AALBORG UNIVERSITET

Det Teknisk-Naturvidenskabelige Basisår
Natuvidenskablig - Software
Strandvejen 12-14
Telefon 96 35 97 31
Fax 98 13 63 93
http://tnb.aau.dk

Title: Web based games: Asteroids

Theme: Reality of models

Project period:
P1, autumn semester 2007

Project group:
A216

Participants:

Frederik Højlund

Jens-Kristian Nielsen

Søren Møller Larsen

Jesper Kjeldgaard

Kenneth Madsen

Peter L. Hansen

Supervisors:
Ulrik Nyman
Søren Kerndrup

Copy's of report: 10

Report page count: 73

Attachments: 0

Report finished: 17. December 2007

Synopsis:

With the purpose of creating a web-based remake of Asteroids, we will research what is required to make a new version of the original game, and how we can offer more content than the original. Furthermore we will create a website that will contain our game and have potential of containing multiple games.
The technical analysis will focus on the selection of programming languages, databases, UML diagrams and other technologies needed in the process of creating the product.
Some aspects of the game will not be developed due to product demarcation. The game will be fully functional, but the possibility of upgrading the spaceship will not be developed. Our game will not apply to the hardcore audience, because the final features will be missing.

Contents

0.1 Preface

This report is the result of A216's P1 project at Aalborg University in 2007 with "Webbased games" as the main topic.

The report consists of four chapters.

1. Problem Analysis
2. Technical Analysis
3. Problem Solving
4. Concluding Part

Besides describing different types of games and problems concerning social behaviur, we have developed a working prototype of a game portal and a prototype of our remake of an asteroids arcade game. The web portal and game is accessible on the Internet at the URL: `http://asteroids.unif.dk/`. Citation references are located in the reference section at the end of the report, just before the appendixes and attachments. Words written in *itallic* is described further in the terminology section.

This report is written with the assumption that the reader has a basic knowledge of computers and computer programming.

Chapter

1

Problem Analysis

1.1 Introduction

Over the last decade the Internet is has become widely available in almost every danish home. Actually 83% of the danish population has access to the Internet. And 33 percent of those have been playing and/or downloading online games[23]. Because of the increasing number of people online a lot of activities moved onto the Internet, including shopping, gaming, movies etc. People are also using the internet as a main source for communication. And therefore there is now online communities for almost any topic imaginable. These things have made the world "smaller", and more interconnected, in the sense that everything is more or less connected via the internet.

This is also affecting the gaming society in several ways. Games are no longer limited to a single user or a single local area network (LAN) but can today be played across greater distances via the internet. This has resulted in the forming of massive online virtual worlds, where people meet, talk and play with each other.

1.1.1 Initiating problems

1. Determine if its possible to add new and modern features to the classic arcade game Asteroids.
2. Combine a "casual gamer" target audience with a more "hardcore" one.
3. Analyse the different stakholders in a webbased game project.

1.1.2 Approach

The report consists of four main parts. We start out with a problem analysis where we cover the subjects we need to answer our initiating problem. The chapter concludes into a problem description and limitation. When the problem has been described in full, we move on to the technical analysis. This chapter contains basic descriptions of the technologies and programming languages we use in the implementation of our product. In the end of this chapter we describe our products in a non-technical form, like gameplay, rules and features. Thereafter is the problem solving section where we describe how we implement our product. The problem solving chapter furthermore includes a implementation demarcation that covers the features we did not have the time to implement. In the fourth and last chapter we conclude our project by looking at it from different perspectives.

1.2 Video games

In this section we describes the main aspects of video games, video game platforms and arcade games. It closes with a rough description of the most played video game genres.

1.2.1 Video games and platforms

In this section we cover what a video game generally is and on what kinds of platforms they are played. It closes with a brief summary of the history of video games.

The source of the following section is [42], unless another source is noted.

The term video game covers many different types of games, that are played on many different types of platforms. Video games are generally shown on some sort of a display device, that can be interacted with through a user interface. Originally the word "video"[14] refers to raster display devices [39], which is a rectangular grid of colored pixels also known as a bitmap, but the term "video" in video game actually refers to any type of display device used. It is also quite common that games make use of the platforms audio devices. The platform is the system on which the game is played. It can pretty much be any digital system, such as cell phones, PDAs, even digital watches! However the most common platforms today are personal computers (PC) and consoles. A console is typically plugged into a TV-screen, but the word also covers hand-held consoles like the Gameboy. Dedicated platforms that only support a single game, called arcade games, also exist, but we'll get back to that in the next section.

To reach out to a wider audience, games are often developed for more than one platform. The main difference between PC and console platforms is that the the consoles have a very distinct user interfaces. Video games for PCs mostly use the mouse and keyboard, and in some cases a joystick. Other platform such as consoles and arcade games mostly use a hand held controller or a joystick. The PC user interface is superior to the console user interface when it comes to handling more complicated and feature rich games. But the console interface has the advantages of being more intuitive and also giving the player a better sense of movement in the game.

If a game has good sales on PC that does not automatically ensure success for all platforms. Hereby meaning that what's popular in games for PC might not be popular on consoles and vice versa. This is due to the differences in the user interface but also the target audience which we will cover later.

The first video games consisted of many different types of display devices and user interfaces. The first video game ever made was the "Carthod Ray Tube Amusement Device" created by Thomas T. Goldsmith and Estle Ray Mann [43]. The cathod ray tube amusement device was basically an analog device,

which allowed the user to control a vector drawn dot on a screen to simulate a missile being fired at targets represented by drawings fixed to the screen, like on a radar screen. Other games were also created such as the NIMROD from 1951, which used a panel of lights to play the game of NIM [2], the OXO for the EDSAC from 1952, which used a graphical display to play tic-tac-toe [16], and "Tennis for Two" from 1958, which used an oscilloscope to display the side view of a tennis court [43]. The first commercially sold, coin operated video game was Computer Space from 1971, which generated a video signal, that was displayed through a regular television. In 1972 it was soon followed by the Magnavox Odyssey which was the very first home console [43]. They were then followed by Atari's Pong, first the arcade version from 1972 and then the home console version in 1975. Pong is based on the sport of table tennis (or "ping pong"), and named after the sound generated by the circuitry when the ball is hit. The huge commercial success of the arcade and home consoles versions of Pong spawned a large number of Pong-clones and caused other companies to develop their own systems, and thus the video game industry was born [18].

1.2.2 Arcade Games

In this section we describe what an arcade games is, what the platform consist of and briefly how it works. We also comment on their popularity in the 80's and early 90's. What have become of the arcade genre and how do people play them today.

The source of the following section is source [32], unless another source is noted.

An arcade game is a coin operated video game designed for entertainment. The arcade games hardware resides in a cabinet, which is usually wired after the *JAMMA-wirering* standard.
It consists of the following:

- a monitor on which the game is displayed using either raster or vector graphics, where raster is most common. Standard resolution is between 262.5 and 315 vertical lines, depending on the refresh rate (usually between 50 and 60 Hz)
- a printed circuit board (the hardware on which the game is actually run)
- a power supply
- a sign above the screen displaying the title of the game
- a border around the monitor that may contain instructions or artwork
- a control panel where the joysticks and buttons are placed
- and finally the coin toss, coin return and coin box, which allows exchange of money or tokens between the consumer and the machine

An arcade game may vary from this, if it has any special controls or such, this just the usual layout[31].
The first popular "arcade games" were early amusement park midway games such as shooting galleries, ball toss games, and the earliest coin-operated machines, such as those which claim to tell a person their fortune or played mechanical music. The old midways of 1920s-era amusement parks (such as Coney Island in New York) provided the inspiration and atmosphere of later arcade games. We have already mentioned the first electronic arcade games, Computer Space and Pong from the early 70's, that started the video game industry. Video game arcades sprang up in shopping malls, and small "corner arcades" appeared in restaurants, grocery stores, bars and movie theaters all over the United States and other countries during the late 70s and early 80s. Games such as Space Invaders(1978), Galaxian(1979), Pac-Man(1980), Battlezone(1980), and Donkey Kong(1981) were especially popular. By the late 80s, the arcade video game trend was beginning to fade due to the advances in home video game console technology. Arcade video games experienced a resurgence with the advent of two-player fighting games such as Street Fighter II(1991), Mortal Combat(1992), Fatal Fury(1992), Killer Instinct(1994), and King of Fighters(1994). By 1996, home video game consoles and computers with 3D accelerator cards had reached technological superiority to the obsolete arcade equipment. The arcade games advantage over previous generations of home systems was their ability to customize and use the latest graphics and sound chips, much as PC games of today do. The golden age of of the arcade games was over.

Arcade games are usually very short and simple, which is both the strength and weakness of the genre. Due to the arcade games simplicity it is very easy to grasp and rise fast trough levels, but it takes many tries to truly master, and thus many coins. But it does not feature in-depth gameplay and strong storyline. Many independent developers are now producing games in the arcade genre that are designed specifically for use on the internet. These games are usually designed with Flash/Java/DHTML and run directly in web-browsers.

1.2.3 Game genres

This section briefly outlines the five most popular video game genres. We have included this section to create a basic understanding of the different game genres which is required to understand some of the sections later in the problem analysis.

When looking into the different aspects of video gaming like addiction, trends and target audiences, it is important to acknowledge that there are many distinct game genres. The five most played genres in general are Massive Multiplayer Online Role-Playing Games(MMORPG), Life Simulation, First-Person Shooter(FPS), Sports and Strategy. We now describe the main characteristics of these different genres.

1.2.3.1 MMORPG

MMORPG is a genre that has it's roots in ordinary Role-Playing Games (RPG). The difference between the two genres is that MMORPG is based on being played online. In a MMORPG the player controls a character in some kind of sci-fi or fantasy world. The main idea behind a MMORPG is that players log on to an online server and joins thousands of other players all around the globe in the same vast online world. In this online world the players can interact with each other by creating groups and battling other groups or non-player characters (characters controlled by the game). This sense of socializing with other players combined with the often endless possibilities to enhance and evolve a character, resolves into a often very time consuming and addictive genre. The most popular MMORPG is World of Warcraft which takes place in a gigantic fantasy world where the main goal is to build a powerful character by killing monsters and collecting items.

There are also MMORPG that are browser-based. Being browser-based makes the game accessible on any computer with a Internet connection. But the browser-based MMORPG lack the ability to control a character by moving around in the game, so the battles are mostly settled by chance and the characters strength. Except for that the browser-based MMORPG have the same features as regular MMORPG.

1.2.3.2 Life Simulation

In Life Simulation games the player controls one or more artificial life characters like humans or animals. The main goal of the Life Simulation genre is to create a realistic game environment where the player can act out real-life situations and hereby getting the feeling of interacting with real people. The most popular Life Simulation game is The Sims. Since it's launch in 2000 The Sims for PC have, according to Gunslot.com[12], sold over 16 million copies world wide making it the 3rd best selling game of all time including games from all platforms.

1.2.3.3 FPS

The FPS genre is one of the most popular genres on PC's and consoles. In general the purpose of a FPS game is to solve some kind of conflict by killing the the opposed evil and hereby save the day. This somehow simple dilemma has fueled game developers to create hundreds of games with various story-lines and themes with only their imagination as a barrier. Throughout a FPS the player finds more weapons as the difficulty rises and the story evolves. Many FPS games have multiplayer features witch enables players to battle other players via the Internet or Local Area Networks.

1.2.3.4 Strategy

The strategy genre covers all strategic games both real-time and turn based. The flow of turn based games is similar to traditional board games where the players each have their own turn. Real-time games follow one timeline for all players and is also the most popular for strategic games. Most strategic games are build around a war scenario where the player must build bases and control units to conquer the enemy.

1.2.3.5 Sports

The Sports genre includes all the obvious sports games like football and golf, but it also covers poker and casino games. The most popular games in the sports genre are the the traditional sports games in the FIFA series. The FIFA series contains games from most popular sports and releases a new version of each game in the series every year.

1.3 Asteroids

This section describes the arcade game: Asteroids, which we have chosen to remake as our webbased game. It briefly comments on the history of the game, as well as the game's features and gameplay.

The source of the following section is [33], unless another source is noted.

Asteroids, released by Atari inc. in 1979, was one of the most popular arcade games during the golden age of arcade games[36]. Asteroids uses vector graphics and a two-dimensional view that wraps around in both screen axes.
The objective of the game is to score points by shooting asteroids and flying saucers. The player controls a spaceship which can be moved in a rectangular space representing an asteroid field. The spaceship can rotate left and right, fire shots and thrust forward. The fire-button allows the player to launch projectiles out the front of the spaceship and when the player presses the thrust-button, the ship accelerates until it reaches a certain velocity - the maximum velocity. The spaceship keeps momentum, though not preserved but decreases over time. Momentum makes it possible to drift in one direction while pointed in another direction. Furthermore it causes direction changes to take longer time. With no brakes, the player is required to visualize the trajectory og the spaceship forward in time to keep it from hitting the asteroids. The player also has the option of using hyperspace, causing the spaceship to vanish and reappear in a random location in the asteroid field. This is a risky move, because the space ship can appear right in front of an asteroid, as well as in open space. Each level begins with a few large asteroids drifting in random directions.
Objects wrap around screen edges - for instance, an asteroid that drifts off the top edge of the screen reappears at the bottom and continues moving in the

same direction. As the player shoots asteroids, they break into smaller asteroids that frequently moves faster and are more difficult to hit. Smaller asteroids also score higher points. The asteroids come in three sizes: Large asteroids that breaks into 2 medium size asteroids, medium asteroids that breaks into 2 smaller asteroids and small asteroids that simply disintegrates when shot. Occasionally, a UFO appears in the playing field. The UFOs are of two kinds: Large saucers fire in random directions, while small saucers aim at the player's ship. Once the screen has been cleared of all asteroids and flying saucers, a new set of large asteroids appears. The number of asteroids is increased by one at each level, up to a maximum of twelve. The player gains one "life" for every 10000 points accumulated and the game is over when the player has lost all lives.

1.4 Addiction

This section describes how addiction is a factor in games. We cover the different aspects of addiction and determine how substantial the difference is between games which can be completed and which cannot.

Addiction can show itself in many ways. When the word addiction is mentioned people tend to associate it with drugs, but people can also become mentally addicted to video games. This is often mentioned as Video Game Addiction and is most noticible in MMORPG. There are also people who get addicted to drama series in television and other types of long term concepts. [30]
Statistics have showed that 70-90% of American children play video games. It also shows that the typical player of a video game is a 30 year old male who plays approx 8 hours a week, however, the most addicted players were the 9 % who played MMORPG, these addictive players are in the hardcore player category.[9]

1.4.1 Infinite games

The key to addiction is often to create something exciting and build it so people can see themselves as a part of it. The technology has evolved a lot which not only results in better graphic but also in the development of communication. Game creators of the biggest games today have developed their own community with chatting functions, voice communication and friends lists. An example of this is *Microsoft's Xbox*, where the player has the ability to chat with the opponents inside all kinds of games and make online friends. Another example is World of Warcraft, which is currently one of the most familiar MMORPG. People can play that game for years, many hours each day, and still have something new to explore or upgrade within the game. MMORPG also provides a range of challenges where a lot of players have to work together to complete a task, which means that the gamers are spending time organizing tactics to use in the game. The conclusion is that people not only get addicted, they also tend to make a of commitment to the game and co-players, to ensure that they have some sort of progress in the game along with the other players. All this could

very well have serious consequences for people who have trouble prioritizing and controlling their gaming habits.[9]

1.4.2 Advantages

There are a lot of advantages when looking from the angle of a game developer. Addiction to a game increases the interest of the game and the time and money people are prepared to spend on it. Furthermore addiction to games has not been studied that much yet, but is discussed widely. However, there are no restrictions of how addictive games may be. This means that game producers are spending a lot of resources on studying how they can make their game addictive without showing it too much. To make a game addictive it needs to be fun to play, which is the reason why so many people tend to play too much; however, there is not really any pros for a gamer to get addicted to a game.[9]

1.4.3 Disadvantages

If a game has a lot of addictive traits the media will often target this game, which in most scenarios will give the game a negative reputation and give the gamers more awareness of their time spent playing this specific game. The only disadvantage for developers when it comes to addiction is the possibility of creating a game that is too addictive. This could create bad publicity for the company and might reduce sales.

There are a lot of side effects to being addicted to a game from a gamers point of view. The society of the game replaces the real society; however, it does not remove all social contact with other people resulting in the gamers isolating themselves. There have been cases of people quitting their education caused by too intense gaming. In a worst case scenario this type of addiction would destroy the players future in terms of education and real life social interaction.[9]

1.5 Social aspects of computer games

In this section we want to look into the social aspect of computer gaming. We will reveal that online gaming is more social than the common public belief.

Computer gaming has often been seen as anti-social, but can multiplayer-gaming on the internet be defined as anti-social? Is there a virtual social network?

Playing video games, has often been thought of, as an anti-social behavior. To most people outside of the gaming community, it may at first seem like a very solitary activity, to sit in front of the computer, all alone, and play some online computer game. Online gamers are actually very social, and spend many hours on socialicing with fellow gamers. Players of online games, dont only chat on the in-game chat functionalities, they often use external programs (ICQ, MSN, etc.), web-based forums and even phones to communicate and organize their

activities. [22]

Studies of online gamers have shown that there is a underlying deep social and complex community, despite of the common contradictory belief. Many games rely on co-operative team play, and thereby leading the way for strong communities and friendships, even beyond the virtual online world. Many of the hardcore gamers meet up in person to discuss complex tactics with their team mates. And thereby form long lasting real-life friendships, people even travel across long distances to meet their team mates. This is a main difference between casual and hardcore gamers, this shows the professional approach and care hardcore gamers take in the battle of being the best team, and why their social skills evolve through online gaming. [28]

In larger teams (or clans, as many online teams are called), there often tends to be hierarchy, where the most active members holds the highest positions. Different society-roles have no or little influence in the forming of teams in this virtual world of cyberspace, teams can form with young as old players, boys and girls - and evens spread across nationalities. This is a unique strength for online multiplayer games, bringing people from all over the world together in a common cause.[17]

1.5.1 Invites

A popular concept in online games, is invites, where one can invite (or challenge) a friend in a game. This concept applies to all genres of online games, from web-based card games to modern MMORPG, and encourages to social involvement between gamers, and even to newcomers. This is sometimes implemented directly as a feature in the game, or just as a function to send an email to a friend.

1.6 Trends in modern games

This section defines some of the main trends in modern PC and console games. This is done by analyzing sales charts and thereby finding the best-selling games of last year and now. We close with an assessment to sum up what we have discovered in this chapter.

Even though were targeting the PC market we also cover the gaming trends on consoles and hand-held consoles to see if there is any trends that might be relevant for our game. Furthermore we include the trends from last year to get more solid idea of whats popular and in which direction the trends are moving. The next three sections describes the trends for PC, browser-based and console gaming.

1.6.0.1 PC

The best selling and most played genres for PC games in 2006 were MMORPG and Life Simulation. According to NPD (market research firm)[21] the best selling PC game of 2005 and 2006 was the MMORPG World of Warcraft(WOW). Blizzard[3], the makers of WOW, announced in July that they have surpassed a staggering 9 million subscribers world-wide, making it the biggest online game ever. The life simulation game The Sims 2 and its expansion packs dominated five out of ten on NPD's best-selling PC games of 2006. Also on the list are three RTS (Real Time Strategy) and one RPG (Role Playing Game).

Best selling PC games of 2006.

1. World of Warcraft
2. The Sims 2
3. The Sims 2 Open for Business Expansion
4. Star Wars: Empire at War
5. The Sims 2 Pets Expansion
6. Elder Scrolls IV: Oblivion
7. Age of Empires III
8. The Sims 2 Family Fun Stuff Expansion
9. Civilization IV
10. The Sims 2 Nightlife Expansion

Source: NPD[21]

The characteristics of the popular games of 2006 shows that they all have features that allows the players to evolve their characters with almost endless possibilities. This tells us that the main trend in PC games from 2006 was that the games had infinite gameplay features. As described in chapter 1.4(Addiction) these features has a tendency to make the players addicted to progressing in the game and therefore they spend more time playing, which in the end creates more publicity for the game. Why spending more time on a game creates more publicity will be described further in our stakeholder analysis.

On a recent NPD chart showing the best selling PC games for the week ending October 13, it becomes clear that the trends have changed a bit since last year. Even though this list only details the sales for one week it draws a pretty good picture of where PC gaming is today. The top-sellers from last year still has good sales which shows that the infinite gameplay games have come to stay.

WOW and The Sims are still well represented on the list, but there is a now wider variety of genres represented in the top-selling games. It is noticeable that most of the games on the list have multiplayer features that are more or less essential to the game. This fact further underlines the tendency that PC games are moving from singleplayer onto a more internet-based gameplay.[20]

Another newcoming aspect of online virtual worlds is, bying property. For instance, in games like "Second Life", "Project Entropia" and others, players can buy virtual objects like land, weapons, cars, houses, etc for real money. Some of the games even allow players to exhange their virtual money to real money, and vise versa. [1]

Best selling PC games of week 41, 2007.

1. Half Life 2: Episode 2 The Orange Box
2. Nancy Drew: Legend Of The Crystal Skull
3. Neverwinter Nights 2: Mask Of The Betrayer
4. The Sims 2 Bon Voyage
5. Enemy Territory: Quake Wars
6. World Of Warcraft: Burning Crusade
7. World Of Warcraft
8. The Sims 2 Deluxe
9. World In Conflict
10. World Of Warcraft: Battle Chest

Source: NPD[20]

1.6.0.2 Browser-based

We were unable to find any relevant information on what the trends in browser-based games, so we searched different gaming sites and found what we believe is most popular and therefore the current trends. The three most popular genres in browser-based games are MMORPG, Puzzles and Arcade. The browser-based MMORPG are popular for the same reasons as regular MMORPG. Puzzles are small games, like Sudoku or rubik's cube, where the player has to think a lot about the next move. Browser-based arcade games are remakes of the old classic arcade games like Tetris, Pac-Man and Asteroids. Web-based puzzle and arcade games have in common that they both are easy to learn and usually does not last more that five minutes. This makes them appealing because it is then easy for the player to go to the site, take a quick game and then leave again.

1.6.0.3 Consoles

The most popular genres for consoles in 2006 were Sports and FPS. The best selling game was Madden NFL 07 which sold over 3.5 million copies to the *PlayStation* alone. The next-best selling game of 2006 was the visually stunning Gears of War for Xbox. [24]

The most recent sales list from April 2007 made by NPD states that the new Nintendo Wii console has taken the lead in games sold. The two most sold games of April were Super Paper Mario (Platform) and Nintendo Wii play. Nintendo Wii play includes nine different sport games and has become very popular because of its use of the gyro controller that is controlled by the players movements. One of the reasons why the Wii is so popular is that it targets some of the health issues of video gaming by adding physical movement. One of the main trends of console gaming today is games that require special controllers like the Nintendo Wii, Guitar Hero and Singstar. Special controllers in games like Guitar Hero and Singstar also makes the games suitable as a fun element at party's. Like on the PC it has also become a trend that console games have online features. [15]

On hand-held game consoles the most popular games are mostly in the genres of Platform and RPG. Even though the RPG Pokemon for Gameboy is the best-selling game of all time with over 20 million copies sold, the trends in what is popular seldom change on this platform. Although new more powerful hand-held consoles like the *PlayStation Portable* and *Nintendo DS* enhance the gaming experience, users tend to stick to the classic genres. Last years top seller was the platform game New Super Mario Brothers, a new addition to the classic Super Mario series, for Nintendo DS. As technology has advanced, some hand-held consoles now have Multiplayer features that connects the consoles via infrared or bluetooth. [29]

1.6.1 Assessment

On the PC platform we found that the main trends were infinite gameplay and multiplayer features. We also found that browser-based remakes of classic arcade games are quite popular on the internet. On the console platform we found further confirmation that gaming in moving onto the internet by including multiplayer features.

1.7 Target Audience

In this section we determine target audiences for casual and hardcore games. Furthermore we discuss the target audience of this project and base it upon these two groups.

Today arcade games have pretty much been replaced by the "casual" games.

The same things describe the games. Easy controls, flat learning curve, fast levels and increasing difficulty. But does the same people play these games? In the "golden age" of arcade gaming, in the early 1980s most arcade games were mostly played by male teenagers. The cost of playing the coin-operated arcade machines raised the average age of the gamers. Most children did not have any income and therefore it was more common that teenagers with afterschool jobs played arcade games. [37][32]

1.7.1 Casual Gamers

According to a recent study in casual gamers by Information Solutions Group (ISG) commissioned by PopCap Games there are some things that describe the common casual gamer. [45] This study shows that 76% of the players are female and that the average age is 48 years. 88% indicated that they felt stress relief when playing these games. There are some things to take into consideration though. The study tested around 2000 PopCap Games customers and only a certain type of people might take these online surveys. Perhaps PopCap games attract a certain type of casual gamer and other developers might have a different target audience. Anyway its safe to say the children and teenagers are not the main target audience for casual games today. Some casual gamers are children and teenagers but it seems a large percentage are mature women.

Why females tend to like casual games could be because casual games are easy to pick up. According to game designer Steve Meretzky these games attract a female audience because "you can learn them quickly without having to read a manual or wade through a long tutorial. The shorter playing time and less-violent themes are also appealing to this demographic." [5] [40] That means the players can sit down and pick up the gameplay in a few minutes because of the flat learning curve. That makes the games much more usable for short playtimes.

Casual gamers typically like playing games with no or little in-depth story and easy learning curve, which are easy to pick up and start playing instantly. Arcade games tend to please casual gamers. Casual and arcade games share almost the same characteristics.

1.7.2 Hardcore Gamers

A study commissioned by the Entertainment Software Association[5] by Ipsos Insight in May 2006 showed that only 38% of all video game players were female. Where the casual games were dominated by the female audience, "hardcore" gamers were mostly consisted of males. Age ranges from 14 to 34 in general[44]. They often have high knowledge about computers in general and pay interest to the products of the gaming and computer industry. They often keep themselves updated about the latest technological advancements.

As an example the playerbase for one of the most complex MMORPG "Eve Online" is 95% males 5% female and has an average age of 27. [26]A large group

of hardcore gamers prefer games where it is possible to make progress. Not just like the increasing speed in games like Tetris, but more like progress over a long stretch of time. They like for example to "build" a character or maybe an online empire. Something that takes weeks, months or even years. These games are often multiplayer games. A steep learning curve is much more common here. Unlike the casual games some hardcore games can take months to learn because of the complexity involved.

1.7.3 Comparison

The main difference between a hardcore gamer and a casual gamer is the time spent playing games. The amount of time a casual players spends gaming is generally limited compared to the time a Hardcore gamer spends. Furthermore a hardcore gamer shows higher interest in games in general. Hardcore gamers are often part of one or more online gaming communities. A Hardcore gamer is also likely to take gaming more serious, making it more than just a pastime. You could say that casual gamers take gaming as a hobby while hardcore gamers takes a more professional approach. That said its important to point out that the line between the two groups is indistinct. Some gamers play both hardcore and casual gamers and some play more "hardcore" than others. [35] [38]

When comparing the group who played arcade games in the 1980s and the hardcore gaming group of today there are similarities. Its mainly the same type of people. The target group of arcade games at the arcades back in the 1980s is mostly the same target group as for hardcore games today. Meaning a new group of gamers has emerged. That is the casual gamer.

By combining a casual game with a hardcore game. For example taking a puzzle game like Tetris and adding RPG elements. The hardcore gamer will be attracted by the character building and the casual gamer will be attracted by the "easy-to-pick-up" Tetris game. By blending two genres one could create something new for the two target groups and something that is suitable for just about anyone, regardless of the players interest in the two different components in the game, or it could end up appealing to neither. Another possibility is to combine a singleplayer game and make it part of a multiplayergame. For instance players have a character in an online world and everytime the characters fight eachother they play Tetris. The winner could for example be determined by highest score.

Our project will be aimed at such a combined target group. We will try adding new modern features to a game with simple gameplay.

1.8 Stakeholder analysis

This section describes the interesting aspects of gaming seen from different perspectives. What is the relation between developers, distributers ans players in game-development?

If this project should be realized and implemented on a website then we would have to do a stakeholder analysis. A stakeholder analysis has the purpose of finding who a project will have influence on, or who it wil be affected by. The connection can be economic aspects, both negative and positive economic aspects but it can also be other aspects. For a game project there are many stakeholders; The developers, the providers, the players and venture companies etc. For a online webgame like asteroids the most common ones is the developers (us), the players and the providers. In the following we have found and described the individual connection between the three kinds of stakeholders and a project about a online webgame like ours.

1.8.1 Developers

There are two groups of developers. The first one is a company of developers, which have more interest in money and the other is a single developer.

A company of developers first obvious interest for creating a project is to make money. If the project is not profitable then there is no reason for beginning the project. The developers can also be interested in a project if it is fulfilling a potential demand and thereby produce a new market for a new type of software. This could also be profitable.

The single developer could have other motivations than profit. Of course profitable projects is prefered, but a developer is also interested in projects that are challenging for him and thereby make him more experienced. These projects could be open source software or a vision that the developer got for a project which he can be very fanatic about.

1.8.2 Providers

The providers in this case has a website which they want to make more entertaining and therefore has planed to implement a game. A game on their website could create:
- increased traffic
- more publicity
- more money

The increased traffic is gained by the players that comes to the website to play the game. They enjoy the game and spread the word to their friends that there is a cool game on the website and urge them to try it out. If the friends try it out and find it enjoyable they spread the word to their friends, which creates a exponential growth in publicity. A community of players on the website creates

lots of publicity for the provider. The main aspect is yet again money in the form of advertising space on the website. Many users and lots of traffic is a solid ground to supply other companies the solution to bring their advertisement at the website and with a simple agreement the providers can rapidly earn a significant amount of money. Another aspect for the provider is the content of the game. If the content is provocative, offending or violent then it could create bad publicity for the providers. This means that if the providers host the game on their website then they have to accept the consequences if someone finds it offending.

1.8.3 Players

The players are very important in the proces of creating and running a online webgame. If there is no players, then the advertisers will not be interested in buying advertising space. This leads to no profit for neither the providers or the developers. But if there are many players then there will be a profit for both the providers and the developers. But what does the players get out of playing? First of all, a quick online game can make players relax and get away from a task they might be stuck in. Secondly, there could be a prize for winning the game and that is a good motivation for players to keep on playing. The advertisers on the webpage could also bring a good offer on their products, so the players could save some money by buying online at the advertisers.

1.8.4 Assessment of impact

Above we have described the connection between a stakeholder and the project. What is left to do is to evaluate the importance of each interest from the stakeholder. We have summarized the description above in the table below.

Stakeholder	Developers	Providers	Players
Stakeholder interests in the project	A vision Fulfilling a demand Money	Money Publicity Traffic	Entertainment Challenging Social
Assessment of impact	Important	Important	Very important
Potential strategies for obtaining support or reducing obstacles	Make players enjoy the game	Connect players with advertising companies	Play the game and create publicity

On the basis of this table we can see that the most important stakeholder is the players. The developers has to create the game in a way that it attracts players to play the game or else there are no market for such a game. This means that in our project we must develop a game that catches the players attention in a way that makes him/her come back to the site and play the game. If we do that then

the providers will be more interested in implementing the game on their website.

The source for the previous section is an article in "Venture #2 2007" - a danish magazine about investment and in this issue the subject is leisure economy[27].

1.9 Summary

Throughout the pre-analysis we have covered the different aspects we thought to be most important before building a web-based game. We started by introducing our initiating problem and then we established some basic knowledge on the game Asteroids and video games in generel. Then, in the sections Addiction and Social aspects of computer games, we found that addictiveness in games could create bad publicity for the developers and have negative effects on the players social life. We also conclude that online gaming is more social than the common public belief. Next up in sections Trends in modern games and Target audience we found the answers to the first two questions from our initiating problems. Finally, in the stakeholder analysis, we determine that for our project to be profitable we need to secure the players return to the site. This could be done by implementing infinite gameplay and multiplayer features. The answers to our initiating problems are:

- The main trends in modern gaming are infinite gameplay and multiplayer features.
- Combining a classic arcade game with new modern gaming trends would create a game suitable for both casual and hardcore gamers.
- There are three main stakeholders, of which the developers and the players are the two most important.

1.10 Problem description

Throughout the problem analysis we have discovered what the situation is regarding video games. We found that an Asteriods game that could reach out to both casual and hardcore gamers would have a great market potential. First of all, we noticed that our game needs to have a lot of factors from the old arcade games; basic graphic, easy to control and hard to master. Furthermore games of that type have basically changed to being web-based games on the internet, which is why our Asteriods game will be online on our web-based game portal as well. Finally we will fill the game with features which allows the user to evolve in the game and play against other players which will create an addictive side of the game. This will drag more players to the game and hopefully create a better reputation for the game. This leads us to the following three questions.

- Which specific features should be included in the game to make it appealing to both casual and hardcore gamers?
- What is a good solution for building a game based on merging casual and hardcore gamers?
- How to create a extendable games portal for setting up numerous web games?

1.11 Problem limitation

During the course of this project, we acknowledge that it is impossible to cover all the aspects of a modern Asteriods game, and all the features of a full scale web-based games portal. We will limit the extent of the project later, when we have described the full game and portal with all the desired features.

Chapter

2

Technical Analysis

2.1 Introduction

When developing a browser-based game like the one we want to make for this project, one can imagine that a great deal of knowledge about the different technical aspects that might come into question, is required. That is why we have chosen to research these things and write a chapter about it. In this chapter we introduce and discuss the different types of technologies we could potentially make use of in the development of our product. There are many different types of programming languages, database types, programming procedures etc. and we want to make sure that the ones we choose to use for our product, are both reliable and easy use learn, because we do not have a lot of time to learn them in.
Conclusively we describe what our basic idea is for our final product. How it works, how it looks, etc.

2.2 Browser Games

This section describes the advantages and disadvantages of developing games for internet browsers instead of stand-alone-clients, as well as a few things about the technologies used for browser games.

The source of the following section is [34], unless another source is noted.
Browser-based games are video games, that are played on the internet through a browser, such as Internet Explorer, Firefox, Opera etc.

2.2.1 Advantages

The main difference between a browser game and a regular computer game is that it does not require the user to install a client to play, apart from the browser itself and common plug-ins. Common plug-ins include Java, Shockwave and Flash, and many of these plug-ins are available through default installations of most modern browsers. Another advantage is that since the client does most of the processing, the server does not receive a heavy bandwidth load of requests. More recent browser-based games use web technologies such as AJAX (see page 30) to make more complicated multiplayer interactions possible.

It is possible to produce games without the use of 3rd party plug-ins. A game can be written in DHTML (dynamic HTML), which is a combination of CSS and javascript. It is normally used for creating graphical menues etc, but could also be used to animate effects for action video games. Games produced with AJAX use a combination of DHTML and server-side scripting.

Some issues can be resolved by using server-side scripting. The most common languages used for this are PHP, ASP, Ruby, Perl, Python and Java. These games are stored on a server, which sends HTML for the user's browser to interpret. HTML is used to format webpages and is therefore not ideal to make more advanced graphics. HTML is covered in detail in the technical analysis.

Server-side scripting can be combined with javascript or AJAX if the developer wants to give the game a more visually appealing look. With all game code on the server side the setting becomes more secure as the player does not have direct access to it, making it more difficult to alter the code and cheat.

2.2.2 Disadvantages:

The plug-in-based browser games requires the user's browser to download and utilize the game code on the client side. This is a security risk and it makes users able to to decompile the code and thereby spoiling fair multiplayer gameplay. The result of this is that browser-game developers often simply avoid the multiplayer genre and focuses on safer single player games.

The downside of DHTML is that it is very visually limited compared to games developed with the use of 3rd party plug-ins.

The problem with server-side scripting is that the server is more easily overburdened, because the processing for every single game is done on the server.

2.3 Web Development Technologies

In this section we cover five web development technologies we might use in the development of our product.

2.3.1 HTTP Protocol

The HyperText Transfer Protocol (HTTP) is the standard protocol that is used when browsing the web with a browser. This communication protocol is in charge of transfering the HTML source (see 2.3.2) to the requesting party. The HTTP protocol is stateless, and thereby considers a request and response as a single entity.
When a client (i.e. a web browser) requests a page, i.e. http://example.org/index.html the web server listing at port 80 (the default port for HTTP communication) will return the html source for the index.html file.

2.3.2 HTML: HyperText Markup Language

HTML is a markup language, which in essence means that it manages formatting and layout of a document. HTML is now developed and maintained by World Wide Web Consortium, originally developed by CERN[1]. When used on the Internet the HTML source markup is sent to the client browser, and it is the browser's task to display the formatted markup in a correct manner. This is subject to several issues, because it is up the the manufacturer of the browsers to support the full HTML standard, as one can imagine not all browsers live up

[1] Centre Européen pour la Recherche Nucléaire

to these requirements. It can then be a challenge to display a document identically in all browsers. The HTML standard itself comes in different versions, as it has evolved over time. The only ones actually in use today is HTML 4.01 and XHTML 1.1. XHTML (Extensible HyperText Markup Language) is a more strict version of HTML and it conforms to valid XML structure.

The basics of HTML is tags, for example the following snippet:

```
<h1>My Heading</h1>
```

will make the text appear as a heading.
Another key feature of HTML is to interconnect documents on the network, known as Hyperlinks.

```
<a href='computer_science.html'>CS Department</a>
```

This will make the text "CS Department" point to the computer_science.html document.

An essential HTML document would be:

```
<html>
<head>
        <title>My document</title>
</head>
<body>
        <h1>Welcome to my document</h1>
        <p>Here goes the contents of my document</p>
</body>
</html>
```

2.3.3 CSS: Cascading Style Sheets

CSS is a technology for webdesigners, which allows them to create more graphical layouts. The technology gives the webdesigner the possibility for adding a specific style for every HTML-element. This technology emerged December 1996 at the World Wide Web Consortium. Before CSS there were no common way to style a HTML-document, but several different ways were developed and it depended on the browser.[4]. Today CSS is widely used and supported by most browsers. The advantages of CSS are many:

- CSS uses pattern matching to apply a style on a certain tag. This method have some significant selectors which refers to the hierarchical order within the HTML-document[13]. The selectors are:
  - Child: The child selector is used if for instance a <b>-tag is within a <p>-tag then the <b>-tag is a child of the <p>-tag. This also means that the <p>-tag is a parent of the <b>-tag. In CSS you write a child selector as p>b.

  - Adjacent: The adjacent selector is used when two tags have the same parent and the one tag occurs after the other. In CSS you write an adjacent selector with a plus sign between the two tags.
  - Descendant: The descendant selector is used when a tag is within another tag. In CSS you write a descendant selector with a space between the two tags.

- You can group HTML-elements by using class-attributes, id-attributes, span-tags or div-tags. In this way you can apply the same style on every tag in the group by only writing the CSS-code once. Another way to group tags in CSS is by using a comma between the tags.

The CSS-code can be implemented in three ways:

2.3.3.1 Directly into an HTML-tag

Inserting CSS-code directly into a HTML-tag is done by using the style-attribute:

```
1 <p style="color:red;">Hello</p>
```

This method is very useful for styling a single tag in a specific way. If the HTML-document consists of many HTML-tags and nearly all tags have to be styled with CSS, then this method is not recommended.

2.3.3.2 In the top of an HTML-document

Inserting CSS-code in the top of an HTML-document is done by using the style-element:

```
1 <style type="text/css">p {color:red;}</style>
```

This method is more convenient to an HTML-document with many tags that are using CSS-code. Instead of writing the CSS-code in the HTML-tag, it is written inside the style-element. In this way you can avoid writing the same CSS-code in multiple HTML-tags that ought to look alike. If the website consists of many HTML-documents, then you have to write the style-element in every HTML-document. The method is therefore only suitable for websites with a single HTML-document.

2.3.3.3 In an external CSS-file

Using an external CSS-file is done by adding this line to the head-tag in a HTML-document:

```
1 <link rel="stylesheet" type="text/css" href="stylesheet.css" />
```

It is possible to edit the layout of many HTML-documents by editing a single CSS-file. It is more convenient for the webdesigner, when you do not have to edit every HTML-document, but only a single CSS-file. Another advantage is that the browser is able to cache the CSS-file and thus saving some server-client traffic

2.3.4 JavaScript

In 1995 Netscape initially created a new web scripting language called LiveScript. Netscape made LiveScript a two-folded blade, server administrators could use this new scripting language to manage their servers and developers could use it to create client-software. For instance, developers could use LiveScript to make sure that the information a client enters into a web form is of the right type; integers, emails, zip-codes etc. Instead of enforcing this on the server side, requiring a (of that time) expensive server request and response. This could be used on the clients computer to save a request/response, instead of wasting server power.

Later Netscape and Sun Microsystems (the developer of the Java language) announced that LiveScript changed name to JavaScript. Netscape had marketing reasons for changing the name, but this change have given many confusions about the connection between Java and JavaScript. Microsoft later acknowledged the power of JavaScript and implemented it in their browser, Internet Explorer 3, under the name JScript.

The JavaScript code itself is embedded into the HTML page and sent to the client, which means that the client user can lookup the JavaScript source along side the HTML markup. Therefore JavaScript is not the language of choice for connecting to a database, sending secret emails etc. JavaScript is often used to make an otherwise static web page more interactive. [10]

Another new and popular collection of technologies is AJAX (Asynchronous JavaScript And XML), which as the name implies, relies heavily on the use of JavaScript and XML. AJAX is actually taking old technologies and stretching them well beyond their original scope. Without going into details, it is sufficient to say that AJAX is used to make "rich client interfaces" on top of existing web technology. AJAX is often associated with the *buzzword* "Web 2.0".[6]

2.4 Programming languages

In this section we have chosen some languages that we thought best for this project. We discuss positive and negative properties of these languages. Furthermore we have decided which languages we will use for this project.

2.4.1 Java

Java is an object oriented programming language originally developed by Sun Microsystems, released 1995. The syntax is similar to C++. A program should be able to be compiled once and then executed on any platform. It runs on a virtual machine (VM), which nearly all operating systems and mobile phones have installed.

Advantages

- Java can be written as applications, applets, webpages and mobile applications.
- It is platform independent and can run on any system with a VM.
- In most cases VM is already installed on the platform.
- We can create many types of projects based on the same source.
- It is object oriented.

Disadvantages

- It is required to install a VM before the Java code can be executed.
- There is a range of decompilers for applets which makes it possible to reverse engineer the program.

2.4.2 C#

C# is a language developed by Microsoft and is only fully compatible with Windows. It is possible to run C# code under other platforms such as UNIX with 3rd party technology such as mono. Programs can be developed for mobile devices if they are running Windows Mobile. C# is a part of the .NET initiative and .NET and is also an object oriented programming language. Based on the advantages of many languages, C# is very similar to Java, but with some extra features. In some cases, you can directly copy Java-code to C# and compile it. All .NET languages like C#, ASP.NET, C++.NET, Visual Basic.NET are able to work together.

Advantages

- .NET can be used for many types of projects like webpages, programs, services, web services, etc.
- Many .NET languages can be run simultaneously on a single platform.
- We can create many types of projects based on the same source.
- .NET is very popular for web development due to server side programming.
- It is object oriented.

Disadvantages

- .NET is only fully compatible with Microsoft Windows.
- You need to have an up-to-date version of .NET; however, this comes with any windows update.

2.4.3 Flash

Flash is much like a Java applet running on a webpage and is mostly used for website development. It can be used as a part of a webpage or as a whole webpage. ActionScript, a language based on JaveScript, is used to code Flash applications. The advantages of Flash is mostly on the graphical part, which is the reason many use this for browser-based game development.

Advantages

- Possible to create graphic effects and is based on drag-and-drop technology.
- The Flash player, which must be installed, can run on several platforms.

Disadvantages

- Adobe Flash, also known as Shockwave Flash or Flash must be installed on the computer.
- The code can easily be decompiled and it is not possible to do any server side running.

2.4.4 Conclusion

We have decided to write the game in Java and the website in C#. This combination will make our product able to run on any operating system and there are no limitations for adapting the game to other devices, such as phones. We will have to learn the languages and since C# and Java are very similar this is an easy way to do it. Most web games are made in flash since it gives some graphical advantages, but learning Flash in the short time we have, is not practical.

2.5 OOP: Object Oriented Programmering

In the language section we determined object oriented programmering as an advantage. This section describes what OOP is and how to develop software using it.

The concept of OOP is the use of classes and instances of these classes, which are the objects. In this section we will use a simplified example about vehicles and cars. The use of this example should make the understanding of OOP more simple. The first piece of code is shown beneath to create an underlying basis for our explanation of OOP:

```
1 public class vehicle {
2         //The fields:
3         private int wheels;
4         private String manufacturer;
5         //The constructor:
6         public vehicle(int _wheels, String _manufacturer){
7                 wheels = _wheels;
```

```
8                   manufacturer = _manufacturer;
9           }
10          //The methods:
11          //Getters and setters:
12          public void setWheels(int _wheels){
13                  wheels = _wheels;
14          }
15          public int getWheels(){
16                  return wheels;
17          }
18          public void setManufacturer(String _manufacturer){
19                  manufacturer = _manufacturer;
20          }
21          public String getManufacturer(){
22                  return manufacturer;
23          }
24          //Specific class-methods:
25          public void Drive(){
26                  //moves the vehicle
27          }
28          public void TyreChange(){
29                  //changes a tyre
30          }
31  }
```

2.5.1 Classes

As you can see in the example, a class is a model of a real-life object[7]. In our example we have written a simplified structure of a vehicle. This class has two variables, “wheels” and “manufacturer”, which characterizes a vehicle. These variables are also commonly known as fields. Setting the state of the fields is an expression that means to alter the variables in the class. When creating an instance of a class, the class uses a constructor to set the state of the fields in the class. A constructor is a method with parameters and it is called when creating an instance of an object. The constructor must, in many languages, be named the same as the class. There can also be multiple constructors inside a class which normally are one without parameters and one with parameters. A class can contain methods. There are mainly two kinds of methods in a class. The first kind is the methods which change or return the state of a field in a class. These methods are called mutators, if they change the variable. And accessors if they return the variable. Secondly is those methods which are specific for the class. In the example there are two mutators, setWheels() and setManufacturer(), and two accessors, getWheels() and getManufacturer(). The two methods Drive() and TyreChange() are specifc methods for a vehicle which brings the aspect of reality to the class. A variable is either private or public. Public means that it can be read or used outside the class, private means it cannot.

2.5.2 Inheritance

A class is able to inherit from another class. To illustrate this we have created a class "car" which inherits from the class "vehicle":

```
public class car extends vehicle{
        //The fields:
        private int mileageRecorder;
        //The constructor:
        public car(int _wheels,
                        String _manufacturer,
                        int _mileageRecorder
                ){
                super(int _wheels, String _manufacturer);
                mileageRecorder = _mileageRecorder;
        }
        //The methods:
        //Getters and setters:
        public getMileageRecorder(){
                return mileageRecorder;
        }
        public setMileageRecorder(int _mileageRecorder){
                mileageRecorder = _mileageRecorder;
        }
        //Specific class-methods:
        public void TurnOnEngine(){
                //Turns the engine on
        }
        public void TurnOffEngine(){
                //Turns the engine off
        }
}
```

As you can see in the example above, the structure of the class "car" is not much different from "vehicle", though there are two lines that differ. Besides the class declaration the first line also has an extra phrase "extends vehicle". This means that this class "car" inherits from "vehicle". Because of the inheritance from "vehicle", the "car" class also contains the fields, constructor and methods from "vehicle". This means that the constructor from "car" calls a method "super" which invoke the constructor from "vehicle" with the parameters it needs. "Car" has furthermore two methods "TurnOnEngine()" and "TurnOffEngine()" which are specific to the class. In OOP multiple inheritance is possible, but depends on the programming language used.

2.5.3 Objects: Instances of classes

As mentioned earlier, objects are instances of classes. This means that when you create an instance of a class you create an object. If you want to create an object named "Lorry_1" from the class "Vehicle" then you would write:

```
Vehicle Lorry_1 = new Vehicle(6,''Scania'');
```

This creates the object "Lorry_1" and the object has 6 wheels and "Scania" as manufacturer. Now you can call the methods of the object by using dot

notation. Dot notation consists of a dot between the name of the object and a method from the object. If "Lorry_1" punctures and we need to change the tyre, we could write:

```
1 Lorry_1.TyreChange();
```

Then the method "TyreChange()" is invoked and the "Lorry_1" changes tyres. OOP is effective when the architecture of the program is schematized. This is usually done by making a UML-diagram, which is described in the following section.

2.6 UML: Unified Modelling Language

UML is an abbrevation for "Unified Modeling Language", and is a way to describe all sorts of models using diagrams. It is used for many things including business processes, data and application structures, behavior and architecture. Currently there exist a lot of different tools to edit and view UML; typically it is saved as XML-files.

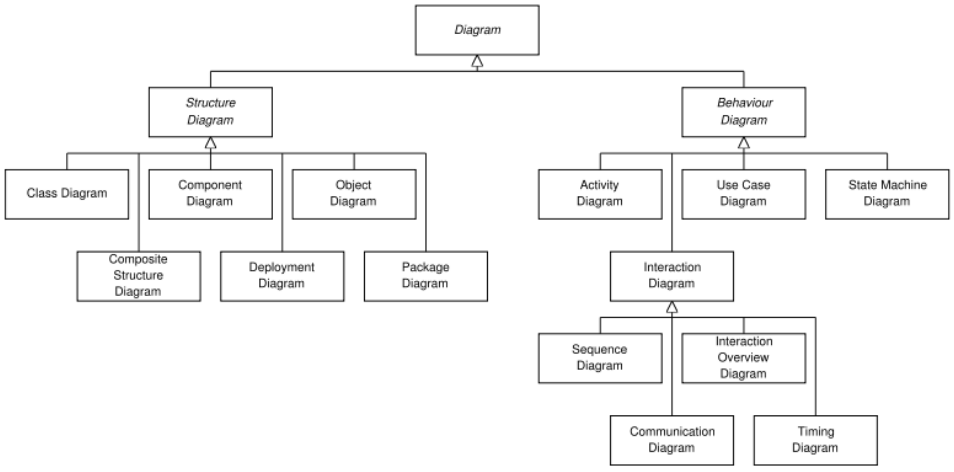


Figure 2.1: Different types of UML diagrams.

The figure represents different types of diagrams which can be created using UML. There are two large groups of diagrams: Structure diagrams and behavior diagrams. Both can be important for developing, but we will focus on the structure. We will especially focus on the class diagram for our website and our Java game. Furthermore we will use a database diagram which is not in the picture. The reason why people choose to write an UML Diagram in the planning phase of a project is to ensure that the structure is correct. We can save alot of unnecessary coding and constructing by planning the structure correctly from the beginning. Therefore UML is always a good solution when beginning on a larger project.[11]

As described above, we will use UML to structure our program. Furthermore we will use it to make diagrams that show the structure of our database. For this purpose we have chosen to use a program package called Visual Paradigm that contains a range of tools to create and view UML. It also contains a program which can interact with our Java development program: Eclipse. Therefore this is the most obvious choice for creating our UML diagrams. The way to use the program is basically drag'n'drop, which makes the usability of the program good.

2.7 Databases

In this section we will describe the basics of computer databases. We cover the the following topics; relational databases, database servers and SQL.

The source used in this section is the book Introduction to SQL by Rick F.[25]

When handling large amounts of data on a website, it is often a good idea to store it in a database. A database is a collection of information organized in such a way that a computer program quickly can obtain desired pieces of information. When using a database to store data it is possible for more than one program to change and store data at a time, which is very important for websites. There are different types of database models, but we will only cover the relational database model, since it is the most used database model.

2.7.1 Relational databases

Relational databases consists of relations in the form of tables, meaning that the content of a table is related. A table consists of columns and rows. The named columns are called attributes. These attributes then have a set of values they are allowed to take which is called their domain. The domains are set by selecting a data type. Relational databases adhere to three basic rules. First, the ordering of the columns in the table does not matter. Second, each row can only contain a single value for each of its attributes. Third, there cannot be identical rows in a table. Therefore each row in a table has a unique identifier called the primary key. Below is an example of how a relational database is structured.

User table

Id	Name	Age	Group_id
1	Chris Tucker	37	2
2	Maria Kristensen	29	1
3	Jeffrey Stallman	46	2

Group table

Group_id	Location	Department
1	Denmark	Sales
2	USA	Development

This picture illustrates two tables: Users and Groups, from a relational database. The top line of both tables contain the attributes and the other lines contain the data. The tables are interconnected in the way that group_id in Users refers to the Group table. This makes it easier to append data to many users, like in our example information about location and department, by just including one more attribute in the Users table.

Attribute name	Data type
Id	Integer (automatic incremation)
Name	Char
Age	Integer
group_id	Integer

The image above shows how the attributes in the Users table are set. The first column sets the name of the attribute and the second column contains the information on which data type the data is stored as. The first row, Id, is set as a primary key and therefore the data type is set to be integers with automatic incrementation. When a new entry is made the table gives that entry a unique Id by automatically incrementing the Id from the last entry. This is done to uphold one of the rules for relational databases, which is that there cannot be identical rows in a table.

2.7.2 Database server

The data in a database is managed by a separate application in the form of a database server or a Database Management System (DBMS). These applications interprets the SQL language and performs the actions on the database according to the SQL statement received. There are many different DMBS's and database servers and they each have their own SQL dialect. SQL dialects have different functions according to the application they are implemented in. Some of the most popular DMBS and database servers are MySQL, Microsoft SQL, DB2 and Oracle.

2.7.3 SQL: Structured Query Language

SQL, Structured Query Language, is a language used for retrieval and management of data in a relational database. SQL is a language used to interact with the SQL server. SQL can be used in two ways, interactively and preprogrammed. When using SQL interactively the user issues an immediate command, in the form of an SQL statement, to the database server and gets an immediate response according to the statement. This method is mostly used by developers. The other way to use SQL is preprogrammed SQL. Here the SQL is embedded into an application written in another language. For instance a website written in PHP where the data fetched by SQL commands is integrated into a webdesign. To describe how an SQL statement works, we will retrieve some data from the Users table above.

Listing 2.1: SQL statement example

```
1 SELECT * FROM Users WHERE Group_id = 1
```

This SQL statement translates into: select all the data from the table Users where the attribute Group_id equals 1. The SELECT command returns the data that match the requirements stated in the FROM and WHERE commands. The other parameters tells which tables and attributes that are used.

2.8 Persistence with HTTP

The following sections describe different technologies to make persistent data across a series of HTTP requests.

A limitation when working with web technologies and especially the HTTP protocol is the absence of tracking capabilities and persistence. The HTTP protocol is short lived, meaning that when a client request and server response has finished, the communication line is closed.

2.8.1 Cookies

The concept of cookies originated from Netscape around 1996. A server can within a HTTP response send a cookie, that the client browser stores. A cookie consists of a series of key/value pairs of data. Any future HTTP request for which the cookie is valid, the client browser will send back the cookie data. This can be used to persist data across multiple HTTP request, which is otherwise not possible. A downside of using cookies is the security aspects, all the data is stored on the client in clear text, so it can easily be exploited. Furthermore, the transmitted message is clear text as well. [19]

2.8.2 Sessions

To overcome some of the security issues, that arise from cookies, websites use the concept of sessions. Instead of storing the actual data on the client browser,

the session data is stored on the server itself. The server needs some form of identification, to keep track of users over multiple requests, that link the client browser to the session data stored on the server machine. Sessions rely on the functionality of browser cookies, to save a unique identifier on the client browser, sometimes a GET parameter is used to pass along this unique identifier, by adding it to every link automatically on a page. This GET functionality is also used as a backup, in case the client browser does not accept cookies. The session data itself, is not stored in the cookie, only the unique identifier is, and the identifier is then used to match the session data with the data stored on the server. This is normally stored in flat files or databases, but vary from environment to environment, and of course the exact configuration of the setup. The unique identifier is sometimes referred to as, a "session token", this identifier usually comes in the form of a non-sequential generated hash value. The advantages of sessions over basic cookies are clear, the client cannot modify the data, because it is stored on the server, and furthermore the data is not transmitted over the network in clear text - it never leaves the server. Another key point regarding session security, is that the session hash is non-sequential, and malicious people cannot predict the next session identifier. [8]

2.9 Vectors

In this section we briefly describe vectors and why they are used in computer games.

When a physicist tries to make models for real-world objects and how they act in time and space, the use of vectors is fundamental. A vector is basically characterized by a magnitude (or length) and a direction. Graphically this comes down to an arrow with a certain length and a certain direction. The tail of the vector is where it begins and the head is the pointy top of the arrow. In physics things like speed, acceleration and momentum are fairly easy to describe using vectors. Also vectors are used to describe points in space using a fixed location as reference. This is called a position-vector. [41]

2.9.1 Vectors in games

Computer games can be models of something in the real world if chosen to. Game rules are of course not limited to the rules of physics in our world and can be modified if desired. Anyhow vectors would still be used to calculate movement and the like even though gravity for example was reverse. It is possible to change physical or mathematical constants even though it could end up making no sense at all.

We wish to make a game involving a spaceship flying in outer space, and the only reasonable tool for calculating movement would be vectors. There is no logical alternatives to vector-based calculations of moving objects in physics. If we want to change the direction of the moving spaceship we would add a vector

with a different direction than the one which describes the movement of the spaceship at the given time. In real life this would be done by pointing a rocket engine somewhere and thereby adding another force to the calculation. In the game this is done by the push of a button. Other objects or obstacles would also be moved with the use of vectors.

2.10 Product Description

In this section we describe our product with all the features we have planned to implement.

2.10.1 Web Portal

In this section we describe our web portal and how the player will interact with it.

The page should be very simple and easy to navigate as it may have many different types of users. The web portal is the website where the player can go to play our browser-based games. It will consists of the following pages: A page where the player can login on his/her account, a page for creating an account, a page for browsing and choosing between our different games, a highscore page, a news page, a community page and a profile editing page. The website must also contain a part of the game itself, in the form of a page that represents a space-station dock, where the player can alter and improve his/her space ship. Regardless of what page is currently displayed, it will always have the following elements: A banner and some link in the upper part of the page, some navigation buttons right under that, a line in the bottom of the page with a few more links and two orange boxes in the right side, where the player can read a few things about the project, as well as a view of the top 10 high score list from the asteroids 2k game.
The login page will be very simple. It should give the player the option to type his/her account name and password and logging into the site, or to enter the account creation page.
At the account-creation page, a new player will be able to enter his/her username, real name, account password and email. This data will be transmitted to the database when the "create user" button is pressed. The player must enter the password twice, as well as enter an email address of the following form "<...>@<...>.<com/dk/uk/...>" (i.e. jensk@asteroids2k.com), to pass the validation check. The reason for the validation check is to ensure that the player does not mistype his/her password and to ensure that it is actually a real email address that has been entered.
If the player wants to change his/her email, password or anything else on his/her account, she [2] can do so on the profile editing page. One simply changes the information in the right box and pushes the save changes button when one is done.
The player will be able to view news articles about the games at the news page,

[2] From now on, we will referrer to he/she as she.

which will be very simply organized. Each news article will have a title, an author and a date, and all the articles will be sorted by date.
The games page will be the place where the player can choose the game he/she wants to play. All the games are organized in two columns. Each game will have a box in which there will be the following items: A screenshot of the game that captures the players attention, a price tag, if we ever wanted to make money on the project, a title and a short description of what the game is about. When the player decides what game she wants to play, she simply clicks on the game box. Of course we will only design one actual game for this project, but the web portal will potentially be able to contain multiple browser-based games.
The highscore page will contain all the highscore lists from the different games. The page will contain a list of the games on the site, and by selecting them, their respective highscore lists will appear.
The community page will contain forums and detailed information about each user. This will allow the players to interact with one another and perhaps plan gaming sessions.
The docking station page is a part of the Asteroids 2k game itself. Here the users will be able to view information about his/her own spaceship, scores, win, losses, etc. This page also has an animated illustration of the players spaceship and it's current condition.

2.10.2 Asteroids 2k

In this section we describe the game and what features we want it to have.

Asteroids 2k will be a remake of the original Asteroids with new features. Game mechanics will be similar to the original, as described in section 1.3. The game will run as an applet at the webportal and will require an internet connection to play. The player will control the spaceship with five buttons. Left and right-arrow keys rotates the ship, the up-arrow gives thrust, the space-bar fires projectiles and the Ctrl-button will activate hyperspace. After every game played the points will be stored on the webpage and can there be used to purchase upgrade modules for the spaceship. The player will have a hangar used for storage of the ship and modules, and a hanger used for fitting or changing the modules on the ship. There will be a shopping section on the webpage where players spend their points on various modules for the ship. The spaceship can be upgraded in four main categories:

- Propulsion modules
  Will enhance the thrust of the ship, top speed and the rate of which it can rotate, making it easier to "dodge" the asteroids.

- Offensive Modules
  Includes missile-turrets with various abilities. Some can fire more than one projectile at once and in different angles. For example firing projectiles out the front and back of the space ship. Other turrets can increase the number of projectiles allowed in flight. Some turrets combine these things. Other turrets fires an omnidirectional energy blast instead of projectiles.

- Ammunition
  Ammunition, or ammo is used with the turrets and different ammo gives different kinds of projectiles. Upgrades will increase velocity and flight time of the projectiles.

- Defensive Modules
  These modules will make the spaceship stronger and able to withstand more shots from enemy UFOs and be able to survive more collisions with asteroids.

Not all modules are listed here because there will be many different versions of them. For example the shield module will have a small version that allows the player to collide with an asteroid once without dying. A bigger version might allow three collisions without destroying the spaceship. Another example could be different versions of the propulsion modules. The small ones gives a little boost to the thrust, while the big ones gives a larger boost. The price for these modules will of course increase the better the module is.

2.10.3 Product summary

Throughout this technical analysis, we have studied the basics of the technologies we need to create our product. Furthermore we have described our product with all the features we wish to implement. The website will be implemented in C# using sessions and SQL. The Game will be implemented in Java.

Chapter

3

Problem Solving

3.1 Introduction

Problem solving is mainly about the solving and developing phase that we endured during this project. First of all we need some diagrams for both the database and Java Applet. If such preparations are not made before the coding have started, we will often have to rewrite some code or rebuild some tables since some part of the the creation did not work as intended. Afterwards we need to figure out which type of design we want on both the webpage and the Java applet. We want the game to be like the original game and we also want the usability of the page/game to be acceptable. After the diagrams and design are ready we can head towards the technical problems of our problem solving. Our product contains many functions and features. We will pick out a some parts within the Website and Java Applet and describe them in detail.

3.2 Product demarcation

This section will describe the parts of the product that has been left out, because of the restriction of this project.

3.2.1 Web portal

To make the best of the time we have, we needed to focus on the essential things, those that could not be sacrificed. The intented community page has been left out, because it is not essential. We have omited the space-dock part and shopping part of the site (intented to be a "garage" where a player could upgrade the ship for earned points) simply because, this is a big and time consuming feature to implement both in the website and the game applet. Another part of the web portal, the API page, is made to a working state, without the necessary security precautions.

3.2.2 Asteroids 2k

As with the web portal we need to focus our attention on the essential part, to make a functional game with the restrictions of our project. The goal is to have a working, classic asteroids game, with basic features:

- Flying capabilities, throttle etc.
- Shooting features.
- Lives
- Points

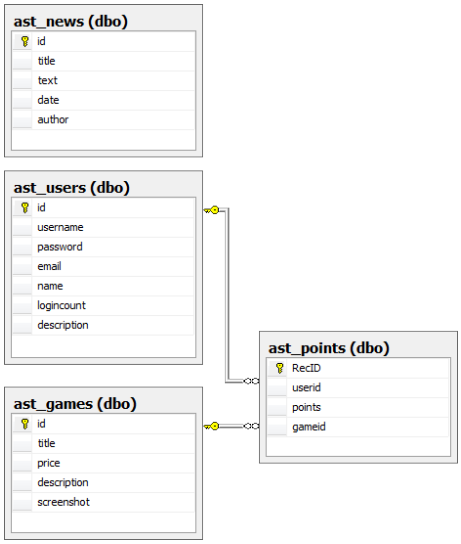
Furthermore we want the game applet to communicate with the web portals API page.
As mention above, we are not going to make it possible to upgrade the functions of the ship.

3.3 Diagrams

This section contains the diagrams of the database, website and Java Applet.

3.3.1 Database

First of all we need to find out which tables we require. We need a place to store news, different games, user informations and user points.



The diagram above shows how we placed all this in the database and the relations is generated by Microsoft SQL Server Management. All the table names starts with 'ast_' to avoid name clashes with tables from other projects in the database. When adding a predefined string to all the names we ensure that tables won't conflict with each other. All tables contains a id which is the primary key. It is named RecID if it is not used for anything else than having a unique number for a row.

ast_news
This is a table for itself, since it does not have anything to do with users, points and the different games. It contains title, text, date and author. A thing, which would be logical if developing further on the website, would be to relate the author to users; however, then we would need some sort of privileges on the users. Every user should not be able to create news which is why we have excluded this relation for now.

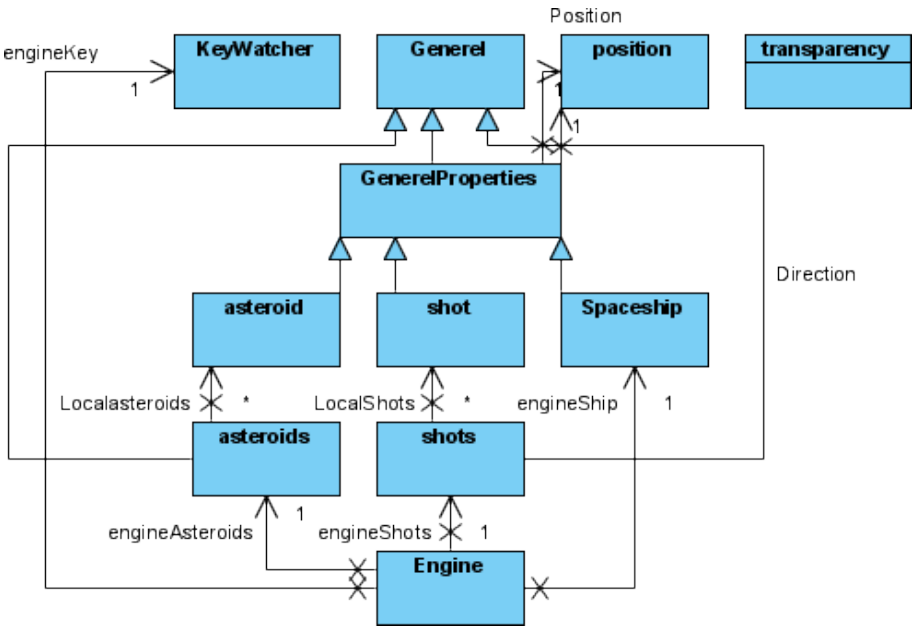
ast_games
The idea of this table is to contain informations about all the games we have; one game for now but the webpage will work as a game portal as stated earlier. For now this table contains a title of the game, a price which is zero for asteroids, a description and a path for a picture of the game.

ast_users

This table contains typical things for users like username, password, email, name, description. It has no privileges attached so all users will be handled like normal users.

ast_points
This is the core of our game at the moment. This is where the points for each game is stored and therefore it is related to both ast_games and ast_users. There should always only be one row for each user and game - if there is a row, the points will be added to the previous points.

3.3.2 Java Applet



The diagram above shows how the classes in the game are structured and the lines is generated by Visual Paradigm directly from our source. The lines with a blue arrow represents that the class inherit from the other class. The lines with a cross and a normal arrow represent, that a variable in that class uses the other class. The name on the variable is written next to the line. This section is split up in the following two subsections: Stand alone classes and drawing classes.

3.3.2.1 Stand-alone classes

The idea is to create classes an object oriented way. The stand-alone classes we need are KeyWatcher, to handle key press, Position to handle positions, Transparency to handle transparency on pictures and Engine to hold all the classes together and run the loop of the game itself. This is why we need these 4 classes.

KeyWatcher
We need a place to handle keystrokes and mouse clicks. The best way would be to build a class for this purpose. This will avoid to much code within the engine itself.

Transparency
Pictures, that are loaded into a program, does not come with a transparent background even though it is created in formats like gif and png. That is why we need a mask color for all the pictures, to ensure the transparency of the pictures at the right places. Otherwise we would have a square around our ship, which will be visible when flying close to the asteroids.

Position
The first class we need to build is the Position class. Since there is no concept of vector points in Java, we need to create a class that can encapsulate this functionality. With this class we build a new point with X and Y coordinates with many decimals and furthermore we can make functions for position calculations within it.

Engine
The engine is where everything is handled. The place where all classes can communicate with each other and the main classes are used. This class will contain time calculations and our main loop. It can be compared with a control panel in a plane where everything is controlled.

3.3.2.2 Drawing classes

The classes that show the items on the screen can be grouped into two categories. One of them is the items themselves, how it should be drawn on the screen. The other category is the list, which contains an array of the items, and processes how the items should react, move and collide with other objects. Though these classes are divided into two categories, they still have something in common. To do this we need a generel class.

Generel class
This class will contain variables which are important for all the classes within the drawing procedure. It will also contain generel functions, like the advanced random function etc.

List classes
Shots & Asteroids
The Shots and Asteroids classes inherits from the Generel class. These classes contains an array of items. They will handle the movement and collision based on the time difference given from the engine. They will also handle the way the map is created to begin with and some of the limitations in the game like maximum shots and asteroids.

Item classes
GenerelProperties
Inherits from Generel, and contains everything items have in common. They all need a position, direction and rotation which will stored in this class.

Shot & asteroid
Inherits from GenerelProperties. The only thing which really is left in Items is the way to be drawn.

Ship

Inherits from GenerelProperties. This class is a bit special because it will both handle everything the other items are handling, and it will also handle what the lists are normally controlling. This is because we only have one ship in our game. If we are going to make enemy ships, this class should be changed to look like an item class and a list class should be controlled to handle an array of this class.

3.4 Technical problems

The following section describes some of the technical problems we faced when developing the website and Java Applet.

3.4.1 Web portal

This section describes how we solved some of the problems we faced when developing the website.

The website is developed in Microsoft ASP.NET 2.0 with C# as the language of choice.

We have chosen to protect our games with user login protection. That means most of our pages can not be seen, unless the user has an registered account on our site.

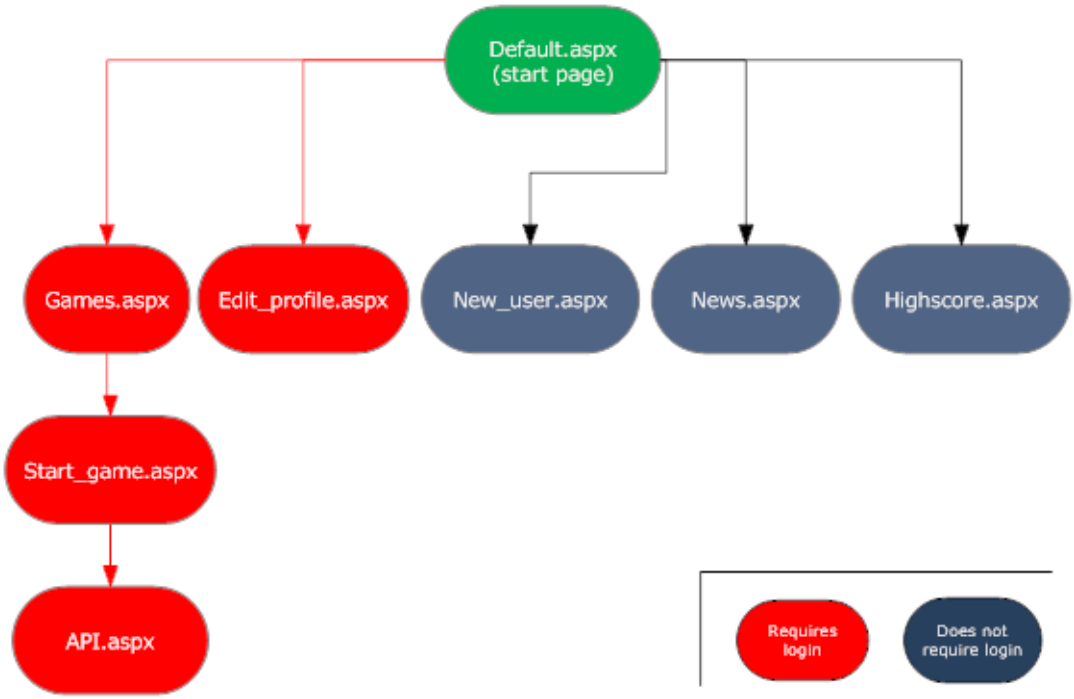


Figure 3.1: Overview of web portal

If the user views a page with protection, she will be redirected to the login page. If a user views a page that does not require login, she will be able to view the page, but a message telling the user to login will be shown.
We have a highscore list in the right column. This is implemented in the MasterPage.master with an ASP.NET Repeater control, using the following SQL query:

Listing 3.1: Highscore SQL query

```
SELECT TOP (10) points, gameid, username FROM ast_points INNER
    JOIN ast_users ON userid = id WHERE gameid = 1 ORDER BY points
     DESC
```

As with the left column high-score, we have implemented the news page and the game page with the use of ASP.NET Repeater control too. The highscore page, where the user can select which game she would like to view the highscore for, we have made an ASP.NET GridView control to display all games on the site, when selected, another GridView control will appear, showing the high-score list for the particular game.

In the following sections we will go into details with the larger and more complex subjects of the web portal development.

3.4.2 Master Pages

When developing a website it is often preferred to have an overall design layout, that apply to all the pages. To accomplish this, Mircosoft ASP.NET offers a concept that is named "Master Pages". A Master Page holds all the elements that must appear on all the individual pages, such as the navigation menu, footer, header etc. When designing a Master Page with all the needed elements, there can be a number of placeholders for content. A placeholder is an element into which the individual pages "inserts" it's content, as illustrated in figure 3.2.

The Master Page has defined a placeholder with:

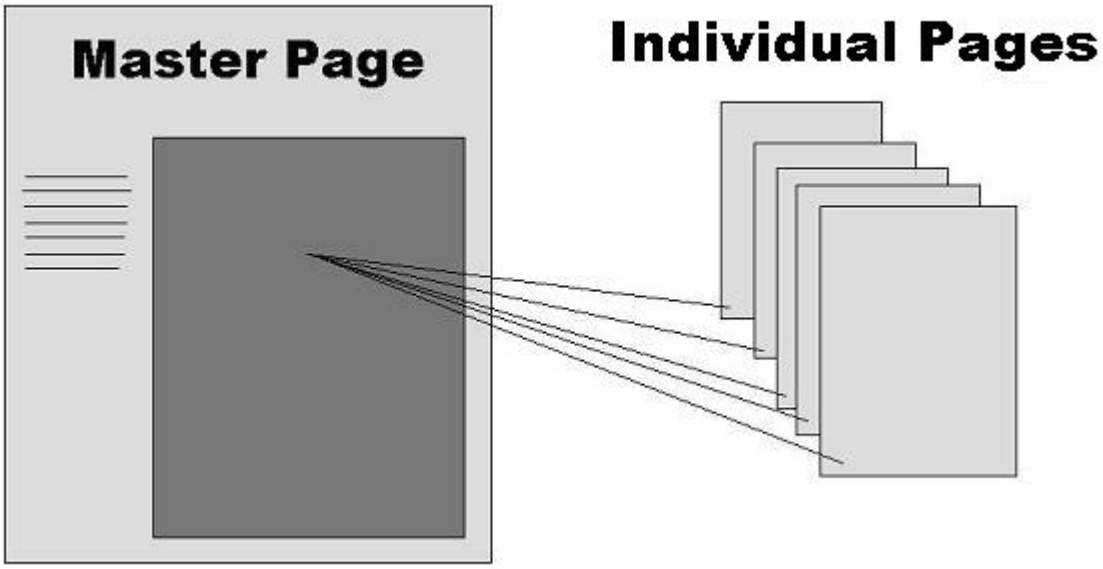
```
<asp:contentplaceholder id='ContentPlaceHolder1' runat='server'>
</asp:contentplaceholder>
```

The ID-attribute specifies the id used for the individual pages to refer to when inserting content into this area. The individual page can insert content into the master page like this:

Listing 3.2: Content Place Holder

```
<asp:Content ID='Content1' ContentPlaceHolderID='
    ContentPlaceHolder1' Runat='Server'>
        Hello World!
</asp:Content>
```

Figure 3.2: MasterPage



This will insert the text "Hello World!" into the content placeholder in the Master Page. A Master Page is not limited to a single placeholder, for more fancier designs, multiple placeholders can be used.

3.4.3 Website design

The primary goal of our project is the contextual and technical process, not the grapichal design. To make our site attractive to the end-user we need to have an appealing design on the site. To keep focus on the contextual and technical part we decided to go with a "Open Source"-design. The idea of "Open Source" in this context is not that we can get the source code of the HTML-document and CSS-file, because everyone can copy the source of a website by viewing the source in a browser! In this context "Open Source" refers to commonly used Open Source licenses, such as GPL-, LGPL-, BSD-, Apache-license which gives everyone the right to copy, modify and use the design. This gives us the opportunity to have a great and appealing design without spending much time on it. We downloaded our design from www.oswd.org - Open Source Web Design.
To implement the design into our application we took the HTML source and placed in our Master Page and placed a ContentPlaceHolder-element in the appropiate place. Then we included the accompanying CSS file with a link tag as described in section 2.3.3 on page 28:

```
1 <link rel='stylesheet' type='text/css' href='stylesheet.css' />
```

This is all that is needed to implement the design with our Master Page in ASP.NET.

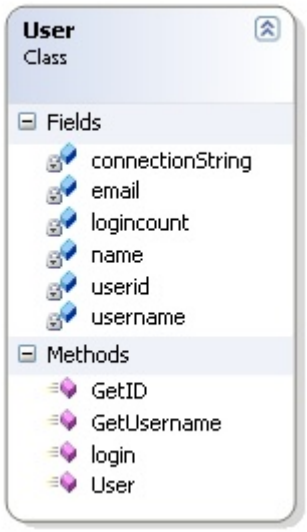
3.4.4 Users

This section describes how we solved the problems regarding users when developing our site. We cover the User class, creating a user, login and logout.

3.4.4.1 The User class

We have created a class called User, that contains five methods we need to have a functioning user system. The User class is in the namespace Asteroids in the C# file User.cs. This is done to prevent any conflicts with existing system classes. We will now describe the four methods one at a time. Below is a diagram over the class User.

User class diagram



We start the user class by creating a connection string where we fetch the connection information for the SQL server. Then we create five variables where the data received from the database will be stored. These variables are all marked private so they cannot be accessed from outside the User class. Now that the connection string and the variables have been created we start defining the methods. The first one is the user method.

Listing 3.3: user method

```
1 public User(string u) {
2 SqlConnection connection = new SqlConnection(connectionString);
3 connection.Open();
4 SqlCommand command = new SqlCommand("SELECT * FROM ast_users WHERE
      username = @username", connection);
```

```
5  command.Parameters.Add(new SqlParameter("username", u));
6  SqlDataReader reader = command.ExecuteReader();
7  reader.Read();
8  username = reader["username"].ToString();
9  real_name = reader["real_name"].ToString();
10 email = reader["email"].ToString();
11 logincount = Int32.Parse(reader["logincount"].ToString());
12 userid = Int32.Parse(reader["id"].ToString());
13 reader.Close();
14 command.CommandText = "UPDATE ast_users SET logincount =
       logincount+1 WHERE username = @username";
15 command.ExecuteScalar();
16 reader.Dispose();
17 command.Dispose();
18 connection.Dispose(); }
```

In the first six lines we use the conection string from earlier to connect to the database, select all the data from the table ast_users where the username that equals the argument received from the login form. The selected data is temporarily stored in a array created by the reader. In the lines 7-12 we convert the data in the reader to strings and save them in the variables we created in the beginning of the class. The next two lines increments the login count for the user by adding 1. The last three lines disposes the reader, command and connection so they do not create any conflicts when used again in the other methods. Next up is the login method.

Listing 3.4: login method

```
1  public static Boolean login(string username, string password) {
2  SqlConnection connection = new SqlConnection(connectionString);
3  connection.Open();
4  SqlCommand command = new SqlCommand("SELECT username FROM
       ast_users WHERE username = @username AND password = @password
       ", connection);
5  command.Parameters.Add(new SqlParameter("username", username));
6  command.Parameters.Add(new SqlParameter("password", password));
7  SqlDataReader reader = command.ExecuteReader();
8  if (reader.Read())
9          return true;
10 else
11         return false; }
```

This method is used when the user presses the Log In button on the welcome page. The method takes two parameters, username and password. On line 4 we set the SQL statement to select the username from the ast_users table where the username and password matches the variables @username and @password. Then, on lines 5-7, we insert the two parameters into the SQL statement and execute the SQL reader method. The last four lines define the output of the method. If the reader is not empty, there has been found a user in the database where both username and password match the parameters. If this is the case then the boolean value true is returned, if not then either the password or username are incorrect and the value false is returned. If that is the case the user gets an error message and is not logged in.

Listing 3.5: GetUsername & GetID methods

```
public string GetUsername() {
        return username; }
public int GetID() {
        return userid; }
```

These last to methods return the the values of the two variables username and userid. These are the variables into which we saved the data returned from the user method. We use the GetID method on the main page to save the userid in the session array. The GetUsername method is used on the main page to display a welcome message containing the users username.

3.4.4.2 Creating a user

Before users are able to access the games they have to create an account. This is done on the page New_user.aspx. There is a link to this page below the login form on the start page. The New_user.aspx page consists of a form with six text boxes and one submit button. In front of each text box is a label describing what the users should type, like password, user name and email etc. We have implemented validators that checks the following; the passwords match each other, the name and user name fields are not empty and that the email is valid. When the user has filled out the form correctly and presses the "Create user" button the data from the text boxes are saved in the database and the user is transferred to the game page. We will now describe the process step by step.

When the "Create user" button is pressed it calls the Createuser method from the C# file New_user.aspx.cs. The Createuser method starts out by creating a connection to the database and checks if the user name already exists. This is done in the following code:

Listing 3.6: Create user 1

```
SqlConnection connection = new SqlConnection(connectionString);
connection.Open();
SqlCommand command = new SqlCommand("SELECT username FROM
    ast_users WHERE (username = @username)", connection);
command.Parameters.Add(new SqlParameter("username", NewUserName.
    Text));
SqlDataReader reader = command.ExecuteReader();
```

First the the connection is fetched from the web.config file where the information on the database is stored, and the connection is opened. Then the SQL command is set to select the username from the ast_users table where the username equals the variable @username. By adding "connection" to the end of the SQL command we ensure that we use the connection we opened earlier. Then we take the new username from the textbox, save it in the variable @username and then insert it into the SQL commmand. Last we use the SQLDataReader method to read the data we select in the SQL command and save it in the variable reader. If the new username exist in the database then the SQLDataReader returns the

username, but if it does not then the reader returns nothing. This leads us to the next part of the Createuser method, an "if else Statement". This statement runs the "if" part if the SQLDataReader returns a value and the "else" part if it does not.

Listing 3.7: Create user 2

```
1 if (reader.HasRows){
2   usernameerror.Text = "Username already exists.";
3   reader.Close();
4   reader.Dispose();}
```

If the username exists in the database, then an error message is shown beside the username field showing the user name is already taken. Then the reader is closed and disposed so we can use it again later with the same connection.

Listing 3.8: Create user 3

```
1 else  {
2 reader.Close();
3 reader.Dispose();
4 command.Parameters.Clear();
5 command.CommandText = "INSERT INTO ast_users (username,password,
      email,name,description) VALUES (@u,@p,@e,@r,@d)";
6 command.Parameters.Add(new SqlParameter("u", NewUserName.Text));
7 command.Parameters.Add(new SqlParameter("p", Password.Text));
8 command.Parameters.Add(new SqlParameter("e", email.Text));
9 command.Parameters.Add(new SqlParameter("r", RealName.Text));
10 command.Parameters.Add(new SqlParameter("d", Description.Text));
11 command.ExecuteScalar();
12 User user = new User(NewUserName.Text);
13 Session["user"] = user;
14 Session["userid"] = user.GetID();
15 HttpContext.Current.Response.Redirect("MainPage.aspx");
```

The else statement is where the data from the textboxes is stored on the database and the user is created. We start by closing and disposing the reader so it is ready to be used again. We also clear the parameters from the SQL command we used earlier. Then we create a new SQL command telling the database to do the following: Insert the five variables into the table ast_users in the according fields. Meaning the @u variable is stored in the "username" field and @p is stored in the "password" field etc. In the next five lines we save the data from the textboxes and save it in their respective variables. Then we execute the SQL command with the ExecuteScalar method. The data is now stored in the database and the user is created. Line 20 sets the user data from the database by using the user method we defined in the User class. To automatically log the user on when the account is created, we store the user data in the two session elements, user and userid. When the session has been saved we redirect the user to the mainpage where the games are listed. After the else statement we close and dispose the SQL connection.

3.4.4.3 Login and logout

One of the other problems we faced when developing the site was that we needed a secure way for users to log in and out of their account. The login is done by entering username and password in the form on the startpage. When the user presses the "log in" button the login method from the User class is called. The logout function uses the LogoutClick method from the MasterPage.master.cs file to clear the session element called user.

Listing 3.9: LogoutClick method

```
protected void LogoutClick(object sender, EventArgs e) {
    Session["user"] = null;
    HttpContext.Current.Response.Redirect("Default.aspx"); }
```

This method clears the session element "user" by setting the value to null. When the session is cleared the user is redirected to the login page.

3.4.5 API

In this section we describe how we solved the problem regarding how the applet communicates with the site, with the use of an API page.

When the player has played a game, we need the applet to update the new score for that particular player, into the database. In the database we have made a table called ast_points (see section 3.3.1), which is the table we want our applet to upload the information to. Instead of uploading the information directly from the applet, it is uploaded to a API web page instead. That page then uploads the information to the database. The applet can communicate with the page in two different ways: query strings (GET parameters) and web forms (Mainly POST parameters). Query strings basically uploads parameters via address line e.g. "www.asteroids.unif.dk/api.aspx?gid=1&uid=5&points=6754" in this example the player played the game with the game ID 1, the player had the user ID 5 and 6754 points are send. Web forms basically do the same thing, but with the use of a form instead of parameters in the browsers address line. The form could be something like the ones used in the new_user.aspx and edit_user.aspx pages.
When a game is over the applet fills the form with the use of the following code:

Listing 3.10: API 2

```
String data = URLEncoder.encode("gameid", "UTF-8") + "=" +
    URLEncoder.encode(Integer.toString(m_gameID), "UTF-8");
data += "&" + URLEncoder.encode("userid", "UTF-8") + "=" +
    URLEncoder.encode(Integer.toString(m_userID), "UTF-8");
data += "&" + URLEncoder.encode("points", "UTF-8") + "=" +
    URLEncoder.encode(Integer.toString(m_points), "UTF-8");

URL url = new URL("http://asteroids.unif.dk/api.aspx");
URLConnection conn = url.openConnection();
```

```
7  conn.setDoOutput(true);
8
9  OutputStreamWriter wr = new OutputStreamWriter(conn.
       getOutputStream());
10 wr.write(data);
11 wr.flush();
```

When the applet has filled the web form, the api.aspx page saves the information in some string variables:

Listing 3.11: API 2

```
1 string gameid = HttpContext.Current.Request.Form.Get("gameid");
2 string userid = HttpContext.Current.Request.Form.Get("userid");
3 string points = HttpContext.Current.Request.Form.Get("points");
```

Then the api.aspx page inserts the information into the database:

Listing 3.12: API 3

```
1 SqlConnection connection = new SqlConnection(connectionString);
2 connection.Open();
3 SqlCommand command = new SqlCommand("UPDATE ast_points SET points
      = @p + points WHERE userid = @uid AND gameid = @gid",
      connection);
4 command.Parameters.Add(new SqlParameter("uid", userid));
5 command.Parameters.Add(new SqlParameter("p", points));
6 command.Parameters.Add(new SqlParameter("gid", gameid));
```

3.4.6 Java Applet

This section consists of two subsections. A list of problems, where we state the problem and give an idea how it could be solved. And samples, where we show some of the solutions we have created.

First of all to ensure a "hands-on" approach we have chosen not to create any menus in-game. This will allow the user to jump right into playing the game. The only thing needed is to activate the Java applet. Furthermore we have chosen to control the game with the arrow keys and spacebar. This increases the usability because these buttons logically describe what actions the ship will do when pressed. For example right and left arrow-key rotates the ship right and left and the up-arrow makes the ship go forward. The spacebar is one of the biggest buttons on the keyboard and is easy to hit in a hurry. The Java game applet is using object oriented programming and consists of multiple classes. This makes it easy to control each type of unit in the game. For example the spaceship, asteroids and projectiles are classes. We have classes with some general properties which other classes inherit from. To get an overview we will describe each class and how they interact with each other and the user.

- Engine.java
  This is the "main" class. This class contains the following methods. Init(), Start(), Stop() and Run(). Init() initializes every component needed for

the game to run when the applet is loaded in the browser. It sets up some local variables for example the background where all graphics are drawn upon. It enables user input via the KeyWatcher.class and makes mouse input possible. A click with the mouse is needed to start the game. Init() also declares the backbuffer which is used for drawing the graphics before displaying it on the screen. This is done to minimize latency caused when the program is drawing directly to the screen. The Start() method checks for an already running thread and if none is found it starts a new thread making the program able to run. Stop() stops the thread when the user leaves the page. Run() is used for outputting graphics. It draws an image to the screen with a 50ms interval. First it draws a black background and then invokes the painter function which draws the spaceship, asteroids and projectiles on the screen. Overall the Engine.class controls the game environment by moving and drawing the objects in the game.

- KeyWatcher.java
  The KeyWatcher checks whether the user is pressing a key. It only reacts to the four keys mentioned earlier so pressing some wrong keys (except generel hot-keys) wont affect the program.

- Position.java
  This class is used for creating a position for the spaceship, asteroids and projectiles. It contains vectorbased calculations used in the game. For example the vector direction and position-vector.

- Generel.java
  This class contains global variables such as height and width for the output screen. Furthermore it contains a random number generator. Shots.class Asteroids.class and GenerelProperties.class inherit from this class.

- GenerelProperties.java
  This class is used for storing the position, direction, rotation angle and texture. Spaceship.class Shot.class and Asteroid.class inherits from this class and the class itself inherits from Generel.class. GenerelProperties.class also includes the move() function which is used to move objects in the game.

- Spaceship.java
  This class forms the spaceship. It contains lives, score and a image used when the spaceship is accelerating. It takes input from the KeyWatcher.class and moves the ship according to these inputs. To control the acceleration of the ship, this class also contains the function which adds direction vectors. It adds the direction vector of where the ship is pointed when accelerating to the direction vector the ship already has. Deacceleration is also controlled by this class and is the reverse version of the acceleration function. It decreases the length of the ships direction vector over time. The score is saved in this class and altered in the asteroids.class. Spaceship.class also draws the image of the ship into the

back-buffer. Another important function is the one which checks for collisions between the spaceship and the asteroids. It calculates the distance between the center of the spaceship and the center of the asteroid. If this distance is less than the sum of half the ships width and half the asteroids width, the ship will lose one life and disappear. When this happens the class checks the spawn area for asteroids. This is done to prevent the ship from spawning right into an asteroid. When the spawn area is clear the reset function will be invoked and the ship will appear in the middle of the screen again.

- Shot.java
  This class forms the projectile with a flight time and a texture image. The projectile is also drawn into the background-buffer just like the spacesip.

- Shots.java
  Keeps track of all the shots. There can be up to five shots at a time and this class moves the projectiles in the angle it was fired and calculates the velocity of the projectile based upon the velocity of the chip. It invokes the individually drawn functions of each shot.

- Asteroid.java
  This class contains a size variable that is used to determine what texture image is used for drawing the asteroid. The class also draws the asteroids into the back-buffer like the other classes.

- Asteroids.java
  Keeps track of all the asteroids. It holds a level variable that determines the number of asteroids spawning at the beginning of each level. Here an important function of this game checks collision between the projectiles and the asteroids. It detects collisions and if possible splits the asteroid into smaller asteroids. This can of course not be done with the small asteroids, only the big and medium sized ones. The last function is the one that increases the number of asteroids for every level and randomly determines where they spawn. The score variable in the spaceship.class is increased every time a projectile collides with an asteroid.

- Transparency.java
  This class is used once when the texture image is loaded into the back-buffer. It replaces the color red in the textures with nothing making the red areas transparent.

- UploadScore.java
  This class is used for interaction with the database of the gameportal. It uses three variables when uploading. A game-id, user-id and the players score. The id's are used to determine what row the score is added to.

3.4.6.1 Sample solvings for occurred problems

During the time of the game programming we had some difficulties with some parts of the programming. Some problems caused us to think more than twice

before continuing the programming. We will not go through all of them but we will list some of the bigger problems and afterwards explain a few of them in details.

- Move objects
  This game relies heavily on the moving objects on the screen. So referring to the section about vectors in games, vectors should be used here to calculate trajectories. See section 2.10.2. How did we solve this? We created the move() function to update the position of every object in the game every 15ms. When the program runs perfectly the move() function is invoked about 66 times per second. The length of every move is controlled by the _TimeDiff which is longer or shorter according to the number of frames per second. Many frames and therefore a short _TimeDiff means short moves and vice versa. The movement is calculated by using two vectors. A position-vector and a direction-vector. The tail of the position-vector is at the upper left corner of the screen and the head is where a given object is. This position-vector is very similar to a point in a coordinate system. The tail of the direction-vector is where the object is and it points in the direction the object is going. We use vector-addition every time the move() function is invoked. The function modifies the length of the direction-vector with _TimeDiff and adds it to the position-vector. This is stored as the new position-vector. The move() function is also moving objects when they hit the edge of the screen. As mentioned earlier the asteroid field wraps around the screen and to do this we simply move objects to the opposite edge. For example if the spaceship flies into the right edge of the screen it will be moved to the left edge of the screen instantly. In the following code line two to nine is used for the "'wrapping"' and line ten and eleven is used for the normal movement inside the playing field.

```
1  public void move(double _Length){
2          if(Position.getX() + this.Texture.getWidth(null)/2 >=
               getWidth())
3                  Position.setX(Position.getX()+Direction.getX
                       ()*_Length - getWidth());
4          if(Position.getY() + this.Texture.getHeight(null)/2
               >= getHeight())
5                  Position.setY(Position.getY()+Direction.getY
                       ()*_Length - getHeight());
6          if(Position.getX() + this.Texture.getWidth(null)/2 <=
               0)
7                  Position.setX(Position.getX()+Direction.getX
                       ()*_Length + getWidth());
8          if(Position.getY() + this.Texture.getHeight(null)/2
               <= 0)
9                  Position.setY(Position.getY()+Direction.getY
                       ()*_Length + getHeight());
10         Position.setX(Position.getX()+Direction.getX()*
               _Length);
11         Position.setY(Position.getY()+Direction.getY()*
               _Length);
12 }
```

- Acceleration and deacceleration
  The spaceship needs to have a non-instant acceleration to make the gameplay work. We mentioned earlier that it was important to control the spaceships trajectory as the player and this should not be made to easy. So to make it both a challenge and a bit more realistic we added an acceleration based upon acceleration in real life. The Acceleration() function modifies the direction-vector of the spaceship when the player holds the thrust-key and the deacceleration modifies the direction-vector when the thrust-key is not pressed. When Acceleration() is invoked a new vector is formed. It is a vector that points in the direction the spaceship is pointed and this vector is controlled by the _TimeDiff. _TimeDiff is multiplied with the length of the vector and the vector is added to the direction-vector of the spaceship. This would of course make the ship possible to fly with limitless speed so we had to check the length of the direction-vector and make a limit. DeAcceleration() works almost the same way. But where Acceleration() adds a vector to the direction-vector, DeAcceleration() justs decreases the length of the direction-vector. Noteworthy is that _TimeDiff is multiplied with two in Acceleration() and divided by two in DeAcceleration(). This means the ship accelerates faster than it deaccelerates. The code for these two functions looks as follows:

```
1  public void Accelerate(double _Length)
2          {
3                  position PushDirection = new position();
4                  PushDirection.SetNewLength(_Length * 2, this.
                       getRotationAngle());
5                  this.getDirection().setX(this.getDirection().
                       getX() + PushDirection.getX());
6                  this.getDirection().setY(this.getDirection().
                       getY() + PushDirection.getY());
7                  if(this.getDirection().GetLength() > 500)
8                          this.getDirection().SetNewLength(500,
                               this.getDirection().GetAngle());
9
10                 accelerating = true;
11         }
12         public void DeAccelerate(double _Length)
13         {
14                 if(this.getDirection().GetLength() < _Length)
15                         this.setDirection(new position(0,0));
16                 else
17                         this.getDirection().SetNewLength(this
                               .getDirection().GetLength() -
                               _Length/2, this.getDirection().
                               GetAngle());
18
19                 accelerating = false;
20         }
```

- Shoot()

This function is invoked everytime the player fires a projectile. To make the game a bit more realistic the trajectory of the projectile is based upon the trajectory of the spaceship. A fast flying spaceship means fast flying projectiles. However it is based upon direction and if the ship is flying fast in one direction and fires in the opposite, the projectile will go slower. This is achieved in the following way. The first thing the function does is to get the direction-vector of the spaceship. Then a new direction for the projectile is created with a fixed length and the direction in which the spaceship is pointed. These two vectors are added and stored in the direction-vector for the projectile. Then the projectile is added to the projectile array with the properties calculated above. It is clear that if these two vectors point in opposite directions they will cancel out each other. It is however not possible to make the projectiles fly backwards or stand still because the fixed length of the projectile vector is always longer than the direction-vector of the spaceship. In the following code GetSpeed is used to get the speed of the spaceship and Direction is the direction of the new projectile.

```
1  public void Shoot(Spaceship _Ship) {
2          position GetSpeed = new position(_Ship.getDirection()
               .getX(),_Ship.getDirection().getY());
3          position Direction = new position();
4          Direction.SetNewLength(300,_Ship.getRotationAngle());
5          GetSpeed.setX(GetSpeed.getX()+ Direction.getX());
6          GetSpeed.setY(GetSpeed.getY()+ Direction.getY());
7          Direction.SetNewLength(GetSpeed.GetLength(), _Ship.
               getRotationAngle());
8          add(
9                   new shot(
10                          getWidth(),
11                          getHeight(),
12                          new position(
13                                  (_Ship.getPosition().getX()+
                                        _Ship.getTexture().
                                        getWidth(null)/2),
14                                  (_Ship.getPosition().getY()+
                                        _Ship.getTexture().
                                        getHeight(null)/2)
15                          ),
16                          Direction,
17                          _Ship.getRotationAngle(),
18                          Engine.shot,2)
19
20         );
21
22 }
```

- Checking for collisions
  A keyfunction in this game is a function which checks for collisions. If this was omitted, the gameplay was non-existing. For example the function checks if a part of the spaceship is touching a part of an asteroid. If it does, the spaceship loses a life and resets. Another example is when a

projectile hits an asteroid, then the asteroid is divided if possible or else it will disappear. The projectile will also disappear when hitting a target. To check if every shot is hitting any asteroids, we use two arrays - one with all the projectiles and the other with all the asteroids. The function runs through the asteroids array and checks if one of the projectiles in the projectile array is hitting an asteroid. Because we use pictures, we use an outline of the objects to check collisions with. This is not always exactly right because not all in-game objects are completely circular. A smarter method could be to use figures drawn by the program and then run through every pixel on the circumference of two objects to see if they cross each other. The code for checking if the spaceship is colliding with an asteroid is shown beneath:

```
1  public void collisioncheck (asteroids _asteroidsEngine) {
2   if (!dead){
3    for(int a = 0; a < _asteroidsEngine.getLocalasteroids().
         length ;a++) {
4     if (_asteroidsEngine.getLocalasteroids()[a] != null){
5      position distance = new position(
6       _asteroidsEngine.getLocalasteroids()[a].getCenter().getX
            () - this.getCenter().getX()
7          ,_asteroidsEngine.getLocalasteroids()[a].getCenter().
               getY() - this.getCenter().getY()
8      );
9      if(distance.GetLength()<=((_asteroidsEngine.
           getLocalasteroids()[a].getTexture().getWidth(null)/2)
           +(this.getTexture().getHeight(null)/2))){
10      Engine.soundCollision.play();
11      reset(true);
12     }
13    }
14   }
15  } else if(respawnTime <= 0 && freeSpawn(_asteroidsEngine))
16   reset(false);
17 }
```

- Checking Spawning Area
  When the spaceship has collided with an asteroid, it will disappear and then respawn after a given time. An early version of game made it clear that spawning directly within the area of an asteroid would cause an immediate collision and the spaceship would lose more than one life. This was very annoying and therefore we needed to do something about it. We made a function called freeSpawn() that checks the center area for asteroids. The so-called spawn area. If an asteroid is located inside the spawning area, the spaceship will not spawn until the spawn area is clear of asteroids. This means that an asteroid just outside the spawning area with a direction directly towards the spaceship is an immediate threat but can be avoided by navigating the spaceship around it. At least this gives the player a chance to avoid losing a life. Another solution could be letting the spaceship stay invincible in a few seconds and then the user would have time enough to navigate the spaceship to an area without asteroids.

The code for checking the spawn area is shown beneath:

```
1  private boolean freeSpawn(asteroids _engineAsteroids){
2   for(int a = 0; a < _engineAsteroids.getLocalasteroids().
        length ;a++) {
3    if (_engineAsteroids.getLocalasteroids()[a] != null){
4     double x1 = ((getWidth()/2) - getTexture().getWidth(null))
          ;
5     double y1 = ((getHeight()/2) - getTexture().getHeight(null
          ));
6     double x2 = _engineAsteroids.getLocalasteroids()[a].
          getCenter().getX();
7     double y2 = _engineAsteroids.getLocalasteroids()[a].
          getCenter().getY();
8     double afstand = Math.sqrt((Math.pow((x2-x1),2)+Math.pow((
          y2-y1), 2)));
9     if (afstand<=40+((_engineAsteroids.getLocalasteroids()[a].
          getTexture().getWidth(null)/2)+(getTexture().getWidth(
          null)/2))){
10     return false;
11    }
12   }
13  }
14  return true;
15 }
```

- DoUpload()
  As mentioned earlier on page 56 about the website API, the applet is filling a form and sends it to the API.aspx. This are done by using a function DoUpload(). The code of the function can be seen on listing 3.10. First the function encodes the data into form-data by using the URLEncoder function. This function needs a string and a encoding scheme - here we use the encoding scheme UTF-8. This translates the data into characters of the application/x-www-form-urlencoded format and hereby insures that the data are the same when they are received by the API. Then we are setting up a connection to the API and create an OutputStreamWriter which is used to stream the encoded characters by first converting them to byte characters and then streams (flushes) the characters to the url. At this point we have made a successfully upload to the API. For getting a response from the API we use an BufferedReader which uses an InputStreamReader to read the message. We are using a "while loop" to read every line in the message. The last thing we do in this function is to close the connections to the OutputStreamWriter and the BufferedReader. But why do we use this method? Well as we mentioned in the technical analysis about programming languages (see page 30), one disadvantage about Java applets was the possibility to decompile the applet. If we were using a direct connection to the database you could get access to the database by decompiling our applet. To avoid this we send the result of each game to an API which inserts the data into the database as described above.

Chapter 4

Conclusive part

4.1 Perspective

This project is not intended to be economicly profitable at all. The project could however easily be converted to a commercial product. The product is created using free and open source-software. This reduces the expenses in the development phase. Writing the code only has one major expense, which is the salary to the developers. When the project is developed and implemented on a website the costs are increased, because domain and server space is not free and have periodic expenses. Secondly the advertising for the website can be very expensive. To cover these expenses we can sell advertisements at the web portal. This could give us a fair income. Another method for earning money is to charge the players for playing games at the web portal. This would require the game to be of a certain quality, otherwise traffic would be reduced and the advertisements rendered worthless.
We could make several efforts to improve chances for commercial success for this project. For instance, we could develop more games to the web portal. The more games we offer on the site, the broader an audience we will cover. By having a large selection of games, different games are more likely to appeal to a certain player. The more traffic on the site, the more the advertisement will be worth. If we have a large amount of games, we could have an exclusive "VIP"-section, that offers a wider range of games, which only paying users are able to play. Another possibility is to add some upgrades to the shop, that cannot be bought with points alone, but with real money (i.e. by credit card). This is very popular in many online games (see section 1.6).
To increase the number of users on our portal, we could make competitions to attract new users, these could have some large prizes, for instance, money prizes, free VIP membership or simply small upgrades to their account.

4.2 Evaluation

In the two following sections we will evaluate our efforts to appeal to our targetaudience and how we made our game addictive. We will also cover the safety of the game.

4.2.1 Target group

Asteroids 2k is a game with a very flat learning curve. Controls are described in two sentences at the webportal and a single click with the mouse is all that is needed to start playing. There is no long manual to read and players can start immediately flying around the asteroidfield shooting asteroids. This is done to affect the casual gamer who tend to like a game without a long tutorial or storyline. Furthermore the game starts out easy with only a few asteroids in the field. The casual gamers like to start out easy and then work towards higher difficulty levels, and that is why we increase the numbers of asteroids each level making it harder and harder to avoid getting hit. It is required to do more complex maneuvers with the spaceship when the size of free space decreases,

but the gameplay stays the same.

To hit the hardcore gamer targetgroup we added extra elements to the game without changing the actual gameplay. These features will add new dimensions to the game without altering the "shoot asteroids" gameplay. Earlier we found out that the hardcore gamer likes to build something over a long time, so that is why we added the option to upgrade the spaceship with many different kinds of modules. This will in addition to the new ways of destroßing asteroids add the posibility to upgrade the spaceship over a longer time span. The hardcore gamer likes to be able to make some kind of progress in the game and make it suit his/her style of gameplay. To insure the hardcore gamers ability to make progress in the game, in addition to getting to a high level, we added these upgrade modules. With different kinds of offensive modules and ammuniton the player can modify the spaceship to fit that style of gameplay. The defensive player would prefer upgrading the spaceship's shield to be able to survive more collisions. There will allways be better modules on the horizon. This makes the player keen on returning, so that she can earn more points for new upgrades.

4.2.2 Addiction

The accumulation of points on the website adds addiction because of different reasons. Players have the possibility to upgrade the spaceship by spending the earned points on upgrades. This will make the game addictive since people who have played for a long time will have an advantage compared to new players. Competitive games are more addictive than other games and that is why we added the highscore on the webportal. The highscore makes players able to compete against each other making the game competitive. The forums will make players able to communicate and create a gaming community and thereby add even more addiction. We have not done anything to limit the addiction generating elements, because the game is, in its current form, still pretty trivial and we barely think that addiction could be a serious problem.

4.2.3 Safety issues

There are several issues regarding safety when it comes to the connection between the API web page and the Java Applet. We have only discussed how the safety could be handled, but we have not developed any actual safety. That is why it is currently very easy to cheat by creating a dummy page that can upload and change any score on the list. A potential hacker could then decrease the score of anyone on the high score list and increase his own score. The solution to increase the safety would be to either encrypt the data or create a validating sum of the data, but it is impossible to make it completely safe. Java Applets can be decompiled and they run on the clients' hardware, so it will always be possible to create a cheat. The only thing one can do as a developer regarding this problem is making cheating so difficult that the amount of time spent on it will not be worth it.

4.3 Conclusion

Throughout this project we have found the following solutions to the problems stated in our problem description.
In the problem analysis and product description, we found that our game should include the following feature to appeal to the selected target audience. A shopping section on the website that allows the player to buy upgrades for his/her ship. Such upgrades would improve the game play by adding new weapons, armor and movement elements. We would have implemented this feature in our game if we have had more time.
We have found that a good solution for creating a browser-based game that applies to both hardcore and casual gamers, is a game that contains the following; A flat learning curve, infinite game play, multiplayer features and the possibility to upgrade the players game character. Our product features a flat learning curve and infinite game play in the form of a unlimited number of levels. We would have implemented the other two features if we have had more time. If we had implemented all the features, we would have had a game that also could attract hardcore gamers.
We have created a website that allows us to easily add more games. This solution solves the problem of creating a game portal that supports multiple games.
We believe that with this product, we have made a good and functional web site, and a very enjoyable game, that people can play for hours. Throughout the research and development processes, we have learned much about programming and web technologies.

4.4 Terminology

MMORPG
A Massive Multiplayer Online Role-Playing Game is a game where a large number of players interact with each pther via the internet. See section 1.2.3.1 for more information.

Buzzword
A buzzword (also known as a fashion word or vogue word) is an idiom, often a neologism, commonly used in managerial, technical, administrative, and sometimes political environments. Though apparently ubiquitous in these environments, the words often have unclear meanings.

Gameplay
How the game is performed and how the player experiences it.

Inferface
An interface defines the communication between a piece of software and a hardware device, or a user.

Multiplayer Game
A multiplayer game is a video game where more than one player interacts with the same game environment at the same time.

Casual and Hardcore gamers
Casual gamers takes gaming as a hobby and hardcore gamers takes a more professional approach towards gaming. Casual and hardcore gamers are defined in section 1.7

Online Game
A game played over the internet, or another type of network.

Browser-based Game
A game that is played directly in an internet browser, maybe with the use of plugins.

Gameboy A portable hand-held videogame console.

JAMMA The Japanese Amusement Machine Manufacturers' Association (JAMMA) is a japaneese trade association. The JAMMA wiring standard made it possible to change the game of an arcade machine by just switching a circuit board.

Bibliography

[1] U. BBC News. Cash card taps virtual game funds. http://news.bbc.co.uk/2/hi/technology/4953620.stm, 2006.

[2] Berkeley.edu. Nimrod. http://jwgibbs.cchem.berkeley.edu/nimrod/, 2007.

[3] BLizzard. World of warcraft surpasses 9 million subscribers worldwide. http://www.blizzard.com/press/070724.shtml, 2007.

[4] W. W. W. Consortium. The css saga. http://www.w3.org/Style/LieBos2e/history/.

[5] Emarketer.com. Gamer demographics spread out. http://www.emarketer.com/Article.aspx?id=1004798, 2006.

[6] D. C. et.al. *AJAX In Action.* 2006.

[7] D. J. B. et.al. *Objects First With Java: A practical introduction using BlueJ.* 2003.

[8] E. N. et.al. *Beginning PHP5, Apache, And MySQL Web Development.* 2005.

[9] Forbes.com. Video game overuse may be an addiction. http://www.forbes.com/health/feeds/hscout/2007/06/22/hscout605801.html, 2007.

[10] D. Goodman. *JavaScript Bible.* 2001.

[11] O. M. Group. Unified modeling language. http://www.uml.org/, 2007.

[12] Gunslot.com. Top 25 best-selling video games of all time. http://www.gunslot.com/blog/top-twenty-five-25-best-selling-video-games-all-time, 2007.

[13] M. Hall. Using style sheets. http://www.brainjar.com/css/using/.

[14] http://www.freepatentsonline.com. Raster monitor for video game displays. http://www.freepatentsonline.com/4813671.html, 2007.

[15] Ign.com. Npd: Best-selling games of april 2007. http://ps3.ign.com/articles/789/789846p1.html, 2007.

[16] B. N. Laboratory. The first video game. http://www.bnl.gov/bnlweb/history/higinbotham.asp, 2007.

[17] U. o. J. Marko Siitonen. How does online gaming affect social interactions? http://www.sciencedaily.com/releases/2007/09/070915110957.htm, 2007.

[18] M. Miller. A history of home video game consoles. http://www.informit.com/articles/article.aspx?p=378141&rl=1, 2005.

[19] Netscape. Persistent client state - http cookies. http://wp.netscape.com/newsref/std/cookie_spec.html, 1996.

[20] NPD. Best-selling pc games october. http://www.guru3d.com/newsitem.php?id=6045, 2007.

[21] NPD. Best selling pc games of 2006. http://flashofsteel.com/index.php/2007/01/19/best-selling-pc-games-of-2006/, 2007.

[22] P. Rubens. In defence of computer games. http://news.bbc.co.uk/2/hi/uk_news/magazine/7013855.stm, 2007.

[23] D. Statistik. Befolkningens brug af internet 2007.

[24] U. Today. What games sold the most in 2006? here's a hint: it's pretty obvious what titles you'll see. http://arstechnica.com/journals/thumbs.ars/2007/01/02/6445, 2007.

[25] R. F. van der Lans. *Introduction to SQL: Mastering the Relational Database Language*. 2006.

[26] virtual economy.org. Interview with ccp. eve-online developers. http://virtual-economy.org/blog/interview_with_ccp_eve_currenc, 2007.

[27] Vækstfonden. Venture #2 - 2007: Oplevelsesøkonomi.
http://www.vf.dk/download_media.asp?media_id=2423.

[28] M. Ward. Gaming ís good for you?
http://news.bbc.co.uk/2/hi/technology/2744449.stm, 2003.

[29] Wikipedia. List of best-selling video games.
http://en.wikipedia.org/wiki/List_of_best-selling_video_games, 2005.

[30] t. f. e. Wikipedia. Video game addiction.
http://en.wikipedia.org/wiki/Game_addiction, 200.

[31] Wikipedia.org. Arcade cabinet.
http://en.wikipedia.org/wiki/Arcade_cabinet, 2007.

[32] Wikipedia.org. Arcade game.
http://en.wikipedia.org/wiki/Arcade_game, 2007.

[33] Wikipedia.org. Asteroids.
http://en.wikipedia.org/wiki/Asteroids_(computer_game), 2007.

[34] Wikipedia.org. Browser games.
http://en.wikipedia.org/wiki/Browser_game, 2007.

[35] Wikipedia.org. Casual gamers.
http://en.wikipedia.org/wiki/Casual_gamer, 2007.

[36] Wikipedia.org. Golden age of arcade games.
http://en.wikipedia.org/wiki/Golden_Age_of_Arcade_Games, 2007.

[37] Wikipedia.org. Golden age of arcade games.
http://en.wikipedia.org/wiki/Golden_age_of_arcade_games, 2007.

[38] Wikipedia.org. Hardcore gamers.
http://en.wikipedia.org/wiki/Hardcore_gamer, 2007.

[39] wikipedia.org. Raster graphics.
http://en.wikipedia.org/wiki/Raster_graphics, 2007.

[40] Wikipedia.org. Steve meretzky.
http://en.wikipedia.org/wiki/Steve_Meretzky, 2007.

[41] Wikipedia.org. Vector.
http://en.wikipedia.org/wiki/Vector, 2007.

[42] Wikipedia.org. Video game.
http://en.wikipedia.org/wiki/Video_games, 2007.

[43] D. Winter. Pong story.
http://www.pong-story.com/intro.htm, 2006.

[44] www.megagames.com. Who is a hardcore gamer? http://www.megagames.com/hardcoregamer.html, 2007.

[45] www.popcap.com. Casual gamers survey. http://www.popcap.com/press/?page=press_releases&release=casualgames_survey_9-13-06, 2006.