than the one obtained by Antonio et al. (2013), which was 69.18%. Only strategies 3 and 4 defined in this study achieved statistically significant values against the best results presented by Antonio et al. (2013).

CONCLUSION

This study presents an intelligent agent for the four-sided dominoes game based on the *expectiminimax* algorithm. The study evaluates four different strategies. These strategies differ from one another by the depth search in each phase of a round.

The aim of defining strategies with different depth searches in each phase of a round evaluated its effect on the number of wins of a playing pair. Two strategies are defined with a shallow search depth in initial and middle round phases, while the other two are defined with a high search depth in initial and middle round phases.

The best results in terms of the maximum number of wins in 5,000 matches is 72.04%. These values are higher than those obtained by Antonio et al. (2013), 69.18%, for the four-sided dominoes game and higher than those obtained by Cruz et al. (2013), 65%, for the 2- sided dominoes game.

An important contribution of this study is the probabilistic modeling of the chance events of the four-sided dominoes game. Differing from backgammon, the calculated probability for a chance event changes with each round, because its value depends on the stones placed on the table.

For future works, we propose reducing the number of visited nodes in the tree search of the *expectiminimax* algorithm and improving the decision process of choosing the best move, through the following actions:

1) Employ the *Star1* and *Star2* algorithms, (Schadd et al. 2009) for minimizing the number of nodes of the search tree;
2) Employ evaluation functions that incorporate the current game state to decide the best move;
3) Employ different strategies for the adversary pair, to evaluate the robustness of the expectiminimax method proposed in this study.

ACKNOWLEDGEMENT

Part of the results presented in this paper were obtained through the project for research and human resources qualification, at the undergraduate and graduate level, in the areas of industrial automation, mobile software and Digital TV devices, sponsored by Samsung Eletronica da Amazonia Ltda, under the terms of Brazilian Federal Law number 8.387/91. We like to thank AcademicEnglishSolutions.com for revising the text

REFERENCES

Antonio, N.S.; Costa Filho, C.F.F.; Costa, M.G.F. 2013. "Optimization of an Evaluation Function of the Four-Sided Dominos Game Using a Genetic Algorithm," *Computational Intelligence and AI in Games, IEEE Transactions on* , vol.5, no.1, pp.33,43.

Arts, A. F. C. 2010. Competitive Play in Stratego. 56 p. M.Sc. thesis (Degree of Master of Science of Artificial Intelligence) - Faculty of Humanities and Sciences of Maastricht University, Maastricht, The Netherlands.

Briesemeister, R. 2009. Analysis and Implementation of the Game OnTop. 74 p. M.Sc. thesis (Degree of Master of Science of Artificial Intelligence) - Maastricht University, Dept. of Knowledge Engineering, Maastricht, Netherlands.

Cruz, A.R da .; Guimaraes, F.G.; Takahashi, R.H.C. 2013. Comparing Strategies to Play a 2-Sided Dominoes Game. In: *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence* (BRICS-CCI & CBIC), 2013 BRICS Congress on , p.310-316, 8-11.

Garza, A. G. de S. 2006. Evaluating Individual Player Strategies in a Collaborative Incomplete-Information Agent-Based Game Playing Environment. In: *Symposium on Computational Intelligence and Games*, p. 211-216.

Hauk, T. 2004. Search in trees with chance nodes. 85 p. M.Sc. thesis (Degree of Master of Science) - Computing science department, university of Alberta, Edmonton, Canada.

Myers, M.M. 2014. Outperforming Game Theoretic Play with Opponent Modeling in Two Player Dominoes. 88 p. M.Sc. thesis (Degree of Master of Science in Electrical Engineering) - Air Force Institute of Technology University, Ohio, United states.

Michie, D. 1966. Game-playing and game-learning automata. In: L. Fox (ed.), Advances in: *Programming and Non-Numerical Computation.* p. 183-200.

Neumann, J. V.; Morgenstern, O. 1944. Theory of Games and Economic Behavior. Princeton University Press.

Ruseel, S. and Norvig, P. 2003. A*rtificial Intelligence: A Modern Approach 2/e.* Englewood Cliffs, NJ, USA: Prentice Hall. .

Schadd, M.P.D.; Winands, M.H.M.; Uiterwijk J.W.H.M. 2009. "CHANCEPROBCUT: Forward pruning in chance nodes". In : *Computational Intelligence and Games*, 2009. CIG 2009. IEEE Symposium, pp.178, 185.

Smed, J. and Hakonen, H. 2006. Algorithms And Networking for Computer Games, UK, Chichester: John Wiley & Sons.

Walpole, R. E.; Myers, R. H,; Myers, S. L.; YE, K E. 2011. Probability and Statistics for Engineers and Scientists. 9th Edition, Pearson.

AUTHOR BIOGRAPHY

ENDREWS SZNYDER SOUZA DA SILVA studied electrical engineering at University of Amazonas Manaus, Amazonas, Brazil, and obtained his degrees in 2013. After graduating, he moved in 2013 to Technological and Information Center of Amazonas Federal University, Manaus, Brazil.

Aggressive versus Loud Virtual Agents

Investigating the influence of sound on the stress response in the use of virtual agents

Charlotte Gerritsen[1], and Willeke van Vught[2]

[1] Netherlands Institute for the Study of Crime and Law Enforcement
Amsterdam, the Netherlands
CGerritsen@nscr.nl

[2] Department of Computer Science
VU University Amsterdam
Amsterdam, the Netherlands
willekevanvught@hotmail.com

KEYWORDS

Serious Gaming; Simulation-based Training; Stress Response; Physiological Measurements; Experiments.

ABSTRACT

People can learn how to deal with negative situations by using serious games containing negative virtual characters, such as angry passengers in a public transport training scenario or hostile suspects in a police training scenario. Unfortunately little research has examined the effects of negative agents on the user. In this paper, an experiment is presented in which the effects of a visible and/or audible virtual agent are compared to the results of the use of a visible and/or audible virtual agent that is loud. During the experiment, heart rate and skin conductance level were measured and a questionnaire was used to measure the participants' subjective experience. The results indicate that a visible angry agent casuses a significantly higher level of arousal than a loud virtual agent.

INTRODUCTION

The use of intelligent virtual agents (IVAs) becomes more common in serious games such as virtual education, training and therapy (Rickel, 2001; Rizza et al, 2008). The interaction between human and IVAs has been the topic of many studies, but these studies most often focus on a positive attitude of the virtual agent. For example, Rickel (2001) uses animated agents as guides, mentors or teammates. These agents have a positive attitude towards the user and their goal is to educate the user.

However, the influence of negative stimuli from a virtual agent on the behaviour of the user has had little to no attention in literature. A negative agent can be helpful in serious games with the goal of teaching users to deal with negative situations and can be applied in various domains such as anti-bullying education or virtual training of aggression de-escalation. There seems to be a need for research into the effects of negative virtual agents on humans.

The VU University Amsterdam and the Netherlands Institute for the Study of Crime and Law Enforcement (NSCR) partnered up in the Simulation-based Training of Resilience in Emergencies and Stressful Situations (STRESS) project (stress.few.vu.nl). In this project the main goal is to develop a serious game that is able to analyse the decision making process and the causes of incorrect decisions under stressful circumstances. The focus is on the development of an electronic training environment for people with a public safety task. In this project, the influence from the negative virtual agent is important for the training environment. The users of the system are confronted with a virtual customer that is not satisfied and behaves aggressively towards the user. The users must learn to identify the behaviour and the appropriate way of dealing with that specific type of behaviour. To have a successful training environment it is important that the scenarios ressemble real-world settings. The situation should evoke a similar mental and physical reaction as it would in real life. To accomplish this, one of our research goals is to develop a negative virtual agent that evokes a similar response to a real life angry customer would.

In earlier work, we compared individuals' stress response when confronted with an aggressive virtual agent with the stress response evoked by an aggressive real person (Blankendaal et al., 2015). We found that in both conditions a substantial stress response was triggered, but some questions remained. In particular, it was not clear whether the stress response triggered in the virtual condition was caused by the virtual agent's aggressive behaviour or simply by the fact that agent raised its voice. This is a relevant question, since in reality people can also be scared by aggressive behaviour if the volume of the aggressor's voice is low. To address this question, the current work aims to investigate the difference in impact between an aggressive and a loud voice of a virtual agent.

This paper is organised as follows. First, the background of the current study is explained. Second, some background information in measuring and evoking stress is provided. Third, a description of the experiment and the analysis of the results are presented. Finally, this paper is concluded with a discussion of the findings from this study.

BACKGROUND

In our previous research (Blankendaal et al., 2015), we investigated the effect of an IVA with a negative attitude on a user. In that study, a comparison was made between an aggressive virtual agent and an aggressive real person. During the experiment, participants were read a story either by an actress or by an IVA (made to ressemble the actress)

on a computer screen. At a fixed moment in the story, the actress/IVA would get angry at the participant. She would claim that the person was not listening to the story and behaved rudely. During the experiment the listener's EDA (electrodermal activity or skin conductance) and heart rate were measured in order to register the physiological response. Once the experiment had ended, participants filled out a 10-point likert scale questionnaire to share their subjective experience.

The main conclusions were that both the aggressive actress and the aggressive agent evoked a significant stress response in the participant. However, the impact of the human aggression was larger than the virtual aggression.

The question remained as to whether or not the participant had a reaction to the aggression or to the sudden increase in volume of the voice during the aggressive episode. To test this, an additional experiment has been designed, presented in this paper, whereby the difference in response to a visible and/or audible aggressive virtual agent is compared to the response to a visible and /or audible virtual agent that only increases the volume of her voice.

MEASURING AND EVOKING STRESS

To determine whether the participant reacted to the aggression or to the change in volume, we measured the stress response. In this case the term stress response refers to the so-called fight-or-flight response (also known as acute stress). This response is evoked when a participant experiences arousal, such as from a threatening stimulus.

According to Prendinger et al. (2006), EDA is a good indicator for measuring arousal during an interaction between real humans and virtual agents. EDA measures change in exocrine sweat which is related to the sympathetic side of the autonomic nervous system (Figner and Murphy, 2011). Since arousal means activity of the sympathetic nerve system, EDA is often used for these measurements and is acknowledged as a good indicator of the overall level of arousal (Figner and Murphy, 2011). Heart rate is also considered a good indicator for the level of arousal because cardiovasculair activity is also influenced by the sympathetic nervous system.

In the current experiment, the potential arousal is caused by aggression portrayed by the virtual agent or by the change in volume of the voice. The main focus of this research is to examine participants' the reactions to these changes. The effect of emotions on physical and mental state has been examined in previous studies. Kleyweg (2012) discusses in his article on emotion theories that a sudden loud sound can cause a physical change that could be interpreted as fear (Kleyweg, 2012). According to Kleyweg (2012), this reaction should be considered a reflex. The changes inside the body do occur but disappear so fast that you do not actually experience the emotion. This could be true in our experiment. According to this theory, a physical change would be visible on the results of the physiological measurements, however the participant would not experience this emotion. To test this, the answers from the questionnaire are used.

In the experiment we distinguish four conditions. In condition A the virtual agent expresses anger by means of changed facial expression, altered volume of the voice and the use of 'angry' words. Condition B is almost identical to condition A but the participant cannot see the face of the agent. In condition C only the volume of the voice is altered; the participant can see the agent but her facial expression remains the same. Condition D is almost identical to condition C but the participant can only hear the agent.

In two of the four conditions the agent is not visible to the participant. This could make it more complicated to recognize emotion. Wallbott and Scherer (1989) investigated the influence of signals and channels on recognizing emotions (Wallbott and Scherer, 1989). They concluded that emotions are recognized relatively well when a video recording is used (62% are correctly recognized) and less well when only audio is used (47% are correctly recognized). This indicates that emotions in conditions A and C would be better recognized than emotions in conditions B and D. The questionnaires will be used to see whether or not the emotions were correctly recognized.

EXPERIMENT

The main question in the experiment is whether a visible and/or audible angry virtual agent causes a different physical and mental state than a visible and/or audible virtual agent who only raises her voice.

The answer to this question helps us to determine if IVAs are suitable for use in applications containing negative emotions. The settings of our previous experiment (Blankendaal et al, 2015) have been kept equal to make the results comparable.

Participants

Thirty-eight people participated in the experiment. All participants were students of the VU University Amsterdam and between 19 and 27 years old. These students were randomly divided over four different conditions:

- Condition A: 5 male and 5 female
- Condition B: 4 male and 5 female
- Condition C: 5 male and 5 female
- Condition D: 5 male and 4 female

Experimental Design

We used a virtual agent for the experiment. This agent was designed with Faceshift motion detection technology (www.faceshift.com), which is able to transfer the mimicry from a real human being to a virtual agent. The virtual agent told a story about the PC revolution (Hasselt et al, 2003). This story was read by the student who conducted the experiment and was recorded with a Microsoft Xbox 360 Kinect camera. The recorded clips were transferred to a virtual 3D character. To give the agent a humanlike appearance an agent was selected from http://www.rocketbox-libraries.com/. We choose a caucasion female between the age of 20-30 (see Figure 1) because this agent resembles the bachelor student and was congruent with her voice.

Fig. 1. Screenshot of the virtual agent used in the experiment.

The participants were divided into four conditions. In all conditions the first 7.35 minutes are the same, whereby the story is told by the agent without any exaggerated emotions. After these 7.35 minutes, two different situations can be distinguished:

1) The virtual agent gets mad at the participant. The facial expression of the agent is clearly angry, she raises her voice and says (freely translated from Dutch) "Just listen to me!! Can't you behave normally?! You can at least pretend that you are listenening or is that too hard, just focusing is not too much to ask is it?! This is ridiculous!! Were you raised by animals? Just look at yourself, you should be ashamed!"

2) The virtual agent raises the volume of her voice but continues telling the story. Her facial expression is the same as during the first 7.35 minutes.

As mentioned previously, the experiment contained four conditions:

Condition A
the virtual agent is visible and gets angry after 7.35 minutes

Condition B
the virtual agent is not visible and gets angry after 7.35 minutes

Condition C
the virtual agent is visible and only raises the volume of her voice after 7.35 minutes

Condition D
the virtual agent is not visible and only raises the volume of her voice after 7.35 minutes

The experiment has a between-group design. The groups in condition A and C each comprise 10 participants and the groups in conditions B and D each comprise 9 participants.
The independent variable is the experimental condition (A, B, C or D) and the dependent variables are EDA in micro siemens, heart rate per minute and the answers to the questionnaire.

Procedure

Prior to conducting the experiment, participants signed an informed consent form in which the experiment was described. However, the participants were told that they were participating in an experiment about *attention and memory*, since it was important that they did not know the exact topic of the experiment beforehand and did not expect the outburst of the agent.

Participants were told that they would listen to a story told by a virtual agent that was able to display emotions and that she may or may not be visible to them. Further, they were told that the agent was able to adapt her behaviour to the behaviour of the listener. We wanted them to believe that she was really angry at them and that it was not prefixed.

The participants were attached to a heart rate and an EDA measuring device and were told to sit in front of the computer screen and not to move because this would interfere with our measurements.

They could start the story by pressing the space bar. After the film started, the participants listened to the first five minutes of the story. Nothing strange happened. After these five minutes, they saw a black screen that stated that the connection was lost and that the story would continue in 30 seconds. A timer counted down from 30 to 0 and a new fragment was started. From this time point the measuring device started recording. The virtual agent continued for 2.35 minutes. At this time point the agent would get angry in condition A and B and in condition C and D she would raise her voice and continue telling the story. After 27 seconds in condition A and B and after 32 seconds in condition C and D a black screen appeared again with the message 'connection lost'. This lasted for 30 seconds and then the experiment ended.

Afterwards the participants filled out a 5-points likert scale questionnaire. The questionnaire had the following statements:

1) I listened carefully when the story was read;
2) I had the feeling the avatar was angry;
3) I believed the experiment was about attention and memory;
4) The avatar scared me when she was angry;
5) I felt personally addressed by her anger;
6) Her anger was believable;
7) The change in volume scared me;
8) I felt personally addressed by the avatar;
9) I believed the avatar was interactive and could react to my behaviour.

They could choose from the following options: totally disagree, disagree, neutral, agree, totally agree. Afterwards the participants had a conversation with the bachelor student that conducted the experiment so that they could explain their answers.

RESULTS

The results of the skin conductance (EDA, in micro siemens) and heart rate (per minute) measurements are shown in Figure 2 and 3.

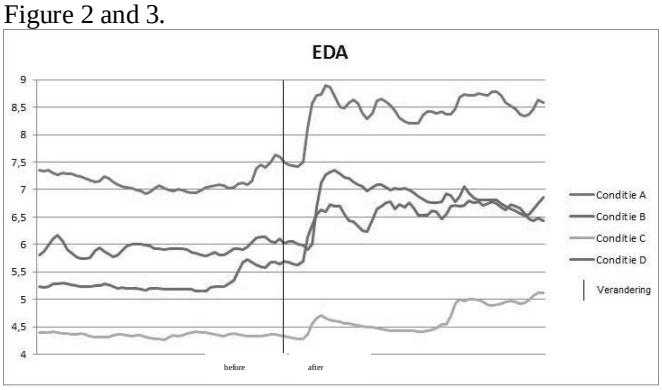


Fig. 2. EDA during the experiment, averaged over all participants per condition.

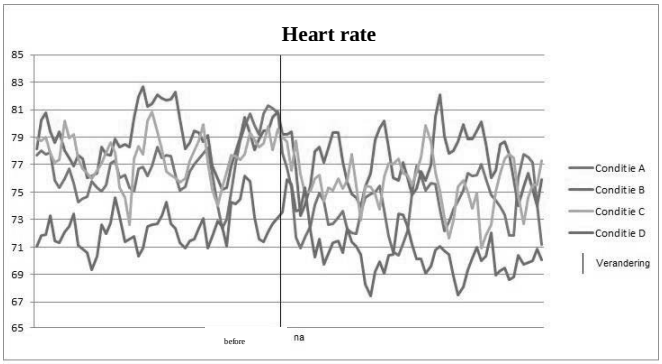


Fig. 3. Heart rate during the experiment averaged over all participants per condition.

Paired samples *t*-tests were conducted for each condition with the averages per condition as variables to test whether the stress response per condition was significant. Further, a one way ANOVA, with a posthoc test between groups with a Bonferroni correction method, was conducted.

The one way ANOVA was used to test the differences between the groups. First, the difference between groups before the outburst was tested to set a base for the EDA and heart rate, as the virtual agents acted the same in all four conditions. Second, the means after the outburst were compared between groups to see if the responses of the groups differ. The groups in each condition are exposed to different situations during the outburst. Finally, the means before and after the outburst are compared per group to test whether there is a significant difference in the stress response between groups. Only the 25 seconds before the outburst and the 25 seconds after the outburst have been used because the outburst itself lasts for approximately 25 seconds and we wanted to use comparable time frames.

EDA

The paired samples *t*-test showed that the reaction from the participant to the outburst is significant for conditions A, B and D: $p = 0.002$, $p = 0.002$ and $p = 0.037$, respectively. Only the reaction of the group in condition C is not significant ($p = 0.253$).

In the 25 seconds before the outburst, no significant difference between the groups is found using the one way ANOVA (p (A,C) = 0.16) and for all other groups ($p > 0.90$). In the 25 seconds after the outburst there is a significant difference between the groups in condition A and C ($p < 0.05$). No significant differences between the other groups has been found.

When comparing the stress response (average mean before and after the outburst per group) only a significant difference is found between conditions A and C ($p < 0.05$).

Finally, we tested whether there was a significant difference between the conditions when we used the 'connection lost' moment instead of the outburst. A paired samples t-test per condition was performed. There is a significant difference between EDA before and after the connection lost moment in condition C ($p<0.05$). In the other conditions no (extra) arousal was caused by the connection lost moment. After the connection lost moment, no significant differences between the conditions have been found.

Heart rate

The results of the heart rate measurements have been compared by using a paired sample t-test per group to test the difference between the average mean before and after the outburst of the virtual agent. Only the reaction in condition A is significant ($p<0.01$). Further, all heart rate levels seem to drop after the outburst. However, in conditions B, C and D this change is not significant.

Figure 3 depicts the mean heart rate per group. The ANOVA test over the 25 seconds before and after the outburst concludes that the changes are not significant.

Questionnaire

As mentioned earlier, the participants all filled out a questionnaire (5 points likert scale).

The answers are depicted in Figures 4 and 5. These answers have been analysed with an ANOVA with Bonferroni correction. A significant difference has been found in question 2 ("I had the feeling the avatar was angry") between conditions A and C ($p<0.01$) and between conditions B and C ($p<0.05$). No significant differences were found between the answers to the other questions.

Question 1 and 3 are more general questions and are not dependent on the different conditions. As can be seen in the graph (Figure 4), the answers to these questions are very similar across the groups. The answers to question 7 ("the change in volume scared me") show that participants in all conditions were scared by the sudden increase in volume. The participants felt somewhat personally addressed by the avatar (question 8) and did not really believe that the avatar was interactive (question 9).

Figure 5 shows the results for questions 4, 5 and 6. These questions were only answered if the participant answered question 2 ("I had the feeling that the avatar was angry") confirmatively. The results shown are the results of the participants that actually answered these questions (*n*A=10, *n*B=8, *n*C=4 en *n*D=7). Interestingly, all participants in condition A, and 8 out of 9 participants in condition B, recognized the anger. While no anger was portrayed in conditions C and D, 4 out of 10 and 7 out of 9, respectively, did recognize it as such. No significant differences were found. In question 4 ("The avatar scared me when she was angry"), a difference can be seen between the conditions with a visible agent (condition A and C) and the conditions without a visible agent (conditions B and D). The group in condition D felt more personally addressed than the other groups (question 5), and a difference can be found in question 6 ("her anger was believable") between the groups that dealt with an angry avatar compared to the groups dealing with the avatar that raised her volume.

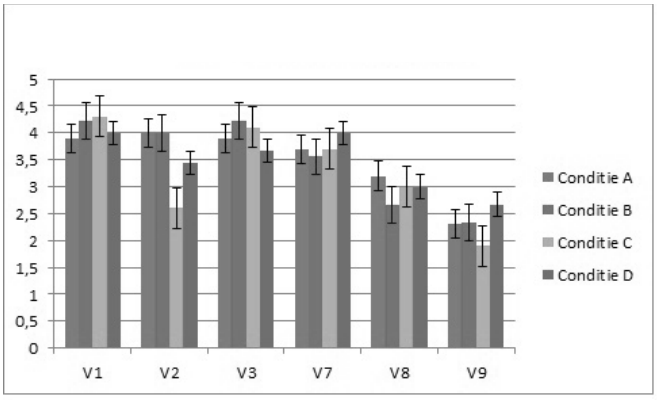


Fig. 4. Results from the questionnaire, filled out by all participants.

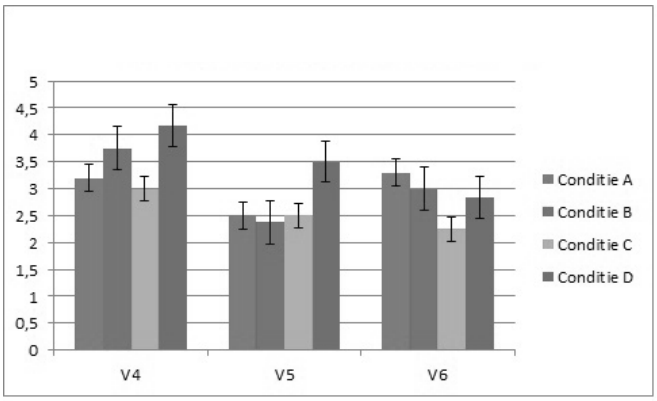


Fig. 5. Results from question 4, 5 and 6 (nA=10, nB=8, nC=4 en nD=7).

DISCUSSION

The results of the questionaire deomonstrate a significant difference in the experience of anger (question 2) between groups A and C. The participants in condition A clearly perceived the virtual agent as being angry while the participants in condition C perceived the agent as not being angry. Given that we wanted to test the physiological response to both situations it is important that both situations could be clearly distinguished.

The objective results show no distiction between the different groups before the outburst. A difference between group A and C can be seen in the graph but this is caused by an outlier in each group. After the outburst, there is a significant difference between the EDA of group A and C. This leads to the conclusion that the angry agent gave a stronger physiological response than the loud agent.
When the averaged means of both groups before and after were compared, we again found a significant difference. The behaviour of the angry agent lead to a higher stress response than the behaviour of the loud agent.

CONCLUSION

The main goal of the research presented in this paper is to see whether or not a visible and/or audible angry virtual agent causes a different physical and mental state than a visible and/or audible virtual agent that raises the volume of her voice. To test this, participants watched and/or listened to a virtual agent who was telling a story. In the middle of the story she would either get angry or raise her voice.

From the results we can conclude that the anger by the virtual agent can result in a stress response. This response was not caused only by the increase in volume. There is a significant difference is the EDA response between participants is condition A (see and hear angry agent) and condition C (see and hear agent that raises her voice). This difference is only visible in the results after the outburst. No significant difference between the groups was found before the outburst.

The level of stress response in the condition with a visible angry agent is high, while the stress response is lacking in the condition with a visible loud agent. The visibility of the emotions seems to be important for recognition. The significant difference found between condition A and C is missing in the audio conditions (B and D).

The fact that anger from a virtual agent causes a stress response indicates that emotions displayed by an agent can have an impact on people interacting with that agent. This is an indicator that IVAs can also be useful for serious games that focus on dealing with negative emotions, such as virtual training environments. In our previous research (Blankendaal et al., 2015), we concluded that an angry virtual agent is almost as scary as an angry human being. Extending this, we have have now found evidence that angry virtual agents are perceived as being more stressful than loud virtual agents.

ACKNOWLEDGEMENTS

This research was supported by funding from the National Initiative Brain and Cognition, coordinated by the Netherlands Organization for Scientific Research (NWO), under grant agreement No. 056-25-013.

The authors wish to thank Tibor Bosse, Jeroen de Man and Marco Otte for many fruitful discussions and their (technical) support during the project. Further, we would like to thank all students who participated in the experiments.

REFERENCES

Blankendaal, R., Bosse, T., Gerritsen, C., de Jong, T., & de Man, T. (2015). Are Aggressive Agents as Scary as Aggressive Humans? In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, ACM Digital Library, pp. 553-561.

Bosse, T. & Provoost, S. (2014). Towards Aggression De-escalation Training with Virtual Agents: A Computational Model. In Proceedings of the 16th Int. Conference on Human-Computer Interaction, HCI'14. Lecture Notes in Computer Science, Springer Verlag, pp. 375-387.

Cannon,W. (1932). Wisdom of the Body. United States: W.W. Norton & company

Chida, Y. & Hamer, M. (2008). Chronic psychosocial factors and acute physiological responses to laboratoryinduced stress in healthy populations: a quantitative review of 30 years of investigation. Psychological Bulletin, 134(6), pp 829-885.

Figner, B., & Murphy, R. O. (2011). Using skin conductance in judgment and decision making research. In Schulte-Mecklenbeck, M., Kuehberger, A. & Ranyard R. (Eds.), A handbook of process tracing methods for decision research (pp. 163-184). New York, NY: Psychology Press

Hasselt, L., Amsenga, J., & Meel A. (2003). De PC revolutie. VPRO.

Kleyweg, R. (2012). Emotietheorieen: Cognitief, noncognitief of beide? Erasmus Student Journal of Philosophy, 3, pp 46-57.

Prendinger, H., Becker, C., & Ishizuka, M. (2006). A study in users' physiological response to an empathic interface agent. International Journal of Humanoid Robotics, 03(3), pp. 371-391.

Rickel, J. (2001). Intelligent Virtual Agents for Education and Training: Opportunities and Challenges. In de Antonio, A., Aylett, R. & Ballin D. (eds.), Intelligent Virtual Agents, Lecture Notes in Computer Science, Springer Verlag, Vol. 2190, pp. 15-22.

Rizza, R., Reger, G., Gahm, G., Difede, J., & Rothbaum, B.O. (2008). Virtual Reality Exposure Therapy for Combat-Related PTSD. In Shiromani, P., Keane, T., & LeDoux, J. (eds.), Post-Traumatic Stress Disorder: Basic Science and Clinical Practice, Springer Verlag, pp. 375-399.

Wallbott, H.G. & Scherer, K.R. (1989). Cues and Channels in Emotion Recognition. Journal of Personality and Social Psychology, 51(4), pp 690-699.

Zoll, C., Enz, S., Schaub, H., Aylett, R., & Paiva, A. (2006). Fighting Bullying with the Help of Autonomous Agents in a Virtual School Environment. In Proceedings of the 7th International Conference on Cognitive Modelling (ECCM).

http://www.faceshift.com

http://www.rocketbox-libraries.com/

http://stress.few.vu.nl

PROCEDURAL PROGRAMMING

Procedural generation of collaborative puzzle-platform game levels

Benjamin van Arkel, Daniël Karavolos, and Anders Bouwer
Amsterdam University of Applied Sciences
Wibautstraat 2-4
1091 GM Amsterdam
The Netherlands

KEYWORDS
Game Design, Procedural Content Generation, Level Generation, Generative Grammars, Automated Game Design, Game Mechanics for Two-Player Collaboration

ABSTRACT

This paper addresses the procedural generation of levels for collaborative puzzle-platform games. To address this issue, we distinguish types of multiplayer interaction, focusing on two-player collaboration, and identify relevant game mechanics for a puzzle-platform game, addressing player movement, interaction with moving game objects, and physical interaction involving both players. These are further formalized as game design patterns. To test the feasibility of the approach, a level generator has been implemented based on a rule-based approach, using the existing tool called Ludoscope and a prototype game developed in the Unity game engine. The level generation procedure results in over 3.7 million possible playable level variations that can be generated automatically. Each of these levels encourages or even requires both players to engage in collaborative gameplay.

1. INTRODUCTION

Procedural content generation (PCG) is the algorithmic creation of content for games, such as assets, levels, worlds, and even whole games. PCG has been part of published computer games since the eighties. Prominent examples include Rogue, Diablo, and Minecraft, among others. Recently, PCG is receiving increasingly more academic attention (Togelius et al. 2011, Hendrikx et al. 2013, Shaker et al. 2014). PCG has been used for various reasons, including working around memory constraints in past decades when hardware was less powerful, increasing replayability by generating variation, making the game development process more efficient, exploring - and perhaps enlarging - the game design space, and formalizing the rules of game design.

Research on the application of PCG for platform games has focused on creating levels for the game Super Mario Bros (Shaker et al. 2011), with its typical obstacles, enemies, and straight levels. However, little work has been carried out on generating platformer levels for multiple players.

In this paper, we explore the potential of PCG to generate levels for puzzle-platform games that involve two-player interaction, in particular, collaboration. We will define game design patterns, which we will then translate to level segments which create gameplay situations featuring certain game mechanics. Similar to Dahlskog and Togelius (2013), we will let a generator combine these level segments to create a level. However, we will use generative grammars to generate the levels instead of evolutionary algorithms.

In section 2, we consider related work on procedural level generation. Section 3 identifies game mechanics that specifically address collaboration, and in section 4, we describe relevant game design patterns for these. Section 5 gives an overview of the implementation of the whole generation process. Section 6 offers several points for discussion and in section 7, we present our conclusions and ideas for future work.

2. RELATED WORK

Although many approaches to procedural level generation are tied to specific games, e.g. (Shaker et al. 2011, Smith et al. 2011, Shaker et al. 2013, Ferreira and Toledo 2014), there are some tools available that are more general in their approach. For example, LudoScope (Dormans 2011; 2012, Dormans and Leijnen 2013), is a tool that allows the transformation of design concepts about missions and space to concrete game levels for playable games, automatically, or semi-automatically in interaction with a human designer (Karavolos et al. 2015). It is based on the principles of model driven engineering and generative grammars. In order to generate levels for a specific game, the designer makes a model of the generation process, breaking it down into steps (modules) that can be executed separately. We will use this tool to generate the levels for our puzzle-platformer prototype.

There are several ways to characterize multiplayer games. Zagal et al. (2006) distinguish three different types of multiplayer games, based on the interaction between players: competition, cooperation and collabo-

ration. Competitive games require a player to confront other players in the game. In such games, players have opposing goals. In cooperative games, opportunities exist for players to work together, from which both players could benefit. However, "a cooperative game does not always guarantee that cooperating players will benefit equally or even benefit at all" (Zagal et al. 2006; p. 2). Collaborative games differ from cooperative games in that the players have the same goal, and they share the rewards or penalties of their decisions, whereas in cooperative games players may have different goals, and achieve these goals independently from each other. So,"the challenge for players in a collaborative game is working together to maximize the team's utility"(Zagal et al. 2006; p. 3), while in cooperative games players only have to consider their own utility. Considering non-competitive multiplayer games, generating levels for collaboration is the more challenging type, because the forced mutual benefits of the cooperation puts an extra constraint on the design space.
The generation process is based on the idea of generating a path within a space and subsequently filling this with rooms containing more concrete gameplay sections. This method partially draws inspiration from methods used for level generation in the game *Spelunky* (Kazemi n.d.), and Karavolos et al. (2015). Design choices like using numbers to define certain types of rooms is something that we have applied to our project as well. However, while Spelunky does not pay much attention to the layout of the individual rooms (entrances and exits being the only important element), we want our level generator to generate levels with a set path in mind that we want the players to follow.

3. GAME MECHANICS FOR COLLABORATIVE PLATFORMERS

A platform game requires the players to be able to jump, so they can traverse platforms. To create a puzzle-platformer, this requirement must be paired with a certain mechanic that creates interactive puzzles that the players can solve. For basic gameplay, the players must be able to run, jump and interact with objects in the game space in different ways, e.g. activating a lever to move platforms or being able to pick up a player or object to change the required jump height or width. Other typical means to challenge the player's skills and add variation to the gameplay are the addition of enemies and dangerous obstacles.
The prototype in this paper incorporates additional objects that affect player movement, i.e. trampolines, moving platforms and grabbable elevators. These objects affect the pacing of the game, and serve to make the platforming aspect of the gameplay more interesting for the players. The prototype also contains a typical implementation of puzzle mechanics, players will be required to work together to solve lock-and-key puzzles. However, the lock-and-key mechanism is implemented as a gate that is opened by pressing a lever.

Most of these game elements require only one player to achieve their effect on the gameplay. The collaborative mechanics involve the lock-and-key mechanism and using each other's head as a platform to reach a higher place within the level, and will be described by the patterns in the next section.

4. GAME DESIGN PATTERNS

There have been several attemps to formalize game design ideas into patterns. Some of these patterns focus on gameplay sections of a level (Reuter et al. 2014, Hullett and Whitehead 2010), others focus on game mechanics (Rocha et al. 2008, Seif El-Nasr et al. 2010). Dahlskog and Togelius (2012) even extract patterns in the form of level segments from handcrafted game levels in order to generate levels with the same style.
The design patterns in this study are defined manually, and inspired by the templates of Reuter et al. (2014) and Hullett and Whitehead (2010). However, they are related to the other patterns as well. For example, the upsy-daisy and the timed lever/gate pattern are a type of 'Shared Puzzles' as defined by Seif El-Nasr et al. (2010).

Defining game design patterns and their instantions as level segments allows us to view the level as a sequence of patterns, or combinations of patterns. Then, the generator can combine these patterns in order to design gameplay situations. Because of space constraints, we will only describe the collaborative patterns we found. For each pattern we give a description of what the pattern entails, as well as possible affordances for the pattern and the resulting consequences for the player experience. Listed below are three of these game design patterns:

- **The Upsy-Daisy**
  Description: An obstacle that requires two players to time their jumps together to get onto a platform. After this, the player can push an object down to the other player so they can both get up and proceed further through the level (See figure 1).
  Affordances:
  - Obstacles near the platform that needs to be reached.
  - Distance of platform above the ground.
  Consequences: Causes a sharp increase in challenge and coordination for the players. Both players must time their jumps together so that one of the players can use the other player's head to get high enough to reach the platform.

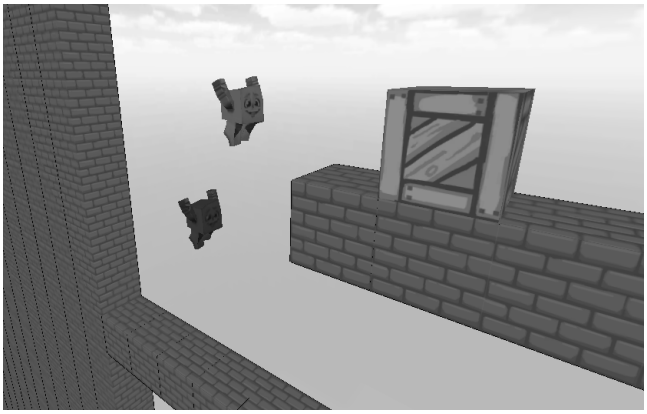


Figure 1: An example of the upsy-daisy pattern.

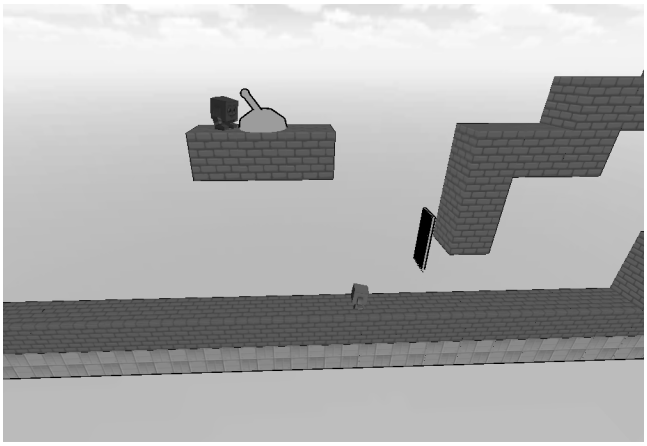


Figure 2: An example of the lever-and-gate pattern.

- **Timed Lever-and-Gate**
  Description: A lever that, when pulled, opens the related gate and starts a timer. When the timer runs out, the lever is reset and the gate closes (As seen in figure 2).
  Affordances:
  - Amount of time before lever resets
  - Number of obstacles between the lever and the gate
  Consequences: An increase in challenge and coordination because of the fact that the players must communicate about which player will go through the gate and which is going to pull the lever.

- **Common Enemy**
  Description: An enemy that cannot be destroyed if the player confronts it alone, since the sides that are facing the player are invulnerable (See figure 3).
  Affordances:
  - Number of enemies placed
  - Amount of health of the enemy
  - Damage done when hit by the enemy
  - Speed of the enemy.
  Consequences: Confronting a common enemy offers a greater challenge than a regular enemy, as it requires more skill to draw the enemy to one player, while the other player hits its blind spot. This also results in a greater sense of competence when the players manage to defeat the enemy. Furthermore, this pattern demands coordination between the two players, since they will need to communicate which player lures the enemy and which one attacks it.

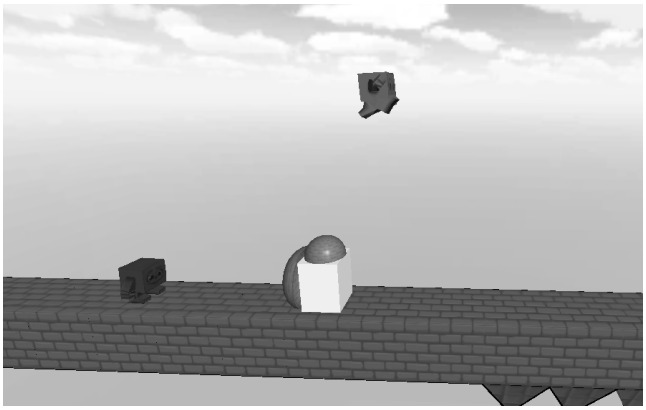


Figure 3: An example of the common enemy.

5. THE LEVEL GENERATION PROCESS

The level generation process is based on a sequence of generative grammars. Each grammar receives input, performs its transformations and sends the output to the next grammar, until the level is finished. The first input is a 6x3 grid of tiles, which are all of the type undefined, except for the leftmost column, which are of the type 'start'. These tiles define the possible positions of the level segment with the initial spawn location of the player. The first grammar transforms two of the three tiles into an undefined tile and transforms one of the tiles adjacent to the remaining tile into an 'end' tile. The next steps of the process, which will be described in more detail in the following subsections, are as follows:

- Path Generation
- Define Level Segments
- Apply Design Patterns
- Final Adjustments.

Path Generation

The path of the level is generated by moving the 'end' tile around in the grid. The path is generated in two passes. First from left to right, then from right to left. We split this process into two steps, because grammars are context-free systems. By marking the orientation of each new tile that is generated we add context to a context-free system and can guarantee a connecting path.

Figure 4: During the first pass, the generator creates a path from the left column to the right column. This path is marked, so the grammar will not attach level segments to these segments during the second pass. The abbreviation for each tile signifies the orientation that the tile will have. 1H stands for a horizontal segment and 1V a vertical one. A tile such as LCD stands for Left-Corner-Down, meaning the player would enter the segment from the left side of the tile and move through a corner, exiting on the right. UCR stands for Up-Corner-Right, DCR for Down-Corner-Right, et cetera.

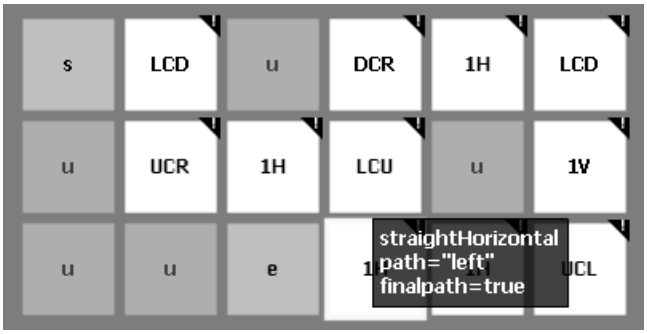


Figure 5: During the second pass, a path to the left is generated. In contrast to the first pass, the length varies due to stochastic rules. Note that the highlighted tile has two variables assigned to it. The *"path="* variable annotates the direction that the player will be going when entering the segment. The variable *"finalpath="* is assigned to the tile which is generated during an iteration, this allows the following iteration to recognize where the path left off.

Define Level Segments

After the generation of the path is completed, each tile is expanded into a 20x20 tile "segment". These segments depict the first step in visualizing what the final level will look like. Each level segment has several templates, which are chosen on the basis of what the segment has been marked as in the previous step. While generating these segments, some might contain encounters. Encounters are the term used to describe instances of design patterns within the level generation process.

Apply Design Patterns

Applying the design patterns to our level generation process is done through the use of what we call encounters. Encounters are used to apply instances of design patterns within the level generator, as shown in figure 6. The larger level segments contain locations for encounters. These encounters are smaller level segments which contain challenges involving certain mechanics. To create an added layer of depth to the generation of the design patterns, it is also possible for an encounter to contain other encounters. So a movement-based jump encounter could contain smaller danger encounters within it. Thus, the generator can create level segments that pose both a movement-based challenge, as well as a danger-based challenge.

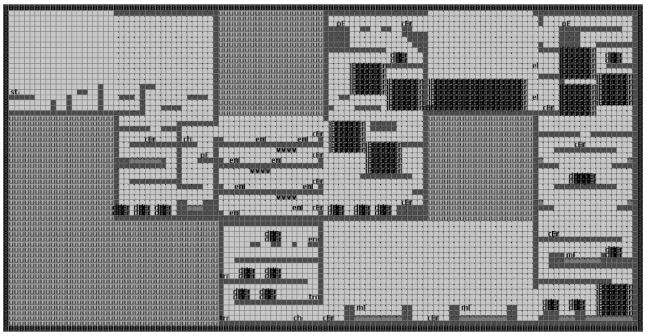


Figure 6: The grammar expands the level into 20x20 segments and gives a first impression on how the final level will be shaped. Note the blue-colored encounters, indicating that these will be translated to specific gameplay situations.

Final Adjustments

The level generator proceeds with doing small adjustments regarding object rotation and removing any variables that are not necessary for the parser. Variables that are used by the lever-and-gate mechanisms, however, are left in the model, since these are important for the parser to identify which gate is linked to which lever. This results in the final level representation (as shown in figure 7), which can be exported as a text file and read by the parser.

Parsing in Unity

To test the playability of the levels that are generated, we have created a prototype in the Unity3D engine that utilizes all of the mechanics described in section 3. This prototype contains a parser that is able to read the text file from Ludoscope and puts every tile into a 2D array. Tiles are placed accordingly through the use of a switch-case programming statement, allowing the parser to instantiate game objects based on the associated tile.

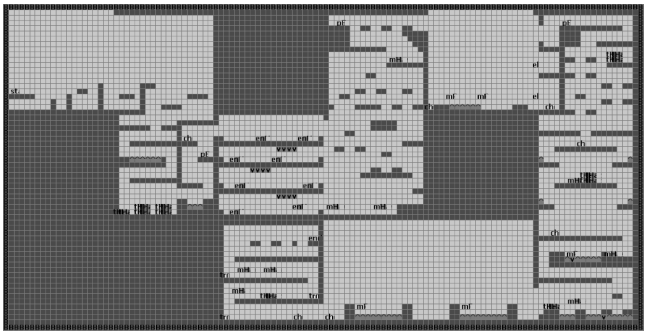


Figure 7: Final level representation, ready for parsing.

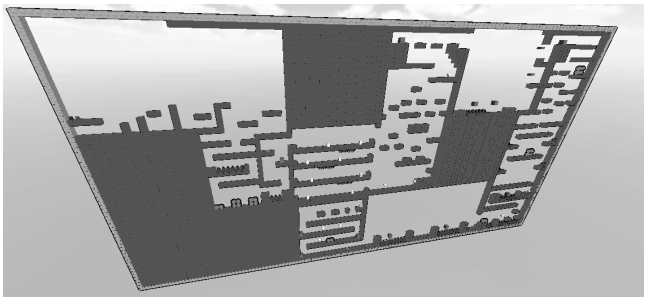


Figure 8: A parsed level within the Unity prototype.

6. DISCUSSION

A good level generator should increase replayability, and be efficient in terms of development costs, i.e. creating the generator should outweigh the cost of handcrafting the levels. Both of these characteristics depend on the variation of the possible levels that can be generated. The generation process described in this paper allows for a lot of variation with a limited set of handcrafted rules. To calculate the minimum number of different levels that could be generated, we compose an equation which takes the number of variations of the shortest path possible, from left to right with one corner segment (as seen in figure 9). The variables within this equation are dictated by the the possible variations for each segment. For this path, it means that the startSegment only has four different possible variations, straightSegment has six (but is executed four times) and finally the corner and endSegment both have three possible variations. So, the designer has to design sixteen level segments.

We can calculate a minimum number of possible level variations with the equations below. We divide the equation in two parts so we can present the difference in possibilities when encounters are included to the equation. The equation below shows the possible variations when excluding encounters from the generation process:

$$\begin{aligned} excludingEncounters = \\ startSegment \times straightSegment^4 \\ \times cornerSegment \times endSegment = \\ 4 \times 6^4 \times 3 \times 3 = 46,656 \end{aligned}$$

Note that the result produced by this equation is excluding the generation of encounters entirely. If we multiply this answer with the number of encounters in just the straightSegment, which has three different variations of encounters and is applied four times within the level, this amount increases significantly. The following equation shows the possible variations when including encounters found in horizontal segments:

$$\begin{aligned} includingEncounters = \\ excludingEncounters \times encounters^4 = \\ 46,656 \times 3^4 = 3,779,136 \end{aligned}$$

These calculations show that for the smallest possible level, we can have over approximately 3.7 million different versions when including the encounters. It is important to note that differences in the levels are much more subtle when including encounters in the calculation as this can be as simple as one platform being moved a few tiles. When excluding these encounters from the calculation, the variation is much more prominent within the level, as it involves changing the layout of at least one segment of the level.

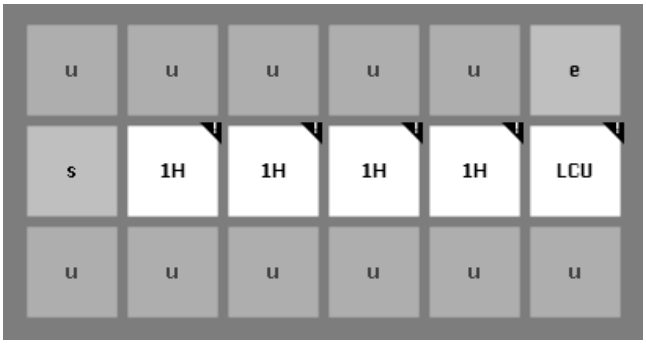


Figure 9: The shortest possible path that can be generated. See the caption of figure 5 for an explanation of the abbreviations.

The approach described in this paper allows game designers to generate a multitude of levels containing collaborative gameplay based on just a dozen handcrafted level segments. Advantages of our approach are the fact that it is rule-based in a way that allows intuitive inspection by human game designers, that it is generic enough to be applied to different games, and that it can, in principle, be applied in a mixed-initiative setting (Karavolos et al. 2015).

In our system, collaboration is enforced by requiring encounters for collaboration in specific parts of the level, e.g. in the patterns for the corners. However, the approach taken in this project is very modular, with forms of local collaboration designed especially for small segments within a level. Except for those areas of the level, there might not be interaction between the players. Collaboration could of course also occur outside of the collaboration segments, and even outside the game. More varied and meaningful ways to control and guarantee collaboration could include adding more cooperative game mechanics, such as the ones described in (Rocha et al. 2008) or adding more types of design patterns, such as the ones in (Seif El-Nasr et al. 2010). Perhaps this creates enough variation to compose a level completely of collaborative patterns. To find out to what extent people actually like or prefer these different kinds of pattern-based levels it is necessary to study how players actually play the game, and where, how, and how much they collaborate.

While designers will need to spend time designing their grammars for the level generator, this can be worth the effort when compared to the amount of time saved when the generator is functioning properly. A functional generator can create levels in a matter of seconds and even then can be tweaked, should the designer feel the need to do so. Furthermore, spending time on finding out how all of these grammars and mechanics are going to interact with each other gives the designer an idea on whether or not some mechanics will fit the game as they are meant to. Finally, any system created by the designer can also be applied to other future projects that contain similar mechanics and/or gameplay, as opposed to games which have a level generator centred around its core mechanics.

Although we have chosen to implement this method of level design to the specfic genre of puzzle-platform games, we believe the approach can be useful for other genres as well. For example, in the shooter genre, a similar method could be used to create levels containing areas fit for different styles of combat. In the roguelike genre, it could be used to generate dungeons, using encounters to generate various kinds of rooms.

7. CONCLUSIONS

In this paper we have shown how design patterns can be used to generate levels for collaborative puzzle-platformers. We have used a method that is based on generative grammars to create a path in space, and transform this path into level segments with variable elements. These variable elements can be transformed into instances of game design patterns. We have identified several game design patterns that incorporate collaboration between players, including the upsy-daisy, the timed lever-and-gate and the common enemy patterns. The variation in the levels this generator can create derives from the combination of variable path length, number of possible level segments, and the number of encounters that each template can contain.

Limitations of the approach were discussed, as well as possible directions for further work, which include more systematic ways to control and enforce levels of collaboration between players, study empirically how players actually collaborate within and around the game, giving the designer more control over the generation process in a mixed-initiative setting, and the application to other genres of games.

ACKNOWLEDGEMENTS

This research was financially supported by the Dutch Stichting Innovatie Associatie (SIA), in the context of the RAAK research project 'Automated Game Design'.

REFERENCES

Dahlskog S. and Togelius J., 2012. *Patterns and procedural content generation: revisiting Mario in world 1 level 1.* In *Proceedings of the First Workshop on Design Patterns in Games*. ACM, 1.

Dahlskog S. and Togelius J., 2013. *Patterns as objectives for level generation.* In *Proceedings of the Second Workshop on Design Patterns in Games*. ACM.

Dormans J., 2011. *Level design as model transformation: A strategy for automated content generation.* In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. ACM, 2.

Dormans J., 2012. *Engineering emergence: Applied theory for game design.* Ph.D. thesis, University of Amsterdam.

Dormans J. and Leijnen S., 2013. *Combinatorial and Exploratory Creativity in Procedural Content Generation.* In *Proceedings of the 4th International Workshop on Procedural Content Generation in Games*.

Ferreira L. and Toledo C., 2014. *A search-based approach for generating Angry Birds levels.* In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE, 1–8.

Hendrikx M.; Meijer S.; Van Der Velden J.; and Iosup A., 2013. *Procedural content generation for games: A survey. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9, no. 1, 1.

Hullett K. and Whitehead J., 2010. *Design patterns in FPS levels.* In *proceedings of the Fifth International Conference on the Foundations of Digital Games*. ACM, 78–85.

Karavolos D.; Bouwer A.; and Bidarra R., 2015. *Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation.* In *Proceedings of the 10th International Conference on the Foundations of Digital Games.*

Kazemi D., n.d. *Spelunky Generator Lessons.* http://tinysubversions.com/spelunkyGen. URL `tinysubversions.com/spelunkyGen`.

Reuter C.; Wendel V.; Göbel S.; and Steinmetz R., 2014. *Game Design Patterns for Collaborative Player Interactions. DiGRA 2014.*

Rocha J.B.; Mascarenhas S.; and Prada R., 2008. *Game mechanics for cooperative games. ZON Digital Games 2008*, 72–80.

Seif El-Nasr M.; Aghabeigi B.; Milam D.; Erfani M.; Lameman B.; Maygoli H.; and Mah S., 2010. *Understanding and evaluating cooperative games.* In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 253–262.

Shaker N.; Shaker M.; and Togelius J., 2013. *Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels.* In *Conference on Artificial Intelligence and Interactive Digital Entertainment.*

Shaker N.; Togelius J.; and Nelson M.J., 2014. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Procedural Content Generation in Games: A Textbook and an Overview of Current Research.*

Shaker N.; Togelius J.; Yannakakis G.N.; Weber B.; Shimizu T.; Hashiyama T.; Sorenson N.; Pasquier P.; Mawhorter P.; Takahashi G.; et al., 2011. *The 2010 Mario AI championship: Level generation track. Computational Intelligence and AI in Games, IEEE Transactions on*, 3, no. 4, 332–347.

Smith G.; Whitehead J.; and Mateas M., 2011. *Tanagra: Reactive Planning and Constraint Solving for Mixed-Initiative Level Design. IEEE Transactions on Computational Intelligence and AI in Games*, 3, no. 3, 201–215.

Togelius J.; Yannakakis G.N.; Stanley K.O.; and Browne C., 2011. *Search-based procedural content generation: A taxonomy and survey. Computational Intelligence and AI in Games, IEEE Transactions on*, 3, no. 3, 172–186.

Zagal J.P.; Rick J.; and Hsi I., 2006. *Collaborative games: Lessons learned from board games. Simulation and Gaming*, 37, no. 1, 24–40.

AUTHOR BIOGRAPHIES

BENJAMIN VAN ARKEL completed his BSc degree in Game Design at the Amsterdam University of Applied Sciences in 2015, on the topic of this paper. In the past, he has worked on visualizing motion capture data of soccer players for Dutch soccer club Ajax. His research interests currently involve procedural content generation and modular level design, subjects that he hopes to apply during his future career as game developer.
benjamin.v.arkel@hotmail.com

DANIËL KARAVOLOS is a researcher at the Amsterdam University of Applied Sciences, and teaches various courses at the Game Development department. He graduated from the University of Amsterdam with a Masters degree in Artificial Intelligence in 2013. His main research interests are computational intelligence, intelligent agents, procedural content generation, and automated game design.
k.d.karavolos@gmail.com

ANDERS BOUWER is a researcher and lecturer at the Amsterdam University of Applied Sciences. He studied Artificial Intelligence at the VU Amsterdam and the University of Edinburgh, and has a PhD from the University of Amsterdam since 2005. His research interests include interactive learning environments, music interaction & cognition, multimodal mobile systems, and automated game design.
a.j.bouwer@hva.nl

HOW TO USE COMBINATORIAL OPTIMIZATION PROBLEMS (TRAVELING SALESMAN PROBLEM) FOR PROCEDURAL LANDSCAPE GENERATION

Alan Ehret[a], Peter Jamieson[a], and Lindsay Grace[b]
[a]Miami University and [b]American University
email: jamiespa@miamioh.edu

KEYWORDS
Landscape Generation, Traveling Salesman Problem

ABSTRACT

In this paper, we examine how the traveling salesman problem (a combinatorial optimization problem) can be used to create virtual landscapes. For this work, we show how the entire search space of ten city TSP instances can be organized into virtual landscapes, and illustrate how by changing the TSP instance problem we can control some properties of these landscapes. We provide three different methodologies for producing landscapes and show results for TSP problem instances. Our results show that our one of our methodologies generates the best aesthetically looking results and can be controlled by the problem.

1. INTRODUCTION

In this work, we demonstrate how combinatorial optimization problems can be used to create procedural virtual landscapes. The point of this work is to show the optimization space (mapped into 3D) to algorithm designers to help them get a general feel for what their problem search space looks like. A secondary benefit of our approach is that using combinatorial optimization problems gives us some control over landscape generation for small scale problems. Thirdly, this control of landscape generation can be compressed and distributed in a multi-player game environment by transmitting the problem instance, and thus, this is a form of compression for landscapes noting that you can also send the initial random seed for a noise generator such as Perlin noise (Perlin 2002).
In this paper we will describe our methodology for creating these landscapes from the traveling salesman problems (TSPs) that are small enough for us to compute all solutions. Our methodology uses the Steinhaus-Johnson-Trotter algorithm (Surhone et al. 2010) to progress through the solution space, but because of the relationship between neighboring solutions, we use various schemes to create more realistic landscapes. With three methodologies, we show how different problem instances (where cities are placed) of the TSP can be used to create different looking landscapes as defined by what they look like and their respective distributions.

2. BACKGROUND

In this section, we will review research in procedural landscape generation for virtual worlds (including games), describe what combinatorial optimization problems are and why they are relevant, and describe the TSP.

2.1 PROCEDURAL LANDSCAPE GENERATION

The video game "No Man's Sky" to be released in 2015 boasts that it contains a massive procedurally generated universe. Procedural generation allows designers to create worlds without the need to build the details. One question we have is can this computation used to create these worlds also be leveraged for other valuable computation? Therefore, this work is a derivative from our harnessing computation (Jamieson et al. 2012) work.
For a high-level survey of papers on landscape generation for virtual worlds and games, Carli provides a thorough review on the subject that includes how to create landscapes as well as other procedural places such as cities (Carli et al. 2011). An earlier and slightly more comprehensive review is done by Haggstrom (Häggström 2006) where they examine procedural generation from landscape all the way to plants and life. Two other major surveys of this area include Cruz (Cruz 2015) where how a full virtual-space can be created using a combination of methods is studied and Togeliu *et. al.* look at search space based procedural generation (Togelius et al. 2011).
In most procedural landscape generation, the key step is the function generating randomness. As mentioned earlier, Perlin noise (Perlin 2002) is a type of noise that has gradients that when looked at from a the perspective of a birds-eye looks very similar to mountain regions. Many researchers have looked at variations on noise to create various different visual effects.

2.2 COMBINATORIAL OPTIMIZATION

Our work focuses on creating hill- and mountain-based terrain using the solution space to what are called com-

binatorial optimization problems. These types of problems are characterized based on they are problems that have a finite set of unique objects that need to be arranged, visited, or used in some way such that, in many cases, the best solution is not feasible to compute as the number of objects grows.

These problems tend to be described by graphs as in the field of graph theory. However, engineers and scientists solve many real-world optimization problems as modeled by graphs and framed as combinatorial optimization problems. For example, the creation and manufacturing of modern day microchips requires a number of combinatorial optimization problems to be reasonably (heuristically) solved; for example, the FPGA placement problem is one small example of many (Jamieson 2010). Another real-world example is the scheduling problem where a solution for a schedule of individuals, machines, or deliveries can be created given constraints; a small example of this is the job-shop scheduling problem (Mencía et al. 2014).

2.2.1 TRAVELING SALESMAN PROBLEM

We have chosen the TSP as our combinatorial optimization problem. Traditionally, problems like these quickly become intractable to optimally solve as the number of objects factorially increases the number of solutions. Meta-heuristic algorithms are used to find reasonable solutions to these problems and are classified as nature inspired versus non-nature inspired (Blum and Roli 2003).

Imagine 4 cities in the set {[A]mstersdam, [T]oronto, [C]incinatti, and [L]ondon}. A tour of these cities includes the two solutions A,T,L,C and A,L,T,C. For these examples, the second tour results in a shorter distance traveled. For this problem, we define distance traveled as the cost function - the cost for a given solution - in this case, distance traveled. The goal of an algorithmic solution for the TSP is to find the minimum cost, and as the number of cities grows this becomes computationally in-feasible.

3. METHODOLOGY

In this work, we use the search space of a TSP problem instance as the noise generation function for generating a landscape. For example, our above TSP with four cities has 24 solutions each of which has a cost (total Cartesian distance). Using these 24 costs as a parameter for elevation, we can create a landscape.

To create a 2D landscape there are some additional challenges. First, what are the x and y coordinates for each cost function z noting that each solution must have a unique x and y coordinate, and related to this, how do we generate every point in the permutation space? Secondly, will this noise generated by a TSP instance and some manipulation look like a landscape?

3.1 GENERATING PERMUTATION POINTS

To traverse the entire search space for tractable TSP problems, we use the Steinhaus-Johnson-Trotter algorithm (Surhone et al. 2010). This algorithm picks objects from the problem in a similar fashion to gray code for binary systems. Given each solution instance we calculate the cost function to get the value for the z-axis. We describe three methods to place solutions in the next section.

3.2 2D PLACEMENT METHODS

Our three different methods to determine the x and y coordinates are called striation, merge, and gradient. For each method we show the pseudo-code noting that the input $TSPcosts$ is an array consisting of all cost function values traversed using the Steinhaus-Johnson-Trotter algorithm.

Algorithm 1 Striation algorithm

```
procedure STRIATION(TSPcosts, Out)
    2Dmat = 2DSquare(TSPcosts)
    while square = nextSquare(2Dmat,100,100) do
        Rotate square random angle (-90 to 90)
    end while
    Out = Gausian filter 2Dmat: 40x40, sigma=300
end procedure
```

The algorithm 1 we call striation since it constructs the 2D map line by line from the Steinhaus-Johnson-Trotter algorithm, which seems to result in striations. We perform some small transformations by rotating groups of 100x100 pixel buckets. The Gaussian filter is used to smooth out the features after we have done our slight transformation.

Algorithm 2 Merge algorithm

```
procedure MERGE(TSPcosts, Out)
    2Dmat = 2DSquare(TSPcosts)
    while square = nextSquare(2Dmat,100,100) do
        Place square randomly in g1
        Place square randomly in g2
    end while
    Out = Poisson image edit (g1, g2)
end procedure
```

The second algorithm 2, called merge, differs from the first algorithm by creating two 2D spaces and using the Poisson image edit (Pérez et al. 2003) to merge the two. The goal is to have less regularity in the landscape that we tend to have in our first algorithm (Algorithm 1).
The third algorithm 3, called gradient, tries to manipulate square regions. This methodology generates some of the best landscapes. Also, note that there is an edge function defined in this methodology. The reason for

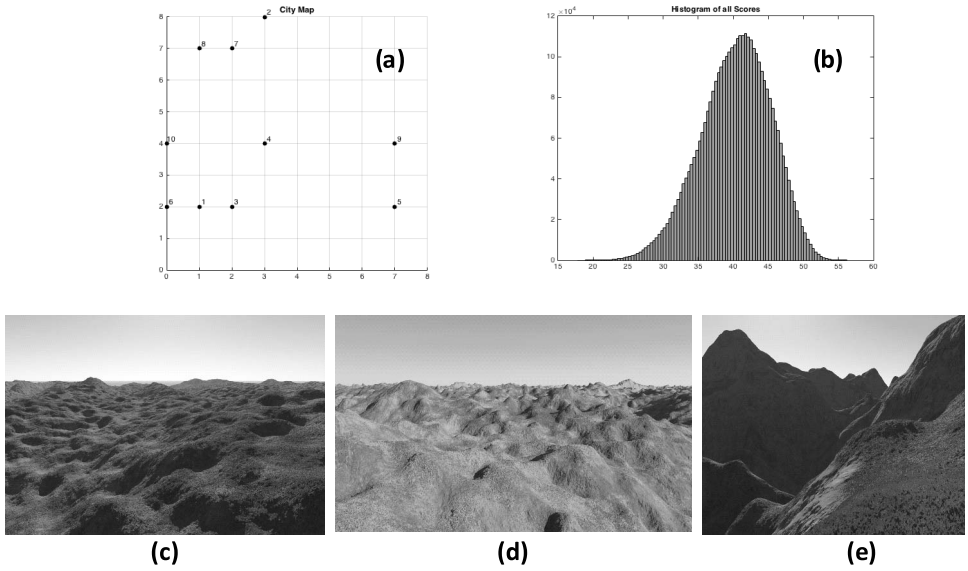


Figure 1: Random Results: (a) the city map (b) the result distribution (c) striation (d) merge (e) gradient

Algorithm 3 Gradient algorithm

```
procedure GRADIENT(TSPcosts, Out)
    2Dmatrix = 2DSquare(TSPcosts)
    r1 and r2 = set of rectangular matrix, side length ratio 10:1 row:col from 2Dmatrix
    z1 and z2 = zero matrix the same size as square
    while t doraversing square regions
        z1[random column]=random rectangle r1s;
        z2[random column]=random rectangle r2s;
        rotate z1 and z2 by a random angle
        g1 = g1+z1;
        g2 = g2+z2;
    end while
    edges = sine waves for each of the 4 edges
    Out = combine edges, g1, and g2
end procedure
```

this is that in future work, we want to use larger TSP instances, but this means that it is impossible to calculate all the possible solutions ahead of time and a continuous process is needed. In this approach we create smaller regions and plan to stitch them together meaning we need a common border. Joining boundaries can be challenging, so for now, our approach is to make the edges of a square tile common by making the edges a common function. This allows us to stitch tiles together seamlessly.

4. RESULTS

In this section, we show how different 10 city TSPs problem instances result in varying results for procedural landscape generation. The first set of results we show are a random TSP instance to provide a comparison point to our other results. In some cases we show the distribution of cost function calculations as these differences result in varying results. Using the free tool Terragen 3 `http://planetside.co.uk/terragen-3-free-download` from Planetside Software we import our data to create virtual landscapes, which we also show in these results.

4.1 RANDOM

For the random TSP problem instance, ten cities are randomly assigned to an 8 by 8 grid as shown in Figure 1 (a). To the left, in Figure 1 (b) we show the distribution of cost functions based on all possible cost function scores noting that the optimal solution sits near 20. Finally, Figure 1 (c), (d), and (e) represent the landscapes generated in Terragen 3 for each of our methodologies in the previous section - striation, merge, and gradient. Note the similarities between Figure 1 (c) and (d) in this case, which both result in what we might describe as a relatively flat bumpy hill range. The stark difference in Figure 1 (e) is due to the gradient of values, which results in a smoother overall look and more variety between the peeks and valleys.

4.2 TWO CLUSTERS

The two cluster TSP instance creates two separated groupings of cities in the upper left and lower right corners of the map as shown in Figure 2 (a). This means that any city paths (beyond two such paths) that cross this separation will add significant distance to the respective cost function, and there should be distinct good and bad solutions in this search space. This is demonstrated in the distribution in Figure 2 (b) where we can

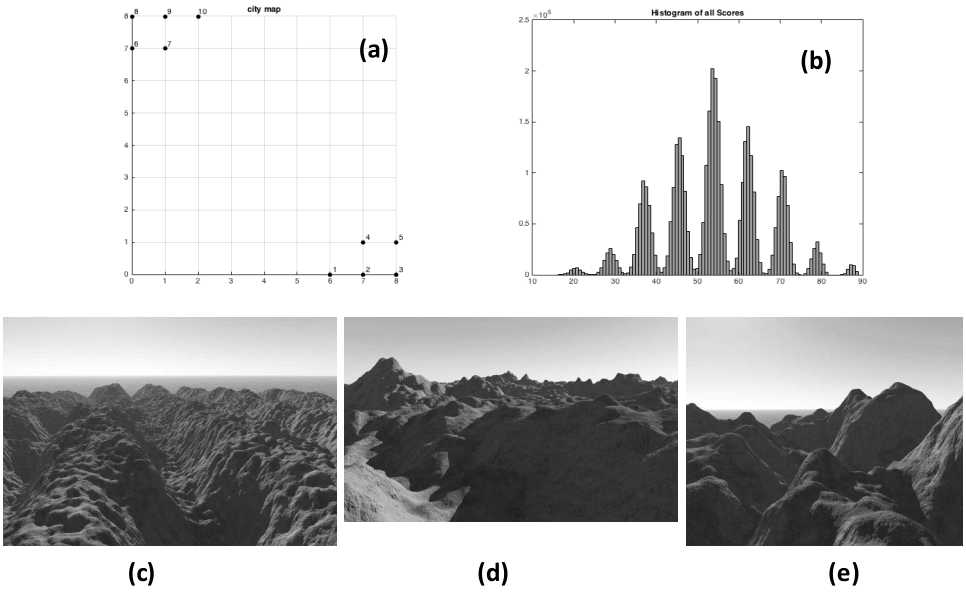


Figure 2: Results: (a) the city map (b) the result distribution (c) striation (d) merge (e) gradient

see what we might call a comb like function or in signals and systems the pulse response as seen in frequency domain.

This results in a very repetitive landscape for the striation method as seen in Figure 2 (c). Figure 2 (d) shows the merge method, which still has the bumpy characteristics, but looks better than random. Finally, Figure 2 (e) results in a much more mountainous landscape compared to the football TSP instance and similar to the random results. This is because there are more bad solutions and excellent solutions and this creates clearer peaks and valleys.

4.3 THREE CLUSTERS

Creating three clusters instead of the previous two changes the results slightly and the arrangement of cities can be seen in Figure 3 (a). The distribution (Figure 3 (b)) has a similar look to the one in Figure 2 (b), but the creation of three clusters has changed the characteristics of this new distribution slightly. For example, there are two bumps from 35 to the left for the optimal values that do not have any symmetrical bumps on the right hand side of the distribution curve. Additionally, the gradient of values is less distinct as compared to the two cluster TSP problem instance.

Figure 3 (c) landscape remains characteristically bumpy and Figure 3 (d) has a look similar to it's predecessors. Figure 3 (e) is also similar to Figure 2 (e) except the severity of the peek to valley change seems to be less in the three cluster TSP problem. This is, likely, the case because the distribution of cost functions has a more continuous distribution of values.

4.6 DISCUSSION

There are two key observations that we have drawn from these results. Firstly, a TSP problem instance that results in a normal-like distribution has a slight impact on the generated landscape depending on characteristics of the curve, but those variations are very small. When a problem instance is created, such as the two cluster and three cluster, the distribution and resulting landscapes are significantly different to the normal curve (as expected), and it appears that based on the number of clusters, we can create different landscapes. Unfortunately, as this approach needs to fully traverse the entire search space, it is not computationally feasible to try larger cluster sets, and we, currently, can not explore how a larger number of groupings (greater than 3), with different distances between groupings impacts our generated landscapes.
The second observation we make is that the nature of the TSP solution space as traversed by Steinhaus-Johnson-Trotter algorithm results in regularities that do not look good in terms of landscape generation.

5. CONCLUSION AND FUTURE WORK

This paper shows how combinatorial optimization problem search spaces can be manipulated and used to procedurally generate virtual landscapes. The reason we demonstrate the viability of this approach is that the search space values can be of interest to algorithmic designers, and the landscapes generated can be controlled by manipulating the TSP problem instance resulting in a slight compression advantage.
In this paper, we focused on showing how the TSP can be used to create virtual landscapes. We provided three

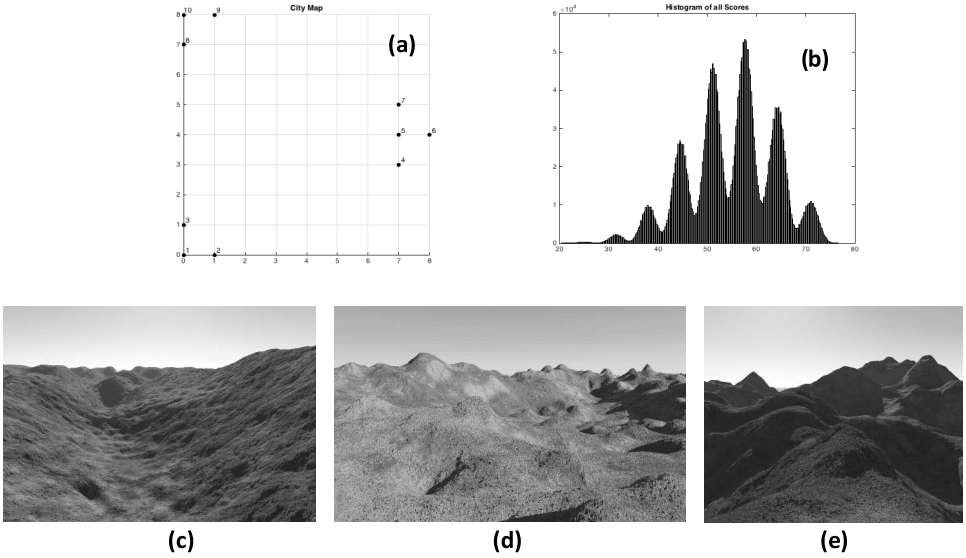


Figure 3: Results: (a) the city map (b) the result distribution (c) striation (d) merge (e) gradient

different methodologies for how to take the TSP cost function values (our z values) and then how to map these into unique x and y coordinates to create a landscape. We then showed how different TSP problem instances resulted in different landscapes. Of the three methods presented in this paper, our gradient approach creates the aesthetically best looking landscapes without the characteristic bumpy look that both the striation and merge methods tend to produce.

For future work, our focus is on using larger TSPs that can not be computationally solved (beyond 11 cities). We have already suggested in our description of the gradient method, how we plan to deal with stitching together tiles of landscape. This stitching itself, can be improved from using simple trigonometric functions, but our larger focus is on how can we create larger landscapes from large TSP problems, and also provide algorithmic benefits to engineers and algorithm designers. Our first approach into this domain will be to use meta-heuristic algorithms, such as genetic algorithms and simulated annealing algorithms, to find local optimal points, and then use these points as seed points for a region/tile in which the additional points will only differ by only one or two objects. That will create regions in which the designer sees similar or neighboring solutions.

REFERENCES

Blum C. and Roli A., 2003. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput Surv*, 35, 268–308. URL `http://doi.acm.org/10.1145/937503.937505`.

Carli D.M.D.; Bevilacqua F.; Pozzer C.T.; and DOrnellas M.C., 2011. *A survey of procedural content generation techniques suitable to game development.* In *Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on*. IEEE, 26–35.

Cruz L.M.V., 2015. *High-Level Techniques for Landscape Creation*. Ph.D. thesis.

Häggström H., 2006. *Real-time generation and rendering of realistic landscapes*. Ph.D. thesis.

Jamieson P., 2010. *Revisiting Genetic Algorithms for the FPGA Placement Problem.* In *GEM*. 16–22. URL `http://www.users.muohio.edu/jamiespa/html_papers/gem_10.pdf`.

Jamieson P.; Grace L.; and Hall J., 2012. *Research Directions for Pushing Harnessing Human Computation to Mainstream Video Games.* In *Meaningful Play.* URL `http://www.users.muohio.edu/jamiespa/html_papers/meaning_12.pdf`.

Mencía R.; Sierra M.R.; Mencía C.; and Varela R., 2014. *A genetic algorithm for job-shop scheduling with operators enhanced by weak lamarckian evolution and search space narrowing. Natural Computing*, 13, no. 2, 179–192.

Pérez P.; Gangnet M.; and Blake A., 2003. *Poisson image editing.* In *ACM Transactions on Graphics (TOG)*. ACM, vol. 22, 313–318.

Perlin K., 2002. *Improving noise*. In *ACM Transactions on Graphics (TOG)*. ACM, vol. 21, 681–682.

Surhone L.M.; Tennoe M.T.; and Henssonow S.F., 2010. *Steinhaus-Johnson-Trotter algorithm.* Betascript Publishing.

Togelius J.; Yannakakis G.N.; Stanley K.O.; and Browne C., 2011. *Search-based procedural content generation: A taxonomy and survey. Computational Intelligence and AI in Games, IEEE Transactions on*, 3, no. 3, 172–186.

ONLINE GAMING

A SERVER-SIDE FRAMEWORK FOR THE EXECUTION OF PROCEDURALLY GENERATED QUESTS IN AN MMORPG

Jonathon Doran
Dept. of Computer Science and Information Systems
Bradley University
jhdoran@bradley.edu

Ian Parberry
Dept. of Computer Science and Engineering
University of North Texas
ian@unt.edu

KEYWORDS
MMO, procedural content generation

ABSTRACT

We describe a framework for executing procedurally generated quests implemented in the MMORPG Everquest using the Open Source EQEmu Everquest server. Quests play out at run-time using a collection of *triggers*, which consist of a testable game state condition and a script that is to be run when the condition is satisfied. We describe the interface between the quest generator and the server which enables the seamless integration of the procedurally generated quests within the existing server architecture. To demonstrate how this process takes place in real time, we analyze a nontrivial procedurally generated quest and describe the key server-controlled actions that derive from it.

INTRODUCTION

Massively Multiplayer Online Role Playing Games (commonly abbreviated MMORPG) pose a significant challenge for procedural content generation. Of the content which might be procedurally generated, the *quest* is perhaps the most difficult. Quests are tasks assigned to players in the game, combining narrative elements and problem solving with combat and survival in a hostile world. Creating these quests requires the creation of in-game agents, items, and dialog. A quest generator must inform the game engine about which tasks need to be performed by both the player and the engine, what the criteria for success is, and when events should occur.

Our quest generator is novel in that it has been integrated into an actual AAA MMO, rather than generating quests for a standalone single-user game. This is the market we feel is most in need of large volumes of content. While we believe that our generator would generate content suitable for any MMO, we selected Everquest as our target demonstration system. The presence of an open-source server emulator (EQEmu) was a significant factor in selecting this game. We will examine the steps needed to support Everquest, with the understanding that other games could be supported with different post-processing.

Procedural quest generators place a number of requirements on a game server. These can arise from the need to have the generator run with little or no human intervention, the need to introduce new quests into a world without breaking any existing functionality, and the need to remove quests without negative consequences. We demonstrate the design and implementation of a procedural quest generator for the MMORPG Everquest using an Open Source emulated server and an Everquest client released by Sony Online Entertainment via Steam in 2010 (See Figure 1). We believe that our framework is general enough to be adaptable to other server architectures provided certain requirements are met.

The remainder of this paper is divided into six sections. In the section "RELATED WORK" we consider related work. In the section "QUEST GENERATION" we briefly describe the procedural quest generator. In the section "SERVER REQUIREMENTS" we discuss the specific requirements that the Everquest server imposes on the execution of procedurally generated quests at run-time. In the section "THE SERVER INTERFACE" we describe the interface between the quest generator and the Everquest server. In the section "THE QUEST & TRIGGER MANAGERS" we show how we were able to meet the server's requirements and control quest execution. In the section "AN EXAMPLE QUEST" we analyze a sample procedurally generated quest and describe how the server is able to make the quest play out in real time.

RELATED WORK

MMORPGs are persistent interactive worlds shared by many players. Players face many challenges in these games, among them are structured activities known as quests. Quests consist of objectives, tasks, and success or failure conditions (Ashmore and Nitsche, 2007; Doran and Parberry, 2011). Dickey notes that players strategize, collaborate, and plan their solutions to these challenges as a major form of gameplay (Dickey, 2007). Rewards from quests are often the primary motivation for players to engage in gameplay, and in these cases a compelling story is not required. Quests can play a significant role in content delivery by providing narrative and guiding player involvement in the world ((Grey and Bryson, 2011; Joslin et al., 2006; Smith et al., 2011; Tomai et al., 2012a). Quests can be used to relate epic stories (Bateman and Boon, 2005). One of our long-term goals is to determine if improved storylines can result in players focusing as much on the storyline as the reward.

There is no intrinsic meaning for quests, only potential mean-

ing (Tronstad, 2002–2003), which means that discovery of this meaning is a task for the player. Tronstad also notes that as quests are a search for meaning, once solved they cannot be performed again, since we only have one opportunity to experience a quest for the first time. To maintain a body of novel content, one must constantly introduce new content into the game. Procedural quest generation is a logical solution, if the generators are capable of providing the meaning of which Tronstad speaks (Ashmore and Nitsche, 2007; Kybartas, 2013; Reed et al., 2011; Sullivan et al., 2011; Tomai et al., 2012b; Zook et al., 2012).

Aarseth's analysis of the quests in Everquest has suggested that the player's task is finding "one acceptable path" (Aarseth, 2004), which is evidence that player agency is limited(Wardrip-Fruin et al., 2009). We can posit that the introduction of additional paths would be of great benefit to playability.

The reduction of resource requirements is important for architectures which need to scale, such as those found in MMORPGs. In the study of computer networks we have seen how explicit notification of relevant events can reduce unwanted traffic, leading to less resources needed to process this traffic (Smed et al., 2002). The reduction of traffic in networks is an appropriate model for reducing event handling in a game. In both situations we consider solicited versus unsolicited event notification and handling. This publisher/subscriber pattern was also discussed in terms of client/server communications (Caltagirone et al., 2002). We have adopted this technique by requiring quests to subscribe to certain types of events at appropriate points during their execution.

QUEST GENERATION

We generate quests using a technique previously described by the authors (Doran and Parberry, 2011), starting with an NPC and selecting a strategy template appropriate for its motivation. These templates are part of a plan library, and can to be expanded to the desired level of complexity. Complications and follow-on quests can be added to a template, causing more strategies to be used. These new strategies can also be expanded as needed.

The generator was modified slightly to assign a number of *points* to strategies representing their difficulty, and randomly dividing these points among new strategies during the expansion process. If we view the set of strategies used as a graph, we see a tree structure growing from the original root goal. Points over a minimum value are allocated randomly to leaves, which then add child nodes that become new leaves. Points are consumed by each of the strategy templates, so the initial point total limits the size of the generated tree.

This differs from planning algorithms, as we start with a viable solution in the form of a trivial goal, and add additional subgoals while preserving the overall strategy; thereby by adding obstacles to the path the player must take to satisfy the original goal. This might be done by making a needed asset hard to obtain, relying on knowledge that is not obvious or commonly known. This process will continue as long as necessary to consume the points allocated to the new branches. Planning algorithms, on the other hand, start with an initial state and a goal and attempt to generate a graph that connects the initial state to the goal state. There is no guarantee that a viable solution exists, and computationally expensive search techniques must be employed to find any solution. Unlike planning solutions, our approach builds solutions in constant time.

Figure 1: A screen shot of the Everquest client

If quest generation fails, which might be due to requiring more points for a branch than are available or exhausting available world assets, the quest generator employs backtracking over a finite set to attempt an alternate solution. In practice we found that the limiting factor in creating large quests is the size of the world knowledge base. For example, a small, finite set of world locations is inadequate unless the generator can reuse locations, but it is preferable to avoid this reuse to help keep quests believable while preserving variety.

Novelty in procedurally generated content is a very important quality, as it creates the variety of content which players desire (Doran and Parberry, 2010). This variety is obtained by changing the subquests (or nodes) created, and the details of each node, such as assets and dialog. By changing the distribution of points among quest nodes, we permit different tree topologies to be created and change the difficulty of the subquests generated with each node. Asset selection (such as NPCs and items) can also have a significant impact on the structure and appearance of a quest. NPCs in particular have motivations which limit the types of subquests possible from the node referencing the NPC. In general, the number of unique combinations of assets and nodes increases significantly with the number of points allocated.

SERVER REQUIREMENTS

Our preliminary work on quest generation dealt with the generation of quests in isolation (Doran and Parberry, 2011). Interfacing a quest generator to a large, commercial quality MMORPG game engine is a complex task that taxes our abstractions to the limit.

We selected the game Everquest for this purpose because the client could easily be purchased, and there is an available

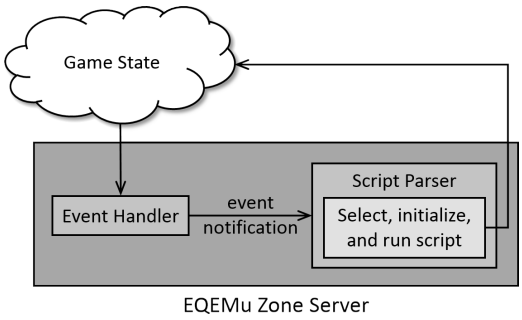


Figure 2: The EQEmu zone server responds to game events by having an event handler send a notification to a script handler, which selects, initializes, and runs a script

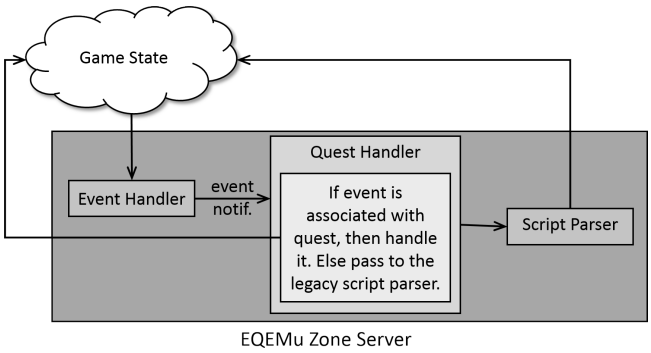


Figure 3: We modify the EQEmu zone server so that our quest handler has the first opportunity to respond to game events

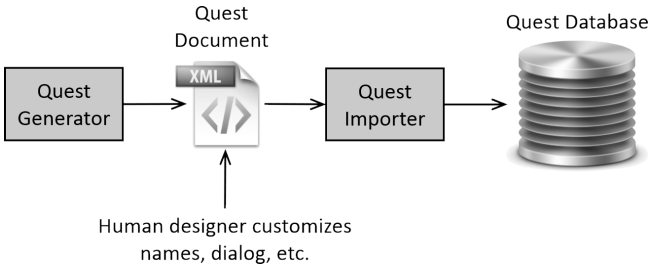


Figure 4: Getting the quest from the quest generator to the quest database

open-source server emulator EQEmu.

An EQEmu quest is implemented as a set of Perl or Lua scripts associated with the corresponding game assets. Each script may declare a handler for any of the supported events and, by interacting with server objects, change the state of the world. To simplify quest management, we enforced the requirement that each quest be implemented with a single script that can be added or removed from the server without impacting any other quests.

The run-time support for the quests created by our generator makes use of information stored in a database. Each quest is represented by a single file containing the information needed to create the script and perform the necessary database updates. We chose to implement this file as an XML document. A quest can therefore be shared with another server by sharing the XML file created by our generator.

The run-time support for quests will require information on game events as they occur. The exact events, and the data associated with them, will of course be different from game to game. In general an event will correspond to some state change in the world, for example, this could be a non-player character (NPC) entering or leaving the world, an item being acquired, or the player visiting some location. The run-time support for quests created by our generator requires custom event handling. The previous EQEmu server generated events by sending an enumeration and several parameters to script parsers, which then created an appropriate initial state for a event handler script and then called one of the scripts associated with an asset (see Figure 2).

This sequence of actions requires that the code that notifies the system of an event must know the type of asset that will handle the event; additionally, any optional parameters must be passed to the script parsers. The resulting event handling code is spread over several classes, and knowledge of how a given event is to be be handled is required of the code that raises the event. Our approach is similar, but stores all information associated with an event into an Event object which is passed to the Trigger Manager. This simplifies the script interface, and provides a general event interface which can easily be extended if new events are desired.

Events are represented in EQEmu by discrete objects that are wholly self-contained. Each event is passed into common event handling code which determines the proper event handler and makes the necessary calls to process it. The system can support as many custom event handlers as necessary. Our run-time is given the first chance to handle each event, and in the case of failure, the event will be passed back to the legacy asset scripts (see Figure 3). Multiple quest systems can coexist at run-time without risk of interference.

THE SERVER INTERFACE

As described above, our quest generator produces a single XML document for each quest, which must then be loaded onto a game server. We implemented a Java quest importer (see Figure 4) that processes these XML documents and, based on the information inside, either adds or removes a quest from the server. This application creates the run-time scripts based on the XML elements and updates the game server database. These operations are obviously game-dependent, but the principle is common to all MMORPG engines: The quest generator must communicate quests to the server using some combination of flat files or databases. Our XML format can in principle be easily extended to other forms of data storage used by a game. Notice that although the generator supplies the structure of the quest, some input is required from a human designer to customize things such as NPC dialog and the names of NPCs and items.

For convenience we define a structure called a *trigger*, which

consists of a test to be applied against the game state and a script to be executed if the test succeeds (see Figure 6, bottom left). For example, the test could specify player arrival at a certain location, and the script could result in an NPC at that location giving an important object to the player. While in principle the firing condition may be an arbitrary Boolean formula involving any number of game state variables, certain triggers are more common (see Table 1). The null trigger, which fires immediately upon creation, is a useful way to compose a sequence of actions.

```
<?xml version="1.0" encoding="utf-8"?>
<quest>
  <title lang="en">graph_0</title>
  <id>5a729d34-30c3-11e4-a10d-001d7d0a5e7c</id>

  <node>
    <name>Root</name>
    <task>gather parts for a Simple Pauldron</task>

    <assets>
      <item>
        <id>\$item_3</id>
        ...
      </item>
      ...
    </assets>

    <triggers>
      <match>
        <id>hail_1</id>
        <zone>394</zone>
        <sequence>0</sequence>
        <regex lang=''en''>\bhail\b</regex>
        <Perl> ... </Perl>
        <task>speak with Blacksmith Jones</task>
        <repeatable>
      </match>
      ...
    </triggers>
    ...
  </node>
  ...
</quest>
```

Figure 5: Structure of a quest file

Figure 5 shows the structure of the XML file produced by the generator. Each quest is given a title, which is used by the importer to display a list of quests. A globally unique identifier (GUID) is assigned by the generator, to establish a unique namespace for names and IDs. When a quest is loaded onto a server, it is assigned a locally unique identifier (such as a counter incremented for each unique quest loaded), and the GUID and title are associated with this identifier. All database modifications are logged with the corresponding quest identifier, allowing the importer to later remove the quest.

Type	Firing Condition
null	immediately
item	item is created
converse	player conversation matches regular expression
give	player gives an item to an NPC
proximity	player enters a certain area
acquire	item enters player's inventory
subquest	subquest completes

Table 1: Some common trigger types and their firing conditions

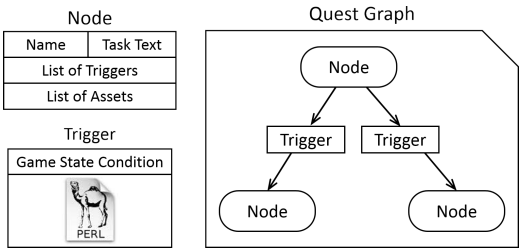


Figure 6: A Quest Graph node (top left), a trigger (bottom left), and a quest graph (right)

Our generator represents quests as graphs, which structure can be seen within the XML document. The Quest Graph consists of a set of nodes with triggers that correspond to graph edges (see Figure 6, right). Each node is assigned a name and an optional task text. The task text can be used within a game to identify quest steps, as we do with Everquest's quest journal (see Figure 7). Each node contains a set of assets that need to be created at runtime, and a set of triggers (see Figure 6, top left). The first use of an asset causes an asset record to be written, and further use of the asset can be performed by reference to the asset id. Game-specific properties are included in this asset record, but we assume that in general any game will assign some set of properties to any object in the world. Our importer creates database entries for quest-specific assets, introducing them into the game and allowing characters to interact with them. If a quest is later removed, the database entries for intermediate (non-reward) items are also removed and these items disappear from the world.

We can consider each quest to be a finite state machine in which the triggers are the events that can change its state. For example, trigger hail_1 has the type "match" which requires player speech to match a regular expression given in the trigger. In Figure 5 we see a trigger which requires the word "hail" to be spoken by the player before the quest will advance. This is a typical trigger word used in Everquest. The Perl element contains functions written by the generator that will be executed if the regular expression is matched, possibly providing a spoken response or other NPC action. This combination of an arbitrary set of triggers and a very capable scripting language can allow any event which might occur in a game to be paired with any server-side responses

Figure 7: A screen shot of the Everquest journal

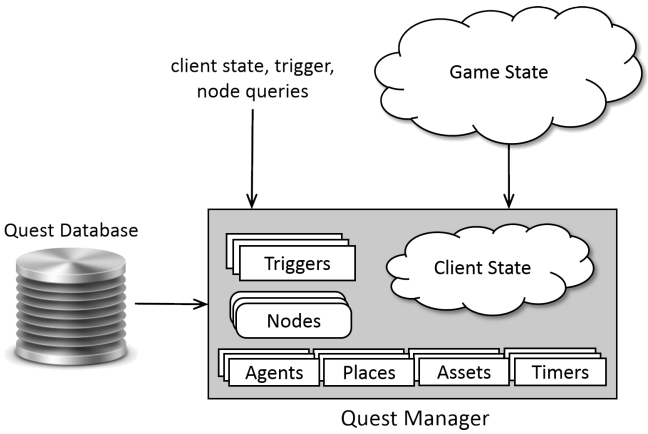


Figure 8: The quest manager manages all of the things associated with a quest, including client state, triggers, graph nodes, timers, and world entities such as places, agents, and assets

that might be required. All game-specific logic is contained in the meta-rules in the quest generator, which are separate from the general rules which might apply to any game.

Each trigger is assigned a sequence number that indicates the order in which triggers are required to fire. It is possible to have several triggers with the same sequence number, and therefore able to be performed at the same point in time. That is, triggers are partially ordered and allow the player to choose which parts of the quest they will work on next. In the section "THE QUEST & TRIGGER MANAGERS" we will show how these sequence numbers are used at run-time. Triggers may be marked as repeatable and/or optional, allowing different combinations of trigger firings to be specified by the generator. Repeatable triggers are needed at points where the player may restart the quest following a failure to complete a later quest stage. For example, a repeatable trigger can be used to permit the initial conversation with an NPC to be repeated if the player fails to advance beyond the first checkpoint. Without checkpoints (and the corresponding rollback logic), the player is committed to either complete the next quest stage or fail the quest. Optional triggers are for events that might occur and necessitate a response, but which are not required to successfully complete the quest. Node, asset, and trigger data are stored in a database for our Everquest server to access. The Perl functions are collected into a single Perl script, which exists in a quest-specific namespace. This means that function names only need to be quest-unique, simplifying the process of working with multiple generated quests.

THE QUEST & TRIGGER MANAGERS

Although the emulated Everquest server was initially capable of processing events and performing quests, it was not able to work with quests produced by our generator. Modifications were made to the server to allow it to execute the procedurally generated quests loaded by the process described in the section "THE SERVER INTERFACE". We created a *quest manager* which provides an interface to scripts, and manages client state, timers, and registrations of world entities that need to be notified when global events (not associated with any client) occur (see Figure 8). For example, NPCs that move along a route of waypoints need an event to be generated when a waypoint is reached. This event triggers the manager to assign the NPC the next waypoint in the route.

The quest manager also has the ability to checkpoint and rollback quest state in the event that part of a quest needs to be repeated. For example, if a player obtains an item needed to complete a quest and then manages to somehow lose it, the quest state is rolled back to the point prior to the player obtaining the item. The quest manager keeps track of modifications to the local world state, so that these can be removed when the quest advances or the player leaves the zone. One of the local modifications supported is selective visibility, which makes assets only visible to players associated with the quest. Association means that the player has the quest, and is at the proper point in the quest to see the asset, or that the player is grouped with someone meeting those requirements. This allows groups of players to cooperate on quests without affecting other players. The quest manager was implemented as a singleton pattern, and therefore exists in its own globally accessible namespace.

Cache managers were created for nodes and triggers that are active in the world. An active node or trigger is one with at least one player at the corresponding graph node, or waiting to complete the trigger. This optimization allows events to be screened against active objects rather than all objects associated with a quest.

The *trigger manager* handles all events generated by the game server, and attempts to match them with active triggers (see Figure 9). The match occurs when the event meets all of the firing requirements, and there is at least one player with the trigger active. Upon detecting a match, the trigger is said to fire and the trigger-specific script is executed. The trigger may or may not complete as a result of this firing. Some triggers require multiple firings before they complete, such as one might find when a player is asked to collect several objects in a set. Each collection advances the state of

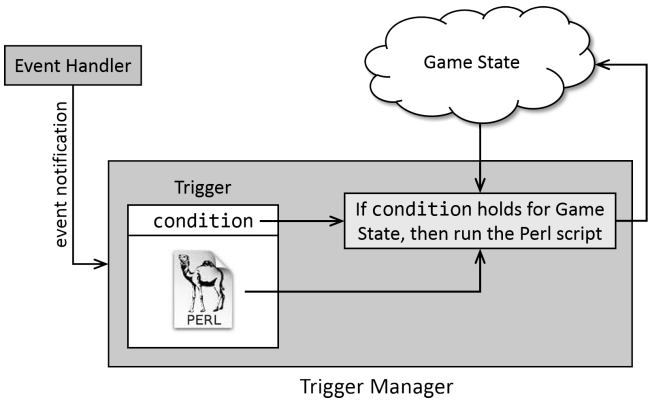


Figure 9: When notified of an event by the EQEmu event handler, the trigger manager determines which triggers should fire by testing them against the game state, then executes their scripts

the trigger, until the terminal state is reached and the trigger completes. Upon completion the node associated with the trigger is notified, and the trigger is deactivated. If there are no outstanding triggers required by this node, the node can then complete and advance the quest state to the node at the next sequence. When the terminal node in the graph completes, the quest ends. It is assumed that the terminal node takes care of any rewards associated with the quest.

AN EXAMPLE QUEST

The capabilities of this system can be seen by viewing a sample quest generated by our generator and playable in-game in Everquest. The overall structure of the nodes in the quest is shown in Figure 10, where the quest starts at the Root node, and the player is required to complete subquests represented by other graph nodes either as prerequisites or postrequisites. Triggers are not shown in this graph, but may be inferred. For example, if a node requires the player to acquire some item, the corresponding trigger would watch for this item to enter the player's inventory. In this example quest, all nodes represent follow-on quests to be completed as part of, or after the preceding node. The quest starts with the character Blacksmith Jones asking the player to gather materials and make a piece of armor. This requires the player to obtain metal panels and a venom sac from a poisonous snake. These ingredients are determined randomly, and the recipe is only usable by the player performing the quest. The player is directed to see Councilmember Ithakis for the metal panels, and the Councilmember offers to give the player the panels in return for a favor. The Councilmember wants the players to locate a lucerne leaf (another ingredient which could be used to make armor), and suggests that the player ask Farmer Jones for help. The farmer is happy to give the player a leaf, which is then turned over to the Councilmember. The players are then asked to deliver a message to a character named Nech Ilya, saying that Councilmember Ithakis has the lucerne he needs. After this is completed, the player receives the metal plates.

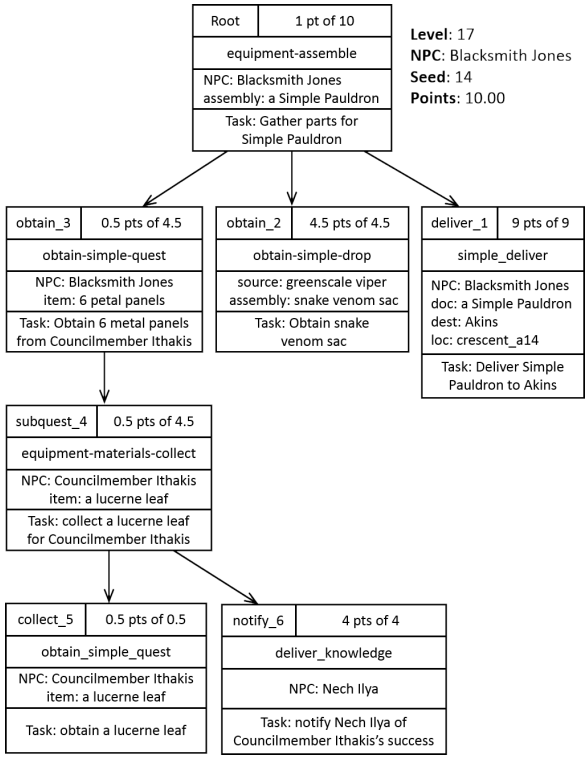


Figure 10: The structure of a sample quest

Blacksmith Jones suggests greenscale vipers might be a good source of venom sacs, and the players must find some of these snakes in the world and kill them until they find one with a rare intact venom sac. With this item, they are able to create the custom armor piece for the Blacksmith, and earn their reward.

Generation of this quest requires the selection of appropriate tasks each NPC would like performed, and the creation of custom items for the quest. Special metal plates and venom sacs are only available to the player running the quest, or any character in the same group or raiding party as the player running the quest. Custom character dialog is created for each participating character, as well as control records which bring the items and characters into the world at the appropriate time.

The quest plays out as follows:

1. When the player enters the Crescent Reach zone, a converse trigger is created, requiring the player to hail Blacksmith Jones.
2. The player hails Blackmith Jones, activating the converse trigger.
3. Blacksmith Jones asks the player to help gather materials for a piece of armor that he is making.
4. Several subquest triggers are activated, causing null triggers at the start of each subquest to fire.

    (a) The null triggers deliver instructions for each subquest.

5. Blacksmith Jones suggests that the player ask a Councilmember for help getting metal panels.
    (a) An item trigger is created and activated requiring 6 metal panels.
6. Councilmember Ithakis demands that the player run an errand (perform a subquest) in return for the metal.
    (a) A subquest trigger is created and activated.
7. Councilmember Ithakis asks the player to bring him a lucerne leaf, which is used as an armor temper.
    (a) A null trigger activates and fires, causing the council member to ask for a lucerne leaf.
8. Councilmember Ithakis directs the player to Farmer Joen, who gives the player the leaf.
    (a) A collection subquest is activated, which activates another null trigger which in turn directs the player to Farmer Joen.
    (b) A converse trigger is activated, looking for the player to mention "need" to Farmer Joen.
    (c) When the player says the magic word, the converse trigger fires causing Farmer Joen to give the player a leaf.
    (d) A give trigger activates, requiring the player to deliver the leaf to Councilmember Ithakis.
9. Councilmember Ithakis asks the player to tell an NPC named Nech Ilya about finding lucerne leaves.
    (a) Councilmember Ithakis demands that the player inform Nech Ilya that he (Councilmember Ithakis) now has a lucerne leaf.
    (b) A subquest trigger activates, which in turn activates a proximity trigger around the area where Nech Ilya will appear.
10. The player must find Nech Ilya, and speak with him.
    (a) When the player enters the area covered by the proximity trigger, it fires
    (b) A signal is scheduled which will spawn Nech Ilya.
    (c) Another signal is scheduled, which will periodically print a random tracking message and then reschedule itself.
    (d) Eventually Nech Ilya spawns, and all of the signals are canceled.
    (e) A converse trigger is activated.
    (f) When the player hails Nech Ilya, the converse trigger fires. Nech Ilya thanks the player, and the subquest completes.
11. The next time the player meets him, Councilmember Ithakis gives them the metal panels.
    (a) A null trigger fires, causing Councilmember Ithakis to deliver the message.
12. Blacksmith Jones suggests that the player hunt greenscale vipers to obtain a venom sac.
    (a) A null trigger fires, causing Blacksmith Jones to deliver the message.
    (b) An acquire trigger is activated, waiting for 1 snake venom sack to enter the player's inventory.
13. The player must locate these snakes and begin killing them. The snakes will rarely have an intact venom sac once killed.
    (a) When a venom sac enters the inventory, the acquire trigger fires and the subquest completes.
14. The player delivers the materials to Blacksmith Jones, who makes the armor piece.
15. Blacksmith Jones gives the new armor piece to the player, and asks that they deliver it to an NPC named Akins.
    (a) A give trigger activates, waiting for the player to give Akins the armor piece.
16. The player finds Akins and gives him the armor.
    (a) When the player gives Akins the armor, the give trigger completes, and the subquest completes.
17. Upon returning to Blacksmith Jones, the quest completes and the Blacksmith rewards the player.

CONCLUSION AND FUTURE WORK

We have demonstrated how procedurally generated quests can be integrated into Everquest. This work was done in support of our earlier quest generation research, as it establishes some of our earlier claims of generality. This is the first case we are aware of where procedural quest generation was applied to a AAA MMORPG. We believe that most if not all MMORPGs require this level of quests. Our generator creates quests that are playable within an existing game, requiring only that the quest is imported into the game server. In its current state, the framework requires some input from a human designer in the form of character names and dialog, which suggests that we explore dialog generation techniques. The current generator creates boilerplate names and dialog, but this is the bare minimum needed for the quest to function. At the same time we observe that while a narrative can be written to explain the quest graph, we suspect that players might prefer a more traditional story arc.
We would very much like to provide evaluation of this system in future work, after we have resolved several technical issues with measurement.

REFERENCES

Aarseth E., 2004. *Quest Games As Post-Narrative Discourse*. *Narrative Across Media: The Languages of Storytelling*, 361–376.

Ashmore C. and Nitsche M., 2007. *The Quest in a Generated World*. In *Proc. 2007 Digital Games Research Assoc.(DiGRA) Conference: Situated Play*. 503–509.

Bateman C. and Boon R., 2005. *21st Century Game Design (Game Development Series)*. Charles River Media, Inc., Rockland, MA, USA. ISBN 1584504293.

Caltagirone S.; Keys M.; Schlief B.; and Willshire M.J., 2002. *Architecture for a Massively Multiplayer Online Role Playing Game Engine*. *Journal of Computing Sciences in Colleges*, 18, no. 2, 105–116.

Dickey M.D., 2007. *Game Design and Learning: A Conjectural Analysis of How Massively Multiple Online Role-playing Games (MMORPGs) Foster Intrinsic Motivation*. *Educational Technology Research and Development*, 55, no. 3, 253–273.

Doran J. and Parberry I., 2010. *Controlled procedural terrain generation using software agents*. *IEEE Transactions on Computational Intelligence and AI in Games*, 2, no. 2, 111–119.

Doran J. and Parberry I., 2011. *A Prototype Quest Generator Based on a Structural Analysis of Quests from Four MMORPGs*. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. ACM, 1–8.

Grey J. and Bryson J.J., 2011. *Procedural Quests: A Focus for Agent Interaction in Role-Playing Games*. In *Proceedings of the AISB 2011 Symposium: AI & Games*. University of Bath, 3–10.

Joslin S.; Brown R.; and Drennan P., 2006. *Modelling Quest Data for Game Designers*. In *Proceedings of the 2006 International Conference on Game Research and Development*. Murdoch University, 184–190.

Kybartas B., 2013. *Design and Analysis of ReGEN*. Ph.D. thesis, McGill University.

Reed A.A.; Samuel B.; Sullivan A.; Grant R.; Grow A.; Lazaro J.; Mahal J.; Kurniawan S.; Walker M.A.; and Wardrip-Fruin N., 2011. *A Step Towards the Future of Role-Playing Games: The SpyFeet Mobile RPG Project*. In *Proceedings of the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Smed J.; Kaukoranta T.; and Hakonen H., 2002. *Aspects of Networking in Multiplayer Computer Games*. *Electronic Library, The*, 20, no. 2, 87–97.

Smith G.; Anderson R.; Kopleck B.; Lindblad Z.; Scott L.; Wardell A.; Whitehead J.; and Mateas M., 2011. *Situating Quests: Design Patterns for Quest and Level Design in Role-Playing Games*. In *Interactive Storytelling*, Springer. 326–329.

Sullivan A.; Mateas M.; and Wardrip-Fruin N., 2011. *Making Quests Playable: Choices, CRPGs, and the Grail Framework*. *Leonardo Electronic Almanac*.

Tomai E.; Salazar R.; and Salinas D., 2012a. *A MMORPG Prototype for Investigating Adaptive Quest Narratives and Player Behavior*. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM.

Tomai E.; Salazar R.; and Salinas D., 2012b. *Adaptive Quests for Dynamic World Change in MMORPGs*. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 286–287.

Tronstad R., 2002–2003. *A Matter of Insignificance: The MUD Puzzle Quest as Seductive Discourse*. *CyberText Yearbook*.

Wardrip-Fruin N.; Mateas M.; Dow S.; and Sali S., 2009. *Agency Reconsidered*. *Breaking New Ground: Innovation in Games, Play, Practice and Theory Proceedings of DiGRA 2009*.

Zook A.; Lee-Urban S.; Riedl M.O.; Holden H.K.; Sottilare R.A.; and Brawner K.W., 2012. *Automated Scenario Generation: Toward Tailored and Optimized Military Training in Virtual Environments*. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 164–171.

ONLINE SKILL LEVEL CLASSIFICATION OF REAL-TIME STRATEGY GAME PLAYERS

Jason M. Blackford and Gary B. Lamont
Department of Electrical and Computer Engineering
Air Force Institute of Technology
2950 Hobson Way
Dayton, OH 45433, USA
email: gary.lamont@afit.edu

KEYWORDS
Player Model, Skill Level, Real-Time Strategy

ABSTRACT

This paper introduces a novel approach for measuring a real-time strategy (RTS) game player's skill level while the game is played. The method scores a player based on the time distance between strategic decisions made by the player during a single game session. The results of this work demonstrate that skilled players score within a specific range regardless of the strategy executed. Our approach can classify a player as skilled or unskilled for the StarCraft: Brood War game and the Spring Engine Balanced Annihilation game.

INTRODUCTION

Our investigation addresses the question of how to measure a real-time strategy (RTS) game player's skill level in executing a known strategy while the game is played. We address player modeling and two approaches for building RTS game player models. Finally, we introduce our strategy execution skill level metric Distance Between Decisions (DBD), and empirically demonstrate the potential of our metric to be used for measuring a player's skill level in the RTS games StarCraft: Brood War and Balanced Annihilation.

Background

In an RTS game, players execute a strategy to advance the skills and technology of their cities and armies in order to defeat their opponent. These strategic skill and technology advancements are defined in the technology tree of an RTS game. A technology tree is a large tree graph that encompasses all the strategic paths a player can traverse to defeat an opponent. Execution of a strategy in an RTS game is a planning problem consisting of two components: goal-ordering and build order execution (Blackford and Lamont 2014).

One way to view strategy execution is as a set of goals to be accomplished. A goal can be to collect a finite amount of resources (e.g. wood, gas, gold, etc) or to build a new building, military upgrades, or produce an attack wave. Good goal-ordering means laying out a plan of execution that takes into consideration resource allocations, dependencies between goals, and the total time to execute the strategy. For more on the importance of goal-ordering in RTS games refer to (Blackford and Lamont 2014).

As for speed of execution of a strategy, a player's build orders will determine how quickly the player reaches their goals. A build order is an ordered sequence of actions or decisions a player makes to execute their chosen strategy (Kovarsky and Buro 2006). To ensure skilled or near-optimal player performance, measured in resource allocations and time to reach a goal, good build orders must be executed by the player. We contend that the ability of a player to execute good goal-ordering and good build orders can be used to measure a player's skill level.

PLAYER MODELING

This section provides a brief description of player modeling and an overview of two approaches for building a player model in RTS games. Both approaches provide insight on how to begin creating a player model through the use of player logs and generating player features for classification. These discussions define the foundation of our approach.

A player model is an abstraction of a player defined in terms of their actions taken during gameplay. Player models allow game designers to determine a player's play style and make enhancements to the game state based on the identified play style (Drachen et al. 2009)(Weber et al. 2011). In an RTS game, a play style is defined differently from a first-person shooter (FPS). In an FPS, understanding a players play-style may simply come down to aggressive or defensive as observed in research conducted on Tomb Raider: Underworld by (Drachen et al. 2009). However, in an RTS game, a play style can reasonably be considered the type of strategy the player is executing. For RTS games, there are generally a small set of strategies a player can select. Each has a set of key actions that make the strategy unique from other strategies. For example, (Weber and Mateas 2009) identify six strategies in the StarCraft: Brood War game. A player cannot engage in an RTS game without choosing a strategy even if the strategy is a bad one.

Supervised Learning

The first RTS player modeling technique to be examined is a machine learning approach introduced by (Weber and Mateas 2009) for determining a player's strategy in an RTS game. The premise of (Weber and Mateas 2009) is to encode a player's RTS play style information, collected in logs during gameplay, into feature vectors, and then to use supervised learning algorithms to classify the feature vectors with respect to predefined strategies or labels. The importance of the feature vectors must be stated clearly. The vectors encode information on the player's path in the technology tree. A player's path is dictated by their strategy. Weber demonstrates successfully the ability of supervised learning algorithms to classify player strategies. The learning algorithms identify the player strategy as the feature vector is recreated over time simulating real-time, in-game classification of a player.

Support Vector Machine

Introduced by (Avontuur 2012) is the use of an optimized support vector machine (SVM) to classify player skill level against the StarCraft: Brood War player rankings. Avontuur uses 44 features derived from player log data to classify player skill level. These features include player actions, resources and timing information.

Avontuur groups all RTS player features for classification into several categories including: visuospatial attention and motor skills, economy, technology, and strategy. The visuospatial attention and motor skills category encapsulates the assumption that an expert player makes faster decisions than a non-expert player. A measure of this feature is the actions per minute (APM) or number of mouseclicks a player executes. The next feature category is economy which encodes the effectiveness of the player to gain resources needed to advance in the technology tree as well as support defensive and offensive operations. The technology category is focused on capturing the total number of upgrades the player commits, research advancements the player engages, and their level within the technology tree. The final category is strategy which encompasses the types of units and buildings the player creates as well as supplies used and gained. Avontuur demonstrates limited success in performing offline skill level classification using these feature categories and an SVM.

METHODOLOGY

Several assumptions are made for our approach. First, for any RTS game there is a finite set of known strategies players can choose to execute. Secondly, as discussed in (Avontuur 2012), skilled RTS players exemplify motor and visuospatial skills that exceed those of other skill levels, implying the ability to make faster strategic decisions. Lastly, given that executing a strategy skillfully entails good goal-ordering and good build orders, then it is reasonable to assume that a player's skill level can be measured to some degree by their ability to execute a strategy. From these assumptions we conjecture that by knowing a player's strategy and the timing of the strategic decisions to execute this strategy, a player's ability to execute the strategy can be estimated.

Actions Per Minute vs Strategic Decisions

For computing our metric, we utilize player log data that is captured during a single game session. The log data is used to build feature vectors that we apply our metric to in order to estimate player skill. The feature vector of each player includes strategic decisions only. We define strategic decisions as actions executed by the player that exist within the player's technology tree (e.g. unit type creation, upgrade, etc) and are determined by a specific RTS game strategy and it's goals. Our definition of strategic decisions is what distinguishes our approach from measuring actions per minute (APM). Actions per minute includes actions like clicking on units and giving commands - these are not strategic decisions but tactical decisions. We do not include in our computation every action or mouseclick a player executes. The feature vectors capture strategic decisions such as the construction of a first, second or third factory for the StarCraft Terran race or strategic upgrades to troops. These strategic decisions can be used to classify a player's strategy as demonstrated in (Weber and Mateas 2009) whereas APM cannot.

Distance Between Strategic Decisions

There are two types of DBD metrics: arithmetic DBD (1) and geometric DBD (2). The E represents the length of the original player vector or the number of strategic decisions defining the feature vector; N is the number of features equal to zero or the total number of strategic decisions that were not made by the player. The values D_{i+1} and D_i are the timestamps for the completion of an action $i+1$ which occurs after an action i. Timestamps are recorded as gamecycles which are computed by the frames per second times the number of seconds that have elapsed. When using geometric DBD (2) feature values of zero must be removed from the product.

$$\frac{(\sum_{i=1}^{E}(D_{i+1} - D_i)) + (\epsilon * N)}{(E - N)} \tag{1}$$

$$\sqrt[E-N]{(\prod_{i=1}^{E}(D_{i+1} - D_i)) * (\epsilon * N)} \tag{2}$$

Averaging features with a value of zero will drive the average down, thereby making an unskilled player appear skilled for being indecisive. To remove this issue, a penalty of ϵ game cycles for each zero, or non-decision, present in a player's vector is added. In addition, rather than divide by E for computing the average, only the total number of decisions the player actually makes, $E - N$, is used. The arithmetic DBD (1) provides a single value or score that shows which

player executed the strategy more quickly throughout a single game session when compared to another player's DBD score. On the other hand, the geometric DBD (2) provides a single value that captures the time consistency between strategic decisions in a players strategy execution. Given two players executing the same strategy, the player with the lower DBD scores is the more skilled player in executing the strategy.

EXPERIMENTATION

For experimentation we examined data from two RTS games including StarCraft: Brood War and Balanced Annihilation. We chose StarCraft: Brood War because of the numerous player data sets readily available and the results of (Weber and Mateas 2009) strategy classification investigation. Balanced Annihilation was used because it was apart of a larger investigation that defined and solved the RTS game multi-objective build-order problem discussed in (Blackford 2014).

StarCraft: Brood War Data Set

The RTS player game data used for experimentation is the same data used in (Weber and Mateas 2009). The data consists entirely of expert StarCraft: Brood War (SC1) player vectors. We limited experimentation to a single game race, the Terran race, and three of the six strategies for the Terran race as discussed in (Weber and Mateas 2009). The three strategies are commonly referred to as Bio, Fast Drop Ship, and Two Factory. The data includes 740 expert player vectors executing Bio, 66 expert player vectors executing Drop Ship, and 18 experts executing Two Factory for the Terran race versus the Zerg race. Each player vector is a feature vector that includes 50 strategic decisions a player can make in a single game session. In addition to the expert data captured by (Weber and Mateas 2009), we generated five beginner player vectors by playing the SC1 game as Terran vs Zerg executing the Bio strategy. These five vectors are the unskilled class. We utilized the SC1 analysis tool LordMartin Replay Browser (LMRB) to build our feature vectors from the recorded player log data.

Balanced Annihilation Data Set

For experimentation with the Balanced Annihilation (BA) game we used two agents. Agent LJD is a BA multi-scale AI developed at the Air Force Institute of Technology (AFIT) for the Spring Engine by (Trapani 2012). We consider agent LJD to behave at an expert level with respect to strategy execution because the agent's strategy manager is scripted by an expert player. Agent BOO (Build Order Optimization) is a modified version of agent LJD. Agent BOO utilizes a multi-objective genetic algorithm (MOEA), case-base reasoning (CBR), and a simulator as a strategic planning tool to discover and execute build orders (Blackford and Lamont 2014) and (Blackford 2014). We collected player data for agents LJD and BOO by having them execute the same strategies on the same maps against a non-existent opponent. The strategies executed are defined by (Trapani 2012) and include Tank Rush, Expansion, and Turtle.

RESULTS

Applying DBD to the SC1 data set reveals that expert players score within a specific DBD range regardless of the strategy they execute. With respect to BA, the DBD metric is able to identify the more skilled player when the same strategy is executed.

StarCraft: Brood War

The arithmetic and geometric DBD results of Fig. 1 and Fig. 2 include an ϵ of 1000 game cycles. A penalty of 1000 game cycles is close to the average window between strategic decisions. At 25 FPS 1000 game cycles is 40 seconds between strategic decisions. In addition, it empirically provides the best separation between the skilled and unskilled data as depicted in the histograms in Fig. 1. Depicted in Fig. 1 is a set of known expert SC1 players in blue and a set of unskilled SC1 players in red. By including penalty cycles, the separation of the data is very clear. This small example validates the utility of a penalty.

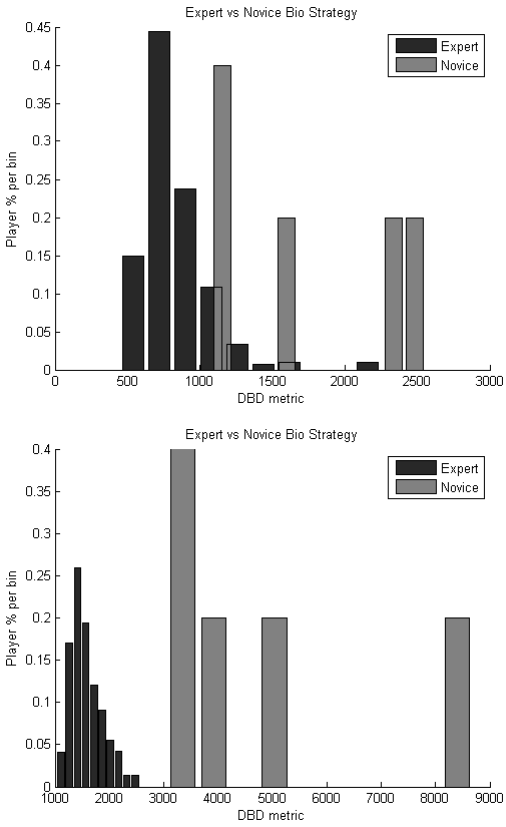


Figure 1: The top histogram demonstrates the separation without a penalty. The histogram on the bottom applies the penalty.

In Fig. 2 are the histograms that depict the distribution of expert players after the arithmetic DBD skill level metric (1) is applied to the player vectors. Observe that the means of the three strategies are within the range of 1466 to 1572 gamecycles with a standard deviation between 250 and 308 gamecycles.

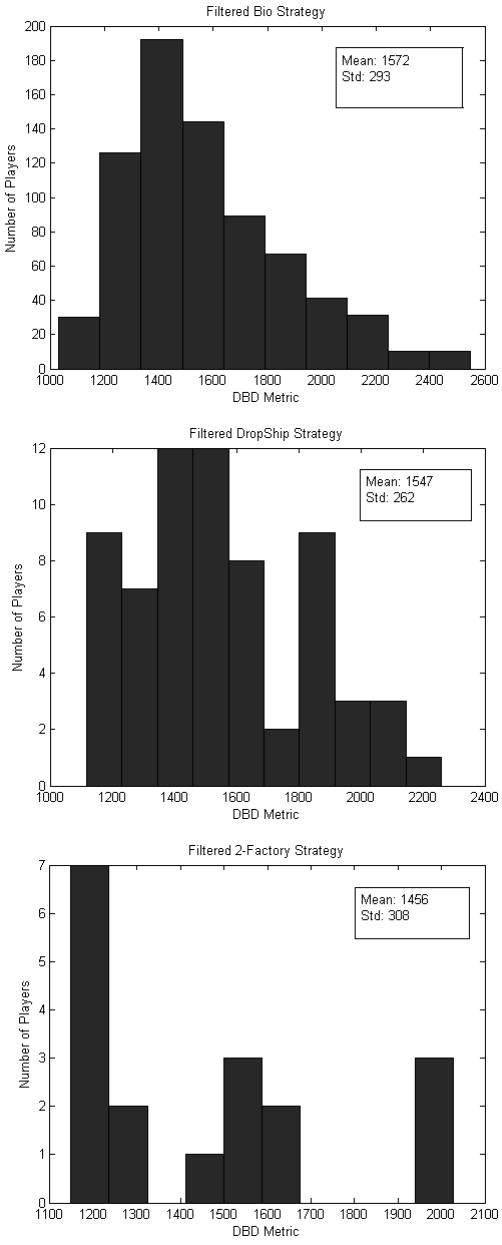


Figure 2: These histograms depict the distribution of the arithmetic DBD metric for expert players of StarCraft: Brood War.

From Fig. 3 it is clear that experts across the three strategies operate within a distinct geometric DBD range between 1.75 to 1.8. Notice that the arithmetic means of the geometric DBD for the expert strategies are all close to 1.8 and the corrected sample standard deviations are close to 0.2. This data reveals a consistency in expert strategy execution regardless of the strategy being executed.

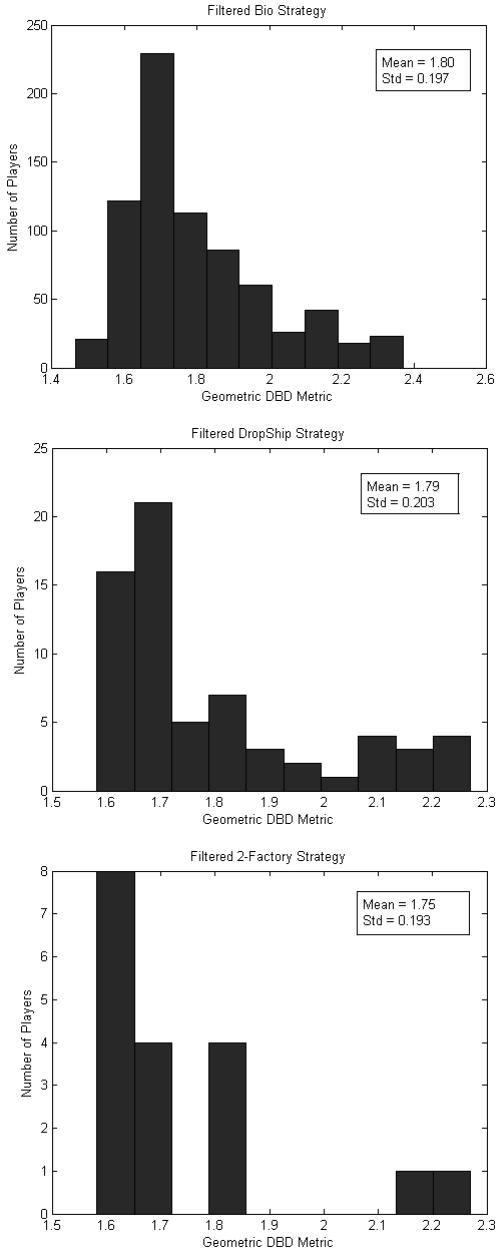


Figure 3: These histograms depict the distribution of the geometric DBD metric for expert players of StarCraft: Brood War.

Balanced Annihilation

The penalty, ϵ, is set to 1000 cycles. Charted in Fig. 4 is the build order timeline of the agents. The timeline displays the goals and goal-ordering of the Tank Rush strategy for each agent. The associated dark and light arrows with each goal relate to the timing of a build order executed by the agents. The images on the arrows depict the unit that was produced by the agent at a time given by the timestamps above each

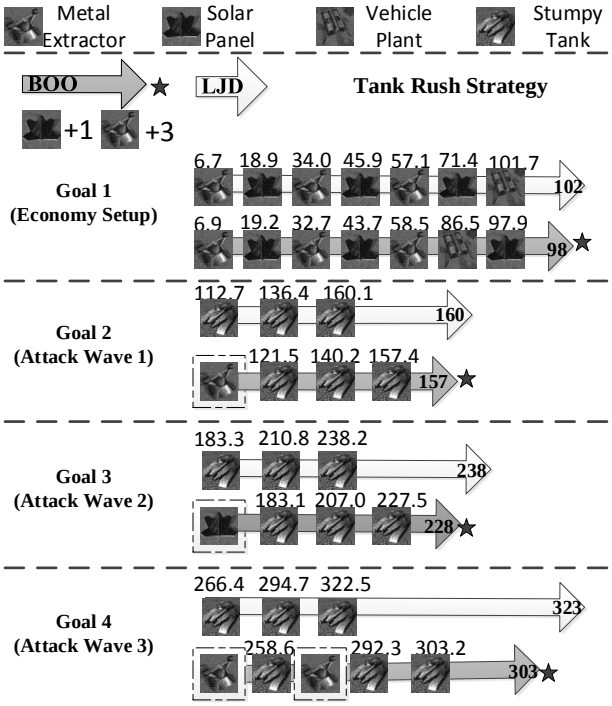


Figure 4: Build Orders for the Tank Rush Strategy.

image. The overall time to complete a goal is captured at the head of the arrows. The agent that reached the goal the fastest has a star next to their arrow. The arrows only identify the strategic decisions that define the Tank Rush strategy. Additional actions that are not strategic decisions are identified with a dashed box and are excluded from the DBD computation. If an agent produced more buildings or units than another agent while executing the same strategy, a + symbol and numeric quantity to the right of a unit is presented next to the agent's name at the top. From Fig.4, it is clear that agent BOO executes the strategy faster than agent LJD.

Table 1 depicts strategic goals for each strategy and the time in seconds when the agents reached the goal. The bolded timestamps in the table identify which agent reached the goal first. In Table 1, the arithmetic DBD values for the Tank Rush strategy agree with the conclusion that agent BOO completes the overall strategy execution faster than agent LJD. The geometric DBD also supports that agent BOO has a tighter timing consistency in executing the Tank Rush strategy.
For the Expansion strategy, agent BOO has a lower arithmetic DBD score but a higher geometric score. The lower arithmetic DBD reflects the fact that agent BOO consistently outperformed agent LJD in achieving five of the six Expansion goals first. The reason for the higher geometric DBD score is that agent BOO created seven additional units not defined by the strategy as strategic decisions. These additional actions introduced delay time between strategic decisions and are reflected in the geometric score. Since agent LJD only executed the necessary strategic decisions dictated by the strategy, it has a lower geometric score. This is also true for the Turtle strategy where agent BOO created eight additional units more than the strategy required.

Table 1: DBD Scores. Goal(G); Turtle (TR); Expansion (Ex); Turtle (Tu); DBD is Arithmetic/Geometric

Agent	Strat.	G1	G2	G3	G4	G5	G6	A/Geo DBD
LJD	TR	102	160	238	323	-	-	18.71/1.12
BOO	TR	**98**	**157**	**228**	**303**	-	-	**15.69/0.94**
LJD	Ex	219	429	573	643	724	**839**	12.9/**0.98**
BOO	Ex	**173**	**356**	**519**	**611**	**706**	848	**12.8**/0.99
LJD	Tu	120	267	434	559	622	865	13.73/**0.978**
BOO	Tu	**119**	**231**	**428**	**546**	**618**	**834**	**13.02**/0.985

CONCLUSION

We have provided interesting scoring metrics for estimating player skill level for RTS games including StarCraft: Brood War and Balanced Annihilation. The DBD metric can be utilized to estimate a player's ability to execute a known strategy. In the future, we intend to explore the use of a larger data set containing a variety of skill levels with the objective of determining DBD thresholds between the skill levels and classifying a player in real-time.

REFERENCES

Avontuur T., 2012. *Modeling player skill in Starcraft II*. Master's thesis, Tilburg University.

Blackford J., 2014. *Online Build-Order Optimization for Real-Time Strategy Agents Using Multi-Objective Evolutionary Algorithms*. Master's thesis, Air Force Institute of Technology.

Blackford J. and Lamont G., 2014. *"The Real-Time Strategy Game Multi-Objective Build Order Problem"*. In *10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 105–111.

Drachen A.; Canossa A.; and Yannakakis G.N., 2009. *"Player modeling using self-organization in Tomb Raider: Underworld"*. *2009 IEEE Symposium on Computational Intelligence and Games*, 1–8.

Kovarsky A. and Buro M., 2006. *"A First Look at Build-Order Optimization in Real-Time Strategy Games"*. In *Proceedings of the GameOn Conference*. 18–22.

Trapani L.D., 2012. *A Real-time Strategy Agent Framework and Strategy Classifier for Computer Generated Forces*. Master's thesis, Air Force Institute of Technology.

Weber B. and Mateas M., 2009. *"A data mining approach to strategy prediction"*. *2009 IEEE Symposium on Computational Intelligence and Games*, 140–147.

Weber B.; Mateas M.; and Jhala A., 2011. *"Using Data Mining to Model Player Experience"*. In *Proceedings of the FDG Workshop on Evaluating Player Experience, 2011*.

AUTHOR LISTING

AUTHOR LISTING