NTNU
Norwegian University of
Science and Technology

Serious Gaming

Serious content in an entertaining framework

Håvard Richvoldsen

Master of Science in Electronics
Submission date: June 2009
Supervisor: Andrew Perkis, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

The term serious game refers to a software or hardware application developed with technology and game design principles for a primary purpose other than entertainment. The genre is an increasingly important medium with respect to education, training and social change. The majority of serious games fall into the category of "edutainment" games as they focus on teaching the same subjects with the same learning principles taught in schools. In this sense, most serious games do not fulfill the potential that the genre promises. For a game to be both entertaining and give the player a sustained learning experience it has to be interesting, playable, enriching, enjoyable and entertaining at the same time.

In this thesis a serious game should be created with the purpose of recruiting high school students to NTNU. The gaming environment shall be a realistic, exciting, and informative, virtual world to make the game both serious and fun at the same time.

This should be divided into three parts; 1. A software discussion to point out what development tool is best suited for the job, 2. Create a serious game with the chosen software, and 3. Analyze the game by classifying it, extract the fun factors, and point out its potential for success.

[1]
RITTERFELD U. et.al.: Serious Games – Mechanisms and Effects, Chapter 2 and 3, 2009

[2]
WARTMANN C., KAUPPI M.: the Blender Gamekit 2nd Edition, 2009.

Assignment given: 15. January 2009
Supervisor: Andrew Perkis, IET

Abstract

This thesis is based on work done at the Norwegian University of Science and Technology (NTNU) in the field of serious gaming. The motivation for the work is to create a serious game with the purpose of recruiting high school students that undertake studies at NTNU within engineering and science.

After considerations of several available tools, Blender was chosen as the best development tool for this kind of game, and used to create "Student Quest - A First Person Student Game". The game analysis shows that the game's Primary Learning Principle is Marketing, the Primary Educational Content is Knowledge Gain through Exploration, the Target Age Group is Middle and High School, and it is developed for a Computer Platform. By extracting the fun factors, we conclude that the game passes the Playability threshold and reaches the Enjoyability threshold. By implementing the potential features suggested, the game may reach the Super Fun threshold and thus has the potential of becoming an extremely entertaining serious game.

Preface

This thesis is submitted by the author as a part of requirement for the degree of Master of Science in Electrical Engineering at the Norwegian University of Science and Technology (NTNU). The study has been performed at the Department of Electronics and Telecommunications, in cooperation with the Norwegian Center of Excellence: the Centre for Quantifiable Quality of Service in Communication Systems (Q2S).

My supervisor during this work has been Andrew Perkis, to whom I am very grateful for giving me the opportunity to explore this new, challenging field of research. He has always been available when I needed help, and has contributed a lot to the game concept.

I would like to thank Pablo Vazquez for helping me with the development for the game. He is a really talented 3D artist and game developer who deserve a big amount of credit for the product in this thesis.

I would also like to thank Ute Ritterfeld for providing me with research material from her book which is due to be published in July 2009, and Tor Ivar Eikaas and Frank Jakobsen from Cyberlab for providing me with the edutainment games.

Håvard Richvoldsen

Contents

1 Introduction **1**

2 Theory **3**

2.1 Serious Games . . . 3

2.1.1 Classifying Serious Games . . . 3

2.1.2 What Makes a Game Seriously Fun? . . . 8

2.2 Gaming Engines . . . 11

2.3 Software Discussion . . . 13

2.3.1 jMonkey Engine . . . 13

2.3.2 Blender . . . 14

2.3.3 Project Wonderland . . . 15

2.3.4 Autodesk ImageModeler . . . 15

2.3.5 Conclusion . . . 16

2.4 Blender Gaming Engine . . . 17

2.4.1 Game Objects . . . 17

2.4.2 Properties . . . 18

2.4.3 Sensors . . . 18

2.4.4 Controllers . . . 19

2.4.5 Actuators . . . 19

2.4.6 The Blender Laws of Physics . . . 20

2.4.7 Sound player . . . 20

2.4.8 Game Engine Python . . . 21

3 Design **23**

3.1 Concept . . . 23

3.2 Potential Features . . . 25

3.2.1 Interactive Menu . . . 26

3.2.2 Sound Zones . . . 26

3.2.3 Story Line . . . 27

3.2.4 Social Experience . . . 27

4 Implementation **29**
4.1 Modeling 29
4.2 Game logic 31
4.2.1 Hallway 31
4.2.2 Room 33
4.2.3 Menu 34
4.3 Installation Wizard 36

5 Discussion **37**
5.1 Classifying our serious game 37
5.1.1 Primary Educational Content, Learning Content, Target Age Group and Platform 37
5.1.2 Extracting the fun factors 38
5.2 Hardware Requirements and Performance 40

6 Conclusion **43**

A Appendix **45**
A.1 Python Scripts 45
A.2 Installation Wizard Script 47

List of Figures

2.1 Partition of Primary Educational Content, from [1] 5
2.2 Partition of Primary Learning Principle, from [1] 6
2.3 Partition of Target Age Group, from [1] 7
2.4 The big 5 fun factor categories in digital gaming, [1] 10
2.5 Screenshot from jME. Picture from [10] 13
2.6 Screenshot of the logics GUI. 17
2.7 Screenshot of Blenders sound player 20
2.8 Screenshot of BGE Python logics. 21

3.1 Comparison of a real photo (top) and the model (bottom). . . 24
3.2 Screenshot of the Electronics room. 25
3.3 Suggestion for layout of the interactive menu. 26

4.1 Screenshot illustrating modeling on top of a background image 29
4.2 Screenshot showing UV mapping. Left side: Faces in the model where the textures is mapped. Right side: Areas of the texture image that is mapped to the faces. 30
4.3 Screenshot of the hallway in bird view. 31
4.4 Overview of the hallway logics 32
4.5 Screenshot of the game menu. 35

5.1 Comparison of HQ (left) and LQ versions (right). Top: Floor patterns. Bottom: Library arc. 41

Introduction

The word *game* refers to a structured activity usually performed for enjoyment, and people have enjoyed playing different kinds of games for all time, i.e. the chinese game Mahjong was developed about 500 B.C. The rise of the computer age in the 1970's gave birth to new kind of games, digital games. At first, the games were simple and had a severe lack of narrativity, like "Pac-Man" and "Spacewar!". In the mid 80's, modern adventure games were born with Sierra's "King's Quest" and "Space Quest" series, which became the start of some of the most popular games on the market today. These game's primary purpose is, without doubt, entertainment.

Computers and digital games have increased exponentially in popularity over time, they have made significant impact upon popular culture, and are assumed to be one of the biggest influence sources to youth in modern society. Consequently, developers have begun creating games that work as educational tools. This has given rise to the new genre *serious games* which is an increasingly important medium with respect to education, training, and social change.

There is a severe lack of studies on serious gaming, as it is a relatively new field of research. Ritterfeld et.al. [1] [2] suggest a classification system for understanding and interpreting the serious games genre, and state what fun factors a game needs to be playable, entertaining, and fun. They point out that the majority of the serious games today focus on education and practicing skills, which means that most serious games are edutainment games. Marketing is the least extensive application of serious games, which is why this thesis focus on creating a serious marketing game.

Today's youth communicate and retrieve information on a number of new platforms, especially computers and the internet. It is important that institutions and companies update their marketing strategies accordingly. This thesis is an example of how NTNU can improve their marketing through the concept of a serious game with the purpose of recruiting high school students. As of today, NTNU arranges several stands, presents keynotes, and delivers brochures to inform potential students about the opportunities at the university. By expanding the marketing of NTNU through a serious game, we aim at the gaming generation by embedding the informational content in an entertaining game setting. The goal of the game, "Student Quest - A First Person Student Game", is to illustrate that education is

indeed serious, but it can be presented in an entertaining way where the gamers can recognize themselves.

Theory

2.1 Serious Games

A *serious game* refers to a software or hardware application developed with game technology and game design principles for a primary purpose other than entertainment. The term serious game was actually used long before computer games, namely in Clark Abts book entitled "Serious Games" (1977). Although his references were primarily to the use of board and cards games, he gave a useful general definition which is still applicable in the modern gaming world;

> "Reduced to its formal essence, a game is an activity among two or more independent decision makers seeking to achieve their objectives in some limiting context. A more conventional definition would say that a game is a context which rules among adversaries trying to win objectives. We are concerned with serious games in the sense that these games have an explicit and carefully though-out educational purpose and are not intended to be played primarily for amusement."

This section is based on work by Ritterfeld and Ratan [1], and Wang, Shen and Ritterfeld [2] where they suggest a framework for classifying serious games and what makes them seriously fun.

2.1.1 Classifying Serious Games

Serious games are increasingly important with respect to education, training and social change. These games are meant to provide deep and sustained learning, and reach a wide span of audiences by building on "the native tongue" of the gaming generation. In the last few years there has been a great increase in focus on developing these games as educators, health advocates etc. are joining industry officials and game designers in advertising serious gaming as a new and groundbreaking way to educate the public. There has been several studies that imply that serious games actually are more effective than more traditional pedagogy tools [3] [4]. Educators are always searching for new pedagogic techniques that mix enjoyment and education. Gaming technology might be a good alternative as the game will serve as

an entertainment frame in which serious content could be embedded. Some researchers claim [3] that any digital game may provide the player with some form of learning opportunities, regardless of whether or not the game is considered serious. The serious games genre has explicit focus on education and is thus associated with positive features such as seriousness, education and learning. Consequently, this genre may influence both user's, developer's, and parent's attitudes and selective exposure of digital gaming toward serious gaming.

Traditionally, game developers develop games with one goal, to entertain the users. The serious games developers takes a stand against this when claiming that the content of a serious game is highly desirable from an educator's perspective. The outcome of playing these games should be solely beneficial for the player by guaranteeing learning experiences, good (serious) intentions, and absolutely no negative effects. Thus, games that could lead to aggression or addiction would not qualify as a serious game. Serious games should always work as intended, providing a self-guided, enjoyable, and sustained learning experience.

To define what the genre serious game is proves to be a non-trivial task. Not only is there a lack of formal research regarding the actual effectiveness of such games, but also the definition of a serious game is vague and needs clarification. The Entertainment Software Rating Board (ESRB) defines an edutainment game as a game that

> "provides the user with specific skills development or reinforcement learning within an entertainment setting where skill development is an integral part of product".

Edutainment games are often referred to as a synonym to serious games. This, however is not the case. All edutainment games are certainly serious games, but the potential of a serious game extends beyond edutainment, thus including almost every game that has an additional purpose besides entertainment. The Social Impact Games website defines serious games as

> "entertaining games with non-entertainment goals".

This gives rise to problems when attempting to identify what such goals are, because the developers may not have a definition that coincides neither with the user's experience nor the psychological reality behind that experience.

In [1] a total of 612 games are represented in the database and used to develop a classification system of serious games. This system takes four dimensions into account: Primary Educational Content, Primary Learning Principle, Target Age Group and Platform. Primary Educational Content

are split into the following categories: academic education, social change, occupation, health, military and marketing. Figure 2.1 illustrates the partition for each of the categories.

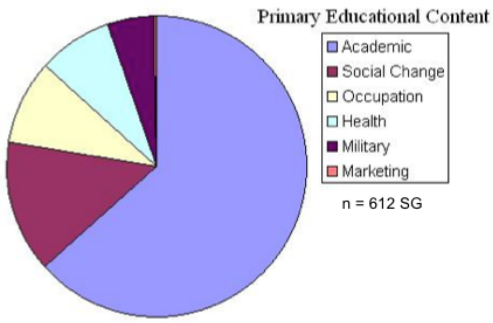

Figure 2.1: Partition of Primary Educational Content, from [1]

Academic content is by far the most represented (63%) within the dataset. Games with this kind of content are designed to teach material traditionally taught within an academic environment, like algebra, nano technology, or physics. Games in which the content is related to social change make up 14%. These games address particular social agendas such as political and social issues. 9% of the games have Primary Educational Content related to occupation. These games are designed to give the player knowledge and skills that can be applied directly in the player's occupation. Health related games (8%) are meant to provide the player with knowledge and habits that provide health, reduce risks and enable coping with health problems. The most popular game in this category is the "Wii Fit" where the goal is to make the players perform different kinds of physical activity like aerobics and tennis. Military games (5%) and games related to marketing (<1%) make up the least represented categories within the dataset.

It is through the Primary Learning Principles dimension that serious games attempt to impart skills, knowledge and ideas to the players. This is based on the understanding that the advantage of digital games is in providing opportunities for exploration, experimentation and problem solving, and not so much in the delivery of curricular content. The primary learning principles can be divided into four categories: practicing skills, knowledge gain through exploration, cognitive problem solving and social problem solving. Figure 2.2 shows the partition for each of the categories.

The majority of the games (48%) in the dataset has practicing skills as the Primary Learning Principle. These games encourage players to practice and improve basic or advanced skills, often by repeatedly practicing a narrow scope of activities and information. About a quarter of the games (24%)

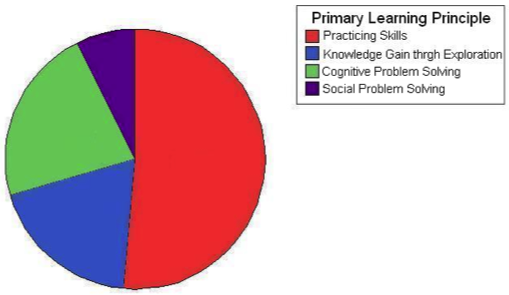

Figure 2.2: Partition of Primary Learning Principle, from [1]

focus on cognitive problem solving. These games offer the player puzzles, brain teasers or complex hypothetical situations that focus on making the player use both creativity and cognitive problem solving. A similar part of the dataset (21%) has the Primary Learning Principle of knowledge gain through exploration. Typical for these games is that the players acquire some kind of information, such as historical or biological facts, without engaging deeply into the subjects. These games often focus on a broad scope of information with little use of repetition, as opposed to practicing skills games. Games related to social problem solving (7%) encourage the players to solve different kinds of social problems through cooperation as a team and collaborating or taking responsibility as members of a society.

Serious games have a rather different partition of the Target Age Group than entertainment games. From figure 2.3 we can see that the majority (78%) of the dataset have elementary, middle and high school as the target age group. This indicates that serious games target younger players than entertainment games, as the average commercial digital game player is 33 years old [5]. This means that the potential market for serious games may be bigger than for traditional games, considering the wide span in targeted age groups.

By examining the interactions of the various categories of educational content and learning principles, Rata and Ritterfeld state that games with both the primary purpose of academic education and the educational goal of skills practice is by far the most prevalent. In the other content areas, knowledge gain through exploration and cognitive problem solving play the superior role. Hence, the majority of serious games attempt to teach the same subjects taught in school by using the same methods, like repetition and practice. Most of the serious games classify therefore as edutainment, according to the ESRB's definition mentioned earlier. This means that most games in the serious game genre are not fulfilling their potential that serious

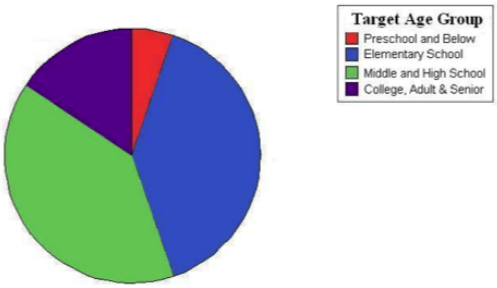

Figure 2.3: Partition of Target Age Group, from [1]

games promise. To do this, the game has to be both interesting, playable, enriching, enjoyable and entertaining, which is by far a non-trivial mix of features. It can be argued that skill practice remains both boring and uninteresting even if it is done through an interactive world with great graphics and a narrative context. Then the game is simply a different way of practicing the same skills, and only works as initial motivation. The problem is that such serious games would not be played deliberately over a long period of time, and be as non-enjoyable as the traditional way of practicing skills. Ritterfeld and Weber [3] state that a successful mixture of entertainment and education in gameplay requires parallel experiences which is best realized in games that invite to exploration and requires complex reasoning.

The game platform may also play a role in the effectiveness of a serious game. Computer-based digital games allow the user to multitask in a more extended way than other platforms like Playstation, Xbox or Nintendo Wii. This may be due to the easy internet accessibility, where the user can switch to the web browser or other applications without looking at a different screen. In addition, the average non computer-based games use more computing and video processing resources, which implies that these games may have more engaging game play and graphics. Differences in the platform's input devices may also impact the effectiveness of the games, i.e. it is much easier to navigate a first person view with a keyboard and mouse than with non-computer interfaces like a gamepad.

The classification system is not a final categorization that should yield all serious games, but is meant merely to serve as a new tool and a guide to understanding and interpreting serious games as a medium. In the next section we provide a summary of research done in [2] on what features make a serious game enjoyable.

2.1.2 What Makes a Game Seriously Fun?

"Fun" is a word which is often linked to game, as games are expected to be fun. "Fun" and "serious" are two words that seldom occur together so one could say that the serious games genre is an oxymoron. To point out exactly what features make a game fun is not trivial and is dependent on many aspects. Fun may be dependent on the individual player, and what makes an entertainment game fun may not be the same as for serious games. By identifying game elements that contribute to overall enjoyability, Wang et al. establish a frame of reference for understanding media enjoyment in both entertaining and serious games. The result of this research is a proposal of "the big 5 in game enjoyment", and a three level threshold model for digital game enjoyment.

The analysis of the dataset indicate the following 5 of 27 fun factor categories as the most frequent: Overall game design, visual presentation, audio presentation, complexity and diversity, and control. These categories are not only the most important for gamers, but also for game designers and developers. The three least frequent categories are fantasy, presence, and interactivity (table 2.1). It is important to note that these categories may not be the least frequent because they're unimportant to game enjoyment, but merely because experienced gamers simply take these features for granted. The fact is that these categories are unique characteristics of games compared to other media formats, such as books and movies [6]. In [7] [8], these factors are pointed out as crucial for offering players an emotionally engaging game experience, which is the core of game enjoyment.

Based on this, the following "Big Five" of digital game enjoyment is proposed: Technological capacity, game design, aesthetic presentation, game play entertainment experience and narrativity. These five clusters are very broad, abstract and lexical dimensions, and are not meant as a final definition, but to provide a potential generic taxonomy in understanding enjoyability factors in digital games.

Further comparison of the content categorial rankings indicate that factors like humor, mechanics and gratification appear most often in positive comments, while control, usability, challenge and artificial intelligence is more often related to negative comments. Factors like characters, social interaction, novelty, realness and gratification turn out to contribute the most in the "fun" games with high rating, while overall entertainment game play experience, story line, and length are the biggest contributers to diminish the entertainment value in "not fun" games. This imply that there exists certain thresholds that a game has to pass to become playable, entertaining and finally super fun.

Table 2.1: Top 5 and bottom 3 Content categories and examples

Content Categories	Examples
1. Overall Game Design	Game elements, logics, rules, procedures, objectives and how they interact
2. Visual Presentation	Style, quality and sophistication of graphics
3. Audio Presentation	Quality of auditory components and effects
4. Complexity and Diversity	Number, level and interconnection of meaningful acts presented to the player in the game
5. Control	Ease of use and comfortable feel of game control devices
25. Fantasy	Need of a fantastical and imaginative experience that is unrealistic and impossible in real life
26. Presence	The player's experience of immersion and feeling of actually being in the virtual world
27. Interactivity	Continous action and reaction loops between the player and the game world

The playability threshold is based on common complaints related to technological capacity and basic game elements like usability, control, challenge and visual presentation. These are typical must-have features, and are expected to be present for a game to be playable. Not many players would be interested in a game that looks ugly, takes forever to load, has numerous glitches and becomes repetitive. Thus, if these features are not present, the player may experience feelings like disappointment, frustration and irritation. A game that passes this threshold is more likely to be picked up for a try by players.

The enjoyment threshold is made up of factors mentioned in both positive and negative ways, and fun factors related to aesthetic content and game design. Typical factors include quality of the visual and auditory presentation, complexity and diversity, mechanics, freedom, levels, balanced degree of challenge, and gratification. Thus, an enjoyable game should have nice graphics, good sound effects and provide the player with a variety of options

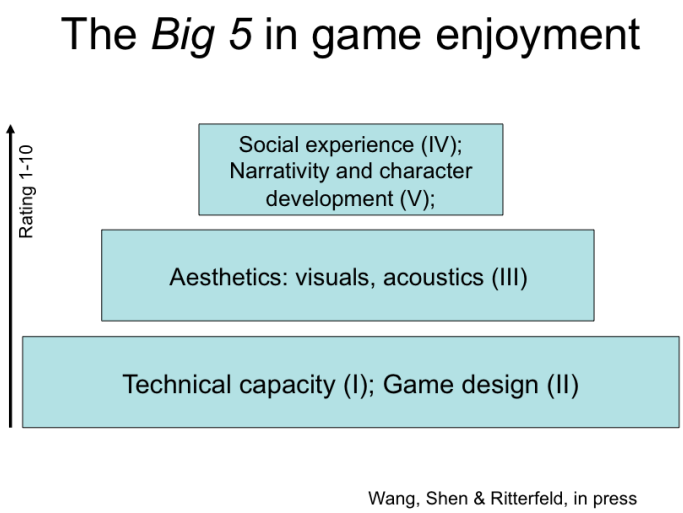

Figure 2.4: The big 5 fun factor categories in digital gaming, [1]

to explore the game world at different levels. Overcoming the enjoyment threshold offers possibilities of an appealing and fun gaming experience.

The final threshold is the super fun threshold which is made up of factors that make games extremely entertaining, and thus popular. They are extracted from the top games in the dataset, in which several has been awarded "gamer's choice" awards. Factors related to the role of narrativity in the game (story lines, characters, and humor), the player's social interaction during and after playing the game-play experience, extraordinary game design elements, and superior quality of aesthetics, are typical super fun factors. The two most prevalent is narrativity and social interaction, hence should organizations and institutions developing serious games focus primarily on the narrative and social aspects of the game, instead of focusing on improving the design and looks. If a game incorporates these super fun factors, it becomes exceptionally entertaining and might become a hit.

The Big Five may overlap across these thresholds, but technological capacity roughly defines the playability threshold and the game design and aesthetic presentation defines the enjoyability threshold. Finally, the narrativity and entertainment game play experience are often the two factors that separate fun games and super fun games. For the future it will be exciting to see whether or not a serious game can overcome the enjoyment threshold and ultimately pass as a super fun serious game.

2.2 Gaming Engines

A gaming engine is a software package designed for the creation and development of video games. Through a game engine, you interact with a 3D world in realtime by controlling objects which can interact with other objects in that world. A gaming engine consists of a framework with a collection of modules for interactive purposes with core functionality including: rendering for 2D and 3D graphics, scripting, a physics engine, collision detection, audio, networking, animation, artificial intelligence (AI), networking, threading, and a scene graph.

For a long time, game developing companies used a lot of resources on developing their own gaming engines for in-house use, and upgrading them as the hardware got faster and better versions were needed. SCI by Sierra and SCUMM by LucasArts powered most of the adventure games distributed in the 1980's and 90's like "King's Quest" and "Indiana Jones - Fate of Atlantis". These engines are quite simple compared to the ones used in more recent games, such as id Tech in Quake and the Unreal engine. Over the past several years, the cost of always having an in-house gaming engine up to date has grown significantly. Several companies have consequently begun specializing in making gaming engines and gaming engines components, instead of developing games. What these companies distribute is called middleware because they provide a flexible and reusable software which provides all the modules needed to develop games, whether you are a skilled programmer or not.

There are two types of gaming engines, called *true* and *fake* engines. The early gaming engines like SCUMM and SCI are fake engines. Fake means that the game logic, or decision making, isn't done on an object level. In these engines a higher intelligence (HI) in the game control all the objects like moving them when appropriate, detecting collisions and keeping track of their condition i.e alive or dead. Real engines, on the other hand, treat each object as its own entity and reports the object's movement and behavior back to the game engine. The advantage of the real engines is that they can simulate reality better since they allow for randomness to occur, as in the real world. In addition, the decision load is distributed between every single object so a single HI won't have to calculate everything.

In addition to these two main types, gaming engines are also divided into three groups based on their complexity. The first, and most complex, is usually referred to as *roll-your-own* gaming engine. SCI and SCUMM are examples of these types of engines that are developed from scratch by the company that also develops the actual game. This means that they use publicly available Application Programming Interfaces like DirectX and

openGL in combination with commercial and open source libraries to build their own engines. These libraries can be physics libraries like Havok and ODE, scene graph libraries like OpenSceneGraph, and GUI libraries like AnTweakBar. The biggest advantage for this type of engines is flexibility. The programmers can choose the components themselves and implement them in whatever way they see fit. On the other hand, roll-your-own engines take the longest amount of time to build and a lot of programming is needed to make the libraries work together.

Mid-level gaming engines are usually referred to *mostly ready* gaming engines. Most engines are in this category with examples like id Tech, the Unreal Engine, and jMonkeyEngine. These engines are easier to use because they contain a lot of default libraries like rendering, GUI, physics, and input, so the user won't need to do any programming to get the engine started. However, these engines still need a bit of programming to get them up and running into a complete game. Typically, both scripting and low level coding is required to get a real game working. These engines are optimized for the general case which means that they don't offer the same amount of flexibility as low level engines. However, since they contain a lot of libraries and built in functions, they usually provide a better result with less effort than roll-your-own engines.

Point and click engines are the highest level gaming engines and are becoming more and more common these days. They include a full tool chain which makes the user able to point and click its way to create a playable game. GameMaker, Torque Game Builder, and Unity3D are examples of these engines which are built to be as user friendly as possible. The problem with these engines is that they are very limiting and they often do only one or two types of gaming genres, or one or two types of graphics modes. In other words, if you want all design possibilities, these engines don't suffice. However, they allow the user to work quickly and create a playable game without too much effort. This is a typical choice for game designers without experience.

2.3 Software Discussion

Baba et al. [9] give an overview of the most important parameters for gaming engines in the implementation of computer games, including edutainment purposes. We will refer to these parameters, like cost, features, ease of use, support, skill required, learning curve, interface, and plugins when choosing what software to use for this project. In addition to these parameters, we will also consider the possibility to implement the Java games from Cyberlab. These games are an important part of the game environment and it's a criteria that they can be used, either as built-in or stand alone applications.

2.3.1 jMonkey Engine

jMonkey Engine (jME) [10] is a scene-graph based graphics API, built to fulfill the lack of full featured graphics engines written in Java. It is written in Java and is thus a true, or object based, gaming engine. The scene graph allows for organization of the game objects in a tree structure, where a parent node can contain any number of children nodes, but a child only contains one parent. This scene-graph structure allows for quick rendering of complex scenes as you can choose which parent nodes you want rendered, and discard the rest.

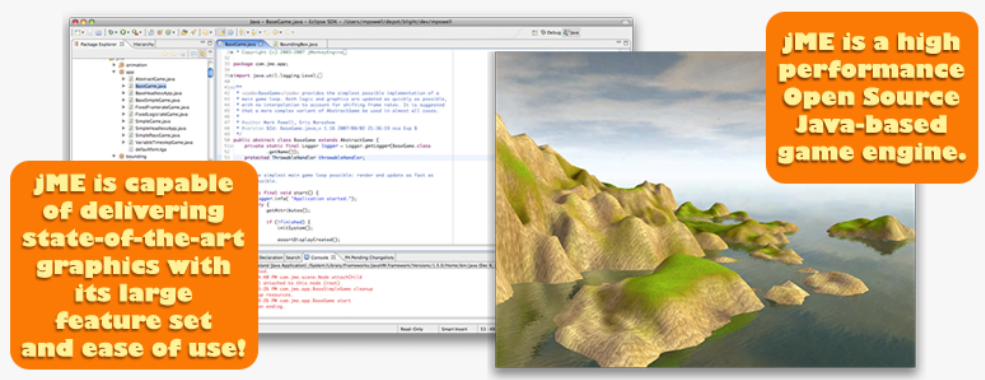

Figure 2.5: Screenshot from jME. Picture from [10]

As mentioned earlier, the jME is a mid level engine, and requires Java programming skills to get the games running. It has a lot of built in modules and plugins like physics, lighting, texture system, camera system, sound, and interface. JME is an open source project which means that it is free of cost, but has a lack of certified documentation. The lack of documentation is a big downside by choosing an open source application. But since it is an

open source software there also exists a lot of user communities with several discussion forums where it is easy to get help, download tutorials etc.

Based on the test we did with the jME, we conclude that it has its advantages at cost, features, support, interface, and plugins. It is a big plus that anyone can extend the source code or develop their own plugins to fix their given problem. Implementation of the games from Cyberlab can be done in an easy manner since both the games and jME are written in Java. However, jME is a mid-level engine and requires extended programming skills. The learning curve is steep and the threshold of mastering is high. As jME is just a gaming engine and not a 3D modeling software, there is no support for modeling "by hand". This is a big vote against jME considering that our virtual world contains a lot of arbitrary shapes and textures which need to be modeled. This means that we might be forced to use some 3D modeling software in addition to jME for the hardcore modeling.

2.3.2 Blender

The Blender modeling software [11] [12] is unique because it combines a modeler, a renderer, a video editing application, and a gaming engine. Its gaming engine is the first to allow creation of complete games without any programming. Through its point-and-click GUI, users with no programming experience can create playable games in an easy manner. The GUI is a graphical layer that visualizes how the game logic bricks work together. In other gaming engines, the user has to make a script telling the engine to attach a controller to the chosen object, for example. In Blender this is done by choosing type of controller and what actions it should perform from a drop down menu. Blender also supports C, C++ and Python scripting so an experienced developer is able to expand the game logic by any features needed. In other words, the Blender Gaming Engine (BGE) is a combination of a high- and mid-level gaming engine, which makes the engine suitable for all kinds of users.

Blender is an open source modeling software developed in C, but the gaming engine is written from scratch in C++. Since C++ is an object-oriented programming language, BGE is a true gaming engine where every object acts as its own entity. The fact that BGE is written in C++ will give rise to a small complication regarding the implementation of the edutainment games, as it is non-trivial to integrate C++ and Java. There has been numerous attempts to write a Java API in Blender, but developers have come to the conclusion that it will be too much effort for a small gain. Thus it is impossible to embed the edutainment games in the Blender model. A smart solution to this problem is to write a Python script that opens a web browser with

the respective URLs.

We used Blender in an earlier project, and based on this we conclude that the main benefits of Blender are cost, features, ease of use, learning curve, support, interface, and plugins. Since Blender is both a 3D modeling software and a gaming engine with all preferable features, it won't be necessary to use additional softwares to get the game up and running.

2.3.3 Project Wonderland

Project Wonderland [13] is a newly started project (august 2008) from Sun Microsystems. It is a Java based toolkit for creating collaborative 3D virtual worlds with multiple users and can be compared to the more known 2nd Life [14]. Within the virtual worlds, users can communicate with immersive audio, share desktop, applications, and documents, and conduct real business through avatars.

Project Wonderland is not a gaming engine, but a framework for building virtual worlds powered by jME, and share the same options in regard to user generated content as the jME. Since Project Wonderland is a relatively new project it is continuously updated with features and plugins from different developers, and the potential here is huge. The project is open source and also free of cost, but the development of the framework is lower level than its engine, jME.

Based on the test with Project Wonderland we conclude that it shares the same advantages and disadvantages as jME. In addition, it has great potential in making a virtual Gløshaugen campus where multiple users visiting the campus can communicate and share information and opinions about the university. However, this framework requires the player to create a Project Wonderland account to be able to play. This is a big disadvantage since our game is meant to be an easy plug-and-play game. Project Wonderland is Java based and low level, so the required skill for using this option is much higher than the three other alternatives.

2.3.4 Autodesk ImageModeler

ImageModeler [15] was suggested as an alternative way to create the photo-realistic virtual world without the use of a gaming engine. This is a licensed, image based modeling software developed by Autodesk, where you can create 3D models from 2D digital images or panoramas. It creates a 100% image based virtual world [16] by stitching, blending and mapping a series of photographs into a spherical panorama.

As this is a licensed software with protected source code, it's not possible to extend the usage of this program beyond the limits set by Autodesk. ImageModeler works good for its cause, but its lack of support for user generated content makes it a poor choice in this setting.

2.3.5 Conclusion

Referring to the parameters of a good gaming engine, we conclude that jME and Blender stands out as the two best options for this project. The biggest difference between the two is that Blender has a high level GUI where it is easy, efficient and time saving to create basic logics etc. By using jME much time will be spent to program every little bit of logic, but in Blender this can be done by a few mouse clicks. These time saving factors, combined with some prior experience with Blender, means that less time will go to waste to learn the basics of the software and more time can be spent developing the actual game. Thus, we conclude that Blender is the best choice of software in this thesis.

2.4 Blender Gaming Engine

Blender is a complete development tool for interactive worlds, including a gaming engine to play the worlds. Its gaming engine is a framework with a collection of modules for interactive purposes like physics, graphics, logic, sound and networking. Functionally the gaming engine processes virtual reality, consisting of content and behaviors, in realtime. By content, we mean the world itself with buildings etc, and the behavior consists of physics, animation and logic. The next subsections explain the most important BGE features used in this thesis.

2.4.1 Game Objects

The objects, or elements, of the world are called *Game Objects*. They behave autonomously through a set of tools called *Logic Bricks* and *Properties*, which are controlled by the *Logic Buttons*. The Logic Buttons are separated in two parts: the left part containing global settings for the Game Objects, and the right containing the local settings, fig. 2.6.

Figure 2.6: Screenshot of the logics GUI.

The most important global setting is whether the object should be calculated with the built-in physics. This is done by selecting whether or not the object is an actor. If the object is set as an actor, there are several ways the physics engine can treat it. A static object acts as surroundings for a level. These objects are movable with LogicBricks, and dynamic objects will react on collision with them. Dynamic objects follow the laws of physics, like falling, bouncing and colliding. Rigid body objects enables the use of more advanced physics. This makes it possible for spheres to roll, and other shapes will tip over, tumble etc. The opposite of a rigid body is a soft body. With this option selected, the physics engine treats the object as a "jelly object" so when it collides with another object, it gets deformed. In addition, if an object is defined as an actor, several physical properties can be edited, like mass, radius, and friction.

2.4.2 Properties

The properties of a Game Object can carry values describing attributes of the object, similar to local variables in a programming language. Since the properties are local variables, no other objects can access these properties, but it is possible to copy the properties or send them to other objects using messages. There are five different types of properties. Boolean, integer, float, and string are the same as in programming languages. The fifth property is timer, which is updated with the actual game time in seconds. An example of the boolean property is to define damaging objects. I.e. an enemy in the game has a true boolean property called "enemy", and if a character detects this property in a collision it looses health.

2.4.3 Sensors

There are three types of logic bricks called *sensors*, *controllers* and *actuators*. Sensors act like real senses of a life form; they react on key presses, collisions, contact with other materials, timer events, or values of properties. The internal state of a sensor can only take boolean values, true or false. This state value is updated at each frame based on the external conditions that the sensor monitors, i.e. keyboard key press/release or mouse movement. At the start of each frame, BGE runs through all the enabled sensors and updates their internal state, and the sensor state remains unchanged through the rest of the frame processing. Transitions between true and false states for the sensors are the primary causes for triggering controllers. The directions of the transition doesn't matter for triggering. Either way, false/true or true/false, the controller is executed. The controller is not executed instantly, but only when all enabled sensors have been updated. Even if multiple sensors are triggering the same controller, the triggered controller is executed only once.

The internal state transition of a sensor triggers all the active controllers attached to it. A controller that is executed will usually check the internal state of the sensor and take appropriate action. The logic controllers evaluate whether or not the controllers should be executed using AND, OR or XOR ports with the state of all the sensors connected to them as input values. This means that the logic controllers do not use the transition information.

As mentioned earlier, transitions between sensor states is the primary cause for triggering controllers, but the controllers may also be triggered at other frames. By using the pulse mode tool, the user may postpone the triggering for any number of frames after the transition. This is widely used in first person shooting games when enabling burst fire. Then the gun shoots three bullets, and if you press the shooting button at any time during the

three shots, the gun won't respond until the first three bullets are shot.

2.4.4 Controllers

Controllers act as the brain for the game logic. They collect events from the sensors and translate them to a result. This includes everything from basic decisions like connecting two or more inputs, to complex Python scripts carrying artificial intelligence. The basic decisions are made by logic ports (AND, OR, NAND, NOR, XOR and XNOR). More complex controllers are the Expression controller and Python controller. The Python controller is the most powerful controller in the gaming engine. This will be extensively used in this project, especially to apply video textures.

2.4.5 Actuators

Actuators are the executing Logic Bricks, and can be compared to muscles. Depending on the type or purpose of an actuator, it can act differently on a pulse. Some inputs are more like keys and get released when no pulse arrives, and some act like a switch which needs to be switched off by a pulse carrying a negative value.

The most basic actuator is the Motion actuator which, as the name indicates, is meant to move objects. Movement can be affected in two ways by this actuator depending if the object is dynamic or non-dynamic. For non-dynamic objects the actuator can move (Loc) and rotate (Rot) the object in x, y and z direction. The values for the Loc and Rot span from -10000 to 10000, and indicates how many units the object move along the respective axis per activation. Dynamic objects have more physics parameters, and the motion is controlled by location, rotation, force, torque, linear velocity, angular velocity and damping. All these parameters can be set to follow either the global or local axis of the objects.

To avoid problems with objects going faster and faster, or to allow more control of the game physics, there is a motion actuator called Servo Control. This variant of the motion actuator allows control of speed with force. The control is a PID controller (Proportional, Integral, Derivate), which means that the force is automatically adapted to achieve the target speed. All the parameters of the Servo actuator are configurable which makes it possible to simulate motion styles like anisotropic friction, flying and sliding.

There are a lot of other actuators available as well, such as Constraint, Distance, Orientation, Force Field, IPO, Sound and Edit Object. In this project we will primarily use the Motion, IPO, and Sound Actuators, for more info on the others, see [11].

2.4.6 The Blender Laws of Physics

The physics engine in Blender is called "Bullet" and is written by Erwin Coumans who is also one of the original authors of the BGE. This engine calculates the physical behaviors of the objects like falling, forces, and collisions. It also features a 3D Collision Detection and Rigid Body Dynamics Library for games, including Playstation 3. The engine runs many aspects of a game like ray tracing, near- and collision sensors, and constraints.

As mentioned earlier, all objects with the Dynamic or Rigid Body option set will be evaluated using the physics laws as defined by the physics engine. The key property for a dynamic object is its mass. Gravity, forces and impulses only work on objects with mass. The default value of gravity is 9.81, so if you use one Blender unit as 1 m and the mass of 1 as 1 kg, the objects will react almost as in the real world. Damping is often use to simulate air drag and maximum speed as it decreases the velocity in percent per second.

2.4.7 Sound player

Blender also has an integrated sound player to apply sounds in the game. Currently only .wav files are supported, but .mp3 and .ogg support is under development and will be available in Blender 2.5. In the sound panel, figure 2.7, you can see the name of the sound objects, edit them, and load new sound objects. Besides playing the sound objects, you can also edit their pitch and loop, and play them in "ping pong" mode which plays the sound forwards, then backwards in a loop.

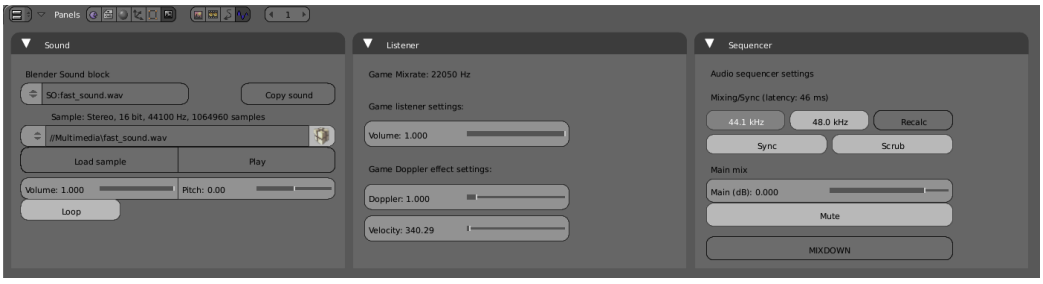

Figure 2.7: Screenshot of Blenders sound player

To get a more realistic sound picture the gaming engine also features calculation of 3D sound. This means the volume of the sound depends on the distance and position between the sound source and the listener. The calculations do not include advanced audio parameters like reverberation time in materials, absorption etc, but it gives the sound a more natural

propagation. To edit the relationship between gain and distance, a slider lets you set the sound attenuation. For example, if a sound passes by the camera, the scaling factor determines how much the sound will gain if it comes towards the camera and how much it will diminish if it goes away. In Blender 2.5 the Doppler effect will also be supported, so the frequency rises as the sound source gets closer to the listener and lowers as the sound distance to the source increases.

2.4.8 Game Engine Python

As mentioned earlier, the core of Blender is written in C/C++ and supports scripting in Python. In Blender there are two incarnations of the Python integration: The Blender Python, meant for extending Blender and its modeling and animation tools, and the BGE Python meant to be used with real time content. To write Python scripts there is a built in text editor in Blender. Here you can load external Python scripts, write new scripts and it also features practical tools like syntax highlighting and line numbering.

Through the BGE Python you can influence LogicBricks by changing their parameters and how events react when triggered by the Logic Bricks. The scripts are activated by objects through their Sensors and the python Controller, as shown in figure 2.8. Built in functions in the GameLogic module lets you access Actuators, Sensors and Controllers, and manipulate the objects in the game. For complete documentation of the Blender Python API, see [23].

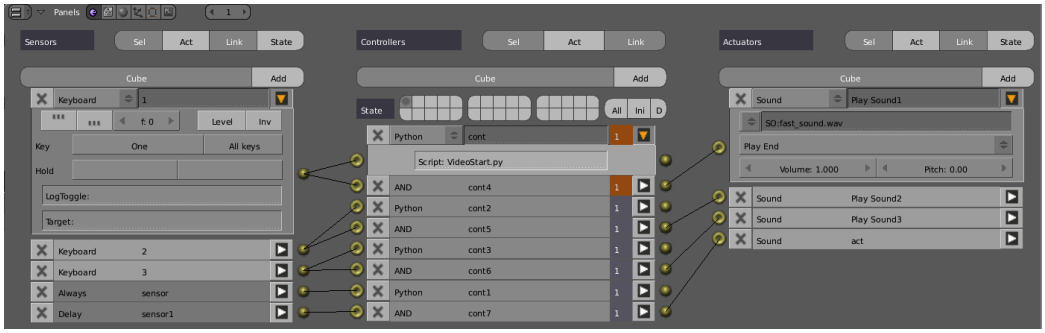

Figure 2.8: Screenshot of BGE Python logics.

Design

In this section we give an overview of the concept and design process of "Student Quest - A First Person Student Game"[1]. Furthermore, we will present several potential features that can be implemented to increase the gaming experience.

3.1 Concept

The scope of this thesis is to create the start of a serious game where high school students can visit a virtual NTNU campus to acquire information about the different studies at NTNU. By placing a serious gaming experience inside the virtual world, the goal is to have an exciting, yet informative, environment. The virtual environment is made up of two different scenes. The main scene is situated inside the hallway of the main building at NTNU Gløshaugen campus, and is meant to serve as an architectural walkthrough. Consequently, the primary focus is sophisticated graphics to make it look as in real life, figure 3.1. To make the game play realistic, physics are as in reality including movement with 6 degrees of freedom (forward, backward, rotate right, rotate left, tilt up and tilt down), gravity, and collision.

On the 2nd floor in the hallway there are 5 doors which lead to the second scene, the rooms which represent different lines of study. These rooms are not present in the actual building and does not hold the same graphical details as the hallway. When entering the room the first object the player notice is the arcade machine where edutainment games may be activated. This is placed in the middle of the room to serve as the primary focus point encouraging the player to try the games. On the side walls there are images illustrating both social and educational aspects of the studies at NTNU. Behind the arcade machine one can see parts of the video texture menu where both interviews and commercials can be played. This is intentionally covered up by the arcade machine to make the player move past the machine and be encouraged to explore the whole room.

Our focus has been to create a template for these rooms where content easily may be replaced in future work. The Electronics room, figure 3.2, is

[1]The title is inspired by Sierra's "King's Quest" series, and the subtitle is a reformulation of the popular video game genre "First Person Shooter Game".

Figure 3.1: Comparison of a real photo (top) and the model (bottom).

the only complete room in this thesis, with pictures, interview, edutainment games and other useful information. Note that the pictures, movies etc. serve as place-holders in the template. To create a new room one can simply copy the .blend file and change the image links on the textures, as the 4 other rooms are examples of.

Figure 3.2: Screenshot of the Electronics room.

The edutainment games are developed by Cyberlab, a company springing from NTNU. They have specialized in making learning games regarding math, physics, cybernetics, electronics etc., and some of them are used in courses at NTNU today. These games are a more entertaining part of the environment, and are meant to work as "appetizers" for students who are curious of what engineering and technology actually is. As an example we use the "hover train" game where the point is to navigate a hover train from Oslo to Trondheim. This is done by inducing different amounts of current in the rails so the train stays "floating" and doesn't jump off or crash down in its tracks. The player thus gets the idea of how a hover train works and how important it is with accurate regulation. These are all topics covered both in cybernetics-, physics- and electronics courses.

3.2 Potential Features

To fulfill the potential of the serious games genre, the game has to be both interesting, playable, enriching, enjoyable and entertaining. The potential features suggested next may contribute to this. Note that these features are not implemented in this thesis due to the time limit. We describe the

implementation of the interactive menu and sound zones in detail for implementation in future work.

3.2.1 Interactive Menu

To make the user feel like a part of the environment, the game needs some kind of interaction features. An interactive menu, as in figure 3.3, will provide this interaction. The menu is made up of two elements. First, the question menu on the left side. Here we place different "frequently asked questions" divided into several subjects such as studies, sports, fraternities etc. By moving the mouse cursor and selecting the category, the users may select what questions they want answered.

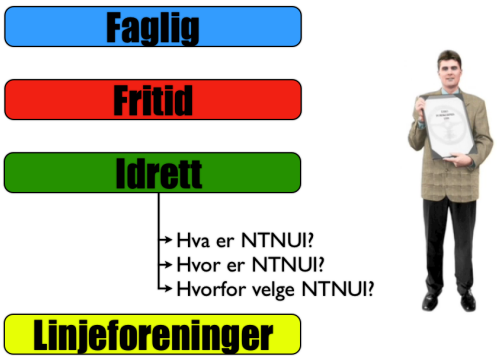

Figure 3.3: Suggestion for layout of the interactive menu.

On the right side, there is a movie file showing a person with monotone behavior. Once a question is clicked, this video texture is swapped with a movie stream where the person answers the question, and finally, the stand-by texture is applied again. Through this menu the user gets the important feeling of feedback. This contributes as a stickiness factor at the same time as the user acquires useful information. Technically this can be implemented in the same manner as the other video textures with the same scripts as in Appendix A.1. The question menu can be implemented using the same IPO techniques used in the menu.blend file, except that instead of opening a new scene when a subject is selected, the questions will become visible as a drop down menu.

3.2.2 Sound Zones

When a user enters the room it is unlikely that he or she is patient enough to listen to a 30 seconds long audio clip describing all the features inside

the room. By dividing the room into sound zones, the user is only provided information relevant to the current zone. As the player moves inside a zone near i.e. the arcade machine, a sound track telling the user about this will be played. When the user moves outside a zone the track will stop, and another one will be activated as the player moves to the next zone. To implement this, assign a property to the floor covering the area where the zones are situated. I.e. one property can be named "arcadesound" and as soon as a collision sensor detects this property, a Sound actuator will play the sound track. If the user moves the camera to another area, the current clip is stopped, and the one relevant to new zone is played.

3.2.3 Story Line

Narrativity is one of the main features that makes a game pass the Super Fun threshold, but this is not embedded in this game. A suggestion of a story line for this game is to make the player start out as a "freshman" on NTNU. As the player explores the building and enters different doors, the character gains experience points in different categories based on the choices made. The first door could be a study line like electronics or cybernetics. Inside these rooms several doors leading to different specialization categories. E.g. inside the Electronics room there are doors to acoustics, signal processing, circuit design etc. and inside these doors there are doors leading to multimedia signal-processing and other specialization areas. Finally, when graduating, the player can attend a virtual career day where the player is offered several jobs based on the choices made along the way. In this way, the player is presented to both the possible lines of study at the university and potential jobs. Thus, the game contains both a story line and character building which are the two most important features in the narrativity factor.

3.2.4 Social Experience

Social experience is pointed out by Wang et.al as the second fun factor that makes a game reach the Super Fun threshold. This feature was never the focus of this thesis, but we will propose it as a potential feature for future work. The creation of a social experience for the user depends on the game to support multiple users. Then each player can have their own avatar and meet other students in the same situation as themselves. It is often hard to initiate contact in real life, so meeting other potential students in a virtual world might ease this process. The players can initiate discussions regarding anything from choice of studies to the fear of starting a new life in Trondheim. In addition, the university could have students present on the virtual campus

24 hours a day so players may ask them questions and get more information about NTNU.

If this feature is to be implemented, Project Wonderland or 2nd Life are the best suited development tools. We recommend to use Project Wonderland since models made in Blender can be exported easily to the jME, which Project Wonderland is built on. By doing this, one would obtain the same aesthetic features while adding the social experience embedded in Project Wonderland. ”

Implementation

In this section we present the implementation of the game divided into three main parts; modeling, logics and installation wizard.

4.1 Modeling

We will not go into great detail of the modeling process since most of the modeling work in this thesis is done with great help from 3D artist Pablo Vazquez. Therefore, the following section only provides a quick summary of the most important techniques used. See [21] for more complete documentation of modeling techniques.

The initial step of the modeling process is to obtain a 2D basis of the surfaces. By using photos similar to figure 4.1 as background images such base is created with proper relative scaling. To expand this to 3D we extrude the faces and model difficult geometrical shapes by hand. The textures of the surfaces are made up by photographs that are automatically mapped onto the selected faces by UV-mapping, figure 4.2. Finally, we apply lighting to make the environment look more alive.

Figure 4.1: Screenshot illustrating modeling on top of a background image

There are built in functions in Blender that calculates ray tracing, shadow casting and reflections, but they are not activated in realtime rendering in the BGE because the calculations are too heavy for the engine to handle. Therefore we use a technique called *baking* [22] to capture the lighting, reflections, and shading in realtime. Shortly summarized, baking renders a

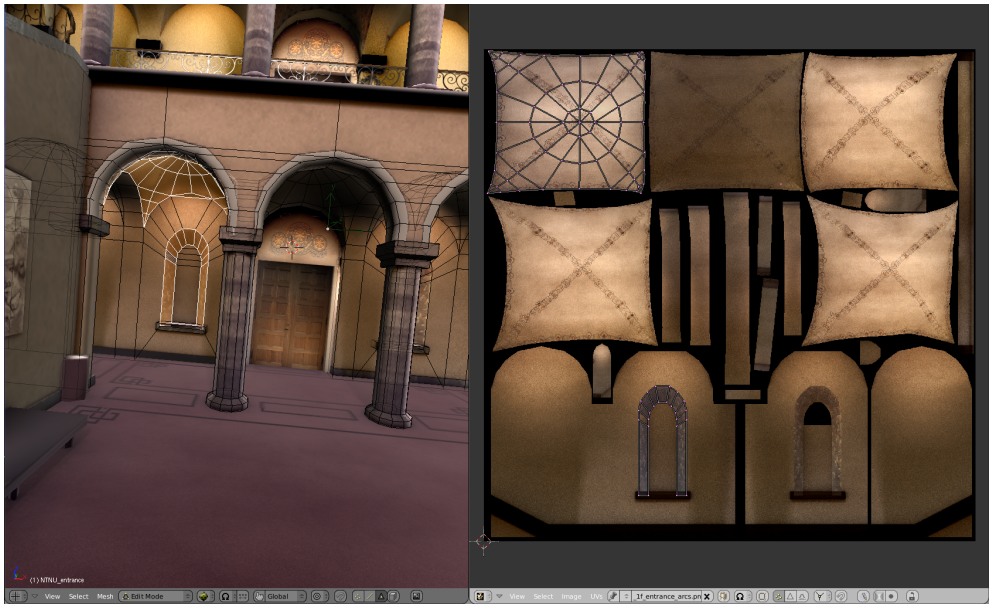

Figure 4.2: Screenshot showing UV mapping. Left side: Faces in the model where the textures is mapped. Right side: Areas of the texture image that is mapped to the faces.

selected viewpoint in the model and calculates how the faces look including lighting, ray tracing, shadowing and reflections. From this render we select which areas to use as textures for the faces in the model, and copy-paste the areas into one image for each object. Right side of figure 4.2 shows the texture faces for the "1st floor entrance" object. After doing several renders to cover every object in the model, we obtain the same lighting conditions etc. in realtime as in one single render. Finally, we UV-map these images onto their respective faces, and the complete model of the hallway looks like figure 4.3. The only drawback with baking is that the lighting is static, so if we use a moving light source the shadows etc. won't move accordingly.

Figure 4.3: Screenshot of the hallway in bird view.

4.2 Game logic

The game consists of 12 .blend files divided into three groups. 6 files are the different starting points in the hallway, one for each door and one initial. The only difference in these files is where the camera and empty has its initial position. In addition there are one .blend file for each room, and one for the menu. The game logics in the respective .blend files are implemented using both the high level GUI and scripting. In the next sections we go further into detail on the Logic Bricks used in the three .blend file groups; hallway, room and menu

4.2.1 Hallway

The hallway and rooms use the same movement logics, a camera to maintain the 1st person view and an *empty-object* to hold the logics. The camera movement can be controlled both by the mouse and the keyboard. There are two choices for the keyboard controllers, the arrow keys and w,a,s,d move forward/backward and rotate left/right, respectively. This is implemented by adding Keyboard sensors, And controllers and Motion actuators and choose which key to trigger the sensor and how many units the actuator shall move its object along the respective axis. I.e. the up-arrow will move the objects location (Loc) 0.1 units along the y-axis, and the left arrow will rotate (Rot) the object -0.1 units around the z-axis, see figure 4.4. For the mouse we make the mouse wheel work similar to the up- and down-arrow, but the rotation is

implemented by a python script, mouseMovement.py in Appendix A.1. By applying this script we get full rotation in all directions, as opposed to the keyboard controllers where we are unable to tilt the view up and down.

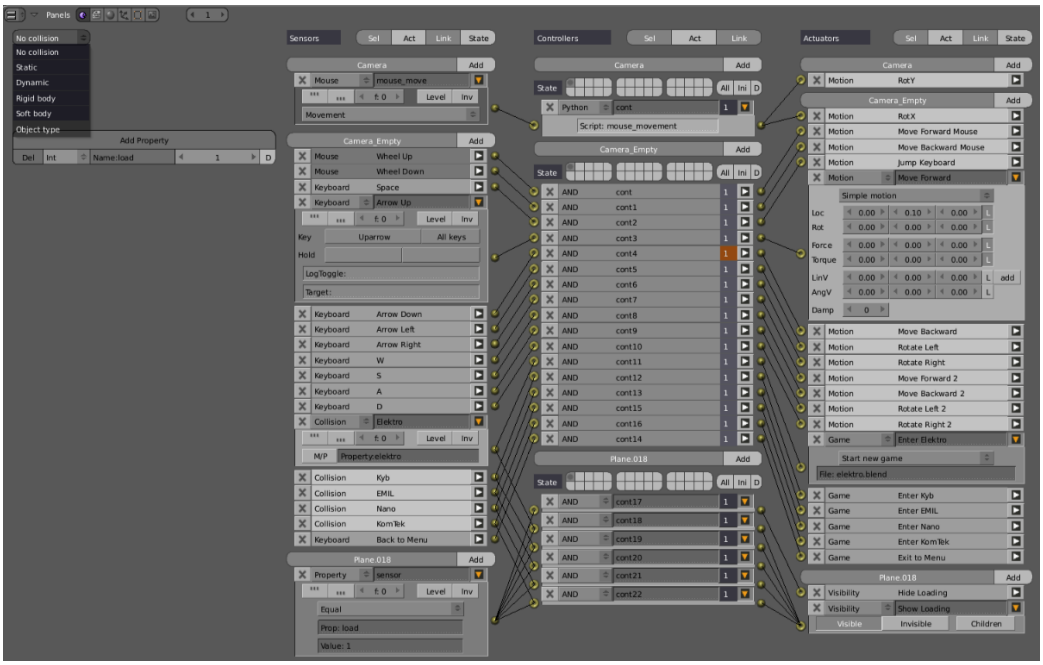

Figure 4.4: Overview of the hallway logics

If the player enters one of the rooms or wants to go back to the menu, the new .blend file is loaded with the Game actuator. Each door on the 2nd floor has an invisible collision plane surrounding it with a true boolean property. If the empty's collision sensors detect a collision with one of these properties, the game actuator opens the respective .blend file. The menu actuator is activated by pressing F1.

When the player either enters or exits a room, it takes a few seconds to load the new scene. To illustrate this loading process, a plane with an image showing "loading..." becomes visible as the Game actuator is activated. The plane is placed directly in front of the camera, and by making the camera a parent to the plane, the plane will move if the camera moves to always maintain its relative position to the camera. This plane has the property "load" which is an integer equal to 1. A sensor continuously checks for this property and if the property is 1, then the plane's Visibility actuator is set to "Invisible". But if the property is equal to 1 and a collision with the door properties is detected, the Visibility actuator is set to "Visible". Thus, the loading image is only visible to the player as the new .blend file is loaded.

4.2.2 Room

The rooms share the same movement logics as the hallway, but in addition there are five Python controllers, one on the arcade machine, three on the video screen, and one on the fraternity logo. As mentioned earlier, the Cyberlab learning games are written in Java so they can't be implemented in-game. Therefore a Python controller on the arcade machine activates the openWebbrowser.py script which opens the default web browser on the right URL. A similar controller opens the fraternity homepage if the player collides with the logo. The video texture scripts are quite complex, and are described next.

Video Texture Module

To play movie files inside BGE is not a standard feature in the current version (2.48). However, because of the open-source community, there are some super-users who have developed a python module called *VideoTexture* which provides all the functionality needed. This module is available in all the latest SVN builds and can be downloaded from [19]. The videoTexture module allows to manipulate textures during the game. Several sources for texture are possible, such as video files, image files, video capture etc. The principle is to identify an existing texture by object and name, then create a new texture with dynamic content and swap the two textures in the GPU. At the end, the new texture is deleted and the old texture restored. We will now briefly explain how the video texture scripts work, appendix A.1. For more detailed documentation on the VideoTexture module, see [20].

The first step is to create a Texture object. This is done in lines (1) - (4) by the functions getCurrentController() and getOwner() from the GameLogic module. Line (6) performs a check if the "video" attribute is true to make sure the texture is created only once. Line (7), performs a search for the right texture or material to apply the video texture on, by the materialID(obj, "IMvideo.png") function. This retrieves the object material which is using the "video.png" file as texture. The video texture object from the Texture class is then created in line (8). Note that the texture object is assigned to a GameLogic "video" attribute. The reason is that the texture object must be persistent across game scripts. A local variable would be deleted at the end of the script and the GPU texture deleted at the same time.

The next step is to create a source object from one of the possible sources available in VideoTexture, i.e. VideoFFMpeg for moving pictures, ImageFFMpeg for still pictures, or ImageRender for rendering from a camera. Since we want a video file as the source, we choose the VideoFFMpeg con-

structor. This constructor takes a file name as argument. Since this game is to work on several computers with different install paths, we use the GameLogic.expandPath() to build an absolute file name to avoid confusion with the location of the file. In lines (9) - (10) the file path of the movie gets stored in the "movie" variable, and a video source object created with the respective file. Finally the video playback is activated in line (11).

Since video playback is not a background process we also need to refresh the texture continuously. This is done in a separate script, videoUpdate.py, that runs on every frame and calls the refresh() method of the texture object.

VideoPause.py is used to switch between two or more sources on the video texture, which simply stops the current source and starts the new. This script is written by the user "ben2610", who is one of the main contributers to the videotexture module.

In addition to the functions used here, there are several others available to manipulate the video, like filtering, scaling, tuning the framerate etc. All the formats and codecs that FFMpeg supports are supported by the VideoTexture module, including AVI, Ogg, Xvid, and JPG. This module is not built-in in the current version (2.48), but will be included in 2.49.

4.2.3 Menu

In the game menu there are options for Play, Controllers, Credits, and Exit. Play and Exit starts the hallway.blend file and exits the game, respectively. Controllers and Credits are both in the same .blend file as the main menu, but are different scenes. All the options can be selected by moving the selector up and down with the arrow keys and pressing the Enter key.

The menu logics are implemented in the following manner; First, we assign a location (Loc) keyframe on each of the 4 options in the menu and place the selector next to the respective options. Play is set to frame 1, Controllers to frame 2 etc. The selector gets an integer property called "selection", with initial value equal to 1. To get this applied in the BGE we add an Always sensor connected to an And controller, connected to an IPO actuator. The actuator is set to "Property" and "Prop:selection". To make the up- and down-arrow keys active, we add four sensors, controllers, and actuators. Two of the sensors are Keyboard sensors, set to up- and down-arrow, and they are connected to their own And controller and Property actuator. The actuators are set to "Add", "Prop:selection" and "Value: +/-1". This means that if you press the up-arrow the selection property gets set to 2 instead of 1, and the selector will the move to its position in frame 2. In addition, there are two property sensors to change the value of the selection property if it is either 0 or 5. This means that if the selector is positioned at frame 1 and

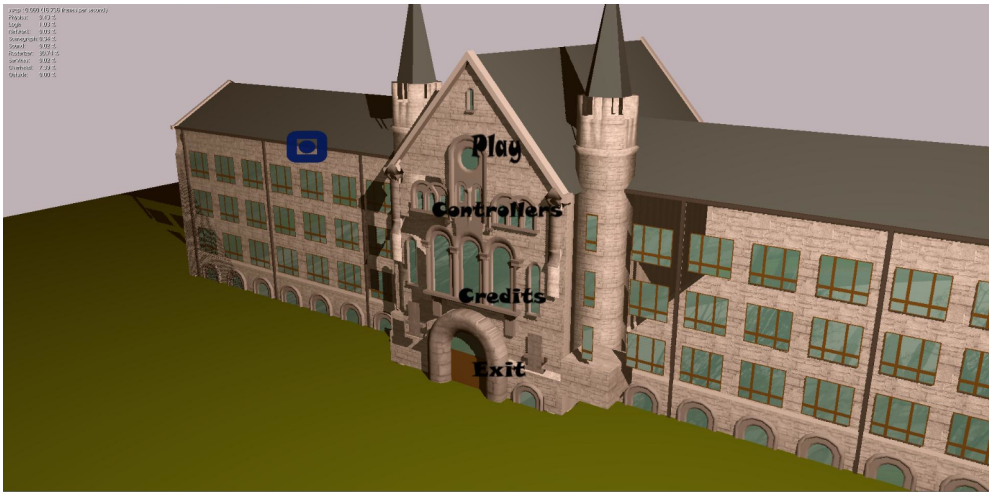

Figure 4.5: Screenshot of the game menu.

the up-arrow is pressed, selection is set to 4 instead of 0 so the selector goes to the bottom of the menu, and vice versa from bottom to top.

Finally, there are 4 more Property sensors, one for each menu option. These are connected to their And controller together with the Keyboard sensor for the Enter key, and further connected to their actuator. I.e. if the selection property is 2 and Enter is pressed, the Scene actuator is activated and starts the Controllers scene. The other three either starts a new game, shows the credits scene, or exits the game.

4.3 Installation Wizard

The game is aimed at all kinds of high school students, including those with less computer skills, and should follow the "plug and play" concept. Therefore, we create a windows standardized installation wizard which guides the user through the installation process step by step. The installation wizard is written in Inno Setup [24] which is an open source installer for Windows programs based on Pascal. Some of its key features are support of all windows versions including Vista, it creates one single setup.exe file for easy distribution, it has a standard Windows 2000/XP wizard interface, and it also creates an uninstallation file. In addition, there are several tutorials and examples included in the "help" file, so it is easy to use. The installation script is placed in Appendix A.2.

To make the game work, the user needs the game files, Blender and a Python compiler, so all these files need to be packed into the same installation. This is solved by splitting the installation in two parts, one for Blender and the game files and one for the Python installation. The Blender folder and Game folder are simply copied to the user defined installation path. To install the Python compiler, we make the installer run the Python.msi file which is a separate installation wizard created by Python. Thus, the only folders visible in the root of the installation folder are the Blender folder, the Game folder, the Python installation file, license and copyright files and the uninstall file. To ease the process even more, the installer places a shortcut on the desktop, called "Student Quest". In other words, all the user has to do is click the "setup.exe" file, click through the installation wizard, double click the desktop icon, and the game starts. This whole procedure should not take any longer than 2 minutes.

As of today, the installation file is approximately 560MB including Blender, the game, and Python. Only 40MB of this is Blender and Python, the rest are the game files. While installing, the user may choose between a High Quality (HQ) and a Low Quality (LQ) version. The game is the same and the model contains the same amount of polygons, but in LQ the texture folder is scaled down to half the size of the HQ version.

Discussion

5.1 Classifying our serious game

In this discussion we classify our game in the four dimensions explained in the theory section, Primary Educational Content, Primary Learning Principle, Target Age Group, and Platform. Furthermore, we point out what fun factors our game contains and lacks, and clarify what Enjoyment thresholds the game passes.

5.1.1 Primary Educational Content, Learning Content, Target Age Group and Platform

As mentioned in section 3.1 is the main purpose of this game to educate the player on studies available at NTNU. This is achieved by promoting the product, NTNU, through an exciting virtual world containing various information sources regarding each line of study at the university. These sources are made up of both "plain" informative material embedded to educate, such as the study plan, and fun content like the edutainment games and commercial movies. Furthermore, we can argue that the target players are potential "customers" in the sense of high school students. Based on this, the game's educational content is both academic and marketing. After playing the game, the user should primarily have gained information about the university, and learned something about the study area they have explored, i.e. through the edutainment games. Thus the marketing of NTNU is by far the most prevalent, and The Primary Educational Content is therefore Marketing. Note that this actually is the least prevalent educational content in the research which indicates that our game might be a new way of using serious games.

The gameplay of our game is split in two settings. When situated in the hallway, the player walks around and explores the building. As of now there are not much educational content in the hallway, so this part works primarily as a walkthrough. Secondly, when situated inside the room, the player explores and uses the different information sources. Most of the sources are primarily one-way, like the pictures and movies, but the edutainment games encourage to a more interactive way of learning. This part of the gameplay is more focused on cognitive problem solving as they engage the player both cognitively and creatively with brain teasers and games. The main focus of

the game is to explore the building and the rooms, and a broad scope of information is presented with a small amount of repetition. Thus the Primary Learning Principle is Knowledge Gain through Exploration. It is important to note that the advantage of serious games is not the educational content itself, but how it is presented. Presenting the content through exploration is a way of implementing information in a more exciting manner than listening to a keynote, or reading brochures.

As this game aims towards recruitment of high school students it is obvious that the Target Age Group, using the categorization from Ratan and Ritterfeld, is Middle and High School. Most serious games have elementary, middle and high school students as targets, and it is important to note that this differs significantly from entertainment games where the average player is 33 years old.

This game is developed on and for computers running Windows since the target player is an average high school student. On high schools in Norway there is an extensive use of Windows computers, so other operating systems are seldom used. However, it is a trivial task to compile the game for Mac OSx, Ubuntu and such since Blender is a cross platform software. Other platforms, like Playstation, Nintendo DS and Nintendo Wii are not considered relevant for this kind of game.

5.1.2 Extracting the fun factors

Technical Capacity and Game Design are two extremely important factors the game has to incorporate to be playable. Game logics, rules, elements and how they work together are typical examples of these kinds of factors. As the primary gameplay in this game is a walkthrough, the logics are based on reality. This means that the camera moves as in real life, with the same laws of physics such as gravity and collision, and is thus intuitive for the player. The controls are also meant to represent reality. The player can walk, run and look around using the keyboard and mouse. For experienced gamers this kind of control setting is common and easy to use, but for the unexperienced gamer there might be some initial complications. The arrow keys are usable for all players, but the mouse look needs a bit more experience to handle. Based on a highly unofficial test done on 10 persons with different background, only 1 person (a 48 years old professor) had severe problems with the movement. The other test subjects (16 - 28 years) had no problem navigating. Although some users may experience problems with the mouse look, the average high school student has some prior experience in computer games, and we therefore state that the controllers are sufficiently easy to use. Furthermore, the game takes an acceptable time to load (see section

5.2), and there are not numerous glitches. Thus, the game at least exceeds the Playability threshold.

Aesthetics is an important factor when it comes to enjoyability. It clearly gives the player a better gaming experience if the quality and sophistication of both visual and auditory content is pleasing. In our case, the graphics has been the top priority when modeling the game environment, and consequently the graphical presentation is of extremely high quality. All the patterns on the walls, arches, ceilings etc. are as in real life and give a high detail level even when the game is played on a 50 inch TV with 1080p resolution. Sound, is not nearly as present in the game, so adding a soundtrack could contribute in setting a certain mood within the game. The only soundtrack used in this game is the "welcome" track which is played whenever a room is entered. This gives the player a kind of feedback, and creates the important feeling of in-game presence. Based on this we state that this game features the majority of aesthetics factors and thus reaches the Enjoyability threshold, but does not exceed it. Note that an implementation of the sound zones will lift the aesthetics to the next level and probably send the game past the Enjoyability threshold.

Narrativity is a feature which this game lacks. As mentioned earlier the game is, as of today, a walkthrough with edutainment and multimedia content. A story line is a feature that probably would take the gaming experience to a higher level and encourage the player to explore the building further after one room is done. Social experience is the other key factor missing in regards to the Super Fun threshold. This is also a feature that could take the gaming experience to the next level, but creating a 2nd Life like application was never the goal of this thesis. If that would have been the case, Project Wonderland or 2nd Life would be a better software choice than Blender. Based on this, the game does not fulfill the requirements to pass the Super Fun threshold or even get close to it. However, by adding the potential features suggested earlier, the game would in fact contain both narrativity and social experience.

5.2 Hardware Requirements and Performance

To run this game, the user needs Windows XP operating system or later. As this game aims at all kinds of high school students, we have decided to develop it on, and for, windows platforms. Furthermore, the BGE runs slower on Mac OSx and it is actually not possible to open the hallway.blend file in Blender running in Mac OSx. It is recommended to have at least 2GHz processor, 1GB RAM and 600MB of free hard-disk space. The game will run on less hardware, but the framerate will decrease significantly to about 5 fps, and the user will experience long lags in the rendering.

To make the materials reflective during realtime rendering, we use real-time GLSL materials. This implementation takes advantage of the OpenGL Shading Language (GLSL). The game engine already supports GLSL shaders, but it also requires a graphics card and drivers that support it. The following cards typically support it:

- ATI Radeon 9x00, Xx00, X1x00, HD2x00, HD3x00 series and newer
- NVidia GeForce FX, 6x00, 7x00, 8x00, 9x00, GTX 2x0 and newer

Intel or VIA graphics cards, ATI Radeon cards older than the Xx00 series, and Nvidia Geforce cards older than the 6x00 series are unlikely to support the GLSL feature.

The graphics cards mentioned above are not state of the art and expensive hardware. The oldest are about 5 years and they should be standard in most stand-alone computers newer than 5 years. Laptops with an integrated graphics controller may have problems supporting this. The consequence is that some of the materials turn pink (default color), and some normal maps won't work. It is important to install the latest drivers as they might fix bugs and improve performance.

There are two noticeable performance bottlenecks. First, the game loading time due to huge amount of image data. When the user presses Play in the menu, the hallway.blend file starts loading. All the textures are also loaded at this point, which leads to a noticeable loading time. Since the original images are shot with a digital SLR camera and the baking textures are made from these, the native resolution of the textures are 2048x2048 pixels. Just inside the hallway there are 30 texture images at 5-7MB each. Experimental tests show that it takes about 30-35 seconds to load the game when the images are in native resolution, which is unacceptable. To compensate, we make two versions of the game, one high and one low quality as mentioned earlier. In the HQ version we optimize the texture size by scaling simple textures, such as the ceiling, down to 256x256 pixels and keep native

resolution on the rest. This results in 15 seconds loading time, which is acceptable for a game with this kind of graphics. In the low quality version all the textures are scaled to 256x256 pixels which results in a halving of the size of the texture folder, from 160MB in HQ to 80MB in LQ. The loading time is also significantly reduced, from 15 to 5 seconds. As seen in figure 5.1 it differs widely in the graphical experience in the HQ and LQ game. In LQ, it is not possible to see the patterns on the walls. and the borders on the floors are blurry. Note that the loading time is independent of the computer hardware as it is the gaming engine that loads the images. For computers below the requirements listed, it is recommended to install the LQ version. The game will render close to 60 FPS in LQ, compared to 5-15 FPS in HQ.

Figure 5.1: Comparison of HQ (left) and LQ versions (right). Top: Floor patterns. Bottom: Library arc.

The second bottleneck is due to collision detection. As the model is now, the physics engine checks for collisions for every vertex on each frame, which demands a huge amount of resources. On some computers, this is seen if the camera is turned quickly. Then the rendering lags and comes down to 0-5 FPS. This can be fixed by applying a collision mesh in the model. A collision mesh is simply a low polygon version of the original model, hence fewer vertices. For up to date gaming computers, the current collision detection should not give rise to noticeable problems. The collision mesh is not made in this thesis considering the time limit, but is simply pointed out for future

work.

Conclusion

The "Serious Games" genre is increasingly important with respect to education, training and social change. The concept of the genre is to serve as an entertainment frame where serious content can be embedded, and reach a wide span of audiences by building on "the native tongue" of the gaming generation. Serious games can be classified in four dimensions: Primary Educational Content, Primary Learning Principle, Target Age Group, and Platform. Serious games span a wide range of purposes and educational goals, with the Academic Education and Practicing Skills categories representing the vast majority of the games. This indicates that most serious games are "edutainment" games and do not fulfill the huge potential of the genre.

To decide if a game is fun or not, a five category clustering of fun factors can be used. "The Big 5 in Game Enjoyment" are Technical Capacity, Game Design, Aesthetics, Narrativity and Character Building, and Social Experience. Relative position ranking of these fun-factor categories imply that there are certain thresholds the game has to pass in order to be Playable, Entertaining, and ultimately Super Fun. If the game features good Technical Capacity and Game Design, the Playability threshold is passed. To pass the Enjoyability threshold, the game's Aesthetics has to be sophisticated and of high quality. Finally, if the game also contains good Narrativity and a Social Experience it might pass the Super Fun threshold and become extremely entertaining.

We have used the concept of serious gaming to create a game for the purpose of recruiting high school students to NTNU. The game is developed in Blender and is meant to be as realistic as possible, both in graphics and game play. In addition, we have created an easy installation wizard so the game is easy to use, even for users with moderate computer skills.

The game has Marketing as it's Primary Educational Content, Knowledge Gain through Exploration as Primary Learning Principle. The Target Age Group is Middle and High School, and it is developed for computer Platform. Marketing is the least prevalent Educational Content in serious games which indicates that our game might be a new way of using serious games.

Due to the Technical Capacity and overall Game Design it is clearly stated that the game passes the Playability threshold. Graphics was the top priority when modeling the game environment, so the detail level is high, the quality is high definition, and the graphics are sophisticated. In other words, the

Aesthetics are sufficient to reach the Enjoyability threshold, but not exceed it. Narrativity and Social Experience are two features that the game lacks. Consequently, we state that the game do not reach, or even come close to, the Super Fun threshold. However, if the potential features suggested here are implemented in future work, there is a good chance that the Super Fun threshold might be reached.

In summary, we have created a good start for a serious game with the purpose of recruiting high school students. The game's primary purpose is marketing NTNU and showing how exciting the studies here are. As of today, the game passes the Playability threshold, reaches the Enjoyability threshold, and by implementing the potential features suggested, the game has the potential to reach the Super Fun threshold and thus become an extremely entertaining serious game.

Appendix

A.1 Python Scripts

VideoStart.py:

```
import VideoTexture

contr = GameLogic.getCurrentController()
obj = contr.getOwner()
if not hasattr(GameLogic, 'video'):
matID = VideoTexture.materialID(obj, 'IMvideo.png')
GameLogic.video = VideoTexture.Texture(obj, matID)
GameLogic.sources = [None,None]
movie = GameLogic.expandPath('//Multimedia/Fast.m4v')
GameLogic.sources[0] = VideoTexture.VideoFFmpeg(movie)
movie = GameLogic.expandPath('//Multimedia/trailer_400p.ogg')
GameLogic.sources[1] = VideoTexture.VideoFFmpeg(movie)
GameLogic.current = 1
GameLogic.video.source = GameLogic.sources[GameLogic.current]
GameLogic.video.source.scale = True
GameLogic.video.source.flip = True
GameLogic.video.source.repeat = 2

if contr.getSensors()[0].positive:
GameLogic.video.source.play()
```

VideoPause.py:

```
cont = GameLogic.getCurrentController()
if hasattr(GameLogic, 'video'):
if cont.getSensors()[0].positive:
print "Here"
GameLogic.current = 1-GameLogic.current
GameLogic.video.source.stop()
GameLogic.video.source = GameLogic.sources[GameLogic.current]
GameLogic.video.source.play()
```

VideoUpdate.py:

```
if hasattr(GameLogic, 'video'):
  GameLogic.video.refresh(True)
```

OpenWebbrowser.py:

```
import webbrowser

webbrowser.open("http://www.pidstop.com/energispillet_v2_1/")
```

MouseMovement.py:

```
##################################################### ####
#
# MouseLook.py Blender 2.45
#
# Clark R Thames
# Released under Creative Commons Attribution License
#
# Tutorial for using MouseLook.py can be found at
#
# www.tutorialsforblender3D.com
#
##################################################### ####
# Import Modules
import Rasterizer

# Get Controller & Owner
cont = GameLogic.getCurrentController()
own = cont.getOwner()

# Sensors & Actuators
mouse_move = cont.getSensor("mouse_move")
RotX = cont.getActuator("RotX")
RotY = cont.getActuator("RotY")

# Values
Width = Rasterizer.getWindowWidth()
Height = Rasterizer.getWindowHeight()
Xpos = mouse_move.getXPosition()
Ypos = mouse_move.getYPosition()
Rot = own.getOrientation()[2][2]
```

```
########## Constraints ##########
#################################
XSensitivity = 0.0007
YSensitivity = 0.0007
TopMax = -0.9
LowMax = 0.9

# Get The Offset
OfX = -(Width/2-Xpos)*XSensitivity*-1
OfY = -(Height/2-Ypos)*YSensitivity*-1

RotX.setDRot(0,0,OfX,1)
RotY.setDRot(0,0,0,1)
if OfY < 0 and Rot > TopMax:
RotY.setDRot(OfY,0,0,1)
elif OfY > 0 and Rot < LowMax:
RotY.setDRot(OfY,0,0,1)

GameLogic.addActiveActuator(RotX,1)
GameLogic.addActiveActuator(RotY,1)
GameLogic.addActiveActuator(RotX,0)
GameLogic.addActiveActuator(RotY,0)

Rasterizer.setMousePosition(Width/2,Height/2)
```

A.2 Installation Wizard Script

```
[Setup]
AppName = Student Quest
AppVerName = Student Quest, Version 1.0
WindowVisible = yes
WindowStartMaximized = no
AppCopyright=Copyright (C) 2009 Norwgian University of Science and Technology - Q2S
DefaultDirName={pf}\Student Quest
DisableProgramGroupPage=yes
SetupIconFile = C:\\Source\Game high\Multimedia\logo.ico
OutputDir=C:\Desktop

[Types]
Name: "high"; Description: "High Quality"
```

```
Name: "low"; Description: "Low Quality"

[Components]
Name: "high"; Description: "High Quality Files"; Types: high; Flags: fixed
Name: "low"; Description: "Low Quality Files"; Types: low; Flags: fixed

[Files]
;Blender
Source: "C:\Documents and Settings\Håvard\Desktop\Source\Blender\*.*";
DestDir: "{app}\Blender"; Flags: recursesubdirs

;The "game" folder with subdirectories high quality
Source: "C:\Documents and Settings\Håvard\Desktop\Source\Game high\*.*";
DestDir: "{app}\Game"; Flags: recursesubdirs; Components: high

;The "game" folder with subdirectories low quality
Source: "C:\Documents and Settings\Håvard\Desktop\Source\Game low\*.*";
DestDir: "{app}\Game"; Flags: recursesubdirs; Components: low

;Python
Source: "C:\Documents and Settings\Håvard\Desktop\Source\python-2.5.4.msi";
DestDir: "{app}";

;License files
Source: "C:\Documents and Settings\Håvard\Desktop\Source\copyright.txt";
DestDir: "{app}"; Flags: isreadme
Source: "C:\Documents and Settings\Håvard\Desktop\Source\GPL-license.txt";
DestDir: "{app}";

[Run]
;Run Python installation
Filename: "msiexec.exe"; Parameters: "/i""{app}\python-2.5.4.msi"""

[Icons]
Name: "{commondesktop}\Student Quest"; Filename: "{app}\Game\play.exe" ;
IconFilename: "{app}\Game\Multimedia\logo.ico"
```

Bibliography

[1] RATAN R., RITTERFELD U.: *Classifying Serious Games*, Serious Games - Mechanisms and Effects, Chapter 2, 2009

[2] WANG H., SHEN C., R., RITTERFELD U.: *Enjoyment of Digital Games: What Makes Them Seriously Fun?*, Serious Games - Mechanisms and Effects, Chapter 3, 2009

[3] RITTERFELD U., WEBER R.: *Video Games for Entertainment and Education*, Playing Video Games - Motives, Responses and Consequences, pp. 399-413, 2006

[4] PRENSKY M.: *Digital game-based learning*, New York, NT: McGraw-Hill, 2006

[5] *http://www.theesa.com*, 2006

[6] GEE J. P..: *Good video games and good learning*, New York: Peter Lang, 2007

[7] KLIMMT C..: *Serious Games - Mechanisms and Effects*, Chapter 16, 2009

[8] WANG H., SINGHAL: *Serious Games - Mechanisms and Effects*, Chapter 17, 2009

[9] BABA A. S, et al: *An Overview of Parameters of Game Engine*, IEEE Multidisciplinary Engineering Education Magazine Sep. 3, vol 2, NO 3, pp. 10-12

[10] *http://www.jmonkeyengine.com*

[11] WARTMANN C., KAUPPI M.: *the Blender Gamekit 2nd Edition*, 2009

[12] *http://www.blender.org*

[13] *https://lg3d-wonderland.dev.java.net/*

[14] *http://www.secondlife.com*

[15] AUTODESK: *Autodesk ImageModeler, Key Features and Benefits*, 2009

[16] HIROSE M.: *Image-Based Virtual World Generation*, IEEE Multimedia Jan-Mar. 1997, vol 4, issue 1, pp. 27-33

[17] APPLE INC: *QuickTime VR*, 2005

[18] CHEN E.: *QuickTime VR - An Image Based Approach to Virtual Environment Navigation*, International Conference on Computer Graphics and Interactive Techniques 1995, pp. 29-38

[19] *http://www.graphicall.org*

[20] *http://wiki.blender.org/index.php/Dev:Source/GameEngine/2.49/VideoTexture*

[21] *http://wiki.blender.org/index.php/Doc:Manual*

[22] *http://wiki.blender.org/index.php/Doc:Manual/Render/Bake*

[23] *http://www.blender.org/documentation/248PythonDoc/*

[24] *http://www.innosetup.com/isinfo.php*