Tampereen teknillinen yliopisto. Julkaisu 1237
Tampere University of Technology. Publication 1237

Juha-Matti Vanhatupa

Tool Support for Computer Role-Playing Game Programming: Foundations, Guidelines and Applications

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB222, at Tampere University of Technology, on the 17th of October 2014, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2014

ISBN 978-952-15-3341-9 (printed)
ISBN 978-952-15-3393-8 (PDF)
ISSN 1459-2045

Abstract

Computer role-playing games (CRPGs) are a genre of computer games, which aim at providing similar player experience than their ancestors, paper-and-pen role-playing games. For a type of digital games, their evolution is already rather long, as the first ones were created in 1980s. Typically CRPGs emphasize character development and to support this huge fantasy worlds and sophisticated storylines are also present. CRPG development has unique challenges, which derive from these typical features. Content creation is a continuous issue, since huge virtual worlds and long storylines must be filled with content. Personalization is also an important issue, because all fun of creating personalized character is lost if it has no effect into the game.

Low starting threshold is important for successful game. It is becoming essential that the player can start playing quickly and she is not required to spent time waiting the installation to be completed. This can be achieved by web-based approach, since web-based games do not require installations. There are also other benefits, such as instant distribution, platform independence, and instant upgrades. As it is likely that in the future CRPGs are web-based, software tools for developing browser-based games are even more important in the future. However regardless of the platform the unique challenges of CRPG development remain the same.

This dissertation addresses these unique challenges and simplifies CRPG development with software tool support. The software tools presented in this dissertation begin with construction of a novel game engine called CAGE. We used this game engine in the game programming course of Department of Software Systems in Tampere University of Technology as a software platform, and students developed their games using it. Several content generators

were created during the research to tackle this challenge. Using those, content, objects, non-player characters, and even quests can be generated into a game world. The last phase of the research targeted browser-based environment, and this dissertation includes surveys of browser-based games and technologies. In this part, we define attributes of a game engine for browser-based games. Finally, we present a prototype browser-based CRPG using content generation.

Preface

This thesis has truly been a quest, a journey across a symbolic, fantastic landscape, an opportunity to overcome challenges and achieve a meaningful goal. Luckily I didn't had to go through it alone and I would like to thank the following people.

First of all, I would like to thank my supervisor Professor Tommi Mikkonen for making this thesis possible in the first place. I thank the pre-examiners of this thesis, Associate professor Julian Togelius and Professor Ian Parberry. I also thank my opponents Senior Lecturer Ian Kenny and Professor Markku Turunen. I like to thank my co-authors Teemu J. Heinimäki and Janne Lautamäki for their excellent work. In addition, I like to thank Professors Kari Systä and Mikko Tiusanen for proofreading and valuable feedback.

I would like to express my gratitude to my wife Laura, and our army of cats. In addition, I like to thank family and friends for their support.

The work has been funded by Graduate School on Software and Systems Engineering (SoSE), Ulla Tuominen Foundation, the Nokia Foundation, The Finnish Foundation for Technology Promotion and Science Foundation of the City of Tampere.

Now at the bottom level of the Dungeon I am ready to encounter the final opponents.

Juha-Matti Vanhatupa
Tampere, August 18, 2014

Contents

List of Included Publications

[I] J-M. Vanhatupa and T. J. Heinimäki. Scriptable Artificial Game Engine for Game Programming Courses. In *Proceedings of 4th Informatics Education Europe (IEE IV)*, pages 27–32. University of Freiburg, November 2009.

[II] J-M. Vanhatupa. Guidelines for Personalizing the Player Experience in Computer Role-Playing Games. In *Proceedings of 6th International Conference on the Foundations of Digital Games (FDG'11)*, pages 46–52. ACM, June/July 2011.

[III] J-M. Vanhatupa. Generative Approach for Extending Computer Role-Playing Games at Run-Time. In *Proceedings of 11th Symposium on Programming Languages and Software Tools and 7th Nordic Workshop on Model Driven Software Engineering (SPLST'09)*, pages 177–188. Department of Software Systems, Tampere University of Technology, August 2009.

[IV] J-M. Vanhatupa. Personalized Side-Quest Generation for Computer Role-Playing Games. In *Proceedings of 12th Symposium on Programming Languages and Software Tools (SPLST'11)*, pages 196–206. Institute of Cybernetics at Tallinn University of Technology, October 2011.

[V] J-M. Vanhatupa. Browser Games for Online Communities. In *International Journal of Wireless & Mobile Networks (IJWMN), vol. 2, no. 3, 2010*, pages 39–47. August 2010.

[VI] J-M. Vanhatupa. On the Development of Browser Games – Current Technologies and the Future. In *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), vol. 5. 2012*, pages 60–68. June 2012.

[VII] J-M. Vanhatupa and J. Lautamäki. Content Generation in a Collaborative Browser-Based Game Environment. In *Handbook of Digital Games*, pages 92–110. Wiley-IEEE Press, May 2014.

The permissions of the copyright holders of the original publications to reprint them in this thesis are hereby acknowledged.

Chapter 1

Introduction

The history of computer games begins with early games like OXO (1952) and Spacewar! (1962). During their evolution, computer games have evolved from these simple games to rich and complex creations. As games have become more complex, the amount of required software has constantly grown. Similarly, the sizes of developer teams have grown constantly in computer game projects. A typical time span of development from an idea to a completed product is about two years and requires work contribution of 20–50 persons [67].

Already in early days of computer game development, software tools were developed to assist in the creation of computer games. Nowadays, different kinds of software tools, such as game engines, modelling tools, and code generators, are an essential part of game development. Tools created for developing a particular game can also be reused when making other similar games.

This thesis focuses on computer role-playing game (CRPG) genre. Games of this genre have unique software development challenges, which derive from huge game worlds and sophisticated storylines. Therefore, software tools are especially important in the development of computer role-playing games (CRPGs). In addition, the border between CRPGs and strategy games can be a shallow one, and term strategic role-playing game is also used for games not clearly belonging to only one of the groups, and many strategy games

contain role-playing elements. For example, it is common that individual heroes in player armies can gain experience.

The background part of this thesis presents the history and origins of the CRPG genre. CRPGs derive from their ancestors paper-and-pen role-playing games (PnP RPGs); and aim at providing similar personalized player experiences as those. Emulating the personalized player experience of PnP RPGs is a unique challenge of the genre, since in PnP RPGs human game master controlling the game can easily create personalized adventures and also apply game rules as needed to support a fluent game flow. However, for a computer program this is a much harder task, as the program requires explicit instructions. CRPG have also similarities to modern board games, miniature war games, and collectible card games. Many themes, characters, and rules used in CRPGs are familiar to the players of those games.

Early CRPGs were often developed by a single programmer and equipped with ASCII graphics. Although these games were competing with games like Pac-Man and Space Invaders, an early CRPG Rogue, for example, was very popular at 1980s, and in a university a network-wide analysis of system usage revealed that Rogue was consuming more CPU cycles than anything else [73]. Through their over three decades long history CRPGs have retained their popularity and gathered enthusiastic players. They have evolved into huge software projects. The use of sophisticated computer graphics and advanced artificial intelligence (AI) is also common in modern CRPGs.

1.1 Motivation

Throughout their history CRPGs have gathered enthusiastic players. The best CRPGs require a diverse set of skills, such as tactics, long-term planning, problem solving, teamwork, and resource management [7].

The soul of any role-playing games (RPGs) is a personalized player character. Depending on the platform of RPGs PnP RPGs, CRPGs, or live action role-playing games (LARPs), the character is created and controlled differently [82], but all together one of the main ideas of the game is to experience adventures in a fantasy game world by controlling this imaginary

character. Many RPGs are collaborative games and players work together to accomplish their goals. Although in a single player CRPG other characters are controlled by the game, it still creates an illusion of collaborative game through computer controlled non-player characters (NPCs).

CRPGs can offer the player experience of PnP RPGs for players that do not have suitable friends to form a play group or who do not have regular hours to spend on playing. Organizing a traditional role-playing session can be very difficult for working adults, since those last for hours and everybody in the group should be able to participate at the same time. Multiplayer CRPGs played on the Web help players find player groups ignoring physical locations, and many support building *ad hoc* player groups or solo playing. Some people are also interested in game worlds and fantasy present in role-playing games, but lack interest in the social side of the games. Therefore playing a single player CRPG can be the best option for them. CRPGs also allow the player to concentrate on management of the player character as much as he or she wants, even though in PnP RPGs that would annoy other players and disrupt the game flow.

Similarly to their precursors PnP RPGs, CRPGs contain huge fantasy worlds inhabited by hundreds of non-player character. In addition, long, sophisticated storylines are typical for both game types. However, in CRPGs these fantasy worlds are presented as a game artefact, a virtual world, instead only imagining them.

Building the fantasy worlds and sophisticated storylines present in CRPGs is an enormous task for game developers. In addition, software sizes of CRPGs have grown since the birth of the genre and this development continues. While the first CRPGs were text-based games programmed by a single programmer, modern CRPGs contain of rich game world presented with computer graphics and video, and also over forty hours of game time [13].

Software tools are the main weapon in the battle against the enormous burden of software developers. Tools can either save time in producing game code, enabled by different kinds of code generators, or by using tools reuse of existing code can be achieved, as by using game engines.

Game engines [25], in other words software product lines [12], [16] used

to build computer games, are currently the most important tools in game development. Modern computer games are very rarely created from scratch, but a game engine is used as a basis instead. Parts of previous games developed by the same company can also be used, especially if the games are created using the same game engine. Reuse can also be achieved if the used tools output a general format, for example XML [23], and resources created using these are used in several games.

In the field of software engineering, programming languages, tools, and methods used to develop software are always changing. Currently, we are experiencing a paradigm shift towards web-based software [71] where applications live on the Web. They are accessed using browser and their resources can be located anywhere in the world. They do not need any installation, or manual upgrades. In addition, the model also allows user collaboration, multiple users accessing the same resources, interacting together. It is most likely this paradigm shift will also affect computer games as well. Many of these new features are very advantageous for computer game development. For example, instant upgrades allow adding new content to the game over the network and the described user collaboration model allows building multiplayer more easily.

This paradigm shift requires also creation of new kind software tools for building computer games. For example, game engines for browser-based games have just appeared. This ongoing development will bring interesting times for game programmers and game designers as well.

1.2 Deriving Research Questions

Creating a modern CRPG is a long and complex project requiring a large development team with experts of many different professions. In addition the development requires different kind of software tools. Consequently, the first research question this thesis addresses is the following:

RQ1. What kind of tool support is advantageous to support software developers' work when implementing CRPGs?

In addition, to development challenges derived from the substantial size of developed software and development project, there are unique software development challenges that are specific to the CRPG genre. CRPGs contain large virtual worlds with numerous NPCs, which means that a lot of application developers' time must be spent on implementing the game world and its characters.

Also creation of personalized player experiences emulating PnP RPGs take also lot of application developers' time. CRPGs aim at providing similar personalized player experience as PnP RPGs. The personalized player experiences are one of the most difficult properties of PnP RPGs for CRPGs to emulate. In PnP RPGs the human game master controlling the game is familiar with player characters and can apply rules as needed to create personalized player experiences. However, a computer requires explicit instructions. This challenge of creating personalized player experiences is the concern of RQ2.

RQ2. How to simplify and support the creation of personalized player experiences in CRPGs?

Since the whole software industry is experiencing the paradigm shift to the Web, most likely the browser will be also the main environment of CRPGs. New kinds of tools are needed in the browser environment. Consequently the third research question is the following:

RQ3. What kind of tool support can be used to assist in the development of CRPGs for the browser-based environment that is constantly alive online?

The research questions are addressed in the included publications as follows. RQ1 is addressed in Publications [I], [III] and [IV]. RQ2 is addressed in Publications [II], [IV] and [VII]. RQ3 is addressed in Publications [V], [VI] and [VII]. Table 1.1 summarizes how publications address different research questions.

Table 1.1: Publications addressing research questions.

	1	2	3	4	5	6	7
RQ1	X		X	X			
RQ2		X		X			X
RQ3					X	X	X

1.3 Research Methodology

This research has been carried out as a constructive research [38]. Constructive research tests theories and proposes solutions to a problem or question. The term construct refers to the new contribution being developed. In computer science these constructions can be theories, algorithms, models, software or frameworks. Commonly a solution is proposed by building prototypes with preconditions.

In the artifact building and evaluation we have carried out action design research (ADR) [46], which includes the following phases: 1) problem formulation, 2) building, intervention, and evaluation 3) reflection and learning, and 4) formalization of learning [46]. At first the problem to be solved is defined. The premises created in the first phase, provide a platform for building software artifact in the second phase. The built artifact is also evaluated in this phase. In the third phase solving the problem moves conceptually from building a solution to particular problem instance to applying that learning to a broader class of problems. The problem and solution are continuously reflected. This state is executed parallel to first two stages. The purpose of the fourth stage is to formalize the learning. When we have the solution to a particular problem, both of these are generalized, to create generalized outcomes.

In this thesis we have built software artifacts and evaluated these to determine if requirements for the built artifacts are fulfilled. In publications included in this thesis we present these software artifacts as solutions to problems presented in the introductory part. Instead of completely solving these problems, presented prototype tools are useful accessories which par-

tially solve the problem or assists the application developer dealing with the problem. The usefulness and benefits of tools are presented in particular publications.

1.4 Contribution

The main contributions of this thesis are:

1. The constructive design of the CAGE game engine [I], an experimental research artefact
   - **(a)** to be an easily understandable platform for students building their game projects, and
   - **(b)** its evaluation in a classroom environment.
2. Software tools to simplify and speed up development and deployment of CRPGs.
3. A model how these tools should be used.

Tools presented in this thesis began from a single player game engine and advanced to describing a prototype game engine for browser-based games. In the last publication [VII] of this thesis, a prototype browser-based multiplayer game using content generation is presented. The examples of game design and content generation chapters can be applied to computer games built on both binary and browser-based platforms.

One of the main principles of this thesis is to give practical support and new ideas for game developers. Ideas of developed tool support presented in the publications should be usable by reading those publications. Game programmers using different platforms can benefit from this thesis.

1.5 Structure of the Thesis

The introductory part of this thesis is organized as follows. Chapter 2 presents background of CRPGs. It addresses other game types close to

CRPGs, such as PnP RPGs and LARPs, presents origins of CRPGs, and introduces fundamental elements of CRPGs and development process of them. Chapter 3 discusses software tools used in game development. Chapter 4 deals with run-time environments of CRPGs. Chapter 5 is an introduction to the included publications. Chapter 6 presents conclusions, and revisits presented research questions.

Chapter 2

Computer Role-Playing Games

This chapter discusses the CRPG genre and presents special challenges relating to the development of CRPGs. These special challenges derive from the unique nature of CRPGs. These include the burden of content creation and the problem of personalizing game content.

CRPGs are computer games that aim at offering similar gaming experience as their precursors, PnP RPGs. First CRPGs were developed at late 1970s, and thus the genre has already had a long evolution for a type of digital games, [7]. Important elements of a CRPG include storyline, virtual world inhabited by characters, and the fact that the results of character actions are resolved using game mechanics (rules).

2.1 Paper-and-Pen Role-Playing Games

Paper-and-pen role-playing games evolved from war games during 1970s. When in war games players controlled entire armies represented by figures or other markers, in PnP RPGs each player controls a single character. All times best-known PnP RPG Dungeons & Dragons (D&D) was originally published in 1974 [81]. D&D is a heroic fantasy game, where players are heroes seeking adventures in a medieval fantasy world that contains monsters and magic. The game introduced many important factors, such as attribute scores: strength, intelligence, wisdom, dexterity, constitution, and

charisma, which are used in several contemporary games, four decades later. Today D&D is still very popular, and new versions of the game are published. Another famous early product was a science fiction PnP RPG Traveller [79] published in 1977. New versions of Traveller are still published. Today there are hundreds of PnP RPGs, and new ones are published each year.

There are PnP RPGs from almost all the themes that can be imagined. Two particular themes have been especially popular, fantasy and science fiction. Many fantasy themes feature similar settings as in Tolkien's books. In science fiction, different space opera themes, the contrary of hard science fiction, have been popular.

Terms *paper-and-pen role-playing games* or *tabletop role-playing games* are normally used to distinguish these actual role-playing games from other forms of RPGs, such as CRPGs and LARPs. In fact, neither a pen, paper, nor table are mandatory for playing a role-playing game. PnP RPG is played by a group, normally from two to six players. Each player except the game master controls his own character.

The player character is an avatar of the player in the imaginary world of the game. The player character is a main aspect of RPGs. Depending on the game its creation can be done very quickly or it can be an even hours long, complex process, including writing a background story for the character. The character creation process is done jointly by the player and the game master, and usually this does not happen during a regular game session but before it. The player has usually only partial control over what kind of character he or she can create, as part of the details are defined in the game mechanics and part by the game master. It is a duty of the game master to ensure that the characters are a suitable for the game campaign and also that the player group has suitable combination of characters from different professions. The character creation process usually involves throwing dice to give values to attributes and other factors, and also selection careers and skills. In Classic Traveller PnP RPG the player character could even die during the character creation process.

A core characteristic of PnP RPG is the freedom to choose the actions and attitudes of the characters, which they control. This freedom of actions

is also difficult to emulate in CRPGs. Although freedom of movement in game places is easy to implement, things get complicated for implementing this freedom for example in dialogues, since usually conversations must be pre-scripted.

Instead of an individual character, the game master controls the game. He or she acts as a narrator and presents the scenario to the players. The game master interprets game rules and can also disregard some rules if those do not support a fluent game flow. The game master has wide range of responsibilities, and he or she is also the supporting force behind the game group [26]. The game system in PnP RPGs is only as good as the ability of the game master to understand and present it to the players. The results of the character actions are resolved using game mechanics, different kinds of dice are commonly used in the process. In addition, miniature figures are sometimes used to present characters and monsters. Those can especially useful in representing combat situations. Figure 2.1 presents miniature figures from war game Warhammer Fantasy Battle [91].

Although normally a computer takes care of the responsibilities of the game masters in CRPGs, a human game master can be used in some CRPGs. Game masters have different kinds of responsibilities depending on game media (PnP RPGs, CRPGs, LARPs) [83]. However, the lack of proper tools limits the use of human game masters in CRPGs. They are, for example, limited to pre-created content and cannot create adventures dynamically at run-time, although dynamic game-time content creation is one of the main functionalities of PnP RPGs game master.

2.2 Other Game Types Close to PnP RPGs

LARPs are another form of RPGs. In a LARP the players dress and act as their characters. LARPs require larger player groups, the size of which can vary from a handful to hundreds. LARPs can happen in public or private areas. Although there are usually no spectators present, for example in games with agent theme, the presence of real people and public places are a common

Figure 2.1: Miniature figures

aspect. There can be limitations; for instance the players can be required to avoid combat in public. A single LARP can last for hours or days.

In LARPs the game master cannot control all events as many are happening simultaneously, and the role of the game master is more focused on organizing the event. The game masters can also play some character at the same time. The game mechanics of LARPs determine how the players can affect the game world, for example, how the combats are resolved. This can include battling with foam weapons, pausing the game play for dice rolling, comparing the character attributes, or determining the winner of the battle in some other method.

Collectible card games are also games that relate to the history of RPGs. These games are played using special decks of cards that the players build before the game, according to precise rules. The deck building process, design and gathering the cards, by trading them with other players or buying them, is a considerable part of the game. Magic: The Gathering [78] is one of the

first collectible card games. It was introduced in 1993, and it had over twelve million players 2011 [44]. Notable Magic cards can be worth of hundreds of dollars. Although these games are not RPGs, there are many common elements, for example, fantasy themes and characters. Figure 2.2 presents Magic: The Gathering game in action.

Figure 2.2: Magic: The Gathering game

Many board games also feature fantasy themes, and in those player control individual characters similarly to PnP RPGs. However, a game master is rarely used in board games, although in some board games one of the players controls the enemies or opponents of the players. Examples of such board games are Doom: The Boardgame [20] and Space Hulk [68]. In addition, character creation is rarely present at all. Instead, all players have the same attributes or the characters are premade and chosen or distributed randomly to the players. Some board games also include scheming and betrayal of other players. For example, in Shadows over Camelot [65] the players control knights in medieval England and work together to protect Camelot. However, one or more of the players are traitors who are secretly playing against the group.

2.3 Origins of Computer Role-Playing Games

First CRPGs were developed late 1970s. Although the influence of PnP RPGs and war games are vital, Barton lists also other sources that have contributed to the birth of CRPGs: sport simulation games, writings of J.R.R. Tolkien and Colossal Cave Adventure, the first true computer adventure game [7]. Early simulation games that were popular 1960s and 1970s used combinations of cards and dice to simulate sports. Those games had many common elements with PnP RPGs, including attributes and statistics of the player. Books of J.R.R Tolkien, published in 1950s, became widely available to mass market in 1960s, were more detailed and complex than earlier fantasy novels. The popularity of Tolkien's books created room for both PnP RPGs and CRPGs. Colossal Cave Adventure by Will Crowther was the first adventure game [7]. The game consisted entirely of text and was played by typing textual commands. Although Colossal Cave Adventure was not CRPG, it included several features that later became important for CRPGs, such as exploring a virtual world, magic, and monsters. Later, multi-user dungeons (MUDs) offered the players a chance to explore virtual worlds together [7].

One of the most famous mainframe CRPGs, Rogue was implemented in early 1980s. The graphics consisted entirely of text, using the alphabet and different symbols to mark different items and monsters. Perhaps the most important feature of Rogue was that the game used randomized dungeons: the game world is different each time the player starts the game. This introduces replayability value that modern CRPGs players only dream of. Based on Rogue, hundreds of roguelike CRPGs have been created. Nethack, a later version of Rogue published in 1987 is still being developed, although new versions are published at a slow pace.

Different tools were developed to support legendary CRPGs. When the first multiplayer dungeon crawl arcade game Gauntlet was developed, the System I of Atari needed to be modified to support four-level circuit boards. The original extension board of Gauntlet threatened to be 28 x 20 inches, which could not be fitted inside a video arcade machine. At the time Gauntlet

was published in 1985, current software engineering electronics could not draw boards of that size, since regular extension board size was 18 x 18 inches [32].

Late eighties, 1987, a real-time game Dungeon Master was published. It was the time when turn-based game flow was almost a standard in CRPGs. It remains the most successful game released on Atari ST platform. Dungeon Master success can be explained by the storyline, which was made by a professional novelist Nancy Holder. She based it on the base scenario suggested by game designers. In addition the game featured a 3D interface combined with real-time game play [7].

Although the beginning of the nineties was a difficult time for the CRPG genre, many classic first person view dungeon exploration CRPGs were published. For example, Eye of the Beholder trilogy (Eye of the Beholder, Eye of the Beholder II: The Legend of Darkmoon, and Eye of the Beholder III: Assault on Myth Drannor) has been popular [7].

At late nineties, one of the most famous titles of CRPGs, Baldur's Gate, was published by BioWare. It used Advanced Dungeons & Dragons' 2nd edition rules, and it was a successful transformation of a classic PnP RPG into a sophisticated CRPG. It was also a quality CRPG published at time when the whole CRPG genre was in ebbing spirits. Baldur's Gate featured especially long game play, over 100 hours. Baldur's Gate was followed by Baldur's Gate II and also extensions packages for both games were released. Baldur's Gate was also a notable release when considering software tools, since it used the Infinity engine, later used by eight other CRPGs. Moreover a first person shooter (FPS) Quake had used a game engine few years earlier.

2.4 Elements and Nature of CRPG

Important elements of a CRPG include a storyline, a virtual world inhabited by characters, and the fact that results of character actions are resolved using game mechanics (rules) [II]. In most modern CRPGs the player controls a single character. However, he can hire or otherwise collect other members to the party.

CRPGs emphasize character development, which is often referred to as the main characteristic of CRPGs. When the player advances in the game storyline, the player character becomes more experienced, gains more abilities or skills, and possesses better items. Experience points are a common method to model character development; each time the player character wins an enemy or succeeds in performing some other meaningful task, he gains experience points. When enough points are gathered, a new level is gained. With each new level the character gains new abilities, skills or other benefits. Some CRPG also include skill trees, where the skills are bought. Often, all possible skills cannot be picked, and the player must choose on what to specialize.

CRPG genre definition is somewhat overlapping with strategy games, and these two genres have many similarities in themes and other characteristics. Many strategy games also use experience points and experience levels. In those, individual units can gather experience and therefore become more efficient by gaining levels. In addition, many CRPGs are party-based: the player controls several characters and the player can hire or remove new members. Therefore, these games totally lack the main character whom the player should identify himself with.

CRPGs are a difficult genre for game developers, since the genre has huge player expectations. In a regular first-person shooter game, the player does not expect to be able to talk to the enemies [62]. Usually he does not even expect to be able to buy items in a game. However, from a CRPG the player expects an epic story that happens in a live fantasy world. He wants to have an effect to the story instead of just walking through it also. Talking to NPCs and buying items from shops have been basic features of CRPGs since 1980s!

The game worlds of CRPGs are huge, and inhabited by different NPCs. Therefore, implementing realistic and believable characters is a demanding task for application developers. CRPGs do not always succeed fully in creating believable characters [40]. Believable behaviour of NPCs is also an important aspect of personalization of the game, since NPCs should treat the player character in an appropriate manner. These personalization aspects are discussed in Publication [II].

The desire to mimic PnP RPGs can be spotted from several factors in CRPGs. For example, damage inflicted by different weapons or spells are commonly described by the sums of different dice, like D4, D6, D8 or 2D6 (1-4,1-6,1-8 and 2-12), although the possible physical shapes of dices do not limit the numbers that could actually be used in CRPGs. Also the fantasy settings of CRPGs are usually very typical for the genre, with the same races (e.g., humans, orcs and elves) and professions (e.g., warriors, wizards and knights) as in RPGs, because the players are already familiar with those.

2.5 Storyline and Quests

The storyline of CRPG contains quests. Quests are individual tasks for the player character to solve. By completing quests the player character can advance in the game storyline. Some of the quests are optional (side-quests) and do not relate to the game storyline. By completing side-quests the player character can gather experiment points, better equipment, or meet new NPCs. Completing some of these side-quests can even hinder advancing in the game storyline, although the reward from completing those can be so valuable that the player character decides to complete them anyway. Side-quests can also offer important information regarding the game world, and therefore they can be used to support the storyline of the game [27]. For example in Dungeon Siege the main theme of the game are the battles raging across the kingdom, and through quests the game designers try to show to the player how the battles affect the lives of both soldiers and commoners [27]. In addition to the quests, the storyline of CRPG includes tutorials and endings. Quests that are mandatory to solve for completing the game belong to critical paths. Originally CRPGs contained only a single critical path, but modern CRPGs can contain several. Figure 2.3 shows a CRPG storyline structure with a single critical path. Using multiple critical paths increases replayability, since part of the game is completely unplayed in the second playtime. Figure 2.4 shows a storyline structure with multiple critical paths.

In CRPGs quests have evolved into the most important tool when guiding

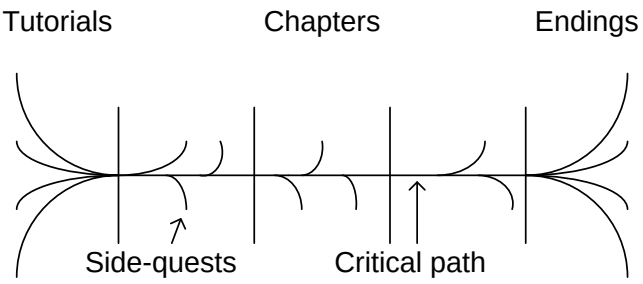

Figure 2.3: CRPG storyline structure with a single critical path

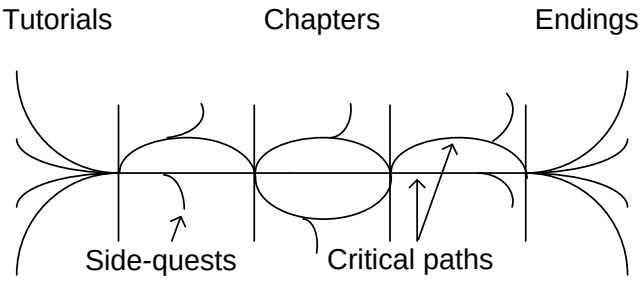

Figure 2.4: CRPG storyline structure with multiple critical paths

the player character through the story [33]. In a personalized CRPG quests and even critical paths are opened and omitted based on the player character decisions and also based on his or her player character. Different attributes of the player character, such as character class, feats, or even gained special equipment, can open or omit new quests.

Quests can be linear missions, or include complex puzzle solving [1]. Since combat systems of CRPGs are robust and combat-based quests are easy to implement, CRPGs rely mostly on those [70].

Although quests are present also in PnP RPGs, they are usually not as clearly defined, since players gain different kinds of tasks to solve. In an individual game scenario, the players have goals and they also have some long-term goals. A scenario of PnP RPGs usually lasts for one or more playtimes. The long-term goals of the players can relate to the campaign or a player character can have personal motives.

There are certain PnP RPGs where quests are strictly defined. For example in Paranoia [58] the objectives for each playtime are given by a high-

ranking NPC at the beginning of the playtime, although there these quest objectives are usually secondary as the players try to kill each other without getting caught. Another difficult task in the world of Paranoia is survival, as player characters are easily killed and the game world is extremely hostile. Although there are five copies (clones) of each player character to be used when the original character dies; some editions also let the player character to purchase new copies.

2.6 Game Development Teams

Computer game development teams contain wide range of professionals of different kinds. Usually a game development team contains 20 - 50 persons with various responsibilities [67]. The core of a game development team is formed by professional game programmers, whose job is to implement the game and software tools that are used to build the game. In addition to game programmers, game development teams contain game designers, artists, plot writers and testers to name some common activities.

Game programmers are responsible of implementing game software and tools used to build the game. Game programmers are specialized in implementing a certain part of the game, like AI, user interface (UI), software tools, or gameplay (game mechanics and other features relating playing the game). Senior game programmers can also have management responsibilities, like time scheduling, in addition to code writing [25]. Some companies have also one or more technical directors, who supervise one or more projects at the high level [25].

Game designer is responsible of creating game rules and imaging the virtual world where the game takes place. He or she is already involved before the game implementation is launched by writing the presentation documents. All game designers are not programmers, and game design itself does not require programming skills, only the understanding of what can be done with the current technology and how long time and how much other resources it takes to create it.

Plot writers create the game plot. Their responsibility is close to that

of a game designer; however, they are not involved in the game mechanics. They write character backgrounds, dialogue, and game plot. Their work is quite similar to that of a scriptwriter of a film.

Artists are responsible of creating the visual image of the game. The artist team is usually led by an art director whose responsibility is to maintain the overall vision and ensure it is followed by the artist team. A job of an individual artist can be 2D or 3D oriented. A 2D oriented artist creates sprites, textures, concept art, background images, or user interface elements. A 3D oriented artist can create 3D models, 3D environment, animations, cut scenes (video episodes inside the game) or take care lightning of the game world. A 3D artist creating 3D models used in the game can also be called *modeller*. A part of 3D oriented art work, for example, creating the cut scenes, can also be subcontracted.

Level designer creates game levels using specific tools. Since these sophisticated tools are used in level design, level design does not require advanced programming skills. Level design can also include testing AI of NPCs moving in the level.

Sound engineer or *composer* is responsible for creating sound effects and positioning sounds into the game. They can also have responsibilities in managing characters' voice acting.

Testers are an important part of any software development project. Testers verify that the end product meets the requirements. In modern CRPGs huge code base and graphical user interface (GUI) complicate testing process. In addition, use of content generation complicates the situation even more, since it is impossible to test all the possible combinations. However, when using content generation, comprehensive testing should be performed. Many CRPGs are also deployed to several platforms, e.g., PC and console version are developed simultaneously. Naturally this leads to more extensive the testing, since all product versions must be tested.

2.7 Game Components

As stated earlier, important elements of CRPG are the storyline, the virtual world inhabited by characters, and the game mechanics that act as the laws of nature in the game. In addition to these, CRPGs contain also other components.

Game world is an imaginary place where the events of the game occur [2]. In CRPGs the game world is usually shown to the player as a 3D virtual world, although some CRPGs do use 2D perspective. In a good CRPG the player can feel being a part of a living world and can also imagine that there are more areas and events than are present in the game.

NPCs of the game world are controlled by AI. CRPGs are fairly AI intensive, since they contain lots of NPCs. In general, CRPGs also require advanced behaviour from game characters than games of other genres. Mistakes in game AI have more time to irritate player since they reapperar more times, because CRPGs are long games with a game time over forty hours [13].

Game mechanics are the artificial laws of nature in the game world. Commonly combat mechanics are important in CRPGs, since the player spends a lot of time in combat or preparing for those. Game mechanics should be tested carefully, since bugs there can ruin the player experience.

UI in a CRPG is the part used for controlling the player character. It also displays information, for example health and weapon in use, to the player. Many CRPGs allow the player to configure UI of the game, e.g., by defining which commands are placed to shortcut icons. Some games take this even further: World of Warcraft [95] lets the player program the UI of the game using Lua scripts [37].

The player character is the main character of the game and one point of the game is to get the player to identify with him. In the beginning of a CRPG, the player character is usually weak and impecunious. The player has to advance in the game to build the character up and finally complete the game. Management of items, spells, and other configuratable assets of the player character is an important part of the game. Character management can be a very sophisticated task, requiring long-range planning. For example,

character classes can have different kind of skill trees (or feat trees) and advancing in some skill can prevent advancing in another one. Usually this advancement in the skill tree is performed by gaining experience, and skill levels are upgraded when enough experience is gathered to reach a new level. When character advances into a new level the player may be able to select new skills or features. Skill trees in CRPGs are very similar structures to technology trees used in strategy games [29]. In some CRPGs the player controls a whole party of characters instead of only a single main character, which increases the number of management tasks and also possible strategies in combats.

2.8 Special Characteristics of Computer Role-Playing Games

When sizes of CRPGs have grown to their current forms, the authorial burden to create game content has also reached unequalled limits. Special reasons why CRPGs generally need more content development time than games from other genres include huge game worlds with dozens of NPCs and long, branching game storylines. Personalization of the game content is also typical only for CRPGs, since the desire for a personalized gaming experience originates from PnP RPGs.

2.8.1 Content Creation Problem

Content is most of what is contained in a game: levels, maps, game rules, textures, stories, items, quests, music, weapons, vehicles, characters etc. However the game engine itself is not considered to be content. Further, non-player character behaviour – NPC AI – is not considered to be content either [76].

Personalization of CRPG increases the application developers' work for creating CRPGs. This is understandable as even a simple branching of the storyline multiplies the needed development input by the number of branches. Modern CRPGs have often multiple endings and sometimes also separate tu-

torials for different character classes. Instead of complete branching or creating different quests, intermediate forms can also be used, where some parts of the quests are modified based on the player character. These intermediate forms require less development work than branches, because part of content can be reused.

Content generation is a method for reducing the application developer work when developing CRPGs. Term *procedural generation* is used to refer content generation taking place algorithmically. Content generation tool or algorithm can use an initialisation file as the basis of the generation. If attributes or other data of the player character are used in the generation process, these tools can also produce personalized content.

Generally content of the game can be generated either at development time or at run-time. Both of these methods have their advantages. Content generated at development time can be evaluated by game developers and the process can be run again until the outcome is the desired one. When using run-time generation the content is generated only when it is needed in the game (it does not need to be shipped with the game), and in theory new content can be created endlessly.

2.8.2 Personalization Problem

A main reason why people play any kind of role-playing games is to have a personalized character, which the player can identify with. The creation of this character can vary from rolling the dice and writing the background story for character (PnP RPGs) to sewing character's cloths (LARPs). In CRPGs the player has usually some control in creating his or her character, but not as much as in other formats.

The fun of having a personalized character is lost, however, if it has no effect in the game. Personalizing the player experience is a key factor to make the player feel involved in the virtual world of the game [84]. Unfortunately, the personalized player experience of PnP RPGs is one of the most difficult properties for CRPGs to emulate. Computers need explicit instructions to create similar personalized playing experience that human game masters in

PnP RPGs create. In PnP RPGs a human game master is familiar with the player character, and can create personalized adventures on the fly. To a computer, the player character is a bunch of numbers, and this data collection is what game programmers must use to when giving explicit instructions for creating personalized adventures to a computer.

The lack of personalization is also discussed in [69], where Sullivan et al. point out that CRPGs have general lack of density of interesting and meaningful choices for the player character within the story. They also point out that complex character creation systems of CRPGs have no effect on the story beyond a few word replacements.

If a storyline of CRPG adapts to the player actions, the player is given these meaningful choices. This adaptation is done by opening quests and thus new storyline branches based on the 1) explicit choices the player character makes during the storyline and 2) implicit choices that the player character has made during the storyline or in the character creation. An example of an implicit choice could be choosing a special ability or skill when gaining a new level.

Individual quests are basic building blocks of the storyline, so personalization of quests is important. Special quests for different character classes can take advantage of special skills of the class. In all, personalized quests shift the game towards the same player experience as PnP RPGs offer, which is one purpose of CRPGs.

Publication [II] presents guidelines for personalizing the player experience in CRPGs. It is divided into three parts covering personalizing: 1) the game storyline, 2) game characters, and 3) game difficulty. We present that use of these guidelines can improve the personalized player experience offered in CRPGs. However, as presented in the paper, when the guidelines are used and more personalization is gained, also the authorial burden to create these personalized stories is increased. Correspondingly, tool support for content generation can be used to used reduce this authorial work.

Chapter 3

Software Tools in Game Development

Human history is a history of tool usage. From the days of first agriculturalist people, tools have played a significant part in human civilization. It is generally accepted that the versatile use of tools is specific to humans, although many animal species are able to use primitive tools [41]. Modern tools are not always concrete physical instruments and different kinds of software tools are an important part of software development. In software development generally, term *software tool* is used to refer to a piece of software used to assist in the development of other software being build [39].

Software tools play a very important role in game programming. Modern computer game development projects are enormous software projects, typically including 20–50 professionals and lasting two years to be a complete product [67]. In addition to common software development tools such as compilers, debuggers, and integrated development environments (IDEs), there are also numerous specific tools used in game programming. For example, game engines, code generators, modelling tools (referred also as digital content creation (DCC) applications [25]), and physics engines are available. On top of ready-made game programming tools, new special tools, e.g., specific level editors or AI tools are implemented in many game projects. These

tools can later be reused to implement similar games. Figure 3.1 shows software tools commonly used in CRPG project.

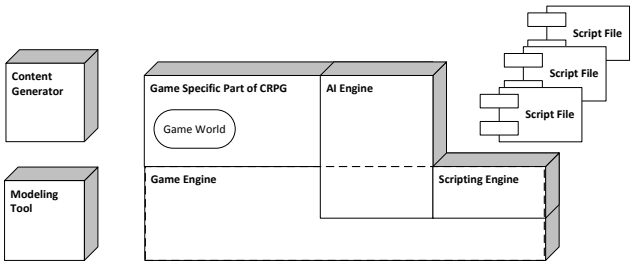

Figure 3.1: Common software tools in CRPG

3.1 Game Engines

Game engines [25], software product-lines [12], [16] to build computer games, are the most important tools used to build computer games. A term *game engine* refers to a software system designed for creation and development of computer game. They enable reuse of software code and sophisticated software components, therefore development of a new game is speeded up rabidly. Especially when developing CRPGs, new games are nearly always build on top of a game engine. Even if the game engine is developed for a certain game, it is most likely used to build several similar games later on.

Features present in game engines vary a lot. Major components of a modern game engine usually include the following:

- Rendering engine. Rendering engine is responsible for producing three-dimensional graphics. It transform data of location and appearance of game world objects into 3D virtual world.

- Physics engine and collision detection. Physics engine is responsible for behaviour of game world objects. It ensures objects cannot overlap or acts otherwise unnaturally. Ready-made collision detection also very useful feature, since game objects can have very complicated shapes.

- AI engine. AI engine controls NPCs in the game. The required functionality for game AI varies depending the game genre [13]. Typically in CRPG at least path finding, and combat behaviour are essential.
- Networking. Networking component is responsible for handling required network connections.
- Sound. Sound component plays sound effects of the game. It is also responsible for playing background music.
- Scripting engine. Scripting engine runs scripts used in the game. Depending on the game, different parts of the game can be implemented using scripts, commonly they are used, for example AI and conversation dialogues.
- Support for IO, threading, and memory management. A game engine requires low-level functionality, such as support for IO, threading, and memory management. CRPGs are massive software with a lot of code lines, and it is important they do not leak memory, and file operations run smoothly.

Some game engines offer IDE, where software tools are included, and IDE works as the user interface for the game developer using the game engine, while others are used through common IDEs, for example, Microsoft's Visual Studio [49].

In addition to the features present, the programming paradigms in game engines vary. Depending on the game engine, most of the game can be implemented by programming through IDE. Models and other parts are created using external tools and loaded into the game in the source code. The other possibility is that the game engine uses integrated graphics environment as the primary method of development. In this approach the game development is mostly done by modifying properties of graphical objects present in the environment. In addition, scripts are written if more complex behaviour is needed for some object. Examples of such game engines are Unity [85], Blender Game Engine [10], a component of Blender 3D content creation suite, and Torque 3D game engine [77].

Before game engines, the game code was written as a single entity, consisting of game logic and game data. The term *game engine* was introduced in mid-90's, with first person shooters (FPSs) like Id Software's Doom [35] and Quake [36]. Quake was also one of the first games open for user modifications (mods). When used with a game upgrade called QuakeWorld, it was possible to program the game using compilable QuakeC programming language. QuakeC enables the makers of modifications to publish their creations in a single bytecode file, therefore without revealing the source code [96].

Nowadays, scripting languages are often used to make modifications for games, and those are not compiled. Scripting interface is sometimes left open on purpose to allow community made modifications for the game, although there are several other ways for making modifications. For example, initialization files can also be modified to change graphics or other features in the game. Existence of mods can extend the lifespan of the game significantly. Famous examples of games with wide moding communities include Civilization IV [15] and The Elder Scrolls IV: Oblivion [72].

Especially AI support of game engines has been improved in recent years. Nowadays, there is an increasing tendency to have general AI routines in the game engine, which allow characters to be designed by the level editors or technical artists [51]. A term *AI engine* is used to refer to the part consisting the AI routines.

Publication [I] of this thesis presents a novel game engine implemented to be used in game programming education. The implemented game engine CAGE includes a C++ core and a scripting interface. The C++ part can be called through the interface using Lua scripts. In addition, CAGE includes an AI framework, which allows easily implementing NPCs using state machines. Students used CAGE as a platform in programming assignments of game programming course.

Recently some game engines have appeared that can be used for implementing browser-based games. These tools require different kind of features than game engines functioning in traditional environment. Today most game engines are implemented for binary environment. However, this might well change in the future. More and more browser-based games are developed

and many games that previously would have been implemented as installable software are nowadays deployed onto the Web.

3.2 Scripts

Different types of scripts are important part of modern CRPGs. The game engine may offer its own scripting language, use a common one, or game developers can implement their own scripting language. Currently Lua [42] is a popular scripting language in game development. It is used also in many famous CRPGs, for example Baldur's Gate and The Witcher [74].

Scripting languages can assist the development process in many ways [14]:

1. They can be used as a quick and easy way of reading variables and game data from initialization files.

2. They can save time and increase productivity, since modifications to the code can be done without recompiling.

3. They can increase creativity, since non-programmers may be able to participate in the development.

4. They provide extensibility, as the scripting interface can be left open for player's mods.

Script code is not compiled before running it. Therefore, in the development phase it is easy and fast to test features implemented using a scripting language. Scripting language can be used for almost everything in the game, although graphic-intensive parts are seldom implemented with scripts. For example, AI features, such as path finding, can be written using scripts. Later, if more efficiency is required, scripts can be converted to compiled programming language. A well-known example are user interface scripts used in World of Warcraft [95]. The World of Warcraft game client consists of two major parts, the game world and the user interface. The game world part cannot be modified through scripting, whereas the user interface part can be affected. The game lets the player to write scripts that modify it. The

user interface part includes action buttons, unit frames, maps, and options windows. Add-ons can be written to add or modify existing elements to add functionality or to show information in a different way [37]. These configurations can allow the player to perform actions faster, there possibly gaining advantage in the game.

The syntax of a scripting language is usually much easier to learn than that of a compiled programming language. A scripting language can help non-programmers like game designers and artists to participate in game development, if they are able to write some of needed scripts themselves. There are also tools and pattern catalogues for assisting non-programmers to participate using scripting language. For example, ScriptEase [47] can be used. It allows designers quickly build complex NPC behaviours without performing explicit programming.

A scripting language separate from a programming language is usually used only in installable computer games. However, in browser-based games the whole game can be implemented using a scripting language, such as server-side JavaScript [17]. In this approach the game functionality is divided into the server and clients. For example, the prototype browser-based game implemented in Publication [VII] was implemented using JavaScript on both client and server side. A term *scripting engine* [61] is sometimes used for the part running the scripts in CRPG.

3.3 Related Work: Game Engines

Designing a new game engine is a demanding task. Therefore we inspected a number of other game engines before deciding to implement a new one.

Unity [85] is a game development tool by Unity Technologies. Unity is programmed by scripts, written in either JavaScript, C#, or a dialect of Python named Boo. All three can use .NET libraries. Unity supports multiple platforms, Windows, Mac, iOS, Android, Nintendo Wii, Play Station 3, and Xbox 360. However licences for Play Station 3, Xbox 360 and Wii must be negotiated separately.

Unity web player plugin allows running Unity games on a browser, so it is

a possible tool for browser-based games, too. For example, a popular browser-based game Battlestar Galactica online [8] is implemented using Unity web player.

Some student groups used Unity in our game programming course with very good results, producing high-quality games. However most of these students had prior experience with the engine. The documentation of Unity is massive and the learning curve to use it is pretty steep. We consider that Unity is not a good selection to be a game engine for students trying learn the basics of game programming because using Unity efficiently requires a lot of time spent studying it.

XNA [98] from Microsoft is a set of tools with a runtime environment. XNA is a framework that can be used to build games for Windows [50], Windows Phone [93] and Xbox360 [97], and the framework encapsulates the differences between platforms. This encapsulation allows game developers to concentrate on game programming and content creation instead of configuration problems. It supports games with 2D and 3D graphics. Especially current Windows Phones seem to be an excellent platform for games build using XNA. Since it enables rabid development process and easy deployment of games. Figure 3.2 shows an XNA application running on a Lumia 800 Windows phone.

At the time we designed the game programming course, XNA was not as mature as today and current Windows Phone did not exist. Also, the lack of scripting engine and the fact that there was no possibility to easily add a scripting language to the game project made it less suitable for the course.

Panda3D [57] is an open source game engine implemented in C++. However, the game development language of the engine is Python. Typically, a developer using Panda3D writes a Python program, but he can also directly access the engine with C++ code. Features of Panda3D include graphics, audio, collision detection, built-in physics engine, and particle system.

Since the approach of Panda3D, a C++ core with a scripting language, was close to the idea of our game engine, Panda3D was a worthy candidate for the game programming course. However, Lua is a much more popular game programming language than Python. In addition, the open source

Figure 3.2: Lumia 800 Windows Phone running XNA application.

community of Panda3D was pretty small, and at the time the engine was not very mature.

Since none of the existing game engines seemed suitable, we decided to implement a new one. However we used ready-made graphic engine Crystal Space 3D [18]. Using a lower level graphic application programming interface (API) instead of a ready-made graphic engine would have increased the workload. Crystal Space 3D also offered the important tool chain for importing 3D models into the game world.

3.4 Software Engineering and CRPGs

The design of computer games differ from traditional applications in many ways. When designing games, the end result often resembles work of art rather than engineering - it is even treated as such by the public [55]. However, the software tools and methods used in the development are similar to traditional applications. It is stated that software engineering for computer games is a field that embraces many techniques and methods from conventional software engineering and adapts them to so as to fit the specific requirements of game development [3].

Prototyping can be beneficial in CRPG development projects. It allows gaining player feedback from the game, in early phase of the development. For example, functionality or playability of the game can be tested with prototype.

Games are typically tested with experts, and players who belong to the target group of the game [56]. These experts are especially useful in the beginning of the game development process, as they can spot obvious problems in the game and can also benefit their experience when testing a game, which is still incomplete. These experts can be, for example, game developers who have not been involved in the development of the game. Players of the target group, who are not previously exposed to the game, are very useful when doing playability testing [56]. They provide realistic feedback, because they are in a similar situation as a new player who has just bought the game, and does not know much in advance.

There are other uncommon methods that can be used in game development. For example, there are research that presents formal specifications can be useful in game development, especially in the design phase [54].

3.5 Game Design and Tool Support

Adams [2] defines game design as a process of

1. imaging a game,
2. defining the way it works,
3. describing the elements that make up the game (conceptual, functional, artistic, and others),
4. transmitting information about the game to the team who will build it.

This kind of a flow treats game design as a creative process, totally separate from the tools used to build the game, at least until to the last phase where the team will build the game. However, even if a game designer is

no programmer or has no experience on the actual tools used, it is very important he or she understands what can be done with current tools and technology, and how difficult this is. Otherwise it is probable that game designer has unrealistic expectations.

In the second phase of Adams' game design process, game mechanics are created. The game mechanics is one of the most important factors of game success, as the players quickly quit playing good-looking games with poor mechanics. Countless hours may be needed when balancing the game mechanics, and it is also common to release updates or patches, which correct bugs in the game mechanics.

In addition, allowed breaking the game mechanics can offer interesting opportunities in games. For example, in Magic: The Gathering [78] and its digital conversions, each new card set offers the player cards with mechanics that somehow break the existing game rules. The players must continuously think using new rule patterns. As an example of the difficulty of game mechanics balancing, there has been several situations, where some new cards from a released set have been too powerful when combined with some other card, and those have been banned from game tournament formats [6]. Some of these problems originate from the fact that it is impossible to test all card combinations of thousands of different Magic: The Gathering cards. While currently cards are tested more carefully, ten years ago it was not rare that some new card broke the rules entirely and some errata were added. Elaborating rules are also used nowadays, but overall rules of the game rarely need to be modified.

3.6 Content Generation Tools

Special reasons why CRPGs require so much application development time for creating their content include huge game worlds with dozens of NPCs and long, branching game storylines. In addition, adding personalization to the game increases the need even further. Content generation tools can be used to create parts of game content, there reducing the authorial burden.

Content generation has long history in CRPGs. The dungeon crawling

game Rogue was released in 1980, and it used procedural content generation to create levels. Each level contains grid of three rooms by three rooms or dead ends where rooms should be. However, later levels contain mazes in place of rooms also. Already in 1980s several rogue-like games, for example Hack and its successors, like Nethack, generated game levels (dungeons) randomly, with far more complex structures than those of original Rogue. Level generation is perhaps the most common case of content generation in CRPGs. There are various methods for procedural content generation and these have strengths and drawbacks [75]. Methods can also be combined for better outcome. A good example of modern game using level generation is the action CRPG Diablo III released in May 2012 [19]. The game uses a random level generator, and also a random encounter generator for providing different player experience for each playtime.

In some games, variety gained from random content can be one of the key features. For example, Slaves to Armok: God of Blood Chapter II: Dwarf Fortress [22] is in part a rogue-like CRPGs and in part a city-building game, where the game world is generated randomly from scratch for each game. Although the game is known from its extremely steep learning curve and use of only ASCII graphics, variety of the game, rich content, and sophisticated game play makes for an excellent completeness. Figure 3.3 shows Dwarf Fortress II game in action.

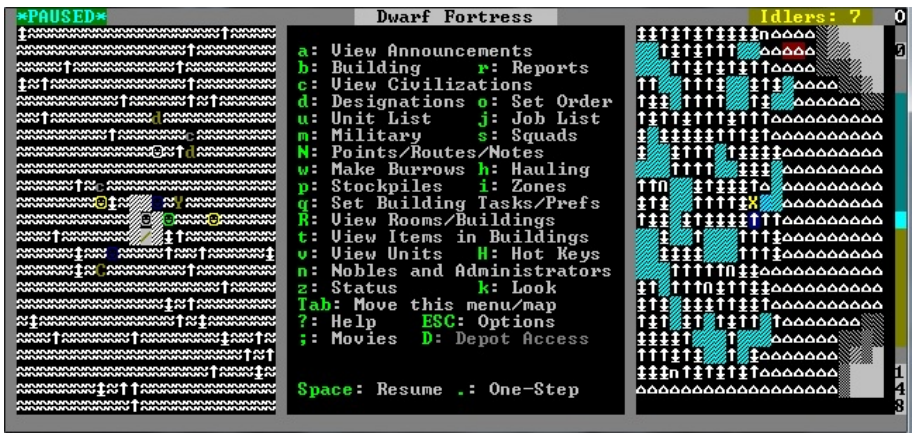

Figure 3.3: Dwarf Fortress game in action

Quests are another common target for content generation tools. Implementing the sophisticated storyline of a CRPG takes hours of application development work and by generating part of quests can ease this workload. There is research [21], which presents that CRPG quests share a common structure. Quest generators can mimic this common structure and produce similar quests that human designers create. The use of quests generators also increases replayability of the game, since the game storyline is different for each playtime. Another possibility is to use generated quests in addition to pre-coded ones, further increasing the total number of quests in the game. Quest generators are commonly used to create side-quests, since without further manual operations the generated content does not relate to the game storyline [90].

Combat systems of CRPGs quickly became very robust. Tables and rolling dice was easy to adapt into electric form, and combat-based quests are also an easy target for quest generation. Even the slightest storytelling elements in the quest increase the difficulty of generating it by tools.

Quest generation tools can also assist in personalizing the player experience of a CRPG. Without these personalization of the storyline must be done manually by branching the storyline for each character class (or what ever fact is used as the personalization factor). When using generators, the attributes of the player character, are used as initial values for the generation process.

If branching the critical path of the game storyline is considered to be too laborious or difficult to implement even with quest generation tools, another option is to use the same critical path, but offer personalized side-quests for player characters. Publication [IV] presents a method to generate personalized side-quests for CRPGs and a prototype implementation to evaluate the presented method. The personalization is achieved using player character attributes as a basis for generation, therefore aiming at creating quests suitable for the player character. In this approach the generation process is performed at run-time.

Important elements in our approach are quest prototypes and a configuration set. The generator uses quest prototypes. Each quest prototype

is meant to be instantiated several times. These quest prototypes consists modification points, which are changing elements inside the quest prototype. A configuration set defines game world rules, for example, what character classes there are and what skills characters can have. Application developer defines quest prototypes and its modification points by writing Lua scripts. The prototype definition includes information on what skills of the player character increase probability of encountering the quests, and what skills decrease this probability. When the player character talks to quest giver NPC, at first the probability for each quest prototype to be instantiated is calculated. Then one of those is picked up randomly based on their probabilities, and it is instantiated. Finally modification points in the quest prototypes are filled using player character data. In addition, difficulty adjustments can also be performed using modification points by including related factor, for example number of enemies into the modification points and giving fighting skill of the player character as other factor.

In addition to game levels and quest generation, there are other possible targets for content generation tools. For example NPCs, items, graphics and sounds can be addressed. It is possible to either generate new instances of existing NPC types or create completely new kinds of NPCs types. In essence items are similar to NPCs as a generation target. Graphics and sounds are different issues, since those relate to how the game world is presented to the player. Currently those are usually not generated, but it is possible that those are more important generation targets in the future since game world sizes continue to grow.

Publication [III] presents a content generation method, where game levels are created from XML files using a generator program. The game developer defines these XML files at development time. Defined XML files contain of object templates for levels and items. Templates include limits for attributes of objects to be generated. In addition, there can be function names inside template, game engine framework calls pre-defined functions in when a particular event happens. By using these function it is possible to create functionality for generated objects, for example define a function that adds ammunition of the player character when he picks up box of ammo. When

the player character advances into a new area in the game, content of the area is generated based on this pre-defined XML file. This generation is performed at run-time.

3.7 Related Work: Other Software Tools Specific for Computer Role-Playing Games

Although a game master is not commonly present in CRPGs, there have been some games with game master tool kits available. In CRPGs a human game master acts similarly to that in LARPs; e.g., he or she can only be present in a certain location of the game world, and the game world rules take care of some duties traditionally tasked to the game master: the game master is not needed to resolve all combats. For example, an important feature of Neverwinter Nights was the DM Client, a tool that allows an individual to take the role of the Dungeon Master, who guides the players through the story and has complete control of the server [83]. Similarly to PnP RPGs, in Vampire: The Masquerade - Redemption one of the players acts as game master. The game mastering system allows placing of NPCs and items.

However, game master tools do not allow dynamic creation of content, and the game master is limited to pre-created content [83]. Dynamic content creation is one of the main responsibilities of game master in PnP RPGs, because it allows game to advance into directions not covered in the scenario.

Chapter 4

Run-Time Environment

Currently CRPGs can be implemented as native (binary) applications or as browser-based applications with no explicit installation needed by the user. However, at the moment full-scale single-player CRPGs are almost always implemented as binary software that are installable to a local computer. In contrast, multiplayer CRPGs can be implemented as binary software or as browser-based applications, although browser-based implementation is more common for strategy multiplayer games.

The browser-based implementation eliminates need for installation, which lowers the starting threshold for playing the game. Games can be compared to other digital entertainment, and e.g., when starting to watch a film it would be ridiculous to wait fifteen minutes to installation. Therefore starting a game should be as easy as starting a watch film, and browser-based implementation is a huge step towards this goal.

Previously lack of sophisticated technologies have hindered implementing browser-based games. The browser was originally designed to be a document viewing and distribution environment, not an application platform. In addition its programming capabilities were not carefully designed feature, but an afterthought [5]. However, in the past decade new technologies and standards have greatly increased programming capabilities of the Web, and nowadays sophisticated web-based applications are common.

Evolution of the Web has headed to a point where the whole software industry is experiencing paradigm shift towards web-based software [71]. The

majority of new software systems and applications are developed for the Web, instead of personal computers. These new software applications do not require installation or manual upgrades, and their distribution is superior to their binary cousins. Most likely this shift to web-based software will include computer games also. Therefore, we expect that browser-based games are the future of most computer games, too.

Early browser-based games were text-based lacking all graphics. Early instances of browser-based games included Earth 2025 opened in 1996 (discontinued at December 2009) and Hattrick opened in 1997 (still ongoing) [28]. When technologies have evolved, browser-based games have gained better graphics and are currently closer to their installable counterparts. However they are still missing graphic-intensive features, like cut scenes. Currently these video episodes also require lots of disk space and streaming those from the Web would require too much resources.

Although almost all single player CRPGs are implemented using installable software, there are signs of a trend towards centralized architecture, where some critical part of the game runs on the server, although in practise the server is usually a group of server computers. For example in Diablo III [19] quest generation is performed on the central server, and the game needs network connection even in a single player mode. In addition, the player characters are stored on the server to prevent players from modifying their characters.

4.1 Multiplayer Internet Games

Multiplayer games played on the Internet can be divided into types based on their attributes. One possible division is presented in [64]. The division attributes are: installable (or downloadable) versus browser-based, and short-term versus long-term. This division into four categories is shown in Figure 4.1.

In this categorization multiplayer Internet games are divided into four categories. On technical side plugin technology could be seen as the intermediate form between installable and browser-based. In this approach the

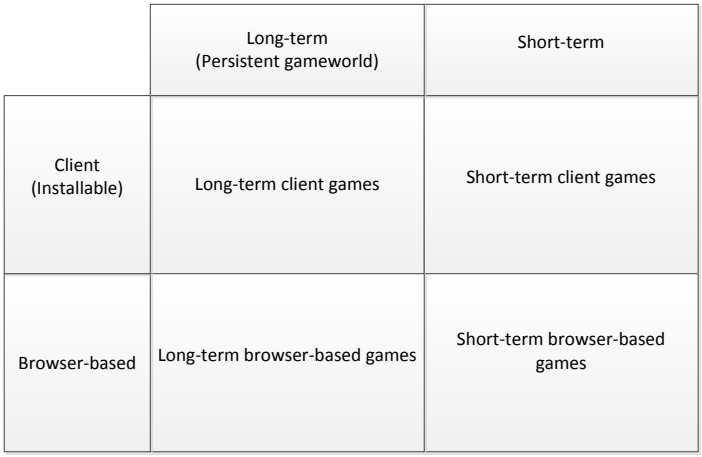

Figure 4.1: Categorization of multiplayer Internet games

plugin used to play the game is installed when it is required, and later it can be used to play other games using the same plugin. However in this categorization plugins are placed into browser-based side.

Long-term client games contain a persistent world and client software installed to a local computer. Typically, massively multiplayer online role-playing games (MMORPGs) belong to this category; for example, Blizzard Entertainment's World of Warcraft mentioned earlier. With over 11.4 million subscribers as of March 2011, World of Warcraft is currently the most-subscribed MMORPG in the World [94]. At the peak of its popularity the game had 12 million subscribers. For comparison, Travian [80] a popular long-term browser-based strategy game, has over 5 million players. The numbers show that installable software using long-term games are ahead, although accessibility of browser-based games is an important competitive advantage, and gathering new players is easier to those, since monthly payments are commonly used in popular MMORPGs with a client software. During recent years the quality of browser-based games has improved, and they are now closer to regular computer games. Currently, browser-based games can be implemented with smaller developer teams, so the development is not as expensive.

Short-term client games do not contain a persistent world, and therefore the game time of a single game is short. There is no game world present, only precoded simple levels. For this category, common example games are multiplayer FPS games played with client software. In these the players usually form *ad hoc* teams and points are gained by killing other players.

Browser-based games do not use specific client software, so software installations are not needed and the games are accessible using a web browser. This accessibility has appealed to many players who have never bought a computer game, and have no intentions to do so. There are also other benefits. For example, updates are distributed instantly.

Short-term browser-based games form a group of multiplayer casual games, played in the browser. The group is very similar to short-term client games, but usually short-term browser-based games are equipped with lower quality graphic and simpler game rules. Also in this group, the accessibility of browser-based implementation can lure more players, since the player can experience the installation process to be too cumbersome for a game that he or she will only play for a couple of times.

Long-term browser-based games contain a persistent world, where players compete against each other. A part of the charm these games' is that each opponent is a real person instead of being computer controlled. In these games, the game time is long or unbounded. For example, in strategic browser-based games, a year can be a typical playtime, after which the winning team or players are declared and the server booted for a new game. The existence of a persistent game world and the fact that the same players are playing the game for a long time is beneficial to creating player communities. Players also form alliances or teams, which compete against other groups. In many long-term browser-based games the winner is the winning team instead of a single player. Websites, forums, and IRC-channels form around these player groups. Online communities built around browser games are discussed in Publication [V].

Long-term browser-based games are often referred to as persistent browser-based games (PBBGs) [60]. By definition, PBBGs do not need to be multiplayer games. However, single-player PBBGs are not very interesting, since

they lack gaming communities and strategic and tactical dimensions, which make multi-player PBBGs a popular and interesting genre. Term *browser game* is used in Publications [V] and [VI]. The defined genre is similar to PBBGs, however our definition is stricter, since it requires multi-player and a playing account, which the PBBG definition overlooks.

Long-term browser-based games have unique game flow. Instead of continuous gaming, the games are played a moment at a time. Usually a player plays couple of moves and after several hours of real life, he returns to continue with more moves. The games do not require 100% attention and a research [63] points out these games are usually played also during work time, lecture time or in school. Some long-term browser-based games offer rewards if the player is more online, while others try to restrict the benefits from being constantly online and emphasize the skill of the player. A good example of rewarding the player from being constantly online is the browser-based strategy game Travian [80], which allows players attack each other at any time, even if other players are actually sleeping. However, resource regrowth is constant and resources of the player increase at the same time if the player is logged on or not.

Hattrick [28] is an example of another type of game flow, where the game does not reward players for being logged on all the time. Hattrick is a leading football management game on the Web. In Hattrick the player builds his or her own football club, manages the players, and competes head to head with other human managers. League games are played only once a week in real time, and playing the game efficiently requires only small amount of time per week. Finding a key strategy and setting correct long-term goals is the key to victory in Hattrick. Of course, there are huge forums where to learn strategies and a player community for those wanting to spend more time with the game.

In addition to technical aspects, the success of PBBG depends also on psychological reasons. Online communities are very strong motivations to keep playing a certain game [64], even if the player has already lost most of his or her interest in the game. This is understandable since the players can

create new friendships with community members and if they do not know these people anywhere else, these contacts are lost if he or she quits playing.

Many browser-based games are very addictive and require lot of time to play effectively. The time requirements for playing effectively are one of the main reasons why players quit playing long-term browser-based games [4]. This time increases as the game advances, and in some point the game begins to feel more like a duty than a pleasure or recreational activity. Social relationships in real life can also affect playing of a certain game. Sometimes the partner of a player might become annoyed from constant playing, and eventually the player quits the game although he or she has not lost interest in it.

4.2 Development Technologies of Browser-Based Games

Traditionally PBBGs have been implemented using server-side scripting with e.g., PHP [24]. However more recently web plugin players have appeared [VI]. These allow the use of sophisticated 3D graphics, and have brought PBBGs closer regular computer games in quality.

A typical characteristic of all web-application development is that there has been plenty of technologies and usually applications are implemented using many of them. While web-applications have became more complex and closer to their desktop counterparts, the technologies have not evolved at the same pace. This has lead to complex set-ups of different technologies mashed together. The same evolution is true also for browser-based games.

Typical user interface of a long-term browser-based game implemented in server-side scripting, Hattrick [28], also mentioned above, is shown in Figure 4.2. Typically such manager type long-term games do not use fancy graphics, a text-based user interface is all that is needed. The user interface has remained mostly the same since a major update year 2000. Server-side scripting works well in this kind of game that does not require graphic-intensive parts. However, the game requires large database support, with

hundreds of thousands of stored objects, but this is no problem for server-side scripting as it is easy to communicate with the database.

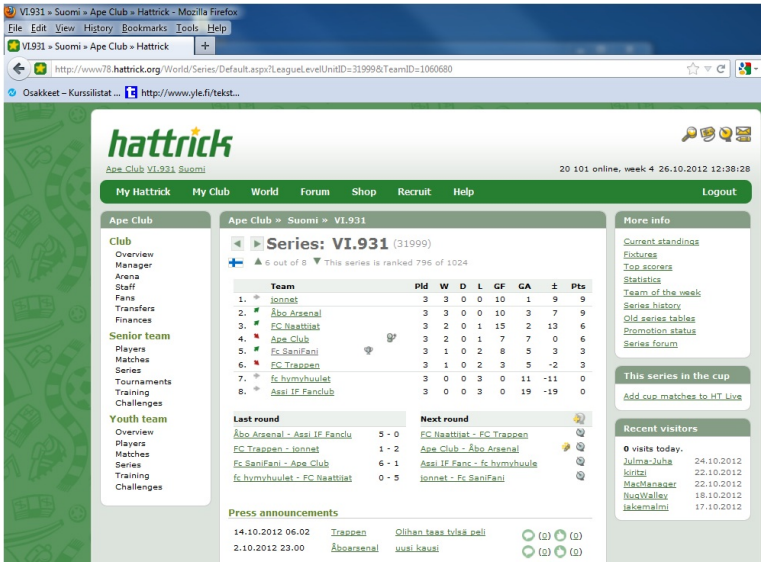

Figure 4.2: Hattrick user interface

Web plugin players are installed into the web browser. They can be used to implement sophisticated graphics. For example, Battlestar Galactica online [8] is implemented using Unity web player [86]. The game is a good example of using plugin technology, and it has graphics up-to-par with a traditional installable game. A drawback of plugin players is that those must be installed separately, but the same plugin player can of course run all games implemented using that technology. Also, almost all browsers are able to offer the installation of the plugin to the user when a game needs one.

Today, a trend from browser plugins to the use of web technologies can be identified. The new HTML5 [34] standard brings new useful features that can be used for implementing browser-based games. HTML5 API is accessed using JavaScript. From all of its features, for example, new canvas element for drawing, drag-and-drop, offline web applications, and document editing possibilities can be useful for developing browser-based games. There is already evidence of its capabilities. For example, a recent project [66] re-created the famous real-time strategy game Command & Conquer entirely in

HTML5. It is probable that HTML5 will affect the development of browser-based games, because its new features can boost browser-based games to operate faster and enable new kind of gaming experience [99]. Moreover, the WebGL [92] standard library brings power of low-level 3D graphics into the web browser without the use of browser plugins. It is a JavaScript API for rendering 3D and 2D graphics.

Previously, performance problems have hindered the use of JavaScript in browser-based games, but these problems are diminishing with the development of sophisticated JavaScript virtual machines. Moreover, as already mentioned HTML5 and WebGL also strengthen the position of JavaScript as an implementation language for browser-based games. These technologies for implementing browser-based games are discussed in Publication [VI].

4.3 Game Engines for Browser-Based Games

Game engines are the most important tool in modern game development, although those have not yet established a position in the development of browser-based games. Currently there are only few game engines that can be used to implement browser-based games.

There are also installable multiplayer game engines. A single player game engine requires specific middleware to operate as multiplayer game engine. Some of existing middleware solutions include a game engine, while others concentrate on networking and must be combined with a game engine to create a multiplayer game. For example, BigWorld [9] is a famous firm providing middleware for multiplayer games. Although these installable multiplayer game engines are a decent solution, they require installation.

A game engine for multiplayer games requires different kinds of features than a traditional game engine designed for single player games. For example, registration of players is an essential feature for a multiplayer game engine. In addition browser-based environment requires some changes, for example a scripting interface is no longer appropriate, since a game engine is most likely already built using a scripting language and it makes no sense to create an interface for using another. It is also impossible to leave the interface open for

players to run their own scripts, although an interface that allows modifying the user interface of the game, similar to that present in a World of Warcraft client would be a possible feature.

The following are important features of a game engine for browser-based games. We consider these to be mandatory in a such game engine. In addition, there are optional features, which provide additional value for a game engine. These features are discussed in detail in Publication [VI]:

- Player registration support. Player registration is the only required action to start playing a browser-based game, so it is natural that a game engine would automate this process.
- Networking component. This is responsible for handling player connections, and ensures that game data relates to the correct players.
- Security features. Security is a cross-cutting aspect and security features should be implemented in the game itself and also at the game engine level.
- Database access. Browser-based games contain thousands of game objects that are stored into a database for persistent storage. The game engine can offer easy access to the database for the game, it can hide the required SQL-statements from the game code, and objects to be stored can be delivered, for example, as function call parameters to the game engine.
- Player messaging system. This lets players send messages to each other. It functions as an in-game email system, and it is needed in most of browser-based games.
- Graphic interface. Graphic interface offers functions for easily presenting graphics.

As mentioned earlier, there are many technologies for building browser-based games, and all of these are potential options to be used in the development when creating a game engine for browser-based games. When we

implemented our prototype CRPG project for [VII], we gained support for a presumption that JavaScript is a suitable programming language for this task. Currently there are few frameworks specific to games that can be used for building browser-based games in JavaScript, and they do not offer nearly enough features to be called game engines. However, there are some libraries that can definitely still boost the development process.

Node.js is a platform for easily building fast, scalable network applications [52]. When using Node.js the whole application is implemented using JavaScript. NowJS [53] a library built on top of Node.js, presents a new namespace called *now*. It acts on the client and the server, and for application developer it makes those parts act like they were a single program, allowing sharing synchronized variables between the client and the server and calling functions in server from client and vice versa. These two frameworks were used in implementing our prototype CRPG project presented in Publication [VII].

Another platform for JavaScript network applications is Meteor.js [48]. It enables data synchronization between the client and the server, and database API is also usable in the client. Applications made with Meteor have local MongoDB database. There are lot of other features also, such as login support for Facebook and Twitter accounts, easy email sending, and instant updating of code.

Chapter 5

Introduction to Included Publications

This chapter presents the publications included in this thesis, and discusses game programming courses where the CAGE game engine was used.

5.1 Publication I: Scriptable Artificial Game Engine for Game Programming Courses

The first publication of this thesis [I] discusses development of the CAGE game engine, and its use in the first game programming course. After the first course implementation we continued the development of CAGE. The idea behind CAGE was to make the beginning phase of a game project as easy as possible. The students can start their game projects by writing small Lua scripts and see how these affect the virtual world of CAGE.

5.1.1 The CAGE Game Engine

The use of Lua scripting language make unnecessary the compiling of the CAGE framework after any modification of Lua scripts. Therefore, effects of modifications are rapid to test. Lua scripts can call C++ functions, which are declared in the Lua interface of CAGE, and Lua scripts can also be called

from the C++ code. Currently, there are over 260 C++ functions that are revealed in the Lua interface. The largest group of functions is meant to manipulate NPCs and objects in the virtual world of CAGE.

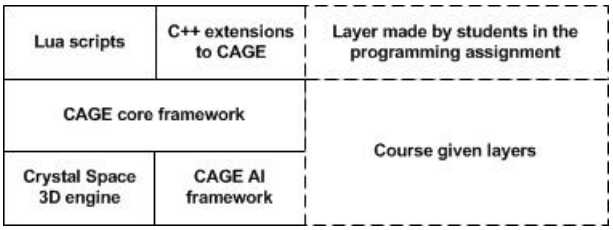

Figure 5.1: Layer architecture of CAGE

CAGE graphics are powered by Crystal Space 3D, an open source 3D engine [18]. It was selected to ensure importing of 3D models and worlds made by the users. 3D models and worlds are created using modelling tool, for example Blender [10]. Figure 5.2 shows the tool chain of creating and importing a 3D model to CAGE. Lua scripts manipulate 3D objects imported into the game world using C++ functions of the scripting interface.

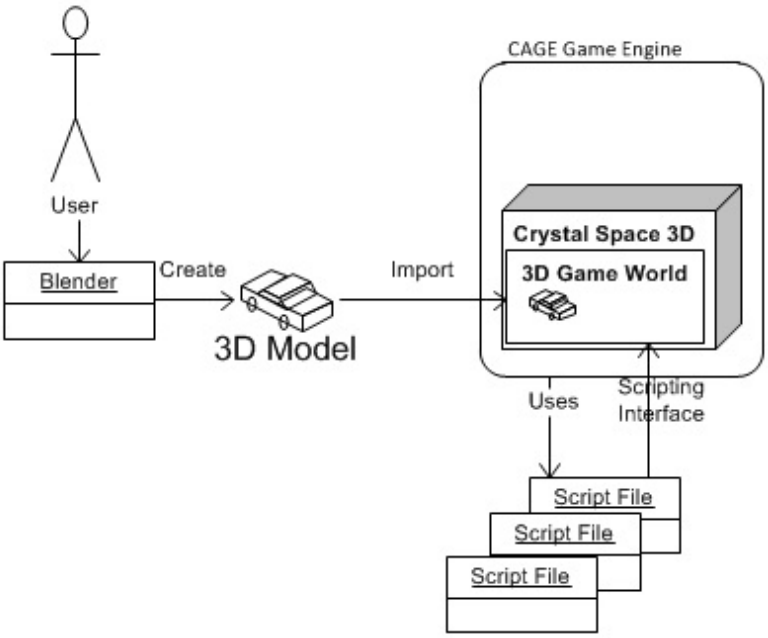

Figure 5.2: Creating and importing 3D model to CAGE

Luabind [43] library is used to generate wrapper code for classes and functions revealed in Lua interface of CAGE. Luabind uses a subset of Boost C++ library [11]. The version of CAGE used in the course includes all the needed libraries and external components in one package to assist students in installing CAGE and saving them from searching correct libraries.

In addition, CAGE offers a state machine interface for implementing artificial intelligence. Artificial intelligence is a wide topic and it was not the focus of game programming course, so it was essential to offer a ready-made method for students to implement artificial intelligence. Later CAGE AI framework was extended. A method for creating more sophisticated AI agents was created and also generic tool support for creating these agents was added [30], [31].

5.1.2 Game Programming Course

This thesis has also a pedagogical contribution, which has not been reported before, since the CAGE game engine was used in the game programming course in three implementations (falls 2008-2010). There were about twenty students participating each year. The game programming course was credited at 6 credit points and the workload of the assignment was meant to be half of the course workload. The course took a single semester (sixteen weeks), and the students had almost all that time to complete the assignment. The last week was reserved for the exam and the assignment was presented to the students on the third week.

The assignment was to implement a strategy or CRPG in pairs. There were also more specific requirements for the game to be implemented:

1. There is a player character, which can be controlled using keyboard and possibly also mouse, which is presented by a 3D model in the game world.

2. The game world is presented from the first or third person perspective (or as a combination of these).

3. There are computer controlled NPCs present, and there are at least three different types of NPCs.

4. There is interaction between the player character and NPCs.

5. There are passive objects, buttons, weapons, etc. of at least three different types.

6. There are user interface components for the player characters attributes.

7. The game is real-time, but it is allowed to stop the game for commanding the player character.

8. There are obstacles in the game world, such as walls, mountains, etc.

9. There is a goal in the game, it can be played, and it is possible to pass the game.

The grading was based on quality of the work. The extent of the implemented game affected. Figure 5.3 shows a screenshot of one student[1] made game implemented for assignment of the game programming course.

There is a pedagogical reason to select one game engine to be the official one in a game programming course, because when one game engine is selected, course personnel are experts using that particular game engine. In this case, they can offer better guidance to the students [45]. We also gave the possibility to use a different game engine in the course if someone wanted, but the course personal could not offer as good support for those.

Feedback was gathered from students taking the course. This student feedback is discussed more detail in [88]. The main findings from the feedback were:

1. The course provided experience of using real game programming tools.

2. It was easy to create a moving hero using CAGE.

[1]The game was made by students Jukka Paukkeri and Antti Veisu in game programming course of year 2010.

Figure 5.3: A game implemented as an assignment for the game programming course.

3. Ready-made collision checks helped in the beginning of the project.

4. The code of CAGE was not bug-free and the external libraries did not always work as documentation claimed.

5. Use of a scripting language provides very fast testing of code, since compiling is not needed, but the error messages are poor or do not exist.

One finding from game programming courses was that fast start is essential. Almost all groups quitting the course, do it in a very early stage, so if the use of a game engine requires reading massive documentation before the students can see any results, they will most likely quit the course.

5.2 Publication II: Guidelines for Personalizing the Player Experience in CRPGs

The second publication [II] presents a set of guidelines to personalize the player experience of CRPGs. These guidelines are divided into three groups: personalization of game storyline, of characters, and of game difficulty. The guidelines assist game developers to personalize the player experience. The tool support for personalization is discussed in the paper too.

Personalization of game storyline can be achieved by adapting the storyline to the decisions of the player. When the player makes a choice, the storyline can adapt to this choice, some later quest can be opened or be omitted. The results will be shown only later in game, to prevent the player testing all possibilities by just loading the game. The Witcher [74] is a good example of this kind storyline adaptation. In the game, the decisions of the player has consequences, which appear after 10-20 hours of gameplay. Effectively eliminating load–save tactic. In addition, quest generation can increase storyline personalization. The attributes of the player character can be used as basis of the quest generation process, and produce suitable quests for the player character. Similarly tutorial chapter of the game should be selected based on the player character, typically based on the profession of the character. Although the tutorial can be entirely or mostly pre-coded, since there are usually only couple of different professions available.

Personalization of characters is mostly achieved by proper AI behaviour. NPCs can be programmed to treat the player character differently based on appearance and reputation of the player character. Currently, this is not handled well in CRPGs, but believable NPCs reactions can create illusion of the living world. Also character backgrounds are not explored properly in modern games, however these offer a good source for quests and larger storyline parts. Hidden hints of game quests or facts can be added to the background, and make it effect to the game.

Personalization of game difficulty is a complex task. The game should be challenging enough, but not too difficult. A difficulty of CRPGs is mostly dependent on the difficulty of combats. The difficulty of these can be based on

the fighting proficiency of the player character. Separate fighting proficiency score should be used instead of experience level, since otherwise the game becomes too difficult for characters of non-combat oriented professions. Game mechanics must be tested carefully because they are important factor in game balancing and bugs or miscalculations in those can easily ruin the player experience entirely. In addition to eight presented guidelines, tool support for personalization is discussed in the paper.

Most CRPGs are collaborative games, which makes implementing personalization more difficult. When party consists of many members, storyline parts cannot be selected or generated only basis on the class or other attributes of the main character, because there is no a single main character. Also NPCs reactions cannot be based on this single character, but to the whole group, which makes these calculations more complex.

5.3 Publication III: Generative Approach for Extending CRPGs at Run-Time

The third publication, [III] presents a content generator program. CRPGs contain huge virtual worlds and those must be filled with objects and NPCs. A part of this task can be done using content generators. The presented generator program uses XML descriptions of game world areas and creates Lua scripts based on those at run-time. These scripts are performed and objects inserted into the game world. Using the presented approach it is possible to reduce loading times of large areas containing many game world objects. The CAGE game engine has been used as the context of the generator program.

We also made performance tests for object generation process, to ensure that it was effective enough to be used. The presented approach was applicable to be used in typical gaming PC. The most of the time was spent on the code handling placing the objects in to the game world. This code is inside of the C++ core of CAGE and therefore the time cannot be reduced further in the generator program. For example, with sixty objects, placing their 3D

models into the game world took seventy times longer than the generator program spent on creating them.

5.4 Publication IV: Personalized Side-Quest Generation for CRPGs

The fourth publication [IV] presents a side-quest generator program tackling also the personalization problem. The generator is based on the ideas presented in [II]. When using the approach, the application developer defines quest prototypes for CRPG, each of those is meant to be instantiated several times. The generator uses attributes of the player character as basis for quest generation to produce suitable side-quests. The changes for encountering a quest is calculated using the attributes of the player character. In addition, blank spots of the quest prototypes are filled based on the attributes of the player character.

The presented example quests were simple side-quests. Although the generator could have been developed even further and possibly also quests in critical path could be generated instead of only side-quests [90]. However when creating quests into the critical path, the generation process must be tested extensively, since errors or inconsistent quests can prevent advancing in the game or ruin the player experience.

5.5 Publication V: Browser Games for Online Communities

The fifth publication [V] discusses browser games and online communities built around those. The used definition browser game is a subcategory of browser-based games: the definition includes, e.g., a multiplayer requirement and existence of a persistent game world. The article presents properties of browser games, browser game types, and includes a short introduction

to technologies used in browser games. It also discusses social communities present in browser games and financial opportunities present in those.

The fact that browser games are played inside web browser lowers the starting threshold of these games considerably luring many players that have never bought a single computer game. The persistent world of browser games ensures that the same players are playing the game for long time. This builds player communities around these games. The player communities are an important aspect in browser games, especially in strategic browser games, where players form guilds to battle each other. These games have unique player group, which consists of a continues flow of new players and static group of core players.

Browser games are normally free to play, therefore the income of game developers must come from the other sources. Common income sources in this genre are selling advertisement space and selling extra features to the players. These income sources and business model of browser-based games is discussed in more detail in [87], [89].

5.6 Publication VI: On the Development of Browser Games – Current Technologies and the Future

The sixth publication [VI] presents technologies used to implement browser games. Both main technology directions, server-side scripting and application run-times are introduced. Also, accessories used in browser-based games are presented. Such as application platforms, game engines and browser plugins affecting game graphics. Especially game engines for browser-based games will be very important tools in the future. Components and properties for a such game engine are presented in the article. In addition, the article discusses the future of browser games.

Server-side scripting is the traditional way of implementing browser-based games. This approach has many advantages, for example many server-side scripting languages enable easy accessing to the database. Easy database

access is required because browser-based games contain a lot of game objects, which needs to be stored into the database for permanent storage. In addition, many browser-based games had strategic nature: the game consists of mostly static web pages presenting game data, and this kind of web pages with good database support are very easy to create using server-side scripting.

However, an increasing number of modern browser-based games contain high-quality graphics, and server-side scripting is not the most efficient way of producing those. Many scripting languages have only limited graphics capabilities. On the other hand, application run-times allow almost as good graphics as most modern game-engines can produce to be run in the Web browser, although they require explicit installation of the application run-time into the browser. Naturally, all browser-based games implemented using the same application run-time can be played with the same installation.

Game engines are the next step in development of browser-based games. Although many modern browser-based games are developed using different frameworks and libraries. These are not yet sophisticated enough to be called game engines. Browser-based games have many common functionalities, which can be implemented in the game engine level, instead of the game code.

5.7 Publication VII: Content Generation in a Collaborative Browser-Based Game Environment

The seventh publication [VII] discusses content generation in browser environment and present a prototype multiplayer browser-based game featuring content generation. Procedural content generation is a common method in installable computer games, but it is rarely used in browser-based games. Integration of procedural content generation can be beneficial for browser-based, for example it can extend the time the player plays the game, since there is always new content. We describe the architecture of Web games,

typically a three-tier architecture (client-side, server-side and database) [59] and discuss architecture of real-time multiplayer games in a browser. In our findings we propose that real-time collaboration is still uncommon in current browser-based games and the games relay on indirect communication, where small lags and delays do not harm the player experience.

In our prototype implementation we used node.js framework. It contains a built-in HTTP server library, which enables running web server without other external software. Node.js enables the entire web application to be developed in JavaScript, both server-side and client-side. In this project we created a simple prototype CRPG, which used content generation to produce quests at run-time into the game.

Chapter 6

Conclusions

This chapter concludes the thesis by summarizing and revisiting the main results of the work. Future work that can be based on this research is outlined also.

6.1 Summary

Software tools are an inseparable part of modern game development. Substantial part of game developers' time is actually spent on implementing the tools to be used in the game development project instead of the end product. Code and tool reuse has been an important aspect in game industry already in the past decade. This thesis describes and addresses numerous issues regarding software tools in game development. Unique problems in CRPGs development are described and different tools to solve them presented.

Chapter 2 includes broad introduction to CRPGs. It begins with history of PnP RPGs and continues to the history of CRPGs. Furthermore it presents professions of game development teams and components of a modern computer game. In this Chapter the unique problems in CRPGs development: content creation and personalization are presented.

Chapter 3 discusses software tools in game development, especially game engines and content generation tools are introduced. Some software tools are specific to CRPGs, and those are are also briefly discussed. Game design's

relation to software tools is discussed. It also presents related work on game engines.

Chapter 4 discusses run-time environment of CRPGs. In addition to traditional binary environment, CRPGs can be develop into browser environment. In the future most likely browser-based implementation are even more common. In the Chapter, scrip usage for CRPGs is also discussed. Browser-based games are divided into several categories based on their properties, and one categorisation is presented. In addition, development technologies of browser-based games and game engines for browser-based games are introduced.

Chapter 5 gives an introduction to included publications. It also presents game programming course, where the CAGE game engine was used as a tool for students to implement their project works.

6.2 Research Questions Revisited

This dissertation addresses three research questions. This Section revisits the questions and gives a brief summary of the main results.

RQ1. What kind of tool support is advantageous to support software developers' work when implementing CRPGs?

Normally the development process of modern computer game is too laborious to be financially beneficial without code reuse. The benefits of game engines have been proven, since those have been used for three decades. The first in-house game engines were developed in 1980s. Basically game engines are software product-lines, which are a widely used approach to increase the productivity of software development [12], [16]. Modelling tools are also commonly used in CRPGs development, to create 3D graphics.

Different kinds of content generators can reduce time spent on application development. They can be used to create game content, e.g., quests, NPCs, and game levels. They can be used at development time, when application developer can verify the results and perform generation again, if he or she is not satisfied with the results. Another option is run-time content generation.

In this case, verifying the results is not possible, because they are produced at the same time when the player is playing the game. However, run-time content generation allows that all content is not needed to import into the game and in theory game content can be endless. Content generators can be advantageous in personalization of CRPGs also.

RQ2. How to simplify and support the creation of personalized player experiences in CRPGs?

Personalized player experience is one main factor when capturing and maintaining interest of the player. However it is also very difficult to create, since computers require explicit instructions.

Software tools can be very useful in personalization, the player character data can be used as a basis of the generation process. This way it is possible to create suitable quests or levels for the player character. Also the difficulty of a level or a quest can be adjusted by the fighting capabilities of the player character. Also the storyline of the game can be adapted based on the player actions. Quests can be opened or omitted based on the choices the player character makes.

Game world NPCs can be programmed to treat different kinds of player characters differently based on their knowledge of the player character, appearance, or behaviour. All these operations increase the illusion level of living and responsive game world. In addition, game design can support personalization. Personalized tutorial chapters, character background utilizing and use of game mechanics that use different character mechanics all produce more personalized player experience. Also good separation of code and game data can help, since more content can be easily added later, possible over the network also.

RQ3. What kind of tool support can be used to assist in the development of CRPGs for the browser-based environment that is constantly alive online?

Currently there are numerous different frameworks and tools that can be used in web-application development. It is also typical that many different

frameworks and tools are used in a single application. This applies also to browser-based CRPGs. However, it is important that fewer software tools and frameworks would be used in a single game project, instead of current software combinations, since using too many different systems makes application development complicated and hinders maintenance. Currently there are a few good software tools that can offer an easy way to implement a web server, e.g., Node.js [52] and assist the development in other ways. For example, Now.js [53] makes server and client to look like a single programming environment from the view of application developers.

Game engines for browser-based games are most likely the next step in the development of browser-based games. Currently many software tools are libraries are used, but those are not yet extensive enough to be called as game engines. Game engines for browser-based games were addressed in Chapter 4 and in more detail in Publication [VI]. Game engines can offer ready-made platforms for web-development and reduce the need for several software frameworks and libraries.

Procedural content generation is a common method in games developed in binary environment. However, it is beneficial also in browser environment, although it is not yet commonly used. Most likely it is common method also in the browser-based CRPGs in the future. Content generation for browser-based games was discussed in Publication [VII]. Content generation can be equally advantageous for personalization in browser-based environment as in traditional binary environment.

6.3 Future Work

There were several directions where the research could be extended. Especially for browser-based CRPGs part offers many interesting research possibilities. New JavaScript and HTML5 features enable building new kind of tools and frameworks. Also the content generation in the web environment presented in Publication [VII] was only a scratch on the surface and combined with HTML5 features, those enable creation of imposing browser-based games.

Generally quest generation tools are only in a elementary phase, although many authors have created tools to generate side-quests. However, these could be developed further and create a quest generator to produce quests into the mandatory path of CRPG [90].

Since software tools are used by application developers, the usability aspect could also be one direction for further research. For example, a test group of application developers would perform certain tasks using a tool, and give their opinion about using it.

Bibliography

[1] E. Aarseth. From Hunt the Wumpus to EverQuest: Introduction to Quest Theory. In *Entertainment Computing - ICEC 2005*, volume 3711 of *Lecture Notes in Computer Science*, pages 496–506. Springer Berlin Heidelberg.

[2] E. Adams. *Fundamentals of Game Design*. New Riders, 2009.

[3] A. Ampatzoglou and I. Stamelos. Software Engineering Research for Computer Games: A Systematic Review. *Information and Software Technology*, 52(9):888 – 901, 2010.

[4] D. Anderson. The Dark Side of MMOGs: Why People Quit Playing. In *Proceedings of CGAMES'2009*, pages 76–80, 2009.

[5] M. Anttonen, A. Salminen, T. Mikkonen, and A. Taivalsaari. Transforming the Web into a Real Application Platform: New Technologies, Emerging Trends and Missing Pieces. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 800–807, New York, NY, USA, 2011. ACM.

[6] Banned / Restricted Lists for DCI-Sanctioned Magic: The Gathering Tournaments. `https://www.wizards.com/magic/magazine/article.aspx?x=judge/resources/banned`, accessed Aug. 9, 2013.

[7] M. Barton. *Dungeons and Desktops: The History of Computer Role-playing Games*. A. K. Peters., 2008.

[8] Battlestar Galactica online, browser-based game. `http://fi.battlestar-galactica.bigpoint.com/`, accessed Oct 31, 2012.

[9] BigWorld Technology. http://www.bigworldtech.com/, accessed Oct 18, 2013.

[10] Blender, the free open source 3D content creation suite. http://www.blender.org/, accessed Jun 15, 2012.

[11] Boost C++ library. http://www.boost.org/, accessed Dec 10, 2011.

[12] J. Bosch. *Design & Use of Software Architectures: Adopting and Evolving a product-Line Approach*. Addison-Wesley., 2000.

[13] Brian Schwab. *AI Game Engine Programming*. Charles River Media, Inc., 2004.

[14] M. Buckland. *Programming game AI by example*. Wordware Publishing, Inc., 2005.

[15] Civilization IV, official web page. http://www.2kgames.com/civ4/, accessed Apr. 3, 2012.

[16] P. Clements and L. Northrop. *Software Product Lines : Practices and Patterns*. Addison-Wesley., 2002.

[17] CommonJS, a project defining APIs for JavaScript outside the browser. http://www.commonjs.org/, accessed Mar. 9, 2013.

[18] Crystal Space 3D Engine. http://www.crystalspace3d.org/main/Main_Page, accessed Dec. 29, 2011.

[19] Diablo III, official web page. http://eu.battle.net/d3/en/, accessed Oct. 26, 2012.

[20] Doom: The Boardgame rules. http://www.fantasyflightgames.com/ffg_content/Doom/doomrules.pdf, accessed Sep. 25, 2013.

[21] J. Doran and I. Parberry. A prototype quest generator based on a structural analysis of quests from four mmorpgs. In *Proceedings of the Second International Workshop on Procedural Content Generation in Games*, pages 1–8, July 2011.

[22] Dwarf Fortress, web page. http://www.bay12games.com/dwarves/, accessed Jun. 4, 2011.

[23] Extensible Markup Language (XML) 1.0 (Fifth Edition). http://www.w3.org/TR/REC-xml/, accessed Jan. 14, 2013.

[24] General-purpose scripting language. http://php.net/, accessed May. 17, 2013.

[25] J. Gregory. *Game Engine Architecture*. A K Peters, Ltd., 2009.

[26] G. Gygax. *Master of the Game*. The putnam Publishing Group, 1989.

[27] N. Hallford and J. Hallford. *Swords & Circuitry: A Designer's Guide to Computer Role-Playing Games*. Premier Press, Incorporated, 2001.

[28] Hattrick online football manager game. http://www.hattrick.org/, accessed Jun. 4, 2011.

[29] T. J. Heinimäki. Technology Trees in Digital Gaming. In *Proceeding of the 16th International Academic MindTrek Conference*, MindTrek '12, pages 27–34, New York, NY, USA, 2012. ACM.

[30] T. J. Heinimäki and J.-M. Vanhatupa. Layered Artificial Intelligence Framework for Autonomous Agents. In *Proceedings of 12th Symposium on Programming Languages and Software Tools (SPLST'11)*, pages 102–113. Institute of Cybernetics at Tallinn University of Technology, October 2011.

[31] T. J. Heinimäki and J.-M. Vanhatupa. Implementing Artificial Intelligence: a Generic Approach with Software Support. *Proceedings of the Estonian Academy of Sciences*, 62:27–38, 2013.

[32] T. Honkala. Legendary Games: Gauntlet. *Pelit, a Finnish video game magazine*, 8:49–52, August 2012.

[33] J. Howard. *Quests: Design, Theory, and History in Games and Narratives*. A K Peters, Ltd., 2008.

[34] HTML5 standard. http://www.w3.org/html/wg/, accessed Oct. 31, 2012.

[35] Id Software Doom. http://www.idsoftware.com/games/doom/, accessed May. 16, 2013.

[36] Id Software Quake. http://www.idsoftware.com/games/quake/, accessed May. 16, 2013.

[37] James Whitehead II and Rick Roe. *World of Warcraft Programming*. Wiley Publishing, Inc., 2010.

[38] P. Järvinen. *On Research Methods*. Tampereen yliopistopaino Oy, 2004.

[39] B. Kerniqhan and P.J.Plauger. *Software Tools*. Addison-Wesley, 1976.

[40] P. Lankoski and S. Björk. Gameplay Design Patterns for Believable Non-Player Characters. In *Situated Play: Proceedings of DiGRA 2007 Conference*, pages 416–423, September 2007.

[41] LiveScience online article: 10 Animals That Use Tools. http://www.livescience.com/9761-10-animals-tools.html, accessed Mar. 26, 2013.

[42] Lua programming language. http://www.lua.org/, accessed Dec. 10, 2011.

[43] Luabind wrapper library. http://www.rasterbar.com/products/luabind.html, accessed Dec. 10, 2011.

[44] Magic: The Gathering volume has doubled since 2008. http://www.icv2.com/articles/news/21471.html, accessed Jun. 18, 2012.

[45] M. Masuch and L. Nacke. Power and Peril of Teaching Game Programming. In *Proceedings of Computer Games: Artificial Intelligence, Design and Education, 5th Game-on International Conference*, pages 347–351. University of Wolverhampton UK.

[46] S. P. M. R. Maung. K. Sein, Ola Henfridsson and R. Lindgren. Action Design Research. *MIS Quarterly*, 35:37–56, June 2011.

[47] M. McNaughton, M. Cutumisu, D. Szafron, J. Schaeffer, J. Redford, and D. Parker. ScriptEase: Generative Design Patterns for Computer Role-Playing Games. In *19th IEEE International Conference on Automated Software Engineering (ASE'04)*, pages 88–99, September 2004.

[48] Meteor, open-source platform. `http://www.meteor.com/`, accessed Aug. 15, 2013.

[49] Microsoft Visual Studio. `http://www.microsoft.com/visualstudio/`, accessed Mar. 16, 2013.

[50] Microsoft Windows. `http://windows.microsoft.com/en-us/windows/home`, accessed Mar. 20, 2013.

[51] I. Millington. *Artifial Intelligence for Games*. Morgan Kaufmann Publishers., 2006.

[52] Node.js platform. `http://nodejs.org/`, accessed Mar. 5, 2013.

[53] NowJS library. `http://nowjs.com/`, accessed Mar. 5, 2013.

[54] T. Nummenmaa, E. Berki, and T. Mikkonen. Exploring Games as Formal Models. In *Proceedings of the 2009 Fourth South-East European Workshop on Formal Methods*, SEEFM '09, pages 60–65, Washington, DC, USA, 2009. IEEE Computer Society.

[55] T. Nummenmaa, A. Kultima, K. Alha, and T. Mikkonen. Applying Lehman's Laws to Game Evolution. In *Proceedings of the 2013 International Workshop on Principles of Software Evolution*, IWPSE 2013, pages 11–17, New York, NY, USA, 2013. ACM.

[56] E. M. Ollila. Using Prototyping and Evaluation Methods in Iterative Design of Innovative Mobile Games. *Doctoral thesis*, 2009.

[57] Panda 3D Game Engine. http://www.panda3d.org/, accessed Nov. 1, 2012.

[58] Paranoia RPG official homepage, Mongoose publishing. http://www.mongoosepublishing.com/rpgs/paranoia.html, accessed Sep. 25, 2012.

[59] R. Peacock. Distributed Architecture Technologies. *IT Professional*, 2:58–60, May 2000.

[60] Persistent browser-based games, a genre definition. http://www.pbbg.org/, accessed Jun. 4, 2011.

[61] S. Rabin. *AI Game Programming Wisdom 3*. Charles River Media, 2006.

[62] Richard Rouse III. *Game Design: Theory & Practice*. Wordware Publishing, Inc., 2005.

[63] D. Schultheiss. Long-term motivations to play MMOGs: A longitudinal study on motivations, experience and behavior. In *Situated Play: Proceedings of DiGRA 2007 Conference*, pages 344–348. The University of Tokyo, September 2007.

[64] D. Schultheiss, N. D. Bowman, and C. Schumann. Community vs soloplaying in multiplayer internetgames. In *Proceedings of The Player Conference 2008*, pages 452–471, 2008.

[65] Shadows over Camelot board game, official homepage. http://www.daysofwonder.com/shadowsovercamelot/en/, accessed Mar. 6, 2013.

[66] A. R. Shankar. *Pro HTML5 Games*. Apress, 2012.

[67] J. Smed. *Algorithms and Networking for Computer Games*. John Wiley & Sons, Ltd, 2006.

[68] Space Hulk Boardgame, Games Workshop. http://www.games-workshop.com/gws/catalog/productDetail.jsp?prodId=prod210009a, accessed Sep. 25, 2013.

[69] A. Sullivan, A. Grow, T. Chirrick, M. Stokols, N. Wardrip-Fruin, and M. Mateas. Extending CRPGs as an Interactive Storytelling Form. In *The Fourth International Conference on Interactive Digital Storytelling (ICIDS)*, volume 7069 of *Lecture Notes in Computer Science*, pages 164–169. Springer, 2011.

[70] A. Sullivan, M. Mateas, and N. Wardrip-Fruin. Rules of Engagement: Moving Beyond Combat-based Quests. In *INT3 '10: Proceedings of the Intelligent Narrative Technologies III Workshop*. ACM, 2010.

[71] A. Taivalsaari, T. Mikkonen, M. Anttonen, and A. Salminen. The Death of Binary Software: End User Software Moves to the Web. In *Proceedings of The Ninth International Conference on Creating, Connecting and Collaborating through Computing (C5'11)*. IEEE Computer Society, January 2011.

[72] The Elder Scrolls IV: Oblivion, official website. `http://www.elderscrolls.com/oblivion/`, accessed Apr. 3, 2012.

[73] The Making Of: Rogue, article in Edge magazine (online). `http://www.edge-online.com/features/making-rogue/`, accessed March. 8, 2013.

[74] The Witcher CRPG. `http://www.thewitcher.com/`, accessed May. 21, 2013.

[75] J. Togelius, T. Justinussen, and A. Hartzen. Compositional Procedural Content Generation. In *Proceedings of the FDG Workshop on Procedural Content Generation (PCG)*. ACM, May/June 2012.

[76] J. Togelius, N. Shaker, and M. J. Nelson. Introduction. In N. Shaker, J. Togelius, and M. J. Nelson, editors, *Procedural Content Generation in Games: a Textbook and an Overview of Current Research*. pcgbook.com, 2014.

[77] Torque 3D game engine. `http://www.garagegames.com/products/torque-3d`, accessed Jun. 15, 2012.

[78] Trading Card Game: Magic The Gathering. http://www.wizards.com/Magic/, accessed Jul. 4, 2011.

[79] Traveller RPG official homepage, Mongoose publishing. http://www.mongoosepublishing.com/rpgs/traveller.html, accessed Feb. 28, 2013.

[80] Travian strategy browser-based game. http://www.travian.com/, accessed Jun. 4, 2011.

[81] M. Tresca. *The Evolution of Fantasy Role-Playing Games*. McFarland & Company, Inc., 2011.

[82] A. Tychsen. Role Playing Games: Comparative Analysis Across Two Media Platforms. In *Proceedings of the 3rd Australasian conference on Interactive entertainment*, IE '06, pages 75–82, Murdoch University, Australia, Australia, 2006. Murdoch University.

[83] A. Tychsen, M. Hitchens, T. Brolund, and M. Kavakli. The Game Master. In *Proceedings of the second Australasian conference on Interactive entertainment*, IE 2005, pages 215–222, Sydney, Australia, Australia, 2005. Creativity & Cognition Studios Press.

[84] A. Tychsen, S. P. Tosca, and T. Brolund. Personalizing the Player Experience in MMORPGs. In *Technologies for Interactive Digital Storytelling and Entertainment, 3rd International Conference*, pages 253–264, 2006.

[85] Unity Game Development Tool. http://unity3d.com/, accessed Jun. 10, 2011.

[86] Unity Web Player. http://unity3d.com/webplayer/, accessed Mar. 5, 2013.

[87] J.-M. Vanhatupa. Business Model of Long-term Browser-based Games – Income without Game Packages. In *Proceedings of 7th International Conference on Next Generation on Web Services Practices (NWeSP'11)*, pages 369–372. IEEE, October 2011.

[88] J.-M. Vanhatupa. Game Engines in Game Programming Education – Experiences from Use of the CAGE Game Engine. In *Proceedings of 11th International Conference on Computing Education Research*. ACM, November 2011.

[89] J.-M. Vanhatupa. Business Model of Long-term Browser-based Games – Income without Game Packages. *International Journal of Computer Information Systems and Industrial Management Applications*, 5:195–202, June 2012.

[90] J.-M. Vanhatupa. Towards Extensible and Personalized Computer Role-Playing Game Fantasy Worlds. In *Proceedings of 4th Computer science and Electronic Engineering Conference (CEEC'12)*, pages 187–190. IEEE, September 2012.

[91] Warhammer Fantasy Battle tabletop wargame. `http://www.games-workshop.com/gws/catalog/landing.jsp?catId=cat440002a&rootCatGameStyle=wh`, accessed Oct. 17, 2013.

[92] WebGL library. `http://www.khronos.org/webgl/`, accessed Oct. 31, 2012.

[93] Windows Phone. `http://www.windowsphone.com/en-us`, accessed Mar. 20, 2013.

[94] World of Warcraft News. `http://www.curse.com/articles/world-of-warcraft-news/956087.aspx`, accessed Jun. 4, 2011.

[95] World of Warcraft, official site. `http://eu.battle.net/wow/en/`, accessed Jun. 4, 2011.

[96] A. Wu. QuakeC Basics. `http://www.bluesnews.com/guide/quakec2.htm`, accessed Aug. 9, 2013.

[97] Xbox 360. `http://www.xbox.com/en-US/`, accessed Mar. 20, 2013.

[98] XNA Game Studio. `http://msdn.microsoft.com/en-us/library/bb200104.aspx`, accessed Oct. 31, 2012.

[99] Y. Zhang. Developing Effect of HTML5 Technology in Web Game. *International Journal on Computational Sciences & Applications (IJCSA)*, 2:21–32, 2012.

Publication I

J-M. Vanhatupa and T. J. Heinimäki. Scriptable Artificial Game Engine for Game Programming Courses. In *Proceedings of 4th Informatics Education Europe (IEE IV)*, pages 27–32. University of Freiburg, November 2009.

Scriptable Artificial Intelligent Game Engine for Game Programming Courses

Juha-Matti Vanhatupa
Department of Software Systems
Tampere University of Technology
Tampere, Finland
juha.vanhatupa@tut.fi

Teemu J. Heinimäki
Department of Software Systems
Tampere University of Technology
Tampere, Finland
teemu.heinimaki@tut.fi

ABSTRACT
Laborious programming assignments are often an important part of university game programming courses. However, if students must build their game projects from scratch, they are forced to use a lot of time on secondary issues for the game programming itself. In this paper we present an approach for easing the student workload by using a scriptable game engine, which readily offers basic computer graphics and artificial intelligence functionality. We have implemented a game engine called CAGE to test our approach. It has been evaluated in a game programming course of Tampere University of Technology. We gained very encouraging results and at the moment we are continuing the development of the CAGE game engine for the upcoming game programming course as well.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer Science Education

General Terms
Algorithms, Design, Human Factors

Keywords
Game Programming Education, 3D Game Engine, Lua

1. INTRODUCTION
Game programming is a difficult topic for the academic education. The subject is scattered and includes several issues, such as game design, artificial intelligence (AI), computer graphics and scripting languages. The creation of a new computer game (or even a demo) from scratch is an enormous task for students, because of required implementation of graphics, AI, and other functions commonly covered in other courses.

In this paper we present an approach to reduce the student workload in implementing graphics and AI in game programming tasks. We promote the use of a game engine, which offers the basic game loop, advanced computer graphics handling, basic functionality for AI and a scripting interface for rapid application development. Students working in the game industry some day will most probably use some ready-made game engine. C++ programming language and Lua scripting language [8] are also common tools in the game industry nowadays and that is why we have chosen to use those when implementing our game engine. This way we can also satisfy some of the industry demands for game programming education.

Lua is a powerful general-purpose, dynamically typed scripting language, which has an excellent performance for such. It can be used with many programming languages, for example C++, C# and Java. It is nowadays used in many popular computer games including Warhammer Online: Age of Reckoning [16] and The Witcher [17]. World of Warcraft [18] uses Lua for user interface scripting. Scripting languages are good tools for AI programming, as they allow fast testing and modularity for AI code.

Generally AI is an interesting, but difficult topic for students. It can steal a lot of implementation time from other topics, especially when implementing AI functionality for the first time. Even if the students are very motivated for implementing their AI functionality, they might run out of time for getting the whole game project implemented, because such a project contains also a lot of other things to do. Our game engine offers a basic functionality for AI and students can use it in their game projects easily. They can, of course, also implement a more sophisticated AI functionality by themselves, if they have time and skills for that.

We have implemented a game engine called CAGE and evaluated it in the game programming course organized at the Tampere University of Technology in fall 2008. The results from the course were very encouraging. We achieved a low drop-out percentage and the overall quality of the student game projects was good. We also gained valuable information to improve our engine for upcoming years.

We proceed as follows. In Section 2 we present our vision on teaching game programming in more detail. In Section 3 we discuss the current implementation of the CAGE tool. In Section 4 we present a case study. Finally, related work is discussed in Section 5 before some concluding remarks in Section 6.

2. BASIC APPROACH
We claim that the use of an easily understandable and flexible game engine with a scripting interface is an excellent way to improve learning in game programming courses. Such a ready-made game engine with good graphics support lets the students use advanced computer graphics without biasing their focus too much into the area of proper computer graphics courses. Offering a scripting language interface speeds up the development process considerably, and if the game engine offers also a basic functionality for AI, the students do not need to waste time in implementing the topics covered in proper artificial intelligence courses, but can use the game engine functions. However, they

should also be able to implement sophisticated artificial intelligence systems by themselves, if they are already familiar with the topic.

When one game engine is promoted to be the course official one, the assistants can offer better help for the students, because they are experts in using that particular game engine. Same conclusions are drawn also in [11]. We decided to implement a novel game engine by ourselves, because there were no satisfying candidates written in C++ offering a good ready-made Lua interface and artificial intelligence functionality suitable to our needs.

In our course the students implement their computer games with Lua scripts possibly also extending the CAGE framework with C++. They make their choice of implementing some feature with C++ or Lua by themselves. A simple shooter or an adventure game could be written entirely using Lua. Possibility to extend the C++ framework offers variability and there are no limitations to the game genre and features. The layer architecture of CAGE is represented in Figure 1. One design factor when implementing CAGE was that the students should be able to start the implementation of their games by writing simple Lua scripts. Later on they could move to extending the C++ framework after realizing, what kind of functionality would really be needed. At this stage they should also have learned the proper use of the interface.

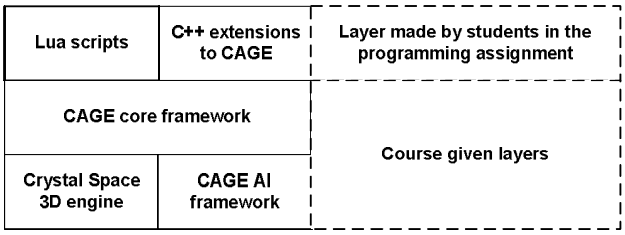

Figure 1. CAGE layer architecture

The Lua interface of the CAGE engine offers a set of C++ functions, callable from Lua script files. The interface offers basic functionalities needed for simple games. The functions are expressive enough to e.g. create new non-player characters (NPCs), modify their behavior, declare items, affect the game world attributes, and declare keyboard and mouse commands of the game.

CAGE also has a basic AI functionality based on state machines. The framework has a state machine prototype that makes it possible to declare state machines easily on Lua script files. State machines can be used to create a simple AI and, for example, shooter games can be implemented using state machines only.

Creating three-dimensional (3D) models and worlds for our engine can be done easily using an external 3D modelling software. Thus, the nature of the game development work using CAGE is separable; it can be divided among the modelling artists and the programmers involved, even in such a way that the modellers do not have to know anything about programming and vice versa. If the focus of a game programming course is in teaching game design or game logics programming, it is possible to use only pre-made 3D models.

One way – and the way we recommended students to use in our course – to create worlds and other 3D models compatible with CAGE is to model them using Blender [4] and export them into usable form using blender2crystal [5] Blender plugin. The easiness of creating their own 3D material allows the students to better devote themselves to other game programming matters when creating their games. Designing and modelling of a 3D world was planned to be one part of our course assignment, although not a very large part, because our focus is in general game programming, not in 3D modelling.

3. CAGE – TOOL IMPLEMENTATION

CAGE is implemented using the C++ programming language and Lua scripts. The graphics of CAGE are implemented using Crystal Space [7], a cross-platform software development kit for real-time 3D graphics. One main reason for choosing Crystal Space was the easiness of importing home-made models and worlds. In the Lua interface of CAGE, Luabind [9] is used to generate the wrapper code needed for the functions and classes. Luabind also uses a minimal subset of Boost C++ library [6]. For the course-given version of CAGE we imported all needed libraries into one package in order to spare the students from the burden of finding and installing them.

The original C++ parts of CAGE were implemented during the summer 2008 using Visual Studio 2005 as integrated development environment (IDE). However, the CAGE code itself is platform-independent and the external libraries used are freely distributed and ported for different platforms including Linux. Thus, CAGE can be compiled and used in different systems with only a little effort. In the summer 2009 the CAGE solution was ported to be further developed using Visual Studio 2008.

The Lua scripting interface of CAGE allows configuring the game world settings quite freely. The 3D representation of the world can be loaded from the scripts and general settings (like gravitation) can be easily set. One can also create dynamically new actor objects, such as creatures, structures or items and set their behavior and properties. The camera behavior is scriptable and the input handling can be done in Lua scripts also. Furthermore, the state machines and different timers and triggers can be used from the scripts. This wide Lua interface allows the game genre and the world behavior to be selected freely.

In our scripting interface a set of functions is given for making different queries about the world state or about the properties of individual actors. Thus, all the information needed for building the game logics into the Lua side of the game can be reached without difficulties. There are also functions for creating and modifying different elements for generating a pleasant graphical user interface. Although the main purpose of our interface is to provide general functionalities usable in totally different games, there are also some more or less genre-specific functions for providing frequently used functionalities without a need to script them at all. These functions allow, for example, checking whether there are objects between certain NPC and the player character. This is necessary for a shooter game, but probably not important at all in some other game genre.

As an example, Listing 1 shows a piece of a Lua script creating three NPCs patrolling around an ammunition trunk. Their mission is to guard it; therefore they want to stop the player character that comes too close and attack if needed. The functions *circular_patrol_around* and *guard* can be seen as genre-specific functions, which could have easily been implemented also in Lua using more general parts of our interface.

Listing 1. Piece of a Lua script creating three NPC guards

```
for i=1,3,1 do
 --Basic initialization in Lua function.
 --X- and Z-coordinates as parameters.
 --The new actor is saved to table "sc".
 spawn_state_creature(ammox+2*i,ammoz)
 --Setting state machine...
 sc[i]:get_state_machine():SetCurrentState(
State_Patrol)
 --Setting the path around our trunk
 --"ammo1" in clockwise direction:
 sc[i]:circular_patrol_around("ammo1",TRUE)
 --Environmental observation settings:
 sc[i]:observe_near_meshes(TRUE)
 sc[i]:set_mesh_observing_radius(10)
 sc[i]:set_actor_perception(TRUE)
 --If the guard sees "player1" closer than
 --5 units from "ammo1", the "warn"
 --function will be called.
 sc[i]:guard("ammo1","player1",5,"warn")
 --Closer than 2 units the "kill" is called
 sc[i]:guard("ammo1","player1",2,"kill")
end
```

As the course-given version of CAGE is started, a little example script creates a barren world, the player character and a NPC for demonstration purposes. The behavior of the NPC is modelled with an example state machine. There are also some user interface components to show how the user interface can be created from a script code. The player is able to shoot the NPC that changes to the attack state after getting hit. In the attack state the NPC pursues the player until hit again. After the second hit it disappears. We thought this was a good example state machine, with simple behavior, but demonstrating possibilities well. This starting point for the students is shown in Figure 2. Although this offered example is close to a shooter game, it does not limit the game genres of student projects in any way.

Figure 2. Starting point for the students using CAGE

The scripting interface of CAGE offers over 150 functions, which can be called from Lua scripts. Numbers of functions related to different CAGE classes are presented in Table 1.

Table 1. Functions in Lua interface classes

Class name or explanation	Number of functions offered	Additional information
Starting parameters	15	Also 5 changeable variables(e.g., the game world model directory)
World	56	
Actor	65	
Item	14	Subclass of Actor
Timer	4	
State Machine	3	

It is obvious that World and Actor are important classes in all computer game genres, and most of the functionality relates to those classes. The starting parameters relate to functionality, which performed during initialization of CAGE. The application developer can use those functions to modify properties of the CAGE. In addition there are five public variables, which can be changed directly from the Lua scripts. These contain directory paths for the game world model files and sounds.

4. GAME PROGRAMMING COURSE

CAGE was evaluated in a game programming course implementation at the Tampere University of Technology in the fall 2008. The course was optional and it could be included to the software engineering major. The course was targeted to final stage master students with knowledge of computer graphics and algorithms. It credited 6 credit units and the programming assignment was planned to be about half of the course workload. The course lecture topics included history of games, game industry, game architectures, game world physics, data structures used in games, sounds, computer graphics and artificial intelligence. Two lectures concerning game design and graphics were given by visiting lecturers from the industry.

The course had a programming assignment that was carried out in pairs. The given task was to make a computer game using CAGE. The developed game could be a first person shooter, an adventure game or a computer role playing game. Time to complete this assignment was limited to the course duration, 13 weeks. We allowed the students to use different game engines if they wished so. However, the ones ending up to this solution could not be offered as good assistance as those using CAGE.

The computer class environment for the course consisted of Visual Studio 2005, CAGE, Crystal Space, Blender, blender2crystal plugin and MakeHuman [10] for creating humanoid shapes easily. Although most students used their own computers for programming, the class was needed for the possible case of someone lacking the environment able to run the applications developed in Visual Studio IDE. CAGE can be used in different environments quite easily, but we wanted to offer the students a possibility to use it as such, in out-of-the-box way. The goal of our course was, after all, to teach game programming, not software composition and porting.

Initially there were twelve groups, and eight of these finally returned their game project. The drop-out percentage could have been better, but it has to be remembered that this was an optional course, which is easy to drop out if the mandatory ones need more effort. It is probable that the groups, which did not return their games, never actually even started their projects. None of the returned projects were even close to be rejected. This is not usually the case in the programming courses of our university.

One of the submitted game projects was a space simulator featuring movement with three degrees of freedom. In the game the player controls an alien ship researching Earth by dropping reconnaissance satellites. During his research flights the player encounters enemies, which must be fight against using the laser weapon of the ship. This space simulator game was a good example of variability of the CAGE engine.

The students praised the easiness of creating a moving hero, ready-made collision checks and modifiability of CAGE. As future improvements the students wished to have more extensive documentation of the scripting interface, ready-made sound support[1] and more precise error messages from their script files. Many students also mentioned the free topic for the implemented game as a good motivator for the course.

The next implementation of our game programming course will take place in the fall 2009. It will be roughly similar to the described one. However, CAGE has been slightly modified for the upcoming course. The main change has been replacing Visual Studio 2005 with Visual Studio 2008 as the C++ IDE. The program code itself has also been improved since the fall 2008 by refactoring it, removing unnecessary code and fixing several bugs. As the students of the previous course implementation required, the interface documentation has been improved considerably. In addition, some major novel features have been added to the development version of the CAGE AI framework, but the achieved advanced AI support will not be included into the course-given version of CAGE for now, for it is experimental and not yet fully tested or evaluated. On the other hand, too advanced AI features could distract the students from their desired focus.

[1] The sound support was intentionally left out of the Lua interface, because we wanted to encourage the students to at least throw a glance at the C++ side also.

5. RELATED WORK

Different demo programs have been used for educational purposes from the early days of the computer science. Compared to the demo programs [3, 12, 15] used in many courses and textbooks, our game engine is a complete platform for the students using it, and it is not just for demonstrating some single feature. Recently also demo game engines have appeared. Unlike SAGE - simple academic game engine [14], our approach supports the use of scripting and offers a basic functionality for AI.

Alice [1] is a teaching tool designed to be students' first expose to object-oriented programming. However, CAGE is designed for educating general game programming. Students learn when using CAGE in programming their games, not during playing in a virtual sandbox. CAGE is designed for advanced students already familiar with object-oriented programming.

STEAMiE educational game engine [13] is a versatile engine focusing strictly on educational content creation. It uses C++ as a "scripting language", as CAGE uses Lua with the purpose to teach the students to use a real scripting language. The authors of STEAMiE admit that script languages can be useful in implementing common scenarios, but criticize using them in complicated ones because of the language limitations. In our opinion, Lua is such a powerful and capable language that it can be successfully used even in complicated cases. If plain Lua is inadequate in some case, C++ extensions to CAGE can be made. Similarly to STEAMiE, the CAGE C++ code is written in modular, object-oriented way and is easily extendable. The easy creation of educative games is possible with CAGE, but it is not limited to them. While the aim of STEAMiE is to produce educative content with little or no training, CAGE aims mainly to educate programmers and give them experience. However, if the development process is wanted to be carried out with no prior experience or training, the CAGE scripting interface can be reduced to few powerful, basic commands for this purpose. In other words, with a very small subset of the CAGE scripting interface commands one can create impressive worlds and games. STEAMiE offers a networking support, which is lacking from CAGE for now, but on the other hand CAGE offers the AI support.

SAI-BOTS Scripted artificial intelligent basic online tank simulator [2] resembles CAGE in using C++ as the language of SAI-BOTS itself and Lua as a script language. SAI-BOTS allows the players to script the properties and the behavior of their tanks with Lua using special tank application programming interface calls and to fight against tanks controlled by each other or a computer. However, SAI-BOTS is a specific (although scriptable) game that cannot be used to create truly different computer games like CAGE. The scripting interface of CAGE is meant to be generic so that it allows the development of different game types.

6. CONCLUSIONS

Programming assignments of game programming courses can be laborious and cause the students to leave even an interesting course. In this paper, we presented an approach to ease the student workload using a scriptable artificial intelligent game engine. In order to evaluate our approach, we implemented the CAGE game engine and used it in a game programming course of the Tampere University of Technology. The results from the course were promising as we achieved a tolerable drop-out percentage, valuable feedback and good quality of the student game projects.

As the future work we will continue the development of CAGE, especially to offer a more sophisticated artificial intelligence framework and to further extend our Lua interface. We will continue using CAGE as the platform for programming assignments of our game programming course for the upcoming years as well. We plan to release CAGE as an open source project in the future in order to allow other universities to use it in their game programming courses also.

7. REFERENCES

[1] Alice web site, http://www.alice.org, retrieved on 28.11.2008

[2] Brandstetter, W., Dye, M., Phillips, J., Porterfield, J., Harris Jr, F., Westphal, B., SAI-BOTS: Scripted Artificial Intelligent Basic Online Tank Simulator. In Proceedings of Software Engineering Research and Practice'2005. pp. 793-799

[3] Buckland, M. 2005. Programming Game AI by Example. Plano: Wordware Publishing, Inc.

[4] Blender 3D content creation suite, http://www.blender.org/, retrieved on 26.11.2008

[5] Blender exporter plugin, http://b2cs.delcorp.org/index.php/Main_Page, retrieved on 26.11.2008

[6] Boost C++ library, http://www.boost.org/, retrieved on 26.11.2008

[7] Crystal Space 3D engine, http://www.crystalspace3d.org/main/Main_Page, retrieved on 26.11.2008

[8] Lua scripting language, http://www.lua.org/, retrieved on 8.4.2009

[9] Luabind wrapper library, http://www.rasterbar.com/products/luabind.html, retrieved on 26.11.2008

[10] MakeHuman 3d tool for 3d models character modelling and animation, http://www.makehuman.org/blog/index.php, retrieved on 26.11.2008

[11] Masuch, M., Nacke, L. Power and peril of teaching game programming - Mehdi, Quasim (Ed.), Gough, Norman (Ed.): Computer games: artificial intelligence, design and education (5th Game-on international conference Reading, UK, 8-10 November 2004). CGAIDE. Wolverhampton: Univ., 2004, pp. 347-351

[12] Millington, Ian. Artificial Intelligence for Games. 2006. Morgan Kaufmann Publishers

[13] Nykl, S., Mourning, C., Leitch, M., Chelberg, D., Franklin, T., and Liu, C., An Overview of the STEAMiE Educational Game Engine, In Proceedings of 38th ASEE/IEEE Frontiers of Education, New York, October 2008.

[14] Parberry, I., Nunn, J. R., Scheinberg, J., Carson, E., and Cole, J., SAGE: A Simple Academic Game Engine, In Proceedings of the Second Annual Microsoft Academic Days on Game Development in Computer Science Education, pp. 90-94, February 2007.

[15] Parberry, I., Kazemzadeh, M. B., and Roden T., The Art and Science of Game Programming, In Proceedings of the 2006 ACM Technical Symposium on Computer Science Education, Houston, TX, pp. 510-514, Mar. 1-5, 2006.

[16] Warhammer Online, http://www.warhammeronline.com/, retrieved on 8.4.2009

[17] The Wicher, http://www.thewitcher.com/intro.asp, retrieved on 8.4.2009

[18] World of Warcraft, http://www.wow-europe.com/en/index.xml, retrieved on 8.4.2009

Publication II

J-M. Vanhatupa. Guidelines for Personalizing the Player Experience in Computer Role-Playing Games. In *Proceedings of 6th International Conference on the Foundations of Digital Games (FDG'11)*, pages 46–52. ACM, June/July 2011.

Guidelines for Personalizing the Player Experience in Computer Role-Playing Games

Juha-Matti Vanhatupa
Department of Software Systems
Tampere University of Technology
Tampere, Finland
juha.vanhatupa@tut.fi

ABSTRACT
Computer role-playing games (CRPGs) are a genre of games that aims at providing similar gaming experience as paper and pen role-playing games. Personalized player experience is one main factor when capturing and maintaining interest of the player. However the player experience of modern CRPGs is usually poorly personalized. In particular, CRPGs use static storyline structures, in which player decisions have no effect and game world characters treat everyone exactly in the same way regardless of race, class and appearance. We claim that the player experience personalization present in CRPGs can be improved. In this paper, we present eight guidelines for personalizing the player experience in CRPGs. The guidelines assist game developers to personalize the player experience. We also discuss current tool support for personalization.

Categories and Subject Descriptors
K.8.0 [**Personal Computing**]: General – *games*

General Terms
Design, Human Factors, Theory

Keywords
Computer role-playing games, Personalization

1. INTRODUCTION
Computer role-playing games (CRPGs) are a genre of games that aims at providing similar gaming experience as paper and pen role-playing games (PnP RPGs). Those let the player live life of someone else and give the player a living virtual world with habitants to explore. Important elements of a CRPG include a storyline, a virtual world inhabited by characters, and the fact that results of character actions are resolved using game mechanics (rules).

FDG'11, June 29-July 1, 2011, Bordeaux, France.

One of the main reasons why people play role-playing games is to have a personalized character, and personalized player experience is a key factor in making players feel involved in a virtual world [21]. Furthermore personalized playing experience is one of the most difficult properties of PnP RPG for CRPGs to emulate. In PnP RPG a human game master controlling the game knows player characters in detail and can apply rules as needed to create personalized playing experience, whereas a computer requires explicit instructions.

Since personalizing the player experience is difficult in CRPGs, most state-of-the-art CRPGs use a hard-coded storyline. A hard-coded storyline consists of dozens of small quests, which the player must solve to complete the game. Although there are also quests that need not be solved to advance in the game, linear structure of mandatory quests (the critical path) creates feeling that the player is travelling in a pipe and his actions do not matter at all. Similarly, even in state-of-the-art CRPGs the player character appearance, attributes, race, or faction, do not usually affect the behavior of the non-player characters (NPCs), although in reality NPCs behavior should be totally different when they encounter different kind of characters.

In this paper we present eight ways to personalize the player experience in CRPGs. We claim that the game storyline could adapt to the player decisions, so that the game storyline has several paths from the beginning to the end instead of one critical path. The player decisions can open new quests and remove old ones. We also claim that it is possible to create a game world where NPCs modify their behavior based on the player character attributes. Furthermore, the game difficulty is scaled to the development of the player character. The claims are presented as generic guidelines for game developers.

The paper is organized as follows. In Section 2 we discuss the background of the CRPG genre and present examples of state-of-the-art CRPGs. In Section 3 we discuss personalizing the game storyline. In Section 4 we present how the player character attributes and history could affect the game. In Section 5 we present guidelines for game difficulty scaling. In Section 6 we go through current tool support for personalization. In Section 7 we present related work, before listing some concluding remarks in Section 8.

2. BACKGROUND & STATE-OF-THE-ART

First PnP RPGs were published during 1970s. Dungeons & Dragons (D&D) [8] was the first commercially available role-playing game. Other early releases were Chivalry & Sorcery and science fiction role-playing game Traveller. PnP RPGs were followed by first CRPGs at late 70s. Most of those were direct adaptations of D&D, like Akalabeth: World of Doom, one of the first CRPGs [2].

Now, almost forty years later, modern CRPGs are enormous game projects, and they consist of huge virtual worlds for the player to explore. CRPGs also have their own personality systems. In the following, we give examples of merits and flaws of state-of-the-art CRPGs.

The Elder Scrolls: Oblivion [17] uses a level-scaling system to maintain interest of the players. Every time the player advances a level all the enemies also became stronger, and if the player trains mostly non-combat skills, the game can easily became too difficult. In addition, when played using a high-level character the scaling system also makes encountered animals, for example boars and bears stronger than demons. However, there are modifications (Mods) for Oblivion, which can be used to correct the auto-leveling of enemies.

World of Warcraft [23] is the most popular CRPG at the moment. It had more than 12 million subscribers in October 2010 [3]. However, as is typical for a massive multiplayer online CRPG, majority of quests and experience received are for activities that revolve around killing [16]. In addition, there were near 8000 quests in the game (as of December 2008), however only 393 are class-specific and only 59 available for a particular class [16, 24]. Very little amount of the game content depends on the player character. However, more content becomes available as the player character gains levels.

In The Witcher [18], player's decisions have consequences, which appear only after 10–20 hours of gameplay. This eliminates the possibility to use load–save tactics: the player plays certain episode several times and loads the game and tries again until the desired results are obtained. Since consequences appear only after ten or more hours of playing, loading the game is no longer a decent solution. This results in the feeling of the game world responding to the player actions. In the dark world of The Witcher there are no clear good or bad decisions, but all of them have some (usually negative) consequences.

Dragon Age: Origins [6] offers six different tutorial chapters for characters with different backgrounds. For example, a dwarf noble starts the game in one of the dwarven cities, and a casteless dwarf commoner starts the game on the streets of the city. Character backgrounds are also used in the game, since the player character can meet NPCs from the tutorial chapter later in the game. The game also has multiple endings based on the player decisions. Different tutorials and endings of Dragon Age: Origins are a step towards right direction in storyline adaption. However as presented in the following section, storyline adaption can be improved even further.

3. PERSONALIZING GAME STORYLINE

A good game storyline is an essential component of CRPG. Good graphics or an excellent user interface cannot save CRPG with a pitiful storyline, because the player loses interest quickly if the game consists only of wandering in some virtual world or the storyline is unconvincing.

Since an average CRPG takes 100–200 hours to complete, the games are divided into smaller sub-tasks, quests. Quests are used in almost all CRPGs. One definition for a quest presented in literature is the following: A quest is a journey across symbolic, fantastic landscape in which a protagonist or player collects objects and talks to characters to overcome challenges and achieve a meaningful goal [11]. Quests should be used to maintain the player interest and create a feeling that the player is advancing in the game, and those can be used to support the game storyline [10]. Solving a quest can offer additional information about the storyline and the player can see how the events of the storyline are affecting NPCs of the game world.

The regular structure of a modern CRPG is presented in Figure 1. The game has a core plot and multiple tutorials and endings. The core plot of the game, the critical path, represents the path of quests that the player must complete to complete the game. The game has numerous quests, some of them belonging to the critical path and the rest being independent of it.

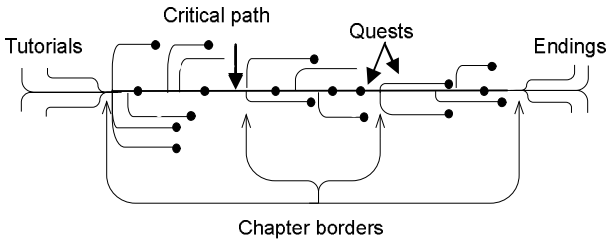

Figure 1. CRPG storyline structure containing the critical path.

3.1 Storyline Adaption

A game storyline can adapt to the player actions. This adds value to the game, because the player has to consider his actions and he can feel that he lives in a responsive world. The first guideline we give for a game developer is:

1. *Make the game storyline adapt to the player actions.*

Certain quests and endings can be made available or left out depending on the player choices. For example, if the player has acted like a chaotic character, robbed and mugged people, he is unlikely to be asked to rescue church leaders from an ambush; a replacing quest could be escaping from the local police forces. A game with multiple routes has several paths instead of a single critical path. A path is part of the storyline, a possible route of the player character from chapter beginning to the chapter ending. The structure of a CRPG storyline with multiple paths is presented in Figure 2.

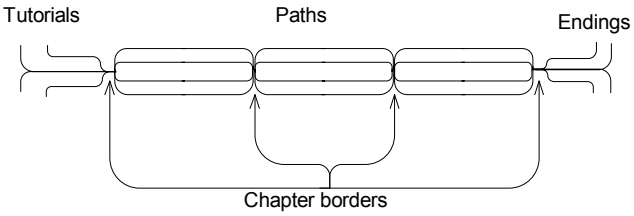

Figure 2. CRPG storyline structure containing several paths.

In the game structure presented in Figure 3, the player has made a choice that has consequences to the storyline of the game. He can no longer choose certain paths in chapters two and three – quests the paths contain are no longer available – and he cannot win the game anymore through one ending, which has been removed. However, his choice has also made available one new path to chapter two (added new quests to the chapter).

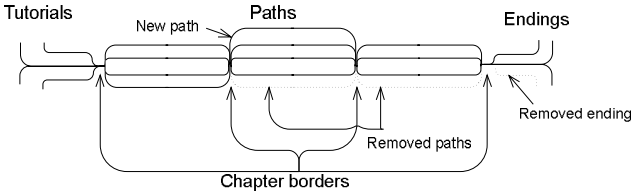

Figure 3. CRPG storyline after the player's decision.

When the game storyline is adapted to the player's decisions, he can feel that decisions have consequences and the player character is living in a responsive world. In addition, a variation in the game storyline adds replayability of CRPG, because in every time there are different quests to solve.

Modern CRPGs are complex games, and tutorial chapters are essential. The player needs time to learn at least basic commands before the real action begins. The second guideline concerns tutorials:

2. *Use a personalized tutorial chapter to teach the player the basics of the game.*

The tutorial chapter is the place, where the player needs to understand how the character is controlled, how its powers, possible spells, and other abilities are unleashed. Not offering tutorial chapter of any kind can lead to player frustration and possibly losing the player in a few minutes. Tutorial chapters should concentrate on things special to the character class. For example in a science fiction CRPG, the tutorial for assassins should be different than the one that marines will face. The assassin tutorial would concentrate on teaching the player using special skills like sniper skill and the marine tutorial would be a training camp episode teaching the player effective use of different weapons and grenades.

3.2 Quest Generation

Quests not belonging to the critical path can be generated. Generated quests should be appropriate for the player character. A guideline for personalized quest generation can be presented:

3. *Generate quests based on quest giver motive and the player character attributes.*

The generation process can be used, when the game is developed to rapidly increase the number of hard-coded quests, and when the quest is needed on the game it is selected from the pool of the pre-generated quests. This process is shown in Figure 4.

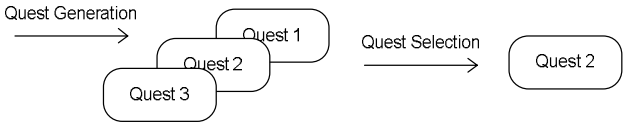

Figure 4. All game quests generated at development time and the needed quest is selected at run-time.

Quests can also be generated at run-time to ensure the game is never out of quests. Using this approach, the quest generation process is run every time a new quest is needed in the game. In this approach, the quest selection phase is trivial since the generation is already based on the character attributes and the quest pool contains only one quest for selection. This approach is shown in Figure 5.

Figure 5. The needed quest generated at run-time

Nevertheless, quest selection (or generation when using the run-time approach) for the player character should be personalized by based on the character attributes and the quest giver NPC's attributes (motive and situation in the game). The generation process is more complicated, if the player is controlling NPC companions in addition to the player character, and it is proper to use NPC companions' attributes also. An example priority list of features that could be used in quest generation in a fantasy CRPG is presented in the following:

1. quest giver motive and situation,
2. class of the player character,
3. race of the player character
4. major skills of the player character, and
5. NPC companions attributes.

In the example, attributes of NPC companions are presented at the bottom of the list with lowest priority, but it is easy to imagine CRPGs where those attributes would be more important.

Complexity of generated quests affects a lot to the generation process. It is easy to generate only combat-based quests that can be solved by killing certain amount of enemies. More complex generation process is needed to produce quests relating to several NPCs, which require, for example transporting items from one NPC to another or negotiations with them. Even more complex quests relating to several NPCs and with several tasks inside them, are at the moment, in the limits of current quest generation technology.

Quest generation could be applied to quests belonging to the critical path, too. However, in this case there should be strict pre-conditions and post-conditions, which the generated quest must fill. The generation mechanism also becomes more difficult to implement, but it could also offer even more variation to CRPG. In the ultimate case there would be no same quests in two playtimes of CRPG.

4. PERSONALIZING CHARACTERS

The player character is a virtual avatar of the player in CRPG. In CRPG the player has usually options to customize his player character in many ways. Those can be rule based mechanics like selecting race or class for character and dividing points between attributes and skills. Customization can also change the visual representation of the avatar in some manner. However, the fun gained from the character customization is lost, if it does not have any effect in the game. It is common for state-of-the-art games to have the same quests for every character. Moreover NPCs commonly treat the player character in the same way regardless of his appearance.

4.1 Non-player Characters

Believable and rational behavior of NPCs is important for keeping the game interesting. Modern artificial intelligence systems used for example in The Elder Scrolls Oblivion and The Witcher allow NPCs to have daily rhythm; they sleep at nights, work at days and also have areas where they spend their evening hours imitating the real humans. Usually artificial intelligence systems do not help at CRPG personalization and NPCs generally treat the player character in the same way regardless of his race, gender, class, appearing or equipment. To correct this problem we give a general guideline for NPC design:

4. *Design NPCs to treat the player character based on his visual attributes, reputation and NPC knowledge of the player character.*

We suggest the following technical solution. When implementing main NPCs of the game, they are given values that define how they regard individuals of each race, class, gender. There could also be bonuses or disadvantage to the player character from carrying specific equipment. When the NPC meets the player character, the value is calculated and it is used to determine reactions of the NPC. For instance, if the value gained is high the NPC is friendly, and with low value NPC might ask the player character to leave, or even attack without a warning. The value should be recalculated each time the player character meets the NPC, since equipment or faction of the player character can change often.

Although a base value for a personalized NPC reaction is gained using this method, other issues like the reputation of the player character and their history in the game must also be taken into account. For example, if the player character has solved a quest provided by the NPC, it is presupposed that NPC treats the player lot better than a completely stranger.

Another issue relating to the NPCs is that even in modern CPRGs, NPCs tend to forget that they have already said something, forcing the player through same conversations again. This problem hinders the creation and evolution of believable NPCs. A solution for this problem requires a combination of good programming, storing the knowledge about which dialogs have already been used, and a credible evolution of characters.

Similarly as quests, NPCs for CRPGs could be generated too. Although quest generation usually includes generation of some NPCs like enemies used in the quest, generation of NPCs with complex behavior is still very difficult to implement. In addition, it is even more difficult to make the generated NPCs to relate the game plot. There are some implementations for generating NPC behavior. For example, the ScriptEase tool [4] can be used to generate ambient behavior of NPCs.

4.2 Character Background

The character background is an important part of the game in PnP RPGs, where the player can usually write his own character background together with the game master. The character background can work as a motivator for the player character for solving certain quests. It can thus have a major effect on the game storyline, depending on the game master of course.

Generally CRPGs lack the use of character backgrounds. Even if the element exists, it is usually presented only as a text to the player and it has no further effect to the game. In many CRPGs the player character has just lost his memory at the beginning of the game and knows nothing about the world he has lived for his whole life. We give the following guideline to emphasis the player character background in CRPGs:

5. *Make the player character background affect the game.*

An example method for CRPG to benefit from the character background could be the following. Although the background is presented as a text in the game, it includes hints about the game storyline. Hints can be generally about the game or specific information concerning certain quests. Examples of hidden hints for character background are presented in Table 1.

Table 1. Examples of hidden hints in character background.

Background text:	Play effect:
"Your father told stories about strange lights in the mountains…"	There is a specific monster lair (quest) in the mountains.
"When you grow up, you learned (shopkeeper NPC in game) was not always honest."	The player knows certain shopkeeper NPC has been dishonest for a long time and can use this information.

In the same manner the character background could include misleading information. The player character might have misunderstood something or things would have changed. This would prevent the player rely blindly on his background.

5. PERSONALIZING GAME DIFFICULTY

When the player advances in CRPG the game should became more difficult. Varying order of quests inside chapters prevents entirely hard-coding difficulty of enemies encountered in quests. Although some base value is known – for example, what kinds of enemies are normally used in the quest – the game needs also dynamic balancing of quests and random encounters.

5.1 Game Balancing

Character development is a main fundamental issue of CRPGs. However, character development includes a potential problem. When the game advances, the player character becomes usually more powerful as he gains experience. Normal ways for gaining experience are overcoming enemies, using skills, or solving quests. To keep the game interesting the difficulty level should

be increased as the character becomes more powerful. Normal ways of dynamically increasing the difficulty level are the following:

- strengthening the enemies,
- changing the type of the enemies to more powerful,
- increasing the number of enemies.

The first way, used for example in The Elder Scrolls: Oblivion [17], can cause frustration, if every enemy is strengthened when the player character gains more power, what is the benefit of gaining levels in the first place? If the player is playing a non-combat oriented character, the game combats should not become more difficult as he gains levels.

Changing the enemy type is a modification of the above strategy. For example instead of goblins, the player could encounter more powerful humanoids like hobgoblins, bugbears or orcs. Even the effects are the same as in the above solution, but cause less frustration, since the player can see effects of gaining levels. However, the player should sometimes encounter weaker creatures, since it would be strange if all of those suddenly vanished from the world.

Increasing the number of enemies is an easy way to make the game more difficult. However, when used badly this method can lead to comical situations: for example when the player has a high-level character, and there suddenly are twenty goblins crowed into very small room, originally designed for only one enemy.

No matter which way is used to make the encountered enemies more difficult to defeat, the player character level should not be the direct indicator for enemy strength. We give the following guideline to assist balancing of the enemies in CRPGs:

6. *Balance the game enemies based on the fighting abilities of the player character instead of the experience level.*

Instead of the player character's experience level, a more sophisticated indicator of his fighting proficiency should be used. For example in fantasy CRPG the value could be calculated from the player character's major fighting skill level, known spell strengths, hit points and armor.

Normally, strengthening enemies is done by increasing the number of hit points, armor or inflicted damage by the enemies. However, strengthening could also be done by improving artificial intelligence. Naturally, this way of strengthening enemies is lot more complicated to use than just increasing stats of the enemies. In CRPG opponents' intelligence points or corresponding factor should be also taken account when strengthening enemies using artificial intelligence to keep intelligence of different kind of enemies in a believable balance.

5.2 Game Mechanics

Usually CRPGs contain lots of different skills, weapons, items, and spells, which all have effects into the game. This creates a lot of combinations and lets the player try different tactics. This character optimization is a part of the fun of CRPGs. However, there is also a possibility that some combination is too powerful, and breaks the game balance. Game mechanics breaking combinations can be hard to spot since some abilities, weapons, and spells can be available only for a certain character class or characters of certain race, and noticing those requires careful testing with every character class and race combination. Game mechanics is an essential element of CRPG and to assist its design we give the following guideline:

7. *Test game mechanics carefully to find possible mechanic breakers.*

In fantasy CRPGs, spells are one possible mechanics breaker. For example in Dragon Age Origins, Mana Clash spell drains enemy spellcasters completely out of mana and deals damage to them based on the amount of mana they lost. The spell is strong enough to kill just about any enemy mage in the game instantly, except the most powerful bosses. The existence of Mana Clash makes enemy spellcasters very easy to kill. Abilities, skills or spells should not be so powerful that all enemies of a certain group can be won using the same tactic over and over again. However, those can grant bonuses or inflict damage to a certain enemy group, as this lets the player to try different tactics.

Games containing lots of character race and class combinations can be problematic. There should be some reason to pick with every one of those. For balancing race and class combinations we give a guideline:

8. *Balance game races and classes, each of those should have rewards.*

All combinations should not be equally good. For instance, if orcs are strong and durable, it is only realistic that most orcs are warrior type classes instead of spellcasting classes. However if a certain race is pitiful in everything, no-one will play with a character of that race, and there is no point of existence of that race in the game. Similarly, each class should be good at something; after all they are all experts in their profession. Traditionally there have been problems in balancing of classes in CRPGs; usually the benefits of playing non-combat oriented classes have been too small.

There is also other type of character class balancing in CRPGs, time-related balancing. In fantasy CRPGs, it is common that spell-casting classes are weak characters at the beginning and gain powerful spells only later at the game. Usually at the end of the game, those are the most powerful character classes. This kind of balancing can be problematic and create some part of the game too difficult or too easy for certain character classes. The player can easily lose interest to the game, when encountering those. Instead of time-related balancing, in a well designed game the character classes should be in a good balance during the whole storyline.

6. TOOL SUPPORT

Game engines are the most important tools in modern CRPG development. Game engines vary widely in the details of their architecture and implementation. Those are also typically somewhat genre specific [9]. For instance, game engines used in CRPGs usually offer scripting language or scripting interface, since scripting is widely used in the genre. However, tool support for personalization by game engine is usually limited to game world editor reducing the work load in world creation and an editor for NPC scripting.

Game master (GM) tool sets can be used in personalization. Those give one player a possibility to act as a traditional role-playing game master. The game master can create personalized content using the GM toolkit. For instance he can create quests that relate to the past of the player character. However GM toolkits have limitations. For example, They cannot be used for run-time content creation, and game masters have to rely on pre-created content [22]. Examples of games containing GM tool sets include Vampire the Masquerade: Redemption (Activision 2000) and Neverwinter Nights (BioWare 2002) [13]. For example, Aurora toolset [1] in Neverwinter Nights is a collection of software tools allowing players to create own adventures and share them as modules. It contains a visual tile-based terrain editor, a script editor, a conversation editor, and an object editor. Bioware no longer supports Aurora Toolset, and its longevity is in the hands of community of hobbyists.

7. RELATED WORK

Personalization of CRPGs has been researched through analysis. Tanenbaum and Bizzocchi [19] have investigated issues of character believability and intelligent personalization through a reading of The Elder Scrolls: Oblivion [17]. Opening sequence (tutorial chapter) of Oblivion simultaneously trains the player in functions of the game and allows them to customize their character through choices they take. They state that Oblivion both succeeds and fails at mapping player behavior to appreciate class assignments.

Lankoski & Björk [12] have analyzed believability of NPC shopkeeper Claudette Perrick in The Elder Scrolls: Oblivion [17] using gameplay design patterns. Authors claim that descriptions of humans require several qualities for people to experience them as believable: human body, self-awareness, intentional states, and self-impelled actions, expression of emotions, ability to use natural language and persistent traits. Our guideline (4) also aims also at character believability; possibly using a method like gameplay design patterns character believability could be further increased.

There are several researches analyzing quests in CRPGs [7, 16]. Some tools that can be used for quest generation have been implemented. For example, ScriptEase generates quest code from quest patterns [20]. Their study showed that quest patterns generated more reliable scripts than manual script writing. After a higher learning curve, the participants of the study preferred to use quest patterns over manual scripting. However, the personalization process in quest generation has not been in the scope of their study.

There is also research relating to the dynamic adaptation game difficulty. Spronck et al. [15] have presented a novel online learning technique called “dynamic scripting” that is able to automatically optimize game artificial intelligence during game-play. In dynamic scripting an adaptive rulebase is used for the generation of intelligent opponents on the fly. In their evaluation, adaptive players were pitted against a collective of manually designed tactics in a simulated CRPG in a module for the Neverwinter nights [13].

The difficulty of CRPGs to emulate the personalized player experience of PnP RPGs originates from difficulty of computer to emulate work of a human GMs used in PnP RPGs. Peinado & Gervás [14] have made a theoretical study of PnP RPGs. Based on the study, they have transferred GM’s player modeling rules, their rules for interaction management, and their improvising algorithms from the real world to a new Interactive Storytelling system. However, there is a rather long way from the present interactive storytelling technology to achieve the same level of improvisation than human GMs of PnP RPGs possess. Delmas et al. [5] present another study including analysis of GM responsibilities. They present a study of tabletop role-playing games and narrative management. From this study they have derived a set of components and data structures to control the interactive storytelling in videogames. They also present an example prototype of social game to validate their architecture.

8. CONCLUSIONS

The personalized player experience of PnP RPGs is one of the most difficult properties for CRPGs to emulate. Computers need explicit instructions to create similar personalized playing experience that human GMs in PnP RPGs create. However, the personalized player experience of CRPGs can be improved with careful game design.

In this paper we presented guidelines for personalizing the player experience in CRPGs. The presented guidelines covered personalization of the game storyline, characters and game mechanics. We believe that use of these guidelines would improve personalized player experience offered in CRPGs. However, an extensive evaluation of the guidelines would need CRPGs designed using those; therefore an extensive evaluation is difficult and time-consuming. To some extent evaluate the work done and as future work, we will continue process of implementing software based on the presented guidelines and thus technically confirm their role in development.

9. REFERENCES

[1] Aurora Neverwinter Toolset, http://nwn.bioware.com/builders/

[2] Barton, M. *Dungeons & Desktops, The History of Computer Role Playing Games*. 2008. A. K. Peters.

[3] Blizzard Entertainment Press Release, http://eu.blizzard.com/en-gb/company/press/pressreleases.html?101007

[4] Cutumisu, M., Szafron, D., Schaeffer, J., McNaughton, M., Roy, T., Onuczko, C. and Carbonaro, M. 2006. Generating Ambient Behaviors in Computer Role-Playing Games. *IEEE Journal of Intelligent Systems* (IEEE IS). Vol 21, No 5, 19-27.

[5] Delmas, G., Champagnat, R., and Augeraud, M. 2009. From Tabletop RPG to Interactive Storytelling: Definition of a Story Manager for Videogames. In *Proceedings of ICIDS*. 2009, 121-126.

[6] Dragon Age, http://dragonage.bioware.com/

[7] Doran J. and Parberry, I. 2010. Towards Procedural Quest Generation: A Structural Analysis of RPG Quests. Technical Report LARC-2010-02, Laboratory for Recreational Computing, Dept. of Computer Science & Engineering, University of North Texas.

[8] Dungeons & Dragons official site, http://www.wizards.com/dnd/

[9] Gregory, J. 2009. *Game Engine Architecture.* A. K. Peters.

[10] Hallford, N. and Hallford, J. 2001. *Swords & Circuitry: A Designer's Guide to Computer Role-Playing Games*. Prima Publishing.

[11] Howard, J. 2008. *Quests: Design, Theory, and History in Games and Narratives.* AK Peters Ltd., Massachusetts.

[12] Lankoski, P. and Björk, S. 2007. Gameplay Design Patterns for Believable Non-Player Characters. In *Situated Play, Proceedings of Digital Games Research Association 2007* (Tokyo, Japan, September 24 – 28, 2007) DiGRA 2007, 416-423.

[13] Neverwinter Nights. BioWare., http://nwn.bioware.com/

[14] Peinado, F., Gervás, P. 2004. Transferring Game Mastering Laws to Interactive Digital Storytelling. In *Proceedings of International Conference on Technologies for Interactive Digital Storytelling and Entertainment.* (Darmstadt, Germany, June 24 – 26, 2004) TIDSE 2004, 48-54.

[15] Spronck, P., Sprinkhuizen-Kuyper, I. and Postma E. 2004. On-line Adaptation of Game Opponent AI with Dynamic Scripting. *International Journal of Intelligent Games & Simulation.* 3, 1 (March 2004), 45-53.

[16] Sullivan A. 2009. A Gender-Inclusive Quest Design in Massive Multiplayer Online Role-Playing Games. In *Proceedings of forth International Conference on Foundations of Digital Games*. (Port Canaveral, FL, USA, April 26 – 30, 2009) FDG 2009, 354-356.

[17] The Elder Scrolls IV: Oblivion. ZeniMax Media Inc., http://www.elderscrolls.com/

[18] The Witcher, http://www.thewitcher.com/

[19] Tanenbaum, J. and Bizzocchi, J. 2009. Close Reading Oblivion: Character Believability and Intelligent Personalization in Games. *Loading…The Journal of the Canadian Games Studies Association*. 3, 4 (2009).

[20] Trenton, M., Szafron, D., Friesen, J. and Onuczko, C. 2010. Quest Patterns for Story-based Computer Games. In *Proceedings of the Sixth Artificial Intelligence and Interactive Digital Entertainment Conference* (Palo Alto, California, USA, October 11 – 13, 2010) AIIDE 2010.

[21] Tychsen, A., Tosca, S. and Brolund, T. 2006. Personalizing the Player Experience in MMORPGs. In *Proceedings of the 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment* (Darmstadt, Germany, December 04 – 06, 2006) TIDSE'06. Springer LNCS.

[22] Tychsen, A., Hitchens, M. and Brolund, T. 2006. The Game Master. In *Proceedings of the 2nd Australian Conference on Interactive Entertainment* (Sydney, Australia, November 23 – 25, 2005) IE2005. 215-222.

[23] World of Warcraft, The Official EU website, http://eu.battle.net/wow/en/

[24] Wowhead.com. World of Warcraft database and tools. http://www.wowhead.com/

Publication III

J-M. Vanhatupa. Generative Approach for Extending Computer Role-Playing Games at Run-Time. In *Proceedings of 11th Symposium on Programming Languages and Software Tools and 7th Nordic Workshop on Model Driven Software Engineering (SPLST'09)*, pages 177–188. Department of Software Systems, Tampere University of Technology, August 2009.

Generative Approach for Extending Computer Role-Playing Games at Run-Time

Juha-Matti Vanhatupa

Department of Software Systems, Tampere University of Technology,
P.O.BOX 553 FI-33101 Tampere, Finland
juha.vanhatupa@tut.fi

Abstract. In modern-day computer role-playing games, virtual worlds contain numerous non-player characters and objects like weapons, armors and other items. The sheer number of items makes programming credible virtual worlds a time consuming task for application developers. This task can be eased using code generation tools to produce parts of game world automatically. In this paper, we introduce a generator to produce parts of game world for computer role-playing games. The application developer specifies a model and game world parts are generated from this model. The produced game world parts are represented as script code, and therefore it can be generated at run-time. Scripts can contain game weapons, places, non-player characters, almost anything that can be represented as a combination of numerical data. We demonstrate the use of our generator by generating scripts containing weapons, items and enemies for our western computer role playing game scenario.

Keywords: Code generation, run-time generation, computer role-playing games

1 Introduction

Computer role-playing games (CRPGs) are a genre of computer games that aims at providing an experience not unlike the original pen and paper role playing games (RPG). CRPGs offer a fictional game world for players to explore. During the evolution of CRPGs, virtual worlds in thems have grown in size. The virtual world of a modern computer role-playing game consists thousands of non-player characters (NPCs) and numerous objects (weapons, armors, items. etc.).

The world and its habitants have to be credible in order to maintain the interest of players. A player must feel his or her character is a part of this fictional world, a world around the character revolves as it is supposed to, and there are a lot places to explore, NPCs to encounter and objects to use. This implies that a lot of game development time has to be invested in programming the game world. Luckily, some parts of the world can be generated automatically using tools.

Computer games in general contain a large number of objects that can be paramerized and a scripting language can be used in initialization of those objects [3]. Using scripts can also save time and increase productivity as modifications and

customizations to the script code can be done without recompilation. The use of scripting languages in computer games has increased, especially Lua scripting language [11]. Lua is a powerful, general purpose, dynamically typed scripting language and it has excellent performance [9]. It was developed in the Pontifical Catholic University of Rio de Janeiro in Brazil, and the first version was created in 1993. Lua is nowadays used in many popular computer games, including for example The Witcher [19], Warhammer Online: Age of Reckoning [18], and World of Warcraft [20].

Computer games' large number of objects can lead to the explosion of the number of script files, and even if writing scripts can be easier than traditional programming, this workload is enormous. This can be eased with a generative approach. The idea of generating script code for computer role playing game has been proposed, for example, in [10, 15]. Such scripts are often used by artificial intelligence (AI) of the game in conversation dialogs and for other actions, and cannot be used to generate parts of the game world, however.

This paper presents an approach that can be used to generate a part of the game world. This part can contain anything that can be represented as a combination of numerical data – weapons, places, NPCs to have some examples. Different sections of the virtual world can be specified in the configuration files of the generator program. These configurations are given in XML format and the workload for writing these configuration files is a lot less than scripting all objects by hand. XML tools [13, 16, 21] can be used to assist the XML configuration files creation, but the format of those XML files is simple enough to be created without tool support also. XML configurations contain limits for attribute values and numbers of objects that will be created (*object templates*). We call an object created in this way a *generated object.*

The generated object classes are implemented in a higher level programming language or in a script language depending on situation. The script files generated in the approach, instantiate, configure and place generated objects into the game world. The context of the generation defines what type of classes exists to be used in the model. Although the generated code does not create new object classes, the attribute values and behavior of generated objects can vary freely. Proper XML tool can assist the user in this creation process, for example, by preventing the user from creating illegal XML object templates into the XML files.

The XML configuration is a model of the system and the script files are generated from this model. The script file generation can be done at run-time, as the player character advances inside the virtual world; the generator program generates new script files, which contain generated objects. Those generated objects are instantly placed into the virtual world. Run-time generation is space efficient and flexible. Run-time generation can also reduce loading times of computer game levels, as objects and NPCs in areas are generated only when the player enters the specific area instead of loading all in one long initialization operation. The second possibility is to do the generation beforehand, this way is most useful when testing the scenario, as this method allows the application developer to browse generated script files.

Use of a generator program to generate parts of the game world leads to greater variation in the game as attribute values of weapons, places and non-player characters vary. The world generation reduces application development work input in virtual world creation, thus releasing resources from mechanical, systematic work.

We have evaluated the approach by extending our existing CAGE game engine [4] with generator program. The CAGE game engine uses Lua script files to initialize its virtual worlds, and it allows run-time addition of those scripts. Therefore, it was an excellent test platform for our script generation approach.

We proceed as follows. Section 2 presents our approach for script generation. In Section 3 discuss current implementation of the approach. In Section 4 we demonstrate the use of the generation approach. Finally, related work is discussed in Section 5 and concluding remarks are presented in Section 6.

2 Overview of the Approach

Our context for script generation was the CAGE game engine. We have developed the CAGE game engine to be a generic platform for different types of computer games. It is also used as platform in the game programming course of Tampere University of Technology. CAGE is implemented using the C++ programming language and the Lua scripting language. CAGE graphics use Crystal Space [5], a crossplatform software development kit for real-time 3D graphics. CAGE offers a wide function interface, which can be used from Lua scripts, e.g. to create creatures, items and set their behavior and properties. NPC behavior can be programmed using Lua scripts and CAGE supports state machines, which states and state transitions are written in the script files. It also enables modifying the camera behavior, the input handling and many other things. This wide Lua interface allows using of CAGE for many different game genres. In the game programming course the students implement their game projects by writing Lua scripts and C++ extensions to CAGE. The architecture of CAGE is shown in figure 1.

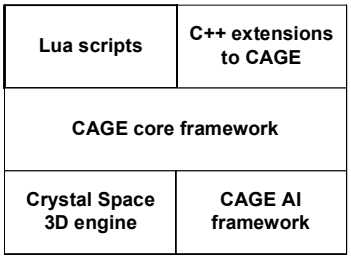

Fig. 1. Architecture of the CAGE

We wanted to create an approach to generate script files, which could be used to create objects into CAGE virtual worlds, and would be general to be used with other game engines as well. In order to demonstrate our approach, we have added a generator program into the CAGE game engine. The approach could be used can be used with other game engines than CAGE, but the C++ part of the generator is wrapper between the current implementation of CAGE and Lua code doing the actual parsing and generation. Therefore it should be rewritten if used in other environment.

Our generation approach begins as the application developer defines XML configuration files, the models of the system. The target application, a computer role-playing game defines the interface of generated objects (creatures, items, weapons, etc.), which can be written into the XML configuration. The declared interface can be actual metamodel (i.e. XML schema), which some XML tools can use, or it can be just a knowledge of the application developer. Depending on the used XML tool this information can be stored into a tool view, or the tool might prevent the user from creating illegal (non-existing) generated object templates. The script files are generated using the XML configuration. Figure 2 depicts the elements of our approach. We will discuss each of these elements in more detail in the following section.

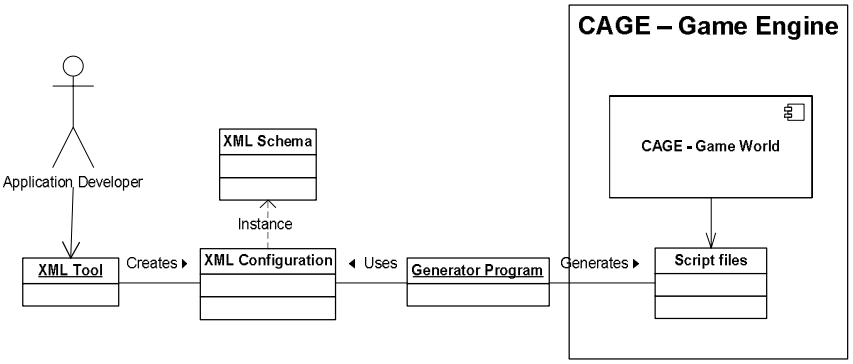

Fig. 2. Elements of our approach

The XML configuration contains data for generated objects. Because we generate script language code and it will not be compiled, the XML can contain elements whose value is of different types, including integers, strings and even function names. Those types have different purpose when generating attribute values for generated objects. Table 1 presents XML element value types, which can be used in the configuring the current implementation.

Table 1. Element types

Element type	Example	Explanation
Integer	`<count>5</count>`	Attribute value is always equal to specified integer value
Integer min	`<hpmin>2</hpmin>`	Value defines lower limit of the attribute value
Integer max	`<hpmax>5</hpmax>`	Value defines upper limit of the attribute value
String	`<name>bird</name>`	Attribute value is always same as the value
Function	`<pick_up function="true"> pick_up_rifle </pick_up>`	Value defines function name and attribute value is always the same

Functions can be used to implement normal behavior of the items or those can be used to create unexpected events, like traps or special attacks. For example, when the player character picks up an item, the game framework calls `pick_up` function. In case of ammo box, in the `pick_up` function body, the player characters ammunition is increased. When XML element specifies a function name the application developer should ensure the given function can be found from script files, otherwise the framework will call null function.

The XML schema defines structure of the XML document. It can be given to the XML tool and then used to create valid XML documents. For example, the VARMA tool reads the XML schema and uses it to ensure validity of user created XML configuration. The used XML schema is domain specific as its specification (the XML configuration) depends on the target domain.

The Generator Program parses the XML file and generates the scripts based on the XML file. Depending on the purpose of the XML element it is processed differently in the generator. Attributes of generated objects can be specified in the XML file either by giving a lower and upper limit for them or by defining single number, string or function name, which will be used as the attribute value. If limits for integer are given, the generator program generates a number between the limits at random.

In our implementation the generator output is collection of Lua script files. However, any script language could be used, only limitation is that it must be executable without compiling as the generated files are accessed at run-time. For each area specified in the XML configuration, the generator program creates a function, which is called when it is needed in the game (the player moves into the area). Code of generated objects of the certain area is placed inside the corresponding function. If

performance optimization is needed and the generation can be done beforehand, it is possible to compile generated Lua script files. Compiled Lua script files run faster, are smaller in size and therefore quicker to load [3].

3 Generator Program Implementation

The generator program of our implementation is also written in C++ and Lua. Although, the C++ part contains only one class, which calls different Lua functions. It works as a wrapper between CAGE and Lua code, which does parsing and generation. The generator also uses the Lua interface of CAGE to place generated objects into the game virtual world. The Architecture of the generator program implementation is represented in Figure 3.

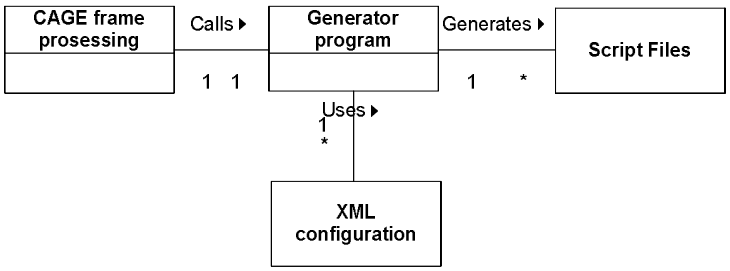

Fig. 3. Architecture of the generator

The game loop of the CAGE game engine calls Lua part of the generator. Called script code checks if the player has moved into new area. The area limits are gain from the XML files. If the player is in a new area the generator parses the rest of the XML file of the area and generates a new Lua function from it. Finally the generated Lua function is called and it creates NPCs, items and other generated objects into the game world.

The other possibility is to call manually the main function of the generator to create script files beforehand. In this way, it is possible to browse generated script files and estimate if those are appropriate and it also offers a possibility to compile the scripts. Similar to run-time generation, using this way the generated objects are inserted into the game world, at the moment the player enters the area.

The Lua XML parser of our implementation is made in such way that missing XML elements do not crash the program, if something is missing, the generator uses default values. Although, if the XML file is made using a tool, which checks the XML validity, there cannot be any missing XML elements.

4 Generation Example

We demonstrate the use of generator program for the CAGE western CRPG scenario by two examples; at first we generate area containing birds and a cottage. In other example we generate a wasteland area containing rocks, cactuses, a few weapons and bandits. In addition to these examples, we tested the efficiency of our implementation with separate test cases. In those we generate larger number of objects into same virtual world. We present efficiency test results after the examples.

A fraction of the XML file for the area is represented in listing 1. The represented XML file defines bird flock of two to ten birds. The z coordinate of the birds is defined always to be 20 to get the bird flock fly in a single line. Flight height of the birds (y coordinate) varies a little to create more realistic looking flock. Each creature type is defined using `creature` XML element and correspondingly item types can be declared using `item` XML element.

Listing 1. Fraction of the area XML file

```
<area1>
  <!-- area size, if object does not define other x and y limits, it is
  generated inside area  -->
  <areax>0</areax>
  <areaxe>300</areaxe>
  <areay>-50<areay>
  <areaye>300<areaye>

  <creature>
    <name>bird</name>
    <minimum>2</minimum>
    <maximum>10</maximum>
    <speed>5</speed>
    <hpmin>3</hpmin>
    <hpmax>5</hpmax>
    <xmin>230</xmin>
    <xmax>300</xmax>
    <ymin>-10</ymin>
    <ymax>-6</ymax>
    <z>20</z>
  </creature>
  ...
```

After the player character has crossed the area border, the generator program reads the XML and generates the script code based on it. The corresponding part of the generated script is shown in listing 2.

Listing 2. Fraction of the generated script

```
function area1()
  bird(280,-10,20,0,5,3)
  bird(291,-8,20,0,5,3)
  bird(284,-7,20,0,5,4)
  bird(265,-7,20,0,5,5)
  bird(232,-7,20,0,5,4)
  ...
  cottage(189,-26,90)
  ...
```

If the generation is done at run-time, the script code is executed instantly and generated objects are placed into the virtual world. Into this particular part of the area is placed a bird flock and a cottage. Those are shown in Figure 4.

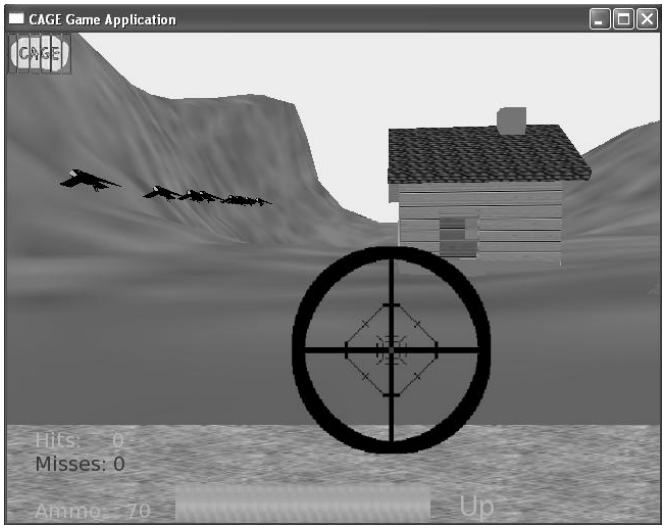

Fig. 4. The virtual world after the generated code is executed.

Depending on the implementation of the target CRPG, the lines generated into the script files can be C++ constructor calls or Lua function calls. In our implementation we generate Lua function calls. Those functions call C++ function to insert correct 3D object into the virtual world and C++ constructor to initialize the object.

In the second example, we created a wasteland, which contained rocks, cactuses, a few weapons and bandits. Listing 3 shows a fraction of the XML configuration, declaring rifles. Figure 5 shows a screenshot from the game prototype when the player has found one of the generated rifles.

Listing 3. Fraction of the XML file declaring rifles

```
<item>
  <name>rifle</name>
  <minimum>1</minimum>
  <maximum>3</maximum>
  <!-- ground is a special attribute, which puts the object into the
  ground -->
  <y>ground</y>
  <pick_up
  function="true">pick_up_rifle</pick_up>
  <weight>20</weight>
  <damagemin>30</damagemin>
  <damagemax>50</damagemax>
  <accuracymin>3</accuracymin>
```

```
    <accuracymax>5</accuracymax>
  </item>
```

The previous example defines from one to three rifles into the area. There are no coordinate definitions except the ground level, therefore the rifles are generated to the ground level, somewhere inside the area. The CAGE framework calls a pick up function of item, when the item is picked, in this particular case it is defined to `pick_up_rifle`.

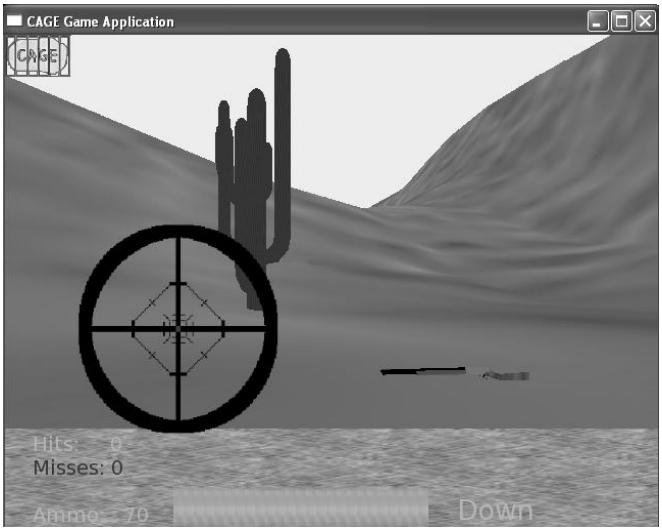

Fig. 5. The player has found a rifle from the wasteland.

In addition to earlier examples, we tested our implementation efficiency by generating an area of 30 birds, 10 cowboys and 20 items into our western CRPG example. It took 1.141s, to generate those objects; most of the time was spent on placing the objects into the virtual world. Script code generation step itself was almost instant, it took 0.016s.

Another efficiency test we made was to generate a flock of 1000 birds into same virtual world as test one. It took 1.922s to generate this massive flock, again most time spent on placing objects; 0.047s to generate script code and 1.822s to placing objects. The generation time 1.922s contains also one file initialization operation, which does not belong to either of those operations. Our test platform was T60 laptop Core Duo T2500 2,0Ghz processor, 1,00 GB of RAM, which is bit out of date of gaming PCs of today.

At the level of rule of thumb, the implementation is fast with reasonable number of generated objects. Moreover, as most of the time is spend on placing the objects, this time cannot be optimized in the generator implementation.

5 Related Work

Huge virtual worlds of modern-day computer role playing games containing thousands of objects and non-player player characters are time consuming programming task for application developers. A need for generative tools is obvious in this domain. Some tool support exists already. For instance, ScriptEase [14] is a model for AI scripting. The model is pattern template based, allowing designers to quickly build complex behaviors without doing explicit programming. The first version of ScriptEase generates scripting code for BioWare's Neverwinder nights [2]. ScriptEase tool has been extended to support sub-quest generation for CRPGs [12]. This generation can be done without doing any manual scripting. They have also defined generative design patterns [6] to help content designers to generate script code for game stories. Use ScriptEase or other generation tool of this kind, eases world creation process and helps designers and other non-programmers to join programming of virtual world. Another example of AI code generation is [8], where Kienze et al. present model-based approach; they use their tools to generate AI code for Electronic Arts Tank Wars game. In their approach the generated code is C++. However, these AI tools cannot help in game world generation and purpose of generative design patterns and our generator is different. While generative design patterns aim to help non-programmers to participate world creation our generator program aims to release programmers from doing boring, mechanical work.

Vu & Veloso [17] have constructed high-level behavior-centric programming language and an automatic code generation system. The language can be automatically translated by generator into C++ code. Once compiled, the C++ code yields an executable that directs the execution of behaviors in the agent's sense-plan-act cycle. However, in this approach the code needs to be compiled and the generated code is used for AI control.

Model-driven engineering technologies can be used in game programming as computer games are nowadays generally huge systems with a lot of different components. In [1] Barros et al. use model-driven game as platform for enchanting software project management education. They propose a high-level process and a model-based infrastructure for game development. Instead of using models only to describe game element behavior, they use models to describe the underlying story and its graphical representation. Their approach is appreciate for simulation-based games and do not aim to be general as our approach. In [7] Furtado & Santos propose approach, which encompasses visual domain-specific languages, semantic validators and code generators to make game developers and designers to work more productively, with a higher level of abstraction and closer to their application domain. Compared to their code generation step, our code generation generates script code instead of high-level language C# and therefore can be executed at run-time.

6 Conclusions

Developers of CRPGs can benefit from use of generative tools. The use of generative tools reduces need of application developer work input in the virtual world creation.

Work is still needed to bind specific objects and places together as quests and other specific events. However, this work relates to the storyline that the game designers have planned, it is more meaningful work for human than the mechanical virtual world creation, and it cannot be easily automated. In addition, several CRPGs contain so-called wastelands or dungeons that are not related to the plot of the game, and a generator can create those places from templates singlehandedly without developer interaction.

Run-time generation can reduce loading times of computer games, instead of one long initialization operation the objects are generated from a model only when those are needed. The virtual world can be divided in areas, which all have specific model containing object templates for the objects of the area

We have presented an approach, which allows object generation at run-time. Generated objects – items and NPCs are inserted into the game world at the moment the player enters a certain area. In order to evaluate our approach, we have extended our existing CAGE environment with a generator program. With the implementation we tested suitability and efficiency of our approach and gained promising, initial results. As future work we will continue development of the CAGE game engine and extend the variety of the generator program.

References

[1] Barros, M., Dantas, A., Veronese, G., Werner, C., Model-driven Game Development: experience and model enhancements in software project management education, Journal of Software Process: Improvement and Practice, Special Issue on Software Process Simulation Modeling, vol. 11(4), pp.411-421. (2006)

[2] BioWare'S Neverwinter Nights, http://nwn.bioware.com/

[3] Buckland, M., Programming Game AI by Example, Plano: Wordware Publishing, Inc, Plano, Texas (2005).

[4] CAGE game engine, http://www.students.tut.fi/~vanhatuj/cage.html

[5] Crystal Space 3D engine, http://www.crystalspace3d.org/main/Main_Page

[6] Cutumisu, M., Onuczko, C., McNaughton, M., Roy, T., Schaeffer, J., Schumacher, A., Siegel, J., Szafron, D.,Waugh, K., Carbonaro, M., Duff, H., and Gillis, S., Scriptease: A generative/adaptive programming paradigm for game scripting. Science of Computer Programming 2007 Vol. 67(1), pp. 32–58. (2007)

[7] Furtado, A. W. B., Santos, A. L. M. Using Domain-Specific Modeling towards Computer Games Development Industrialization, The 6th OOPSLA Workshop on Domain-Specific Modeling (DSM06), Portland, USA, October 2006, pp. 1-14

[8] Kienzle, J., Denault, A., Vangheluwe, H., Model-based Design of Computer-Controlled Game Character Behavior, MoDELS 2007, Nashville, USA, Oct 2007, pp. 650-665.

[9] Lua Scripting Language WWW site, http://www.lua.org/

[10] McNaughton, M., Schaeffer, J., Szafron, D., Parker D., and Redford J., Code Generation for AI Scripting in Computer Role-Playing Games, Challenges in Game AI Workshop at AAAI '04, San Jose, CA, USA, July 2004, pp. 129-133.

[11] Millington, I., Artificial Intelligence for Games. Morgan Kaufmann Publishers, San Francisco, 2006.

[12] Onuczko, C., Szafron, D., Schaeffer, J., Cutumisu, M., Siegel, J., Waugh, K., Schumacher, A., Automatic Story Generation for Computer Role-Playing Games, AIIDE 2006, California, USA, June 2006, pp. 147-148.

[13] oXygen XML editor WWW site, http://www.oxygenxml.com/

[14] ScriptEase WWW site, http://www.cs.ualberta.ca/~script/

[15] Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., and Postma, E., Adaptive game AI with dynamic scripting. Machine Learning, Vol. 63(3), pp. 217–248. (2006)

[16] Vanhatupa, J-M., Hammouda, I., Korhonen, M., Koskimies, K., Model-driven Approach for Interactive Configuration Description in Component-based Software Product-lines, NW-Mode'2007, Ronneby, Sweden, August 2007, pp. 127-142.

[17] Vu, T., Veloso, M., Behavior Programming Language and Automated Code Generation for Agent Behavior Control, AAMAS04, New York, USA, July 2004

[18] Warhammer Online: Age of Reckoning WWW site, http://www.warhammeronline.com/

[19] The Witcher WWW site, http://www.thewitcher.com/intro.asp

[20] World of Warcraft WWW site, http://www.wow-europe.com/en/index.xml

[21] XMLSpy WWW site, http://www.altova.com/products/xmlspy/xml_editor.html

Publication IV

J-M. Vanhatupa. Personalized Side-Quest Generation for Computer Role-Playing Games. In *Proceedings of 12th Symposium on Programming Languages and Software Tools (SPLST'11)*, pages 196–206. Institute of Cybernetics at Tallinn University of Technology, October 2011.

Personalized Side-Quest Generation for Computer Role-Playing Games

Juha-Matti Vanhatupa
Department of Software Systems
Tampere University of Technology
Tampere, Finland
juha.vanhatupa@tut.fi

Abstract

Modern computer role-playing games contain numerous quests, individual puzzles to be solved by the player character. Usually quests are predefined by the application developer, and a lot of application development time is spent in programming those. Predefined quests are however error-prone and difficult to maintain. In addition, predefined quests appear in the same form in every playtime and no matter what kind of player character the player is playing. In this paper, we present a generative approach to produce personalized side-quests for computer role-playing games. Using this approach, it is possible to replace the predefined quests and ease the work of the application developer, or simply use generated quests in addition to hard-coded ones, therefore further increasing the number quests in the game. The quest generation is based on attributes of the player character, which ensures those are suitable for the game. To evaluate our approach, we have implanted quest generator program and used it with an existing environment the CAGE game engine. We present example cases, where generated quests are used in a western computer role-playing game prototype.

1 Introduction

Computer role-playing games (CRPGs) usually contain a sophisticated story, which is divided into a huge number of quests. A game contains a main quest defining the game plot and numerous side quests. By accomplishing those side quests the player can learn more about the game world, gain better equipment or experience and meet new non-player characters (NPCs).

Generally, a quest is a complex structure containing a lot of modifying elements. There are research works analyzing quest structure [5, 6, 12]. Quests have become the most important mechanism when guiding the player through the game plot [6]. In our approach we define quest as a small story part with a problem that the player character can solve, and there is a reward for solving the problem.

Personalizing the playing experience is a key factor in making the players of computer games feel involved in the virtual world [13]. In CRPGs, the player character is a virtual avatar of the player and usually a collection of different number values are used to present how good the player character is in certain skills and many other purposes. For example, these numbers present how well the player character shoots with the rifle or how much damage he can sustain. Although quests are used almost in all CRPGs, those are personalized in any way only in extremely rare cases, and usually same quests are available to the player despite all of the player character's class, race, skills and other attributes.

Offering a huge number of credible quests is one of the main problems the modern CRPGs encounter. A common solution to this is to pre-define enormous number of quests. However, in numerous practical cases this is not a decent solution, because in some point the game runs out of predefined quests. In addition, creation of those side-quests is mechanical, repetitive work and automating this task releases resources to more meaningful work. Only a few games have any random or generated quests, and need for quest generators has been noticed [9].

In this paper we describe an approach for generating personalized side-quests at run-time. The generated quests are based on the attributes of the player character. In addition, attributes of quest giving

NPC are also used. Using this approach, it is possible to replace the predefined quests and ease the work of the application developer, or simply use generated quests in addition to hard-coded ones, therefore further increasing the number quests in the game, and to add personalized story element into the game. In our approach, the application developer gives the quest generator a configuration of character classes, races, skills and quest prototypes. Quest prototypes are defined by the application developer; The rest of the configuration is defined by the CRPG, which is used as the context of the quest generation. Although the quest prototypes need manual scripting at the development time, is the needed time lot less than scripting all the quests, as quests prototypes are instantiated several times.

To evaluate our approach, we have implemented a quest generator program, which generates quests at run-time based on the attributes of the player character. The quest generator and configuration sets for it are implemented using Lua scripting language [2]. We have used this approach with an existing environment known as the CAGE game engine [16] to present a case where generated quests are used in a prototype western CRPG.

This paper is based on quest generation ideas presented in previous paper [15]. However, the presented approach has been evolved and an implementation has been composed for this publication. Previously we have extended CAGE with a generator program to allow extending CRPGs at run-time [14]. Those extensions were made by generating items, NPCs and places at run-time. However the previous approach could not be used in quest generation, because these two problems are distinct. When using the previous approach, the application developer needed to manually bind generated specific objects and NPCs to create quests.

The rest of the paper is organized as following. In Section 2, we present our approach for personalized quest generation. In Section 3, we present a realization of the approach. In Section 4, we show the generation examples. In Section 5 we discuss the related work, before concluding remarks in Section 6.

2 Generative Approach

Generally, quest generation for CRPG can be performed at development time or at run-time. In first approach, the generation process is used when the game is developed to rapidly increase the number of quests in the game. When a quest is needed in the game it is selected from the pool of the pre-generated quests. When using the second approach, quests are generated when those are needed in game. Using this method, the generation process is run every time a quest is needed in the game. These two methods are shown in Figures 1 and 2.

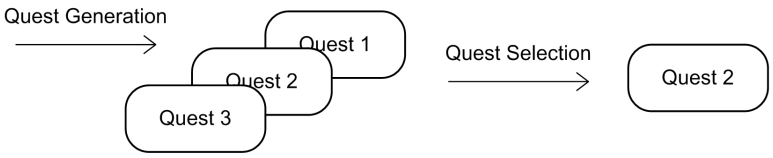

Figure 1: Quests generated at development time and selected at run-time

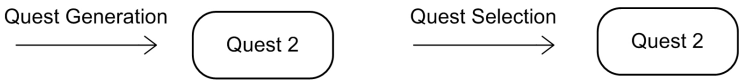

Figure 2: Quests generated and selected at run-time

In our approach, we use the later approach and generate quests at run-time. Since run-time generation offers better support for personalization. For example some quests are available only for certain character

classes, and when generation is based on the player character attributes, we avoid generating quests that are never used. If quests are generated at development time, the quest selection is based on character attributes, instead the quest generation. Therefore quest that are never used will be generated, because player character class and attributes do not affect the generation (only for selection). Run-time generation also ensures that the game never runs out of quests.

Complete diagram of the generation phase of our approach is shown in the Figure 3. Quest prototypes consists modification points, which are changing elements inside the quest prototype. In the later part of the generation, the modification points of the generated quest are filled based on the player character attributes. For example, quest difficulty can be altered based on the fighting skills of the player character. These modification points are defined to the quest prototypes by the application developer, as the quest prototypes are defined.

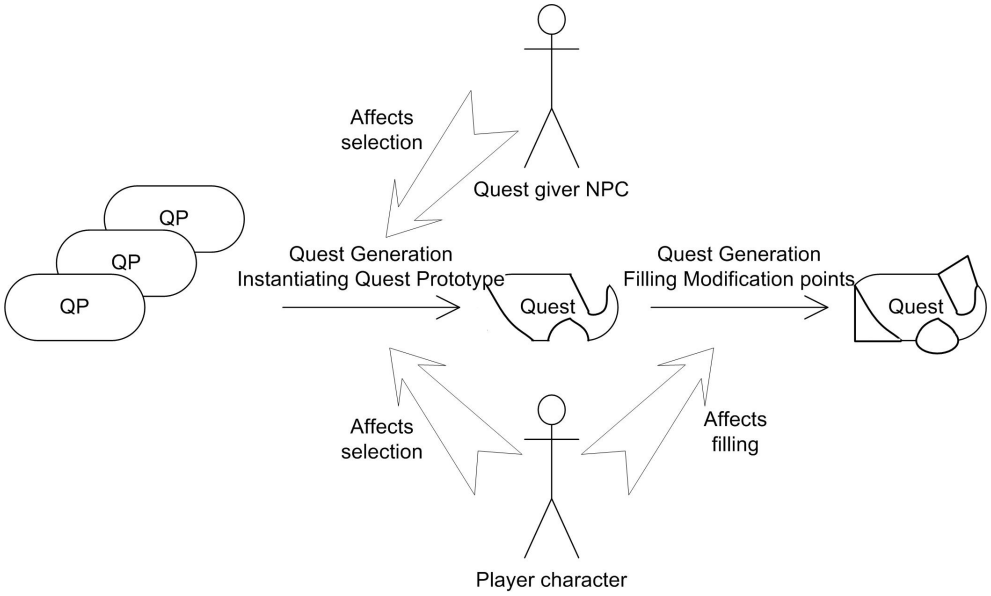

Figure 3: Elements of the approach

The quest generation is performed when the player character talks to one of the quest giver NPCs. However, CAGE lacks of a conversation engine, therefore in our realization of the approach no actual dialogue with the NPC is performed. Instead of the dialogue, the generated quest is outputted to the player character. The same small short cut is used in quests including conversations with different NPCs, as the conversations are resulted immediately. The following steps are performed when the player character talks to quest giver NPC:

1. The CAGE game engine calls the generator program. The data structure containing the player character attributes, and the quest giver NPC attributes (e.g. motive and organisation), is given as a parameter to the quest generator.
2. Probabilities of quest prototypes are calculated using data gained in the previous step.
3. One quest prototype is drawn.
4. New quest is instantiated from the quest prototype.
5. Code of the generated quest is performed. During this step, modification points are filled using player character data.

The context of quest generation implementation was an existing environment, the CAGE game engine. We have developed the CAGE game engine to be a generic platform for different types of computer games. It is also used as a platform in the game programming course of Tampere University of Technology. CAGE is implemented using the C++ programming language and the Lua scripting language. CAGE graphics use Crystal Space [1], a crossplatform software development kit for real-time 3D graphics. CAGE offers a wide function interface, which can be used from Lua scripts, e.g. to create creatures, items and set their behaviour and properties. CAGE supports state machines, which states and state transitions are written in script files. It also enables modifying the camera behaviour, the input handling and offers several other functionalities. This wide Lua interface allows using of CAGE for many different game genres. In our approach the NPCs and objects belonging to the generated quests are inserted into the game world of CAGE using the Lua interface. The elements of approach are presented in Figure 4.

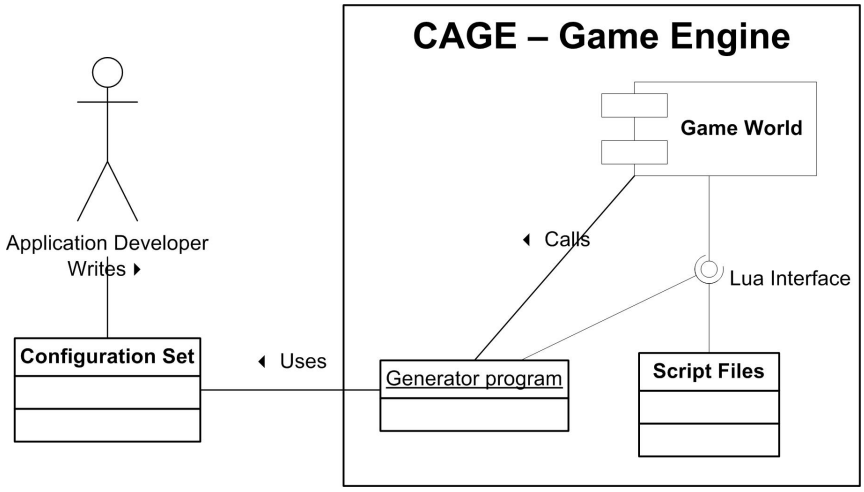

Figure 4: Elements of the approach

3 Realization of The Approach

In order to demonstrate our approach, we have implemented a quest generator program, which can generate personalized quests at run-time. The tool has been implemented using Lua scripting language. We have used the implemented tool with an existing environment the CAGE game engine. In our realization of the approach, when the player character talks to one of the quest giver NPCs of the game world, the CAGE game engine calls the quest generator program with the data structure containing the player character attributes, and the quest giver NPC attributes (e.g. motive and organisation) as a parameter.

The quest generator performs run-time generation of quest using configuration set, the player character attributes and the quest giver NPC attributes. In the following we describe structure of configuration sets and go through the generation process in more detail.

3.1 Configuration Sets

A configuration set is a collection of quest prototypes and character classes, races and skills. Quest prototypes are defined by the application developer; others are defined in the used CRPG. The example configuration set was designed for a western CRPG and consists of character classes and skills suitable for such. The player character can belong to the following classes, presented in Table 1.

Table 1: Character classes in configuration set used in the examples

Class	Effect on the character's skill set
Sheriff	good fighting skills
Bandit	mediocre fighting skills and thievery skills
Gunslinger	extraordinary fighting skills, low values in other skills

Character skills are also used in quest generation. For example if character has high value in back-stabbing skill, he is much more likely to be offered assassinate missions. In our configuration sets, the scale for the character skills was from zero to ten. Zero meaning the character has no knowledge of the skill and ten meaning he is a superior in the skill. A skill starts to affect the generation if it has a value three and the effect increases with higher values. A fraction of the skill list used in the configuration set is presented in Table 2. Our current implementation uses five attributes (strength, intelligence, agility, charisma and education) and fourteen skills. In our implementation we concentrated on how skills in the configuration set affect on the generation and at the moment part of the skills have only minor game effects. A full-scale CRPG could easily have dozens of character classes and over hundred skills. In addition, a fantasy CRPG can have player characters of different races, for example dwarves and elves.

Table 2: Examples of skills used in the configuration set

Skill	Description
Running	The moving speed of the player characters is increased.
Persuasion	The player character gains better rewards from quests.
Rifle	The player character hits the target with a rifle easier.

The application developer defines the quest prototypes. The quest prototypes are meant to be instantiated several times each. If a special quest, which happens only once in the game, is needed, it can be scripted in the same way as quests were implemented without using the approach. Quests instantiated from same prototype appear in a different forms in the game, because of the modification points. An example quest prototype definition is shown in Listing 1.

```
-- quest creation, reward, name, quest function name
q1 = Quest.create(100, "find_item", find_item);
-- positively affecting skills and attributes
pos_skill_list = {"search", "find food","running"}
-- negatively affecting skills and attributes
neg_skill_list = {"gambling"}
-- connecting previously created lists to quest prototype
q1:add_pskill_list(pos_skill_list);
q1:add_nskill_list(neg_skill_list);
-- NPCs of any faction can offer quests from this quest prototype
q1:add_faction("any");
-- adding quest prototype to the generator
table.insert(quest_type_list,q1);
```

Listing 1: Quest prototype definition

Parameters in the first line are experience gained for completing the quest, the quest name and the function, which will be called when the quest is performed. In this particular case, in the find_item function, the correct item is inserted into the game world and the end conditions of the quest are specified, in this case the quest ends as the player picks the item up.

List of positive effecting skills is specified in the line 4. If the player character has some of these skills, the probability of the quest prototype is increased. Similarly negative effecting skills, specified in the line 6 decrease the probability of the quest prototype. Faction of quest prototype is specified in the line 11, in this case, the player character can gain the quest from NPC of any faction.

When a quest prototype has been selected in generation, and a new quest is created using it, quest prototype function is called. An example of quest prototype function is presented in Listing 2.

```
function kill_outlaw_gang()
-- creating new modification point, and defining attributes and skills
  mp = Modpoint.create({"rifle", "level"});
  num = mp:high_skill("rifle", 5);

  for n=1,num do
    local cowboy = cowboy_new(random_location_for_group(center_point));
    table.insert(living_enemies, cowboy);
  end
end
```

Listing 2: Quest prototype function definition

In the code example, random_location_for_group() function returns x,y and z coordinates for the created bandits. The effect of skill or attribute is modelled using frequency function. The form of function and maximum return value are defined into the modification point. In previous listing this is shown in line 4. In the example, if the player has the high skill on rifle, he most likely is going to encounter five or four bandits. An application interface for frequency functions, that are used in modification points, is described in Table 3.

Table 3: Possible frequency functions used in modification points.

Function	Description
random(m,n)	Creates a random number between m and n. Uses random generation of Lua with current time as a seed number.
high_skill(skill, m)	Emphasis on high values of the distribution using character skill.
low_skill(skill, m)	Emphasis on low values of the distribution using character skill.

3.2 Generation Process

The quest generation process uses i) the quest prototypes gained from the configuration set, ii) the attributes of the player character and iii) the attributes of the quest giver NPC. Each quest prototype has probability to be chosen, which is calculated based these values. The probabilities of quest prototypes

are calculated each time before generation of a new quest. In the calculations the probabilities are normalized, therefore sum of quest probabilities is always one. The probabilities of quests prototypes from one test drive of generation process are seen in Figure 5.

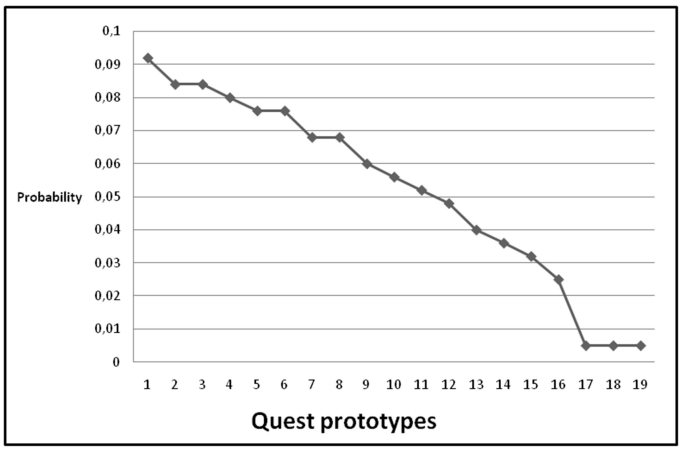

Figure 5: Probabilities of the quest prototypes

Originally each quest prototype has equal value to be chosen. Values are modified by affecting skills and quests solved earlier. Three quests prototypes in the Figure 5 with the highest numbers are the ones, which quest instances have already been solved in the current game; therefore their probabilities are decreased to prevent a quest of same type appearing soon again. A positively affecting skill increases probabilities if the player character has a value three in skill, and the effect increases with higher values. However, a negatively effecting skill decreases the probability starting with skill value one.

When quest prototype has been selected from the quest pool, an instance of quest is generated from the quest prototype. Function linked to the quest prototype is called, which generates needed items, creatures and NPCs to the virtual world of CAGE.

4 Generation Examples

We demonstrate the use of the generator program in a western CRPG prototype by presenting example cases. At first we describe the player character attributes and then present two examples of generated quests for that character. In the examples, the player character is a sheriff. He has high values in combat attributes, therefore combat type quests, for example catching outlaws are the most common generation outputs. The attributes and skills of the player character are shown in Figure 6.

In the first example, the player character has gained a quest to get rid of the bandit group terrorizing the village. When the bandits see the player character, they attack him. The attack of the bandits is shown in the Figure 7. Variables of this example quest, for example, the location of the bandit group is given randomly. However, the number of bandits and weaponry of bandits come from the modification point, and are based on the attributes of the player character.

In the second example, catch rare bird – quest prototype was instantiated. Items, creatures and NPCs relating to the quest were inserted into the virtual world, in this case, the bird and the small village. The bird and the houses relating to the quest are shown in the Figure 8. In this quest, the player character needs to catch valuable bird, which has escaped from the village. No modification points were used in this quest, and all modifying parameters, the starting location of the bird, moving speed and other used variables were generated randomly.

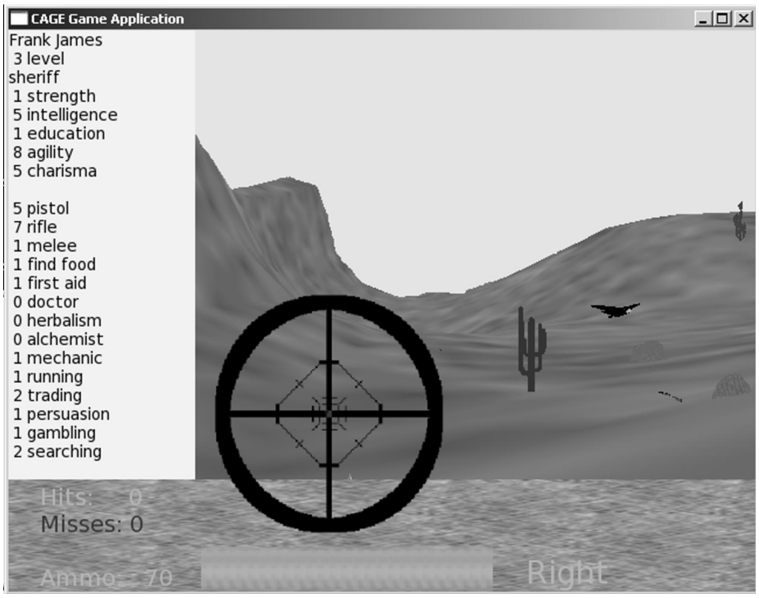

Figure 6: Attributes and skills of the player character.

Figure 7: Kill outlaw quest – a huge group of enemies is attacking the player character.

The presented approach assists the application developer in creation of quests; instead of scripting each individual quest the application developer defines quest prototypes. Application developer work is still needed to create meaningful configuration set for the used CRPG. Short quests like ones presented here, are very easy to define in configuration sets, and the approach enables also more sophisticated ones with more events and NPCs. In case of longer quests, more application developer work is needed as he or she must program specific events into the quest prototype. However, in that case number of modification points increases as well and therefore instantiated quest has lot of changing elements and has higher replayability value.

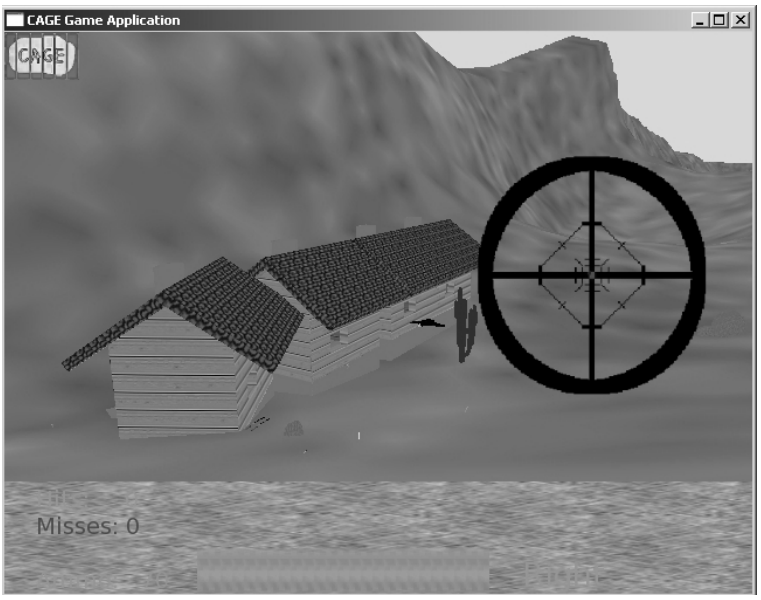

Figure 8: Catch rare bird quest – the player character is catching the escaped bird.

5 Related Work

SQUEGE [8] is a quest generator with somewhat similar mission than ours. Also the quest patterns used in their solution resemble with our quest prototypes. ScriptEase [7] is a generative design pattern tool created by partly same authors. It can be used with SQUEGE to automate the scripting procedure. However, when using the quest patterns the quests are generated in the development stage of the game, and those are not personalized for the player character.

QuestBrowser [10] is a brainstorming tool for aiding the application developer. It aims at offering more possible ways of accomplishing quests in CRPGs. Usually in a CRPG quest there is only one way of accomplishing a certain quest and the player lacks of choice. Although QuestBrowser can help the application developer in creation of quests for CRPGs, it does not aim at automatic generation of those.

Traditionally most CRPG quests are based on combat. The Grail Framework [11] is a framework designed to address this issue, and produce goal-based quests, instead of combat based ones. The GrailGM is the run-time game master portion of the Grail Framework. As the player character acts in the game world, GrailGM uses filters to find quests from the quest library that are suitable to the player character. Although their system uses special skills of player characters, character attributes are not used in the personalization. In addition, game difficulty balancing issues are not addressed.

Doran and Parberry [3] have made structural analysis of CRPG quests. Based on the study of several online CRPG quests, they believe that quests have well defined structure that may be used to generate new quests. They further believe that trough analysis of the reasons of NPCs grant quests they are able to determine the types of quests that are appropriate in various situations, and that may lead to a greater sense of realism. They have also evaluated their analysis and propose a classification of RPG quests based on this structure, and describe a prototype quest generator based on this classification [4]. However, they tell that further work is needed before they can demonstrate its ability to generate quests equal in quality to hand-crafted quests. We share the opinion of the well defined structure of CRPG quests and consider that NPCs reasons could increase sense of realism.

6 Conclusions

Huge virtual worlds of CRPGs are an advantageous domain for generative tools. Typically lot of application developer time is spent on programming those. Great amount of this time is spent on scripting predefined quests, and even more time is needed if personalized quests for different kind of player characters are created manually. But the personalized playing experience is a key factor in making the players of computer games feel involved in the virtual world, therefore creation of personalized quests should be seriously considered when developing CRPGs. Luckily, use of generative tools can reduce application developer time spent on these programming tasks.

In this paper, we presented a generative approach, which can be used to generate personalized side-quests for CRPGs at run-time. The approach can be used to replace the predefined side-quests and ease the work of the application developer, or simply use generated quests in addition to hard-coded ones, therefore further increasing the number side-quests in the game. The generation process is based on the player character attributes, which ensures those are suitable for the player character.

Although the personalization aspect of the quest generation process is well studied in this approach, the generated side-quests are still rather small in size. However, there are no technical limitations that would limit extending the quest sizes and more complicated quest prototypes, and therefore longer generated quests, can be specified by the application developer. We will continue the research of generative approaches for CRPGs in the future.

References

[1] Crystal Space 3D. `http://www.crystalspace3d.org/main/Main_Page`.

[2] Lua scripting language. `http://www.lua.org/`.

[3] Jonathon Doran and Ian Parberry. Towards Procedural Quest Generation: A Structural Analysis of RPG Quests. Technical Report LARC–2010–02, Laboratory for Recreational Computing, Dept. of Computer Science & Engineering, Univ. of North Texas, 2010.

[4] Jonathon Doran and Ian Parberry. A Prototype Quest Generator Based on a Structural Analysis of Quests from Four MMORPGs. In *Proceedings of Foundations of Digital Games, Procedural Content for Computer Games Workshop - FDG 2011*. ACM, 2011.

[5] Aarseth Espen. From Hunt the Wumpus to EverQuest: Introduction to Quest Theory. In *Proceedings of 4th International Conference on Entertainment Computing - ICEC 2005*. Springer, 2005.

[6] Jeff Howard. *Quests, Design, Theory, and History in Games and Narratives*. A K Peters, Ltd., 2008.

[7] Matthew McNaughton, Maria Cutumisu, Duane Szafron, Jonathan Schaeffer, James Redford, and Dominique Parker. ScriptEase: Generative Design Patterns for Computer Role-Playing Games. In *Proceedings of the 19th International Conference on Automated Software Engineering*, pages 88–99. IEEE, 2004.

[8] Curtis Onuczko, Duane Szafron, Jonathan Schaeffer, Maria Cutumisu, Jeff Siegel, Kevin Waugh, and Allan Schumacher. A Demonstration of SQUEGE: A CRPG Sub-Quest Generator. In *Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment International Conference - AIIDE 2007*, pages 110–111. The AAAI Press, 2007.

[9] Brian Schwab. *AI Game Engine Programming*. Charles River Media, Inc, 2004.

[10] Anne Sullivan, Michael Mateas, and Noah Wardrip–Fruin. QuestBrowser: Making Quests Playable with Computer-Assisted Design. In *Proceedings of the 8th Digital Arts and Culture Conference - DAC 2009*, 2009.

[11] Anne Sullivan, Michael Mateas, and Noah Wardrip–Fruin. Rules of Engagement: Moving Beyond Combat-Based Quests. In *Proceedings of Foundations of Digital Games, Intelligent Narrative Technologies III Workshop - FDG 2010*, 2010.

[12] Susan Tosca. The Quest Problem in Computer Games. In *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment - TIDSE 2003*. Fraunhofer IRB Verlag, 2003.

[13] Anders Tychsen, Susan Tosca, and Thea Brolund. Personalizing the Player Experience in MMORPGs. In *Proceedings of 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment - TIDSE 2006*, pages 253–264. Springer, 2006.

[14] Juha-Matti Vanhatupa. Generative Approach for Extending Computer Role-Playing Games at Run-Time. In *Proceedings of 11th the Symposium on Programming Languages and Software Tools and 7th Nordic Workshop on Model Driven Software Engineering - SPLST 2009*, pages 177–188. Tampere University of Technology. Department of Software Systems, Report 5, 2009.

[15] Juha-Matti Vanhatupa. Guidelines for Personalizing the Player Experience in Computer Role-Playing Games. In *Proceedings of 6th International Conference on the Foundations of Digital Games - FDG 2011*, pages 46–52. ACM, 2011.

[16] Juha-Matti Vanhatupa and Teemu J. Heinimäki. Scriptable Artificial Intelligent Game Engine for Game Programming Courses. In *Proceedings of Informatics Education Europe IV - IEE IV*, pages 27–32, 2009.

Publication V

J-M. Vanhatupa. Browser Games for Online Communities. In *International Journal of Wireless & Mobile Networks (IJWMN), vol. 2, no. 3, 2010*, pages 39–47. August 2010.

BROWSER GAMES FOR ONLINE COMMUNITIES

Juha-Matti Vanhatupa

Department of Software Systems, Tampere University of Technology, Tampere, Finland
juha.vanhatupa@tut.fi

ABSTRACT

Games played directly inside the web browser have many benefits. Browser games do not need software installation. Furthermore since the web has become the ultimate collaboration environment, the games are available for numerous players that can play in collaborative fashion. Through history online communities have birth alongside with browser games. Nowadays online communities have achieved massive user numbers and those can be important part of the browser game itself. This article targets at analyzing and categorizing of browser games. We also discuss financial opportunities relating to browser games and technologies used in those.

KEYWORDS

Browser Games, Game Analysis, Online Communities

1. INTRODUCTION

Multiplayer games played inside a web browser have benefits over traditional computer games. Browser games offer a good platform for social gaming, because players can interact with each other – attack other players, form alliances, create armies, and work together to accomplish greater goals.

The fact that games are played inside a web browser considerably lowers start-up threshold of gaming as software installation is not needed. The browser games are available also for players, who have never bought a single computer game and have no intensions to ever do so. Browser games can be played anywhere with normal web browser and in short time periods in the middle of player's normal life, few moves in a browser game takes only few minutes instead of hours.

As start-up threshold is small, browser games are easy to advertise to friends. Friends playing some browser game are a strong lure to get people play a certain game. This is especially typical in Facebook [1] application distribution, because people send invitations to joining applications to their friends. Facebook applications also encourage this behaviour by offering bonuses for player with lot of friends playing or at least added the same application to their Facebook accounts.

Huge online communities have been formed around browser games and gaming ecosystems. Many browser game forums have hundreds of thousands users. Furthermore, many browser game clans have created their own forums, Wikipedias and IRC-channels to support gaming and associated activities. Successful strategy game clans must be well organized and the need of good communication strategy is obvious.

DOI : 10.5121/ijwmn.2010.2303

Instead of computer controlled foes, a browser game can offer a battlefield, where each opponent is a human player. This can be very strong motivator for competitive players to play browser games. Artificial intelligence controlled opponents can also be used as opponents, or end antagonist like Natars in Travian [2]. In Travian strategy game, Natars appear after almost a real-time year of intensive player against player battle to prevent the leading player alliance from winning the game.

The player needs no installation packages to play browser games, therefore no money can be gain from selling those to the players. We believe this is the future of game industry. In future people do not walk into game store to buy games; they just play them through their web browsers. The profit of the game developers must come from somewhere else than selling game packages. They can make money by advertising, selling new units, other game resource or extra features to the players.

Although time for single move in a browser game can be very short, the duration of a browser game is usually very long or eternal. Therefore it is possible to create richness impossible to normal multiplayer games. For example, some Hattrick [3] players have played the game with the same team more than ten real-life years.

Even though browser games have existed for some time now, there have been no comprehensive studies about the subject. This paper focuses on browser games properties and categorizing those. We have discovered five common features of the browser games and made a categorization, which divides the browser games into five categories. We also discuss financial opportunities involved in browser game genre. These results were found based on a survey, which targeted important browser games. This paper is based on an earlier conference article [4], but it has been extended with respect to numerous technical details as well as community dimension.

We proceed as follows. In the next Section we briefly discuss history of browser games. In Section 3 we present our definition of browser game. In Section 4 we discuss our categorization of browser games. In Section 5 we discuss social communities, which have birth alongside browser games. In Section 6 we briefly look at technologies used in browser games. In Section 7 we discuss financial opportunities included in browser game genre. Finally, related work is discussed in Section 8, before concluding remarks in Section 9.

2. HISTORY OF BROWSER GAMES

Browser games have similarities to multi-user dungeons (MUDs), which appeared in late 1970s [5]. Those allowed multiple players to explore virtual dungeons, despite the fact that those focused on social interaction between players. Combat systems were usually rather simple. Like browser games, MUDs were multiplayer games, game time was very long or eternal and each player had one account and therefore controlled a single character. This was also the time when the first online game communities started to form. Players noticed that they enjoy playing with certain people and formed their communities to ensure they would be online to play at the same time.

Later came Earth 2025, released in October 1996, which claims to be the first of the generation of unique, interactive games designed to be played directly on the web [6]. Since those days web access has become more and more general, and this growth has led to an increasing number of browser games. People spend more time on the web than ever before. Web browser games have come popular and spread into many different game genres. Nowadays game communities are easily formed, because many games offer build in forum in game pages. Many clans build their own private forums, in addition to public one.

One main motivator in recent growth of browser games is the popularity of the Facebook social utility [1]. Anyone with some programming experience can write his or her own browser game and offer it to the other users of the Facebook. Facebook had more than 400 million active users in July 2010 and popular Facebook browser games can have fifteen million or more active users monthly [7]. For example, a popular browser game at the moment Mafia Wars [8] had more than 23 millions monthly active users in January 2010. There are more popular applications in Facebook, which are listed as game applications, but those do not fill our definition of a browser game.

3. BROWSER GAME PROPERTIES

To fill our definition of browser game, a game has to fulfil five features. The features have been found by surveying number of browser games. In the survey we played the game at least the time needed to research all the important features of the game. The testing time for a single game was from an hour to more than a year.

Browser games are for (i) **multiplayer gaming.** The game has a lot, probably thousands of users, who can interact with each other. There is more interaction than shared score list between the players. Usually players can, for example, attack each other and form alliances. There are usually several servers for the game and each of those is running a separate instance of the game. Active players can play their favourite game in several servers at the same time.

The games are played in the (ii) **web browser**. Usually no installation of any application is needed. Though some games offer graphic packages, but installation of those do not modify the rules of the game. Packages affect only to the graphic of the game. In case of Facebook application, the player has to allow the application to access his or her account, however this operation is not regular software installation.

The games are (iii) **always on**. It is possible to get attacked or receive message even when not online. Some games allow sitters, a player can name couple of other players who can log in his or her account when he or she is not online. Sitters can play using the account like the original player. This can be especially helpful during the players' holiday trips or other pauses in normal gaming. It is also possible to gain some advantage by playing in different timing than other players, as it is hard to defend against sneak attacks that happen during a night time.

The (iv) **duration** of games is very long or eternal. Game time of browser games can go on in real time, the game can be turn-based, or it can queue actions and for making them happen on a time base. The games are usually restarted after the end. In some intensive browser games the length of a single match is very short, just few minutes; however the ranking of the players is continuous.

Each player has (usually) a single (v) **account**, and he or she controls one character, group or nation. Some games offer second or even third account as additional feature if the player pays an extra fee. But those extra accounts are not used in the same instance of the game as the first account. Those can be used, for example playing with other character class than the original account was created.

4. BROWSER GAME TYPES

Browser games can be divided into several different types. Browser games in general are good at creating large worlds with a lot of players, which explains the large number of strategy games and role-playing games. Several games also combine these two types, for example, Dogs of

Seas [9] and Dark Orbit [10]. We have divided browser games into five types. In the following we go through those five types.

(i) **Strategy games** emphasize decision-making skills of the player. There is usually a large world with a lot of players and player formed alliances. The player has to battle his or her way to the top. Travian [2] is a good example of strategy game. In the start of the game the player gets one village and he or she has to build powerful nation. Players can form alliances and wage wars. Length of one Travian game is about a year.

(ii) **Role-playing games** concentrate on character development and role-playing. The player usually controls only one character instead of a nation or group. A good example of role-playing games is Forumwarz [11], a parody role-playing game that takes place on the Internet. In Forumwarz the player tries to wipe out Internet forums.

(iii) **Manager games** put player into decision-making position of a team or other group. An example of those games is Hat Trick [3], a popular football manager game. In Hat Trick each player gains control of a football team and competes against other player's teams. The players registering in the game are divided into divisions based on their real-world residence location and matches are played some once a week. Outcome of the matches are calculated by the simulation engine of the game, based on the players' attributes and other variables.

(iv) **Shooter games** are perhaps the most simple browser games. The goal is to shoot target and get points. Games vary, but the goal is always the same. Example of this group is Stick Arena [12]. Were players run around the maps, picking up weapons along the way and use those to kill opponents. The players are ranked using numbers of deaths they have inflicted.

(v) **Social networking games** are the last group. There is no combat or character development; instead the game focuses on social interaction. The Habbo Hotel [13] is an example of those applications. The Habbo Hotel also offers virtual goods for sale and the player can decorate his or her avatars' apartment with those.

5. SOCIAL COMMUNITIES

Many browser games have inspired the creation of large social communities. For example, Hattrick [3] is played in 124 countries and there are more than 850 000 active users. Each country has its own forums and in Hattrick, there are over 60000 forum posts each week. Including other forums, each league gets its own forum, helping the eight players in same league to communicate. Usual Hattrick forum conversations relate to the game mechanics as beginners seek help or hint how to develop their teams. Hattrick is a very slow processing game, only one league match is played in a real-time week and forum discussions and other activities (e.g. player transfers) are essential part of the game. Hattrick has been running since August 1997 and current season, which started in March 2010 is number 42 [14].

In strategy games, successful clans have also created their own private forums for inner communication. Those private forums are protected by passwords. Private forums help the players to coordinate attacks, ambushes, resource transfer operations, inform other clan members and other activities needing good timing and coordination between clan players.

Private Wikipedias are also used tool for inner communication of player clans. Those can be used e.g. experienced players to share their knowledge. Clan beginners can learn inside tips about the game from these private pages. While forums are used communication, Wikipedias are places to store static information.

IRC-channels can be excellent fast communication method. It is possible to reach even whole clan with message. However, effectiveness of IRC decreases if the responding player is active IRC user and has several IRC-channels open or even several IRC-clients with several IRC-channels open, because there are many conversations going on and he or she might be following some other conversation when the important message arrives. Also non game related discussions in the clan IRC-channel can decrease channel effectiveness as some players stop following it.

Some online communities have formed static groups; the group is originally formed playing certain game and when the game ends, the group has decided to stay together and continue playing; they play new matches of same game or perhaps move to play another browser game. Big clan might have several games running in different servers and all the clan players are not playing in all of those. Players in clans usually have strict ranking and players have own responsibilities, for example certain player is responsible for informing other players and other player can be responsible for setting ambushes. Those responsibilities relate to the game and are formed based whatever is needed for the clan to be successful in the game.

Static player groups make the strategy games more difficult for beginners. Even the beginner starts the game from same point as everybody else, he or she might get confused when realizing that other player have formed alliances before the game has even started yet. Beginning of the game is essential point in many browser games and strong clan tries to ensure its territory. They can do it by destroying other players or just attacking continuously, which makes attacked players too weak to defend themselves and the clan members collect resources from defenceless player until he or she quits the game. Obviously, if first experience from a certain game is being forced to be resource field (farm) of other players or being destroyed in a few days, it is not best advertisement for the game and possibly the player will never try it again.

Clans might encounter problems when selecting the members. If a clan is formed based on player location in the game, the clan might get a lot of inactive players, which do not commit to the game enough. Clan accepting all the players, which want to join it, can grow quickly and appear to be a strong. However, it can be easily won by a smaller, but well organized clan with committed members. The well organized clan can act much faster and sometimes it might take real-time days before inactive players of big clan even notice being attacked.

In those Facebook applications that fill our definition of a browser game, the social community forms into the application page. Facebook itself is a social forum and there is no need to build separate pages for discussions, because application page is already a forum. Usually in this case, people appear with their own names, instead of aliases.

6. BROWSER GAME TECHNOLOGIES

There are several technologies to implement browser games. The game can run based on a web browser plugin, for example, Flash Player, Shockwave or Java. The games usually assist the player in downloading the correct plugin if it does not exist. The second possibility is to use server-side scripting, for example, PHP, ASP or Java (using jsp-pages). The server executes a script and responses to the client with a dynamic web page.

In [15] Häsel introduces two architectural choices for browser games, client-server architecture and Peer-to-Peer architecture. In client-server architecture, the serves keeps clients up to date, clients requests are delivered to the server, which responses and delivers data to the other clients. This can be problematic in real-time browser games as it generates lot of network traffic and the server must process requests of all clients.

In Peer-to-Peer architecture each peer has same responsibilities as every other peer. Instead sending actions to the server, each player communicates with each other player. It uses less bandwidth and can be more efficient than client-server architecture. However, effectiveness measuring of Peer-to-Peer architecture can be problematic. Some models for effectiveness measuring have been developed, for example [16] and even a special Peer-to-Peer networks to optimize effectiveness of certain tasks have been developed [17, 18]. Cheating can be also hard to avoid in Peer-to-Peer architectures, because every client has its own copy of the game state.

Nowadays many browser games contain good graphics and need more resources from the client and server than early text-based browser games. However, most of browser games are still played because those games have an interesting game idea and good playability. It has not been proven that browser games with good graphics would have great advantage in luring players. Although, a game with nice graphics is probably more attractive when the player sees it first time compared to, for example a game using almost only text interface.

In addition to graphic packages, there are lot of web plugins, which act as accessories in certain browser game. Those can reveal additional information for the player, for example Foxtrick [19] in Hattrick [3] reveals more information about player attributes than it is normally possible to see. Hattrick offers separate interface for web plugins, which those can use to collect data from Hattrick. Plugins can then present this data in various forms and diagrams. Players can set security code in their Hattrick account and when using web plugin, insert the same security code and this way, allow web plugin read privileges to their Hattrick account. With read privileges web plugins cannot harm the player account. Because of good support functionality, there are dozens of web plugins available for Hattrick.

Web plugins can also modify user interface, many web plugin modifies interface of Travian [2], in a different ways, for example add new button for sending troops faster than usual way. It is even possible to get sound signal from incoming attacks using web plugin. Some web plugins are even considered to be too useful for the player and offer unfair advantage, therefore there are web plugins, which use is prohibit. For example, crop finders, which can be used to find hexes with 9 or 15 crops (normally a hex has 4 or 5 crops), are banned from Travian.

7. FINANCIAL OPPORTUNITIES IN BROWSER GAMES

Advertising is very common in browser games. Although the game itself is almost never disturbed with pop-ups, banners are a common sight in browser games. The developer sells advertising space for companies and gains money when people see or click the banners. For example, Facebook advertises itself as possibility to reach exact audience and connect to real customers. Facebook also allows advertiser to track her or his advertisement progress in real-time and choose target audience by user's location, age, gender, education or other user information [20].

Extra features are also a common way of making money in browser games. Usually it is possible to buy some virtual currency unit with real money and this virtual currency can be used to buy extra features, units, or user interface shortcuts. In addition to extra features, Hattrick [3] also sells T-shirts and possibility to get important match events, for example, goals and player injuries by SMS messages directly to cell phone.

In The Continuum™ [21], the online collectible wargame from Seven Lights, the game is started with buying units with real money. Starter pack has 40 units and Booster pack has 15. Anyone can play for free, because they receive one booster pack at the registration and can play with them as long as he or she wants. However, new units can only be gain by buying new

starters or booster. This idea is originally used by collective card games and online versions of those, for example Magic: The Gathering [22].

Extra accounts can be sold to the players. For example, in Forumwarz [11] each player can create one account for free of charge. The second and third accounts can only be bought. Other two classes cannot be played without buying new accounts. The successful browser games or even the whole developer company can sometimes be sold to other company. For example, success of Club Penguin [23], led to that the developer company, New Horizon Interactive was sold to the Walt Disney Company in August 2007 for the sum of $350 million [24].

The players gain rarely any real money from playing the browser games. Even cash prizes are common feature of for example, Internet Poker, real world cash prizes are seldom used in browser games. However, Dark Orbit [10] offers real money prizes. Each month's winner gains a sum of several thousand Euros.

Sometimes possibility of gaining valuable virtual property encourages players to cheat other players. For example in Habbo Hotel [13] some players have stolen virtual goods from others and sold those to other players with real money. Players can also sell accounts for browser games, for example beginner might want to buy an experienced character instead of acquiring the experience for his character. Selling accounts to other players is usually prohibited in browser games.

8. RELATED WORK

There are some studies about browser games, which analyses player motivations, the players, and cheating in browser games.

Schultheiss [25] has studied long-term motivations to play MMOGs and found several reasons for people to play MMOGs. Most important factors, which lead to play researched browser game Space Merchant Realms [26] were competition, learning, integration of reality and escapism. The research also points out that there are certain group of core players and on the other hand a fluctuating group of players, who come to the game and do not remain a longer time. This can be explained by many reasons; one of those is the difficulty or dominance of the core players. To fix this problem, several games give beginners some advantage, for example, they cannot be attacked at the beginning, to ease this learning phase. Despite the beginning advantage, a complex browser games can be very difficult for beginners. Game start is usually very important moment of the game and critical mistakes at that point can ruin player's rest of the game. Core players are also looking for easy points or resources at beginners' expense. In addition, in browser games were players are divided into areas and players can only attack other players near him or her, core players might want to eliminate all players, which start the game in same area, just to eliminate the possible threat in the future.

Cheating is one of the main problems in all MMOGs, not only in browser games. In [27] Kabus et al. represent that retaining control over the global game state is necessary to avoid cheating and to lock out players without valid submission. This control retaining might make use of Peer-to-Peer architecture difficult.

9. CONCLUSIONS

Online and multiplayer are current trends in computer games and number of browser games is constantly growing. Browser games are usually lot smaller in size than traditional computer games; therefore development of those is faster. However, by developing browser games, it is usually hard to find financial opportunities, as there are no game packages to sell to the players. The profit must come from other sources, for example from additional features sold to the

players or from advertisement inside the games. Although this advertisement must be convenient, because disturbing advertisement might get some players to quit playing the game.

One main feature of many analyzed browser games was a huge virtual world with a lot of players. It is obvious that huge consistent virtual worlds are fascinating for the players as it ensures a lot of opponents and possible allies.

Online communities have always grown alongside browser games. Those online communities vary from forum users to well organized clans. Most popular browser games have spread over hundred countries, which all have national forums. Many Facebook browser games have also formed huge online communities, as those games can have fifteen million active users monthly and very active user communities.

The number of different browser games has grown in recent years and this development continues. Especially computer graphics used in browser games has improved in recent years. Browser games also vary more than before and it is possible that entirely new game types will appear in the near future. This hints that there still is lot of potential in browser games as new technologies and ideas come into practise.

REFERENCES

[1] Facebook Social Utility, http://www.facebook.com/

[2] Travian Browser Game, http://www.travian.com/

[3] Hattrick Browser Game, http://www.hattrick.org/

[4] Vanhatupa Juha-Matti (2010) “Browser Games: The New Frontier of Social Gaming”, In Proc of Second International Conference of Wireless & Mobile Networks. CCIS Vol. 84, pp. 349-355, Springer Berlin Heidenberg

[5] Barton Matt, (2008) *Dungeons & Desktops, The History of Computer Role Playing Games,* A.K. Peters, pp. 37-43.

[6] Earth 2025 Browser Game, http://games.swirve.com/earth

[7] Facebook Statistics, http://www.facebook.com/press/info.php?statistics

[8] Mafia Wars Facebook application, http://www.facebook.com/apps/application.php?id=10979261223

[9] Dogs of the Seas Browser Game, http://www.dogsoftheseas.com/

[10] DarkOrbit Browser Game, http://www.darkorbit.com/

[11] ForumWarz Browser Game, http://www.forumwarz.com/

[12] Stick Arena Browser Game, http://www.xgenstudios.com/play/stickarena

[13] Habbo Hotel Virtual World, http://www.habbo.com/

[14] Hattrick history, http://hattrick.org/Help/History.aspx

[15] Häsel Mathias, (2007) “Rich Internet Architectures for Browser-Based Multiplayer Real-Time Games –Design and Implementation Issues of virtual-kicker.com”, In: Enokido, T./Barolli, L./Takizawa, M. (Eds.), Network-Based Information Systems: First International Conference, NBiS 2007. LNCS, vol. 4658, pp. 157-166, Berlin/Heidelberg: Springer-Verlag

[16] Mizanian Kiarash, Vasef Mehdi, Analoui Morteza, (2010) ”Bandwidth Modeling and Estimation in Peer to Peer Networks”, International Journal of Computer Networks & Communications (IJCNC), Vol. 2, No. 3.

[17] Klemm Alexander, Lindemann Christoph, Waldhorst Oliver P., (2003), "A Special-Purpose Peer-to-Peer File Sharing System for Mobile ad hoc networks", In Proc of Vehicular Technology Conference, VTC2003, IEEE 58th Vol.4, pp. 2758-2763.

[18] Sharma Abhishek, Shi Hao, (2010) "Innovative Rated-Resource Peer-to-Peer Network", International Journal of Computer Networks & Communications (IJCNC), Vol. 2, No. 2.

[19] Foxtrick Hattrick web plugin, http://www.ht-foxtrick.com/forum/portal.php

[20] Facebook Advertising, http://www.facebook.com/advertising/

[21] The Continuum Browser Game, http://www.thecontinuum.com/

[22] Magic: The Gathering collectible card game, http://www.wizards.com/magic/multiverse/default.aspx

[23] Club Penguin Browser Game, http://www.clubpenguin.com/

[24] Walmsley Andrew, Kids' Virtual Worlds are Maturing Nicely, http://www.marketingmagazine.co.uk/news/756021/

[25] Schultheiss Daniel, (2007) "Long-term Motivations to Play MMOGs: A Longitudinal Study on Motivations, Experience and Behavior", In Proc of the DiGRA 2007 - Situated Play,Digital Games Research Association International Conference 2007, 344-348.

[26] Space Merchant Realms Browser Game, http://www.smrealms.de/

[27] Kabus Patrick, Terpstra Wesley, Cilia Mariano, Buchmann, Alejandro, (2005) "Addressing Cheating in Distributed MMOGs", In Proc of the 4th ACM SIGCOMM workshop on Network and system support for games.

Authors

Juha-Matti Vanhatupa is a postgraduate student in Tampere University of Technology. He received his M.Sc. in Software Engineering at 2007. His research interests include browser games, tool support for game programming and content generation for computer games.

Publication VI

J-M. Vanhatupa. On the Development of Browser Games – Current Technologies and the Future. In *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), vol. 5. 2012*, pages 60–68. June 2012.

International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 5 (2013) pp. 060-068

On the Development of Browser Games – Current Technologies and the Future

Juha-Matti Vanhatupa

Department of Software Systems,
Tampere University of Technology,
P.O.Box 553, FIN-33101 Tampere, Finland
juha.vanhatupa@tut.fi

***Abstract*: Browser games are played directly in web browsers. Consequently they do not need software installation. A modern browser game is a sophisticated combination of client and server software. Nowadays there is a wide range of different technologies used to implement browser games. Traditional implementation technologies have gained new competitors, browser plugin players, which allow sophisticated graphics. In this paper we present technologies that are used to implement browser games and also accessories used in those. Especially we concentrate on use of game engines in browser game development, and give thoughts what kind of features such game engine should possess. Towards the end of the paper we discuss the future of the browser game genre, and also the future possibilities of a web browser as a gaming platform.**

***Keywords*:** browser games, multiplayer games, web technologies, web browser, application runtime, server-side scripting.

I. Introduction

Browser games are computer games played directly in the web browser and do not need software installation. This accessibility has made those extremely popular and browser games have lured many players who would never buy a regular computer game to start playing.

In the recent years, the evolution of browser games has been fast and a lot of new ones have been launched. There is also been evolution on technologies used in browser games. Traditional implementation technologies, PHP, ASP and Java jsp-pages have gained new competitors in the form of browser plugin players, which allow use of sophisticated 3D graphics.

In addition to implementation technologies, different kinds of accessories for browser games have appeared. The games can be located inside an application platform, for example Facebook social utility [16]. Different browser plugins are used to enchant play experience either to process game data for the player, or to modify the game graphics. Game engines have a strong position in regular computer game development and recently those have appeared into browser game development also. Most likely eventually those will establish same position in browser game development also.

In our previous paper [53] we presented a definition for a browser game, different browser game types, and financial opportunities related to the genre. The paper was later extended with community aspect and published in [54]. In this paper, we present technologies used to implement browser games and technologies used in browser game accessories. The rationale is that because of different software platforms (Web vs. binary), a game engine for browser games needs different kind of features than a traditional game engine. Currently there is no common understanding what features are important for such game engine. In this paper we present a proposal for these feature sets. We also discuss the future of the browser game genre as a whole. This paper is based on earlier conference article [55], but it has been extended with numerous technical details and with game engine specifications aspect.

The rest of the paper is structured as follows. In Section 2 we present our definition of a browser game and background from our previous article. In Section 3 we present technologies used in browser games. In Section 4 we discuss accessories used alongside browser games. In Section 5 we go through background of game engines, and in Section 6 we discuss game engines for browser game development. In Section 7 we focus on use of web browser as a gaming platform. In Section 8 we discuss the future of the browser games. In Section 9 we present related work, before concluding remarks in Section 10

II. Background

In this paper, we are interested and discuss browser-based games, which fulfill properties presented in our earlier paper [53]. Our category is quite similar to the category of long-term browser-based games, often referred as persistent browser-based games, presented in [45]. The main difference is that our definition is stricter, for example the game must be a multiplayer game to belong our browser game genre. There are also lots of other browser-based games, since for-example a solitaire played in Internet does not fulfill multiplayer property and by no means is an always running persistent

world necessary for a browser-based game. In this article, we use term browser game and are interested about browser-based games, which fulfill following properties:

1) the game is a multiplayer game,
2) the game can be played directly in web browser (no separate game specific software installation),
3) the game is always on,
4) the game duration is long or eternal and
5) each player has an account, which is used to play the game.

Properties 3 and 4 together with the fact that results of several actions – for example moving troops or reallocating resources – can be seen only after real world time has passed, creates an unique type of gameplay to browser games. The player can make a few actions requiring only a moment of real world time and return later to see the results of these actions. Whereas regular computer games and massively multiplayer online role-playing games (MMORPGs) require hours long continuous play. The study [30] among over 8200 players of strategy browser game Travian [48] suggests that flexibility, in essence easy-in easy-out gameplay, was one of the two primary reasons to play browser games. The other primary reason found was social relationship involved in the game play. Browser games are social games, and huge online communities have birth alongside browser games [54]. The online comminute gathered alongside the game can be a huge motivating factor for playing, and when considering to quit playing, the community might be the reason why players keep playing [45].

Originally browser games appeared at 1990s, for example the Earth 2025 [15] was released in October 1996. It is said to be the first one of unique, interactive games designed to be played directly on the web. Nowadays web access has become almost standard in modern houses and modern mobile phones are able to access the web. This development has lead to the explosion of the number of browser games. One important motivator in recent growth of browser games is the popularity of application platforms like Facebook social utility [16]. Facebook lets users to program their own browser games, and supports several programming languages.

MMORPGs are also popular multiplayer games. Those are separated from browser games, because those games have installation software, which must be bought. However, the gameplay of a role-playing browser game, for example RuneScape [44] can be very close a regular MMORPG as it is can also be played in hours long intensive sessions. Thus the different genres easily merge to gain the best properties of both genres.

III. Browser Game Technologies

A browser game consists of several components. The server-side application runs the game and is executed at the server. A web browser used to access the game acts as the client-side application. Databases are used in the server-side to store huge amount of data needed in browser games. Components of a browser game are presented in Figure 1.

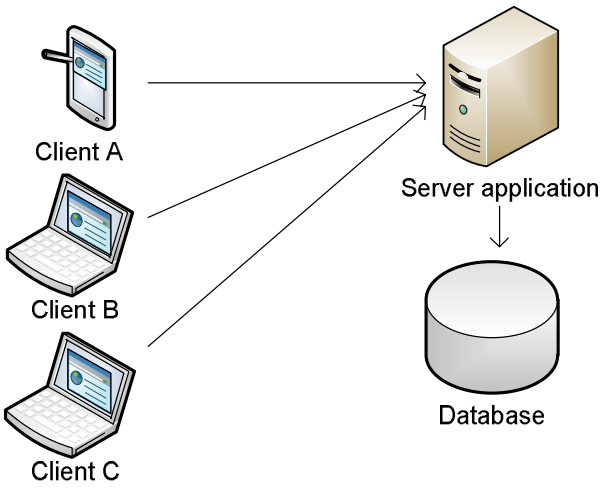

Figure 1. Browser game components.

A. Server Application

The server side applications must be robust and fast reacting. Main technologies are application runtimes, which are installed into web browser as plugins and server-side scripting. In addition to these two main technologies there are others used to implement the user interface and communication with server, and these are listed as a third group.

1) Application runtimes

Adobe Flash player [2] is a cross-platform browser-based application runtime. It is usually used for dynamic games with 2D graphics. Flash player combines animation engine with programming environment. Flash is built on an object-oriented language called ActionScript [1]. Flash Player has advantage in running graphics when compared to server side-scripting as PHP. However, 3D graphics are not as good as 2D graphics in Flash. Flash Player has small system resource requirements and can be run in Windows, Linux and Mac operating systems. A good example of browser game using Flash Player is Dark Orbit [13], a strategy / shooter game with over 35 million registered users. The player controls his starship and fights against other players. Dark Orbit also offers chance to win real money.

Adobe Shockwave Player [3] can be used 3D games with rich graphic environment. However, at the moment it is mostly used for single player games played in web browsers instead of multiplayer browser games. Shockwave Player is not available for Linux operating system.

Microsoft Silverlight [36] is a development platform that can be used to create applications for web, desktop and mobile devices. Silverlight applications can be written in any .NET programming language and .NET development tools can be used in programming Silverlight applications. At the moment there is not yet many browser games developed using Silverlight, but it is a potential environment for those games. There is an open source implementation for Linux named

Moonlight [39], which is developed by Novell in cooperation with Microsoft.

Java applets [27] are usually small applications run in the Java runtime in web browser. However, it can be used for full-scale browser games, RuneScape [44] is a very popular game launched already 1999 is done using a Java applet.

Unity web player [50] can run games built using Unity game engine [51]. The games build on top of Unity have smooth graphics. For example, Battlestar Galactica Online [9] launched at February 2011(beta test), gathered two million registered players in three months [10], and at the moment has almost nine million registered players. However, Unity web player is not available on Linux and in general it has not yet spread wide.

Modern application runtimes can run many different operating systems. Available operating systems for application runtimes are presented in Table 1.

Table 1. Application runtimes in different operating systems.

Application runtime	Windows	Linux	Mac
Flash Player	Yes	Yes	Yes
Shockwave	Yes	-	Yes
Silverlight	Yes	(Yes) [1]	Yes
Java	Yes	Yes	Yes
Unity	Yes	-	Yes

1) Silverlight has an open source implementation for Linux.

2) Server-side Scripting

PHP [42] is a general purpose scripting language, especially suited for web development. PHP scripts can be embedded into HTML [23] documents. When a client browser requests file from the server, PHP is first interpreted in the server to HTML and then delivered to the browser. PHP is widely used in text-based browser games. It is more popular in browser games than its competitors, for example Perl or Python.

Perl programming language [41] used with CGI [11] are widely used for web development. Perl has powerful text processing facilities, and those are also useful when dealing with SQL. It is made to be easy to use, efficient and complete; however, using Perl, it is also easy to make unsecure web application.

Python [43] is another widely used programming language for web applications. It has a wide collection of web development tools and framework. However, those are not designed for web game development.

JavaServer Pages (JSP) [29] is a technology used to create dynamic web pages, which separates the user interface from content generation. JSP technology uses XML-like tags to encapsulate the logic that generates the content for web page. The page accesses server-side resources such as JavaBeans through those tags. JavaServer Pages is an extension of Java Servlet technology, which makes easier to combine fixed or static templates with dynamic content. Hyperiums [25] is a strategy browser game running since 2001 build on Java technologies. Hyperiums is purely text-based.

Active Server Pages (ASP) [34] is Microsoft's first server-side scripting environment. It allows creating and running dynamic, interactive web server applications. Usually VBScript is used with ASP, but it can be programmed with any programming language, which supports ActiveX scripting. ASP.NET [35] is a successor of ASP and no further versions of ASP are planned.

Server-side JavaScript (SSJS) [12] refers to JavaScript [28] run in the server-side, not downloaded by the application. In addition to client-side JavaScript, server-side JavaScript allows an application to communicate with a database, sharing information between the users and perform file manipulations on a server. Server-side JavaScript web pages can also contain client-side JavaScript code.

3) User Interface and Communication

In addition to application runtimes and server-side scripting other technologies are used too. However, a browser games are rarely implemented using only those, commonly those are used as a combination with mentioned ones. For example JavaScript can be used to handle mouse clicks, and AJAX to transfer data to PHP script acting as a game logic.

JavaScript [28] is a scripting language that can be used to add dynamic content to web pages. It can be used to implement web games and simple graphics. JavaScript was originally created to run in the client-side, but since SSJS is now available, a term client-side JavaScript (CSJS) is also in use. WebGL [57] is an extension of JavaScript that allows use of 3D graphics in a web browser. Currently, WebGL is supported by Google Chrome, Mozilla Firefox 4, Safari and Opera browsers.

AJAX [7] is a combination of web development technologies used in the client-side to create interactive web applications. These technologies communicate with the server without interfering with the shown web page. JavaScript is used to combine these technologies. Since AJAX is a combination of technologies already exists in browsers, AJAX requires no plugins or other additional components. However, AJAX/JavaScript combination has difficulties to handle differences between web browsers and need explicit code to cover those.

B. Client Application

The only client application needed to run a browser game is usually a web browser. The applications are also independent from the operating system of the client computer. Figure 2 shows a typical user interface of browser game, Hattrick football manager game [20].

However, each operating system cannot run every browser, and this sometimes limits playing of a certain browser game. Similar limitations relate also to other software in different operating systems, for example, at the moment iPad is lacking Adobe Flash Player support and browser games using it cannot be played with iPad.

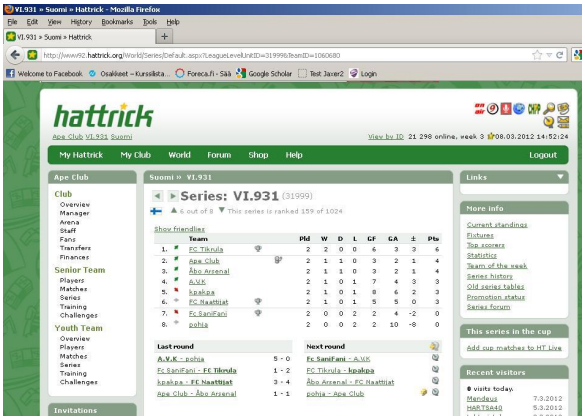

Figure 2. Hattrick Browser game user interface (Copyright Hattrick Ltd, used with permission).

Modern mobile phones can execute web browsers quite easily and those have increased popularity of browser games. Many browser games offer mobile interfaces, which are simpler versions of actual ones. Those have fewer graphic and game user interface is structured for narrow screens of mobile phones. Special installable versions of browser games for mobile phones have also been made. These special versions use the same resources as original browser games.

C. *Databases*

Browser games need reliable and fast databases. Since any database errors and breakdowns can affect the gaming experience of thousands players, reliability is an important requirement. However, it is not extraordinary that a game situation in a browser game needs to be reversed because of a database error. Usually a game is shut down for some time and reversed to previous valid situation.

Knowledge about which database is actually used is usually secret for security reasons. Most likely choices for database management systems in browser games are for instance MySQL [40] or similar. For example, Hattrick football management browser game mentioned above is believed to run Microsoft SQL [37] on a Hitachi server park; however the actual implementation of the game database system is known only by the Hattrick Team [22].

D. *Architecture*

In [26] Häsel introduces two architectural choices for browser games. The first, client-server architecture is traditionally used for online games. The server keeps clients up to data and client requests are delivered to the server, which responds and sends data to the other clients.

The second, peer-to-peer architecture relies on peers that have same responsibilities. Each peer has also own copy of the game state. Peer-to-peer architecture uses less bandwidth and can be more effective than client-server architecture. In this architecture cheating can be hard to prevent as each player has own copy of the game state, which can be vulnerable to abuse.

IV. Accessories

In addition to the browser game itself, other applications can be used in when playing the game or in the development phase. The game can be located inside an application platform, for example in the Facebook social utility [16]. Modern downloadable games are usually developed using game engines, but those are yet rarely used in browser game development. Browser plugins can enchant play experience, either processing game data for the player or by modifying the game graphics. Traditional installable application can also be used for some specific tasks, where installation as a browser plugin would not give any advantage, e.g. complicated calculations, which are made weekly in real time.

A. *Application Platforms*

Browser games located in separate application platforms, for example Facebook platform, can be programmed in several different programming languages. The application code itself is run in a separate server. Separate application platform can offer lot of publicity for the application. However, if function calls are linked through application platform API to the application, it can downgrade the performance.

Application platform can be an advantageous environment for a browser game, since the player has already registered to the platform and he can start playing simply clicking the game. This further lowers the starting threshold for a browser game. Other benefit is that the players attract other players to play certain games. A word about good games spreads rapidly inside an application platform, either by a platform specific way, like invitations in Facebook, or by using forums inside it. This kind of spreading is natural since many application platforms are social networking platforms, and therefore reflect the social relationships of their users [33].

B. *Game Engines*

Game engines [18] are the most important tools in traditional computer game development, but those have not established in browser game development. There are potential for game engines in browser game development, since browser games have many common features, which could be implemented in a product platform instead of the game itself. At present, some game engines have been developed for browser game development.

Unity game engine [51] can be used to develop browser games, which run on Unity web player [50]. The game engine is originally developed for regular computer game computer games. Unity offers support for networking, scripting, physics, lightning and advanced graphics. Unity can be scripted using three programming languages, JavaScript, C#, or dialect of Python named Boo.

Unusual Engine [52] is a game engine for browser-based games. It is build on a standard Adobe Flash [2] and ActionScript 3 [1]. Features of Unusual Engine include networking, 2D and 3D graphics, physic engine, sound, scripting, animation. Unusual Engine also has Facebook support. Game engines are discussed in more detail in the following sections.

C. Browser Plugins

A browser game can offer separate interfaces for browser plugins. Good example of separate interface for browser plugins is in Hattrick [20]. Application developers can develop programs and get those approved by the Hattrick team; those programs can be noticed from CHPP (Certified Hattrick Product Provider) logo. There are general guidelines on what a CHPP program may do and or not. For example, a CHPP program may not track opponent's player formation changes, injuries or sold players. The system also makes it possible to arrange friendly matches through Facebook.

Graphic packages installed as browser plugins can modify browser game graphics. There can be several different themes for game. These graphic packages do not change the rules of the game and use of those has no in-game benefits. For example there are several graphic packages for Travian. Those change the appearance of buildings, maps, units, or any other graphics item available in Travian.

Browser plugins are rarely used to implement basic functionality of a browser game. However, for example The Nethernet [47] is played through a special Firefox toolbar. The story of game is about an Internet war between Order and Chaos. The players browse the web and leave traps and other tools to the websites, which affect others visiting same websites. The Nethernet toolbar needs to be installed to play the game, and it is installed to the web browser as plugin and not as a traditional software installation.

D. Other Accessories

Separate installable applications are used as accessories, when playing a browser games. Those can be more suitable than browser plugins if the operation is only needed from time to time and it would be overkill to launch the application every time the browser game is played. For example, Hattrick Manager [21] calculates optional line-up for the player's team, tracks players training progress, and offers many ways to sort and present game data.

There are no limitations for programming languages used in those external applications. However, the application needs to be able to call functions in the browser game API if it wants to access it. Usually these external applications must access the browser game API to load the game data, which is then used in the application.

V. Traditional Game Engines

A modern game engine is a massive collection of software components. Typical components of a traditional game engine are presented as layer architecture in Figure 3. The core system refers to the most essential systems required by the game engine, for example memory management, IO access, and possibly other components required by the used programming language and environment. In most game engines the game loop, which calls other components, is located into the core system.

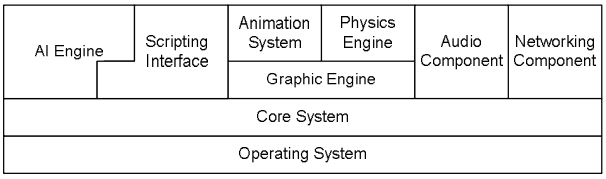

Figure 3. Game engine components.

In addition to the run-time component of a game engine, several separated tools are used when creating the game. Most of these tools relate to content creation, for example, game world editor, 3D modeling tool (for creating game characters and items), audio tool. In theory, for those tools it does not matter is the game browser-based or a regular computer game. However, at the moment, browsers are not capable of executing similar quality computer graphics as can be used in traditional computer games. Although the situation is changing and browsers are becoming more powerful application platform.

VI. Game Engines for Browser Games

A game engine for browser games needs different kind of features than a traditional game engine used to build installable computer games. We have divided the features into mandatory and optional categories. Mandatory features are useful for all browser games. Optional features are features that game engine for browser-based games can benefit, however those features i) are not useful for every type of browser games or ii) the benefit of having those is rather small. In addition to these groups we have listed few features that are common in traditional game engines, but are not significantly useful in when building browser games.

A. Mandatory Features

Registration is the only mandatory step needed to start playing a browser-based game. Since every browser-based game needs registration system, the functionality could be easily moved to the game engine. An application platform can also be used to automate the registration process, and therefore lower the starting threshold for playing the game. Though, if application platform handles the registration, registration functionality of the game engine is still needed to execute player data to the game database.

Networking component is responsible for handling network connections. The game engine would also ensure that game data relates to correct players. In the game code the players are seen only as a game object and no information about how the player is connected is present, that information is handled in the game engine.

Security features should be implemented in the game engine, and in addition also in the game itself, because security is cross-cutting aspect. Building security for browser-based game starts from design of the game application logic. Since the data send from the client can be manipulated, the client should not be able directly to command the server: (add points to this player or move this player to …). The server should always check what the game status is and ensure validity of client commands.

Database access is essential for every browser-based game. It suits well for implemented in game engine level. A game engine can offer easy access interface for using the database, in this way the SQL statements are used only in the game engine. The game code calls the game engine and data that needs to be updated into the database is, for example delivered as a parameter.

Player messaging system lets players send messages to each others. It works as in-game email system. This kind of slow messaging is suitable for most browser-based games. However, many browser games based on the player's fast reflexes, for example shooter games, are too fast proceeding that the players has no time for writing messages during matchers. In those games, the messaging system could be used between matches. In general, a player message system is very useful feature for a game engine.

Graphic interface offers functions for easily presenting graphics. It is necessary for all other types than text-based games. Publishing text-based games to be played in browser is truly a minor activity nowadays, therefore it can be generalized that graphic interface is a mandatory feature. Quality of a graphic interface can vary a lot from a few functions capable of creating 2D graphics into full-scale 3D interface.

B. Optional Features

Real-time player chat is a real-time chat between players. Need for instant messaging depends heavily on the type of the game. It might not be realistic to offer instant open chat in a browser-based game with a thousand players. However, in such game, a clan might need an internal communication channel, and for example IRC is commonly used for such.

Audio component is also most useful in fast proceeding action browser-based games. Many strategy browser-based games do not include sound effects or music any kind. The game flow of those games is slow; therefore sounds are not needed for confirmations to user actions. In addition strategic browser-based games are commonly played in public places for example, in workplaces; the players would mute the sounds anyway.

Animation system refers to ready-made interface offering possibility to add animations to graphic objects in the game. Benefits of animation interface are linked with level of graphics needed by the game. Usually benefits of possibility to animate graphics are very limited in development of strategic browser-based games. However, a ready-made animation interface could be very useful in fast-proceeding action games.

Facebook support is one potential feature for a game engine at the moment. Facebook social utility [16] is currently the most popular website in the United States; it is very popular in elsewhere too. An application platform can be advantageous environment for browser-based game, since those automate the registration process and can offer publicity for the game.

Artificial intelligence (AI) engine is useful in most browser-based games. However, all of them do not include computer controlled players. For most browser-based games AI engine does not need to be very sophisticated, basic functionality that enables simple AI enemies to be implemented easily is enough.

Physic engine is a common feature in traditional game engines. However, its usefulness is questionable in a game engine for browser-based games, because only a small amount of those use a kind of 3D graphics that benefits from a physic engine. Although there are many non-browser, installable games using heavily physic engine. For example Angry birds [6], is currently very popular, because modern touch screen devices have made new forms of gaming possible. There are also browser-based conversions of these games; however, usually those are not multiplayer games and do not fill other properties also.

C. Left out

A scripting interface is a very common feature of traditional game engines. It can speed up the development of game rapidly and can be left open, to provide an interface for players to build their own modifications from the game. However, many browser-based games are already built using a script language; therefore there is no use of building a separate interface for another scripting language. In addition, because a server-side part of the browser-based game is located to a server, it would be unwise to let players execute their own script code to the game.

VII. Web Browser as a Gaming Platform

The Web has gone through a long evolution from a simple document browsing and distributing environment to a rich software platform [8]. These early roots of Web are still evident and traditionally it has been difficult to build sophisticated, interactive applications without using plugins. This situation has hindered browser game development; earlier browser games were purely text-based. Recently plugins have spread and quality of graphics and other features of browser games have improved.

Forthcoming technologies will improve the situation even further, those aim at improving the use of the Web as a real application platform. The forthcoming HTML5 standard [24] adds new features to existing HTML standard. From several new features, some affecting game development are: ability to embed video and audio directly to web pages, drag-and-drop capabilities, graphic features and offline storage database. This offline storage database possibly can be used to allow the player make his moves in browser game even the network connection is not available, and update the moves when the network reconnects.

WebGL [57], already mentioned in the technologies section, is a cross-platform web standard for hardware accelerated 3D graphics. It allows use of 3D graphics natively in the web browser without plugins. Because this eliminates the need to install separate plugins for gaming, it can have great impact to browser game development and increase the popularity of browser games.

In general, the future of web browser as a gaming platform seems bright. Currently new browser games are already in

quality close to their installable cousins and forthcoming technologies will shorten the gap between those even further.

VIII. Future of Browser Games

The software industry is currently experiencing a paradigm shift towards web-based software [46]. The majority of new software applications intended for desktop computers are web applications. Web-based software requires no installation or manual upgrades and their distribution is superior to conventional desktop-style applications.

We believe that the trend includes computer games as well as other software applications, and playing directly from the web without installation on the local computer, is finally the future of all computer games. Playing can be also paid directly on web. At the moment, almost all browser games are free to play, but when the regular computer games transforms to be played from the web, the profit must be created by monthly payments. However, the gameplay of browser games and regular computer games are different: short sessions often and constantly on versus hours long intensive play. Therefore they remain as separate game genres; even if both would be played using web browser.

There are also signs that people use more time using applications, which use the web directly, but less time on browsing the web [4]. If this trend continues and external applications are created to connect web resources, it helps the user to access the browser game without selecting the game from his bookmarks or typing the web address. Although the player would use a platform specific, external application (native app) when accessing the game from his computer or mobile phone, the game is still accessible from public computers through a web browser. This battle between browser-based and native applications [38] will settle in the future. However, as presented for success of browser games it does not matter if native apps are created to access those. Of course, it can be argued if a browser game accessed by native app is truly a browser game, because native app is installable software, but this is discussion is very close to nit-picking.

In the future mobile phones are even more close to personal computers. This opens more market for browser games, because people carry gaming device all the time, instead of that gaming is limited to the time they are at front of a computer.

Although, game engines have not established position in browser game development, it is probable that those will be important tools in the future in the field. Currently application platforms automate the registration process in browser games, but there are lots of common development issues, for example networking, database access, player chats, graphics, which can be implemented in a game engine. A game engine for browser games would also add security features like authentication of browser clients.

In addition to technologic issues, the success of browser games depends on social aspect. The nature of browser games is highly addictive, and usually the time spent on a browser game increase daily. When the player starts playing a new browser game, the game might need only a minute at the first day. Then two minutes at the second day, and so on. In strategy games the controlled territory expanses and the player needs more time just to play at same level as before. Even in manager games, where the basic functionality of the game, for example matches played per week, is not expanding, time used for searching new players and other supportive actions increases as the player advances in the game. The amount of real world time that the game demands for the player to be affective in the game world and interference with the gamers' personal life, are two important reasons why people quit playing online games [5]. However, the social pressure can be smaller and constant playing more acceptable, if the browser game is located inside a popular application platform, which the player's family and friends use a lot. These social reasons to quit playing are difficult to overcome by new technology innovations.

IX. Related Work

Development of sophisticated web applications has become increasingly complex. Kuuskeri and Mikkonen [32] aim at implementing "fat clients" and running application mostly on the client. They have created a set of guidelines how applications should be divided between the server and the client. By following these directives application developers can address the issues of complexity that are common in modern web development. Their implementation is done using server-side JavaScript. "Fat clients" could reduce the server load and make a browser game run smoother as the client could handle most user events. Security issues are intentionally left out of the paper. However, those are very important in browser games and secure environment is one requirement for successful browser game.

The main drawback for games to use a peer-to-peer architecture is lack of a central authority that regulates access and prevents cheating. Hampel et al. [19] present a peer-to-peer architecture based on an overlay network using distributed hash tables with support for persistent object storage and event distribution. They present concept of sets of controlled peers that supervise each others. This kind of redundancy can prevent cheating and improve stability by eliminating single point of failures. Goodman and Verbrugge [17] present a design for a cheat-resistant, scalable hybrid network model for massive multiplayer online games. In their design, the gamestate is still controlled by a centralized server, acting both a login point and arbiter of client behavior. Computations are distributed between clients to reduce computational load and thus increase scalability of the central server. Calculations are verified through a simple peer auditing scheme. They present that cheating can be effectively controlled in a semi peer-to-peer system with good scalability and acceptable overhead.

Performance is one important requirement for implementation technology of a browser game. PHP is a widely used scripting language in text-based browser games. Trent et al. [49] present performance comparison of PHP and JSP as server-side scripting languages. They present that under high loads JSP tends to perform better than PHP,

however, if a 5%-10% difference in throughput and performance is acceptable, then implementers of a web system can achieve similar results using either PHP or JSP.

In addition to already presented game engines for browser games, Unity [51] and Unusual Engine [52], there are also other game engines for browser game development.

Vision [56] is a commercial multi-platform game engine by Trinigy. Vision was originally released in 2003. By using WebVision, projects made using Vision game engine can be ported to be the browser-based. WebVision is free for anyone licensing Vision game engine.

Aves is a HTML5 game development framework for JavaScript that does not use plugins to run. At the moment Aves is owned by Zynga and it is not available for market, however there has been discussion about releasing the gaming component of Aves as open source. Most likely we will see more game engines using JavaScript/HTML5 combination in the future as the forthcoming standard HTML5 becomes general.

Lively Kernel [31] is a platform for developing and hosting client side applications implemented in JavaScript only. Although Lively Kernel is not designed to be a game engine, there are game applications present also in it, and it implements some features required for a game engine. For example user chat and user identification (registration) are present in the latest version.

Facebook [16] platform already mentioned earlier contains wide range of game applications. Di Loreto and Abdelkader [14] show that in the Facebook context playfulness is linked to blending of personal aspects and social aspects. The presence of friends in Facebook is the lever to push players return to use the application. In the end, the users themselves create playfulness.

X. Conclusions

Since web access has become common and modern mobile phones can also be used to play browser games, people have more opportunities to play. Generally, the future of browser games looks bright and it is probable that new technologies, for example specific game engines for browser games, allow even more sophisticated browser games. However, non-technical issues are also affecting success of browser games. Browser games are always running, therefore they are very addictive and need a lot of time.

In this paper, we presented current technologies used to implement browser games. Evolution has brought browser games in quality closer to their regular counterparts. Most likely this trend continues when game engines for browser games become common and produce same quality improvement as they did to the regular computer games. New forthcoming web standards and technologies, such as HTML5 will also make the Web better gaming platform, improving the quality of browser games.

Based on recent development towards web-based apps, we claim that finally all computer games are played directly in the web and not installed to the local computer. The trend will cause improvement on web technologies, since almost all software development is then web application programming and the current wide collection of different web technologies evolves into more sophisticated technologies and standards. Possibly with the evolution of web technologies, the number of technologies used to implement browser games, will decrease, and some set of technologies and game engines, instead of the current wide collection, will have a dominant position in the browser game development.

References

[1] ActionScript Technology Center, http://www.adobe.com/devnet/actionscript.html
[2] Adobe Flash Player, http://www.adobe.com/products/flashplayer/
[3] Adobe Shockwave Player, http://www.adobe.com/products/shockwaveplayer/
[4] C. Anderson and M. Wolff, “The Web is Dead. Long Live the Internet”. *Wired*. pp. 118-127, 2010.
[5] D. Anderson, “The Dark Side of MMOGs: Why People Quit Playing”. In *Proceedings of CGAMES’2009*, pp. 76–80, 2009.
[6] Angry Birds game, http://www.angrybirds.com/
[7] AJAX, http://www.adaptivepath.com/ideas/essays/archives/000385.php
[8] M. Anttonen, A. Salminen, T. Mikkonen and A. Taivalsaari. ”Transforming the Web into a Real Application Platform: New Technologies, Emerging Trends and Missing Pieces”. In *Proceedings of the 2011 ACM Symposium on Applied Computing* (SAC 2011), pp. 800-807, 2011.
[9] Battlestar Galactica Online, browser game, http://us.battlestar-galactica.bigpoint.com/
[10] Bigpoint company press release, http://bigpoint.net/2011/05/battlestar-galactica-online-blasts-past-two-million-registered-players-in-three-months
[11] Common gateway interface, http://www.w3.org/CGI/
[12] CommonJS, a project defining APIs for JavaScript outside the browser, http://www.commonjs.org/
[13] Dark Orbit browser game, http://www.darkorbit.com/
[14] I. Di Loreto and G. Abdelkader, “Facebook Games: Between Social and Personal Aspect”. *International Journal of Computer Information Systems and Industrial Applications*, III, pp. 713–723, 2011.
[15] Earth 2025 browser game, http://games.swirve.com/earth
[16] Facebook social utility, http://www.facebook.com/
[17] J. Goodman, C. Verbrugge, “A Peer Auditing Scheme for Cheat Detection in MMOGs”. In *Proceedings of 7th ACM SIGCOMM Workshop on Network and System Support for Games* (Net Games’08), 2008.
[18] J. Gregory, *Game Engine Architecture*. AK Peters, Ltd, 2009.
[19] T. Hampel, T. Bopp, and R. Hinn, “A Peer-to-Peer Architecture for Massive Multiplayer Online Games”. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games* (Net Games’06), 2006.
[20] Hattrick browser game, http://www.hattrick.org/

[21] Hattrick Manager Application www site, http:// www.hattrickmanager.org/
[22] Hattrick Wiki Datapage page, http://wiki.hattrick.org/wiki/Database
[23] HTML, http://www.w3.org/html/
[24] HTML5, http://www.w3.org/TR/html5/
[25] Hyperiums browser game, http://www.hyperiums.com/
[26] M. Häsel, "Rich Internet Architectures for Browser-Based Multiplayer Real-Time Games – Design and Implementation Issues of virtual-kicker.com". In *Proceedings of Network-Based Information Systems: First International Conference* (NBiS 2007), LNCS, vol. 4658, pp. 157–166, 2007.
[27] Java Applets, http://java.sun.com/applets/
[28] JavaScript Mozilla's Official documentation, https://developer.mozilla.org/en/JavaScript
[29] JavaServer Pages, http://java.sun.com/products/jsp/overview.html
[30] C. Klimmt, H. Schmid, and J. Orthmann, "Exploring the Enjoyment of Playing Browser Games". *CyberPsychology & Behavior*, XII (2), pp. 231-234, 2009.
[31] J. Kuuskeri, J. Lautamäki and T. Mikkonen, "Peer-to-Peer Collaboration in the Lively Kernel". In *Proceedings of the 2010 ACM Symposium on Applied Computing* (SAC 2010), pp. 812-817, 2010.
[32] J. Kuuskeri, and T. Mikkonen, "Partitioning Web Applications Between the Server and the Client". In *Proceedings of 2009 ACM Symposium on Applied Computing* (SAC 2009), pp. 647–652, 2009.
[33] N. Mattar and T. Pfeiffer, "Interactive 3D Graphs for Web-based Social Networking Platforms". *International Journal of Computer Information Systems and Industrial Applications*, III, pp. 427-434, 2011.
[34] Microsoft Active Server Pages (ASP), http://msdn.microsoft.com/en-us/library/aa286483.aspx
[35] Microsoft ASP.NET, http://www.asp.net/
[36] Microsoft Silverlight, http://www.microsoft.com/silverlight/
[37] Microsoft SQL Server, http://www.microsoft.com/sqlserver/en/us/default.aspx
[38] T. Mikkonen and A. Taivalsaari. "Reports of the Web's Death are Greatly Exaggerated", *Computer*, XLIV (5), pp. 30-36, 2011.
[39] Moonlight, http://www.mono-project.com/Moonlight
[40] MySQL, http://www.mysql.com/products/enterprise/database/
[41] Perl programming language, http://www.perl.org/
[42] PHP scripting language, http://www.php.net/
[43] Python programming language, http://www.python.org/
[44] RuneScape browser game, http://www.runescape.com/title.ws
[45] D. Schultheiss, N.D. Bowman, C. Schumann, "Community vs solo-playing in multiplayer internet games". In *Proceedings of The [Player] Conference*, pp. 452-471, 2008.
[46] A. Taivalsaari, T. Mikkonen, M. Anttonen, and A. Salminen, "The Death of Binary Software: End User Software Moves to the Web". In *Proceedings of The Ninth International Conference on Creating, Connecting and Collaborating through Computing* (C5'11), IEEE, 2011.
[47] The Nethernet browser game, http://www.pmog.com/
[48] Travian browser game, http://www.travian.com/
[49] S. Trent, M. Tatsubori, T. Suzumura, A. Tozawa and T. Onodera, "Performance Comparison of PHP and JSP as Server-Side Scripting Languages". In *Proceedings of Middleware 2008, ACM/IFIP/USENIX 9th International Middleware Conference*, pp. 164-182, 2008.
[50] Unity Web Player, http://unity3d.com/webplayer/
[51] Unity game engine, http://unity3d.com/
[52] Unusual Engine, http://www.unusualengine.com/
[53] J-M. Vanhatupa. "Browser Games: The New Frontier of Social Gaming". In *Proceedings of The Second conference on Wireless & Mobile Networks* (WiMo 2010), pp. 349-355, 2010.
[54] J-M. Vanhatupa. "Browser Games for Online Communities", *International Journal of Wireless & Mobile Networks*, II (3), pp. 39-47, 2010.
[55] J-M. Vanhatupa. "On the Development of Browser Games – Technologies of an Emerging Genre", In *Proceedings of 7th International Conference on Next Generation Web Services and Practices* (NWeSP), pp. 363-368, 2011.
[56] Vision game engine, http://www.trinigy.net/en/products/vision-engine
[57] WebGL, http://www.khronos.org/webgl/

Author Biographies

Juha-Matti Vanhatupa Juha-Matti Vanhatupa is a postgraduate student in Department of Software Systems at Tampere University of Technology. He received M.Sc. in software engineering at 2007. His research interests include browser games, tool support for game programming and content generation for computer games.

Publication VII

J-M. Vanhatupa and J. Lautamäki. Content Generation in a Collaborative Browser-Based Game Environment. In *Handbook of Digital Games*, pages 92–110. Wiley-IEEE Press, May 2014.