Freie Universität Berlin

Fachbereich Informatik und Mathematik

AG Technische Informatik

Dipl. Inform. Dipl. Kaufm. Tobias Fritsch

DOCTORAL DISSERTATION

Dissertation zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften im Fachbereich Mathematik und Informatik der Freien Universität Berlin

Next Generation Massive Multiplayer Games in a Mobile Context

Eingereicht bei: Prof. Dr.-Ing. Jochen Schiller
Prof. Dr. Mark Claypool

Dipl. Inform. Dipl. Kaufm.	**Tobias Fritsch**
Matr.-Nr.	3581795
Anschrift	Angermünder Straße 1a
	12305 Berlin
Tel.	030 74682460
E-Mail	T.Fritsch@gmx.net

Ort/Datum Berlin, den 22.09.2007

Gutachter:

Prof. Dr.-Ing. Jochen Schiller

Prof. Dr. Ing. Knut Reinert

Prof. Dr. Günter Rote

Dr. Achim Liers

Prüfungsdatum der letzten Prüfung: 29.05.2008

Acknowledgements

The work in this thesis presents the results of my scientific studies as a PhD student at the Freie Universität Berlin. Research was mainly conducted by the mobile gaming workgroup, which is a project of the Freie Universität Berlin's computer science department. I would like to take this opportunity to thank everyone for supporting me during this time.

First of all, I owe my deepest gratitude to Prof. Jochen Schiller, who was the primary adviser of this doctorial dissertation. His immense motivation always encouraged me to continue working unrelentlessly in order to achieve the scheduled tasks. By providing me with tremendous flexibility during my first three years, he further supported my inspiration and therefore significantly helped to complete this work. His knowledge of and insight into technical computer science is remarkable; he always managed to maintain an overview of the large scope of this work.

Furthermore, I would like to thank Dr. Hartmut Ritter for acting as my second adviser. His profound knowledge and practical experience helped me understand my task as a PhD student much more clearly. His amazing ideas and motivation substantially modified my work and gave me insights into the scientific importance.

I would also like to thank Benjamin Voigt for his inspiration over the last three years. I had the fortitude to have a friend who was consistently honest with me, hence giving me the clearest feedback to point out problematic fields. His support of and insight into the psychologically related fields of my research have been most valuable for me, thus I am very grateful.

On par with that, I would also like to thank Peter Harmjanz for his loyal support. His thorough perusal and correction of my drafts ensured long-term high quality. Due to his immense insights into the gaming scene and player-oriented feedback, I was inspired and advised more than just once. Even with tight deadlines, he always found the time to help me with my research, hence he deserves my deepest regards.

Finally, I would like to thank my mother for her unique and absolutely irreplaceable support during my education and my ongoing scientific work. Without her skill to coordinate and manage, her ability to encourage and motivate me, her strength and love, I would not be half the man that I am. Therefore, I would especially like to dedicate this work to her.

Abstract (English):

Within the last few years, the importance of (multiplayer) computer games has experienced immense growth. On par with that, the size of persistent virtual environments (VEs) has also increased. Today, more than 120 MMOGs (Massive Multiplayer Online Games) ranging from FPS (first person shooter) to classic RPG (role playing game) settings exist. This illustrates the significant influence of the growing number of simultaneously playing users.

The way that games are played has also evolved. Not only has the community and variety of gaming grown in numbers; but the way games are understood has changed fundamentally, too. In fact, terms like “hardcore” (a behaviour regarding the game; describes a far above average interest in the game content and a strong motivation to achieve) and “casual” (attempts to play the game with below average interest) have become well known in the player scene. Hence, player behaviour is one of the most valuable influencing factors.

Furthermore, the importance of mobile games has increased rapidly as well, leaving no doubt that the current game evolution aims for more flexibility towards locations. Thus, new problematic fields in the mobile sector are arising, especially with regard to real-time applications.

This dissertation will focus on the current game evolution and pinpoint the most important related approaches. It will subsequently introduce certain techniques to improve aspects of each of the most significant influencing factors: massive multiplayer, mobile gaming and player behaviour.

The first research approach aims to understand the underlying player preferences in order to create better software solutions. As a part of this, the effects of virtual fragmentation (difference between real world and virtual world behaviour) and game time distribution will then be evaluated and statistically analyzed.

The second approach aims to improve the mobile gaming performance by analyzing limitations in input and display of current mobile devices. By using a software solution, the matching time for players is improved drastically and by integrating mobile support for instant messengers, each user can also communicate with in-game friends who play common Internet games.

The third approach introduces a middleware solution to support MMOG games. By designing a message-based structure and creating a generic application, it can be expanded for the upcoming next generation MMOGs.

Abstract (German):

In den letzten Jahren gewann die Wichtigkeit von (Multiplayer) Computerspielen zunehmend an Bedeutung. Zusammen damit vergrößerten sich die virtuelle Umgebungen (VEs). Heute existieren mehr als 120 verschiedene MMOGs (Massive Multiplayer Online Games), die von FPS (first person shooter) bis hin zu klassischen RPG (role playing game) Settings reichen. Diese Entwicklung illustriert den signifikanten Einfluss der zunehmenden Anzahl an (gleichzeitig) spielenden Nutzern.

Des weiteren hat sich die Art Spiele zu spielen weiter entwickelt. Nicht nur die Anzahl der Spieler und die Vielfalt der Spiele hat zugenommen; vielmehr hat sich die Art Computerspiele zu begreifen fundamental verändert. Tatsächlich sind Termini, wie „hardcore“ und „casual“ in der Spielerszene sehr bekannt geworden. Deshalb ist das Spielerverhalten einer der wertvollsten Einflussfaktoren.

Außerdem hat sich die Wichtigkeit von mobilen Spielen rapide entwickelt und lässt keinen Zweifel offen, dass die aktuelle Entwicklung klar auf höhere Flexibilität seitens der physischen Orte setzt. Dem entsprechend entstehen neue Problemfelder im mobilen Kontext, besonders in Verbindung mit zeitkritischen Applikationen.

Diese Dissertation fokussiert sich auf die derzeitige Spieleentwicklung und zeigt die wichtigsten verwandten Themenansätze auf. Darauf folgend werden verschiedene Techniken vorgestellt, um Aspekte der wichtigen Einflussfaktoren zu verbessern: Massive Multiplayer, Mobile Gaming und Spielerverhalten.

Der erste Ansatz beinhaltet eine Evaluation von Spielerverhalten, um die Erstellung effektiverer Software zu ermöglichen. Besonderer Fokus liegt hierbei auf dem Effekt der vitual fragementation (Unterschied zwischen realem und virtuellem Verhalten) und der Gesamtspielzeit, die beide statistisch analysiert werden.

Der zweite Ansatz zielt auf mobile Spiele ab, deren Eingabe- und Darstellungslimitationen evaluiert werden. Mit einer Softwarelösung in Form einer mobilen Lobby entsteht die Möglichkeit die Spielevermittlungszeit drastisch zu senken. Des weiteren ermöglicht die Integration von Instant Messengern für mobile Endgeräte die Kommunikation mit Spielern aus klassischen Internetspielen.

Der dritte Ansatz beinhaltet eine Middleware Applikation, um speziell MMOGs zu unterstützen, durch das nachrichtenbasierte Design ermöglicht dies sowohl Skallierbarkeit als auch Integration für künftige MMOGs.

Table of Contents

Page

List of Figures

List of Tables

1. Introduction

"Free advice is worth the price", *Robert Half*.

The gaming market's enormous growth over the last five to ten years reflects the acceptance and importance of multimedia entertainment. One element of this market is that the gaming industry's total revenue already exceeds that of the film industry by far (box sales); and growth is still ongoing. Within this evolution, one can observe that especially interactive multiplayer and mobile applications are growing rapidly.

Without a doubt this evolution also significantly increases the number of related research approaches and hence their scientific importance. The research field of gaming contains a wide variety of problematic fields, ranging from graphical performance and efficient network structure design on to social influencing factors like player behaviour. Due to the number of different problems, a single best practice solution cannot be created for all of them.

The current gaming multiplayer scene is made up of four main game types: FPS (first person shooter), RTS (real-time strategy), RPG (role playing game) and SG (sport games). Each of them covers its own set of problems and its own set of players.

In general, gaming applications are very graphic intensive; a major part of the modern video card design and their visual computing performance relies on the needs of game applications. Due to growth of the Internet, the multiplayer aspect of computer games also contains an important problematic field. Most of the applications are strictly real-time-based and have less to no tolerance towards latency (delay between sender and receiver), jitters (variance in the delay between sender and receiver) and package loss (loss of pieces of data between sender and receiver). Therefore, gaming research correlates highly with real-time video streaming research. To understand the next generation of games, one must look at the current state-of-the-art as well as the user groups' needs.

Besides obvious aspects arising from computer gaming like efficient network models or cheating protection, there are also social aspects. Several research approaches take a certain player behavior as a given assumption; but ultimately, the users are the ones

who determine how a game is played. By taking the users' behavior into account, the design of current games can be further improved and already existing games can be customized to fit the individual needs. Therefore, user behavior is one of the most important influencing factors for both game design as well as for conceptual improvement.

Another factor is the speed of game applications. The rapid pace in the game environment leaves practically no room for typing conversations during the action. Thus, players tend to use third party applications like voice communication tools in order to coordinate in-game activities better. Hence, gaming research is not only limited to mere game design, but further features like audio and video chat will also be more common in next generation games (for instance several online games already implement voice chat as an optional feature, so users do not need to download external applications).

Furthermore, the general gaming trend focuses on MMOGs (Massive Multiplayer Online Games) as the most popular game type. The integration of players into persistent online environments (online worlds that exist 24 hours a day) creates a completely new social structure. Players therefore even spend money in order to play in these persistent online worlds as long as they are rich in content and highly interactive. As a result, MMOGs are becoming the dominating game type, because they offer players the necessary game depth and social interaction.

A look at the MMOG market underpins this trend further. In the year 2000, only 15 commercial MMOGs were released, mostly focusing on the RPG sector. In contrast to those numbers, six years later more than 150 MMOGs were launched and at least 50 more were scheduled [MMOR].

By significantly increasing the number of simultaneously playing users, the aspect of scalability and game speed becomes ever more important. Those game types open up a completely new set of problems due to their massive number of players (normal multiplayer sessions now contain two to 20 players, whereas MMOGs have several thousand players), such as hosting the game environment on multi-server architectures or coping with latency at several game events with 200 or more players. To solve the current problems and combine the ideas of massive and mobile gaming,

it is necessary to understand the current network structure and the effect of latency on the game.

Another aspect is the huge size of the game worlds, taking multiple servers to host the complete virtual environment (VE). Thus, techniques for separating the world into different parts are needed. Two main game structures now exist since the emergence of the first MMOGs. The simple implementation approach is the zone-based structure, which basically separates the whole game world into different zones (smaller parts). Those are small enough to host them on a single server. One must differentiate between the in-game view (players' perspective) and technical realization. The zone borders are interlinked and from a player's perspective each time he/she reaches a border, a loading screen appears and interrupts the game play. After the loading is completed, the player starts in the next area. From a technical perspective, zone borders are used for handovers. The player leaves server A and is forwarded to server B that hosts the next zone. Furthermore, the loading screens are used to clear the non-persistent information (like graphical animations and models) from the last zone in order to release RAM for the next zone.

The other implementation approach is the seamless environment, where zone borders do not exist. For the player, this is like a completely seamless, never-ending world, although the game world is separated. The important connecting areas are hosted on two different servers. From a player's perspective, the gaming environment is seamless, and no loading screens appear. However, from a technical perspective, there are different zones: as soon as the player reaches the border of the zone of server #1, he/she automatically enters a handover area. This section is synchronized between the two adjoined servers. The handover to server #2 does not take place immediately, it occurs after the player has completely crossed the handover area. The disadvantage of this approach is the high programming complexity and the growing traffic in case of events in the border zones.

Another one of the most important trends is the growth of handheld devices (mobile gaming devices). Not only are commercial, game-oriented handhelds such as the Nintendo DS or the Sony PSP used; but the significant market penetration of more than 90% in younger peer groups also promotes the mobile phone as a common gaming platform as indicated in [Frit 2006.4]. The growing number of releases and

the relatively strong market demand for mobile devices underscore the mobile gaming sector's potential.

Together with the increasing social acceptance of mobile games, one can observe rapid changes in the devices' technical equipment. For instance current mobile phones feature color displays, RAM and hard disks in order to run mobile computer applications. As a result of technological change, the gaming industry will further orientate itself towards the next generation of gaming, which is mobile. As an example: at [Mobi] overall game revenue of mobile phone games in the US market shows a 61% increase from 2005 to 2006, and current mobile game revenue amounts to USD 151 million for the fourth quarter of 2006. A further increase is expected in the future.

With the major trends in computer game design, a) growing player numbers and b) a strict orientation towards a mobile environment, one can observe the wide range of related problem sets. This is further underscored by the growing social acceptance, the major influence of the player behaviour and the rapidly increasing size of the gaming market.

1.1 Problem Statement

Due to the high complexity of the gaming sector and the individual requirements for each of the game types, the resulting problematic fields are customized as well. Although the four main game types also feature a good number of similar key characteristics, they only differ in terms of emphasizing the influencing factors.

For example, network effects like latency have an impact on each of the different game types, as results from [Frit 2006.4], [Clay 2005], [Beig 2004] and [Shel 2003] show. Hence, the user's performance decreases with an increase in latency. Nevertheless, the impact on RTSs and FPSs is higher than in RPGs or SGs. Therefore, the effect of latency is one of the main aspects for FPSs and RTSs. Table 1.1 gives an overview of the problematic fields of each of the four main game types.

Network effects. The category of network effects includes latency, jitters and packet loss. Generally, all of the effects have a strong negative influence on the user's performance. Due to an increase in latency or jitter (latency spikes) the reaction time

is reduced, making it harder for the player to react to game events. Due to the fast paced design of real-time games, a packet loss of an important in-game action (like shooting) can also have negative consequences (for example, an opponent is not dying). As related studies show, players with diverting network conditions and similar gaming abilities reveal considerable differences in performance.

Table 1.1. Illustration of the significant problematic fields of the four main game types

Game type	Main problematic fields (technical view)	Players' perspective
FPS	Network effects (latency, jitter, packet loss), in-game communication, cheat protection	Inadequate game events (opponent is not dying, bullets are not hitting)
RTS	Network effects (latency, jitter, packet loss), synchronous game engine, cheat protection	Units are jumping (incorrect positioning), actions are not performed (latency)
(MMO)RPG	Social interaction, in-game and out-of-game behaviour, scalability, network structure and network architecture	Other players behave poorly (social interaction problems), slow game play, frequent loading screens
SG	In-game communication, cheat protection, personalized game content (player behaviour analysis)	Inaccurate game play (positioning problems), repeating game content

Game behaviour (in-game and out-of-game). This section features a wide variety of player interactions, both in-game (inside the virtual world, mostly interaction with the players' game character) and out-of-game (real world behaviour outside of the virtual environment). Even though potential cheating behaviour is excluded, considerable player misbehaviour still remains. This can range from DOS attacks on to brute force methods to corrupt an in-game server. One example of this is the attack

on blizzard Diablo2 hardcore servers; normal players lose their character permanently after death, whereas the entire server could break down for some players, so a backup must be used to restore the players' character. Furthermore, game influence on the players' real world behaviour is an important aspect as well.

In-game communication. With the growing focus of multiplayer architecture, in-game communication between group members is also becoming more important. The additional bandwidth required of audio or video communication leads to less remaining bandwidth for the game engine and hence to a lower player performance. The result is a trade-off between enhanced communication quality and in-game performance.

Cheat protection. Cheating is one of the most serious problems for all of the games. With fiercer in-game competition, especially professional, competitive games from the FPS and RTS section suffer from substantial game cheating. Hence, reducing vulnerability and categorizing different cheating mechanisms is the main aspect of this section.

Social interaction. By creating persistent online environments, the social interaction structure between the different players has changed. Thus, aspects like tolerated in-game behaviour, the creation of player groups, large player communities (like guilds or clans) and the value of in-game goods are addressed in this section.

Scalability. Due to the growing number of simultaneous players in the MMOG game design, the current TCP and UDP network layers are reaching their limits. Especially due to high network condition requirements it is important to distribute in-game load in between the multiple servers. Also game events and raiding (a large group of cooperating players try to kill a single strong opponent) are addressed as typical scenarios that require a flexible and dynamic in-game scaling.

Network architecture. This section covers the usability of different network architectures with regard to individual requirements of each game type. The common architectures are S/C (server client), P2P (peer-to-peer), multi-server and hybrid architectures; each with their individual advantages and disadvantages. This section addresses possibilities to further improve the underlying network structure.

Technical aspects. Although the users determine how the software is used and whether or not a game is entertaining, several technically related aspects are still required to approach computer gaming from an academic point of view. Due to the continuously growing community, new technical solutions are needed, especially in the problematic fields of massive multi-player gaming and mobile gaming. These aspects cannot only be solved by creating newer technology (however, especially the effect of high performance video cards is a major factor for next generation game design); moreover, efficient solutions for individual problems of scalability (rising user numbers), quality of service (supporting the virtual environment with necessary bandwidth) and hardware expenditure are needed. The technical aspects mentioned will be an important focus of all approaches presented in this thesis.

1.2 Scientific Contribution

This thesis contains various attempts to improve the current computer gaming design. Main focus relies on the aspects of mobile computer gaming, massive multi-player online gaming and on the influence of player behaviour. The computer gaming research field has a wide variety of different aspects, each of the approaches is described individually.

MCChat. Especially in a mobile gaming environment, a quick game setup is important due to the low average game time. The current mobile game scene features a large set of problems; one of which is the individual connection mechanism of each mobile multi-player game. Thus physically, neighbours might not be able to find each other; the MCChat lobby offers a dynamic way for a fast game setup and a general place to meet potential opponents. It is based on the J2ME SDK (Java Platform 2, Micro Edition), which aims to create a common standard for the majority of mobile devices. The generic implementation enables other mobile devices like PDAs to use the common lobby in order to find potential opponents.

In-game IM integration. Another problem is the variety of different in-game chats for each of the games. With regard to the MMOG gaming scene, a single player is still unable to combine the communication with out-of-game friends and in-game buddies. In-game IM integration, which is introduced in this thesis, has a generic mechanism to embed current instant messengers like ICQ, AIM and MSN into

MMOGs, by offering the game all interfaces for basic chat functionalities. If a publisher integrates the in-game IM framework into a MMOG, then a player could combine in-game and real-world buddy lists and reduce the additional third party applications while playing. Furthermore in a mobile content, players who are not running the client could still stay in touch with in-game friends (without needing to run the game client).

This is an alternative approach, because it does not downgrade a single game engine to fit the mobile device requirements. It uses already existing technologies (instant messengers and existing in-game chats) to support the pure communication aspect of persistent online environments in a mobile context. The structure of the in-game IM framework is generic to limit it neither towards a single game client nor towards a specific instant messenger. Due to the adoptable interfaces for the game engine, IM support can be easily customized for the MMOG. Only the GUI needs to be integrated by the publisher (all interfaces for basic communication are provided and they only need to be included in the different MMOGs).

Furthermore, namespace regulations have also been taken into account to prevent name collusions during the IM in-game integration. This allows the user to keep his/her individual in-game nickname, even while using the instant messenger in an online game.

4MOG middleware. One possible solution for a general problem is to create a software application that acts as a middleware between two layers. In the case of the 4MOG middleware, the approach combines interfaces towards the game engine and the underlying network structure. Generally (for game applications), middleware software can either be completely integrated (internally) or separated from the game engine. The 4MOG middleware acts as an integrated (directly communicating with the game engine) application; it offers the game developer a set of core functions that are required for any MMOG.

The reduction of programming effort based on the middleware's functionality provided allows the game developer to focus more on in-game content and on an effective engine design. The middleware's communication architecture is based on a message-oriented system, which reduces communication overhead and also makes it suitable for devices with scarce resources.

Furthermore, the middleware offers mechanisms to store important player data. This data can be used by the game to keep track of the different players or to store them in a database.

Distribution of online player behaviour. Several attempts evaluate the influence of computer gaming on real world behaviour. In order to create reliable data for further player evaluation, this approach analyzes how much leisure time each player invests in gaming (and how seriously players take the computer games). With the gathered data, game-related effects in players' behaviour can be identified. Therefore, a statistical analysis with a large data sample helps to evaluate the time each player invests (with regard to the average player, special game types and hardcore users). The relevant hypotheses and explanations can be found in Section 4.3 and their validation in Section 7.1.

Technical realization relies on the implementation of a database system. The design should support multiple languages in order to automatically set up the page in the target language. Furthermore, due to intelligent data clean-up strategies, illegal and unrealistic data sets are eliminated, leaving only the valuable answers for further statistical evaluation. Moreover, the general database setup acts as a common platform for further questionnaires and reduces the overall time needed to set up an appropriate online survey for game-related questions.

The goal of the survey is to analyze the overall time invested in gaming (compared to available leisure time) as well as to evaluate gaming dedication (how far is a gamer willing to go in order to play more). Furthermore, this survey aims to understand influencing factors for high games times, such as demographic factors, motivation or game type.

Virtual fragmentation. This approach focuses on the difference between real-world and in-game behaviour, the difference is called virtual fragmentation. A large questionnaire with demographic and personalized questions based on the psychological five-factor model evaluates the most important personality factors and compares in-game with real-world behaviour. The results further underpin the strong influence of computer gaming on real-world interactions. Hypotheses of the approach can be found in Section 4.4, the brief statistical analysis of the survey is featured in Section 7.2.

The combination of each of the five sub-approaches described improve several parts of the current computer gaming research field and contribute towards more effective and intelligent software designs as well as a deeper understanding about the players' motivation.

The goal of the survey is to analyze whether or not a significant difference between online gaming behaviour and real-world behaviour exists. This difference will be measured with the help of the five-factor model. If the respective behaviours differ significantly, they will be analyzed in order to identify potential reasons to explain them. As one can see (in Section 4), from a technical point of view both surveys are based on the same database structure and the same algorithmic support for further statistical evaluation.

Overall Contribution. The different topics in this thesis cover vital aspects of the current game design. Since the research field of computer games is relatively new, underlying player data is mostly missing. The game producers often do not share confidential data about their customer base and the player behaviour. Therefore, the first step in understanding current trends in the computer gaming field is the analysis of its users. With focus on current online games, both of the surveys aim to evaluate rudimentary questions: the overall game time (distribution of online player behaviour) and the behavioural influence on the real world (virtual fragmentation).

After these aspects have been analyzed, this thesis focuses on the two major areas in current computer gaming evolution: (1) the aspect of mobile gaming and (2) the growing number of simultaneous players. Both approaches require an understanding of the computer player. The mobile gaming evaluates the sub-group of casual players in a mobile environment, documents their needs and recommends solutions for the current technology of mobile devices to improve the gaming situation. On the other side, the massive multi-player approach aims to improve the quality of current MMOGs by giving the developer the opportunity to focus on the content instead of the network coding.

Both areas share common attributes in their problem-solving approach, since it is the computer player and his/her needs that are the origin of the design. In both cases, the solutions do not mainly aim to improve the current technical implementation, rather

they focus on a very user desire-driven method to create the most adoptable improvement for the current computer gaming situation.

1.3 Thesis Overview

The remainder of this thesis is organized as follows.

Chapter 2 introduces the necessary background information about the evolution in computer games. Moreover, the focus relies on the three main aspects of mobile gaming, massive multi-player online gaming and player behaviour as important influencing factors for next generation software design. Within the sub-sections, the most important mechanisms are explained in detail to provide an overview of the current scientific research field of computer gaming.

Chapter 3 presents the related attempts and therefore focuses on four categories: mobile gaming, massive multi-player gaming, middleware approaches and player behaviour analysis. The respective advantages and disadvantages of the most important related work of each section are evaluated and summarized afterwards.

Chapter 4 introduces the first approach “understanding player behaviour”. Hence it gives an overview of the motivation and the details of the two sub-sections “distribution of online player behaviour” and “virtual fragmentation”. Both aim to understand the influence of computer games and their frequent usage.

Chapter 5 contains the second research approach: “Next generation mobile gaming”. This section focuses on the problematic field of mobile environments and possible game design under these circumstances. The two sub-sections “MCChat” and “IM-Integration” are further described in detail; both aim to improve the mobile gaming situation due to a fast matching time and increased communication.

Chapter 6 features the third approach: “4MOG middleware”. The section also therefore focuses on the respective advantages and disadvantages of middleware applications as well as the explicit design of the 4MOG architecture. Moreover, an underlying testbed is described in order to evaluate the middleware application’s scalability.

Chapter 7 summarizes the contributions of all three approaches, and then compares the results with related work. Furthermore, statistical results of the testbed and the

user questionnaires are analyzed in-depth statistically. Moreover, a final chart illustrates the new approaches in contrast to the related work to pinpoint the overall contributions.

Chapter 8 then summarizes and concludes the contribution of this thesis and provides a critical analysis of current scientific limitations in game research. Additionally, it includes an outlook on further work.

Chapter 9 contains an appendix with the list of the most important abbreviations and further minor statistical results. Chapter 10 then features a complete reference list.

2. Background

"The ancestor of every action is a thought", Ralph Waldo Emerson.

The research approaches and the most important mechanisms for computer gaming are described in this chapter. It contains an overview of the current game evolution, especially in the last five years and also features statistical numbers in order to underscore the growing scientific importance of this research field. Furthermore, three of the main aspects for the next generation game design are introduced: mobile gaming, massive multi-player online gaming and player behaviour. Each of them contains the major related design techniques that are used for the individual problematic fields.

2.1 Game Evolution

Computer games can be categorized into genres by different factors. Hence, a distinct allocation with explicit attributes is difficult. Because of the vast variety of new ideas and the publishers' motivation to contact a large number of potential customers, many games tend to fall into more than one of the genres at the same time.

The aspect of constantly evolving computer techniques is another important factor for the difficult classification of computer games. That is because due to new technologies, new genres can be created; the most well-known example of such an evolution is the MMOG sector, which relies on the technological evolution of the last ten years.

In the related literature several different classification strategies are introduced [Myer 1990], [Wiec 2002]. Table 2.1 gives an overview of the most common categorization of games with a few of their individual traits. Nevertheless, the list of traits is not complete and it will change over time due to new technological and software design modifications.

Table 2.1. Overview of the different game types

Game genre	Individual traits	Examples
FPS	Highest graphical performance	Halo, Counter-strike
RTS	AI design, high complexity	Warcraft, Command & Conquer
SG	Realistic game engine	FIFA, Pro Evolution Soccer
RPG	In-depth content, interaction	WoW, Everquest
Puzzle/Logic	Simple & easy game design	Tetris, Punisher
Edutainment	Strict content orientation	Learning by numbers
Miscellaneous	High individuality	Bad Milk, Black & White

From a business science perspective, the market for computer and video games has experienced tremendous growth over the last ten years, with constantly increasing revenues. Today, the area of video games is one of the leading market segments in the electronic media industry, already exceeding the revenues of box sales from the video sector.

According to [DFC], current growth of revenue in the computer gaming industry will even increase: "*By 2006 we forecast revenue growing to $5.2 billion with continued steady growth so that worldwide online game revenue reaches $9.8 billion by 2009. By 2009, we forecast that Asia will still be the largest market, but Europe with forecasted 2009 online game revenue of $2.2 billion will be the fastest growing*." (DFC Intelligent Articles, 2007). Figure 2.1 illustrates the expected income broken down by user type and platform. Both factors are important because they strongly influence the way the games are played. As one can see, the gaming market will be dominated by personal computers and consoles.

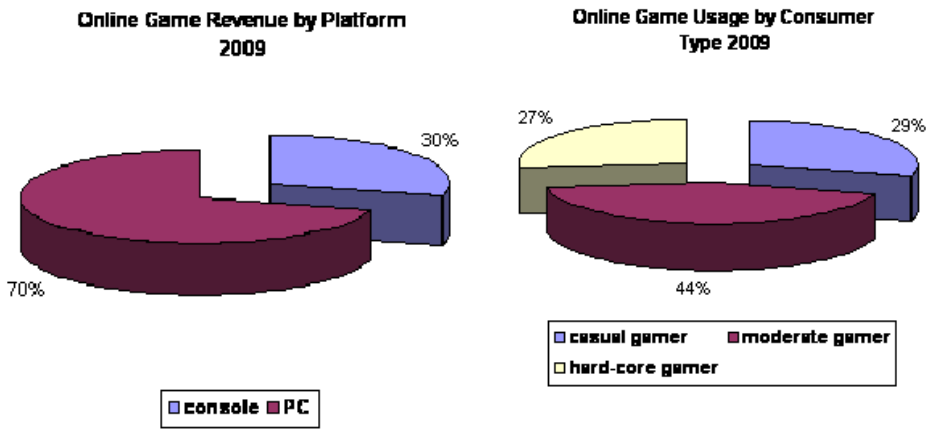


Figure 2.1. Distribution of the expected revenue in the computer gaming industry in 2009, broken down by platform and user type

Not only do these growing rates reflect the significant improvement in game design, but they also show a change in the social acceptance of computer games. Gaming is no longer only designed for a marginal group of hardcore players (they are expected to only make up 27% of the market share in 2009); it is becoming interesting for a much larger audience.

A very important aspect in the interaction between players is the possibility to play against human opponents. The very first computer games such as Pong already provided PvP aspects. Due to the increasing performance in computer networks, these multi-player possibilities have been improved further. Figure 2.2 illustrates multi-player settings for each of the four most important game types: RTS, FPS, RPG and SG.

The very first network multi-player sessions were connected by using direct wired connections like null modem cables. During the evolution in network design, LANs overtook multi-player connection aspect, and small groups of players, usually 4-12, played together in networks.

Figure 2.2. Screenshots of examples from every game type: the screenshot on the upper left shows World of Warcraft (RPG), and the upper right shows Counter-Strike (FPS), bottom left shows Command & Conquer 3 (RTS) and bottom right shows Fifa 2006 (SG) [Gams]

Due to the immense growth of the Internet combined with the increasing performance of broadband and cable connections, the player audience increased, too. This trend can be observed by the maximum number of players supported in parallel in the game design. During the years from 1995 to 2000, most of the games featured multi-player support of up to 12 or 16 players. After the successful release of the first MMOGs in 1998 (refer to [Ever] and [Ulti]), the maximum number supported for non-massive multi-player games grew continuously. Up-to-date servers with more than 40 or 50 simultaneously playing users are state-of-the-art.

On par with the enormous growth in numbers, techniques to reduce bandwidth requirements increased as well. One of the main aspects in real-time games is the frequent updating of the current in-game positioning. Hence, the technique of Path-Prediction has been implemented in most of the current games [Li 2006]. The

individual movement is therefore predicted and pre-calculated in order to reduce the data overhead.

Another differentiation for online games is the categorization by the number of simultaneously playing users in a single game session. The number of players mainly depends on the game design and on the game genre; not every setting allows a huge number of participants. Therefore, MMOGs today describe the group of games with a very large number of simultaneous users, usually at least 200+. Nevertheless, the resulting number of players can surpass more than 20,000 users during peak times.

The group of massive multi-player games is not completely selective; opinions differ regarding the first games in this section. By decreasing the term of "massive", in a broader sense Neverwinter Nights [Neve] has been one of the first MMOGs in the year 1991. The initial size of the game session was limited to 50 simultaneous players. The first persistent online worlds with massive multi-player support were released in 1998 with Ultima Online [Ulti] and Everquest [Ever], the player numbers for each shard were initially set to a maximum of 2,500 players. In the current generation of games, the new term of UMMORPGs was introduced by Eve Online with a simultaneous player record of 30,538 users on September 4, 2006 [EveO].

The four most important game types (from Figure 2.2) will be used to evaluate the approaches of this thesis, thus each of them is explained in detail (for a description of the game-related terms refer to the glossary):

First Person Shooter (FPS). The FPS game section features most of the current games. In most cases, the player can explore instanced areas by using a variety of weapons to compete with opponents. The perspective focuses on the first person, most body parts of the own character are therefore not visible. Due to the competitive nature and the fast-paced game play, this game type is very popular; typical game sessions are 30-60 minutes. Both individual competitions against a single opponent as well as teamwork in a small group are features of current FPS games.

Real-Time Strategy (RTS). This section contains games where the player takes control over a civilization or a large number of military forces. By commanding build structures as well as a large number of units at the same time, the player tries to eliminate his/her opponent. Tactical knowledge as well as fast computer interaction

are needed in order to be victorious. Due to the high competitive factor, these games are also very popular within the player community.

Role Playing Game (RPG). By focusing on a single character or a small group of personalized characters the player obtains the role of a (group of) hero(s). Usually each player only controls a single character in order to develop the characters' unique abilities and grow stronger over time. The game content has a strong focus on exploring a persistent online environment and a long-time relationship between the player and his/her character. In most RPGs, competition focuses on PvE (player versus environment), although several next generation MMOGs already include different PvP (player versus player) aspects.

Sport Games (SG). The section of sport games features various types of real-world sports. The most popular games feature soccer and basketball scenarios, where the player takes over the role of a complete team. By competing with other teams, each player individually tries to achieve as many cup wins as possible. One important aspect is the PvP interaction over the last few years. The game design of current SGs focuses on creating a realistic game engine in order to support a fair environment. The number of players, however, are strictly limited, usually only two to four players compete simultaneously.

2.1.1 In-Game Communication

In-game communication is now an important research field due to the growing importance of competitive games. Communication among the players is also an important aspect for the management of online game sessions as well as strategies. Hence, a wide variety of communication media is already provided by most online games today. These features are generally textual communication methods in different separated in-game channels (one for trading, clan [a union of players with the same goal], guild [a long-term, large scaling collaboration of players], group [a temporary in-game collaboration of fellow players], friends, etc.). In each of these channels, the players can communicate and, depending on the game type, even send XML links to show their equipment to others.

Most of the persistent online games also offer the opportunity to send personal in-game messages to other players. By doing so, the player can chose exactly one single

recipient for a private conversation. Friends can usually also be stored in buddy lists to check their online status and to communicate directly with each of them. The chat-logs are nevertheless not generally stored in-game, although several MMOGs offer the opportunity to create local logs of each conversation.

One of the most important disadvantages of text conversation is the reduced awareness during the process of typing. Quite often, players cannot type and play at the same time (this especially occurs during fast-paced games like FPS, RTS, etc.).

Possible solutions for this problem are communication commands to improve the speed of typing. A player can use shortcuts to trigger short, pre-selected messages for all teammates (like “enemy spotted”, etc.); FPS games often use this technique.

By making further use of the audio technology, current player teams are frequently also based on voice communication tools from third parties. These communication solutions even enhance flexibility due to a wider variance of voice commands, although a microphone and more bandwidth are required in order to run the game and the communication tool simultaneously. Therefore, current third party applications like TeamSpeak provide optimized communication methods with low processor load and different voice encoding strategies for constant data transfer rates. With regard to strict input limitations like game consoles or mobile devices, one should keep in mind that voice communication is even more important.

2.1.2 Technical Realization

Over the years, technical realization of multi-player computer gaming has changed dramatically. One of the main influencing factors is undoubtedly the evolution in network technologies. Hence, the most important techniques are described in detail:

Figure 2.3. Example of a split screen game (Off-Road Arena) from [Gams]

Split Screen. Due to the limited possibilities for a multi-player mode, the first games used a single screen for all players. Therefore, two to four players could interact with each other without using an underlying network. This technique is still frequently used in the console sector, where the TV is the only output device available. Whenever the game world cannot limit the movement of each player in order to ensure that no one leaves the screen, it is necessary to split the perspective up into two individual views. Figure 2.3 illustrates an example of a current split screen design.

Single Screen. Just like the split screen, a single screen only uses a single output device for multiple players, although the game engine needs to ensure that none of the players can leave the screen. Common examples of a single screen technique are round-based games, where both players compete asynchronously.

Network multiplayer solutions. However, both approaches rely on a software solution geared towards space limitation with a single output device. By connecting different output devices and synchronizing the game environments of each player, one can use additional hardware resources to massively increase the maximum number of potential players.

The general idea behind a multiplayer network includes the usage of each user's hardware in order to create a distinct perspective. By using different techniques to

exchange important in-game data, consistency of the game worlds is ensured. The result is a common game status that allows each player to interact and ensure a logical concept behind the game. However, information exchange between the users can be based on different network strategies.

Historically, the network technology is based on wired cable connections that used the COM port in order to exchange in-game data. Even in mobile environments, the devices are connected through a direct link if the users are playing a multiplayer game. As soon as more players need to be connected, the Local Area Network (LAN) with wired connections replaces the direct links.

Basically, the most distributed technology for the setup of LANs (Ethernet) is developed in different versions (10 Mbit/s, 100 Mbit/s and 1Gbit/s). Moreover, different users can also be connected that have distinct standards with a router that automatically calculates the maximum possible bandwidth between two clients. One of the major disadvantages, especially by using bus topologies, is the high probability of a network failure. As soon as the connection of a single client is interrupted, the whole network collapses. Thus with additional hardware (Hubs), most of the LAN networks are configured as a star topology to significantly increase the failure tolerance.

The next step in the network evolution is the usage of wireless connections, which enables players to interact in mobile environments without the usage of cables. The data rate varies between 11 Mbit/s and 54 Mbit/s; also two different network structures are supported: the fixed infrastructure and the ad-hoc modus. Although WLANs offer tremendous flexibility, the aspects of security and quality of service are the main aspects. Due to the obstacles, packages can be lost which is why especially connection sensitive game types like FPSs or RTSs are seldom based on a mobile network. Another technology for wireless connection is Bluetooth, which also offers the possibility for wireless data transfer. The maximum data rates of Bluetooth 2.0 (with enhanced data rate) is 2.1 Mbit/s; however, the lower bandwidth is not necessarily a problem for game applications. Bluetooth offers a lower average latency and if no obstacles or other devices interrupt the usage, then the range is also large enough to support most of the mobile games. The usage of a certain technology

depends strongly on the game requirements and the bandwidth as well as on the maximum tolerated latency.

Nevertheless, LANs also have a limited number of players; depending on the structure and the hardware, 16 to 200 players are supported simultaneously. One of the main problems is the sharing of common resources, because as soon as two clients attempt to use the same cable, all packages sent clash. Although technologies like CSMA/CD already reduce the overall number of collisions, the resulting package loss significantly increases with the number of participating clients and the frequency of data transfers; hence scalability is strictly limited.

In the last ten years, the trend towards the Internet has significantly changed maximum player capacities. Due to the high bandwidth and the increasing number of Internet households, persistent online environments can be created. Therefore, numbers of more than 10,000 players in the MMOG sector are becoming increasingly common.

2.2 Mobile Gaming

The term “mobile gaming” includes a wide variety of technologies which allow communication beyond the traditional limitation of fixed network structures. Although the term “mobile” does not only refer to the network structure, it can also refer to the mobility of the players themselves.

Hence, mobility of the communication is the opposite of the classical fixed networks, because it allows the user to take the input devices wherever he/she wants. Traditional communication techniques on the other hand require a more specific location. With the increasing importance of portable communication technology such as mobile phones or PDAs, these devices achieve a high market penetration. Nevertheless, the equipment is also capable of running entertainment software like mobile games. Both [Clay 2005] and [Taru 2006] describe the influence of mobile devices on gaming and compare the next generation professional handhelds like Nintendo DS and Sony PSP with the current mobile phones.

The other main aspect is the mobility of the communication participants; their equipment does not necessarily need to be mobile as well. Such mobility can be

achieved by a special service (personalized terminal design), which is accessible from various locations. Moreover, the service itself can have a fixed position in a network (for example on a fixed server with a HTML front-end), but only single users can access this service by using pre-defined access points (limitation by using a firewall with fixed IP addresses that are allowed access). Each of these points acts as a mobile terminal and offers the user his/her personalized features after logging in.

Furthermore, both aspects can be combined by establishing a service that can also be accessed by mobile equipment. One of the most precise implementations of mobility can be found in the mobile telephone network. By equipping the user with radio access technologies and supporting an individual tracking, each mobile phone user has considerable flexibility. Technologies like GPS allow accurate tracking of the user, which can also be used for entertainment applications, so-called location-aware games. In these games a user interacts in the real world and as soon as another player with a mobile phone is close to his/her current position, they can virtually fight against each other. One example of this is a mobile mercenary (soldiers that can fight against each other) scenario, which is shown in [Unde]. Newer mobile networks use IP-based structures to further improve the routing of communication data.

In a mobile environment, communication methods are limited as well. For a majority of the mobile games, communication is achieved by using real-time audio. Besides the obvious advantage of audio communication (for example with mobile phones), text-based messages can be delivered and instant chats can be created. The most significant problem is input limitation, even with fast typing it is not possible to react quickly enough. Especially with regard to the FPSs and RTSs, these circumstances lead to a significant in-game disadvantage.

However, additional game-related technical issues occur. With regard to scalability, one should keep in mind that classic communication integrates two participants. With an increasing number of potential players and high requirements of the game application concerning latency, jitter and packet loss, the standard GSM protocol has reached its performance limit.

Furthermore, one should also take into account that demand for a high QoS (quality of service) for mobile real-time game applications is also significantly higher compared to the standard mobile phone communication. A disconnected player

cannot simply rejoin a running game (this holds true as long as no main server for login and player management exists); hence a longer period with low to no signal can corrupt a whole multiplayer game session.

The transfer mode systems for mobile games mainly use either GSM or UMTS for mobile phones and WLAN for the next generation handhelds and PDAs. Both GSM and UMTS offer high accessibility in most European locations, although the main downside is the high latency and package loss [Heis]. Currently, latency in GSM networks is around 300-400ms, which is (depending on the game type) very high. One should keep in mind that an average latency of 100ms already has a strong influence on FPS and RTS games.

On the other hand, WLAN provides a more stable connection type (lower average latency, lower package loss), but the transfer range is strictly limited. As indicated above, Bluetooth has a lower bandwidth compared to LAN and the range is also limited. However the latency is better, especially for latency sensitive gaming applications that can be the most important factor for a technology decision. Currently, the trend reveals that the pure gaming handhelds (Nintendo DS [Nint] and Sony PSP [Sony]) both use WLAN as their communication method. The multiplayer game design of their applications seldom considers “user reaction” as an important aspect. As a result for the current handhelds, for their design decision, the relative higher latency of the WLAN technology is compensated by the higher bandwidth and range.

The mobile game market is growing rapidly, although it is still limited due to the mobile devices. Nevertheless, the game design for PDAs and mobile phone games has reached an entirely new level over the last five years. Current trends clearly indicate that the most well-known games feature clones of already successful PC (personal computer) and console versions as shown in [Harm 2004].

In order to summarize the most important aspects of the current mobile gaming evolution, Table 2.2 gives an overview of the major terms for mobile gaming. It includes a short definition of the technical protocols, the aspect of user mobility as well as the common mobile devices that are used for game design.

Table 2.2. Overview of the most important mobile terms

Mobile term	Description	Individual traits (game-related)
GMS/ UTMS	WAN (World Area Network) mobile protocols for long range data communication	High frequent data communication, cellular structure, game instability
WLAN/ Bluetooth	LAN (Local Area Network) mobile protocols for short range data communication	Limited range, higher bandwidth, access point structure, problem: Obstacles
User mobility	Location independent flexibility of the end-user; achieved due to terminals or handheld devices	Individual user behaviour, limited amount of time, quality of service important
Mobile phone	Common mobile phones can act as a game device	High market penetration, scarce resources, limited input possibilities, no common standard
PDA	Personal digital assistant, also capable of running current mobile games	Independent operation system, large displays, lack of video card support, touch-screen
Nintendo DS/ Sony PSP	Current purely game-oriented mobile handheld devices	High graphical performance, incompatible with other games, individual input devices

2.3 Massive Online Gaming

Online games that are played over the Internet also have their individual set of problems; mainly network factors like latency, jitter and package loss affect the in-game activities. Besides the influence of geographical conditions for the routing, a large overhead occurs by administrating multiple thousands of players in a MMOG. Therefore, each of the game types has its preferences for the network structure, the most common network designs are described in detail.

Client/Server structure (C/S). The most frequently used network structure for online games (especially for MMOGs) is the pure C/S system. A single (group) of high performance servers awaits contact with potential game actors (clients). Hence

the server offers a service that can be used by each client. In fact, each in-game action, like movement or interaction with the game world, is sent to the server, which acts like a centralized judge. After accepting an action, the server announces the change for all clients affecteds, so they can up date their local game world and ensure its consistency. Figure 2.4 illustrates a basic S/C model with a single server and four connected clients.

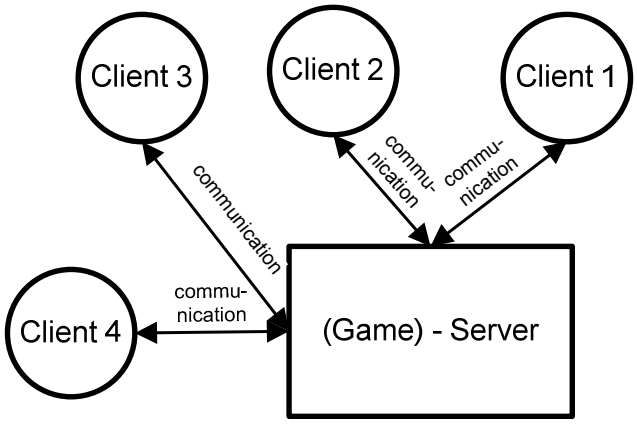


Figure 2.4. Illustration of a server-client structure

In most cases the publisher offers the player community at least one online server to login. The complete in-game environment is kept on the server; every interaction is verified before it affects the game world. Thus, every action by each of the players is checked at least once by the server. With any doubt, the main resulting disadvantage is the bottleneck position of the central server in this architecture. An increasing number of clients leads to an over-proportional increase in the resulting events, therefore the maximum number of players on a single server (shard) in MMOGs is capped to prevent the reaching of a critical number.

The scalability of the S/C structure is rather low due to the more than proportional increase in verifications required with regard to a growing number of simultaneous players. Distributed events as shown in [Yama 2005] or encoded message systems [Endo 2006] help to reduce the overall number of verifications, although the general bottleneck problem is still not resolved.

Another downside is the high dependence on the server. As soon as the server crashes, the game world with all non-persistent data is lost. Therefore, backup strategies are needed to ensure roll-backs in case of emergencies. On the other hand, these roll-backs indicate a major advantage of the S/C structure. Due to the centralized architecture, consistency of the in-game world can be ensured with comparably low effort, which in turn also reduces potential cheating opportunities.

Peer-to-Peer structure (P2P). Besides the centralized version, the network can also be organized as a distributed system. The P2P system only contains clients with equal rights. Every participant is allowed to offer and use services for and from other clients. With regard to P2P gaming structures, each of the participating clients must calculate a consistent local version of the game world. As a result, either the entire or at least a part of the common game world must be hosted by the client. Figure 2.5 illustrates a P2P network model with five participating clients.

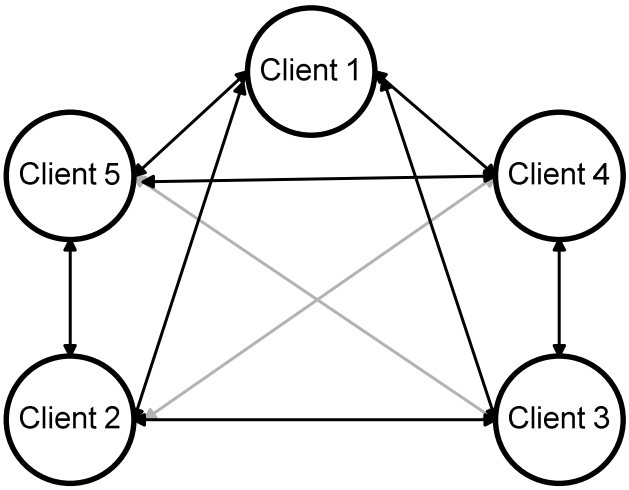


Figure 2.5. A peer-to-peer network with five participating clients, the arrows represent network communication between the clients

Updated messages for in-game events are sent to all participating clients; however, each of them is responsible for updating the game world correctly. Individual computation of each client in a P2P system with a comparison of results afterwards reduces inconsistencies compared to a typical P2P system. Two disadvantages still exist: first, the overhead increases with the number of comparisons (which is

especially negative for latency sensitive games) and second, even after results have been compared, loopholes still exist for cheating (for example a group of players that cheats together and compares their "modified" results with each other). This architecture was first introduced in [Gaut 1998] as a scalable P2P game engine that contains several cheat protection mechanisms.

The main advantage of a scalable P2P system is the non-existence of any bottlenecks. Communication between the clients therefore enables the reduction of an indirect server communication's average latency time. Another advantage is the minimized amount of bandwidth from the publisher, which reduces overall hosting costs. Furthermore, a possible breakdown of the server cannot occur, although depending on the implementation of the P2P network, as soon as multiple clients are disconnected or when they go offline, the resulting game world might collapse, too. In that case, the required backup would not be as accurate as in the C/S model, persistent data could even get lost, which demotes the P2P structure for persistent online environments of MMOGs.

Hybrid network structures. The advantages of the P2P and S/C structure are combined in the hybrid network system. It features the cost efficient P2P bandwidth distribution as well as the lower average latency, combined with the S/C advantages of a high publisher control and a centralized organization. The resulting differences in network performance of the three architectures are evaluated in [Pell 2003]. Figure 2.6 shows a hybrid network structure with a central arbiter node.

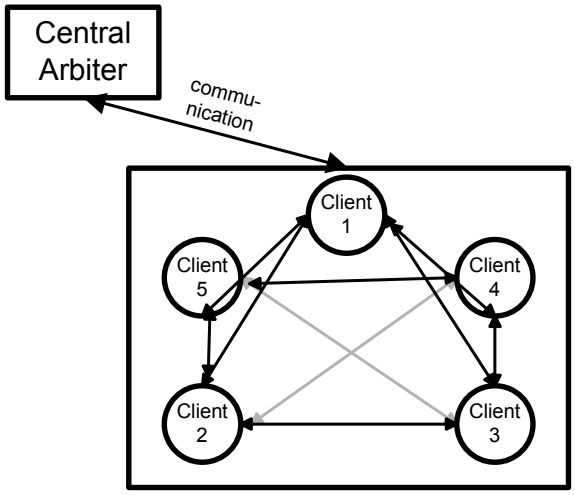


Figure 2.6. Example of a hybrid system with a central arbiter, the arrows represent network communication between the clients

A hybrid system with a central arbiter works similarly to a P2P structure, because messages are exchanged between the clients and are also sent to the central arbiter. This node acts as a listener of the network traffic and simulates the game world with the events received. The peers still calculate the game world locally, and as soon as inconsistencies with the central arbiter occur, all other clients are informed in order to synchronize the distinct game worlds. The resulting network structure is less vulnerable against cheating and shows a better scalability than the pure S/C implementation. Nevertheless, complex implementation and the slight loss of control from the publisher are the main downsides that prohibit a usage for commercial games today.

On the other hand, the combination of different servers is frequently used. Especially publishers with monthly fees require a separate login system as well as a secure verification database. Furthermore, another high performance database is required for all in-game objects and characters to store the important persistent player data. Moreover, individual zone and shard servers are needed to distribute the load in the game world. An illustration of a simple MMOG network structure is shown in Figure 2.7.

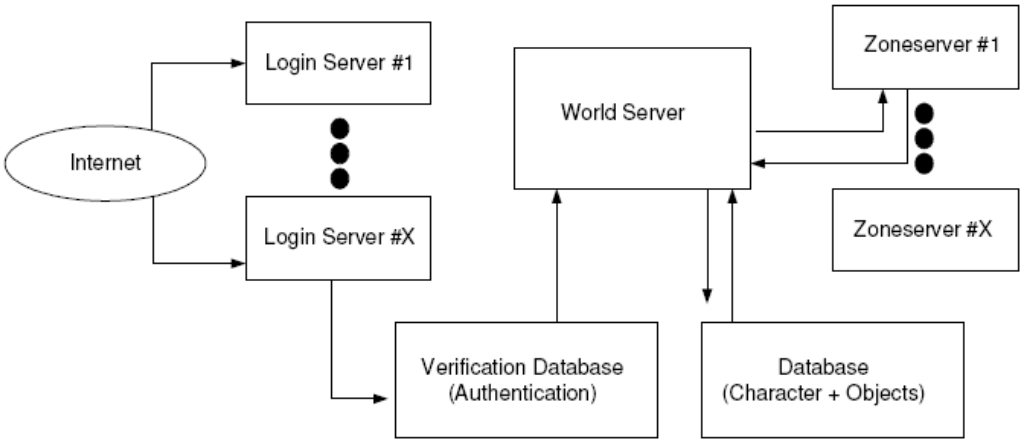


Figure 2.7. Illustration of a common MMOG multi-server architecture [Frit 2005.2] , the arrows represent network traffic between the different stations

The typical MMOG multi-server architecture, as shown in Figure 2.7, includes several login servers. These are directly linked to the authentication database in order to receive valid information as to whether or not a player possesses an active account or is already logged in. If the player is already logged in (character exists in the persistent online world), then either the current login session is terminated or the attempt to login will be aborted. This strict system is required to allow multiple users to play with a single account at the same time; especially due to Internet-sharing the IP addresses can be similar. Furthermore, the online world must remain consistent and it must be possible to ensure that the same character can exist online multiple times.

The world server hosts the consistent online environment and is responsible for every zone server (part of the online world) as well as the database for items (potential treasure that a player can gather), drop-tables (list of loot that every NPC possesses) and, of course, player characters (including all items and skills that a player possesses). As one can see, this position creates a typical bottleneck situation. As soon as the world server receives too many packets, it will have to either set priorities or try to solve the in-game actions with a best practice approach (tries to handle as much as possible).

This can lead to inconsistency or the lack of important persistent data change (like gaining a level). Hence, it is a technical trade-off between the options of restricting the maximum number of possible players per world server (sharding) or creating a multi-server system for the world servers (ultra massive multiplayer online role-playing game). Both approaches have their advantages and disadvantages: by reducing the maximum number of potential players, the problem of scalability is solved by not dealing with a number greater than “n” players. On the other hand, implementation complexity is drastically reduced because the integration of a multi-server system requires additional support for database access (interlocking and deadlock problematic) as well as a consistent game world management (handovers between each of the world servers).

Table 2.3. Overview of the different connection types

Network type	Description	Individual traits (game-related)
P2P	A pure client hosted network topology, information is stored locally with each user	Low cost, high scalability, problem consistent game world, persistent data needs to be distributed
S/C	Server-Client: The server acts as a controlling entity, no direct data communication between clients	Bottleneck position, low scalability, central controlling mechanism, single data channel, typical network type
Hybrid	Mixed structure between P2P and S/C, the arbiter acts as a controlling instance	Problem: Important data needs to be controlled by arbiter, difficult implementation, positive aspects of both structures
Multi Server	Typical MMOG server structure, additional controlling unit for authentification	Complex system, bottleneck position of the world server, medium scalability, database connection

In order to summarize the most important aspects of the current massive multiplayer gaming evolution, Table 2.3 gives an overview of the different network types. It includes a short description as well as an analysis of the game-related aspects.

2.4 Player Behaviour in Online Games

The aspect of player behaviour and its influence in computer gaming has been attracting ever more interest over the last five years. Nevertheless, the topic is not only limited to the adoption of in-game behaviour, one of the main aspects also includes the player's performance under problematic conditions such as high latency, jitter and package loss. Thus, each of the most important network effects is described in detail.

Latency. Especially in the MMOG sector, where S/C network structures are most common, the round-trip time of a message is an important factor for the clients. Generally, a high round-trip time (which equals a high latency) has a strong negative influence on the player's behaviour, therefore massively reducing the in-game performance compared to other players.

A bad connection to a server can have multiple reasons, ranging from a high physical distance, a high server load (bottleneck position) on to a slow Internet connection (the effect is called "last mile"). In order to prevent large physical distances between the clients and the server, next generation MMOGs often offer different access points, i.e. one for Europe, one for Asia and one for the United States.

The problem of slow Internet connections is also diminishing; more and more households in Germany have Internet access and especially households with a high number of people (3+) show a high availability of Internet access [Stat]. One possible result of the Internet sharing is also the increase in bandwidth, which further reduces the users' limitation from a gaming perspective (because of the higher available bandwidth). One of the remaining problems is the error correction protocol in most high bandwidth connections, which increases the overall latency by 60ms to 80ms, hence several gaming flat-rates with low bandwidth and no correction protocols are offered.

The tolerated latency differs for each game type, while FPSs and RTSs require a maximum latency of 150ms, SGs and RPGs can run fluently with a round-trip time of up to 500ms. As soon as the latency increases, one can observe significantly lower in-game performance. In order to illustrate the differences, Figure 2.8 shows the influence of latency on player performance in a FPS (Quake 3); with higher latency, the average kills/minute decrease. The slope of the curve is continuous, although one should keep in mind that a reduction of 40 kills/death to 30 kills/death will further increase with a higher latency (see Figure 2.8).

Figure 2.8 illustrates the example of induced latency in user performance for the FPS Unreal Tournament. As one can clearly see, the kills per match decrease significantly with a higher induced latency. The deaths per match were also increasing. Both effects together drastically lower the player's performance due to the faster reaction time in-game. The current game situation received is one of the most important data

sources for player interaction. Assuming a more or less normally distributed human reaction time of 100ms to 200ms, an additional latency of 200ms leads to an unequal game situation.

Not only is the pure latency responsible for the significantly lower performance, but the main factor in PvP situations is also the relative difference between latencies of the competing users. If every user receives additional latency, then the relative reaction is only affected partially; whereas a total difference of more than 200ms will drastically effect the in-game situation.

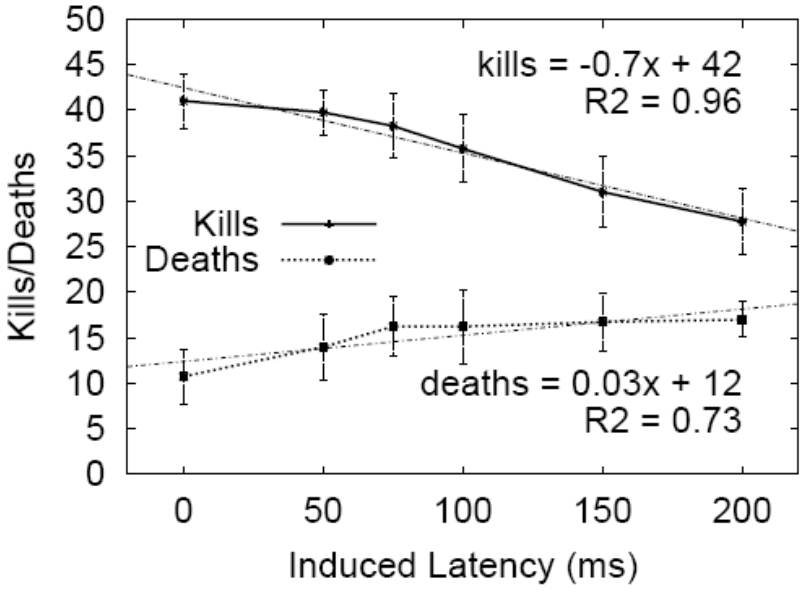


Figure 2.8. Influence of latency on the game performance exemplified by the Unreal Tournament (FPS) [Beig 2004]

A similar example can be seen in Figure 2.9, an observation of a RTS. In this case, the testbed setup included different tasks for the participants. One of the tasks was the exploration of an area with different induced latencies. As one can see in Figure 2.9, the overall time needed by the players to explore their game environment proportionally increased with the induced latency. Generally, the reaction of the game avatar was delayed due to the higher RTT (round-trip time) for the TCP packages from the client to the server. As the related paper [Shel 2004] describes in

detail, these effects were also noticeable during other parts of the game like building and especially during the human interaction (PvP).

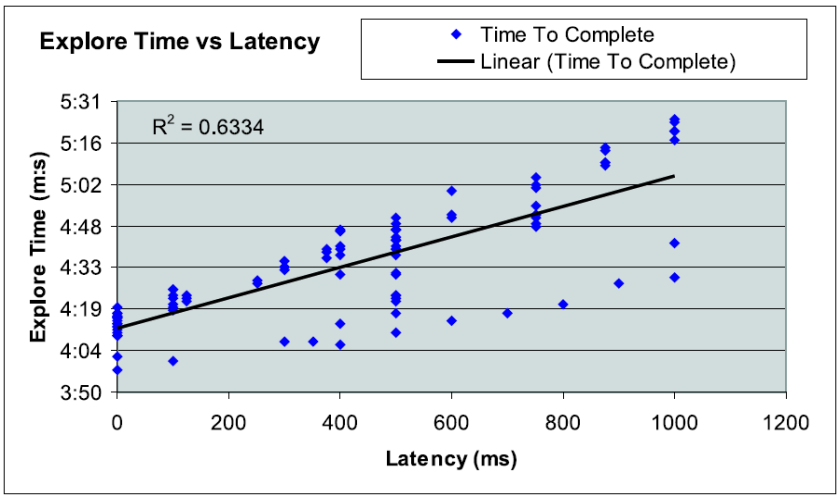


Figure 2.9. Influence of latency on the game performance exemplified by Warcraft III (RTS) [Shel 2004]

Packet loss. If a packet is lost while the game is running, the game world can become inconsistent. For example, a player who is shooting at an opponent might notice that his bullet never reaches the target because the information is lost in the network. The game server can reduce this effect by predicting possibilities for upcoming behaviour of the participating players. With regard to movement, the current movement is pre-calculated by using the current movement speed and direction and only in case of differences between the real movement and the predicted movement will the difference be corrected (this mechanism is called Dead Reckoning). However, important packages like shooting or interacting with an important in-game object cannot be predicted, thus making a loss of these important actions very harmful for a consistent game environment.

Jitter. The aspect of jitters in an online game is another negative network effect. It describes the variance of the latency, the higher a potential maximum of latency is, the more influence on game performance can be observed. The so-called “lag spikes” often occur during action intensive times, where several participants are interacting with each other simultaneously.

Typical game interactions like anticipating the current movement of an opponent and pre-aiming to a specific location are also corrupted by a high variance in latency. The game becomes more random, hence the treatment of each player relies more on the network effects. Jitters can occur for several reasons; the main cause is the usage of different routing paths or a distinct load of the Internet access (often caused by file sharing). A strong jitter effect has similar results as those of a package loss, important information is received too late and therefore the game world becomes inconsistent.

2.5 Summary

This sections contained a detailed description of game evolution in computer gaming. Therefore the main game types, their individual requirements and the relevance of computer entertainment have been outlined. The three main computer gaming aspects of this thesis: mobile gaming, massive online gaming and player behaviour have also been explained in detail in order to give an overview of the most frequently used techniques. This includes a summary of the most important network models, display techniques, mobile gaming effects as well as network effects.

3. Related Work

> "I often quote myself. It adds spice to my conversation", George Bernard Shaw.

The rapidly evolving field of computer gaming research has attracted a large number of different research approaches. This chapter will give an overview of the most important related work with a focus on three main aspects: mobile gaming, massive multiplayer gaming and player behaviour influence. The usability of middleware applications will then be presented and concrete examples of related work will subsequently be summarized.

Each of the approaches introduced will focus on one of the four central aspects in computer gaming: **mobile gaming**, **massive multiplayer gaming** and **middleware design** as well as **player behaviour analysis**. All of them are described in detail in their section and are analyzed afterwards with regard to the assessment parameters of this dissertation: scalability, mobility, reusability, quality of service, user behaviour and hardware expenditure.

3.1 Definition of the Assessment Parameters

The highly varying characteristics of the related work approaches make it difficult to compare them directly against each other or find a "one-best-way" solution. Hence Table 3.2 at the end of the chapter introduces an overview to assess the different approaches with regard to the parameters: scalability, mobility, reusability, quality of service, user behaviour and hardware expenditure. The general parameters are defined as follows:

Scalability. This factor measures how well a given approach performs with an increasing number of players and/or network size. The growing importance of MMOGs and the large number of simultaneously acting users promote scalability as an important factor. Of particular interest is how stable and effective the mechanisms identified perform with a critical mass and whether or not the overhead generated is

what makes the game unplayable. Each attempt can range from (++/ Excellent) to (--/ Very poor).

Mobility. This measurement includes the scientific contribution to mobile environments. Besides the growing numbers of simultaneous players, mobility is one of the main factors in next generation game design. Hence this assessment parameter describes the potential of each related work attempt to use a mobile environment for gaming. By integrating mobile protocols such as GPRS/UTMS or using wearable gaming mechanisms, each attempt can range from (++/ Excellent) to (--/ Very poor).

Reusability. The reusability of a given attempt describes how well it can be adapted to the next generation of games. Especially different game genres often require a special solution; flexible ideas can be transferred to a wider selection of target applications. Thus the reusability can vary from (++/ Very flexible) to (--/ Highly specialized).

Quality of Service. This parameter indicates how much an attempt takes potential data loss into account. Especially in persistent game environments, such as MMOG worlds, the stored data has a high value. The players' satisfaction is based on stable data management; less to zero error tolerance is given. The quality of service orientation of an aspect describes the probability to prevent data loss of persistent and relevant data, ranging from (++/ Excellent) to (--/ Very poor).

User Behaviour. The user behaviour is an important influencing factor for the evolution of any given game. This assessment parameter represents how much the related work was influenced by user behaviour research. The actual player behaviour decides how games are played online. If the player community does not adopt the game mechanisms as given, then this will lead to completely different behaviour. The level of user behaviour analysis varies between (++/ Excellent) and (--/ Very poor).

Hardware Expenditure. Hardware expenditure describes the effort to reduce physical resources needed to realize the given attempts. It contains the necessary expenses for the hardware as well as the opportunities to use other related devices as a flexible alternative. Therefore it can range between (++/ Very high) and (--/ Very low).

The assessment parameters used above to value the given related attempts are weighed and use the following symbols as given:

++	:	*Excellent, very high*
+	:	*Good, high*
0	:	*None, average, N/A*
-	:	*Poor, low*
- -	:	*Very poor, very low*

3.2 Mobile Aware Games

The aspect of mobility in computer gaming offers the user higher flexibility in the choice of gaming locations. Besides the conventional multiplayer aspects over the Internet, mobile (aware) games enable the player to use the application during travelling time. Even the gaming applications themselves can use the mobility aspect to merge the virtual world with the real environment and create new content, so-called wearable or location aware gaming [Bert 2006], [Liu 2006]. However, this gaming attempt also contains some major problematic fields.

It is well understood that distributed multiplayer games in a mobile content require a certain degree of support from the network in order to function correctly. Especially real-time aware game types such as FPS or RTS applications suffer from the game relevant effects in mobile environments.

Due to the high packet latency of GPRS and UTMS/3G, which can be up to one or two seconds (1000 to 2000ms), FPS and RTS players will notice a significant gap between in-game actions and the game engine. Therefore especially real-time applications are not capable of running under these conditions. As [Clay 2005], [Smed 2002], [Shel 2004] and [Bern 2001] underscore, the latency limit for RTS and FPS games lies between 100 to 150ms (10 to 20 times lower than the provided situation). Even latency robust applications like RPG games show an observable disturbance over 500ms [Frit 2005.2]. WLAN and Bluetooth offer a better latency performance of about 50 to 100ms; depending on the quality of the network.

Nevertheless, the range of both wireless techniques limits mobility. One would need a completely overlapping network of WLAN access points in order to resolve the latency issue.

Besides the influence of latency, games also suffer from jitters. Again especially FPS and RTS applications show significant differences with a higher number of jitters. So-called “latency spikes” often occur during action-intensive game situations (large fights with a large number of participants). Both GPRS and UMTS offer a wide range and stable network coverage, although it is well known that heavy latency spikes can occur. Local wireless techniques (WLAN and Bluetooth) on the other hand offer a smaller area of network coverage and they also have a major increase in jitters due to local obstacles (the effect of local obstacles is more significant compared to GPRS).

Another important aspect for any of the given game types is the network effect of packet loss. In the best case scenario, a lost package only contained movement information which can be restored by using rendering techniques like dead-reckoning. The worst case scenario includes game-relevant data like a shot or an important capture. Both mobile network techniques (WLAN and Bluetooth) offer a stable performance, and packet loss is relatively small. However, major data peaks on a single WLAN node can cause a bottleneck and thus create packet loss.

Besides the importance of network techniques, mobile devices have a considerable impact on gaming as well. Currently mobile phones, PDAs and commercial gaming handhelds (such as Nintendo DS and Sony PSP) are capable of running high performance games. As [Clay 2005] and [Jane 2005] clearly underpin, there is a significant difference in next generation mobile gaming devices between purely game oriented devices like the Nintendo DS and multimedia handhelds like mobile phones. Due to the vast variety of devices, users are specializing in just one of them, which reduces the number of potential game partners.

Table 3.1. Overview of network technologies for mobile gaming

Technique	Advantages	Disadvantages
GPRS	High network coverage, scalability, local detection	High latency, low data rate
UMTS/3G	High network coverage, high data rate	High latency, high mobile phone requirements
WLAN	High data rates, low latency, stable network	Local coverage, access points, bottleneck, low scalability.
Bluetooth	Very high data rate, low latency, stable network	Local coverage, obstacles, very low scalability

3.2.1 Asynchronous Mobile Gaming

Early work in the mobile gaming sector created the attempt to include asynchronous playing. Apart from the mainstream of real-time interaction, asynchronous playing gives the player the opportunity to make the next move time-independently. The best known example is distance chess, where both players take their turn and then wait for the response [Bogo 2004].

By using GPRS or UMTS this attempt can be implemented into a mobile environment as well [Bamf 2006]. This implementation eliminates the downsides of GPRS like high latency, because the applications do not need to run in real-time. With enough time, the packet loss problem is also eliminated thanks to reliable protocols like TCP.

In [Bamf 2006] the simple feature of text messaging is used to provide the “input” for the game. Users participate to create an online novel by further writing the current incomplete story. Those techniques used are easy to implement, although they greatly reduce the number of target applications. Due to the cellular structure of GPRS/UTMS and high availability, the user can generally play wherever and whenever desired. Thus the aspect of mobility as well as time independence is greatly supported by this attempt.

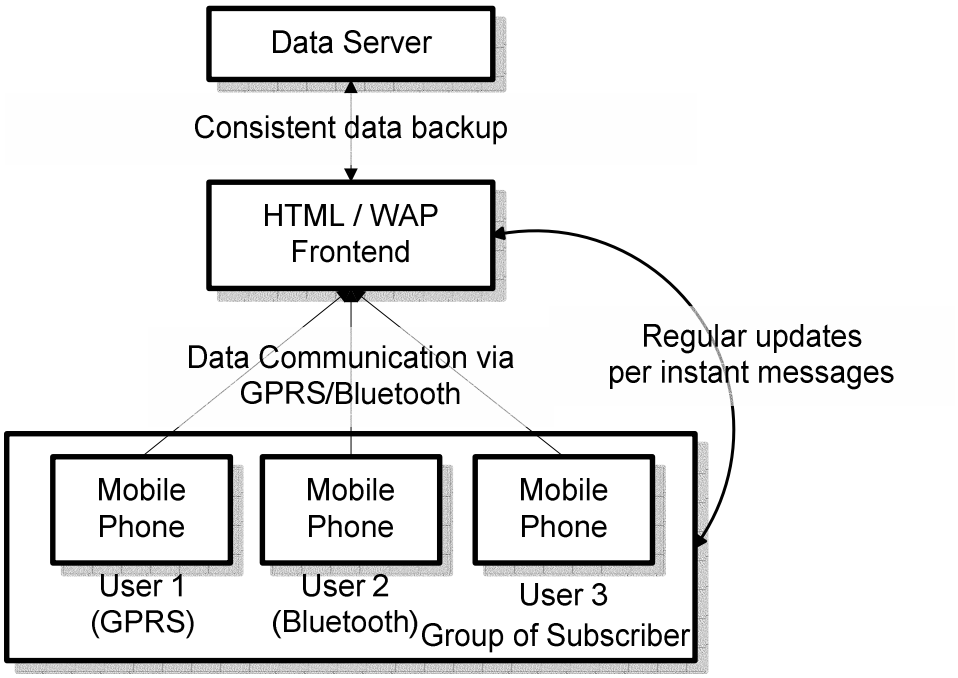


Figure 3.1. Illustration of the asynchronous game design. The network uses a data server for consistent data backup

With regard to the aspect of scalability, one can abstract from the simple 1 vs. 1 distance chess scenario. By increasing the number of potential players, the game mechanisms need to feature at least one interlocking mechanism. In order to keep the "game world" consistent, it is vital to ensure that each of the players has the latest update. As an example: [Bamf 2006] introduces an online novel, where multiple users can intend to write at the same time, the resulting story would include an inconsistency, because each of the users does not know about the actions of the others. The downside of interlocking is reduced interactivity, a user is only allowed to continue writing the story if no other user is doing the same thing at the same time. Hence, the application does not scale with a growing number of players.

The reusability strongly depends on the users' behaviour; because of the variety of misbehaviours a single player might acquire negative experiences. An AI (artificial intelligence) can be integrated in order to keep the game running even when players fail to answer for a longer period of time. Although besides the effect of interlocking (a certain player keeps the game stalled by not moving/reacting), there is no on-the-fly content evaluation. Generally, a user could send a large number of messages in a

short period of time (others would need to read through all of the text to be updated again).

Due to the missing factor of achievements [Bart] in the asynchronous games, players are not motivated to cheat the system. This aspect nevertheless bears some severe disadvantages as well. By limiting the variety of genres, one is excluding a major number of potential players; hence creating a special solution that is exclusively usable for asynchronous games. Furthermore, strong focus on pure GPRS/UMTS data transfer increases the gaming cost and excludes several devices (like the next generation handhelds, Nintendo DS and Sony PSP).

3.2.2 Near-Field-Areas and Mobile Games

One of the most important aspects in gaming is the capability of communicating with the other users. Due to the increase in different multiplayer game types, most of the players will agree that a multiplayer modus certainly enriches the game design. Therefore, the development in the mobile gaming sector is strictly multiplayer oriented (the usage of a single device with a split screen would significantly reduce the playability).

As a part of the mobile gaming trend, one aspect is the usage of near-field-area technology such as Bluetooth, WLAN or RFID technology combined with the real world (also called location aware gaming or wearable gaming). The main idea behind the attempt is to integrate one of the available wireless communication methods in order to create a virtual environment that uses the real world. Thus each object (which should be modelled in the virtual environment) needs additional hardware in order to communicate with the player or store necessary information. As one can imagine, there are several ways to implement near-field game zones; each of the technologies varies slightly in communication range, hardware effort and usability. While maintaining generality, this section will delve into the usage of RFID technology as an example of how to integrate gaming in a near-field area context.

As previously mentioned, a potential way to integrate mobile multiplayer gaming is the usage of RFID (radio frequency identification) tags [Garn 2006], [Petr 2005], [Coul 2006]. A RFID tag is an object that can basically be attached to a product or person for identification purposes; therefore it contains a silicon chip and an

antennae. For gaming purposes, the onboard chip can be used to store game-related data, however, it is necessary to understand that due to the size limitation it is not possible to store a large amount of game world data on the tags. Depending on their size and of course the price, next generation tags do not offer more than a few MB (Megabyte) space. Generally two different types of tags exist: the passive and active tags. Passive tags do not have their own internal power supply, so they need an antenna that is designed to collect power from the incoming radio signal and to also transmit the outbound signal back. The lack of onboard power supply further decreases the size of the tags. The active tags on the other hand feature their own internal power source, which is mainly used to power the outgoing signals. As a result the size increases, however, the reliability of active tags is typically higher compared to their passive counterparts. Current tags vary in size (depending on their antenna and whether or not they are active tags) and signalling range, which is usually 10cm to 5m for passive tags and 20m to 100m for active tags.

This attempt relies on the huge market penetration of mobile phones; with over 643 million mobile phone devices sold in 2004 and a forecast of over three billion subscribers by the year 2010, the potential user group is massive [Gonz 2005]. Mobile phones constantly offer more features, integrating a RFID reader to interact with RFID tags is a possible expansion for next generation mobile phones. It is important to understand that common mobile handhelds do not feature a RFID reader, yet integration into the devices is needed in order to benefit from the huge mobile phone market. Hence using RFID, which enables basically every object to be connected to the Internet, creates a completely new network structure.

An example scenario for a wearable gaming implementation can be found in [Garn 2006]. This scenario includes virtual spraying on walls, the RFID tags are attached to normal walls and a player who sees one of the tags can use a RFID reader to see which picture is virtually “painted” on the wall. Furthermore, this picture can be overwritten with an own tag. Sustained motivation for the players is to distribute the own picture to the greatest extent possible. By integrating the RFID tags into the real environment, one uses given structures to expand the game world. One can attach the RFID tags to real world objects; it is possible to create a location aware gaming environment where the minor information in the onboard memory of the RFID tag

acts as the game world information. Figure 3.2 illustrates the physical design of a RFID tag from [Garn 2006].

Generally, other wearable gaming examples can include different real world objects with RFID tags on them. An implementation of mine-sweeper would use the tags to store information about the number of adjacent mines or would be a mine itself. The players could use the RFID reader to look for mines in a “virtual mine field”. Possible other scenarios could be Pac-man or hide-and-seek, supported by the RFID technology.

In terms of scalability, RFID tags offer the option to generally integrate as many objects as desired into the game world. However, as soon as two or more games use the same technique, it is necessary to clarify which tags belong to which game. Objects can either be marked or additional information from the game provider is required. One can also imagine tags that support different games at the same time, as long as the game mechanisms and the radio signal differ from each other, multiple game information of distinct games could be integrated into the same tag.

Another important aspect is that the technical requirements do not scale with the number of players. Assuming that each of the players requires at least one RFID tag in order to be satisfied, the required number of tags will soon pose a problem because when the community is large, especially replacing the inactive or old RFID tags will increase significantly with the number of players. Physical limitations can also occur. If multiple tags are hosted in the same location, objects can already be “used” by other players.

Figure 3.2. Exemplary picture of a passive RFID tag from [Garn 2006], the tag itself is located within the paper bag on the bottom left-hand side

One of the major advantages is the high factor of mobility that is created by the RFID tags. By integrating RFID readers into mobile phones, they can act as the main gaming device in this approach and offer great flexibility. A good example of this trend is the next generation of mobile phones from Nokia, older versions are supported with separate RFID kits (such as Nokia 5140), meanwhile newer models already have an integrated RFID reader [Noki]. Other handhelds can also benefit from this technique; generally PDAs or next generation gaming handhelds (like Sony PSP or Nintendo DS) are capable of RFID tag reader integration; currently such integration is considered as an opportunity for future models. Hence with the required number of tags, a large number of potential players can be supported by merging technology already deployed.

The RFID tags also offer suitable reusability for other (gaming) applications. Especially data stored locally can contain information for several games at once, thus using the same device for multiple game worlds. This technique offers a stable network as long as an individual node does not become too popular. Well-known nodes will probably use more energy because they will be updated frequently, thus the rate of replacement will be significantly higher. Also simultaneous access by multiple players can cause deadlocks in a single node. Therefore it is important to

physically spread the network and scale the number of RFID tags with the number of potential players. Due to the asynchronous game nature each contestant can use the game network whenever or wherever he/she wants to.

The factor of security is one of the disadvantages of RFID tags. By allowing the users to upload data to the nodes, one must keep in mind that the next player will receive this data (this can be prevented by using encryption methods for each game). In terms of mobile phone security a “direct” data exchange can cause considerable security problems because harmful data can corrupt the mobile devices of other players. Hence a specific solution for a single game can prevent the abuse of harmful data transfer, although as soon as multiple RFID tags are attached to the same object or a single tag features data for multiple games, a more generalized solution is needed. One option would be to expand the J2ME SDK, which runs on most of the next generation mobile phones. If a RFID reader is also supported, the J2ME engine could provide basic security functionalities and therefore offer a general platform to increase the security of the RFID attempt.

In concluding the positive and negative aspects of the attempt, the integration of RFID tags into location aware gaming is based on the combination of already existing technologies. Due to the low prices and high flexibility of the RFID tags, the resulting game world can be designed individually. Although one should also keep the limitations like physical space, a missing integration of RFID readers in mobile devices and security in mind.

3.3 Massive Multiplayer Games

Another central aspect of computer gaming is the constantly growing number of simultaneously playing users. With the creation of the first persistent online worlds [Ever] and [Ulti], the gaming behaviour of the users changed completely. The current game evolution leaves no doubt about the importance of MMOGs, therefore creating all forms of massive online worlds like MMOFPS (massive multiplayer online first person shooter), MMOSpace (massive multiplayer online space game), MMORPG (massive multiplayer online role playing game) and even UMMORPG (ultra massive multiplayer online role playing game). One of the characteristics of MMOGs is that

they usually consist of a virtual world with a time and space concept similar to that of the real world.

The research field concerning the massive multiplayer game genre also includes a wide variety of different analyses, mainly focusing on the most important aspects caused by the large number of players. One of these aspects is the efficient distribution of players between the (multiple) servers available. Since MMOGs (massive multiplayer online games) are typically built as a strict client-server system, they suffer significantly from the inherent scalability problem of the selected architecture. Not only instancing (creating an instance [personalized copy] of a zone, only a specific group is allowed to enter the instance) and sharding (creating a parallel game world, which runs on a different server) help to split the player community into manageable masses. As [Lu 2006] shows, an effective strategy for load balancing in between client-(multi) server architecture is required to generate a stable environment even during peak times. Hence distributing the points of interest in-game, such as running several events in parallel or splitting a popular zone up into sub-content further helps to even the loads in between the servers. The so-called in-game interest management [Boul 2006] is one of the most common ways to realize such a distribution. Originally developed in the FPS game research the interest management can reduce the broadcasted data up to a factor of six by only showing the player the relevant objects. A zone of interest is created and only in-game events that affect this zone will be transferred to the player, thus significantly reducing the overall data transfer.

Together with pure balancing between the servers, the in-game zones (even in a seamless MMOG) also pose a major coordination problem. The latency for any player interaction increases due to the server's bottleneck character. With regard to player grouping, the analysis of group communication [Vik 2006] offers further support for a more flexible in-game scaling. Therefore, they analyze and compare different algorithms to cope with the increased in-game communication latency and the massive server load.

With the growing complexity of online worlds, pure zone balancing (distribution of the persistent online world into different sub-zones) often fails to distribute the game world efficiently. Hence a more detailed distribution is described in [Vlee 2005],

using a micro cell balancing algorithm. By breaking the game world down into even smaller pieces and creating effective hand-over strategies between the multiple servers, it is possible to further even the loads. Nevertheless, as soon as an in-game attraction motivates a huge number of players to visit the same local area it is still difficult to separate the resulting overload.

But fairly distributing the load in between the in-game servers does not suffice for massive multiplayer games. Especially network protocols reach their scalability limits with numbers higher than 10,000 simultaneous players. Server traffic analysis indicates a similar player pattern concerning the daily routine, creating a significantly higher number of packages during the peak times in the evening and on the weekend. A controversy as to whether TCP or UDP meet the requirements of next generation MMOGs is now under way. However, both protocols clearly have advantages and disadvantages [Chen 2006.3].

3.3.1 Using and Detecting AIs in MMOGs

In light of developing MMOGs and their persistent worlds, there was also a need to implement a macro-economy with in-game money and funds. As soon as in-game values became re-sellable by auctioning them online for real money, there was also a huge endeavour to design Bots [Lehd 2005], [Bart 2004] and [Cast 2001]. Bartle [Bart 2004] determined that the persistent world of Everquest was the 78th richest country with regard to its GDP by calculating it with all potential citizens (players) and their estimated income if they sold their in-game money. Today, the reselling of in-game goods has become a sizable industry on its own with major downsides. A negative example of the reselling mechanism is the so-called sweat-shops. Low-cost workers from third world countries are forced to play online in eight to ten hour shifts every day in order to harness online treasures. These treasures are sold for real money afterwards.

In this context, one needs to distinguish between two different aspects of AI usage in MMOGs, the Bots (automated gathering programs) and the NPCs (non-player characters). The automated scripts created by the player are called “Bots”; these programs are mainly used to gather in-game resources without playing the game [Chen 2006.4]. In most cases, this takes place in a program that takes control over a

human character and an automated script that is run locally on the client's computer. It can be quite difficult to predict AI scripts; especially due to low error tolerance (banning a real player would have fatal consequences). Hence many MMOGs cannot cope with the constantly evolving community of automated programs because most of the players are not willing to invest the necessary amount of time to replay the same scenario multiple times.

The creation of NPCs enriches the virtual environments; these programs are often part of the online world itself. [Merr 2006] classifies NPCs into three different categories: enemies, partners and support characters. With no loss in generality, this thesis will focus on the detailed description of the support character role. The other two aspects of NPC design differ in their in-game usage, but the underlying implementation methods are similar. NPCs offer a way for players to interact with computer controlled virtual characters, which can, for example, act as a merchant. One possible way to implement NPCs is introduced in [Merr 2006], which shows an interest-based learning design model for realistic NPCs.

From a technical point of view the problem in NPC design is their behaviour, they are supposed to behave like humans. In particular, this signifies that players appreciate it if a NPC has its own daily routine like going to the pub in the morning and coming back late at night. Simple actions can be implemented by using pre-defined scripts (also called reflexive agents), but many online MMOGs also feature complete towns. NPCs have their individual roles in the town, like a blacksmith, merchant, fisherman, etc. A vision for the next generation implementation of NPCs is a detailed interaction between NPCs (also called learning agents). The main difference is that reflexive agents show a static behaviour, whereas learning agents can adopt in-game events and create a new behaviour based on their experience. As a possible example, the fisherman could bring his catch to the cook in order to produce new food for the community (which players could buy afterwards). The players also appreciate social interactions like fights or talks. Especially players with a high social focus, so called socialisers [Bart] will greatly benefit from detailed NPC interaction.

With regard to the current MMOGs, most of the persistent online worlds feature more than 1,000 NPCs, mainly to populate the important in-game locations such as towns or dungeons. The scripting of daily autonomous NPC routines is therefore one

of the major aspects for high quality game content. An improved or automatic algorithm therefore provides the AI with opportunities to greatly increase usability rather than just remaining inactive. These algorithms are more complex versions of simple scripts; their purpose is to allow the NPC to have an individual behaviour. A downside on the other hand is the prohibition of illegal or meaningless NPC behaviour (like committing suicide).

One possible implementation method for more complex NPC behaviour is the usage of learning AI scripts, so-called MRL agents (motivated reinforcement learning). The basic idea behind them is that every possible action has its individual rate of interest. Hence the MRL agents use an intrinsic motivation process to identify interesting events. The higher the rate is, the more likely it will be performed next. By performing a task, its interest rates drops, therefore making it more likely that other actions can be promoted to be performed next.

As soon as new objects or possible activities enter the interest radius of the NPC, the initial rate of interest will be high, reflecting a curious nature. For example, a player who enters the working area of an NPC would receive its full attention, maybe a greeting or a welcome ritual. The longer the player stays within the area of interest, the more the NPC gets used to him/her. Generally, the agents can continually identify new events about which they would like to learn. The adopted behaviour of a MRL agent is the result of its experience. With new events, the behaviour of the NPC can change over time.

With regard to scalability, the usage of Bots and AI scripts clearly outperforms most of the other content approaches. Most of the graphical emotes (social actions that the character can perform like waving and cheering) and movement will be rendered locally, leaving only the location and decision task on the server. Even with high numbers of players, NPCs can still interact seamlessly. Scarce resources such as a merchant's rare materials can either be customized or locked as soon as they are purchased. The attempt is generally also valuable for a mobile environment, especially when the number of potential human players in certain areas is low. However the world consistency, especially due to the influence of the NPCs, can pose a problem in massively distributed mobile networks. This can be exemplified by the merchant of a rare resource who offers it to a player in area A and then another

player in area B also requests the same resource. If the game world is not interlinked, this can lead to an asynchronous status.

The reusability of the AI attempt in the next generation of games is appropriate because it offers a generic framework, leaving decisions (like the rate of interest erosion or the selection of potential behaviour) open for the concrete game design. Nevertheless, one should keep in mind that security and player behaviour are the major disadvantages of the AI approach. As soon as a player has the opportunity to influence a NPC, this can certainly lead to destructive behaviour like luring or blocking an important NPC to annoy fellow players. Also the regulations of tolerated behaviour should not extend reasonable boundaries. No player will be satisfied by following an important quest NPC for an hour and playing hide and seek, while back-tracking its trail over the whole city.

The usage of commonly shared NPCs requires a server-client network, leaving fewer opportunities for an alternative solution. As the example of the mobile gaming sector indicates, the concept can be transferred in general, although one should keep in mind that inconsistency or cheating opportunities will increase significantly when doing so.

3.3.2 P2P Architectures in MMOGs

The common MMOG architecture uses a classic client-(multi) server model. Nevertheless, there are also other opportunities to create scaleable and persistent online environments. As [Hamp 2006] and [Assi 2006] introduce, the usage of advanced peer-to-peer mechanisms can also create a distributed online world. This concept of a P2P MMOG structure offers architectural support for a large number of players.

In order to support a stable game world with persistent data, the authors combine existing attempts. The DHT (distributed hash table) Overlay-Network uses Pastry [Rows 2001] to achieve robustness against potential network failures. Therefore both the communication and persistency rely on the Pastry infrastructure. Additionally, the object management includes a separate technique called PAST [Drus 2001]. This extension of the basic functionalities of Pastry is required to build an in-game consistency. Especially with regard to the importance of persistent character data it is

absolutely essential to have high reliability by replicating the objects. The worst case scenario for a MMOG is the constant loss of persistent data, because this will lead to a significant loss of players. The only known example is a complete server crash at the SOE (Sony Online Entertainment) labs in California, which was caused by a hurricane. No official data about the long-term customer loss have ever been published, but afterwards SOE decided to maintain an additional server data backup in a different location in order to reduce the chance of losing persistent data again.

The third part of the architecture includes the event-based messaging system Scribe, which enables multicasting in the pastry architecture. Important game events (depending on the area of interest) will therefore be routed to all potential peers. Figure 3.4 illustrates the combination of all three techniques for the architecture.

With regard to scalability, the approach offers a significant reduction of the overhead caused by simple broadcasting, which would basically announce every game event to all other clients. The Pastry architecture allows the system to find related peers more effectively and thus reduces the path length (which describes the routing length that a single packet needs to travel to reach its destination). The results included for in-game load balancing indicate the support of even a large number of clients.

Often in-game neighbours are highly distributed in the real world and thus it is important to reduce the average RTT (round-trip time) for a packet and the included peers as much as possible. The best case would be a network that knows about its physical structure. With this additional knowledge it would be possible to select the shortest path for each communication.

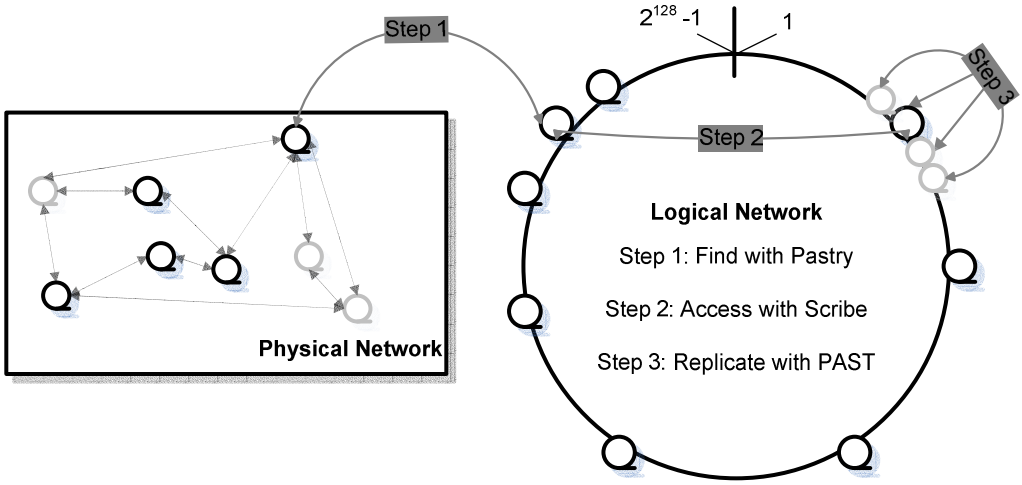


Figure 3.4. Illustration of the combination of different technologies to accomplish object access and synchronization in the P2P MMOG architecture: The search algorithm includes Pastry [Rows 2001], the file access is done by SCRIBE [Cast 2002] and PAST [Drus 2001]

Mobility still appears to be a problem with this approach, even by using the different techniques to balance the load and minimize broadcasting overhead. A P2P structure still creates significantly more traffic for a single client compared to a server-client structure. The decision about the network type highly depends on the game itself: a game with a lot of persistent data will prefer the classical server-client architecture. The number of players also influences the network decision because even with the progress in distribution techniques, P2P networks do not scale with an improving number of players. Thus the P2P approach offers a flexible solution for a lower number of players with mainly non-persistent game content.

The reusability is excellent and the hardware expenditure is also quite good because the architecture can also be applied to different devices. Even the next generation in games could use the existing network layer in order to create a new persistent game world. One of the downsides for the constantly evolving MMOG sector is that older games will probably have insufficient peers to distribute the data. With a decreasing number of potential players or a growing number of inactive peers, the game architecture fails to ensure data storage of persistent game information.

The drawback of a P2P-based MMOG architecture is the quality of service. Especially when using commercial pay-to-play mechanisms, an authentification server is absolutely essential. Furthermore, by allowing the clients to spread data one cannot ensure that groups will abuse the system in order to either harm other players or to obtain an in-game advantage. The problem of cheating is described in [Izai 2006], especially with regard to group cheating it is nearly impossible to prevent user abuse. Several time-related cheats can be minimized, but especially instanced areas or loot distribution (duping [illegal copying of an in-game item] or the creation of personalized items) can be massively influenced by the players, assuming they cheat as a team.

3.3.3 Public Server and FreeMMG

A similar attempt is the creation of hybrid structures in order to reduce the overall bandwidth requirement for the provider and to use as many external resources as possible by keeping the internal security level as high as possible. Examples of these trade-off design attempts are described in [Cham 2006] and [Ceci 2004]. The public server attempt assigns the most important functionality (billing, authentification) to the publisher; meanwhile the in-game activity (movement, fighting, and environment interaction) is spread among public servers.

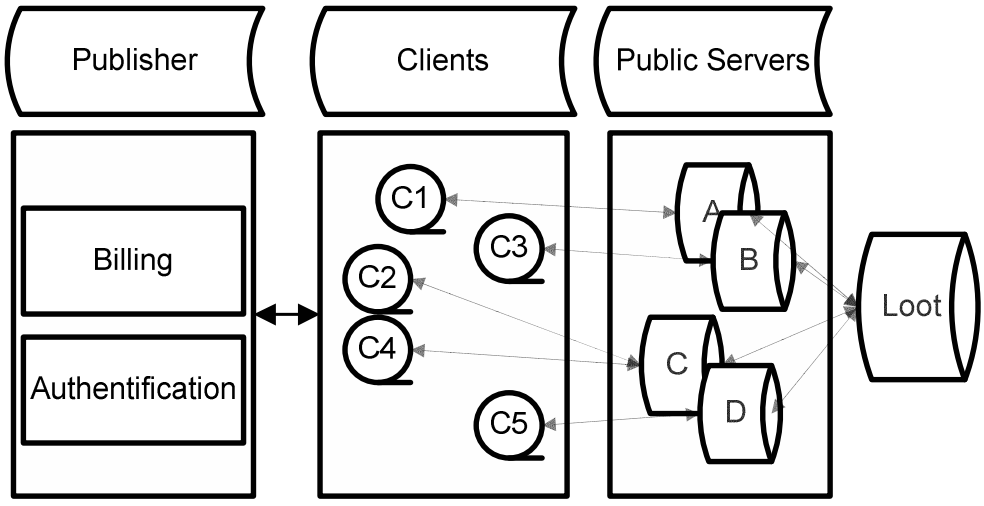


Figure 3.5. A FreeMMG architecture, the publisher still hosts the billing and authentification server [Cham 2006]

Figure 3.5 illustrates the general game design. The underlying architecture is similar to the FPS public server structure. Important in-game decisions are still strongly influenced by the central loot server (also provided by the publisher). However, a significant amount of in-game broadcasting will be transferred to the local public servers. So-called in-game tokens enable a player to “buy” permanent items from the loot server, each player earns them by being active on a public server (overall time counts).

With regard to scalability, this system strictly reduces the maximum number of simultaneous players. Even strong public servers can only handle a very limited number of players at the same time (example: “Tribes 2” with up to 200 players [Trib]). Thus grouping and social interaction will also be reduced because players might not be able to join in the same target server together. Several MMOGs offer a large endgame (the game content for experienced players with in-game characters on the highest level) raiding content; hence the design would not fit the necessary requirements (40+ players attempting to simultaneously reach an in-game goal). Furthermore by highly distributing the game world, the resulting game environment loses its persistence.

The aspect of mobility also appears to be supported insufficiently. Especially in a mobile environment, the effects discussed will increase. It is unlikely that mobile servers will exist, because keeping the mobile players synchronized with Internet-broadband users will lead to an unequal distribution of latency. Thus especially in a competitive PvP environment, mobile users will have major disadvantages.

Reusability is high for the next generation of games. As long as the in-game content fits the required specifications (which are: no PvP, a highly distributable game world, a server client version that can run on public servers, etc.) the public server bandwidth can be reused for the successor application. Furthermore, by maintaining the most valuable functionality on the publisher’s side, the attempt generates a high level of service quality. Persistent data can be stored on the main server, which reduces the probability of data loss. Another important aspect is that the public servers can offer a better connection due to closer physical routing. Hence each player can choose his/her server to optimize network performance. The hardware expenditure is comparably low due to the classic server/client structure. It shows that

no effort was made to reduce the necessary physical resources. Especially with lower available bandwidth (current MMOGs need at least 8 kB/s) and the reduced performance of mobile gaming devices, the architecture introduced cannot just simply be transferred to the mobile gaming sector.

The major disadvantage of the public server attempt is the lack of an evaluation of player behaviour. Distributing the game data to public servers triggers a significant risk of promoting player cheating due to data modification. Furthermore, rewarding the overall online presence with better in-game items also leads to a strong motivation to use a BOT program (third party application, which acts as an automated program in-game) or being AFK (away from keyboard), because rewards are no longer directly achieved by playing in the online environment, and instead they are given for online presence. Especially the aspect of group-related cheating is further promoted by giving the players a common area to exclusively meet as a group (their local server).

3.4 Gaming Middleware

Another method to handle recurring problems (see Section 2.4) is the design of a middleware application for games. Generally, a middleware application is computer software that connects software components or applications. The usage of such an application should reduce the complexity of the remaining problem (in this case game design) in order to provide the basic functionalities. As a result, the programmer of the game can focus on the remaining game-specific implementation (like game design or content). The general idea behind a gaming middleware is illustrated in [Kane 2005], showing an example of how to reuse a significant number of game binaries.

The design of a middleware application is not strictly limited to supporting already existing functionalities. For example [Tang 2005] features a free ranking system that offers games a method to integrate new aspects into a player community. As both examples show, the field of potential middleware sites varies substantially. Thus the applications are not only limited with regard to improving the network layer with game-specific functionalities.

With regard to the high individuality of gaming applications one should also take into account that multiple middleware applications could support the game at the same time. They do not need to be mutually exclusive as long as each of them solves the problem without interfering with any other middleware applications. Hence the architectural concept should be kept as generic as possible (therefore supporting a high reusability). Especially in the mobile game environment with a high number of game clones, the existing middleware applications could often be reused.

The major downside of the very specialized game requirements is the fragmentation into sub-problematic fields for each of them. Thus a single middleware application that supports all game types will produce a large overhead in order to solve every possible game-related problem. In most cases, especially with resource scarce devices, such a general solution will therefore be ineffective due to the massive data overhead.

Furthermore, the next generation game design is not completely based on the usage of middleware applications. A successful game can still be create by using no middleware at all. With regard to the substantial upturn in costs for game design (especially in the MMOG sector), reusing effective solutions provides a method to minimize the costs. Another valid argument is the higher development time due to increased beta-testing (beta stage of the software lifecycle in the game production) and large in-game content requirements. By relying on existing solutions like middleware applications, one can focus on the game content and balancing (equalizing the in-game mechanisms, for example by adjusting each of the available classes so that none of them is “stronger” than the others).

3.4.1 Middleware as a Service Platform

The considerable investment for the hosting infrastructure of MMOGs is addressed in [Shai 2004] and [Sing 2004]. The origin of the attempt in [Shai 2004] is the risk factor of creating a complex server network for the persistent online world, when the game success is hard to predict. Especially in 2005 and 2006 the MMOG market witnessed tremendous change by creating the second generation of MMORPGs (Everquest2 and World of Warcraft), therefore increasing the development costs to multiple millions of dollars for each game.

The service platform attempt aims to create a shared, on-demand platform for online games. By supporting the publisher with open standards and further utilities for game development, the overall costs should be reduced.

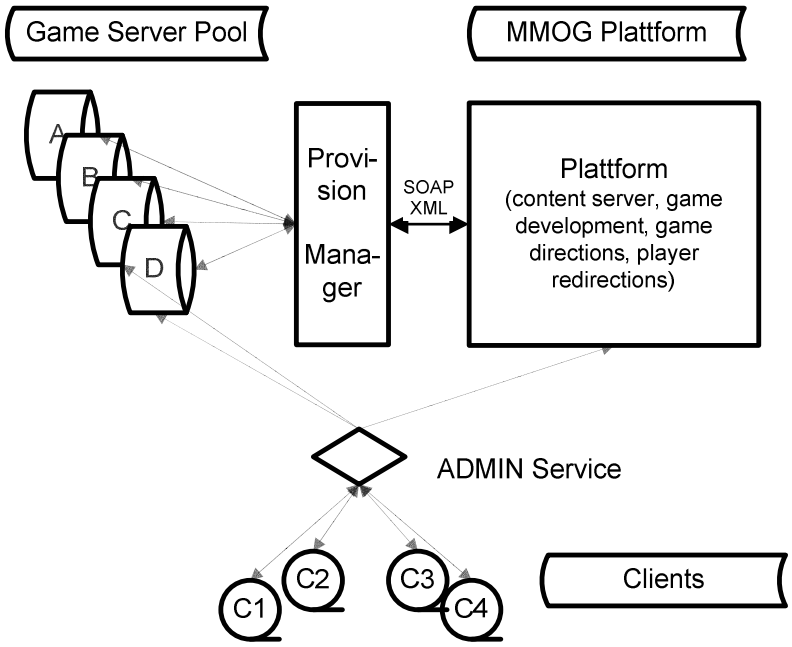


Figure 3.6. Illustration of the MMOG platform for content control and central game development in [Shai 2004]

Figure 3.6 features the architecture of the platform, the provision manager and the potential pool of game servers. The game server pool is basically the underlying hardware; a publisher can lease a certain number of servers that are capable of running the game world. The other important middleware aspect is the MMOG platform, which has a database for common game content. As a connector, the provision manager enables the game servers from the server pool to access the game-specific content from the platform, therefore the game content complexity can be increased by reusing a pool of common quests. The administrative service redirects the clients to their server and if needed provides them with content from the platform.

The design of the platform is kept as general as possible in order to support the basic game functionalities (for different game types). By requesting resources of the game

platform, a publisher will receive as many servers from the game server pool as necessary to host the game world. Depending on the game type in the MMOG sector, the required hardware can differ. For example, current MMORGPs require a large server performance to keep the persistent online environment running at all times. Other games like MMOFPS require significantly less capacity due to the smaller environment. With the service platform a provider can reduce the risk when launching a MMOG by receiving an individual server support as well as the required functionalities of the MMOG platform.

The scalability server attempt is based on a brute force technique; with a major increase in player numbers a large server cluster is needed. Other MMOGs (like Eve Online) handle the increasing number of players due to an intelligent server structure and do not only compensate them because of the integration of additional hardware. Furthermore, the MMOG platform also generates additional overhead by providing functionalities for content and player management. Common techniques like sharding can be used to split the online environment up into similar server clusters. However, using the server resource pool to support cross-functionalities in between the different shards is not explicitly mentioned. A highly loaded server could therefore use the bandwidth of a low populated server by deploying a common game instance server.

With regard to mobility the attempt does not support mobile gaming; the focus relies on massive multiplayer online game environments. Especially the aspect of reusability is mainly addressed by creating a platform with functionalities for different game types of MMOGs. Hence the server pool is a flexible way to assign required bandwidth to the applications. Even successors of the current game generation could adopt the interfaces used and thus reduce the overall amount of required development effort.

The major disadvantage of the attempt is the lack of quality of service and the high specialization in hardware expenditure. Therefore, regular backup strategies will ensure that a re-rolling can be carried out in extreme cases. By outsourcing the infrastructure to a third party application, this leads to a higher probability of data loss. Also the very specialized structure of the service platform excludes mobile- or

P2P-based attempts. User behaviour is not addressed at all, leaving cheating or bypassing login servers unmentioned.

3.4.2 Middleware Example: OpenPING Middleware

In contrast to the general middleware application for different game types, several attempts aim to create an in-game middleware support. As an example: [Okan 2004], [Akka 2004.2] and [Akka 2004] address different methods to integrate third party applications into the game. One of the major advantages is the resource speciality of any of the given attempts. By reducing the number of potentially supported games, the required data overhead also decreases significantly. Thus specialized applications are capable of coping more effectively with the game requirements. The OpenPING in-game middleware in [Okan 2004] offers several interfaces for potential base programs that cover the basic game functionalities. Furthermore, it supports meta interfaces in order to adjust the implementation strategy by focusing on the game requirements.

Figure 3.7 illustrates the example of the OpenPING in-game middleware application that runs on the game kernel. In contrast to the service platform, this attempt aims to integrate the middleware into the game application. The base interfaces offer a flexible solution for current MMOGs, including the general in-game functions (like move, send, attack, etc.). Additionally, the meta interfaces allow third party applications to enhance the game further, an example for such an application would be an improved GUI or AddOns (small programs) that a user can install. The OpenPing middleware interacts with both the basic and the meta interfaces and creates the basic rules for each external program. The commands from the interfaces are translated and sent to the game kernel (game engine), where they can be executed.

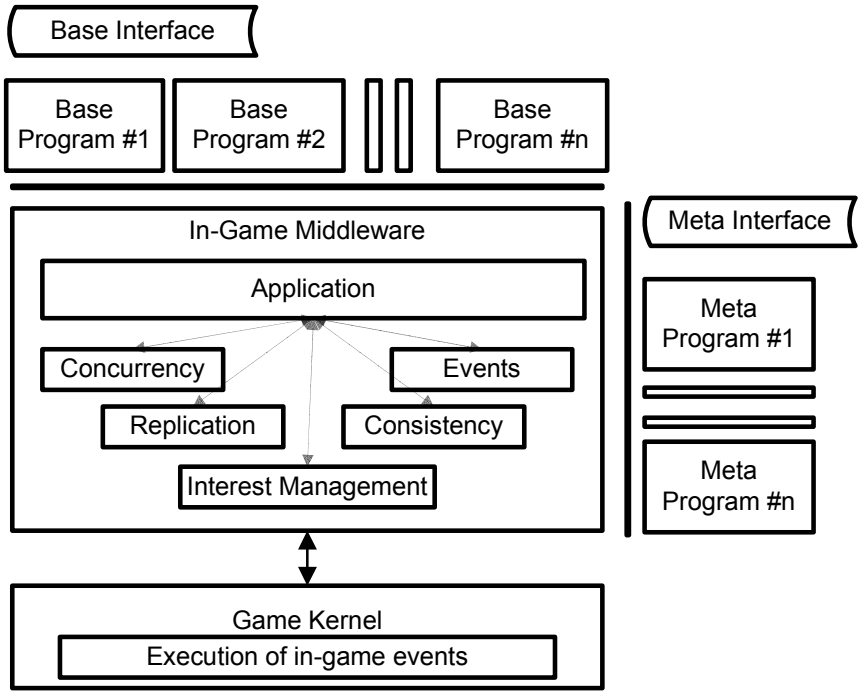


Figure 3.7. Design concept of the OpenPING platform. Inner structure and outer relations towards meta and base interfaces are shown in the diagram [Okan 2004]

The system design of the OpenPING middleware focuses on the aspect of computational reflection [Smit 1982], which ensures a uniform program structure. Based on a classification of application object behaviour, the middleware application will predict possible events, hence reacting in time and supporting the game application with the necessary resources. For example, a large number of players gathers to attempt a fight against an important NPC. In this situation, the middleware could anticipate the requirements of additional resources for the next few minutes in order to support the game with necessary bandwidth and server performance.

With the interfaces for further meta programs, the middleware can be extended with additional features without losing its inner consistency. Especially for a MMOG with regular updates, a generic update of the middleware ensures that the current predictions are still effective.

With regard to scalability, the middleware offers the design of light-weight applications to predict the players' behaviour. Nevertheless, due to the complexity of persistent online worlds an empirical testbed is missing, therefore the prediction with

very large number of players (10000+) is not documented. Another important aspect is the additional computation time that would be required to predict each possible event. Especially with a seamless online world, handovers between zone servers also need to be taken into account, hence further increasing the complexity.

The focus towards in-game middleware application for massive multiplayer online games results in a missing mobility factor. However, other related projects aim to support the mobile environment with a fitting J2ME-based middleware solution like [Pell 2005] and [Open].

Furthermore, due to the high specialization, the reusability of each in-game middleware application is low. Only a possible successor could re-use the existing interfaces if the game design is related enough. Nevertheless there are certain advantages as well, like the high quality of service. By focusing on a single game and implementing a middleware application, this can further support the effect of persistent data backup. By predicting important in-game events, an asynchronous backup can be carried out, thus further reducing the probability of losing important data. For example after receiving an important item, an individual character backup would ensure that this item cannot be lost if the server crashes.

The in-game architecture also includes potential user behaviour, creating reactions for in-game events. By effectively predicting the behaviour of a user, the game content and the in-game stability can be improved significantly. Although the additional effort for calculation requires hardware and more user information (position, last item used, last action, etc.). Hence the hardware expenditure is rather limited, excluding resource scarce devices and highly distributed mobile environments.

3.4.3 Patch Scheduling and Middleware Support

This attempt [Cham 2005] offers a middleware solution for different MMOGs. In persistent online environments, the player numbers vary depending on the real-life time. This variance leads to a recognizable difference in streaming bandwidth during the different days of the week. By ensuring that the client software is continuously updated, the game provider needs to patch each client when in-game changes have been made in order to prevent cheating or asynchrony.

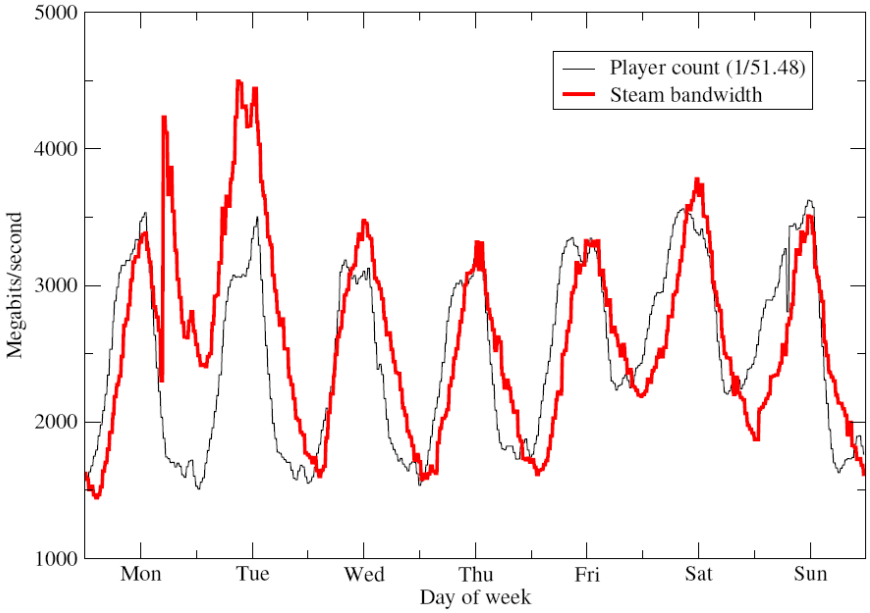


Figure 3.8. An overview of the required bandwidth with regard to the time and the day exemplified by a Counter-Strike server from [Cham 2005]

Figure 3.8 illustrates the basis streaming bandwidth of a Counter-Strike server. The authentification tool is called steam, which gives each user a unique ID and uses check-sum techniques in order to ensure that the local data matches the newest patch. The example of Figure 3.8 shows a popular US server and its upload towards the steam tool in MB/s. As one can clearly see, a daily routine exists because the peak times (afternoon and early evening) feature a generally higher level of required bandwidth. Additionally, on Monday a new version of Counter-Strike was released and the clients needed to update their local version. The resulting effect is that the average bandwidth as well as the peak bandwidth on Monday and Tuesday increased considerably.

Patching the client creates a significant amount of further bandwidth that is needed to support each user with the newest software update. Therefore, a middleware application in [Cham 2005] introduces how to distribute the additionally occurring bandwidth overhead. The application automatically scales the amount of potential bandwidth with regard to the overall number of downloading clients. Hence, it reduces the peaks during the first hours after the new update has been released and encourages players to wait for a few hours in order to receive a better performance.

It is also conceivable that a multi-agent attempt can be used [Bare 2006] or a P2P system to distribute the new update, although this can lead to a significantly increased chance of abusing the P2P data exchange by uploading modified data.

With regard to scalability, the attempt offers a self-regulating mechanism to scale down all currently downloading clients. The more clients downloading the update simultaneously, the less bandwidth they will receive, even if more potential might exist. Hence this attempt is scaling with two effects: first of all, some clients will avoid the peak hours after the patch release, which will smoothen the request rate during the first few hours. The second effect will be a public server mirroring by the gaming community; they will probably create further servers to download the current patch. Thus the patching mechanism will smooth the distribution of the additional bandwidth, although less bandwidth can also lead to negative feedback because many users are not willing to wait until they can play again. This negative effect reflects the low inclusion of player behaviour analysis in the bandwidth attempt.

Neither mobility nor quality of service is explicitly addressed in the attempt. It is possible to use the given method to distribute data in a mobile environment as well. Depending on the cost structure (pay per minute or pay per data rate) it will either not improve the given situation at all or it will even make it worse by reducing the potential data rates.

The reusability of the patching middleware is supported due to the generic design structure. The problematic field is similar in most of the current MMOGs, especially in the most frequently played games.

3.5 Player Behaviour

During the last five years, the aspect of online communities and social interaction has achieved significant importance for the game design. Most of the successful publishers realized that integrating the player community into the creation process is a necessary step in order to create long-term relationships with each customer. Hence the quality of in-game content clearly became more important. Due to the creation of persistent online environments it was necessary to expand the balancing and quality testing software phase. One result is the creation of beta test periods that allow the publisher to gain a better understanding of the current game quality.

Another important aspect is the interaction in virtual environments (VEs) as described in [Mann 2000]. By spending a considerable percent of leisure time on the MMOG, the players begin to set up their own language and behavioural rules. If the game does not offer sufficient support, then the community often looks for ways to improve the current situation. For example, a missing trading system in a MMOG is often bypassed by forums or the creation of special channels. On par with the players' motivation, [Klas 2006] illustrates why it is important for a publisher to build up long-term relationships and to listen closely to the community's needs.

However, not only does the player behaviour influence the creation process of modern game design. As [Vill 2006] illustrates, each of the games also requires a very specialized input, often twenty or more keys are used to send game engine commands (mainly movement and actions). Thus the input device addresses the need for a flexible way to support the users' needs. As the results show, it is possible to further improve the quality of games by changing the input opportunities. Furthermore, [Seha 2006] also introduces methods for alternative execution control with regard to player opportunities.

Although player behaviour often strictly refers to electronic computer games, there are also methods to integrate real world devices into gaming. The example of air hockey over a distance in [Muel 2006] shows the importance of user comments and behaviour for the development of new game mechanisms. Strategies to improve the design of an air hockey table were addressed by listening to the test players, a clear similarity to the beta testing phase of MMOGs. One of the largest issues addressed by the users was the feedback of the air hockey table. As soon as the perception of a player does not correlate with the output of the game engine (whether electronically or physically), the user will try to interpret the situation and learn from possible mistakes as [ElNa 2006] shows. In the example of air hockey [Muel 2006], the devices were not accurate enough, hence creating a distracting situation.

3.5.1 Cheating in Computer Games

One of the most significant problems addressed in the research field of player behaviour is cheating. In order to understand the motivation of a cheater one can use the taxonomy of Bartle [Bart] to analyze the players' goals. Nevertheless, there are

many more reasons to cheat; the main motivation is the fast success. Therefore, the first step towards protecting the in-game environment from cheating is to analyze existing methods of corrupting the game, as listed in [Yan 2005].

Most cheating is either based on third party applications or on delaying own actions as long as possible to react to potential actions of other players. An example of a large-scale cheating history can be found in Counter-Strike [Coun]; the high level of competition and the relatively old game engine results in an extremely professional and sizable cheating community. Methods to protect especially FPS and RTS games are introduced in [Agga 2005] and [Mönc 2006].

Generally, a P2P environment with distributed game data is more vulnerable for cheating attempts than classic server-client architectures. Therefore it is risky to use one of the different attempts in order to implement P2P or hybrid structures in a massive multiplayer environment [Kabu 2005]. The advantage of lower overall bandwidth for the publisher can lead to an expansion of the cheating community on the other hand.

The attempt to realize new game mechanisms without enabling the players to abuse them for their own advantage is introduced in [Smed 2004]. The new game effect of “bullet time”, referring to the slow-motion used in [Max] Max Payne and [Ente] Enter the Matrix, is included in a multiplayer environment. By slowing the environment down for a player, the user has more time to react and can perform more accurately. Hence the implementation of a local perception filter enables the environment of a single player to be slowed down by using temporal distortions. The corresponding game effect is a significantly slower environment for a single player while the rest of the game environment runs in real-time. The individual zones of temporal contour are shaped like a cylinder, as shown in Figure 3.9.

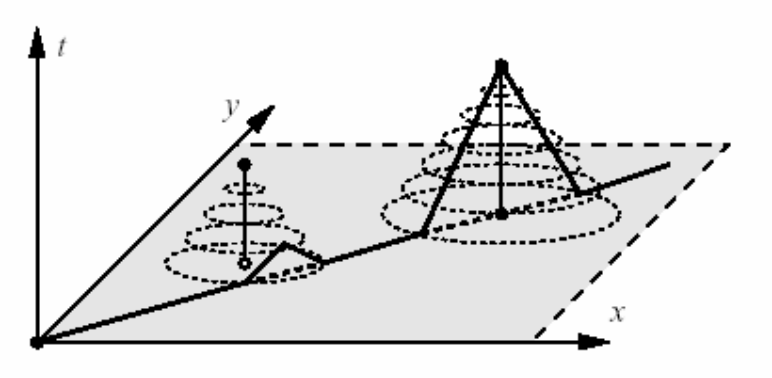


Figure 3.9. Figure of a 3D illustration of two virtual players with linear delay functions in order to build a temporal contour (necessary to realize slow motion and bullet time effects in multiplayer games) [Smed 2004]

Although each of the players has their own perception of the game world and temporal asynchronies are allowed, the effect of bullet time can be limited to a single player. However, the longer the effect is active, the more frequently differences between game states occur.

With regard to scalability this attempt has a major downside. Due to the limitation of the local filters, players cannot interact directly with one another. Rather, all of the interaction takes place by using passive entities, thus the closer a player gets to another one, the larger the temporal distortion becomes. Usually, player interaction increases with the number of potential other players, in MMOG environments the users tend to distribute unequally. Especially in MMOFPS with multiple dozens of simultaneously acting users, the effect of temporal distortion will not scale with a growing number of players.

The aspect of mobility is not addressed in the attempt. The assessment parameter hardware expenditure is high because the resources needed to calculate the effect of "bullet time" can be calculated on the client's computer. Generally, it is also possible to use the asynchronous game concept in networks with higher latency as well. However, by limiting it to a specific genre and a very specific game engine design, it is not possible to reuse the attempt for many other games. Implementing the effect of bullet time in a multiplayer environment requires significant conceptual decisions, thus making it hard to transfer to other game genres like MMORPGs.

Both aspects, user behaviour and quality of service, are addressed as the main focus. By implementing a non-existing effect into a multiplayer environment, the attempt generates a new way to interact in virtual environments. By strictly limiting the bullet time effect to the local perception of each user, the attempt avoids data loss due to game inconsistencies.

3.5.2 Categorization of User Behaviour

The categorization of users and their behaviour has become a research field for game development over the last few years. The main goal is to understand the motivation of a user and therefore predict possible behaviour. Most of the current classifications are based on the taxonomy of Bartle [Bart], who divided players into achievers, socialisers, explorers and killers. Many further effects like third places in online worlds [Duch 2004] could be explained by using the model's motivation factors.

The creation of test environments for specific game types further enables the underscoring of how important effects like latency, jitter or package loss are. The highly varying requirements of each game type influence the players' reaction; hence each of the games has its impact on the level of influence with regard to the users' performance. Examples for the measurement of user performance are shown in [Beig 2004] and [Shel 2003], evaluating the FPS and RTS game types.

Both attempts underpin the importance of user behaviour for network researchers. To gain a deeper understanding of the player interaction in MMOGs, the attempt in [Chen 2006] focuses on the analysis of network traffic. By comparing real world locations with in-game locations and behaviour (such as team setup, friends, etc.) it is shown that neighbours and team mates tend to be closer to each other in the network topology. The Venn diagram in Figure 3.10 attempts to understand the mechanisms for this correlation. Generally, every player can be seen as an independent actor, sub-groups are solo players or socially interacting players. It is important to understand that a single player can have attributes of more than one sub-group, depending on the situation and game. For example, a user could be strongly group oriented in a MMORPG and additionally plays a FPS just for fun. During the game time in the FPS, the user wants to play alone. This example would illustrate different behaviour, which is based on the game type.

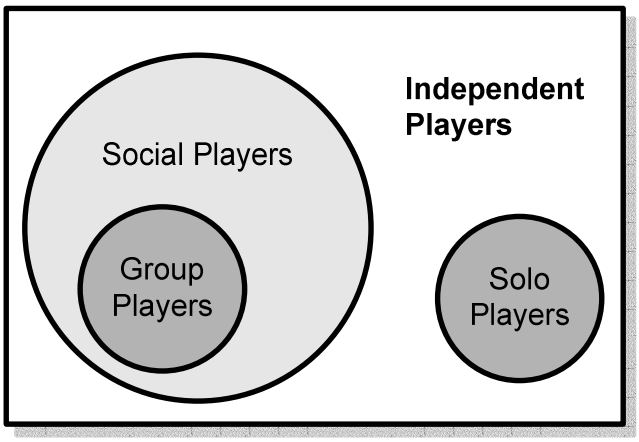


Figure 3.10. The Venn diagram for player classification: The main focus relies on group vs. solo motivation [Chen 2006]

The attempt classifies online players into sub-groups with regard to their in-game behaviour (for instance neighbour, partner, etc.). The game time, session length and online behaviour of each sub-group is analyzed separately. The idea behind the diagram is to analyze whether several users, which are locally neighbours in the real world, also have similar online behaviour. An example of group players would be several users that live close to each other and who started to play a MMOG on the same server. By evaluating their behaviour one can predict needs, an advantage of this technique is that a publisher might want to know how the user group is distributed. Game servers near the location of the majority of users would help to increase the game performance.

Within the section of user categorization the attempts towards game type-specific analysis are strictly limited to either the game or the related game type. Hence, the attempt [Chen 2006] focuses entirely on a medium-sized MMORPG. The results of local player aggregation and the closer position in the network topology can be a social effect of the Asian gaming culture. Furthermore, it is unclear if this effect will remain stable with regard to an increasing number of players per realm.

The attempt does not include the aspect of quality of service, like a data management or game quality for the analyzed players. Furthermore, the strict focus on a server-client model and a massive multiplayer environment also excludes the factor of mobility. Although technically, it is possible to gather local neighbours by using the

cellular structure of mobile phone devices (GPRS/UMTS) and to design sub-groups depending on the physical position.

Nevertheless, the main focus of the attempt relies on the user behaviour analysis. Based on the Venn model, the statistical analysis clearly indicates that the distribution of the current player community does not allow further clustering or local fixed-sized partitioning. Additionally, the attempt offers an opportunity to analyze bigger MMORPGs in another social environment in order to compare the results. Therefore, a reusability for future surveys containing MMORPG player behaviour exists.

3.5.3 Game Influence in Real Life

In order to understand the external effects described in [Mysi 2006], one must first be aware of in-game interaction and the design of a social culture. Persistent online environments have created a completely new form of interaction rules, as shown in [Jako 2003] and [Chen 2006.2]. The social interaction in MMOGs [Chen 2006.2] therefore turns out to be the major influencing factor, especially if the user spends a large amount of leisure time playing. By creating new virtual spaces (persistent online environments) the social interaction in between the generation of gamers changes fundamentally.

A topic that has often been addressed in the media is the influence of gaming on real-life behaviour. It is known that frequent behaviour has an influence on human socialization. Hence, hardcore gamers with only a few or no other real-life activities besides gaming are very likely to easily adopt the virtual behaviour. Especially games with content focused on violence are often considered to be a major influencing factor for people to perpetrate heinous crimes. Positive and negative external effects are evaluated in detail in [Mysi 2006] and [Fran 2006] in order to analyze the significance of influence.

Frequent game playing does not only have negative effects. A list of possible positive effects for frequent users of MMOGs is evaluated in [Fran 2006]. The research focuses on the increase in typing capability and knowledge acquired by playing a MMORPG. However, the observed behaviours and skills are very closely linked to the usage of computers; it is understandable that by frequently using the

keyboard to communicate with others, the typing skill will not decrease. Also the increased in-game knowledge is clearly addressable for the players' basic motivation to understand the game mechanisms and improve his/her character. However, it is also shown that other non-related positive effects occur like increased potential to use the Internet for research. It has been statistically shown that the increase is stable over time and leads to a deeper understanding of using next generation media. Social interactions with other players have also been observed in depth and common behaviour [Chen 2006.2] has been frequently adopted by the new players. General social rules were accepted and even further transmitted to other players.

With regard to scalability (statistical scalability of the method used and not network scalability), the attempt is aimed at a very limited number of test subjects. Hence, the evaluation focused on pre-selected targets which seemed to hold no or only a little potential for extraordinary real-life behaviour. The attempt did not focus on the mental state of each participant and the major player community as targets. The methods introduced to gather the required players' data clearly does not scale with a larger number of participants.

Also the game structure for the attempt was a common server-client MMORPG, therefore not taking mobility aspects into account. The attempt does not include the underlying architecture or devices used for measurement (except for the keyboard), which slightly limits the possible input devices. Furthermore, quality of service mechanisms or data management was also not included. By losing permanent game data like characters, it is most likely that many players will quit the game.

The major focus relies on the user behaviour analysis; with verified results and more influencing factors it would be possible to determine whether long-time correlations between effective usage of modern communication structure and online gaming exist.

3.6 Summary

Varios attempts have been presented within the related work section. Each of them mainly focuses on one of the four central aspects: mobile gaming, massive multiplayer gaming and middleware design as well as player behaviour analysis.

Due to the complexity of the set of problems, most attempts offer a single solution for a specialized problematic area of next generation game design. Especially the MMOG section features various aspects that cannot be easily resolved (like latency, scalability and combination with the mobile environment). Nevertheless, each of the attempts is kept as generic as possible, therefore supporting a larger number of potential games.

Table 3.2 gives an overview of the attempts in the related work section and illustrates the advantages and disadvantages with regard to the assessment parameters: scalability, mobility, reusability, quality of service, user behaviour and hardware expenditure.

Table 3.2. Assessment of the related attempts, using the six game-related factors.

	Scalability	Mobility	Reusability	QoS	User Behaviour	Hardware Expenditure
Mobile Aware Games						
Asynchronous Mobile Gaming	- -	++	0	-	-	++
RFID Tags	-	++	+	- -	-	+
Massive Multiplayer Games						
Interest-based AIs	++	-	+	-	- -	0
P2P MMOG Architecture	++	0	++	- -	- -	+
Public Server Architecture	0	-	++	+	- -	-
Gaming Middleware						
Service Platform	-	0	++	-	0	- -
OpenPING Middleware	+	0	-	+	+	0
Patch Scheduling	+	-	+	0	- -	0
User Categorization						
Bullet Time in Multiplayer	- -	0	-	+	++	+
Player Behaviour and Design	0	-	+	0	++	0

4. Approach I: Understanding Player Behaviour

> "The right word might be effective, but no word was as effective as a rightly timed pause", Mark Twain.

This chapter introduces the player behaviour approach towards the current gaming situation. Hence, the underlying architecture of the survey database as well as the online questionnaires "Distribution of Online Player Behaviour" and "Virtual Fragmentation" are introduced. This description includes the operationalization and scales used for the creation of the surveys. Furthermore, the advertising strategy and selection of the user group is explained in detail.

4.1 Motivation and Overview

Today, the Internet features over 120 different MMOGs with various settings and scenarios. That even includes new game types like UMMORPGs or MMOFPSs. All of them have a common factor: one player always represents a single character (toon) at a time. Thus the identification with one's in-game character is significantly higher than when compared to more dynamic games like common FPS where the in-game character changes more frequently.

This leads to the question of in-game behaviour. With the release of the first MMORPGs, both the player community and their way to play games were completely unknown. The so-called first generation MMORPGs were played for various reasons (such as having the opportunity to play in a persistent online world and new character evolution depth) [Lazz 2006]. Thus discrepancies of the players lead to their game demeanour [Jako 2003]; second generation MMORPGs subsequently featured player guides and dictionaries for in-game etiquette and speech to give the new audience an insight into the online worlds. Furthermore, the more experienced players from the first generation MMORPGs also created an informal set of behaviour rules that influenced newcomers. Like in real environments, players needed to find a common way to interact with each other. The

changes mentioned also entail the question about how much of the real player attitude actually remains.

Another important aspect in persistent online games is the influencing factor of competition. Depending on the players' motivation [Bart 2004] this factor varies highly from one person to another. The aspect of competition can generally be divided into solo/group competition and PvP/PvE competition. Table 4.1 gives an overview of the resulting combinations from both dimensions with a pair of example games for each section.

Table 4.1. A two-dimensional illustration of competition possibilities

	Solo Competition	**Group Competition**
PvP Competition	RTS – Warcraft III FPS – Quake	Tactical FPS – Counter Strike RPG – Guild Wars
PvE Competition	Classic arcade games or browser games	RPG – Everquest2

Solo competition describes the gaming style of a single player who aims to reach the highest possible score compared to other single players. It does not necessarily imply a pure 1 versus 1 situation; therefore the classic death match (a free-for-all game mode where each player fights against the rest) would also be classified as a solo competition. The "solo" aspect rather refers to a prohibition of teaming between rivalling players. A common example of solo competition is RTS. The game engine takes on the role of a judge and thus either allows or prevents actions. Moreover, the term of game balance is important for competitive games.

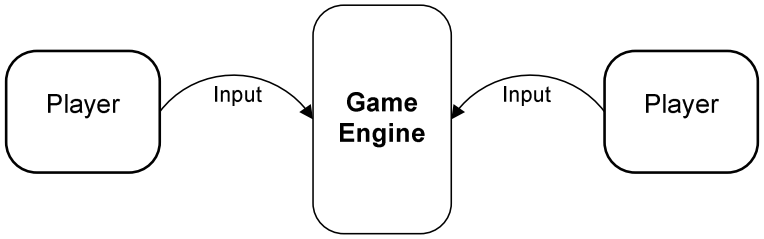


Figure 4.1. The solo competition – the game engine acts as a judge

Group competition on the other hand includes the explicit creation of a team to fight against another team. Two or more pre-arranged teams are trying to reach a goal. The single player acts as a part of the team; winning or losing no longer only depends on the single person. Additionally, the game mechanisms change fundamentally when compared to a solo competition game. The idea of a role distribution (like healer, damage dealer and protective class), with individual task fields, gives the group competition a higher level of complexity. Most of the current competitive SG and FPS feature group competition as the main multiplayer element.

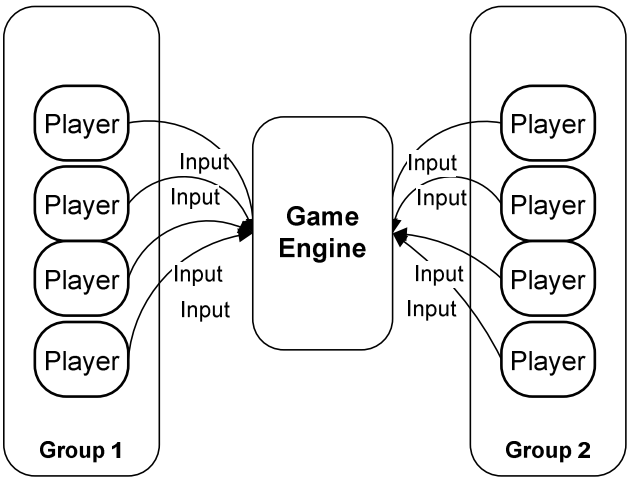


Figure 4.2. The group competition – an example of two different groups competing

A totally different classification is carried out by the dimension of PvP vs. PvE competition. This has nothing to do with the above-mentioned solo vs. group competition – it is a completely different separation.

PvP is the abbreviation for "player versus player". By demonstrating higher scores or kills one user wins. Examples of that are typically 1 vs. 1 FPS games or RTS games. Once the game is over the results can be saved; meanwhile playing is simultaneous (both players need to interact in the same virtual environment). The requirement of simultaneous gaming excludes asynchronous games like browser games, that feature a high score table. Generally, a player needs to interact directly with other human opponents in a PvP situation. For example, two clans are fighting against each other and the results are saved in a ranking system, as shown in Figure 4.3. Exceptions

from this rule exist, although for differentiation in this thesis, the PvP aspect focuses on the direct player interaction.

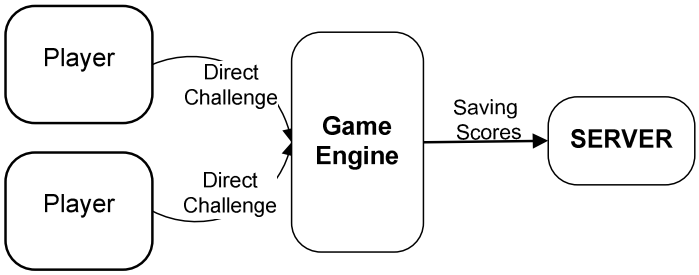


Figure 4.3. The PvP competition – the game runs simultaneously: Afterwards the results are saved on the server

In contrast to the direct comparison of skills against other human players, the PvE competition focuses on achievements (i.e. reaching something most effectively/the fastest/first). Player versus environment (PvE) implies that the opponents are controlled by the computer (AIs). A good example of a PvE challenge is the unique "boss-type" in MMORPGs that require multiple players to team up in order to eliminate them.

The aspect of challenge in this context means that the players do not play directly against each other. Rather they work on a goal set by the game environment. Whoever reaches this goal first "wins" the competition.

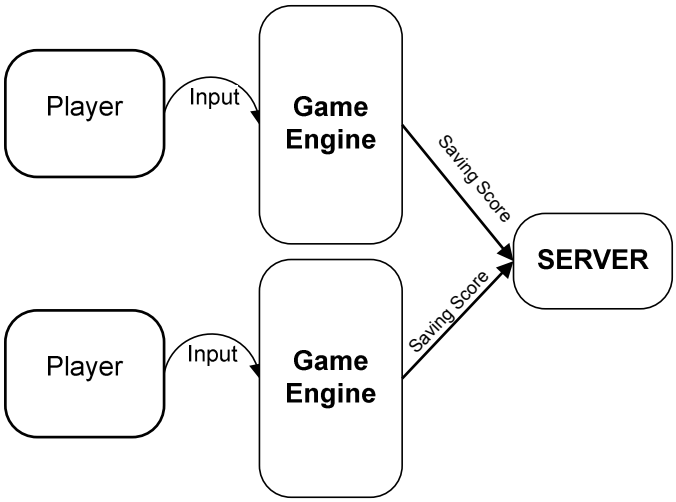


Figure 4.4. The PvE competition – the game runs in two different instances and asynchronously: after both players/groups have completed their tasks, the results will be saved on the server

Typical examples of indirect PvE challenges are MMORPGs. The game world is divided into instances, which mirror the same encounter (the challenge) to different groups of players; speed and efficiency are measured and compared afterwards. Figure 4.4 illustrates the situation.

Especially with regard to the time invested in a single game, "pro-gaming" (professional gaming) is one of the most recent results of the increasing importance of video games in multimedia entertainment. The first professional gaming scene was developed in Korea, where players earn money for professional electronic sports competitions. With the increase in network technology and multiplayer game support, each of the competitive game types (RTS, FPS and SG) established their own pro-gaming tournaments. Up-to-date tournament series like CPL (Cyberathlete Professional League) or ESL (Electronic Sports League) feature a wide variety of current competitive game favourites with total winning prizes of $100,000 and more. These high values once again reflect two important facts: the growing acceptance of gaming as a sport and the interest of sponsors to reach even more players.

Sweat-shops on the other hand are the downside of the growing importance of games. Virtual worlds often contain goods such as items or treasures. With the tremendous success of persistent online worlds these goods received real world values [Mann 2000]. eBay and even the game companies themselves offer an easy way to sell these virtual goods.

As a result of growing interest combined with real world values, third world workers are playing in so-called sweat-shops in eight to 12 hour shifts in order to receive re-sellable virtual goods. The publishers of the MMOGs react in different ways. Sony Online Entertainment, for example, opened its own virtual shop for Everquest and Everquest 2, where players can buy in-game items for dollars. In contrast, Blizzard Entertainment completely prohibits the re-selling of virtual goods or accounts, because they claim that they are the company's virtual assets. From a players' perspective, both reactions have less to no influence on illegal re-selling, the higher

the demand for virtual treasures, the more companies will provide them. A possible solution for this is a change in the game design, as long as inequalities (different level of game achievements) in the game motivate players to pay money in order to improve their character, the sweat-shop industry will continue to gather virtual resources.

4.2 Technical Background for the Player Analysis

The player analysis includes considerable psychological factors, although it is vital to pinpoint that the realization requires several technical aspects. These aspects will be the focus in this dissertation; because each of them will be evaluated in depth.

An analysis of player behaviour is the starting point to gain a deeper understanding of next generation game design, because users define the way that games are played and therefore they are responsible for upcoming trends. Thus it is necessary to create an online questionnaire which can be reused for further studies. Also the questionnaires themselves must be designed correctly: especially with regard to a valid statistical analysis, the factors of reliability and validity are the main aspects for a good survey design.

A further aspect is the multi-language support, in order to clarify the online survey as much as possible, it is necessary to support different languages for the web pages. An automated configuration with an appropriate database design is required to provide the multilingual support in general (this also includes further potential surveys).

The main part for a general statistical analysis of the gathered information is the extraction of data sets from the survey. It is important to keep the data consistent and to eliminate faulty data sets. This includes both incomplete data sets as well as unrealistic answers (like age: 15 years, marital status: widowed).

4.2.1 Operationalization

Before one can technically implement a survey, it is necessary to evaluate other possible interviewing methods in order to decide whether an online survey is the best method to gather information about the users' behaviour. Furthermore, an academic questionnaire needs to be reliable. Common methods to gather information about

users are standardized interviews, non-standardized interviews and online questionnaires. Each of the methods has its own set of advantages and disadvantages. A detailed description of the different research design, methods and structure of a survey can be found in [Kuma 2000].

Generally, one can differentiate between three academic approach techniques: exploratory research, descriptive research and causal research. In this case descriptive research will be used. In fact: "*The purpose of descriptive research is to provide an accurate snapshot of some aspects of the market environment*" (Kumar, 2000, p. 60). For the gaming scene this means, in particular, that this approach aims to point out the current players' motivation.

First of all, it is necessary to use an appropriate measurement method. Generally three different methods to gather users' information exist: the telephone interview, the personal interview and the mail survey [Kuma 2000]. An online survey falls into the category of mail surveys. The main argument for online surveys is the expected large number of participants. Especially for statistical analysis it is absolutely essential to have a large set of answers in descriptive research. Furthermore, the close relation between the Internet as a platform for the survey and the frequent use of the Internet by the gamers also prefers this interview method.

Another aspect for qualitative research is whether the online questionnaire has open-end questions or closed questions. This decision depends on the goal of the survey. If detailed information about very special aspects of games is needed, then open-questions are the appropriate method. For more generalized information about player behaviour the online survey needs closed questions with a limited set of answering options. The possible sources of problems for online surveys also need to be addressed. Therefore, the questionnaire must be as accurate as possible.

One needs to take the possible sources for missing reliability into account for every questionnaire. The main problems are: ambiguities, chain questions, expert questions, leading questions or the interviewer influence. The interviewer's influence can be omitted for online surveys.

Ambiguities: can occur if a term has multiple meanings. Especially in a gaming content, it is important to clarify the statements before asking a question. This can be done by using examples so the user understands the exact meaning of the question.

Chain questions: are a set of questions which are closely related to each other (same topic) and guide the interviewee through the assumed answers. Therefore, they also have a strong influence, which can in turn lead to scientific inaccuracies.

Expert questions: like ambiguities, the missing understanding of technical terms or unexplained words can lead to reactance and wrong answers. In order to prevent such reactions, one must explain every special term before asking the question.

Leading questions: these questions give the interviewee a direct hint about the socially acknowledged or "correct" answer and therefore this will significantly influence the results. All of the questions in the online survey must be clear but as neutral as possible so as not to influence the user.

Although the approach of understanding the players' behaviour also includes reusable technical implementation, it is necessary to point out that the questions need to be designed for each survey individually.

In order to decide on the dimensions of a single survey, the users' available leisure time is an important aspect. It is assumed that online gamers spend a limited amount of time on scientific questionnaires, since many users are motivated to maximize their game time [Fritsch 2006.3]. Hence one should limit the overall amount of items in a single questionnaire, in this thesis the maximum number of items is set to 20 - 30. This decision is based on the pre-survey of [Fritsch 2006.3]; the users reported that a set of 50 questions is too long whereas 20 - 30 questions are appropriate. Furthermore, pre-standardized answers increase the likelihood of complete and well-structured answers, and many users prefer standardized answers over open questions.

Another important aspect is the design of scales for the online survey. The measurement can be differentiated into four different types of scales: nominal scales, ordinal scales, interval scales and ratio scales. Depending on the motivation, one needs to decide which scales should be used for the questions (compare to [Kumar 2000]).

Both online questionnaires of this thesis use an interval scale with index numbers, since the scale with five items offers the user a limited amount of differentiation. However, the main advantage is that most users can answer correctly. Furthermore, the results can be compared statistically (each answer receives a value, average

values can be calculated, etc.). In general, the questions are designed as statements and the participants need to decide whether they agree or disagree with the statement. This technique is very similar to the Lickert scales [Kuma 2000]. Each question offers five different answers that are gathered by appling the Mean opinion score (MOS translates into Mean Opinion Score, with regard to this questionnaire it ranges from the value 1 = strongly disagree to the value 5 = strongly agree).

From a scientific point of view, the MOS is usually used for descriptive research with a technical background. Compared to the scale of Lickert, which uses a score from +2, +1, 0, -1, -2 instead of 5, 4, 3, 2, 1 the difference in the analysis relies on the results. By applying the MOS, the expected median will be “3” instead of “0”. One advantage of the Lickert scale compared to other single item scales (like a comparative scale or a constant sum scale) is the understandable survey design [Kuma 2000]. Multiple item scales (more dimensional questions, tables, etc.) would offer a more complex survey design; although without previous data or samples a more complex design runs a high risk of eventual errors. With the purpose of gathering underlying data for further scientific attempts, the MOS five point scale for a simple and understandable design of the online survey is used. The precise list of questions for each survey can be found in the appendix.

4.2.2 Database Architecture

First of all it is necessary to decide how an online survey should be implemented. At [Dmoz] one can find a list of service suppliers for polls and surveys. Most of them use similar implementations for surveys; however, a fee is generally charged per questionnaire. Another option to implement a questionnaire is to design an appropriate database architecture and to create a system for further surveys. The main advantage of an individual database system is that once it is created, the effort to create a follow-up survey is reduced substantially. Generally, the database architecture has a cost advantage in the long run, because one does not need to pay for every survey. With more experience and multiple questionnaires one can also use the learning effects to further improve the database system, for example, by integrating a new scale for the next surveys. The downside of this design decision is the initial effort required to implement the system. Based on the fact that the research

field of computer games still requires considerable rudimentary information about the player behaviour, the advantages of creating an own database compensate for the initial effort.

An appropriate database design is one of the main aspects for a large scaling questionnaire, thus it is important to make the database as reusable as possible. Basically, two different techniques for data collection exist: the database and the file system. The main difference between them is that the database requires an additional server which runs the storage structure. As an alternative implementation method, the file system stores the answers in a file (the internal format can be chosen accordingly) and either creates a single file or stores each of the answers separately.

The major advantage of a database system is flexibility. Once data is gathered, it is still possible to view only a certain subset or rearrange the order, which would entail additional work for the file storing system. The database with its DBMS (database management system) offers the greater flexibility and reusability, which clearly promotes it as the chosen method for the surveys. The RDBMS (relational database management system) is used to design the database (most common form of database design today).

Moreover, another important aspect is the independence of the web page design and the underlying database. In the user survey case, the ASP (active server pages) technique is used to generate the web interface automatically. One advantage is that these web pages are created during runtime and offer a graphical interface for the user. Not only the design, but also the language of the questions must be appropriate. Each of the users can access the web front-end from a simple browser, answer all of the questions, and results are then stored with a unique ID in the database.

Due to the survey's large-scale, it is necessary to consider that an interlocking problem can occur. As soon as multiple users try to submit their results at the same time, the database needs to be writing protected for a short period in order to prevent that two results are stored with the same ID. The concrete implementation includes the usage of session IDs (on the application level) and transactions (on the database level). Basically, the technique generates a unique ID as soon as the user loads the webpage. This identification number is used to keep track of each query that a user is performing (in the case of a user survey he/she is only performing one query most of

the time, which is the submission of answers). Nevertheless, one can also perceive a survey that requires multiple pages and sub-queries during the answering process. In fact (with multiple pages) it would be possible to restore a corrupted session by using the given session ID. Hence the transaction design offers the most flexible solution and prevents deadlocks with the simple answering mechanism.

Furthermore, the problem of multiple participations needs to be addressed in this context. One of the downsides of an online survey is the limited control over the user group. From a technical point of view, it is problematic to prevent a user from submitting his/her results multiple times. Possible solutions for this problem are either the usage of personalized (email address-based) accounts to participate in the vote, the usage of cookies (which are stored locally and indicate that the user has already participated) or IP-catching mechanisms (further storing of the IP to prevent a user from submitting multiple times). Besides the fact that each of the methods contains an additional programming overhead, it is also obvious that all of them can be avoided by the user as well. This aspect will be discussed in-depth in section 4.2.3.

The decision on an open participation reduces the above-mentioned programming overhead and also increases the potential number of participating users (because creating an account will significantly reduce the likelihood of a gamer participating in the survey). Nevertheless, the tremendous advantage of having a large and reliable survey prevails over the disadvantages.

Figure 4.5 illustrates the ER model (entity relationship model) of the database and gives a graphical overview of the structure. Each of the four sub-tables contain one of the entities: survey, questions, answers and users.

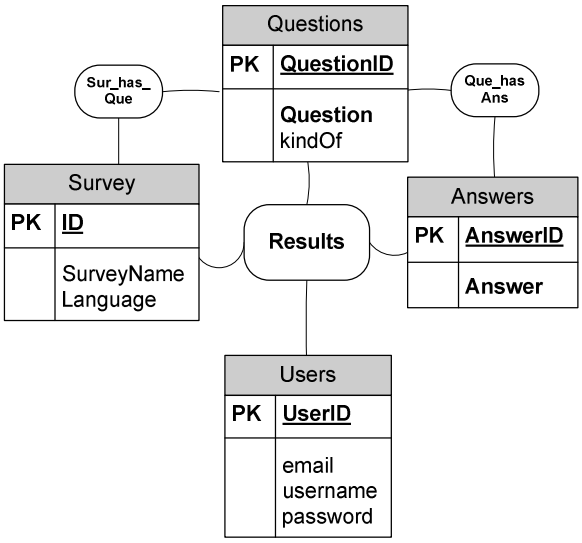


Figure 4.5. Illustration of the database with ASP support, all entities are connected with a single central relation

In order to clarify the assignment of entities, each of them is described in detail:

Survey. This table contains the general parameters like the survey name, the caption for each survey section as well as all available languages. Each of the surveys can be identified due to a unique number. Especially for the administration of data warehousing (coordination and information seeking in huge data pools) it is necessary to select each of the survey answers specifically.

Questions. This table contains the complete set of all questions that are used in the surveys. Each of the questions has its unique ID to identify them faster and connect them with the appropriate surveys. Also each question has a string with the question formulation itself. It is vital to understand that the same question in other languages will still have the same ID, which is important for data gathering after the survey is completed.

Answers. The answers table contains all of the questions' answer options. Due to the high amount of standardization most of the questions have similar answering patterns. The language support for this table includes a translated version of the answer for each of the languages supported (Spanish, English and German).

Users. This table includes all gathered information about the user. For the two game-related surveys these data sets mainly contain demographic values such as age,

location, gender, occupation and so on. By separating user information from answers gathered, one can extend the current usage of the database. A possible usage case for upcoming online surveys would be a unique ID for each participant. This ID allows the user to participate in multiple surveys without resubmitting the demographic data each time, since it is already stored in the user table.

As soon as a user submits his/her answer, a new data set is created with all of the user's answers and demographic values. Table users can therefore be regarded as the result table, which needs to be analyzed as soon as the survey has been completed. By using the central relation, each of the results can be directed toward a specific question and answer. These are stored as numbers and can be identified through their unique IDs. Especially for a statistical analysis the method of storing numbers instead of the entire answer further reduces consecutive work.

The order of questions in the online survey is determined by the front-end design. The ASP pages are created just in time and contain all of the survey questions. If one wants to rearrange the order for the user, then this needs to be done in the .NET framework, where the ASP pages are created. When the user submits his/her answers, interactive elements of the webpage (such as radio buttons) are read and the results contain information about the answer and the related question. Therefore swapping two questions in the ASP page would just rearrange the order of the results during submission. Since every answer is submitted with its corresponding question ID, they can still be identified explicitly.

The central relation "results" connects each of the four tables together and ensures that as soon as the user submits the answers through the ASP front-end, the information will be stored correctly. Furthermore, each survey can be easily extended by additional questions. The expansion for a new survey, new questions or new answering mechanisms is easy, because the central relation keeps the database concept consistent.

4.2.3 Language Support and Views for the Online Survey

One of the advantages of database usage is the support for multiple languages. The ASP front-end does not need to be redesigned for different languages due to the dynamic creation of the pages.

Each question and each answer (as long as they do not only contain numbers) already exists in multiple language versions in the database. The unique ID number for every question and answer combined with the related language identifier enables the database to automatically select the correct answer and the correct language at the same time. This is important because the setup of the pages (that the users will see when they participate) only contains dynamical elements, the headers, the question-text, and the possible answers are merely links to the appropriate database entry.

In order to ensure an adaptable method for multi-lingual support, the database system creates an index web page for each of the surveys. This page contains the rudimentary information about the questionnaire and can only be accessed by the designer of the survey. Furthermore, each of the automatically generated elements features an additional text field to enter the translated question. Instead of implementing directly in the database system, this interface allows the host of the survey to submit the translated version by completing the relevant fields.

During runtime, the ASP pages created can be regarded as limited views on the database. Only the corresponding information (match in language and survey) will be displayed. By doing so, consistency of displaying order is ensured. This signifies that the surveys in different languages are completely coherent and the answering values can be obtained (language independently) from the database.

This method has two main advantages: an individual page setup (language supporting) and the opportunity of a quick expansion. One should notice that as long as highly standardized questions are used in the survey (for example standard demographic values or mean opinion score answers), it is only necessary to link the new questions to already existing answers (which significantly reduces the workload). Also, questions from previous surveys can be reused by creating a new survey and adding the appropriate question IDs. By offering the index web pages, no further technical knowledge about the database design is required.

4.2.4 User selection

This section covers an introduction to the problematic field of sampling, including a description of the population and different sample approaches. Each of the sample approaches is introduced with advantages and disadvantages for the problematic field

of computer gaming. In contrast to this background knowledge, Section 4.5.2 describes the actual client selection for the surveys in-depth, including the age and gender distribution as well as an explanation as to which techniques have been applied.

To ensure a reliable statistical analysis, the underlying population needs to be represented accordingly. Generally in statistics, a population is the entire set of elements (in our case users), which is observed. Depending on the population, it is often not possible to include the whole population in the selection. For example, Germany has more than 80 million inhabitants; a survey including (most) of them would lead to substantial costs.

The typical approach to solve this problem is the selection of an appropriate sample. A sample is a representative subset of the entire population. Since the population is too large to enumerate all of the values in it, the sample can be understood as a subset with a manageable size. The selection of a fitting sample is a complex process, which can be defined as a trade-off between the effort of interviewing each subject within the sample and the aspect of creating a representative subset with only a limited number of elements. For example, the extrapolation for the next election is only representative if the sample represents the population as precisely as possible. Figure 4.6 illustrates the sampling; the sample itself should be a subset with a similar distribution.

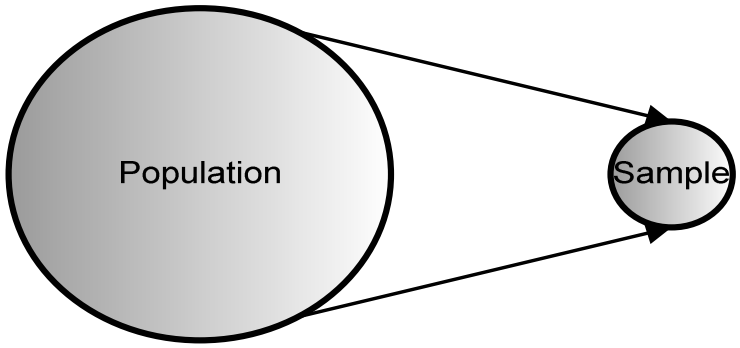


Figure 4.6 Illustration of the sampling process

Various techniques to create a sample exist:

Random sampling: This sampling method often uses larger sample sets without restrictions in participation. The subjects of the sample are chosen randomly. For example, this takes place during telephone interviews by using the telephone book or a random number generator. Depending on the exact method, the sample can be non-representative because its distribution differs too considerably from the distribution of the population. For example, by taking a sample in a football stadium, it is most likely that the distribution of the sample will contain more male subjects compared to the whole population. However, the sampling decision is always closely related to the statistical question. If the survey only includes the opinion of football fans then the stadium sampling approach will most likely contain a good sample of the overall football fan population. In contrast, the approach would fail for a survey that should forecast the upcoming election.

Quota sampling: In quota sampling, the population is first segmented into exclusive sub-groups. The decision as to whether or not a subject is member of the sample is based on the distribution of sub-groups in the sample. For example, the group of male subjects between 40 and 45 years of age with an income higher than EUR 40,000 in Germany is the equivalent of 1% of the population. A survey about the next election, which uses n=1000 participants and the quota sampling method therefore needs to include exactly 10 male subjects between 40 and 45 years of age with an income higher than EUR 40,000. Statistically, this method has the advantage of accurately selecting an equally distributed sample (compared to the population). The main disadvantage is the interviewer's influence, because in most cases the different subsets do not match equally during the interviewing process. This occurs because the quota sampling needs to match each sub-group precisely, which can be time-consuming for the smaller sub-groups. Therefore, the interviewer often needs to decide which subjects of the overrepresentedsub-groups need to be excluded (interviewer's influence).

Cluster sampling: The cluster sampling method aims to reduce the sampling effort by creating clusters for certain areas, timeframes, etc. It is an example of a "two-staged sampling" approach. In the first stage, a sample area (or timeframe) is chosen, afterwards the second stage includes sampling within the selected area. For example, within the forecasting survey for the next election the country is previously divided

into election districts. Afterwards, only a small number of subjects in each of the districts are used for the sample. This method can reduce travel expenses and time invested by the interviewer (due to clustering in the first step).

The general size of a sample depends on the survey's approach; [Higg 2001] provides a good introduction to the different sampling methods and the recommended size of samples. Generally, a good data collection involves:

- A structured and predefined sampling process
- For interviewer influence: note down the contextual events
- Keeping the data in time order (refer to Section 4.2.5)
- Recording and cleaning of non-responses as well as incomplete responses (refer to Section 4.2.5)

This section will discuss the first two elements for a good data collection: a structured sampling process and the interviewer influence/contextual events.

Before regarding the sampling methodology for the survey, it is necessary to understand the population of computer gamer. The examples regarded above differ from the population of gamers in three significant ways:

a) The population of gamers is dynamic

b) The sub-groups (clusters) are not selective

c) A regional clustering of computer gamers is not possible

The population of gamers itself is highly dynamic; the currently strong market growth reflects that even people with no previous gaming experience are becoming attracted to computer games. This growth leads to a shift in the consistency of the population. A quota sampling approach would require using a sample that represents the consistency of the population precisely. The increasing social acceptance leads to a significant shift in the consistency (i.e. more females and older people are starting to play computer games). Therefore, both clustering and quota techniques fail to represent the population exactly.

Furthermore, sub-groups of game types are not selective. The prediction of results for the next election now includes selective attributes like age and income. The

objective of the distribution of the online gaming survey is to split the population up into game type sub-groups. These groups, however, are not selective as Figure 4.7 shows. In this example, a sub-group of players who prefer to play both RTS and FPS games exists.

The gaming industry also promotes this trend, since fiercer competition between the game producers is leading to a more specialized game design. Hybrid games not purely classified as one of the game types are created.

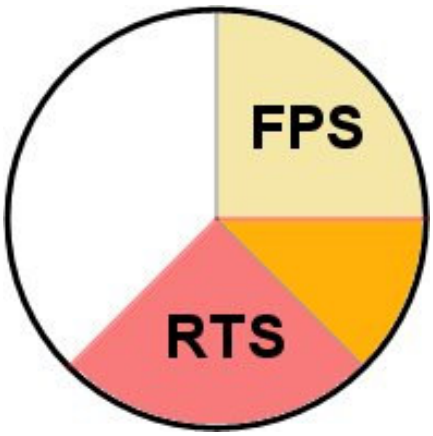


Figure 4.7. The population of computer gamers divided into sub-groups by game type.

A regional clustering of the computer gamers would also be impossible because network games provide international servers. Therefore, the local place of residence is still an important variable for the statistical analysis. However, in the case of computer gamers, it cannot provide an accurate clustering.

4.2.5 Data Cleaning Mechanisms for Online Surveys

By using an open participation method for the online survey, the major disadvantage is the high potential of multiple participations by a single user. Although the target audience is as large as possible (due to no regulations like accounts, cookies, IP-catching), it is still possible to reduce the negative effects of multiple participations from a single user by creating an algorithmic solution after collecting the data.

The idea of an “post-algorithmic solution” is the reduction of negative effects without limiting the potential answering group too much. It can be regarded as a

trade-off between the radical solutions of either neglecting all negative effects or creating a method to completely prevent multiple participations by means of security mechanisms.

Before analyzing the procedure, one should first look at the main negative effects of multiple participations. These effects are illustrated in Table 4.2 with basic examples. Two of the three main negative effects (repeatedly submitting the same answers and submitting unrealistic combinations) can be detected by using an algorithm that compares the gathered data.

Table 4.2. Possible effects of multiple participations by the same user

Effect	Description and Result	Example
Repeatedly submitting the same answer multiple times	The same answer is submitted multiple times, increases the relative number compared to the rest of the data pool	Consequently submitting profiles with a young age in order to lower the average age score
Submitting unrealistic combination of answering possibilities	Creating unrealistic or fake answering combinations in order to make the correlation incorrect	Submitting profiles with an age of 15 years and a martial status of “widowed”
Submitting random answers	Randomly selected answers are submitted multiple times with no underlying scheme	Using an automated program or randomly clicking and submitting multiple times; creates white noise

The technical implementation of the procedure works on tables. Since results in the database can be viewed as a single table with all data sets, this table can be stored for further analysis. Any spreadsheet program (like Excel) can host the table. The algorithm for the data cleaning comprises two different macros, one for each problem described (repeatedly submitting the same answers and submitting unrealistic combinations).

Data can be analyzed once the survey has been completed. The effect of repeated submissions of the same answer can be analyzed by comparing locally related submissions in the database. If multiple entries within a constant range (x data sets in front and x data sets after the current entry) have a high similarity (90%+ identical values), then the probability of multiple participations is high. The duplicated data sets can be eliminated by reducing the remaining data pool and clearing the negative effects. This data cleaning is implemented technically by means of a macro in Excel that compares the 50 adjacent data sets before and after the current data set (each value of two data sets is compared). If their similiarity is too high (like 95% similar answers), the duplicate is eliminated. The order of data sets equals the submission order. If a single user submits an answer multiple times within a short time period, then these submissions will be located close to each other in the table.

Another problem, the submission of unrealistic combinations, can also be identified by using a simple algorithmic prevention measure. The data sets can be tested and the algorithm can decide whether or not inconsistencies occur. For instance, since the minimum age for marriage is 16 or 18 years (depending on the country), consistency of the data pool can be improved by eliminating every data set with a young age and a non-matching martial status. This part of the algorithm cannot be applied to every survey. Instead, it needs to be designed individually because the definition of an "unrealistic combination" depends on the set of questions. For the two surveys in this thesis, elimination focuses on the demographic data. This especially includes the removal of extremely old participations or further unrealistic combinations (like 14 years of age and divorced). Generally, the same technique can be used for the other questions as well, but for the surveys in this thesis, focus relies on a large set of answers. The stricter the method to eliminate data sets is, the more likely one eliminates correct data sets.

The only problem which cannot be identified is the submission of random answers. These negative effects will remain because it is not possible to define which legal (consistent) data sets have been submitted by a user and which data sets have been created randomly. One could try to search for patterns in the data pool, however, the probability of eliminating valuable real answers increases significantly.

4.2.6 Regression Analysis as a Statistical Method

One aspect in the statistical analysis is the creation of an underlying model for the players. This is especially interesting for the distribution of the online player behaviour analysis because a model would indicate factors that hardcore gamers have in common. Generally, the statistical analysis for this example aims to understand how one can decide if a player can be regarded as "hardcore" (see Section 4.3). Furthermore, the influence of demographic factors on gaming behaviour is evaluated. Therefore, it is necessary to analyze the influence of each given independent variable. With regard to the total time invested in gaming, the related model should only include the most statistically significant variables.

The regression analysis starts by testing the influence of demographic and personalized user information with regard to his/her total game time. Some factors (like occupation or gender) might have a more significant impact on the total gaming time than others. By eliminating irrelevant influencing variables, the model increases in the coefficient of determination and thus becomes statistically more significant. Finally, the remaining influencing factors indicate the characteristics of a typical "hardcore" gamer (regarding the example of game time analysis; however, a regression analysis can also be conducted for other problematic fields like virtual fragmentation or chatting behaviour).

4.2.7 Hypotheses

In general, a statistical hypothesis test is a method of making a statistical decision based on experimental data. A null-hypothesis testing answers the question of "*how well the findings fit the possibility that chance factors alone might be responsible*" [Cram 2004]. De facto this means: if findings for the influencing factors are significant enough, the hypothesis is rejected. For example, in a pregnancy test the hormone level is measured. Generally, the null-hypothesis is expected, which states that the subject is not pregnant. However, if the hormone level rises above a certain value x, then the high value of this influencing factor alone implies that the null-hypothesis cannot hold true. Therefore, it must be rejected based on the statistical findings. As a result the opposite turns out to be true, which means that the subject is most likely pregnant. The hypothesis tests are usually applied with a certain degree

of confidence, which is generally either 95% or 99%. The correct result of the example would be: with a 95% (99%) chance the null-hypothesis can be rejected, which implies that the subject is most likely pregnant.

Several preparations for the observed data exist to structure the hypothesis test correctly:

(1) The null-hypothesis must be stated in mathematical/statistical terms that make it possible to calculate the probability of possible samples assuming the hypothesis is correct.

(2) One needs to design a test statistic that summarizes the hypothesis-relevant information in the sample (factors that influence the hypothesis). Subsequently, the distribution of the test statistic can be used to calculate the probability sets of possible values.

(3) Furthermore, critical values need to be defined, effectively a confidence interval around the target value (0 in case of a null-hypothesis) is created, if the value of the test lies within this interval, then the result is not significant enough to reject the null-hypothesis.

4.3 Questionnaire I: Distribution of Online Player Behaviour

This section features an introduction to the background of the survey "distribution of online player behaviour" and hypotheses are subsequently described. These hypotheses are evaluated in the experimental result section of this thesis (7.1).

4.3.1 Background

One of the main aspects addressed with regard to player behaviour is the strong influence of in-game content for real world behaviour. The most important aspect in this approach is the time each player invests in the game. Generally, the more time a player spends in the persistent online environment, the easier he/she will adopt given concepts. In order to understand how much dedication towards gaming an average player shows, one must look closely at the social deterministic. Another aspect in this study is the analysis of the heavy user ratio, the so-called hardcore gamers.

Hardcore is defined as being: (i) extremely explicit, (ii) intensely loyal and (iii) stubbornly resistant to change or improvement [Ency]. In the matter of gaming the term hardcore refers to option iii), being stubbornly persistent in playing the same game with far above average interest. This definition goes along with the previous usage of hardcore as "heavy usage"; hardcore players tend to play much more intensively and longer than normal users. Casual playing, on the other hand, is the opposite: approaching the game with average interest (which includes lower time invested compared to the hardcore gamers).

To obtain the most accurate results possible, one must look at each of the game types separately. Therefore, the survey [Frit 2006] includes the four different game types: RTS, FPS, RPG and SG – each with a general and personalized set of questions. Both sets were reduced to the most important questions. Long surveys often show a lower reliability due to the overproportional decrease in user concentration. Thus the number of questions did not exceed 20. The main goal of the survey is to yield reliable results for further evaluations as well as to analyze the real behaviour of the current online gaming community.

The general set of questions in the survey contains demographic information about the user such as age, gender, marital status, educational level and nationality. It also features questions about the amount of leisure time, the gaming experience and the overall game time (per week/day). This set is highly user-related and is utilized to build up correlations with game-related questions.

The personalized questions, however, are more closely related to the game type. Not every game type features the same type of challenges as shown above. It contains specialized questions about in-game activities and daily scheduling, i.e. "Would you call in sick from work so you can play more?"

4.3.2 Hypotheses for Distribution of Online Behaviour

This section contains hypotheses for the "distribution of online behaviour" survey and an explanation for each of them.

It is most likely that players from different game types will also have a different gaming behaviour. One of the major aspects is the average time per week that a

player spends on computer games. A significant difference between the different game types is expected. In order to statistically analyze this statement, a null-hypothesis is created which states the exact opposite (no difference between the game type is expected). The estimated finding is reassigned to the alternative. During the statistical analysis this null-hypothesis will be evaluated. If the initial estimation holds true (a difference exists), then the null-hypothesis can be rejected and the alternative is true.

(1) **$H0_1$: No difference in the average game time per week between the game types exists**.

This initial hypothesis suggests the further evaluation of the game types. Since the age obviously influences the decision of the game types (for example MMORPGs with their monthly costs are less interesting for younger players), it might also have an impact on the overall game time. Therefore, one can suggest that demographic factors (such as age, leisure time, gender) also influence the game time. A simple correlation between these two factors would be obvious because students clearly have more leisure time compared to adults with a job. To harmonize these numbers it is important to compare the relative game time (which is total game time per day divided by the available leisure time per day). The resulting number ranges between 0 and 1, where 0 indicates that the individual spends no leisure time at all on gaming, whereas 1 indicates that 100% of the leisure time is spent on computer gaming. As is the case for hypothesis 1, the estimation is allocated to the alternative and the null-hypothesis states the contrary.

(2) $H0_2$: The relative gaming time (game time divided by available leisure time) is not influenced by demographic factors or the game type.

Finally, the behaviour of hardcore players is very interesting. Since this group takes the games far more seriously, one can suggest that their attitude towards real life events will be similar as well. If this holds true, then the overall gaming time (not relative game time) should be proportionate to the number of positive answers regarding the player behaviour questions. The player behaviour questions each indicate a motivation to focus on the virtual world instead of the real world. If this correlation holds true, then players with a higher overall game time show a common attitude towards real world events.

(3) $H0_3$: The overall gaming time (hours per week invested in gaming) is not influenced by the players' attitude towards real-life events (family, real world events, job).

These hypotheses are evaluated in Section 7.1 together with further experimental results; they will serve as a guideline for the experimental evaluation.

4.4 Questionnaire II: Virtual Fragmentation

This section features an introduction to the background of the survey "virtual fragmentation" and subsequently, hypotheses are described. These hypotheses are evaluated in the experimental result section of this thesis (7.2).

4.4.1 Background

The complexity of social interaction between players grows proportionally with the complexity of the virtual environment. Hence, one of the important related topics is the difference between in-game and out-of-game behaviour of each player. The underlying effect of virtual fragmentation is defined as the discrepancy between real world behaviour and virtual world behaviour. The greater the difference is, the larger the virtual fragmentation. In order to measure such a difference one must look at the real world and in-game behaviour separately, and then address the reasons for potential differences or specific game behaviour adoption.

One part of analyzing differences between in-game and real world behaviour is the selection of a model. Human behaviour, even in terms of gaming, can be very complex. Several models have a high degree of complexity and although they are reliable and valid, the gaming behaviour might not have been measured most accurately with them. Therefore, the five-factor model [Digm 1990] and [Ewen 1998], which is both reliable and simple enough to perfectly adjust it to gaming, was selected.

The five-factor model divides the human character into five personality factors. Each of the factors defines a part of the character and thus the behaviour. Based on the model of Costa and McCrae [McCr 1996] this approach divides the personality into neuroticism, extraversion, openness to experience, agreeableness and

conscientiousness. Each of the factors can be considered. Together with demographic values it is possible to statistically analyze the main reasons for virtual fragmentation. Figure 4.8 illustrates the main idea of the virtual fragmentation survey with regard to the five-factor model.

However, gaming communities also evolve along the games. The interaction relocates to other communication measures like forums or chats. It is difficult to take those interactions into account, although they are obviously game-related. Thus the narrower definition of gaming behaviour refers to the in-game interaction between two or more players.

Computer games feature typical groups of players: the taxonomy of Bartle [Bart] divides them into achiever, explorer, socialiser and killer. Each of them shows their individual online behaviour and thus each player type has an appropriate characteristic. In order to understand the virtual fragmentation [Frit 2007.2] it is necessary to further combine the taxonomy of Bartle with the five-factor model. The approach aims to understand whether the different player types have an influence on the level of virtual fragmentation. The results of both surveys are analyzed in chapter 7.

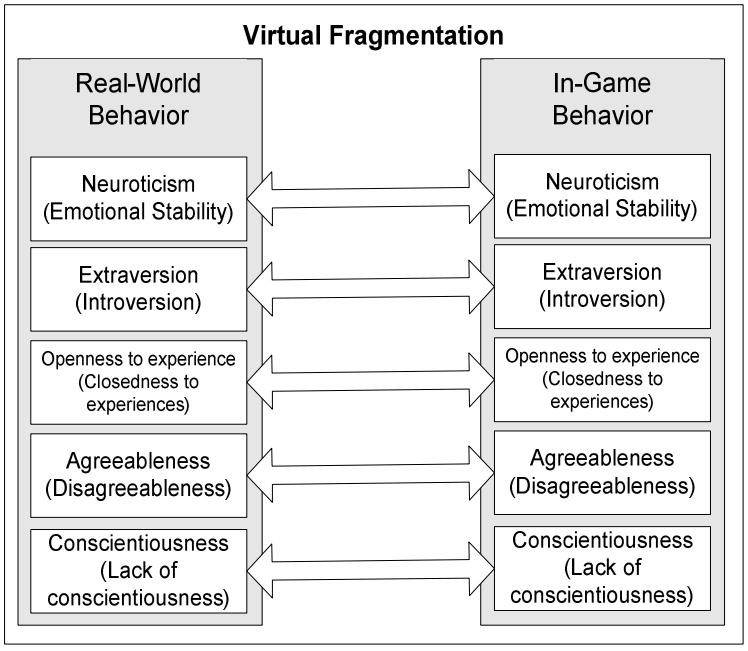


Figure 4.8. Illustration of the idea behind the virtual fragmentation survey. The differences between real world and in-game behaviour with regard to each of the five personality factors are shown. The arrows indicate which attributes will be compared to identify a potential effect of virtual fragmentation

4.4.2 Hypotheses for Virtual Fragmentation

This section contains hypotheses for the "virtual fragmentation" survey and an explanation for each of them. In contrast to the hypotheses from the survey "distribution of gaming behaviour", the hypotheses in this section have interdependencies. The major goal of the survey is to pinpoint the gap between real world and virtual world behaviour. Hence if no gap can be shown, a further analysis of possible influencing factors would be meaningless.

The virtual fragmentation survey aims to determine a difference between the real world and the virtual world behaviour. In order to achieve this, one needs to show that the average value for virtual behaviour differs significantly from the average value for real world behaviour. Each of the behavioural traits is measured with a Lickert-type scale from 1-5, referenced as Mean Opinion Score (MOS). The statistical evaluation is analogous to the evaluation in Section 4.3.2, the estimated statement is allocated to the alternative, whereas a null-hypothesis is created stating the contrary.

(4) $H0_4$: No difference between the average MOS values for real world behaviour and the average MOS values for virtual behaviour exists.

If this hypothesis can be rejected, a further aspect needs to be evaluated. In case of a rejection, a gap between real world and virtual behaviour exists although the influencing factors for the gap are unknown. In order to analyze possible influencing factors, a general null-hypothesis for each of them is required. It is most likely that the demographic factors (age, gender, educational level) will have a high influence on the level of virtual fragmentation. However, further aspects like the game type can be possible causes of an influence as well. To substantiate this influence, a null-hypothesis will be evaluated because if it can be rejected, then a statistically significant influence of the factor (age, gender, educational level or game type) on the level of virtual fragmentation can be shown.

(5) $H0_5$: Further factors (such as age, gender, educational level and game type) have no significant influence on the level of virtual fragmentation (size of the gap between virtual and real world behaviour).

These hypotheses are evaluated in Section 7.2 together with further experimental results; they will serve as a guideline for the experimental evaluation.

4.5 Methodology and Advertisement

This section covers the main aspects of actual sampling for both approaches (distribution of online gaming behaviour and virtual fragmentation). The sampling section explains the user selection mechanism; the methodology section contains information about the number of participants and discusses the reliability of the samples.

4.5.1 Sampling

The sampling for both approaches uses the random sampling technique for various reasons. First of all, the quota sampling method would not be appropriate for the population of the survey, because the population of computer gamers is still changing. In contrast to surveys that use data that has been available for a long time (like election predictions or price indexes) the field of computer gaming research is relatively new. Therefore underlying data (to match with) is missing. This also influences the clustering technique, because no reliable data regarding the local distribution of computer gamers exist, since most of the games do not keep track of their players' real world location.

Another important reason for making use of the random sampling method is the effort vs. benefit. The selection of individual users for the sample is comparably low-cost in the random sampling method, because especially with larger sample sizes (n=1000+), the quota and cluster sampling approach both require a large preparation effort. To illustrate this disadvantage: the quota sampling would require an exact representation of the population in the sample. To ensure this, standardized offline interviews would be needed (because the interviewer needs to ensure the matching of factors like age, education and gender).

A large sample is also more resistant against typical problems of Internet surveys. A single user can therefore not influence the outcome of the survey (due to multiple participation or submission of random numbers) as much as he/she could in a smaller survey. Furthermore, the probability of creating a representative sample without bias due to random sampling increases with the number of participants. The chance of having a biased sample increases with a decreasing number of participants. For example, a sample size of 25 users can be biased because 10 of the users are friends with similar gaming behaviour.

The Internet survey method also promotes the random sampling method, since one of the major aspects in the creation of the surveys was the size of the sample. Both surveys aim to receive as many answers as possible. With an overall number of more than 30,000 users (in both surveys together) the sample achieved the goal of having a very large data set to work with. Typically, a different sample size leads to a different accuracy in measurement. Generally with all else being equal, a larger sample size n will lead to an increase in accurate estimates of various properties of the population. As a rule of the thumb, a sample size of n=100 already provides reliable results. Compared to an overall sample size of 7,100 cleaned data sets for the survey "distribution of online gaming behaviour" and 5,800 cleaned data sets for the survey "virtual fragmentation", the underlying samples provide an excellent example of the gamers' population.

The rule of the large number underscores that the higher the sample size, the more likely that the population will be represented accurately. Furthermore, the statistical central limit theorem states that the sum of a large number of independent and identically distributed random variables is approximately normally distributed (which is the case in both surveys because the participating users neither have an influence on each other nor do they vary in their distribution). To ensure this distribution, the Jaque-Bera test was used on both the original sample and the cleaned sample; both samples were distributed normally.

The target group for the evaluation in both surveys (virtual fragmentation and distribution of online game behaviour) is the community of online players who play competitive online games on either PCs or consoles. This population is the focus of the survey evaluation for two reasons.

First of all, the overall Internet gaming community also features a wide variety of minor games, such as browser games, text games or freeware. Since those players show a very distinct gaming behaviour, the focus for this thesis relies on current top-selling online game genres, which are RTS, RPG, FPS and SG. Afterwards, findings are compared to other surveys like the casual player evaluation in [Case 2007]. The remaining sub-sections (like casual players) also have a significant market share, however, the amount of casual games exceeds the commercial games from game genres observed in this thesis by far. Their production effort, their average game time and their graphics are lower/less detailed compared to the top-selling online games.

The second important aspect is the interaction within these games. The remaining game types like Beat-em-up, Puzzles, Edutainment, Casual games and asynchronous games seldom feature any player interaction besides the game content. This signifies no predominant chatting or in-game voice. Some of them are mostly single player oriented. Thus effects like virtual fragmentation will most likely not occur, because the players have little to no interaction options besides the game content. A detailed comparison of the demographic factors between the surveys in [Thee 2007], [Casu 2006] and [Fritsch 2007] is featured in Section 7.1.

4.5.2 Methodology

The approach to reach a large sample size included general decisions for both surveys:

- The survey method chosen is the online questionnaire
- The distribution of game time survey aims to understand influencing factors for a high computer gaming time. Furthermore, it should provide general data about the overall game time of current online players
- The virtual fragmentation surveys aims to pinpoint differences between real world behaviour (modelled with the five factor model) and virtual behaviour. Also the influencing factors of demographic factors need to be analyzed
- Demographic factors for both surveys are age, gender, location, leisure time and education/employment
- Analyzed game types for the surveys are FPS, RTS, RPG and SG

As discussed above, the online questionnaire and the decision to receive as many answers as possible only work well with the random sampling method. In the first phase, the ASP webpage was prepared as well as a standardized advertising text. In order to ensure an international character of the survey, the advertising text was translated into German, English, French and Spain and used in accordance with the local forums. Afterwards, target forums were selected: as a guideline the forums of the top 10 selling games of FPS, RTS, RPG and SG as well as the relevant inofficial fan sites are used. For a closer relation with the player community, many community forums (clan, guild, and server) were selected. In addition, general game-related community sides were selected, which provide a player base for a certain genre. These can be regarded as accumulations of players with a preference for one of the game types. Forums were selected carefully, because distribution can influence the distribution of the sample. However, the overall number of forums selected was over 250, again this decision clearly promotes the motivation for a very large sample size (and thus reliable data).

In contrast to the surveys of [Park 2006] and [Casu 2007], the evaluation of the online surveys in this thesis does not focus on the entire population of all players in all genres. As indicated in [Park 2006], the US gaming market is segmented. A relatively large amount of players from this survey are occasional gamers or hobby gamers. These players, however, are explicitly not the target group of online surveys in this thesis because in most cases they do not play FPS, RTS, RPG or SGs.

As [Casu 2007] indicates, the population of casual gamers shows an intercommunity regarding demographic factors and the players' motivation to frequently change their games. The overall time invested in games, the background knowledge and the understanding of game mechanisms is relatively low, the main focus of this group of players is pure entertainment. As part of the analysis, demographic data was compared between online gamers and casual gamers.

In phase two, the advertisement was posted on the more than 250 different gaming forums in one night, so the starting date for each forum was practically identical. The data collection period was two weeks (14 days), in order to also receive information from players who do not visit the forum on a daily base.

The rate of return shows a different behaviour between the top-selling game pages and online communities. Generally, the rate of participants from the online community is far higher compared to the top-selling games. One recognizable exception to this trend exists, a majority of nearly 7,000 answers overall is based on the World of Warcraft forum. A large amount of players from that forum demonstrate interest in the survey. This fact is one substantiation for the high amount of RPG players who answered the questionnaire.

For the "distribution of online game behaviour" survey: after eliminating the incomplete data sets, the total number of answers equals 7,100 records, which serve as the data pool. However, the number of answers for each of the game types varies. It was determined that the set contains more than 4,500 users from the MMORPG scene, about 1,300 users from the FPS sector, 1,100 RTS users and only 197 SG players (compare to Table 4.3).

The distribution of online gaming behaviour questionnaires contained 20 questions; the virtual fragmentation questionnaire contained 38 questions (the higher number of questions was necessary to provide enough individual questions for each of the five factors). Table 4.3 gives an overview of the number of participants in each of the surveys as well as the number of remaining cleaned data sets. Both surveys had a similar number of participants before the data cleaning. The number of remaining data sets is smaller for virtual fragmentation. One reason for that is the higher number of questions, only complete answers were included in the cleaned data set.

Table 4.3. Overview of the number of participants and the number of cleaned data sets in both surveys

Survey	Number of data sets			Number of cleaned data sets		
Distribution of online gaming behaviour	16101	2899 FPS	18.0%	7102	1297 FPS	18.3%
		2525 RTS	15.7%		1101 RTS	15.5%
		10225 RPG	63.5%		4501 RPG	63.4%
		452 SG	2.8%		197 SG	2.8%
Virtual	14172	2990 FPS	21.1%	5795	1292 FPS	22.3%

Fragmentation		1942 RTS	13.7%		817 RTS	14.1%
		8673 RPG	61.2%		3501 RPG	60.4%
		567 SG	4.0%		185 SG	3.2%

Distribution in Table 4.3 shows one of the disadvantages of the technique of online surveys. Although the sport games section was equally activated with a similar number of forums (game-related and top 10 selling game forums), the reply rate is significantly lower compared to the average response rate. In contrast to that, the rate of return is very high for the RPG sector. An analysis of the different forums substantiates possible influencing factors; the average number of new posts per day is lower in the sport games forum and much higher in the RPG forums. This means that the players' general activity from the genre is much higher/lower compared to others. For example: the RPG forums are very active, making it more likely that players see and respond to the advertisement.

In order to prevent this unequal rate of return for future surveys, it is necessary to harmonize the findings. A harmonization factor therefore needs to be created based on the number of overall players in the genre. With such a factor the overall influence of bigger sub-groups would be reduced, whereas influence for small sub-groups (such as the SG sector) would be larger.

Another option is to pre-select participants. The approach in this thesis contains active sampling, which means that participants are contacted for the first time. One disadvantage of this method is that the participants answering are most likely extremely interested/involved in the topic.

Passive sampling on the other hand indicates that users participate in the survey who have already participated in previous surveys. Usually they reveal a more indifferent behaviour to specific questions, whereas their overall involvement is high. According to [ADM 2001] a mixture of active and passive sampling generally leads to the best sample. Therefore for future surveys, the database will be used to store the participants' e-mail addresses pending their consent.

In order to ensure that the cleaning algorithm does not significantly change the distribution of the sample, the following aspects were compared between the original data sets and the cleaned data sets: (a) distribution of age, gender, leisure time, (b)

quantity of game type subsets (FPS, RTS, RPG, SG) and (c) normal distribution. This analysis underpinned that the cleaning neither changed the demographic values nor the relative quantity of game types. Both samples were also distributed normally.

The gender distribution of data collected is similar to other online gaming surveys. A large majority of participants are male. Table 4.4 gives an overview of the rate of males in the sub-groups. As one can see, the gender distribution is not significantly influenced by the data cleaning algorithm. Overall, both surveys have a large majority of over 90% (95% for online gaming behaviour) male subjects within the sample. The different game types have a minor influence on gender distribution; female players tend to prefer RPG games over FPS and RTS games. The values in Table 4.4 are percentage values of the target sub-group (percent of male players compared to the overall population of the sub-group).

Table 4.4. Gender distribution in the online surveys

Survey	Male rate before cleanup		Male rate after cleanup	
Distribution of online gaming behaviour	**Overall**	**95.8%**	**Overall**	**96.0%**
	FPS	97.9%	FPS	97.8%
	RTS	98.9%	RTS	99.2%
	RPG	92.3%	RPG	92.3%
	SG	94.5%	SG	95.3%
Virtual Fragmen-tation	**Overall**	**91.2%**	**Overall**	**91.0%**
	FPS	93.1%	FPS	92.7%
	RTS	95.9%	RTS	96.4%
	RPG	89.9%	RPG	90.1%
	SG	89.5%	SG	89.6%

Another interesting aspect is the influence of the survey language. The advertising strategy included forum entries in all of the featured languages: Spanish, English, French and German. The majority of replies in both surveys were in English. Overall 60.2% of the data sets were originally collected in English, 25.5% in German, 7.8%

in French and 6.5% in Spanish. None of the languages has shown a significant influence on the demographic values of the surveys (age, gender, educational level).

4.6 Summary

Both social factors, the importance of player behaviour as a main influence for the game design and the acceptance of gaming in the culture, have a significant importance for the approaches. Without understanding the player's motivation, one cannot efficiently improve current gaming problems. Hence this chapter introduced different ways to define game challenges for a player to help analyze the motivation for strong game addiction. Afterwards, general guidelines for the design of surveys are included, which help to make the online questionnaire reliable. Furthermore, two main approaches have been introduced to measure the distribution of online player behaviour and the difference between in-game and real world player characteristics. The results of both approaches are discussed in chapter 7.

As a side note, one should keep in mind that the technical details for a database setup and a semi-automated data selection are implemented in order to reuse these methods for further surveys. Especially with regard to the scalability and performance of the game-related surveys, the database approach introduced promotes further large-scaling online questionnaires.

5. Approach II: Next Generation Mobile Gaming

> "Home computers are being called upon to perform new functions, including the consumption of the homework formally eaten by the dog", Doug Larson.

The emphasis of this chapter relies on the gaming aspect of mobility. Hence it contains a brief introduction to the motivation of the mobile gaming field as well as a short summary of the important network aspects. The technical details include two main aspects: the usage of J2ME technology for standardization and the integration of instant messenger communication in a mobile environment. Afterwards two approaches towards an innovative gaming concept are described in detail. The first includes a general improvement of player communication and the second features the design of a mobile lobby platform. Furthermore, the importance of communication in the mobile environment is underscored by a survey.

5.1 Motivation and Overview

The mobile game scenario presents its own set of problems for real-time applications. State-of-the-art today is that GPRS and UMTS would provide a stable connection to a server, supporting login functions and cheating protection. The natures of the 3rd generation protocols (cdma2000/UMTS) clearly promote a S/C over a P2P structure. However, depending on the country, prices for a MB of data on mobile devices vary up to 10 dollars. Thus most of the end users cannot afford to play under those pricing circumstances.

The other option is to create local subnets with WLAN or Bluetooth. Although possibilities exist to promote one of the nodes to be a local server and create small server-client architectures, experience shows that the risk of a network split and the disadvantage of having a single bottleneck obviously do not promote this solution. Instead, pure P2P or hybrid systems offer the advantage of having a flexible data distribution and a stable network. The lack of cheating protection appears to be the greatest disadvantage.

In order to create a mobile solution that fits multiple games and combines next generation technology with the needs of a gaming community, one must keep several aspects in mind: the current network structure certainly has its limitations when it comes to latency-sensitive applications like FPS or RTS. Hence the design must use techniques like WLAN or Bluetooth to support these game types. Another important aspect is the constantly growing number of simultaneous users of these applications, especially with a focus on current evolution in mobile MMOGs.

5.1.1 Mobile Communication for Games

One of the most important aspects in the mobile game design is the usage of communication. One must differentiate between handheld devices and their capabilities; the mobile gaming market has a wide variety of different handhelds. Older devices (like the Game-Boy) only support fixed link connections to other handhelds, which require additional cables. Newer game oriented devices (like the Nintendo DS or Sony PSP) use WLAN as the wireless communication method. Mobile phones on the other hand are based on UTMS/3G network technology in order to communicate with others. From a gaming perspective, this network can also be used for gaming applications, although disadvantages like high latency occur. Before developing an application for the mobile sector one must look at the current communication between players within the Internet.

Besides pure in-game chatting, a large part of communication is done by instant messengers. These applications do not only offer the pure service of exchanging messages over a network. In general, they allow the user to communicate with others in real-time by using a form of text-based messages. This communication can also include the transfer of data files. The clear contrast to the classical e-mail communication relies on the fact that the messages are being sent and received at the same time (besides delay, network loss, etc).

Another important aspect of instant messengers is the option to register other users as buddies and build a list of friends. Each contact has its own chat history and status. Therefore, the user can see who is online and available and even (depending on the protocol) leave messages for offline contacts. Current clients are even capable of

transferring files with peer-to-peer sharing; furthermore, audio and video chat options are also available.

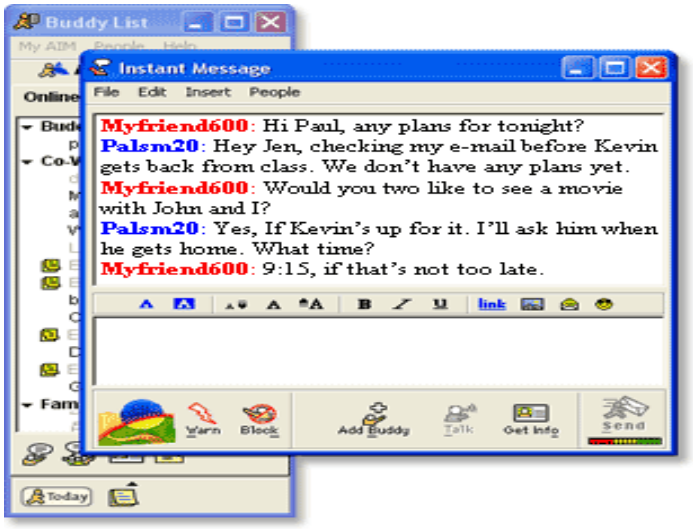


Figure 5.1. Current AOL IM version: Left window shows the buddy list, right window is the ongoing instant message chat

Admittedly not all of the features mentioned are necessary to improve the chat situation in games. A voice and video communication would increase the options in a player group and even give them an advantage for the game content. Although such a feature should be optional for now, not every player has a microphone. As an example, the current gaming oriented devices (Nintendo DS and Sony PSP) do not have any support for voice communication yet. On the other hand, mobile phones have the necessary hardware, but the bandwidth of a multi-user video chat can easily exceed the capacity of a current UMTS connection. Moreover, displays of most next-generation handhelds lack the size to depict an additional video window (this problem remains for every current mobile handheld, since most of the displays do not exceed 400x300 pixels).

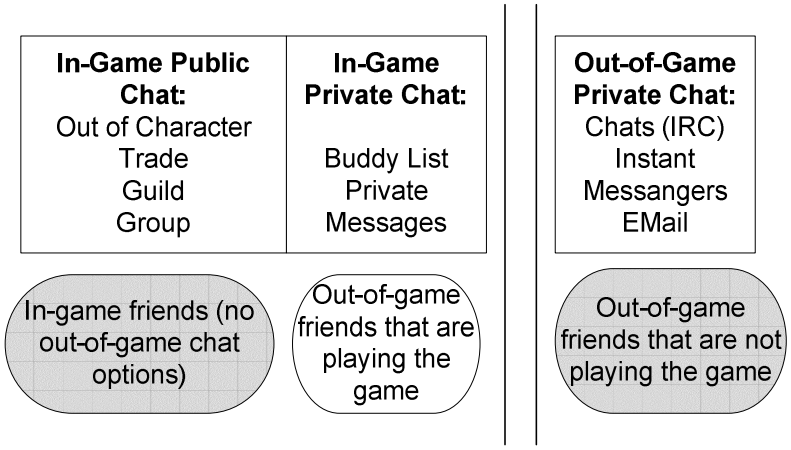


Figure 5.2. Illustration of the current chat situation in MMOGs

While playing online games, a single user has several options to communicate with others. Generally, they are divided into "in-game" and "out-of-game" chats. While the in-game chat uses protocols of the game itself, the player is limited to communicate only with other players who are currently playing the same game. The out-of-game communication (like instant messengers) relies on different protocols, depending on the messenger used.

The player's motivation to communicate with others while playing the game strictly depends on the content and speed of the game. A fast-paced game setup like an FPS death-match in Quake leaves less to no room for further chats. In contrast, many other MMORPGs feature a wide online world that also requires traveling. During this time many players want to communicate with others. This need also includes out-of-game communication, which cannot be done by the in-game protocol itself. Therefore, players frequently have an additional IM client (third party software) running in order to communicate with other out-of-game contacts.

As one can see in Figure 5.2, the current problem relies on the fact that players cannot chat with in-game friends when they are not running the game client (except if they have added them separately to their e-mail/IM buddy list). Furthermore, while playing in full-screen mode, offline real-life friends and those who are not playing the game simultaneously are not available either. The general idea is to merge both in-game and out-of-game contact lists in order to give the player the opportunity to be connected in the most flexible way when playing even while using only his/her

instant messenger. By integrating features of the instant messenger into current (mobile) games, the game chat would be expanded substantially (further details in section 5.3).

Before taking a closer look at the possibilities to merge both technologies, one must first consider the underlying online games and their communication features. The significant difference between Internet MMOGs and mobile MMOGs is the performance of the network technologies. Today, there are already several solutions for persistent online environments on the Internet, however, the mobile gaming scene still lacks those solutions. Existing mobile MMORPGs do not really offer a persistent online world, because by using GRPS and UMTS as the protocols, the resulting network latency completely prohibits any real-time game elements [Fant], [AOF].

Since a mobile MMOG is still not available to integrate the instant messengers, the target MMOG must be taken from the Internet sector. This is how a cooperation (between the Freie Universität Berlin and CCP Gaming) was forged with Eve Online; main focus of the integration relies on the aspect of scalability. Eve Online is a so-called second generation UMMORPG. Moreover, the game itself reveals similarities to the traditional MMORPGs, nevertheless, there are important differences. As an UMMORPG, the game itself does not feature different shards and therefore does not split the game world up into several realms running in parallel. The result is a single realm with up to 25,000 users playing simultaneously. Consequently, the chat volume exceeds the already high number of other MMORPGs by far, which further suitably addresses the scalability problem.

The in-game client offers separate individual group, clan, and private channels as well as public OOC (out of character) and trade channels. Just like in other second generation MMORPGs, one can have a list of in-game buddies to directly communicate with (private chat). The buddy list does not include real-life contacts and it also does not offer the opportunity to leave a message for an offline player.

5.1.2 Analysis of the User Group for Mobile Applications

In order to contribute a new approach to the mobile gaming situation, one must first evaluate the current problems by analyzing the players' specific needs. The study on mobile gaming refers to the basic user survey of [Frit 2006.4], which shows an ASP

online questionnaire with a set of 20 questions. The questionnaire underscored current mobile gaming preferences: a quick and easy game setup, an average game time of less than 15 minutes and high player fluctuation. The main findings are summarized as follows:

- The typical user group of mobile phone games is a young peer group from in the 12 - 20 year age bracket with a high interest in games, high expectations and a low budget.
- There is a clear correlation between age and game time as well as age and interest in games. Generally, younger people show an affinity to computer games.
- The average usage time of mobile phone games is much less than 15 consecutive minutes. Longer game sessions are rare exceptions.
- A correlation between age and game time spent on mobile phone games could not be ascertained.

These findings influence both of the given research approaches in this section. On the one hand, insights into user behaviour are used to create the mobile gaming lobby. This attempt results in the fact that a majority of users prefers a fast game setup and the average game sessions do not exceed 15 minutes in most cases. The gaming lobby and its implementation are described in detail in section 5.3.

On the other hand, the aspect of communication in a mobile environment also has an influence. In fact, most of the players rate the current mobile games as rather un-attractive due to the poor quality of graphics and the strictly limited game play. One option to design a gaming-related application for mobile devices is the usage of instant messengers [Frit 2006.4]. With these messengers, players can stay in contact with in-game friends from PC games without the need to run the game client. Therefore, the second approach aims to integrate current instant messenger technologies into persistent online environments. Furthermore, integration should allow mobile devices to run a chat client that enables communication with in-game friends from the PC games.

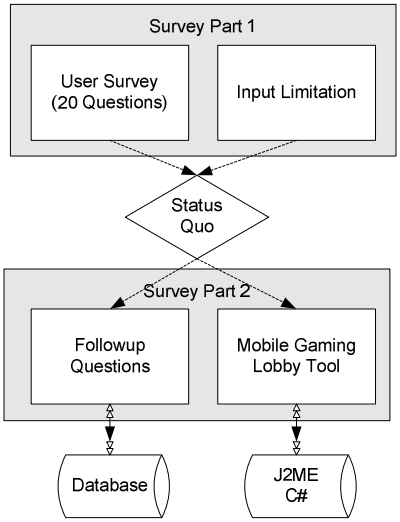


Figure 5.3. An overview of the complete survey, including database & J2ME relation

Figure 5.3 shows an overview of the mobile phone gaming research approach. In the first part, a short survey with the 20 most important mobile gaming-related questions was designed. Furthermore, the effect of input limitations on game performance was evaluated by using two test groups of players. The testbed required that one of the groups plays the given games on mobile phones, whereas the other group uses an emulator for mobile phones and a standard PC with a monitor display. The purpose of the emulator is to integrate the same mobile phone games on a standard PC. This enables the player to use a keyboard for the controls.

Each of the groups consists of six players who had to play the three different test games (from the RTS, FPS and arcade genre). By giving the players a fixed task, the required time was measured and compared afterwards. In order to make the findings more reliable, each game was played multiple (three) times. As a main finding from the first part of the survey, it was observable [Frit 2006.4] that the players with regular cell phones were significantly slower compared to the players who used emulators.

The second part of the approach is based on results from the first half (including the findings about game performance and the online survey). Hence, a follow-up survey integrates more technically related questions about the gaming behavior, especially player performance and preferences for game setup. Furthermore, the approach

includes the design of a mobile gaming tool, which is described in-depth later in this section.

By analyzing the mobile players' behaviour in the follow-up survey from Figure 5.3, the aim was to understand how the preferences are split up between different countries of Europe. The follow-up survey is based on results of its predecessor [Frit 2006.4]. Hence, the set of questions is reduced as much as possible to obtain a high number of answers and thus reliable results. The survey uses the database system from Figure 4.5: all questions were translated into German, English, French and Spanish. Moreover, the results are the source for the both research approaches in Figure 5.3. Table 5.1 illustrates the main questions of the follow-up survey.

Table 5.1. Main questions of the follow-up survey in mobile gaming

Question	Answer options
Would you like to play multiplayer mobile phone games with your friends?	(yes/no)
Would you also play these games with random participants?	(yes/no)
How important is a fast game setup with mobile phones for you?	(MOS: 1 to 5)
Would you pay to play mobile phone games?	(yes/ no)
Which game type do you prefer on mobile phones?	(FPS, RTS, Puzzle, Arcade, Action)
Do you take mobile phone games seriously?	(MOS: 1 to 5)
Would you also play mobile phone games if you had a next generation handheld like Sony PSP or Nintendo DS?	(yes/no)
How do you like the graphics on Nintendo DS and Sony PSP?	(MOS: 1 to 5)
How do you like the graphics for games on current mobile phones?	(MOS: 1 to 5)
On average, how long would you play games with multiplayer support for mobile phones?	(Minutes)

One main aspect is: How much impact does playing mobile games with friends/strangers have? And if there is a significant impact, how can that affect game design? Due to the results from the predecessor in [Frit 2006.4] a very short game time on average is already estimated; combined with explicit knowledge about the game preferences towards friends and strangers in mobile gaming. This will help to clarify the situation further.

Overall, 1,123 people participated, and thereof 1,080 participants answered all ten questions. The questionnaire moreover included a small demographic part (location, gender, age) to statistically analyze correlations with those values. The results underpin important aspects of the current mobile phone gaming situation:

Data gathered indicated that a majority of over 90% would like to play mobile phone games with their friends, whereas only 42% (36% women, 45% men) would consider playing the same games with random people (see Figure 5.4). Women tend to prefer not playing those games with strangers even more than men. These figures indicated that most users enjoy playing the games with friends. Especially when traveling with friends, a mobile game offers the opportunity to experience something together. For this purpose an in-game network setup is adequate, because the users probably have one particular game in common and can start a multiplayer session with it.

The other aspect of the results is that around 40% of the users are willing to play a randomly matched multiplayer game in a mobile context. Firstly, this means that more than 50% of the users prefer rather not to play instead of being matched with unknown people. However, the remaining 42% indicate that there is demand for random matching; the typical game situation for a random matching could be a bus or an underground train. An example: travelers on the train want to spend their time on a fast game setup with other gamers. One can assume that it is difficult to find a common game since the players do not know each other. Without knowing which potential games all other players have installed on their devices, it is not possible to create a multiplayer session quickly. Further results of the survey also indicate that other important aspects of the game design (such as content, graphical performance and game setup) do not meet up to the users' expectations.

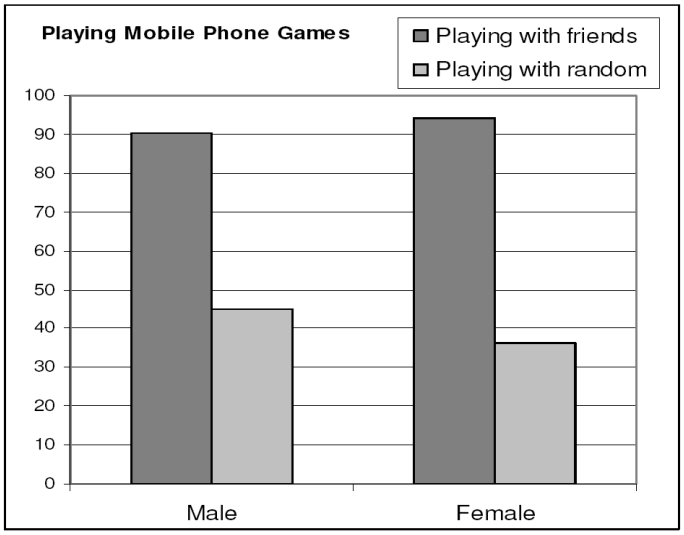


Figure 5.4. Male and female opinion on playing mobile phone games with friends / random people

Figure 5.4 shows the differences in player preferences for mobile gaming. It turns out that a majority of players prefers to play with friends instead of with random opponents. A deeper statistical analysis of the survey questions as well as the main findings from the lobby design can be found in [Frit 2006.3].

5.2 The Factor Mobility and its Technical Aspects

The aspect of mobile gaming offers various technical effects to solve the trade-off between wireless connections on the one hand and strict network condition requirements from the games on the other hand. Several important factors for the mobile gaming research approach in this thesis will be described in-depth in the following section.

Thus it is important to clarify the main problems of mobile gaming in order to decide whether or not they can be improved. This especially includes focus on a certain subset of problems, because it is not possible to completely solve the trade-off between resource-constrained devices (mobile gaming handhelds) and high user expectations (game requirements).

Afterwards, one should analyze the different mobile devices in order to point out their individual advantages. On par with that, especially for the mobile phone

section, it is necessary to identify a common standard to develop applications (due to the large number of different cellular phone types). The J2ME SDK (Java 2 Micro Edition) aims to create such a standard and support a majority of all next generation mobile phones. Based on the given J2ME standard, the technical details of the lobby tool (from the second part of the mobile gaming research approach) need to be analyzed in detail. In particular, the creation of generic interfaces for new mobile games needs to be addressed.

Moreover, another aspect is the integration of game-related functionalities into a mobile environment. Therefore, the technical details of next generation in-game communication must be observed. Main focus will still be on the usage of instant messengers as additional communication platforms while playing online games. Based on the relevance of these messengers and their frequent usage, a technical solution to adopt the chatting mechanism for persistent online worlds is described in detail. The result is an implementation design of generic software integration for every instant messenger; the provided interfaces offer any MMOG the opportunity to integrate the messenger into their game engine and to thus merge two existing technologies in order to improve the quality of communication.

5.2.1 Problematic Field: Mobile Gaming

The scientific research field of computer gaming in a mobile context offers a wide variety of problems. For example, the server-client model for the network infrastructure cannot simply be applied to mobile networks. In order to introduce the most common game-related problems, Table 5.2 gives an overview of the four main categories and a short description of the effects and results. Further problems exist, although due to the highly specialized mobile sector applications, these problems only partially affect the computer games.

Table 5.2. Illustration of the relevant problems in mobile game design

Problem	Description	Result
Missing Common Standard	Huge variety of different handhelds (mobile phones, PDA, gaming handhelds), no common game design standard for game interfaces, technology and network interfaces.	Individualization, adoption of successful game mechanisms (game cloning), missing compatibility because of incompatible network interfaces.
Resource-Constrained Devices	High game requirements based on games from the PC and consoles, adoption of known game mechanics. Players are influenced by the graphical standard from PC and console games.	High expectations of players conflict with the limited graphical performance of the mobile devices.
Input Limitation	Strictly limited input options, missing mouse/joystick, cellular input structure for mobile phones (depending on the mobile phones – usually ITU-T keypads are used for dialling)	Further game specialization, no common input standard, especially for mobile phones different key bindings exist, strictly reduced interactivity.
Protocol Structure	Available options: WLAN/Bluetooth or GSM/UMTS, high latency, instable network status, possible packet loss	Influence of the game mechanics, in-game inconsistencies, reduced interactivity.

Missing Common Standard. One of the major problematic fields is the missing standard for computer games in the mobile environment. Mobile handhelds (cellular phones, PDAs, gaming handhelds) have their own individual technical standards which creates a compatibility problem. One negative example is the design of quake 3 mobile, which was the mobile game of the year in 2005. Although it has a superior design, the game itself only supports a very limited number of devices (Samsung Nexus and LG VX360).

This effect is even more drastic for the next generation game handhelds (like Sony PSP and Nintendo DS). The game structure clearly prohibits adopting a game from another gaming platform.

But both of the mobile gaming approaches in this thesis will mainly focus on the technical aspect of missing common standards and they aim to close the current gap between the mobile devices. Technical details will be discussed in the following sections.

Resource-Constrained Devices. The technical configuration of mobile devices has witnessed significant change over the last five years. Especially mobile phones have been further developed and are now capable of distributing enough resources for simple gaming applications. Although one can still observe a major difference between gaming oriented handhelds and mobile phones, the hardware's general performance has increased significantly. A trend towards touch screens can also be observed: the most popular example is the I-Phone because the device offers a single touch screen for most of the functionalities. Newer models from Sony Ericsson and other retailers also feature touch screens, and therefore one can expect that the input technology of mobile devices will change within the next five years (towards including more touch screens and larger displays).

However, resource-constrained devices still pose a problem that will remain consistent over time because the evolution of mobile handhelds will always be compared to the alternative solutions (PC or consoles). It is unlikely (due to the release of the Playstation 3 and the continually enhanced performance of PCs) that the current trend in fixed platform evolution for gaming has come to an end yet.

Input Limitation. The effect of input limitation (due to the limited space) is still one of the most significant disadvantages of mobile devices. Especially the current generation of mobile phones only provides a very limited cellular structure (usually ITU-T keypad, which is also used for dialling), which reduces the game performance noticeably [Lazz 2006].

The solution for such limitations can either be a software-based approach (finding a common standard for mobile phones to integrate further button options) or the design of more intelligent hardware solutions. One should also notice that the trend towards touch screen displays for a smoother input can be observed for any of the mobile

devices (PDAs already have a touch screen, the Nintendo DS has an additional touch screen input and even mobile phones like the Apple I-Phone or the Samsung Nexus integrate them).

Protocol Structure. The current network structure (as described in Section 2.2) features the trade-off between the available solutions. Currently, both opportunities (UTMS and GSM) offer insufficient game support for various reasons. Nevertheless, the development of current mobile games aims to integrate WLAN and Bluetooth for most of the multiplayer games because these techniques do not require additional communication with a central server.

5.2.2 Mobile Devices and Java Micro Edition

One major aspect in the mobile gaming environment is the lack of standardization of the devices. As an example: most of the games from [Jamb] support a very limited list of mobile phones (generally between two and 20 different models). Also gaming oriented handhelds (like the Nintendo DS and Sony PSP) strictly limit their games so they cannot be played on other devices. In the case of mobile phones, missing standardization can be attempted with a software solution. J2ME is required to create a common standard for applications on mobile devices.

The J2ME (Java Platform 2, Micro Edition) integrates the programming language Java for embedded consumer products such as mobile phones and PDAs. It aims to integrate a basic set of features. So-called “profiles” offer an API for the devices. The profiles for mobile phones are called Mobile Information Device Profile (MIDP) and applications that use the functionality of the profiles are called MIDlet. The idea behind the MIDP is to standardize input functionalities and displays of mobile phones. Especially games for mobile phones frequently use J2ME because it offers the opportunity to run the game application on more devices.

Gaming applications often use special functions (mostly relatively graphical intensive, fast-paced commands). The design of mobile phones has changed significantly; the iPhone [IPho] already integrates a large touch screen as the next generation of mobile phone inputs. As soon as this technology becomes more important for other mobile phones, members of the JCP (Java Community Process) can submit a proposal to expand the J2ME library. This process is one of the major

advantages of the J2ME platform because newer technological features will be integrated. As soon as these functions are supported, every game MIDlet can use them.

5.2.3 Academic Approach: Lobby Tool

In this approach the J2ME platform is used for the design of a game-related application: the mobile lobby. This approach abstracts from the pure creation of new game applications. Players' preferences indicate that an easy game setup and a short overall game time are significant influencing factors (refer to [Frit 2006.4] for a deeper analysis of players' preferences for mobile games). One method to reduce the game setup time and to increase the chance to connect to other players also improves the usability of all mobile phone games.

Unfortunately, the current mobile phone design offers a personalized matching mechanism for every multiplayer game. Once the gaming application starts, one can look for potential opponents (who also need to have a running version of the game), in order to set up a multiplayer session. It is not possible to find players with other games. Due to the very limited time in most mobile environments, the probability of finding a potential opponent is relatively low. With a constantly increasing number of new game releases, the chances for two players to start the same mobile phone game decreases even more.

A general solution for this problem is the creation of a lobby system (based on J2ME), which supports the majority of current mobile phones games. The lobby itself needs to be designed as a chatting environment, where all users have their own profile. In general, the lobby tool is game-independent, which means that every other user in range can be found. As soon as more than one person is in the chat mode, one can look for a common game. The profile of each player contains the names of installed game applications on his/her device. Therefore, other players receive information about which games a user can theoretically play. This knowledge further speeds up the organization of game sessions.

A faster game setup can be realized by creating interfaces to each of the installed games, so players can meet at the lobby and directly access the games on their mobile devices. A technique like this would make it necessary for the game design to

integrate the appropriate interfaces into the games. Without support of the games, it is only possible to start the target gaming application, but afterwards it would require a manual multiplayer setup.

By using the J2ME SDK as the underlying technology, it is also possible to integrate further devices (like PDAs) that are capable of running a Java virtual machine. The common standards for the mobile phone sector might not be applicable for other mobile handhelds (this strictly depends on the goodwill of the mobile phone manufacturers and the development of further PDA support).

5.2.4 Problematic Field: Instant Messenger

Besides the aspect of player matching in a mobile environment, there are also other ways to improve the current game situation. One possible option is the creation of game-related applications for the users. Currently, mobile devices do not offer the necessary hardware to compete with personal computers. Furthermore, both consoles and personal computers are being redesigned continuously to further improve the hardware's performance. Over the long run, mobile devices (due to their size) will always offer less performance compared to stationary systems (like the personal computer or a console).

As long as newly released games have high system requirements, which can only be matched by the personal computers, mobile device games will be less attractive. Currently, most users complain about the poor graphics within the games [Fritsch 2006.3]. Therefore, alternative methods need to be developed to support mobile gaming. One method to support gaming in a mobile environment is the integration of mobile devices into current MMOGs.

The aspect of player communication plays an important role: many players want to stay in touch with friends who also play online. The mobile devices' constrained resources prevent them from running the game engine, which is needed to chat with others. By integrating a method to run the pure chat environment on mobile devices, a user can stay in touch with in-game friends even when travelling. The game itself cannot be played in a mobile mode due to the missing hardware requirements. However, the chat itself does not require a high performance; therefore a text interface for mobile devices can be designed.

Since the hardware limitation is not the only problem for mobile devices, another relevant aspect is the limited input structure, which also affects the general game play [Fritsch 2006.3]. Besides the obvious problems of controlling a fast-paced game with such limited input possibilities, a further in-game communication would expand the users' possibilities.

It is therefore necessary to broaden the common term of mobile gaming from a participating perspective to a more generalized view, which also includes game-related activities. In the fixed multiplayer environment, instant messengers have become an important alternative communication method. Especially in persistent online environments (MMOGs) players tend to use additional communication software in order to stay in touch with non-playing friends.

The protocol structure of instant messengers relies on messages (depending on the publisher). Generally, ICQ/AIM uses the OSCAR protocol (Open Service for Communication in Real-Time), whereas the MSN (Microsoft Messenger) uses its own protocol. These protocols are not compatible, although multi-protocol applications exist (which use sniffing methods to create appropriate commands). The instant messengers offer a way to integrate user communication into games and they also provide constrained-resource devices (like PDAs and mobile phones) with the opportunity to run the applications. By using them, one can connect a mobile device player with a stationary system. Two main interfaces need to be designed: one interface (GUI) for the mobile devices, which allows the user to chat while being mobile. The second interface is the integration of instant messengers into gaming applications, so players on stationary systems can use the in-game chat to communicate with their friends who are travelling.

5.2.5 Academic Approach: Generic IM Integration

In order to integrate game-related activities in a mobile environment, players would need to demonstrate clear interest in using applications that are specifically related to gaming (like instant messengers to communicate with fellow players) but without a real game content themselves. Mobile devices do not offer the necessary hardware support to run a client of a next generation MMOG, as mentioned above. Hence, in-game chatting mechanisms (in-game channels) of commercial fixed network

MMOGs are also not available, which leads to the fact that the user cannot communicate with in-game contacts in a mobile environment.

In order to create an independent solution for all MMOGs, it is necessary to integrate a popular communication method (like instant messengers) into all of the current games and to thus create a common solution. With the TOC (Talk to OSCAR) protocol, the basic functionalities of ICQ and AIM can be used without the need to run the original graphical interface of the instant messenger client. Therefore, a generic implementation (a software middleware approach) enables these functionalities to be contributed to every MMOG by creating general interfaces. As a result, publishers can implement the graphical front-end of the instant messenger themselves and rely on the given communication interfaces.

As an example, it is possible to use the TOC gateway server to embed the rudimentary functionality of AIM and ICQ into a current MMOG (this thesis focuses on the exemplary implementation for Eve Online [EveO]). The commands available with the parameters required from the TOC server are included in the software application and generic interfaces and they offer simple commands like "*send* <*string*>". Due to the generic nature, it is generally possible to integrate every common instant messenger into different MMOGs by using the same interface method. Accordingly, a player could see all of his/her online buddies, even those from different games, in a merged friends list.

The advantage of integrating instant messengers in a MMOG environment is the possibility to create mobile support with its own GUI (graphical user interface). J2ME offers the necessary functions to create a simple animated chat client with a buddy list and write/receive options. With this application, the user can connect to his/her instant messenger account and communicate with any of his/her friends online. Assuming that the instant messenger is integrated into different MMOGs, it would increase the communication potential even further, because the user can stay in touch with any of the in-game friends without the need to run a game client.

The major technical problem of this approach is the creation of a common namespace, because the in-game nicknames should be kept consistent in the instant messenger as well. However, some of the in-game nicknames might already be stored in the instant messenger. It is therefore necessary to create new instant

messenger accounts for all of the users. Each of the game accounts is linked to an appropriate instant messenger account, which will be created simultaneously. As a general naming possibility for the Eve Online example, the resulting user name would be *nickname@eveonline.com*.

The OSCAR protocol offers the user a direct forwarding method. As soon as the new account is created, each message can be forwarded to the user's original ICQ account automatically. As an example, the user HUGO from Eve Online creates a new account: HUGO@eveonline.com. Furthermore, the user already has an active ICQ account; therefore it is possible to forward any of the messages from the HUGO@eveonline.com account to the user's original account.

The creation of an additional account per player also creates overhead. Therefore, it is necessary to evaluate whether or not this overhead exceeds the usability of the approach. By creating a new account for each user, the in-game nicknames (like HUGO) can also be used in a chat session. This consistent naming helps the in-game user to identify the communication partner, because he/she has the same nickname as in-game. A player will therefore receive messages from buddies with the relevant nicknames, and therefore no name collusions occur. Furthermore, it is also not possible to assume the identity of another player, because as soon as the account HUGO@eveonline.com is created, an additional account with the same name cannot be created.

With the simultaneous integration of more games, further accounts will be created. It is also necessary to discuss whether these multiple chat accounts scale with usability. Since a player can have the name HUGO in three different games, a separate account with HUGO@gamename.com will be created for each of them. However, this is absolutely essential because one cannot assume that the same user is called HUGO in all three games. Particularly well-known, popular names are quickly taken after a game has been released. Players can also show a switching name behaviour from one MMOG to another; a fixed name approach would not support this behaviour.

Generally, it is possible to merge any two given IM accounts by using a namespace server (like OPIS in case of AIM and ICQ). The namespace server forwards every message from one target account to another; in case of the Eve Online example all

messages from the newly created nickname@eveonline.com account would be forwarded directly to the user's existing account.

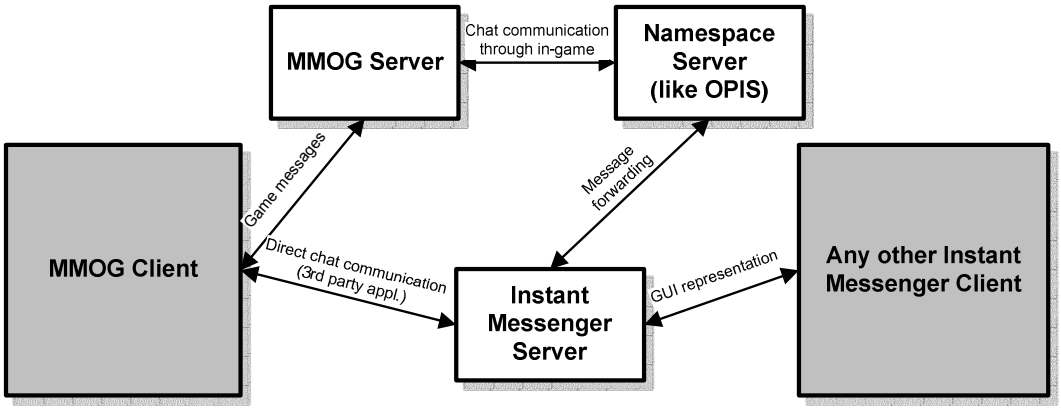


Figure 5.5. Creation of a common namespace by using a namespace server

Figure 5.5 illustrates the common namespace and communication between the in-game MMOG client and the instant messenger client. As soon as the MMOG client tries to connect to the instant messenger server, a request is sent to the namespace server (in the case of AIM and ICQ this is OPIS). The namespace server communicates with the appropriate MMOG server in order to ensure that the password and user name are correct and afterwards, it replies with a list of forwarding targets (this list includes the user's normal instant messenger account).

By using the flexible namespace technique, both important aspects for a consistent user naming policy are fulfilled:

(1) Each user of the game receives a unique name that is the equivalent of his/her in-game nickname and

(2) Each user can furthermore maintain already existing instant messenger accounts.

The in-game representation of messages will be shown with the relevant nickname, whereas the common instant messenger client will still use the original account.

5.3 Implementation of a Mobile Gaming Communication

Based on the follow-up survey and communication situation in the mobile gaming environment, this section contains an introduction to the two research approaches. The MCChat offers a flexible solution for a mobile lobby to greatly reduce the matching time for mobile gaming. The IM integration in games uses the high importance of instant messengers for next generation game communication and closes the gap between in-game and real world buddy list contacts.

5.3.1 Approach I: MCChat – A Mobile Communication Lobby

The up-to-date mechanism for game finding in the mobile phone sector includes that each of the games features its own connection utility. Especially location aware games have a big disadvantage because potential players do not run the game client all the time and thus even if two players were at the same location (like subway train station or in the bus) the probability of having both with a running game client is very low.

In order to increase the chance of finding other matching players, one should have a general software portal to track ongoing games. The software platform then offers interfaces to each of the mobile games, therefore giving all participants the opportunity to start the game session directly from the lobby. The mobile lobby approach features a text-based chatting application (like the Battle Net from Blizzard [Bliz]) to create a mobile forum.

Game oriented handhelds (like the Nintendo DS and Sony PSP) already have their own individual lobby systems. Both devices use the WLAN technology (IEEE 802.11b) for network communication. A player with these devices can search for ongoing game sessions. Depending on the device, only a very limited number of configurations are needed. For example, the Sony PSP offers three WLAN channels (1,6,11) to support different game sessions at the same location. As soon as two players are connected in a game session, one of them can act as a host and start a common multiplayer game.

A general game lobby for other mobile devices must be game-independent and capable of handling even larger numbers of incoming players (10+) without previous

network detection (ad-hoc network). Each of the clients can configure its game interfaces, thus each other player can see the installed mobile games, which further speeds up the selection of potential opponents. Moreover the lobby acts as a portal, where every player can chat while waiting for the next game to start.

The mobile gaming lobby uses the DirectPlay API [Dire]. This API is designed to support communication in multi-user applications such as games. The application can use the given methods to send information without the knowledge of the underlying network. The lobby application is designed in Virtual Studio .NET 2005 with the DirectPlay for Pocket-PC-SDK. It uses TCP/IP as the transport protocol.

The implementation itself contains the WinMain() function to start the graphical user interface. This is displayable on PDAs as well as notebooks; integration into mobile phones is also possible. A player can use the search button to look for game sessions. If a game session is already ongoing, then a player can select the session and join the chat (this is implemented with the JoinChat() function). It is also possible to host a session (with the HostChat() function) in order to wait for others to join. The host of the game session will also be the host of the mobile game. If the host disconnects, the session will automatically be closed.

The underlying functions for the network setup use the DirectPlay methods. Basically, as soon as a player starts to host a session, the DirectPlay method automatically assigns a port between 2302 and 2400. The search method uses a broadcast to identify all ongoing game sessions and lists the result for the user. Once a player decides to join a session, a direct connection to the host is established. The name table for the existing players is updated and the new player joins the chat. This example can be found as a flow diagram in the appendix (Figure 9.22). For detailed information about the implementation refer to [Lehm 2006].

Game sessions are started by a player who acts as a host and waits until enough other players have joined the game session. By starting the game from the mobile lobby, the chat application will also terminate itself and all gathered players join the target game. Figure 5.5 illustrates the prototype of the mobile gaming lobby.

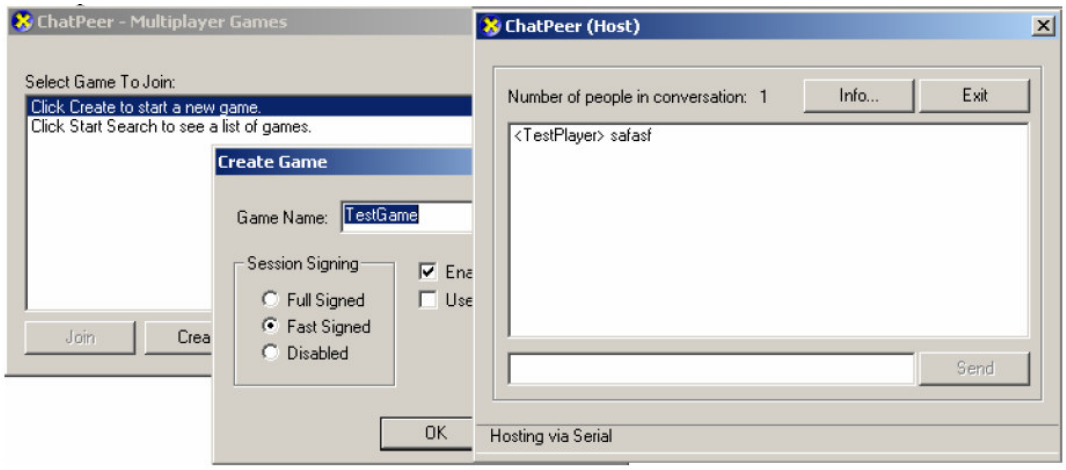


Figure 5.6. Screenshots of the mobile gaming lobby

Both PDA and mobile phones have the opportunity to run customized software from 3rd party applications. The mobile gaming lobby was designed in .NET and uses the J2ME platform to support as many mobile devices as possible (PDAs and mobile phones). The underlying network technology is WLAN, because every current PDA as well as most of the next generation mobile phones supports this technology. It would also be possible to integrate Bluetooth as the network technology. The advantages of WLAN (higher range, common standard among PDAs, etc.) are responsible for the design decision [Fritsch 2006.3].

The mobile lobby is implemented in .NET. The prototype is designed to run on PDAs. This decision is based on the technical equipment which is available for the testbed. One should keep in mind that due to the usage of J2ME, it is also possible to convert the lobby to a MIDlet that runs on mobile phones. For evaluation purposes (testbed) the implementation runs on PDAs. The devices deployed are iPac PocketPCs H4155 with 400 MHz Intel processor, 64MB RAM and a 240x320 pixel TFT display.

Each player only has to select a nickname and they will automatically be connected to the lobby. In the current version, the application contains a single chat-room, where all players can meet. No further configuration is needed; the underlying ad-hoc network manages the IP addresses on its own. Therefore, the lobby offers a fast and very simple way to get in contact with as many other players as possible.

5.3.2 Approach II: In-Game IM – Instant Messengers in MMOGs

By integrating instant messengers into a MMORPG, a highly flexible solution is needed which is as expandable as possible for both new instant messengers and new MMORPGs. However, the current work focuses on the integration of a single instant messenger (the AOL IM) into a target MMORPG (Eve Online).

Before taking a closer look at the general idea of a next generation in-game message interface, one must analyze the given structure of the target instant messenger. The AIM was initially only available for AOL customers; nevertheless after 1998 the instant messenger was opened up for a wider audience. Besides ICQ, the AIM is available for every common operating system (including mobile systems).

Basically, AOL and ICQ use a single closed-source protocol called OSCAR (open service for communication in real-time). Several servers are organized in the accordant network: one central authorization server and several BOS (Basic OSCAR Service). Figure 5.7 illustrates the login procedure. Admittedly, there is no reliable documentation available; although since the SDK for third party applications was released in 2006, there is a second protocol. The TOC protocol is a text-based wrapper that uses gateway servers to translate TOC commands into OSCAR packets and vise versa. Its complexity is therefore reduced for the basic IM features (like login, send, receive and buddy list functions).

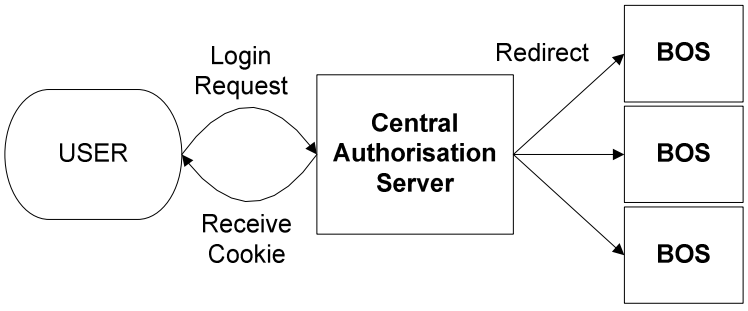


Figure 5.7. Login at central authorization server. By submitting the password successfully, one receives a cookie and is redirected to a BOS

Based on the idea to implement an instant messenger into MMOGs, it is necessary to design a middleware application between the game engine and the OSCAR SDK.

The design provides general interfaces in the game for all chat functionalities such as send, receive, login, and add. The other part of the middleware is the interface to the TOC gateway. This is implemented by OSCAR SDK, which features the basic commands.

The general implementation approach is illustrated in Figure 5.8, where the Eve Online client is combined with the interfaces of the TOC gateway servers. By using the basic TOC commands, a player does not need to have a separate ICQ or AIM. Instead, the user can log in with his/her existing account with the Eve Online game client and the middleware will offer the necessary chat functionalities. As a result, the chat will be visible in-game instead of via a 3rd party application.

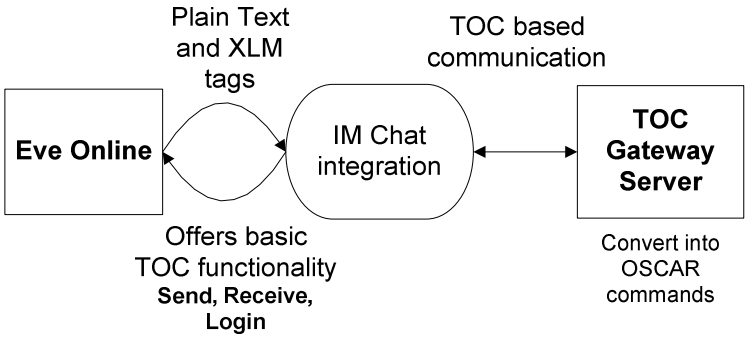


Figure 5.8. Integration of the AIM into Eve Online: communication takes place through simple TOC commands

Due to its basic command structure, the game client (Eve Online) has many ways to implement the graphical part of the user interface, either by using the standard buddy list options with separate pop-up windows and mouse support or by breaking it down into an IRC text-based chatting system. As a result, the user can chat with his/her buddies without the need to install separate instant messenger software. In a mobile context on the other hand, a simple GUI is created, which allows resource-constrained devices (like PDAs and mobile phones) to run an instant messenger chat. Game-related activities can therefore be performed while being mobile and the user has the opportunity to stay in touch with in-game friends.

The IM interface design is split up into three main classes: **ClientManager**, **ClientInterface** and **BuddyList**. As one can see in Figure 5.8, those classes form the

center of the UML diagram. By further analyzing the three classes, one will see the distinct function that each of them fulfils:

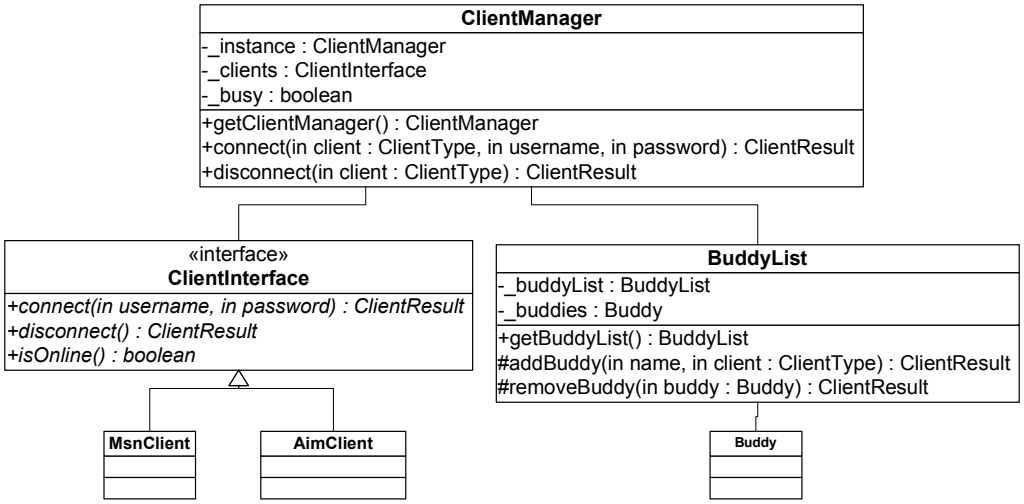


Figure 5.9. UML diagram of IM Chat integration

The ClientManager is responsible for authentification; thereby it is vital that only one instance exists in parallel. It handles the login with the client, its username and its password by creating an appropriate ClientInterface. The ClientManager offers the central interfaces to the game engine. All basic functionalities (such as send, receive, login, logout, etc.) are requested directly at this class. However, the options are limited to AIM interfaces. The current version of IM Integration only supports the AIM and ICQ chat client, but the generic structure easily enables further integration of any other client. According to Figure 5.8, this class represents the interfaces to the game engine. Only graphical implementation is missing because it needs to be implemented individually for each MMOG.

The ClientInterface, on the other hand, is responsible for direct communication with the TOC gateway servers. Once an interface method at the ClientManager is requested, the relevant procedure at the ClientInterface class is requested to communicate with the TOC server. The ClientManager translates the interface commands from the MMOG into commands that can be understood by a TOC gateway server. It uses the TOC command structure in order to send and receive data from the network, hence offering all necessary interfaces to the TOC servers. An

example would be a simple “connect” operation, which is requested by the ClientManager and translated by ClientInterface class.

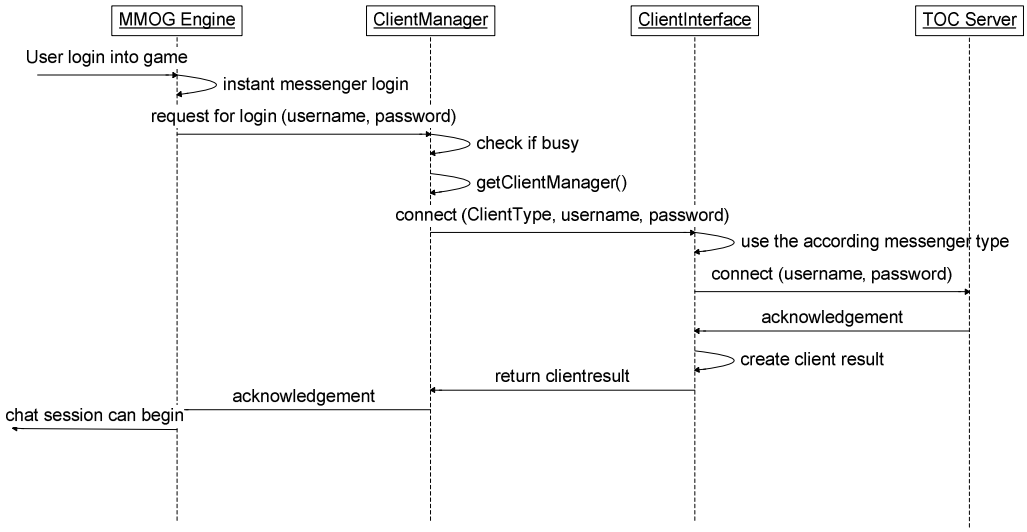


Figure 5.10. Sequence diagram of the login procedure for IM integration

The sequence diagram in figure 5.10 illustrates the data flow during the login procedure. The game engine tries to log the user in at his/her instant messenger account as soon as he/she logs into the game. The initial request uses the interface at the ClientManger class. The relevant information plus the ClientType are forwarded to the ClientInterface class. This class chooses a corresponding message type (in the case of the flow diagram it is ICQ/AIM) and communicates with the gateway server. Initial information from the MMOG engine is translated into commands that the TOC server can interpret. After an acknowledgement is returned, the ClientInterface notifies the ClientManager and so on.

The third main class is the BuddyList. This class manages contacts in the current users’ friends list. Only methods like add and remove are supported. This class has a slightly different function, as its functions have nothing to do with the direct chat mechanisms. The message structure for these functions for the TOC gateway server differs slightly from the simple communication commands, therefore a disjunction into the two sub-classes BuddyList and ClientInterface is chosen.

The IM interface was intentionally designed to fit into the Eve-Online client; however, due to the lack of cooperation from CCP gaming, it was ultimately not possible to integrate the client into the game. Alternative open source applications for large MMOGs did not exist; most of the projects were still in the developmental phase. Therefore, the implementation of a simple GUI (in addition to the graphical interface for mobile clients) was required. To substantiate the functionality, this simple GUI was designed. Moreover, a testbed was required to further verify the performance of available features. The analysis of testbed results is discussed in chapter 7.

5.4 Summary

The chapter on next generation mobile games introduced the special motivation regarding the mobility aspect. It was determined that the problem of insufficient mobile network support is mainly responsible for the current lack of mobile MMOGs.

In order to improve the current mobile gaming scene, a survey evaluated the users' behaviour. Based on the results, two main approaches have been introduced. The MCChat features a mobile lobby platform to significantly reduce the setup time of mobile game scenarios. On the other hand, the integration of instant messengers allows the players in mobile environments to communicate with in-game friends (even those from Internet MMOGs) without running the game client.

6. Approach III: 4MOG Middleware

"Think? Why think! We have computers to do that for us", Jean Rostand.

This chapter covers the 4MOG middleware approach, which offers a software solution with interfaces to the game client (upwards) and network layer (downwards). The approach includes the design of a middleware application which reduces technical implementation efforts. Thanks to this reduction, the designer can focus more on the content of the games, hence increasing their quality. The first section outlines requirements of the middleware and illustrates the planning process. In the second part of the chapter, the design and implementation is described in detail. Finally, the validation section contains a comparison with a similar middleware as well as a testbed setup. Results of the testbed are described in chapter 7.

6.1 Requirements

The first step in creating a middleware application is to define its requirements. It is necessary to analyze the situation in order to determine whether if a problem actually exists. If a problem exists, it is necessary to evaluate it closely and to create a solution as precisely as possible. Therefore, this section contains a detailed overview of the problematic field. Its purpose is to outline the motivation in creating the middleware.

Functional requirements are then introduced briefly. The section describes the middleware's interfaces, which also contain information about the core functions. Each of the functions is described in detail to clarify middleware usability.

Furthermore, non-functional requirements are introduced. This section contains information about additional aspects like the human factor, documentation or hardware considerations. These aspects also need to be addressed since they can be important influencing factors for the usage of middleware. Especially the human factor (player behaviour) plays a vital role in the gaming context [Fritsch 2006.3].

The constraints of middleware are introduced as well. These constraints pinpoint restrictions that the middleware has, in order to clearly describe what exactly the middleware can do and which features are not supported. This section shows the limitations with regard to the client, the network, the protocol and the hardware.

Finally, this section features a detailed usage case model to illustrate the benefits from the middleware application. A most understandable description of the software helps to support a smooth implementation.

6.1.1 Overview and Motivation

Two main aspects need to be considered to understand the motivation of creating a middleware application. The first aspect is whether or not the middleware is necessary at all, because non-repeating problems can also be solved with a specific single solution. The second aspect is the decision as to which functionality a middleware should have.

Compared to the current stand-alone game architecture, a middleware is an additional software interface. Generally, the middleware application needs to improve the initial situation. In the case of gaming, this can either be done by defining standards for multiple games (especially important in the mobile phone sector with hundreds of different devices) or by creating a performance improvement. However, it is necessary to keep in mind that a middleware also has developmental costs. In order to be useful, the middleware's functionality needs to compensate for these costs. Currently, most of the commercial games do not use any middleware for two main reasons:

(1) The game industry often produces "clones" of existing game types. Successful software pieces, like a game engine, are often resold for smaller game producers.

(2) Fixed costs of game development are very high compared to the variable cost of duplicating a CD or DVD. This leads to the fact that most commercial game producers are not willing to share their solutions with others for free because they have already invested money in them. Commercial software solutions exist (like game engines), which are sold

to multiple game companies, however, especially the small companies cannot afford them and they would need a freeware solution.

The development of a gaming middleware needs to take these aspects into account. In order to create a successful middleware application, the functionality must improve the current game development situation. One method to achieve this is to reduce the effort needed to produce a game (because the other method would be to enhance the performance, which is more complex to implement with middleware software). The general problem for small game companies is the high initial developmental cost for MMOGs. Currently, standard features in MMOG development include technical aspects (such as connection, authentification, and storage), the creation of the content (design of the game world and the NPCs) and the balancing (beta testing of the content with player feedback). In order to reduce the overall cost, the middleware application needs to cut implementation costs (for example by creating a solution for the technical aspects), so the developer can focus more intensively on the other parts.

A game designer from Electronic Arts [EA] stated that about 80% of the game development effort is individual content. This includes the design of the online world, the setting, the storyline and interaction in the virtual environment. In fact, the creation of individual content cannot be automated.

As an example for an MMORPG, the design of content includes the creation of quests for the players. Each quest contains an individual storyline, sometimes multiple triggers for events during the quest as well as a NPC to interact with. Once the quest is designed, most of the work cannot be used for another quest because players expect varying content. Most of the players strongly demand non-repeating content in the games. In order to be innovative, one can frequently not reuse already existing content. The general principle of in-game interaction between a player and a NPC remains similar. However, the difference between quests is the individual quest text, the dialogue, the quest content (player's duty), etc.

Therefore, one needs to look at the remaining 20% of the game design effort. Most of the remaining tasks are technical implementations such as login methods, player authentification, in-game object management and a chat system. These functionalities can be found in any of the current MMOGs. Due to their frequent

usage, these features offer an opportunity to reduce the development effort. Since it is expected that every upcoming MMOG will also adopt these features, a common middleware solution would reduce the developmental costs.

A good example for the current game development situation is the creation of the current top-selling MMOGs. As an example, the creation of Everquest 2 [Ever] included the casting of Hollywood actors for the in-game characters. More than 250 programmers participated in the project [Soe], overall production costs have exceeded $100 million. Since this is just an example, other games have similar costs. The game developing industry recognizes this trend. Current top-selling MMOGs need to sell at least 1 million copies to cover the initial costs [Digi]; this number is expected to swing up further in the future. Therefore, a common reduction of costs is needed to give the developers the opportunity to focus on the creation of game content.

Before focusing on the concrete software design, it is necessary to pinpoint the stakeholders of the middleware application. The large game producers like Sony Online Entertainment and Electronic Arts frequently release online games. Therefore, they already have customized solutions for the technical background of their games. As an example: Sony Online Entertainment released Everquest 1 in 1998, the underlying zone system, the server architecture and the login methods are reused in all of their other MMOGs (like Planetside, Everquest 2, Star Wars Galaxy, etc.) [Soe].

Due to their already existing customized solutions, it is unlikely that large game producers will require a middleware. However, the creation of MMOGs has increased drastically over the last five years. As [MMOR] shows, currently more than 200 different MMOGs have already been released or are in the developmental phase. Most of them are produced by smaller companies. Since it is time-consuming to create an MMOG (on average a game needs 3-5 years of development), cost reduction is especially interesting for smaller companies.

The objective of smaller game producers is to create high quality game software that can compete with top-selling MMOGs. Large companies like ID software [ID] are developing the next generation in gaming engines on a regular basis. Their main income is based on re-selling the engine instead of creating game applications on

their own. A middleware application with the basic technical functions (player administration, chat, object management, etc.) would reduce the costs of creating a game because the smaller companies would not need to purchase these functionalities from big retailers. This would especially support the smaller game companies in producing MMOGs and it would give them the opportunity to focus more on creating content in their software.

Figure 6.1 illustrates the advantages of middleware from the different perspectives (the user and the game designer). The middleware's functionality offers a basic core set of functions for the game design. This core set includes a simple player authentification, player tracking, network support (simple server-client structure) and basic chat features.

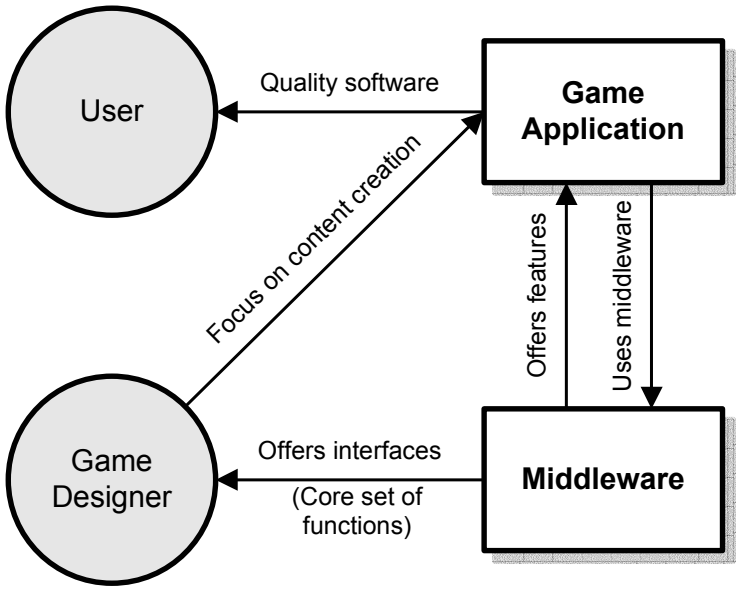


Figure 6.1. The 4MOG application and advantages from the user's and the game designer's perspective

From a game designer's perspective, the middleware offers interfaces for standard game features (such as chatting). By using these interfaces, technical implementation effort can be reduced. Especially in MMOGs, the balancing aspect (a fair design of the different player classes) and the creation of an online world is time-consuming. The advantage of a middleware system for the designer relies on the set of core

functions. Simple implementations for some of the functions (i.e. connect) exist, although a complete core set covers a larger problematic field. By using the middleware application, the designer is supported in game creation process. The middleware's interfaces can be used without precise knowledge of the implementation of each function (black-box principle). Since multiple interfaces exist (and not only a single connect function), the technical implementation effort is reduced.

From a player's perspective, the software's quality level is expected to improve (assuming that additional time is used for content creation). A well designed MMOG requires sufficient time to configure the content accordingly. The middleware does not influence the content directly; it shifts the developer's focus: from technical implementation to more individualized content creation.

6.1.2 Functional Requirements

After an initial introduction to the problematic field of the current MMOG design, one must precisely define the functionality and requirements of the middleware.

The core set of the middleware describes the basic functions for any MMOG (such as authentification, player tracking and chatting). These functions must be designed with regard to the current game development. As an example, the chat function must support different channels (i.e. personal chats, group chats), since most of the online games allocate communication that way. With regard to the evolution of mobile gaming, these features will be important in the future as well since mobile games are often clones of successful PC or console games [Jamb], [Harm 2004].

The player's and the designer's perspective are different points of view for the middleware. The middleware directly influences the developer, whereas influence on the user is indirect. The middleware offers the following core functions:

Administration of Player Data: The administration of player data can be performed by the middleware if the developer uses the interfaces. This administration includes the storage of all relevant data for each participating player, such as character level, class, location, etc. Generally, the middleware functions must be generic so that they can also support other in-game objects (like barrels, chests, etc.). Therefore, the

middleware can also keep track of these objects. A generic class design also offers the opportunity to customize this administrative function, depending on the MMOG. As an example: the function “position()” could either contain 2D or 3D coordinates.

The main disadvantage of object tracking in MMOGs is scalability. With a growing number of objects, the server’s response time increases. Common design strategies for this problem are either instancing (creating a copy of a part of the online world for a group of players) or distribution of players with in-game events (creating multiple spots of interest so players are allocated more evenly across the game world). After separating the players, the in-game load is distributed on different servers.

Combined with the middleware’s monitoring aspect, it is possible to detect time-based cheating (like speed hacks for a higher movement). By tracking the relevant data, it is possible to roll back the data if the movement speed exceeds the game limitations.

Send and Receive Operations: The middleware application should handle messages between the server and the client implicitly. In particular, it has a standard repertoire of message types (like status updates, login methods, etc.), which the developer can use. These standard messages contain a fixed syntax so the middleware can identify them. As an example: the login method must at least contain the user name and password. In order to not limit the application types, the middleware must offer an opportunity for additional message functions. These individual messages can be created by the developer, whereas the middleware offers a guideline to create them. The send and receive operations are required for every MMOG. An implementation of communication between server and the clients is the middleware’s focus. This focus includes the design of messages between server and client (plus the opportunity to create individual message types).

Creation and Termination of Game Sessions: Depending on the game type, it is necessary to also implement typical client- and server-based functionalities like the creation of a new game session. Further game relevant parameters such as a maximum number of players or the connectors for the next zones (if the world is zone-based), should also be offered as an optional feature. Clients can request a server list and then authenticate at one of the servers.

Communication Functionalities: The communication aspect is also important for the design of the middleware application. General in-game "send" and "receive" functions as well as a channel system offer typical communication methods of current MMOGs. Due to the expandability it is generally possible to integrate further communication concepts (like the IM in-game integration) into the middleware as well, which further supports communication with out-of-game friends.

Monitoring: Another feature should be the monitoring of player relevant data. By tracking each player's in-game activities, this method can help to gather important information about the in-game world distribution (which is required for load balancing) or the usage of certain abilities (which is required for in-game balancing of the content). As an example for in-game distribution, the middleware could store player distribution in the online world in a log file. A developer could analyze the log files afterwards. Additional in-game content in other parts of the online environment can help to even the loads. Since the monitoring aspect features a wide area of sub-problematic fields (like further features such as cheat protection, load balancing, content balancing, individual in-game content, etc.), the focus of the middleware relies on connectivity between clients and the server. The middleware's monitoring function is reduced to a tracking of the in-game distribution of players.

6.1.3 Non-functional Requirements

This section covers the middleware's non-functional requirements. These are immanent factors of the problematic field. Thus, their actual influence on the middleware cannot be analyzed completely during the planning process. Nevertheless, it is important to discuss them during conceptual planning since they will have an influence on the software usage at a later stage.

Human factor: The middleware is not completely autonomous; a major part of its functionality depends on the way it is adopted by the developers. Since the middleware aims to support game developers, one can assume that most of them will have expert knowledge of computer game structure. It is not expected that hobby programmers use middleware on a regular basis because the design of an MMOG is overly time-consuming for just one person.

A detailed description of methods helps to clarify the middleware's functionality. The developers will be more likely to adopt the design structure if class diagrams provide a design overview. This is important for the interface design because the designer needs to understand what the middleware interfaces offer in particular.

In contrast to released software which needs to offer functions that have already been implemented, the middleware contains abstract methods that can be implemented further. The middleware's final version needs to feature well documented classes [JavaD] in order to clarify the methods.

The middleware should offer a simple GUI to support the monitoring functionality. The framework and interfaces of the middleware do not necessarily need a graphical representation because well documented classes and adoptive interfaces are more important for the game developers.

Performance: The middleware needs to support all of the current MMOGs. This leads to performance requirements; especially player and object management are calculated in real-time. With regard to [Beig 2004], the players' latency should not exceed 300ms because performance decreases continually with an increase in latency. As a result: all of the middleware's player functions that call for acknowledgements in real-time should be aware of this latency.

The middleware also has interfaces to the network. In the case of MMOGs, a server-client structure is assumed because every commercial MMOG uses this network structure. Generally, a P2P solution is also possible, although security issues (MMOGs often have strict authentification methods) clearly do not promote this structure.

Another aspect is the middleware application's scalability. Most current MMOGs feature between 1,000 and 15,000 players simultaneously. With a growing number of users, the response time increases (which has negative effects on the players' performance). Since the MMOG server is the network architecture's bottleneck, the middleware needs to support rapid communication between clients and the server. The size of the packets should be as small as possible to reduce the effect of an increasing number of players. Furthermore, the degree of communication between the middleware and the server must be minimized as well. Both TCP and UDP are

used as protocols in the middleware since they are the standard for current MMOGs [Game].

Security: The middleware's security is vital from a developer's perspective. The player and object tracking mechanism especially contains important data about the in-game world. This data needs to be safeguarded because a player could obtain an advantage over others by knowing the precise location of objects (such as treasures). Therefore, the methods for players are not allowed to access this secure information.

Furthermore, the middleware needs to be robust with regard to the large number of players. If the middleware design does not scale with a large number of players, it could shut down during the peak times. A negative example of a shutdown is the server structure of Diablo 2 [Diab]. Players could shut down the main game server by sending too many messages in a short period of time. This brute force technique has led to the misbehaviour of constantly rebooting the main server to obtain a game advantage. As an example: instead of permanently losing a character (due to demise), the game server was attacked to force a roll-back. Such behaviour needs to be anticipated for the middleware as well. It must be robust against a high number of incoming packages. One possible technique for the middleware could be the regulation of player spamming behaviour (see Section 6.3.2).

6.1.4 Constraints

This section covers constraints for the 4MOG middleware. When analyzing the problem, limitations need to be pinpointed. The more generalized the task for the middleware is, the more difficult it is to find a solution for the overriding problem. Therefore, the 4MOG middleware will focus on certain aspects which have been implemented in detail:

Client restrictions: The 4MOG middleware is designed for a server-client structure. The clients are assumed to be high-end personal computers, which are capable of running current graphic engines. It is theoretically also possible to use the middleware on resource-constrained devices (mobile phones, handhelds, etc.) as well. But the problematic field in the mobile environment is completely different. In particular, the performance of handhelds is not high enough to support a current game engine. Furthermore, communication with the clients poses additional

problems (GPRS has a high latency, WLAN and Bluetooth only have a limited range).

Assuming that personal computers are up-to-date, local calculation can be carried out by the clients. This reduces the server's load because many calculations can be performed by clients themselves. Especially rendering (illustrating the game world) and movement prediction can be calculated by the client, which reduces network traffic. As an example: the movement of other players can be sent via a simple message from the server to the client, containing only the new coordinates. The local rendering and the prediction for further movement can be carried out by the client itself.

Network restrictions: A decision on the middleware's design (server-client) is based on the fact that the server-client structure is the common design among current MMOGs. A hybrid or peer-to-peer network would reduce the server's bottleneck position, but the downside is a security problem and the problem of storing persistent game data. The security problem includes the fact that peer-to-peer networks offer more options for the players to cheat within the game. Furthermore, a central login method is missing. As many MMOGs have a monthly fee, they also need to ensure that playing for free is not possible.

The underlying protocol for middleware is TCP and UDP since both protocols are frequently used in the current MMOG design. The main difference between them is that TCP is connection-based. It uses the acknowledgement function to ensure that no packages are lost during the transaction. UDP is faster because it does not use explicit server-client connections. Depending on the game scenario, the developer has to decide whether TCP or UDP should be used. As an example: the coordinates of player movement do not necessarily need to be sent to different clients individually. Broadcasting for all clients within the AOI (area of interest) in regular intervals reduces overall network traffic. UDP would be preferred for this scenario since a player's movement information will be updated frequently. Even if a single packet is lost during the transaction, it is possible to smooth the in-game movement after receiving the next location update (this technique is called dead reckoning).

Middleware limitation: The middleware is limited to a core set of functions (administration, send/receive, game sessions, communication, monitoring), which

support the game developer by offering interfaces for the game's technical implementation. These methods could also be used to integrate more functionality. As an example: the gathered data of player monitoring could be analyzed and used to integrate more individual content in the games. In the current 4MOG middleware application, additional methods have not been included for two reasons:

1) The integration of too many different functions in the middleware reduces its credibility. The existing functions cover the described problematic field; they support the developer in the game design phase. More functionality would significantly increase the programming effort and would not support all types of MMOGs. As an example: an additional ladder system (in-game playing ranking) supports the design of FPS and RTS games although it is not required in the design of MMORPGs. Therefore, it is necessary to focus on the implementation of the core functions first.

2) Before the middleware can be expanded with further features, it is necessary to analyze how the developers actually use the 4MOG application. This can be done by receiving feedback (from the developers) about the middleware's current version. With a deeper insight into actual usage, a follow-up application with further features can be designed.

6.1.5 System Model

This section covers the illustration of the 4MOG functionality with a usage case diagram. In the design phase, it is important to clarify the methods for all participants. In the case of the 4MOG middleware, game developers are the target group. The software offers five different types of functions: administration, send/receive, game session, communication and monitoring functions.

Figure 6.2 shows the core set functions and their connection to the game developer. Since the "GameDeveloper" is the central actor, he/she is supported by the middleware. As an example: the administrative functions include methods to handle in-game objects (like player characters, NPCs, etc.). If a player picks up a new item, the middleware function to update the inventory can be used to store the change.

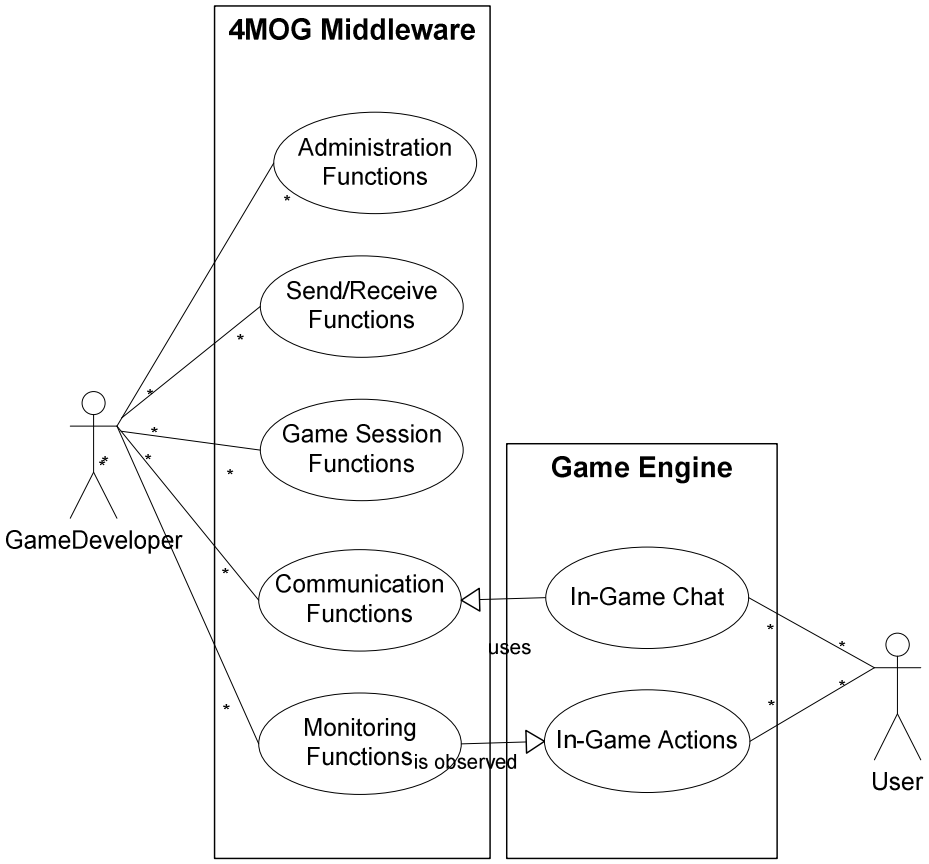


Figure 6.2. Usage Case diagram of the 4MOG middleware

The user does not use the middleware's interfaces directly. The objective of the 4MOG design is to make the game design easier. The middleware can be regarded as a developmental tool since the interfaces offer an easy way to integrate the rudimentary technical functions of the MMOG. It is necessary to clarify that the developer also needs to connect the given interfaces of the middleware to the game engine (the game engine is built on top of the middleware).

As an example: if the user wants to chat with in-game friends, he/she uses the in-game communication methods (like sending a text to a fellow player). This means that the interaction from a user's perspective is limited to the game engine. The game engine on the other hand uses the middleware's interfaces for this communication, in particular, the game engine requests a function of the middleware with the parameter of the target player and the message string. The middleware communicates with the server (in order to receive information as to whether the target player is online) and forwards the message to that player.

6.2 Design

The design section must be differentiated from the requirements section. This part of the middleware design includes the concrete decision about the structure of the middleware and its specific implementation. Before one can look at the modularization, a few fundamental implementation decisions need to be taken:

Programming language: The major aspects for a decision on the programming language are compatibility and performance. The middleware needs to be compatible with the current computer game design. Since the constraints include a focus on the current MMOG design for personal computers, one needs to look at the programming languages deployed. Most of the high speed games (FPS, RTS) are implemented by using C++, whereas Java plays an important role for simulations and MMORPGs. Platform independence is also important; several MMOGs (like World of Warcraft) also support the Mac OS. Platform independence would further support Java as the programming language.

The aspect of performance also influences the decision. C++ supports latency sensitive games better than Java because the game applications run faster. The overhead of the Java Virtual Machine is a problem in the game design, especially for resource-constrained devices. Therefore, the performance aspect clearly promotes C++ as the programming language.

When considering the constraints, the assumed hardware specification of the client includes an up-to-date personal computer. Furthermore, most of the MMOGs released are RPGs (which are not as latency sensitive as FPSs and RTSs); therefore compatibility is more important than performance. With regard to the trade-off between performance and compatibility, the 4MOG middleware uses Java as the programming language.

RPC vs. MOM: A further factor besides the programming language decision is the communication design: the central aspect of this decision is whether the middleware should be implemented as a MOM (message oriented middleware) or as a RPC (remote procedure calls) version. The RPC implementation of the middleware is the classic implementation strategy for client-server systems. Generally, a RPC is

initiated by the client, which sends a request message to the server. This request uses one of the interfaces provided by the server with specific parameters. As an example: the client uses a *send(<playername>, <message>)* method on the server. The advantage of this technique is the clear separation between client and server, which enables the client to only call up pre-defined methods on the server. Results are calculated by the server and the client receives an answer afterwards. In the example, this answer could include a simple acknowledgement or an error message if the receiver is offline.

The MOM design includes the usage of messages to communicate between server and client. A queue system stores the messages from the clients on the server, so they can be executed in proper order. The advantage of MOM implementation is the ability to store and transform the messages. Since the size of each message is small, their storage is easy and a logging mechanism is possible.

Two main aspects promote the MOM implementation over the RPC version. The smaller package size of the MOM implementation reduces overall network traffic. This is important for MMOGs since the scalability (number of players that participate in the virtual world) affects all types of games in this genre.

Furthermore, the MOM system offers greater flexibility for the clients [Hsia 05]. As an example: after sending a message to the server, the client can use forecast algorithms that include the most probable answers. In case of movement in the games, a client could calculate the movement of opponents while waiting at their exact position (answer from the server). Once the answer is received, the difference can be reduced by synchronizing the local calculation and the exact data from the server. Both of the arguments clearly promote the MOM implementation over the RPC version.

6.2.1 Modularization

Modularization describes the middleware's general structure. This section contains an overview of the client and the server side of the 4MOG application.

Since the server-client structure is implemented as a MOM, both sides use a MessageListener for communication. This listener is an implementation of the

standard observer pattern. In order to receive an overview of the middleware's modularization, Figure 6.3 introduces the complete UML diagram of the server.

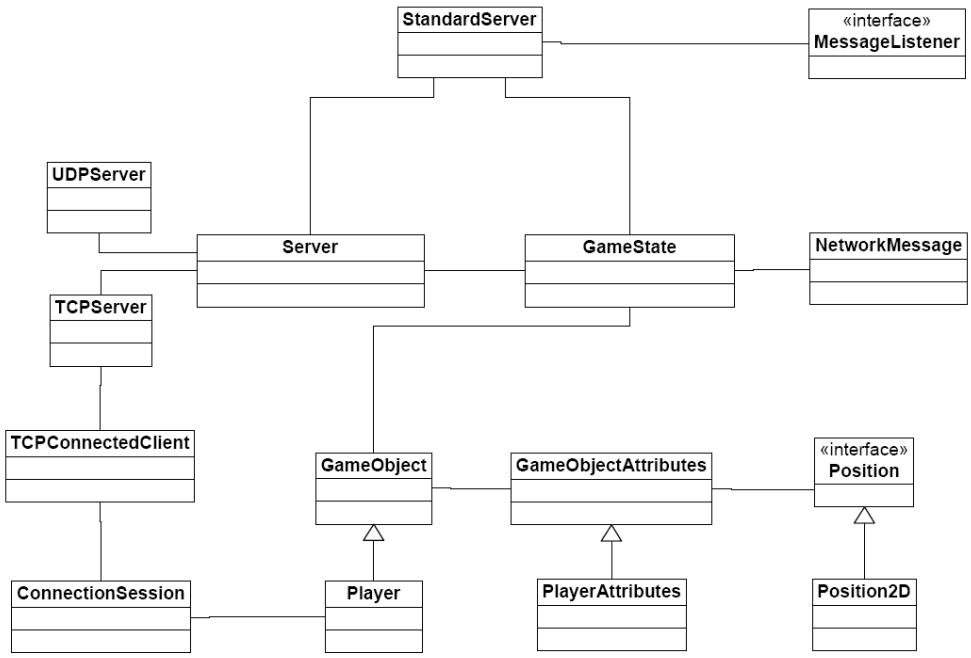


Figure 6.3. UML class diagram of the 4MOG server side

The server uses the MessageListener to communicate with the clients and offers interfaces for functions as shown in the UML diagram. Two main classes are in the central position of the UML diagram. The GameState contains all important information about the player location, objects and the status of the virtual world. This class offers functions for both clients and the server to receive information about the current game world. As an example: the GameState contains information about the position of all other players, and the server can receive specific information if needed (like the position of all team mates). This information is stored in the GameObject class; player characters as well as attributes are also stored. The separate storage of object attributes enables parts of the game object to be updated without submitting the complete object whenever it changes. As an example: if a player receives a negative effect which reduces a certain attribute, then it is possible to submit the temporary decrease to the database without having to update the entire character immediately.

The network communication of the server side uses UDP and TCP interfaces. In the case of TCP, the server administrates all the connections to the clients. This includes the distribution of ports and the clean up after a connection session is closed. The exact functionality of each class is described in the next section.

The client version of the middleware has a similar structure because the client also has interfaces to the GameState, the network and the MessageListener. The functionality of the client class differs significantly. Figure 6.4 gives an overview of the structure of the client side of the 4MOG middleware.

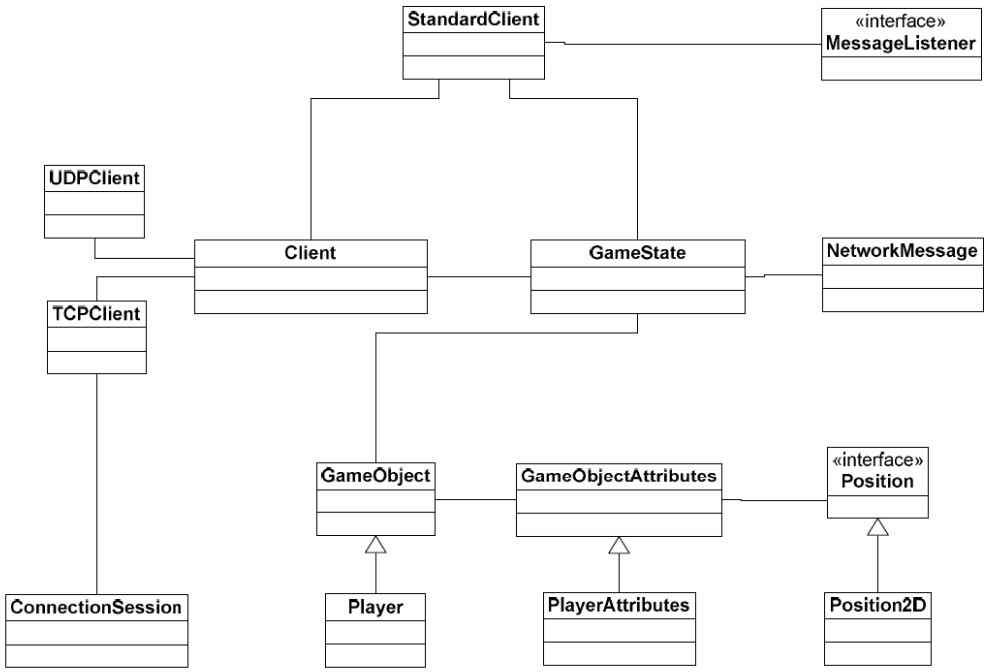


Figure 6.4. UML class diagram of the 4MOG client side

The client shows similarities compared to the server side of the middleware. One common aspect is the MessageListener, which is needed for communication between server and client. The client uses the MessageListener to send status updates like movement to the server. The transfer of data through the network is implemented with the UDP and TCP client manager.

The main difference between the server and the client side of the 4MOG middleware is the access at the GameState. The server has full access to all information (including all objects and players), whereas the client only has a limited number of

interfaces at the GameState. These functions contain non-critical information like the in-game time or information about the own character. As an example: the client could use the "/time" function to receive information about the in-game time because the virtual world's time could differ from that of the real world.

6.2.2 Functionality

This chapter contains information about the functionality of all classes in the middleware. Each of them is described in detail:

GameState: The GameState class contains information about the game world, including the location and status of all objects. Figure 6.5 features an UML diagram of the GameState class; it is shown that both players and object have certain common attributes.

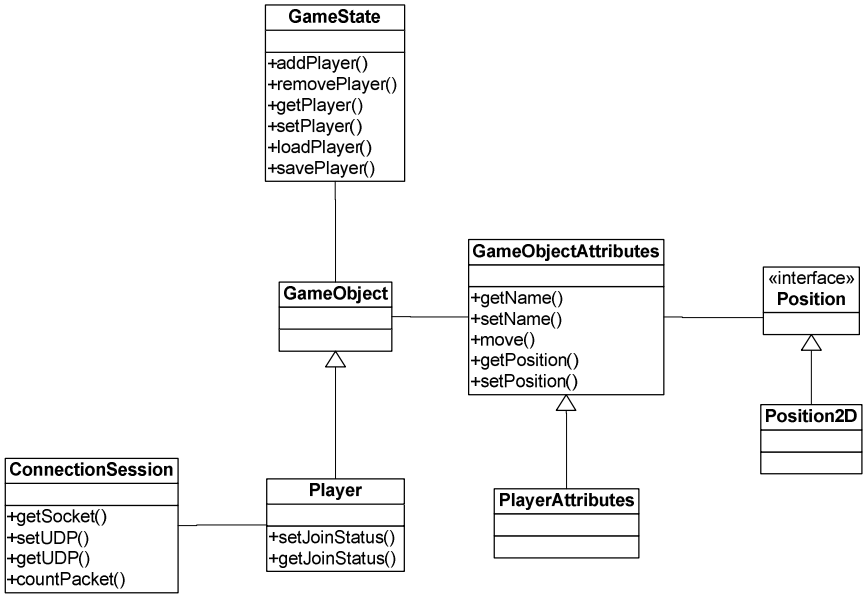


Figure 6.5. UML diagram of the Gamestate class

Both the client and the server can communicate with the GameState class (refer to figures 6.3 and 6.4). The main differences in communication with the GameState are the available interfaces. The server can access all information about the game world, whereas a client only has access to general information (like the in-game time). This security design is necessary in order to prevent cheating. If a player would have

access to all information about other players and objects, he/she would obtain an advantage compared to the others.

The functions of the GameState class contain all necessary methods to administrate game objects (including players). A player can join, leave or move in the game world. The relevant changes are carried out by the GameState class.

Client and Server: The client class is an implementation of the network functionality for each of the clients, whereas the server class represents the server's network functionality. Both of them communicate with the GameState. Before a client can join the game, it is necessary to establish a connection. The middleware uses a ConnectionSession for TCP connections between the client and the server. This class creates a logical connection between the server and the client. Since UDP message transfer is also supported, a TCP connection is required to establish the initial connection between the client and the server.

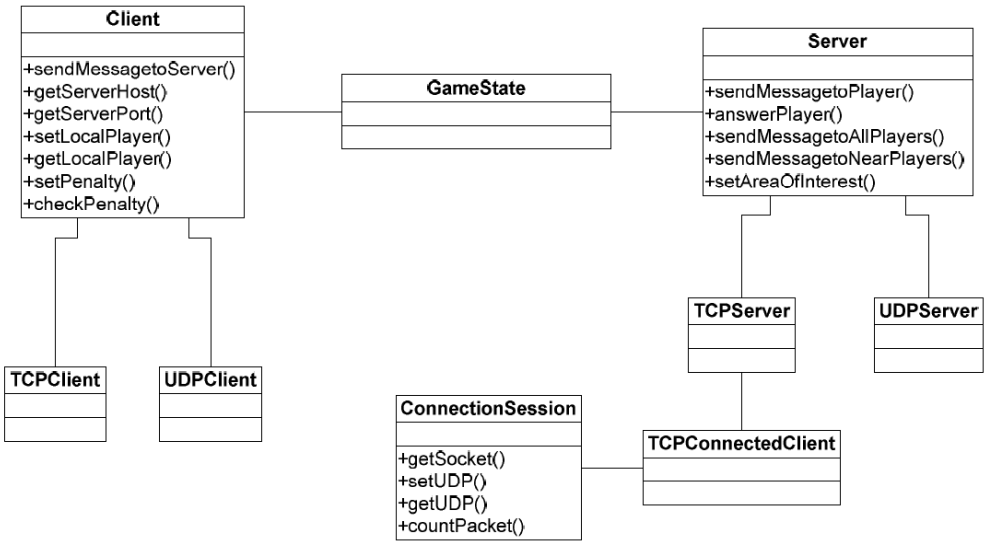


Figure 6.6. UML diagram of the server and client class

The middleware implementation of the connection is realized with a multi-threading approach. Each new client that connects to the server receives an individual thread that handles the connection. Figure 6.6 illustrates the connection of the client and the server to the GameState. The client uses a separate thread for the TCP and the UDP connection to the server (TCPClient and UDPClient), which awaits messages from

the server. On the other hand, the UDPServer also monitors incoming messages from the clients at a specific port.

The initial TCP connection is realized through the ConnectionSession class. As soon as a new player joins the game, the server creates a TCPConnectedClient thread that handles the socket and initializes the connection. Furthermore, the player registers at the GameState class. If the registration is successful, the player then receives a unique ID and the client creates its network threads. If the player is rejected by the GameState (banned or the maximum number of players is reached), then the ConnectionSession is terminated.

StandardClient and StandardServer: In contrast to the pure network functions of the server and the client class, the remaining functionality for the server and the client are implemented in the StandardClient and StandardServer class. In the middleware architecture, the StandardClient uses the client as an interface for the network functionalities. In order to illustrate the differences in the classes, figure 6.7 gives an overview of the StandardClient implementation of the middleware.

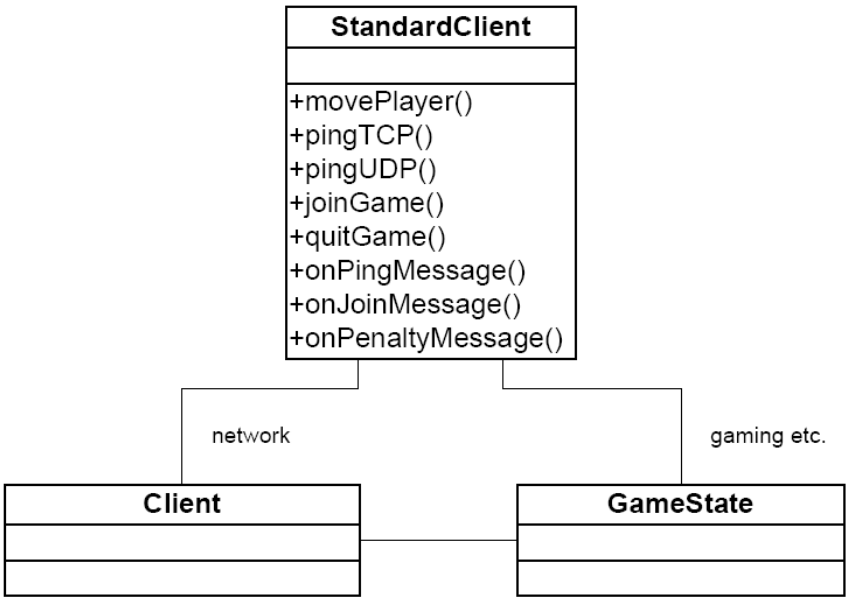


Figure 6.7. UML diagram of the StandardClient class

The StandardClient represents the middleware application from the perspective of a client and offers the standard functions to join, quit or move in the game world. As

shown in figure 6.7, the network aspect of the client is implemented by the client class, whereas communication with the game is carried out by the GameState class. Furthermore, the game application can define customized messages that implement additional functions for the client. This option allows the designer to customize the client for the corresponding MMOG.

The design of the StandardServer is analogous; figure 6.8 illustrates the UML diagram:

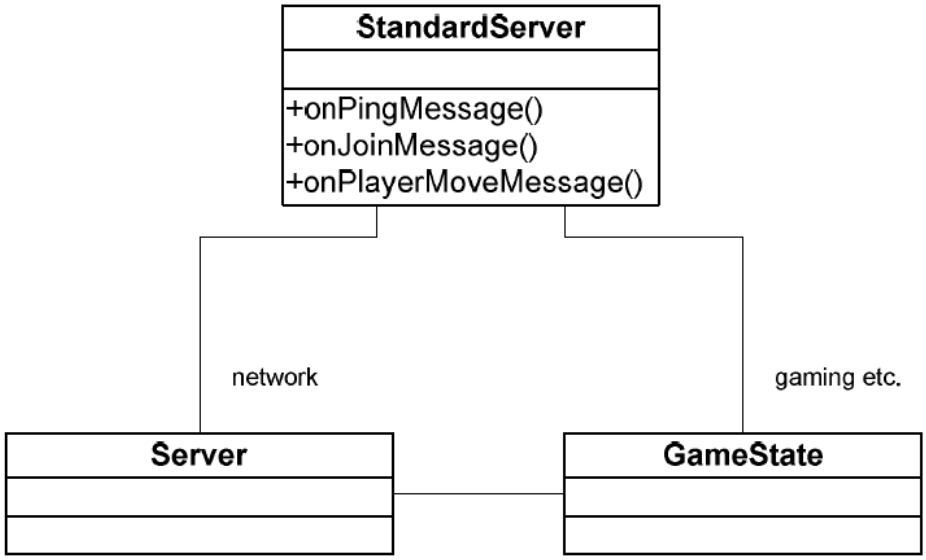


Figure 6.8. UML diagram of the StandardServer class

6.3 Validation

In this section the middleware application is validated in order to ensure credibility. The validation process is one of the most important aspects in software creation because an innovative application's credibility relies on its results during this process.

Therefore, the application is compared with a similar middleware called GASP [Pell 2005]. The functionality and constraints of both applications are evaluated. Since the constraints have a significant influence on the usage of the middleware (because they describe the limitations), one first needs to analyze the two systems individually. As similarities exist, the implementation principles are discussed in particular.

The comparison of both middleware applications is also discussed with regard to specific scenarios. Both applications have a slightly different usage case scenario, which means that it is difficult to illustrate a common performance test. The comparison will therefore be based on an argumentation. Nevertheless, it is still important to evaluate the 4MOG middleware's functionality:

Therefore, the creation of a testbed is described, which is used to further validate the 4MOG middleware application. The testbed focuses on one of the core set functions of the middleware: player tracking. In order to illustrate a potential scenario for the middleware, a certain spamming behaviour is assumed. Spamming describes the player's behaviour to continuously send a large number of packets to the server. In this testbed, the 4MOG middleware's functionality is used to minimize the effects of such player misbehaviour.

6.3.1 Comparison with other Middleware

This section includes a comparison of the 4MOG middleware with other middleware applications. Each of the following subsections includes a short introduction to its middleware. Subsequently, the main differences between the respective middleware and 4MOG are introduced and finally common aspects are highlighted. The comparison focuses on clarifying major differences between 4MOG and related software approaches.

6.3.1.1 Comparison with Zoidcom

Introduction: Zoidcom was originally a by-product of the multiplayer space shooter "Operation Black Sun". After attempting several approaches to implement efficient networking into the game engine, the 4th iteration of the net code became a stand-alone library. With this library, Zoidcom aims to provide most of the current space and FPS games with network functionality.

Differences: The developmental strategy of Zoidcom is based on the generalization of an already running implementation. Thus focus of the library relies on the support of space and FPS games. Zoidcom itself is designed as a library, whereas 4MOG is a middleware. The main difference is that 4MOG already implemented the methods and offers interfaces. On the other hand, Zoidcom offers a well documented library with a set of functions to manage the games' network aspect. These functions include

bit streams; a technique to use a UDP-based protocol with advanced functionality. With the additional functions, the bit stream protocol measures the ping time between clients and servers regularly in order to improve/decrease the data traffic. Furthermore, an automatically bundling of very small data into packets is used to even the loads on each packet sent.

Zoidcom is based on a very protocol driven approach, whereas 4MOG uses a set of core functions which are important for all MMOGs. The game focus between both projects is also differs slightly because Zoidcom offers functionality for FPS and space simulations. On the other hand, 4MOG offers core function support for MMOGs.

Similarities: Both projects have certain aspects in common. The concept of player tracking is available in the Zoidcom library as well as in the 4MOG middleware. In both cases, a client side prediction (dead reckoning) is used, but data for movement and error correction is double-checked with the server at all times to prevent illegal player activities.

6.3.1.2 Comparison with ENet

Introduction: ENet evolved as a UDP networking layer for the multiplayer FPS game “Cube”. ENet's purpose is to provide a thin and robust network communication layer on top of UDP. Similar to Zoidcom, the evolution of ENet is based on an implementation of a FPS game.

Differences: ENet uses UDP as the protocol and also provides a simple connection interface over which communication is possible with a foreign host. As long as the data stream is active, ENet monitors the network conditions from one host to another, including a measurement of round-trip times. In contrast, 4MOG uses a TCP connection-based approach for client management. Especially in MMOGs with a large client management overhead, a pure UDP-based approach can lead to inconsistencies and disconnects due to timeouts or lost packages.

Again, the number of packages plays an important role in the design of ENet. Rather than sending only a single byte stream that complicates the delineation of packets, ENet uses connections as multiple, properly sequenced packet streams. To keep the order consistent, ENet provides sequencing for all packets. This approach is again

focused on the lower protocol level and aims to reduce the overall amount of data sent from one client to another. The game type focus of both applications differs as well: for normal FPS games which are very latency sensitive, this approach supports the games better than the 4MOG middleware. However, in the MMOG area with a growing number of players, the player tracking and TCP connections to each of the clients provides a stable environment which is required for the games.

Similarities: Both applications have one major aspect in common. ENet and 4MOG both provide a throttling mechanism to reduce the overall bandwidth required. For ENet, this dynamic throttle responds to deviations from normal network connections to rectify various types of network congestion. Latency spikes in FPS are therefore reduced, making them more playable in situations where a lot of in-game action occurs. 4MOG on the other hand aims to keep the long-term bandwidth relatively constant in order to prevent DOS attacks. Thus, an individual penalty system for the clients is used in case the number of their messages to the server exceeds the limitations.

6.3.1.3 Comparison with ZIG

Introduction: ZIG is designed as a C++ framework with interfaces to build a multiplayer client-server game on top of it. The framework itself contains two main classes: “zigclient_c” and “zigserver_c” – both of them can be extended in the game to create the game server and the client class. Therefore, Zig is not a middleware or library from a classical point of view. Rather, it is a class-oriented guideline for the implementation of a client-server gaming application.

Differences: The Zig approach uses the UDP protocol with additional functionality, including an option to make the packet transfer reliable (with checksums). In contrast to the Zig approach, 4MOG uses TCP for important data to provide a stable game environment and uses UDP for broadcasting game data with less priority.

The network layer, which Zig aims to improve, is mainly the UDP network protocol. The additional functions are a policy management for each of the UDP streams that allow priorities to be set for more important data. Furthermore, a data bundling ensures packets of equal size. In contrast, the 4MOG middleware provides a core set of functions which is required for MMOGs. The 4MOG middleware approaches the game support by offering interfaces to the game engine. In comparison, Zig only

contains two classes that act as a guideline for the game design. the scope of the two projects differs as well because Zig focuses on improving the network-related aspects (such as latency or bandwidth) without providing additional functionality for the in-game support. Especially for a higher number of participants, the focus on UDP as the network protocol leads to a disadvantage due to possible inconsistencies and disconnects because of packet loss.

Similarities: Both applications have the logging mechanism in common. Zig creates an overall log file which stores all network transfer-related information. Again this approach is very detailed, but with a higher number of participants, the logging mechanism needs to be adjusted in order to prevent a data overhead. Generally, any movement and action within the game world can be tracked by using the Zig log file. A similar approach is used by the 4MOG middleware; each client has its own log file which stores the client-related data like connection time, movement and in-game activities. Logging mechanisms in general provide an efficient mechanism for game producers to review in-game activity and to identify potential player misbehaviour.

6.3.1.4 Comparison with Raknet

Introduction: Raknet is a cross-platform C++ game networking engine. This middleware offers interfaces to game engines; the network functionality is implemented within the middleware. Raknet uses an advanced networking API that provides services based on the Windows systems Winsock. Any other application that also uses Winsock can communicate with the middleware.

Differences: Unlike 4MOG, Raknet does not focus on a single game type or game function. In contrast, the approach focuses on additional functionality that can be implemented for any given multiplayer game (like an auto-patcher or in-game chat), although some functionalities offered may not fit with the game type requirements. The auto-patcher option, for instance, is not necessarily required in a SG or a RTS because the number of patches for these genres is relatively low compared to those in the MMOG sector.

Raknet also offers an additional voice communication which is implemented in UDP. In contrast, 4MOG focuses on the core functionality since many users do not want an integrated voice chat (compare to Counter-Strike) and because it generates additional overhead. The core functions of 4MOG only focus on the functionality required the

most for MMOGs like user management. Another major difference is the internal communication of the middleware approaches. Raknet uses a RPC (remote procedure call) design, whereas 4MOG uses a MOM version (message oriented middleware). The differences in the two design methods are described above.

Similarities: Both of the approaches use a middleware instead of abstract classes (see Zig) or libraries (see Zoidcom). This decision has several implications, including a hidden internal design (black-box principle) as well as predefined interfaces for the game engine. Also both middleware approaches use TCP for the initial client management and UDP for broadcasting support. Due to their interfaces to the game, both 4MOG and Raknet show similarities in providing additional direct support for the game engine.

6.3.1.5 Comparison with GASP

Introduction: The GASP middleware project [Pell 2005] aims to improve current multiplayer games with a similar approach. As an open source project, it endeavors to develop a middleware application for mobile multiplayer games, although the focus is slightly different.

First of all, the implementation mechanism of the GASP middleware creates sessions for each of the users. This enables the player to choose his/her game at the beginning by requesting necessary data from the server. After receiving an answer, an individual session is created for the user with a session number and further information about the game. From a technical point of view, the main difference between the two concepts is the generalization of a completely game independent structure of the GASP middleware on the one hand and the game-specific middleware framework of the 4MOG approach on the other hand.

Differences: The GASP middleware offers interfaces for the players like the opportunity to choose a game. It also supports multiple games, whereas the 4MOG middleware focuses on the MMOG sector. The approach of the 4MOG application offers interfaces to the game engine and the network which the developer can use to reduce the technical implementation efforts of the MMOGSs.

As a result of the decision, the GASP middleware needs a central platform representation to support the different games, whereas the 4MOG middleware

focuses on a lean implementation strategy that can be customized for each of the games. The GASP middleware can be created as a stand-alone application. The 4MOG middleware aims to be integrated into the game, thus becoming a part of it. Generally, it is possible to integrate sub-classes for further common demands like the integration of instance management (for instanced MMOGs), but the final implementation for 4MOG middleware will be customized for any of the applications. This will therefore reduce the overhead of unnecessary functionalities. An example for this are the core set functions, which can be regarded as the "least common denominator" of technical functionality for MMOGs.

As a combination of both methods, one could consider a modular implementation structure, which categorizes each of the features into a specific sub-module. By distributing the game-supporting mechanics to these modules, the publisher could decide whether or not a feature is required for the game and customize the final middleware application.

Similarities: There are also similarities in the design of the both middleware applications. The GASP middleware is also implemented as a message oriented application. This decision reflects the importance to keep the network traffic as small as possible to support a large number of players.

As a conclusion: the main difference between the two applications is their functionality. While the 4MOG middleware aims to solve a specific problem (supporting the developers), the GASP middleware has a much larger scope. Nevertheless, the two applications have the MOM implementation in common in order to reduce network overhead.

The comparison of both middleware applications does not promote one system over the other; their usage clearly depends on the scenario. For example: the typical scenario that promotes 4MOG middleware over GASP middleware is the design of a specific MMOG in a smaller game company. That is because the 4MOG middleware offers explicit functionalities for the technical aspects of game creation. For this scenario, the 4MOG middleware would therefore be superior since the GASP functionality aims to support different types of games or even a game collection, which is not needed for a single MMOG. An interface for multiple games is not required if only one game needs to be created.

In contrast to this, other scenarios such as the design of an online game collection would clearly promote the GASP middleware because it offers support for multiple game types and focuses on the interfaces for a game selection. The 4MOG middleware offers interfaces for the core functions (such as connection, communication, etc.). However, if the usage case aims to create a bundle of casual arcade games, a chat might not be required. Therefore, the GASP middleware suits this usage case better.

6.3.1.6 Overview of the comparison

This section gives an overview of the different middleware and library approaches. It summarizes the main similarities and differences in a table. Some aspects of these solutions, however, cannot be compared like Zoidcom for example – because the advantage of the additional functionality strictly depends on the situation. For some games, the middleware might offer considerable support because all of the functions are mandatory, wherease other applications might not benefit from it at all.

Table 6.1. Overview of the different middleware and library approaches to support computer games' network functionalities

Name	Game type	Protocol	Approach	Functions
Zoidcom	FPS& space	UDP	Library	Player tracking, UDP enhancement
ENet	FPS	UDP	Library	Streams, UDP enhancements, throttling
ZIG	All	UDP	2 classes	Logging system, UDP enhancements
Raknet	All	Winsock & UDP	Middleware	Additional functionality, auto-patcher, login, in-game chat
GASP	All	TCP/UDP	Middleware	Additional functions, game lobby, user management
4MOG	MMOG	TCP/UDP	Middleware	Core functions for development, throttling mechanism, logging

6.3.2 Testbed

GASP, Zoidcom, ENet, Raknet, ZIG and the 4MOG middleware cannot be compared directly with regard to technology since all of these applications have a different focus.

In contrast, this section contains a testbed for the 4MOG middleware application. It is vital to pinpoint that the MMOG scenario does not have the same problems as mobile gaming. Latency is also an important issue, however, especially with a growing number of players, scalability is still the most important issue for MMOGs. With only a single server (which is the bottleneck in the network design), one

important aspect for MMOGs is the amount of network traffic which is sent/received by the server.

As mentioned earlier, one of the options to "cheat" in MMOGs is to abuse the number of packages that one player can send. Therefore, the middleware needs to be robust against client spamming behaviour.

The testbed itself is independent from the mobile environment and uses a local area network for the connection. Nine different computers are used in the local area network, each of them with following configuration:

> *Windows XP workstation, Pentium 3GHz processor, 1GB RAM and a 100Mbit/s network adapter.*

Lab1 is used to run the server version of the 4MOG middleware, whereas Lab2 – Lab9 run 10 4MOG clients each. Therefore, the final testbed contains 80 different clients that are connected to the server, as shown in Figure 6.5.

The server simulates a two-dimensional game world with a size of 200x200 fields. Each of the given 80 clients is randomly placed in one of them. During the test scenario, every player moves randomly with an individual speed. After a player has performed he/her movement, the other players in range will receive a notification about the new position. Moreover, the clients are divided into four different player types with their own spamming behaviour.

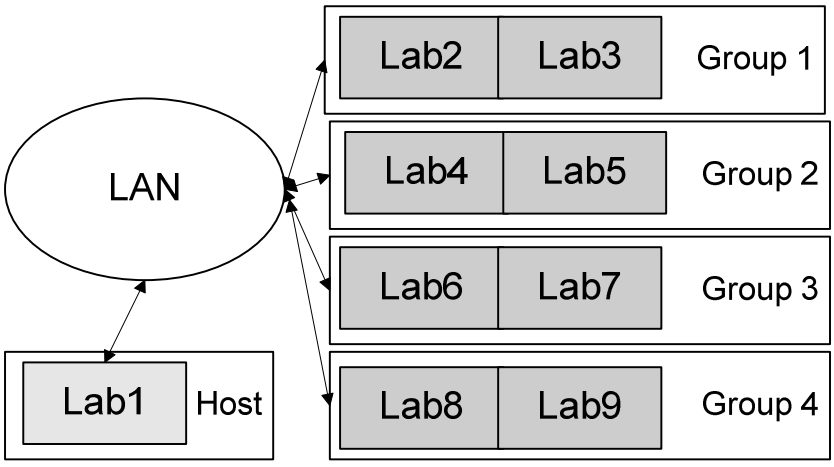


Figure 6.5. Illustration of the testbed: Group1 is configured to be the very strong spamming group, group 2 sends a high number of packages, group 3 is a normal number and group 4 reveals low spamming behaviour

Additionally, a percentage value is responsible for the client's probability to randomly change its behaviour, which entails either a stronger or less stronger spam. Although the underlying player model is kept relatively simple, both reactions can occur in actual reality. Either a player reduces the overall number of messages and slows down or he/she keeps on spamming in ??defiance??.

The maximum number of packets is capped to 20 per second: the first computer of each group (Lab2, 4, 6, 8) always contains the solid players who rarely change their behaviour. Therefore, the rate to adopt a new spamming behaviour (randomly between 1 and 20 packets per second) is set to 33%, which signifies a 67% probability of maintaining the old rate. The second computer of each group (Lab 3, 5, 7, 9) is designed to adopt new behaviours with a 66% probability. In fact, after each cycle (for the testbed this is 1 minute) the spamming behaviour is recalculated, hence regular spammers tend to fall back to their initial rates. The length of the cycles can vary, for this testbed 1minute gives the clients enough time to adjust their behaviour before falling back into the initial spamming strategy. The results of the testbed are discussed in the next chapter.

Table 6.2. Group setup of the testbed

Group Number (with related clients)	Spam behaviour (package per second)
Group 1 (Lab2,3)	Very high (12 per sec)
Group 2 (Lab4,5)	High (9 per sec)
Group 3 (Lab6,7)	Average (6 per sec)
Group 4 (Lab8,9)	Low (3 per sec)

6.4 Summary

This chapter summarizes the advantages and disadvantages of using a middleware application. The section covers three main aspects of middleware creation: the definition of requirements, the implementation and the validation. Each of them is described individually. Furthermore, the UML diagram of the core classes is explained as well as the resulting testbed with 80 clients and different spamming strategies. For further information, Chapter 7 illustrates the results in-depth and compares them with related attempts.

7. Experimental Results

"By far the best proof is experience", Sir Francis Bacon.

This chapter analyzes the statistical results of the player behaviour surveys in detail. With regard to other surveys, the results and statistical correlations are evaluated. Likewise, the most important findings from the virtual fragmentation survey are discussed. Furthermore, the testbed results for the mobile lobby combined with the results of the follow-up survey from chapter 5 are analyzed. Moreover, the 4MOG testbed of the middleware prototype is evaluated and compared with similar software approaches. Finally, the contribution of each of the three main approaches is evaluated by comparing them to the status of the problematic fields: player behaviour, mobile gaming and massive multiplayer gaming.

7.1 Statistical Analysis of Player Behaviour

The large numbers of replies from the online questionnaire exceed the expectations. Overall, more than 30,000 users participated in the survey; half of them answered the questionnaire "distribution of online gaming behaviour". After eliminating the incomplete data sets, the total number of answers equals 7,100 records which serve as the data pool. However, the number of answers for each of the game types varies. It was determined that the set contains more than 4,500 users from the MMORPG scene, around 1,300 users from the FPS section, 1,100 RTS users and only 197 SG players. Another important aspect is the distribution type of the data pool; therefore the Jarque-Bera is used in order to prove the deviation from normality.

Before taking a closer look at the general results, one should first evaluate the game types. The survey is individualized for each of the four main game types: FPS, RTS, RPG and SG. Each of the sections in the survey contains a common set of questions about the user activities focused on the game (such as forum activities, web pages, communication tools and in-game videos). Furthermore, all of them include game type-related questions about expectations and motivation. For example: an analysis of the computer gaming priority in the users' daily routines.

The idea behind the survey is to analyze the influencing factors for the time that players spend on their game(s). Furthermore, the influence of demographic values (age, gender, leisure time, etc.) on the overall game time needs to be analyzed. Beyond that, each section contains an analysis of the different competition types. From previous interviews/surveys [Fritsch 2006.1] it has been determined that users tend to answer more inaccurately with regard to higher numbers. For instance, it is hard to estimate the exact number of hours that someone worked last year.

Thus, the number of hours per day for game time and leisure time is capped to a maximum of 8+ hours. The set of users that chose the highest possible answer for game time includes all users who play eight or more hours a day. This classification has been performed in the survey for two main reasons: firstly, the users are not motivated to over-estimate their own game time. In a pre-survey without the limitation, many users stated that they play 20 to 24 hours a day (each day). For rare exceptions such a time might hold true (examples of hardcore gaming are often found in the MMOG sector), but in many cases, users had difficulties in correctly estimating their "average" game time per day.

The second important reason for the classification of 8+ hours is the purpose of the survey. The main objective is to evaluate whether or not a player can be classified as a "hardcore" user. Since the definition of "hardcore" gaming depends on the viewpoint, the borderline between casual and hardcore gaming might vary. With regard to the average working time for most people (which is about 40 hours per week in Europe), this survey defines that a hardcore player invests at least that much time in gaming. The maximum possible game time per week is 56 hours (7 days * 8 hours), which already exceeds the average working time by far.

However, one cannot assume that this maximum number accurately reflects the overall limit of game time per week. In fact, the data pool shows that a large percentage of players reach the 56 hour limit, which means they play at least 56 hours a week. In order to analyze the real maximum game time, individual interviews were held with players from top clans of the FPS and RTS sectors.

To understand the gaming behaviour, one must observe the demographic values first. In our observation, the survey focuses on current online gamers from the FPS, RTS, RPG and SG sectors only. Therefore, as indicated in Section 4, the overall

demographic values differ slightly from those in [Thee 2007]. The data pool reveals certain similarities such as the marital status and gender. A large majority of the players in our sample are single; FPS: 93.0%, RTS: 97.8% and RPG: 89.9%. The only exception in this context is the sports games sector; only 68.4% of the players in the sample are single. The other possible options are married, divorced and widowed; all of them show a very small number of matches (except married in the sports game sector). Furthermore, the gender distribution also shows similarities. The rate of males in the sample according to game types is: FPS: 97.8%, SG 95.3%, RTS: 99.2% and RPG: 92.3%. Our sample accurately reflects the distribution of the online gamer population. Especially in online games, the ratio of female players (according to [Bliz]) is relatively low. Exceptions occur (like "The Sims" [Sims]); however, for the majority of the current best-selling online games from the FPS, RTS, RPG and SG sector, nearly all players are male.

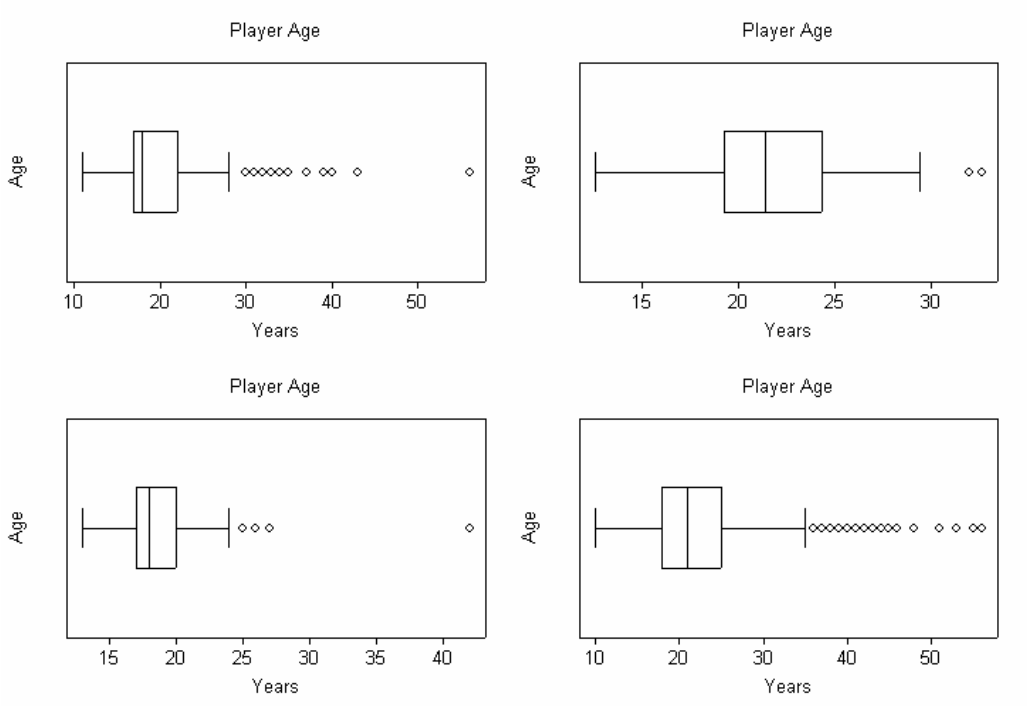


Figure 7.1. Age distribution among the different game types: From upper left to bottom right: FPS, SG, RTS and RPG

There are also differences in demographic values between the game types. The average age differs significantly among the four types. Figure 7.1 shows four

different box plots, where the leftmost line indicates the x0 quantile, a representation of the lowest value. Analogously, the rightmost line is the x1 quantile, a representation of the highest value of the data set (circles on the right-hand side are single values that can statically be regarded as escapees). An analysis of the data pool shows more or less consistent values of x0, the youngest participants are always around 10 to 12 years old. An explanation for that is either the value of the technical equipment and/or the content of the games.

Differences between game types occur in the area of the x0.25 and the x0.75 quantile (the x0.25 quantile describes the number of years under which 25% of the players are; x0.75 accordingly describes the 75% borderline). The box between the x0.25 and the x0.75 quantile contains the 50% average aged players (players who are older than 25% of the users and who are younger than 25% of the users). The x0.5 quantile is represented by the line inside the box; this quantile represents the median of the players' age.

As one can see, the FPS sector in the sample has a median age of 18.1 years with a sigma (standard deviation) of 5.7. The peak age is around 15 to 18 years, while the maximum age (except for a few insignificant outliers) is 27 years. RTS games show a similar distribution, hence keeping this in mind both game types feature a solo and PvP competition. Also the pay-to-play (monthly fees) method from most MMORPGs tends to be a reasonable factor for younger gamers to stay with the FPS and RTS sector where most games only have the initial purchase cost.

On the other hand, RPGs in our sample show a higher affinity to a mature player community. The arithmetic average age is 22.01 years with a sigma of 6.2. Similar to that distribution is the SG box plot; although it is distributed more uniformly. At least RPGs feature a variety of team challenges and a strong focus on PvE; both are indicators for the significant differences.

Compared to other survey results, the age and gender distribution shows significant differences. In [Thee 2007] the overall US game market is evaluated. Their findings indicate that:

(1) The average game player is 35 years old and has been playing games for 12 years.

(2) The average age of the most frequent game buyer is 40 years old. In 2008, 96 percent of computer game buyers and 86 percent of console game buyers were over the age of 18. And, 83% of game players under the age of 18 report that they receive their parents' permission when renting or buying games, and 94% say their parents are present when they buy games.

These differences in the survey results are based on different samples because [Thee 2007] focuses on the US market and includes all game genres (also casual games, browser games, etc.). The participants were selected due to their buying behaviour and were contacted directly. The games purchased do not necessarily contain a multiplayer option or any form of online interaction between the players.

As shown, the average age in the sample of [Thee 2007] is 35 years, whereas the distribution of online gaming behaviour reveals an average age of 18.1 to 22.1 years, depending on the game type. This large gap is based on two circumstances:

- the more mature players with high gaming experience (12 years) tend to play casual, and single player games rather than online games
- many players stay in contact with game concepts that they are familiar with

One should keep in mind that the Internet and the network aspect for computer games became popular during the 1990s, and current blockbusters (like MMOGs) started in 1998. Therefore, the older player generation is familiar with single player and hot-seat game concepts. One possible explanation for this gap is that older players prefer known game concepts over new game concepts. Therefore, the number of competitive online games in this age group is low. This evidence is further underpinned by [Casu 2006], one of the main statements in the report is: "*Casual games are replacing television viewing as an important stress reliever after work and during lunch hours*".

In fact, [Casu 2006] reveals that the number of older players (35+) in this segment comes to 62%, most of them only play short sessions, which leads to the next aspect. The amount of leisure time available is lower for the mature player group, therefore game preferences for these players differ from current competitive online games. These players prefer simple game mechanisms, an easy competition with an AI and

the opportunity to switch the game off at any time instead of competing intensively with other players.

(3) Forty percent of all game players are women. In fact, women over the age of 18 represent a significantly greater portion of the game playing population (33 percent) than boys aged 17 or younger (18 percent).

The argumentation for the gender distribution likewise compares the distribution of online game behaviour survey with the survey results from [Casu 2006] and [Thee 2007]. Again, a significant difference in the number of female players can be found. Since 51% of the casual players from the sample of [Casu 2006] are female, this shows a clear difference in the gaming interest. Women in any age group show significantly less interest in in-game mechanisms or competition intensive online games. Thus, the sample for the distribution of online game behaviour only contains 2.8% female players.

This evaluation of the age distribution among game types leads to an analysis of the total game time per week. As hardcore players are mainly categorized by being stubbornly persistent in playing the same game with far above average interest than other players, this also increases their average game time [Frit 2006]. By analyzing the correlation between high game time and nationality in [Frit 2006] one can observe that there are countries with a significantly higher average game time. Sweden, Norway, the UK and the Netherlands score explicitly higher than Poland or Switzerland.

In order to substantiate this difference, a one-way ANOVA model was used. The analysis of variance (ANOVA) describes a collection of statistical models in which the variance observed is partitioned into components (explanatory variables). There are certain assumptions for the ANOVA which are necessary in order to use the ANOVA. The assumptions are (1) *the independence of cases*, (2) *normal distribution* of the data and (3) *equal variance* between the sub-groups.

In general, the F-Test and the T-Test are methods related to the ANOVA model. Depending on the underlying data and the number of groups that one wants to compare, either the F-Test or T-Test must be used. The T-Test directly compares two sub-groups and decides whether or not a statistically significant difference exists. The F-Test on the other hand compares multiple groups for a potential difference. By

doing so, the F-Test indicates whether at least one sub-group shows statistically significant differences; however, it does not indicate which sub-group is different.

For the statistical analysis of the demographic data and the hypothesis in the following sections, the assumptions for the tests hold true. For each case the independence of the variables exists, no influence between two questions was measured. Furthermore, the variables are confirmed to be normally distributed due to the Jaque-Bera test. Finally, the variances of the sub-groups are similar, in fact, for most of the sub-groups they are nearly the same (due to the large size of the sample).

The nationality is suggested to be an influencing variable for the game time; therefore the sample is clustered into sub-groups by the attribute nationality. Each of the countries selected had at least n=10 participants in it, therefore the degree of freedom for the F-Test is at least 9 (it is effectively much higher when comparing two of the major countries). The results of the F-Test for several sub-groups indicate that the F value is much higher than $F_{.05}(9,9) = 3.388$, therefore one can reject the null hypothesis. In other words, the average game time is influenced by the nationality.

The countries with a higher average game time can be regarded as game-affine countries. One possible explanation for this behaviour could be the difference in the national connectivity standard. As an example: Scandinavian countries have several providers that offer explicit gaming flat rates (low bandwidth compared to other DSL connections, but no error correction and therefore a latency of around 20 ms for most game servers). This finding indicates that the nationality is also an influencing factor for the game behaviour.

Another interesting approach is the language, because all of the high scoring countries (except for the Netherlands) have native languages that are limited to their region, whereas English is their most important foreign language. Most online games have a majority of English-speaking servers. Poland and Switzerland also have a strong relation to the German language. Especially international MMOG retailers like SOE and EA often initially release their games in English. Some best-selling games (like Everquest or Linerage) have never had a translated version.

Hypothesis 1:

One of the major aspects in the analysis of the game time is the potential influence of the different game types. The relevant null-hypothesis is:

> **$H0_1$: No difference in the average game time per week between the game types exists**.

If the overall game time is influenced by the game type, the null-hypothesis needs to be rejected. It can only be rejected if a statistically significant difference between the game types can be identified.

The statistical analysis makes it necessary to cluster the sample into sub-groups with the same game type. The smallest sub-group has a size of n=197 entries (SG players), therefore the degree of freedom for the F-Test is 196. Since the remaining sub-clusters have more than m=1,000 subjects in them, the effect of the critical value for the F-Test becomes relatively small, therefore the $F_{.05}(197, 1000)$ is taken. For larger values of n, the critical value of the F test becomes smaller. With an $\alpha=0.05$ the critical value $F_{.05}(197, 1000) = 1.16$ (smallest value is reached for $F_{.05}(\infty,\infty) = 1$) . The F-Test for the sub-groups indicates an F-value of 1.9, which is significantly higher than the critical value. Thus, the null-hypothesis can be rejected. The result of the F-Test can be interpreted as follows: a significant difference between the sub-groups exists, although the F-Test alone does not indicate which of the sub-groups differ. Further T-Tests (direct comparison of two sub-groups) underscored the finding. As a result, the participants' average game time is influenced by their game type.

When considering the average game time of each sub-group cluster, both FPS and RTS reveal a relatively similar game time as opposed to SG and RPG. Based on confidence intervals with $\alpha=0.05$, the players from the FPS game type generally tend to invest $15.2 \leq x \leq 17.1$ hours per week on average in gaming. The respective RTS average hours of game time per week interval is (16.1; 18.9). Both game types also feature a group of players with significantly more time than that mentioned above (20% in both game types scored over 40 hours per week).

On average, the SG community spends significantly less time on gaming: the confidence interval for the average hours of game time per week is (10.2; 11.8),

which is the lowest score among the game types. Again, approximately 20% of the player base scores noticeably higher than that. The RPG player community has the most outstanding score, the interval of this sub-group for their average hours of game time per week is (32.1; 33.2). By taking a closer look at the distribution, one can observe that more than 40% of all players tend to play more than 40 hours a week, several of them even reached the 56 hour borderline. This large number of players from the RPG sector with 40+ hours a week is an interesting point for a deeper evaluation of the game time.

Hypothesis 2:

$H0_2$: The relative gaming time (game time divided by available leisure time) is not influenced by demographic factors or the game type.

The correlation between game time and different game types has already been discussed above. To further analyze differences in game time among the game types, a new key performance indicator (KPI) has been created.

> *The relative game time equals the total game time per week divided by the total leisure time per week.*

This number can range from 0 to 1, where 0 equals no time of the available leisure time is spent on gaming and 1 indicates that all of the available leisure time is spent on computer gaming.

Again, the score of the sub-group RTS players and FPS players is relatively equal. The average relative game time confidence interval for the RTS sector is (0.41; 0.49). Accordingly, the average related game time confidence interval for the FPS sector is (0.47; 0.55). This signifies that for any given sample of the FPS sub-group with a 95% probability, their average game time per week is 0.47 to 0.55 (which equals 47% to 55% of the available leisure time). The SG sub-group once again scores significantly lower with a confidence interval of (0.24; 0.33) and the RPG sub-group scores significantly higher compared to FPS and RTS with a confidence interval of (0.75; 0.79).

In order to compare the different game types, a F-Test with all of the sub-groups is used. The F-value of 2.55 is significantly higher than the critical value of $F_{.05}$(197,

1000) = 1.16. The null-hypothesis can therefore be rejected because a significant impact on the relative game time exists due to the game types. By comparing the different confidence intervals, the most significant difference can be found between the SG sub-group and the RPG sub-group. These confidence intervals show a major gap, which leads to the conclusion that the null-hypothesis can be rejected because the game type has a statistically significant influence on the relative game time.

Hypothesis 3:

$H0_3$: The overall gaming time (hours per week invested in gaming) is not influenced by the players' attitude towards real-life events (family, real world events, job).

The concept of different content in each of the game types leads to the game-related activities. The analysis of the overall game time per week also includes possible relations with real-life events. If a correlation between the overall game time per week and these real-life events exists, then hardcore players would not only demonstrate a similar behaviour in-game, but they would also share a common attitude towards real world events. The survey includes a common set of five real world-related questions which are equally applicable for all game types. These questions are:

(1) Would you eat at the computer in order to not miss out on an in-game event?

(2) Is the game the main topic in your conversations with friends?

(3) Do you plan your day around in-game activities?

(4) Do you hide your amount of game time in front of your family/friends?

(5) Would you neglect your job in order to play more?

If a correlation between in-game attitude and real world events exists, then the null-hypotheses must be rejected. Statistically, the evaluation starts by clustering the whole sample into sub-groups. For this survey, six sub-groups were chosen according to the number of real-life questions answered positively. A positive correlation would indicate: the more positive answers to real-life questions, the higher the overall game time. The smallest sub-group is the sub-group with 0

positive answers to these questions with overall n=121 members, therefore for the F-Test the degree of freedom is 120.

The F-Test compares the average overall game time of the six sub-groups with each other. The critical value for $F_{.05}(120,500) = 1.25$, the most significant F value was reached between the sub-group with 0 positive answers and the sub-group with 5 positive answers. With a F value of $3.1 > F_{.05}(120,500) = 1.25$ the null-hypothesis can be rejected. Most of the other F-Tests also indicated a correlation between the number of positive real world answers and the overall game time per week.

Figure 7.2 illustrates the percentage of positive answers divided by game types. Again, the RPG player community scored significantly higher than the others. For instance more than 67% of all RPG players eat at the computer, 75.90% plan their day around the game and 81.00% would neglect their job to play more.

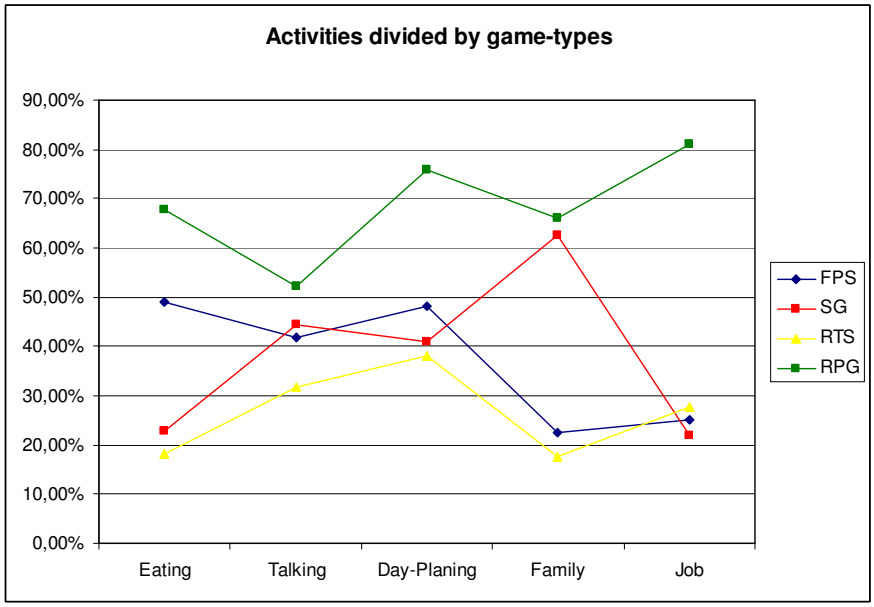


Figure 7.2. Game-related activities divided among different game types. Each bar on the Y-axis equals 10% of the users (belonging to the relevant game type) in the survey. The percentage of positive answers is shown in the diagram

The only mentionable difference is the low number of players in the FPS and RTS genre, who would hide their game time in front of their family and friends (only 22.50% and 17.50%). Another mentionable difference in the three game types FPS,

RTS and SG is a high percentage of SG players who would hide their amount of gaming in front of their friends and family.

Table 7.1 concludes the context discussed by looking at the percentage of players who answered 4 out of 5/5 out of 5 questions with yes. The section 4 out of 5 already contains all the players who answered 5 out of 5 questions. All game types except for the RPG sector feature an approximately 20% player base of hardcore gamers (similar to the 20% mentioned for game time observations). Admittedly, the rate of players with 5 out of 5 positive answers is relatively small (this is due to the normal distribution).

Table 7.1. Measurement of percentage of users with 5 out of 5 and 4 out of 5 questions answered positively

	5 out of 5	**4 out of 5**
FPS	9.28%	22.39%
SG	8.33 %	23.12 %
RTS	6.20 %	20.80 %
RPG	15.51 %	38.53 %

7.2 Statistical Analysis of Virtual Fragmentation

Overall, more than 14,000 users participated in the virtual fragmentation survey. However, the number of valid data sets is only 5,800; most of them were excluded from the statistical evaluation because they were incomplete. Compared to the "distribution of online game behavior" survey, the number of potential multiple answers by a single participant (repeated answers) and random answers (white noise) were lower. This section briefly introduces the most important survey results. A more detailed version of the results can be found in [Frits 2007.2].

Demographic values:

As already indicated in section 4.2, the user group in our survey is not homogeneous. The gender distribution clearly indicates more male users; the confidence interval with α=.05 for the percentage of male players is (88.4; 89.3). This indicates that with a probability of 95%, any random sub-group in our sample will have 88.4%-89.3% male participants on average. Compared to our analysis of hardcore gaming behavior [Frit 2006] this number is surprisingly high. The marital status indicates similar results as in [Frit 2006], most of the users were single 86% (highly correlating with the age, nearly 98% of the players under 25 were solo); 12.6% were married, 1.1% divorced and 0.3% widowed. When contemplating the sample's distribution, the user group for virtual fragmentation was distributed in a typical manner with high values in high school and the university, and low values in worker and no professional areas. A relatively low number of 7.2% are unemployed, 4.0% worker/apprenticeship, 38.8% high school and 50.0% college students. This distribution is illustrated in Figure 9.23 (see Appendix).

Hypothesis 4:

The first step in analyzing the gap between real world and virtual behavior is to compare the MOS values with each of the five attributes. The underlying hypothesis is:

$H0_4$: No difference between average MOS values for real world behaviour and average MOS values for virtual behaviour exists.

In order to reject this null-hypothesis, at least one contrary example must be found. Therefore, the attribute of conscientiousness is evaluated in detail. A high value in this attribute includes behaviour that is thoughtful, goal-driven, organized and mindful with regard to details.

Figure 7.3 illustrates the difference between the average mean opinion score of virtual conscientiousness and real world conscientiousness. Both values are based on the respective questions (questions from the conscientiousness inventory).

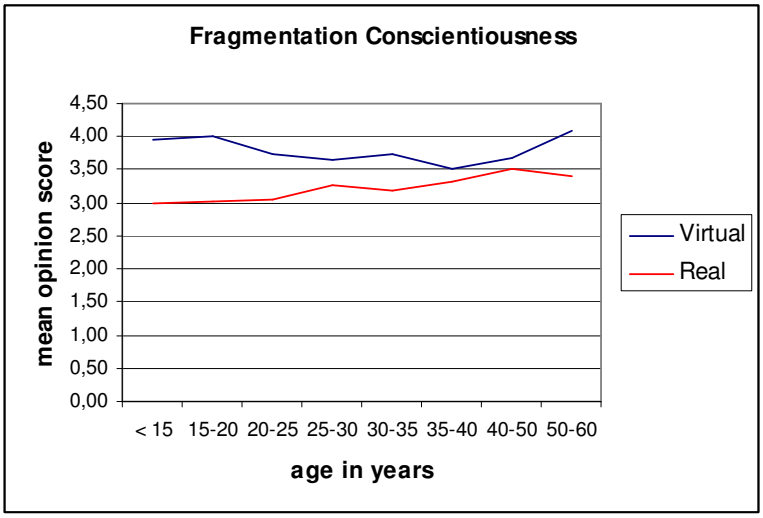


Figure 7.3. Virtual fragmentation of the attribute "conscientiousness"; the MOS scores marked in red range from 1 to 5.

The virtual conscientiousness is higher for any age group, however, especially the younger players under 20 reveal a larger gap. A difference of nearly 1.0 (average virtual C = 4.0 and average real C = 3.0) clearly shows different behaviour. The real world conscientiousness shows a slightly positive slope towards age; mature users tend to be more goal-oriented and mindful. In contrast, especially young players show a very high score in virtual conscientiousness, thus indicating a well-organized and goal-oriented online behaviour.

The statistical analysis of this gap includes a one-way ANOVA (requirements normal distribution, independence is given). The sample size is n= 5,801, therefore the degree of freedom is 5,800. Since only two sub-groups are compared with each other, the two-sided T-Test is used for this analysis. With an α=.05 the respective

critical value $T_{.05}(5800,5800) = 1.96$. For this analysis, the average answer values for real world and virtual world conscientiousness are compared. The T value of 2.57 > $T_{.05}$ (5.800, 5,800) = 1.96 ranges in the critical area, which leads to the conclusion that the null-hypothesis can be rejected. A significant difference between real world and virtual world values exists, further T-Tests of the other four attributes as well as the overall gap underscore this finding.

Hypothesis 5:

The second important hypothesis describes the further influencing factors for the virtual fragmentation effect. In order to prove an influence of these factors, the null-hypothesis must be rejected:

> **$H0_5$: Further factors (such as age, gender, educational level and game type) do not have a significant influence on the level of virtual fragmentation (size of the gap between virtual and real world behaviour).**

Again, at least one statistically significant contrary example must be found to reject the null-hypothesis. The data for age distribution provides an excellent influencing factor for further analysis.

The age distribution within the sample is not distributed equally. The x0 quantile (youngest participant) is 11 years, the median (x0.5 quantile) is 21.2 years and both the x0.25 and x0.75 quantile show a close age distribution around the median. This means that 50% of the survey users are between 16 and 25 years of age. This age distribution is illustrated in Figure 7.4.

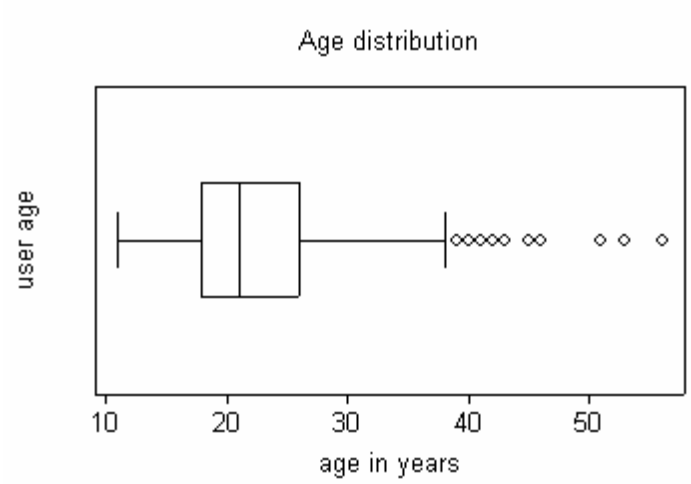


Figure 7.4. Box plot of age distribution in the survey "virtual fragmentation"

As an example, the differences between real world and virtual world mean opinion scores for the attribute agreeableness are evaluated. A high value indicates attributes like trust, altruism, kindness and affection. Figure 9.24 illustrates the findings divided among the age groups.

For this test, each of the attributes is clustered into sub-groups by age. The youngest participants are around 10 years of age, the oldest (except for outliers are 40 years of age). A clustering with five years per cluster creates eight sub-groups. Afterwards, the average MOS scores for the real world and virtual world attributes are compared by means of a one-way ANOVA test.

The most significant result was determined for the attribute of agreeableness. Overall, the smallest sub-group is the group of 36 to 40 years of age with an overall number of 112 subjects, thus the degree of freedom is 111. The younger sub-groups (10 to 15 years and 16 to 20 years of age) have a T value of 2.94 and 3.12 compared to the critical value $T_{.05}(110,110) = 1.98$ which shows a significant difference between real world and virtual world average mean opinion scores. In contrast, the middle age sub-groups (26 to 30 years, 31 to 35 years and 41 to 45 years of age) show T values with no significant difference between the average real world and virtual world mean opinion scores.

Both results together show that the younger age groups differ significantly in terms of statistics, whereas the middle age groups do not differ. This leads to the

conclusion that the initial null-hypothesis can be rejected because a contrary example has been identified. The evaluation of the other four attributes further underscores the influence of age on the size of the gap between average real world and virtual world mean opinion scores.

7.3 Results of the Mobile Lobby Survey

More than 1,100 users participated in the underlying online survey, and 1,080 of them answered all ten questions. The questionnaire also included a small demographic part (location, gender, age) for the further analysis. Moreover, the gathered results underpin important aspects of the current mobile phone gaming situation:

The data show that a majority of over 90% would like to play mobile phone games with their friends; although only 42% (36% women, 45% men) would consider playing the same games with random people; Figure 7.3 summarizes this aspect. The statistical analysis shows a minor correlation (0.51) between gender and preference to play mobile phone games with friends. Women tend to prefer not playing these games with strangers even more so than men. Hence, the game design should seriously take into account that random matching obviously does not match the users' preferences.

Game preference results show a more surprising outcome. The probe shows a strong correlation between location and favorites in mobile phone games. Most participants in all countries prefer arcade games (31.5% in Sweden up to 64% in the UK). Overall, the other game types (RTS, FPS, action and puzzle) are practically identical with around 15% each. However, there is a tremendous variance with regard to the countries. One possible explanation for the popularity of arcade games is their very simple game control mechanisms. Nevertheless, only 10% of the users would pay to play mobile phone games. The average mean opinion score of taking mobile games seriously is 2.1 (1= definitely not serious, 5= very serious), which further reflects the casual nature of mobile games. Both facts together show that location, gender and age independent (only very minor correlation has been determined) players are not willing to pay or take the current mobile phone games seriously. The main objective appears to be playing quickly with friends.

Current mobile games on next generation handhelds feature better graphical support like 3D rendering on the Sony PSP or shading and pseudo 3D design on the Nintendo DS. If one compares the mean opinion scores of graphic popularity, it turns out that the handhelds' (MOS [mean opinion score] overall 3.78) score is significantly higher than the score for mobile phones (MOS overall 2.33). This means that the average player is satisfied with the graphical performance on handhelds, whereas on average, players tend to dislike them on mobile phones. Additionally, the data indicates a strong linear negative correlation (0.77) between age and the difference. Mature players perceive less difference within the two devices. As a result, one can conclude that shading and layer usage would further increase the acceptance of current mobile phone games.

The time required for a complete game setup was rated with a mean opinion score from 1 (very unimportant) to 5 (very important). With an overall score of 4.43, most players have a strong focus on a fast game setup. Nevertheless, the preferences varied between countries, ranging from 3.89 in Italy to 4.90 in Germany. This noticeable result leads to the software solution of a mobile gaming lobby in order to decrease the search and setup time, hence the matching time is drastically reduced and it is more likely that further opponents can be found.

7.4 Testbed Results of the 4MOG Middleware

One of the main problems in the MMOG middleware sector is the reduction of overall bandwidth because of the servers' bottleneck attribute. With a growing number of players, the bottleneck results in a delayed answer. If the server is massively overloaded, then the worst case scenario is a shutdown. Therefore, the 4MOG approach to reduce the number of packets includes a simple penalty system. The idea is to permit the users to attack the server with brute force in order to reboot it. If a user keeps on spamming over a certain period, then he/she receives a penalty. The longer this user continues to ignore the penalty, the tougher the consequences will be. With a rising penalty level, the users' maximum number of packages that one can send decreases. However, if the target changes his/her behaviour, then the penalty level will decrease again until it reaches the initial status.

As an indicator for over-proportional bandwidth consumption, the testbed uses a certain number of messages which a player is allowed to send per minute. If the user shows an unacceptable behaviour for more than 50% of the time, then he/she receives a higher penalty level. The first penalty level slightly reduces the number of packages allowed, whereas the reduction of possible packages also increases with an increasing penalty level. If a user continues to send too many packages, then the penalty level increases to its maximum which throttles the user. The middleware system keeps track of the number of messages each second and compares it to the limitation. As an example: if a user sends more than ten messages per second over 30 times in a minute (more than 50% of the time) then he/she receives a penalty level, which reduces the maximum data rate. If the player continues to send the maximum data allowed, it will lead to the next penalty level which reduces the data rate even more.

With regard to the middleware's other functionality, the penalty level system reflects the focus of the prototype implementation. The purpose of the testbed is to ensure that the message communication is implemented and that the penalty system works as intended. The additional functionalities (like player tracking, object movement, etc.) have not been fully implemented in the prototype yet since the initial testbed focuses on just one function of the middleware: network communication between server and client.

The penalty level system forces users to adopt a certain behaviour in order to reduce their individual communication behaviour with the server. One should take into account that the design of the penalty level must be individualized for each of the MMOGs. The objective of this approach is to ensure that a normal player in the game world will not receive a penalty level, whereas players who use brute force mechanisms to overload the server should receive penalty levels. A static penalty system is used for the testbed, which does not take the situation into account. For example: if a challenging fight is ongoing, users tend to take more action and thus to send more packages. Table 7.2 illustrates the different penalty levels and the relevant restrictions for further bandwidth consumption. As one can see, the first penalty level reduces the maximum number of packages for the client to 10 per second. With an increasing penalty level, the number of packages decreases even further. The

examples of the testbed do not refer to a certain MMOG; the numbers are used to test the functionality of the penalty level system.

Table 7.5. Penalty level system of the 4MOG middleware testbed

Penalty level	Maximum number of packages
Level 1	10 per sec / 30 times per min
Level 2	8 per sec / 30 times per min
Level 3	6 per sec / 30 times per min
Level 4	4 per sec / 30 times per min

The numbers are based on the technical data of Half Life [Half] and the technical report about the network structure of this game [Hend 2001]. According to the description of the Half Life server, the average number of packages from a single client per second is between four and six. These numbers can be transferred to a MMOG scenario, since the movement and interaction with the game from a client's perspective is relatively similar. Other game genres also indicate similar numbers. For example: top players from the RTS genre have around 150-200 APM (actions per minute), which approximate four to five actions per second. Since not every action requires communication with the server (for example, units are selected locally), the resulting number of communications with the server is even lower.

As expected, the clients in group one (strong spammers) received the highest penalty level on average during the entire timeframe. As illustrated in Figure 7.6, the penalty level for all players starts at 0, whereas most members from group one already reach penalty level 2 after only two minutes. Other groups score significantly lower both on average and over the long run. Especially the group of Lab 2, with less probability of adopting a new behaviour and with high initial bandwidth consumption, almost always reaches the highest penalty level very quickly. The figure also encompasses an interesting observation on the long-term trend of groups 2 and 3. Usually group 3 is expected to reach a high level faster, but only over the long run is group 3 capable of scoring higher on average compared to group 2. The test itself was conducted three times in order to ensure necessary reliability. An intelligent client system that

actually “reacts” to the given penalty level would probably construe other results; nevertheless, focus of the testbed relies on prototype’s functionality and not on the integration of next generation AI.

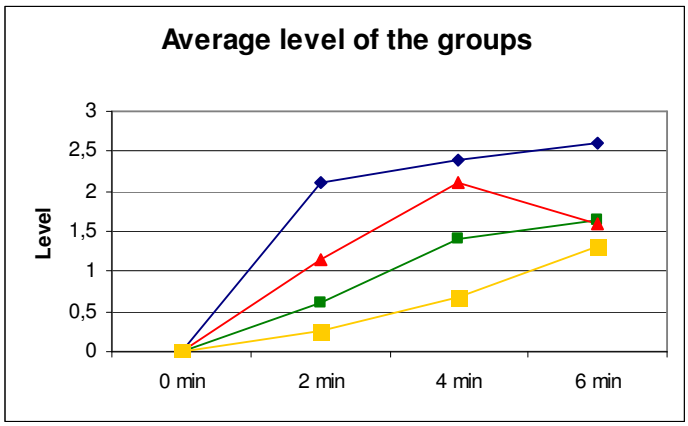


Figure 7.6. Illustration of the average penalty level for each group. The colors are: blue equals group 1, green equals group 2, red equals group 3 and yellow equals group 4

Another aspect of the testbed is the tracking of average RTT (round-trip times) and the usage of AOI (area of interest) management to significantly reduce overall latency. Therefore, the 4MOG middleware tracks the RTT for every client by using regular pings from the client and the logging system of Gamestate. The area of interest (AOI or ROI for region of interest) concept is a common design pattern in computer games today. The main idea is to limit the information that each client receives. Therefore, only events that can be seen directly by a client are submitted. If a virtual object obscures the event, then the player does not necessarily need to receive the game information. This method enables the number of packages for each client to be reduced, hence also decreasing overall network traffic.

The test aims to integrate a commonly used technique from the FPS game design into the prototype to further reduce the overall number of packages. The AOI value defines the number of fields (geometrically a square) from the current player position that is observed. If an event occurs outside of the area of interest, then the player will

not receive any information about it. Figure 7.7 features a simple example of an AOI field and the game world.

The lower the AOI value, the lower the overall number of packages that are received. In a game world with a total size of 200x200 game fields, the AOI value for the testbed reduces the fields step-wise down to 180, 150, 120, 90, 60 and 30. One should keep in mind that an area of interest of 30 still covers a significant range compared to the whole game world (30 fields in each direction equal a width 60 fields and a number of 3,600 fields overall). By covering 3,600 fields (AOI range 30) out of 40,000 fields (200x200 game fields), the client still receives information about 9% of the complete game field. In order to illustrate the size of this area of interest: current MMORPGs feature very large persistent online worlds. Everquest [Ever], for example, contains far more than 250 different in-game zones and the view of the player therefore never exceeds even 0.1% of the entire online world. For a MMOG, the AOI concept can be implemented even more strictly. As the game world is very large, the user will only need to overview a very limited part of it.

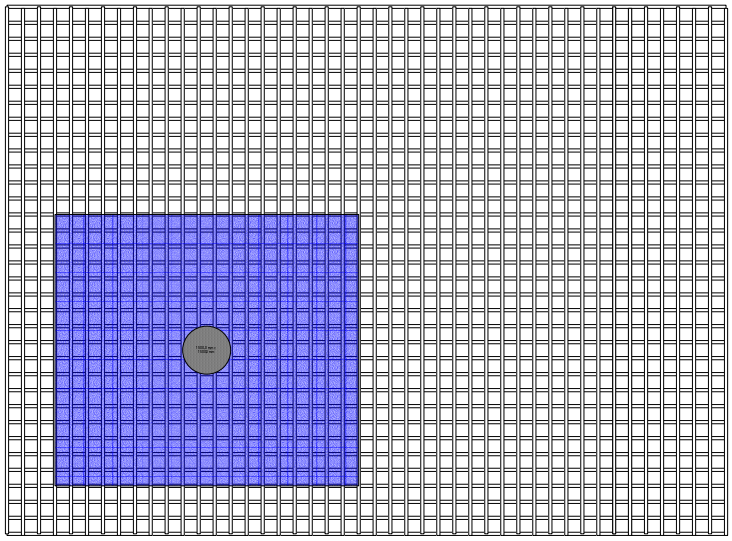


Figure 7.7. Diagram of the game field: each of the cells represents a single game field. The AOI is indicated as a rectangle around the player's position

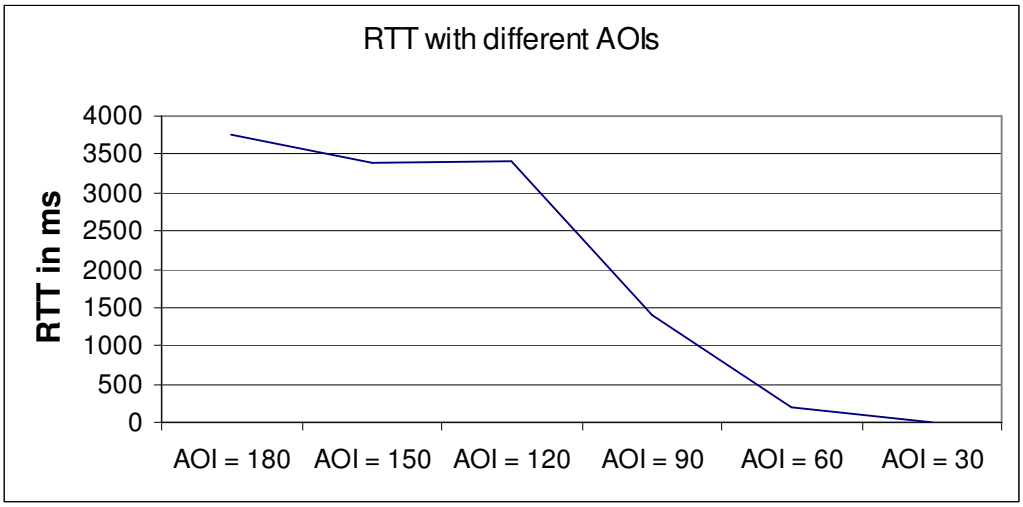


Figure 7.8. Diagram of the average RTT with decreasing AOI values and ensuing round-trip times

As illustrated in Figure 7.8, one can see that the overall RTT is reduced drastically as soon as the AOI value reaches 90. That is because players in the center no longer receive information about the entire game field. Even with a fairly high AOI rate of 30 to 60, the RTT times are 200+ times lower than compared to normal broadcasting methods.

In concluding the analysis, the testbed shows that the penalty system is capable of reducing the bandwidth by tracking each player. Supported by the middleware prototype and the Gamestate logging mechanism, a gaming provider could define certain rules for different situations and thus offer each player (depending on the behaviour) fair treatment. Furthermore, once an AOI system has been implemented, the testbed shows that the average RTT time (also tracked by using the logging system of the middleware) is reduced significantly by forwarding the necessary messages only. Both of them are a first test regarding scalability, which is a significant aspect for every MMOG.

7.5 Comparison with the Related Work

In order to illustrate the contribution of the individual approaches, Table 7.3 includes an overview of the related work introduced and the approaches in this thesis. All of them are evaluated with regard to the six main parameters: scalability, mobility, reusability, quality of service, user behaviour and hardware expenditure. Chapter 3.6 describes each of the parameters in detail.

The experimental results in this section cover different aspects of the research. First of all, the statistical analysis of player behaviour aims to understand the player's motivation and it therefore provides reliable data for further user-oriented studies. With regard to Table 7.3, this includes the analysis of player game time behaviour as well as the effect of virtual fragmentation. Both approaches are compared with the related work afterwards in this section.

Another aspect of computer gaming is the mobile environment; the statistical analysis includes a survey about the mobile gaming lobby. However, due to lack of cooperation with Eve Online, it was unfortunately not possible to test the Instant Messenger integration (which is the second contribution in the field of mobile gaming) at a real MMOG. Since the open source MMOGs do not provide the necessary number of players (most of them are not even playable currently) and other professional companies did not answer the requests for participation, it was not possible to set up an appropriate testbed with a real MMOG to evaluate the game interfaces of IM integration. Nevertheless, it is also possible to at least substantiate the middleware's functionality; therefore a GUI was designed (which uses the IM's game interfaces). The following section compares the MC Chat and the IM Integration with the related work.

Finally, the third topic covers the research field of gaming middleware applications. The 4MOG middleware (which supports MMOGs with a core set of functions) is compared to the related work.

Table 7.3. Assessment of the related attempts and the approaches of this thesis

	Scal-ability	Mobility	Reus-ability	QoS	User Behaviour	Hardware Expenditure
Mobile Aware Games						
Asynchronous mobile gaming	- -	++	0	-	-	++
RFID tags	-	++	+	- -	-	+
MCChat	**-**	**++**	**++**	**0**	**0**	**+**
IM Integration	**+**	**++**	**++**	**-**	**-**	**0**
Massive Multiplayer Games						
Interest-based AIs	++	-	+	-	- -	0
P2P MMOG architecture	++	0	++	- -	- -	+
Public server architecture	0	-	++	+	- -	-
Gaming Middleware						
Service Platform	-	0	++	-	0	- -
OpenPING Middleware	+	0	-	+	+	0
Patch Scheduling	+	-	+	0	- -	0
4MOG middleware	**++**	**-**	**+**	**0**	**+**	**0**
User Categorization						
Bullet time in multiplayer	- -	0	-	+	++	+
Player behaviour and design	0	-	+	0	++	0
Hardcore game behaviour	**0**	**0**	**++**	**0**	**++**	**0**
Virtual Fragmentation	**0**	**0**	**+**	**0**	**++**	**0**

7.5.1 Player Behaviour

The research field of the influence of a player's behaviour on computer games includes various aspects; hence the approaches in this thesis aim to only improve certain aspects. Both of the questionnaires offer a wide range of interesting findings, especially concerning the cultural influence on game time. With the relatively high player effort in the MMORPG landscape, these findings offer a basis on which further approaches can be built. As a conclusion to the findings about leisure time, one can observe that not only do RPG players have the highest available leisure time, but they also have the highest percentage rate of game time. An intelligent game design that does not automatically reward a massive time investment would also help to scale in-game balancing better and therefore reduce the effects of strong addiction.

On par with the game time distribution analysis, the effects from [Frit 2007.2] clearly underpin the strong influence of virtual worlds on individual behaviour. Based on the taxonomy of Bartle [Frit 2006.4] and the Five-Factor Model a clear categorization helps to personalize further game content, for example, by using a learning game application that can create specialized content for each of the online players.

Compared to the player preferences and game design analysis from the related work section (3.5.2 and 3.5.3), the approaches in this section offer a more basic evaluation. Both of the results aim to include scalability, therefore both MMOGs and classic small multiplayer environments have been analyzed separately. The strong focus relies on the reusability and the integration of player behaviour. The game time distribution analysis offers basic figures on which further surveys and player categorization can be built. The effect of strong addiction has been addressed in literature more frequently over the last five years although no statistically reliable figures were provided.

On the other hand, the effect of virtual fragmentation clearly opens up a new opportunity for the game design. By integrating personalized content into the next generation games, each of the players will engage in the online world more intensively because the content reflects the players' interests. Combined with the architecture of intelligent AI design (like in the approach of interest-based AI design from section 3.3.1) it would also be possible to create more realistic NPCs.

The results of the surveys are representative, since the advertising strategy included more than 100 different game forums. The users of these forums are explicitly game-oriented, reflecting the current game scene. In order to obtain the most representative set of participants possible, the advertisement was evenly distributed among the four game types: RTS, RPG, FPS and SG. The mechanism used to collect the data (online survey) also supports a correct survey design since the users are familiar with the Internet environment and the mechanisms.

Although the analysis of player behaviour offers a wide variety of possible improvements, one should still keep in mind that these underlying factors cannot solve more specific problems. By understanding the player's motivation and the mechanisms, one can design better software applications, however, the technical limitations still remain. For example: movement and language perception would be excellent for an online environment, however, they can currently not be realized.

7.5.2 Mobile Lobby

The mobile gaming section contains the greatest number of technical limitations, however, the potential of wireless and flexible gaming applications is growing continuously. The approaches in this thesis aim to improve the actual usage of current equipment in order to optimize the possibilities for mobile gaming.

By taking a closer look at the current mobile phone gaming section, one undoubtedly realizes that the users' expectations significantly exceed the limits of current mobile gaming technology. Therefore, most of the serious gamers [Frit 2006.4] prefer to play on consoles or PC platforms due to the higher performance. Under consideration of these unequal conditions and the need for a fast game setup, the current mobile game design fails to support the players' interests. The approach introduced to understand the mobile gaming sector is based on statistical data from the user survey.

The surveys' results are representative; the advertising strategy explicitly included game forums with mobile content (such as Sony PSP, Nintendo DS, etc.). Therefore, the survey participants reflect the users with an interest in mobile gaming. The mechanism used to collect the data (online survey) also supports a correct survey design, since the users are familiar with the Internet environment and the

mechanisms because many of the mobile gaming users also play games on their personal computer.

By gaining insights into game type preferences as well as the importance of playing with friends, the game design can be changed accordingly in the future. Furthermore, the J2ME-based mobile phone lobby offers a flexible solution for a fast game setup, hence offering the opportunity to find possible opponents for the player for all of the installed games. The focus of this approach relies on the mobility and reusability of the application. Especially in the mobile sector, it is important to create common standards to support the large number of different mobile handheld devices.

Another important aspect is the integration of game-related activities in a mobile environment. With the current display and input limitations, it is also possible to integrate game-related activities such as chatting in the mobile sector. Due to the limitations in performance, current game clients from PC and console games will not run properly on the next generation handhelds.

Therefore, the integration of instant messenger applications into current MMOGs aims to merge in-game and out-of-game contacts and to create a common standard for the players. The IM application is kept as generic as possible; hence the focus relies on both mobility and reusability. It is necessary to support as many MMOGs and IMs as possible to create a flexible system for all players.

By integrating these aspects into a mobile environment, a player could focus on game-related activities like in-game scheduling or chatting instead of playing games that he/she does not enjoy. Compared to the related work of asynchronous mobile gaming (3.2.1) and RFID location aware gaming (3.2.2), the approaches introduced differ slightly from the current mobile gaming strategy. Instead of integrating new technologies in order to create low-performing applications, the current resources should be used more effectively in order to improve mobile gaming. The definition of standards and reusability for future applications are particularly important aspects.

7.5.3 4MOG Middleware

The tremendous growth of the research field for massive multiplayer gaming reflects the great importance of the game type for the upcoming evolution. Current related

work within this section, especially the design of improved patch scheduling (3.4.3), further increases the number of possible mechanisms for a more effective interaction with a high number of simultaneous players.

The 4MOG middleware approach in this thesis aims to integrate current techniques between the network and the game layer, hence offering various interfaces for both sides. The purpose of the middleware is to support the developer with functions in order to simplify the technical implementation of the MMOG. Therefore, the developer can spend more time on creating individual game content. One of the main advantages is the focus on reusability. Especially for the growing landscape of MMOGs, it is important to reuse effective problem solutions in order to further develop the games without substantially increasing the overall development time.

In clear contrast to other middleware applications like the OpenPING middleware (3.4.2) or the service platform (3.4.1), the problem addressed in the 4MOG middleware does not focus on effective network distribution. Although the distribution of player load and effective content management are important issues for the current game design, the approaches introduced offer different effective solutions as well.

Therefore, the 4MOG middleware improves player tracking and uses a penalty level system to prevent the users from spamming the server. The testbed of the 4MOG middleware is used to determine whether the spamming behaviour of the clients can be reduced with the penalty level system. Both the penalty level and the AOI (area of interest) are examples of how network traffic can be reduced, hence increasing the scalability (which is important for all MMOGs). One disadvantage of the current middleware application is the lack of mobile availability; the current technical equipment in the mobile gaming sector does not scale with high numbers of simultaneous players.

Compared with the GASP middleware, the closest related approach, both applications have similarities in their underlying design. The 4MOG application also uses a message-based implementation due to the higher performance and possibility to pre-calculate locally. However, the 4MOG middleware focuses on functionalities for MMOG design and aims to support the developer, whereas the GASP middleware offers more generalized support for all multiplayer games. The GASP

middleware does not include a penalty system yet because the focus of the GASP application is not only on the MMOG sector. As explained earlier, a reduction of the server network traffic becomes more important in a massive multiplayer scenario due to the significantly higher number of clients.

8. Conclusion and Future Work

> "Two wrongs don't make a right, but they make a good excuse", Thomas Szasz.

This chapter contains a general conclusion about the problematic field of computer gaming. With regard to the main aspects of player behaviour, mobile gaming and middleware solutions for massive multiplayer gaming, the results are briefly summarized. Furthermore, limitations of the approaches are discussed in order to illustrate the current limitations in computer games science. Finally, possibilities for further research options are given.

8.1 Conclusion

The field of computer game design is still connected to a wide variety of problems. As indicated in the introduction, a "best practice" solution does not exist. One should therefore consider each of the approaches separately in order to understand the contribution. Furthermore, it is also necessary to address the approaches' limitations. Especially in the upcoming work, it is vital to keep these options in mind in order to create a realistic plan for future development.

Therefore, it is essential to give an overview of the contribution of the dissertation in order to summarize the different parts of the thesis. Table 8.1 gives a brief overview of the most important contributions (especially source code and programming contributions) that extend the modelling and analysis of this thesis.

In the context of mobile aware games, both approaches have contributed important programs to integrate mobility into the current game design. The MCChat lobby offers a flexible framework for further game expansion. With the necessary support of game developers, such an application could be used as the standard gateway for a mobile game setup. Another research approach (the IM integration) generates a middleware application to support the external chat mechanisms in the current

generation of massive multiplayer games. It therefore helps to move one step closer to a global player community that is not limited due to different games.

Table 8.1. Overview of the contribution of this thesis

Emphasis	Scientific Contribution
Mobile Aware Games	The integration of a lobby tool that enables different clients (mostly PDAs) to build an AdHoc network and have a common lobby to find other players faster and more reliably called **MCChat**; another important contribution was the integration of the external communication technologies and the theoretical support of any game client: **IM Integration**
Gaming Middleware	The creation of a flexible middleware solution that is strongly MMOG-oriented and supports the designer of the games with a core set of functions: **4MOG Middleware**
User Categorization	Detailed statistical analysis of player behaviour and the design of a reusable survey platform in order to support the important aspect of empirical analysis: **Hardcore Game Behaviour**, **Virtual Fragmentation**

The aspect of massive multiplayer gaming includes the design of the 4MOG middleware, which is a lightweight communication-oriented gaming middleware. The framework and necessary decisions regarding the design are discussed in this thesis. A prototype application has been created for the testbed. With a further implementation (mostly by implementing abstract classes), it would be possible to create an application that supports different MMOGs better (with individual functions that can be used optionally). As discussed above, it would also be possible to use the current tracking system in order to reduce cheating possibilities.

Finally, the emphasis of user categorization also includes computer science's contribution. The design of a framework for further surveys, which also supports different languages and handles the efficient and swift integration of appropriate web front-ends, further helps to increase the number of upcoming surveys in order to

verify future approaches. Additionally, the gathered data about player behaviour and game-specific effects serve as a fundamental reference for further papers as well.

8.1.1 Player Behaviour

In the player behaviour section, two main approaches are described in detail. The first evaluates the average game time with regard to social demographic factors, game type and occupation. The European questionnaire has clearly underscored that the average game time of MMOGs (especially MMORPGs) is significantly higher compared to other game types. The player group also shows differences with regard to age variance and social behaviour. Moreover, the amount of hardcore players is around 10% in all of the game types evaluated - these players take the game very seriously and therefore prioritize it over work and/or education. The results can be used to integrate mechanisms for a more intelligent game achievement that is less time-consuming.

The second approach describes the difference between in-game and out-of-game behaviour, which is called virtual fragmentation. A survey with game-related questions uses the Five-Factor Model in order to understand which parts of the behaviour are influenced. The ensuing statistical analysis shows that several factors are responsible for the differences observed. On the one hand, the cultural factor plays an important role. The social acceptance of gaming in different European countries as well as the technological equipment appear to be the main influencing factors.

On the other hand, the game type and the way that players approach the game are responsible for differences in Five-Factor Model attributes. The results can be used further to customize games in the future. As long as the necessary information for each player is provided (like classification of the game behaviour) it is possible to integrate more specific content for each user (like personalized quests or on-the-fly created landscapes that fit the users' preferences).

8.1.2 Mobile Gaming

In the mobile gaming section, the thesis describes the main technological problems of the current mobile game design. In order to analyze the players' perspective, a

survey evaluates the mobile users' preferences. Therefore, two approaches in this thesis improve the current mobile game environment.

The MCChat offers a flexible lobby application that creates a common chat platform with interfaces to the mobile games. Hence, the matching time for mobile game sessions is significantly reduced when using a common gateway.

The importance of mobile communication has also experienced tremendous growth over the last few years. Thus, the IM integration aims to implement any current instant messenger into MMOGs by offering interfaces for the basic chat functionalities. Existing buddy lists of in-game and out-of-game contacts can therefore be merged. In terms of a mobile environment, this integration reveals that even when the game client is not running, a user can still stay in touch with in-game friends.

8.1.3 4MOG Middleware

The aspect of massive online gaming has a significant influence on the current game design. As [MMOR] shows, the number of MMOGs released over the last few years is increasing rapidly. With regard to the number of players of top-selling MMOGs (like World of Warcraft or Linerage 2), one can observe that the games are becoming very popular (both games have over 7 million subscribers worldwide who pay to play the games).

The 4MOG middleware therefore provides the core set of functions for designers of MMOGs with interfaces for network features in order to reduce technical programming efforts. Also by tracking each player, further features can be included like cheat protection for movement exploits, a specific analysis of the player behaviour or a detailed statistic of the in-game load balance.

The testbed results indicate a stable scalability towards the TCP and UDP layer, also the technique of AOI management is used as an example to reduce the overall network bandwidth. Furthermore, another aspect of the bandwidth reduction was analyzed; the testbed included a penalty level system to permit brute force attacks on the game server.

8.2 Future Work

The current work presented in this thesis covers different aspects of the game research (such as player characterization, mobile gaming and massive multiplayer gaming). Since the field of game research is comparably young, some parts of the work (like the player behaviour analysis) are performed with the objective of creating underlying data for further research approaches. As demonstrated, the different approaches aim to increase both the understanding of player interaction as well as to further increase the quality of main aspects of next generation game design (like mobile and massive multiplayer gaming).

Before providing an outlook for future improvements, one must also take a close look at the limitations of current game development with regard to the three main aspects of this thesis.

Player behaviour. As indicated, the understanding of current player behaviour is a prerequisite for a better game design. Identifying negative results from in-game values like the creation of sweatshops helps to understand the current situation, although it does not create a solution for it. An intelligent game design could prevent these abusive strategies, but strictly limiting in-game trading or reducing the players' opportunities will also have negative consequences. This dilemma depicts one of the limitations; even with a deep understanding of the game mechanisms, a solution can only be identified by the industry. As long as the current game design fulfills the needs of the player community, the alternatives will not be taken into account.

Furthermore, the statistical analysis also has its limitations; by understanding the average player it is not possible to exclude negative effects. The most well-known example is the predicted negative adoption of in-game violence for real world behaviour. Several research approaches aim to prove the positive aspects of computer gaming, however, it is not the median (average) player who should be evaluated. Since statistical analyses always feature a small mistake probability, even with a 5% error factor it leaves considerable room for a single exception. For the example of in-game violence and real-life behaviour, this means that even if a clear statistical correlation between computer gaming and a positive real-life behaviour is shown, there can be still exceptions (like a single person running amok after playing

a FPS). Therefore, it is also not possible to exclude certain behaviour; a prediction always aims to understand the majority.

With regard to the limitations mentioned, the upcoming player behaviour analysis can extend current results. One of the main problems addressed for the future is the influence of long-time game sessions on the player's performance. Using in-game calculation as well as eye movement tracking can help to understand the general change in perception after a long game session. Furthermore, the psychological effects can be used for an improved game design; personalized content that depends on the player's characteristics would take the current game design to a new level.

Mobile gaming. This section probably includes the greatest number of limitations with regard to technology. The vision behind mobile gaming is a completely mobile environment that allows the player to use entertainment software everywhere. Unfortunately, this cannot be realized with the current equipment. Two main factors prevent such a mobile environment:

Most of the current handhelds cannot support next generation games. This is especially due to the strict limitations in display and input. The negative limitation effects are evaluated in-depth in [Frit 2006.4], showing that PC performance with the same games is superior as opposed to the mobile devices. Also due to the maximum resolution of 480x320 pixels, it is not possible to use current 3D models from PC games because their complexity (number of polygons, texture depth) exceeds the performance of the mobile devices' graphical adapters. This reduces the graphical quality further compared to the console and PC sector.

Furthermore, the structure of the mobile network is a major problem for next generation gaming. Both the GSM/UMTS and the WLAN/Bluetooth opportunities have individual advantages and disadvantages for game design, although real-time games (like FPS or RTS) set strict requirements for maximum latency, which GSM/UMTS cannot provide. In addition, in order to be truly mobile, a network connection must be established at any and all locations. However, this aspect is not provided either due to the limited signal range of WLAN and Bluetooth. In summarizing both disadvantages, neither GSM/UMTS nor WLAN/Bluetooth can support real-time mobile games completely (the design must take the drawbacks of the network structure used into account).

Besides these two main problematic fields, the aspect of the lack of standardization also poses a barrier to the creation of a common game design for all platforms. Mobile game devices today are broken down into the gaming-oriented handhelds (like Nintendo DS or Sony PSP), mobile phones and PDAs; each of the categories features different sub-groups as well. Especially in the mobile phone sector that has a wide variety of devices, the missing standards lead to multiple, non-compatible versions of the same game.

Undoubtedly, the current technical situation strictly limits the evolution of mobile game design. Nevertheless, the mobile lobby and integration of IM software for the mobile devices offer flexible solutions to improve certain aspects (like a faster game setup). In the future, the implementation of the mobile lobby can be further improved by supporting additional next generation devices, thus creating a common standard for spontaneous game setups. Based on the evolution of a next generation in handheld devices, newer evaluations can lead to deeper insights. Due to the rapidly evolving mobile gaming field, next generation devices will reduce technological limitations, hence creating the preconditions for further approaches.

Massive multiplayer gaming. As one can see, most attempts for game design are created in the massive multiplayer game sector. Due to the scalability problem of the S/C structure and the opposing motivation of high controllability by the publisher, most of the network solutions tend to offer a better performance, but one disadvantage is that they limit the controlling options. Therefore, these attempts present theoretical advantages with regard to scalability, but none of them have been implemented in a commercial game to date.

An analysis of the limitations of massive multiplayer gaming today leads to the identification of several aspects that are responsible for current problems. The setup of a persistent online world with the players' income also creates a virtual economy. Although the second generation of MMOGs integrates mechanisms to limit in-game inflation, several problems still arise (like real world values of the virtual goods). As described above, the current commercial game design does not support intelligent solutions due to a lack of understanding of the players' motivation.

Another important limitation is the online game world's size. Nevertheless, the usage of shards to equally distribute the overall number of players on multiple servers aims

to limit the maximum number of simultaneous players to less than 15,000 users. A new network protocol and higher server performance would increase this maximum number slightly, however, the general problem of scalability combined with less to no loss in control is still not resolved.

Moreover, the limitation of available content is another significant topic. The AI of the NPCs has improved significantly over the last five years. However, the evolution has just begun, leading to growing interaction complexity and the individual motivation of each NPC. Complexity of the game worlds is expected to increase further which is why expanding the static persistent environments with on-the-fly content or expanding the game world is another concept that has not been realized yet.

By taking the various limitations into account, one can understand that research in the field of massive multiplayer gaming still needs to address a wide variety of problematic fields. Extending beyond the middleware approach introduced in this thesis, the upcoming work will certainly include a further improvement of the 4MOG application. One possible evolution would be to integrate player characteristics for each client. In-game content could be personalized with data gathered, which would subsequently create special events for each of the players. For example, based on the classification of Bartle, an explorer would receive an additional, generically created area to explore. The game would therefore reward each gaming strategy by understanding each client's preferences.

9. Appendix

"In the future everyone will be famous for fifteen minutes", Andy Warhol.

This section contains the most important abbreviations as well as a reference for further statistical results.

9.1 Abbreviations and Glossary

AI	Artificial Intelligence (mostly NPCs)
AFK	Away From Keyboard
ASP	Active Server Pages
Beta test	Beta stage of the software lifecycle in game production, mainly with the active help of participating test players
Bot	Automated program to act in-game, often used for farming
Casual	In terms of gaming: approaching the game with (below) average interest
Clan	A union of players with the same goal, often a cooperative institution
Cellular Input	A mobile phone's input options, available key can differ in terms of number and placement depending on the phone. Generally, the input includes the typical ITU-T keypad for dialling
Death match	A special game type where each battles against all of the others; the player with the highest score wins
DHT	Distributed Hash Table
Endgame	The game content for experienced players with in-game characters on the highest level

ER model	Entity Relationship Model (for database design)
Farm(ing)	Playing the game in order to only gather resourses like gold or other treasures
FPS	First Person Shooter
Group	A temporary in-game collaboration of fellow players who have the same goal
Guild	A long-term, larger scaling collaboration of players who have a similar game attitude
Hardcore	A behaviour towards the game; describes a far above average interest in the game content and a strong motivation to achieve
MMOFPS	Massive Multiplayer Online First Person Shooter
MMOG	Massive Multiplayer Online Game
MMORPG	Massive Multiplayer Online Role Playing Game
MMOSpace	Massive Multiplayer Online Space Setting
MOS	Mean Opinion Score.
NPC	Non-Player Character
OSCAR	Open Service for Communication in Real-Time
P2P	Peer-To-Peer
PC	Player Character or Personal Computer
PvE	Player versus Environment
PvP	Player versus Player
Raiding	A large group of cooperating players who have teamed up to beat a single strong opponent
RPG	Role Playing Game
RTS	Real-Time Strategy
S/C	Server/Client Structure

SG	Sports Game
UMMORPG	Ultra Massive Multiplayer Online Role Playing Game

9.2 Additional Figures

Figure 9.1. Illustration (ER model) of the GASP middleware server domain model. It serves as a general specification for programming.

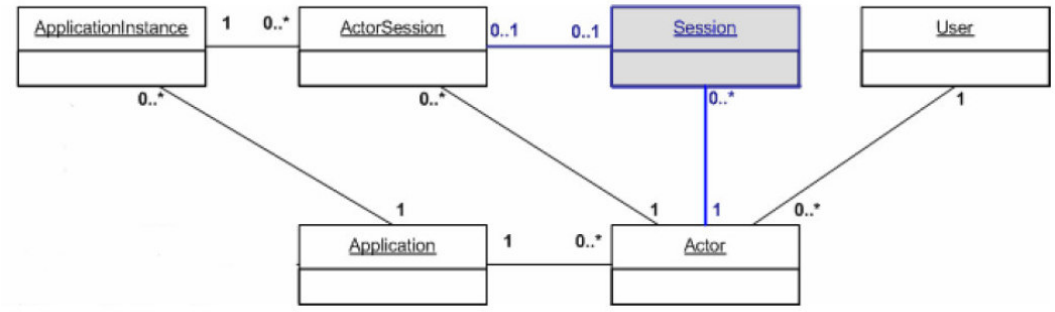


Figure 9.2. UML Diagram of the 4MMOG middleware network classes. The network interface is implemented in detail by a server and a client (with individual functionalities).

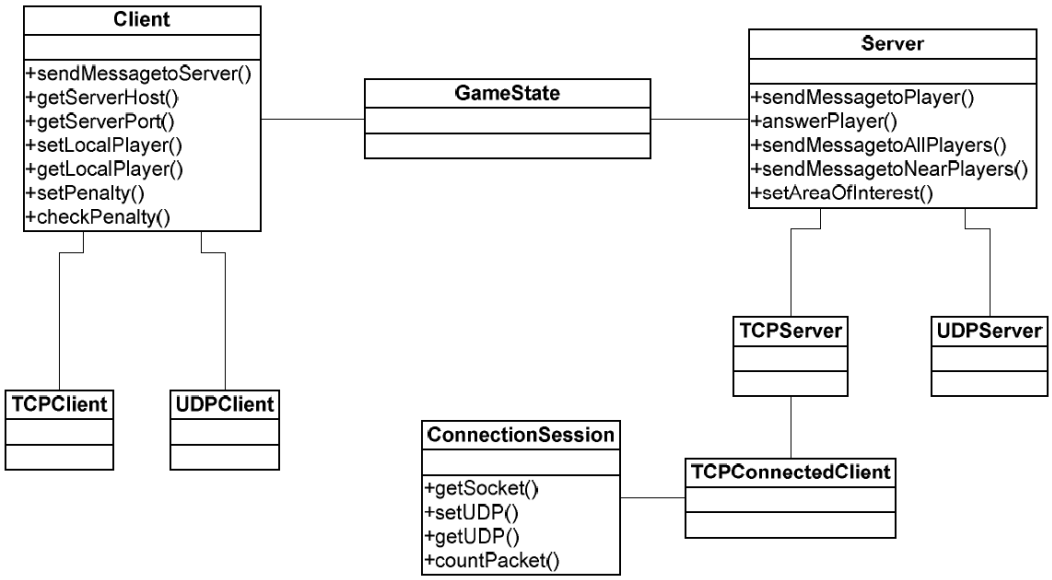


Figure 9.3. UML view of the 4MMOG middleware's client side

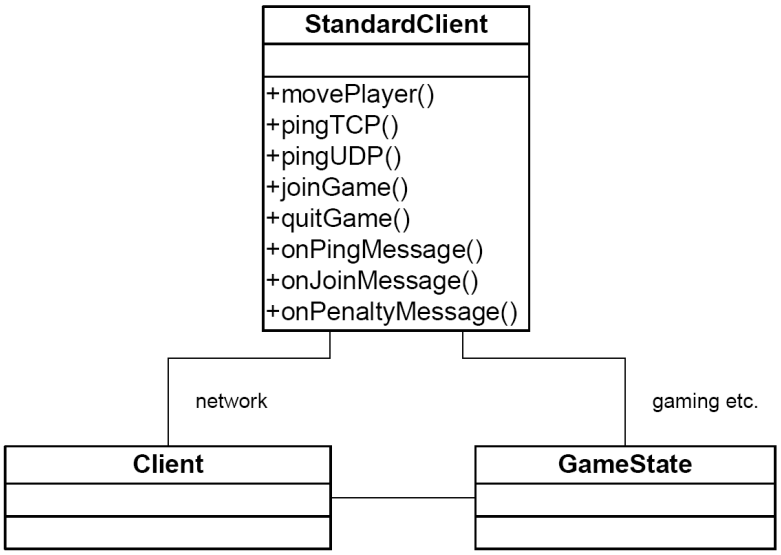


Figure 9.4. UML view of the 4MMOG middleware's server side.

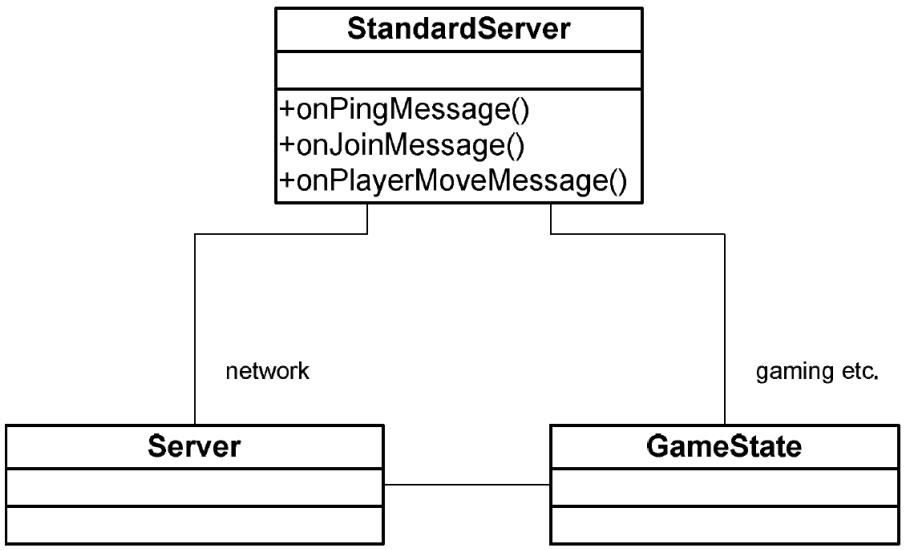


Figure 9.5. Example of the message system in the middleware application. Each message has its own type (implemented in XML), which enables all important parameters to be specified and the existing message system to be expanded if necessary.

```
<Message id="0" name="Penalty" type="tcp" priority="0">
    <MessageDesc>When server detects flooding give user a penalty</MessageDesc>
    <Parameters>
        <Param>
            <ParamType>int</ParamType>
            <ParamName>intervallLength</ParamName>
        </Param>
        <Param>
            <ParamType>int</ParamType>
            <ParamName>numberOfMessagesAllowed</ParamName>
        </Param>
        <Param>
            <ParamType>int</ParamType>
            <ParamName>duration</ParamName>
        </Param>
        <Param>
            <ParamType>int</ParamType>
            <ParamName>state</ParamName>
        </Param>
    </Parameters>
</Message>
```

Figure 9.7. UML Diagram of the complete message system. It also includes a limited set of functions for the NetworkMessage() class. As one can see, the MessageListener uses the observer pattern to minimize the communication efforts required.

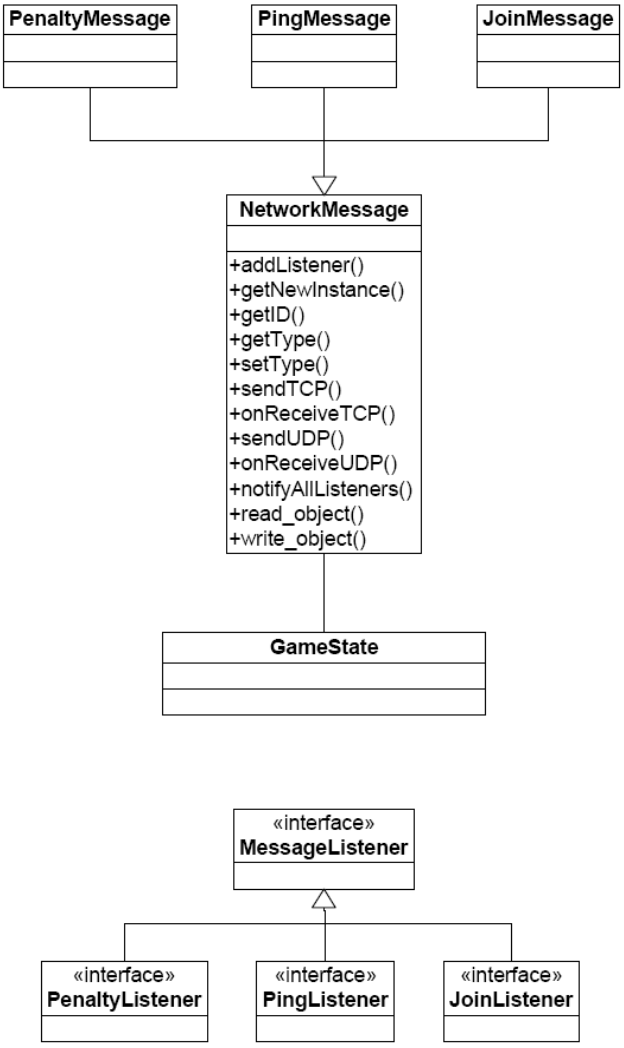


Figure 9.8. Complete overview of the middleware's client side.

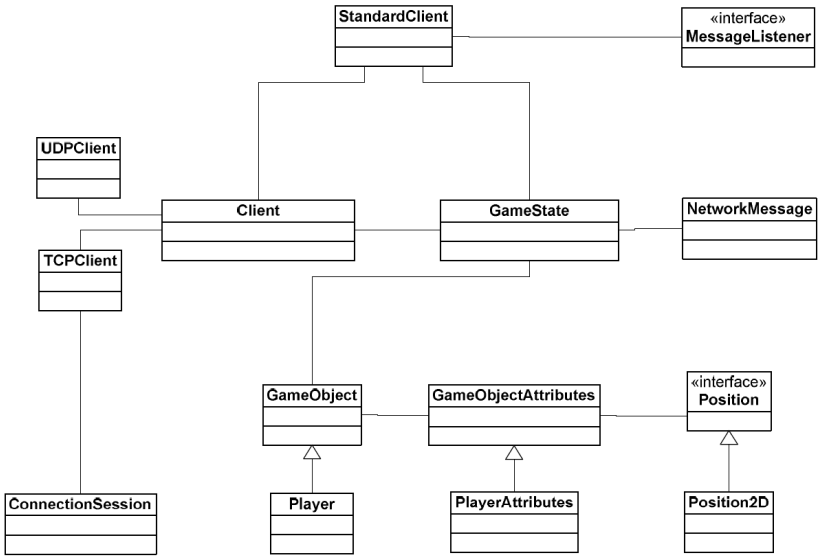


Figure 9.9. Complete overview of the middleware's server side.

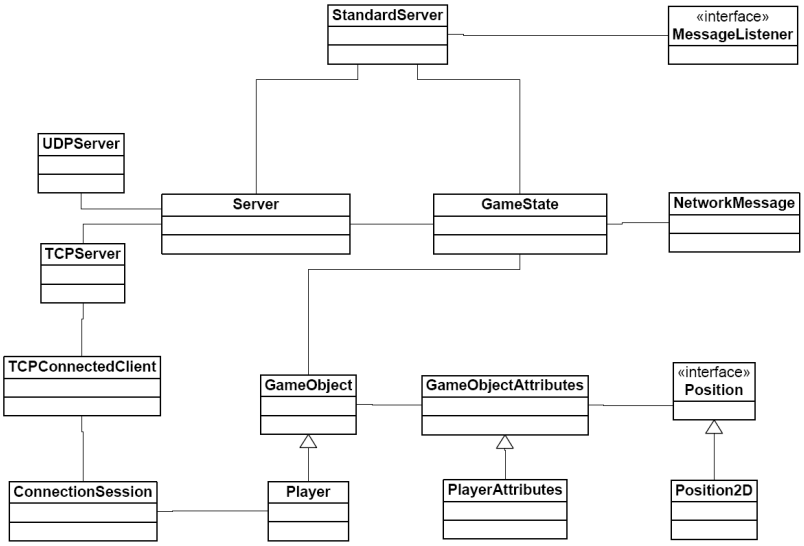


Figure 9.10. First solution approach for the IM integration. On the EVE client side, extension was necessary to (a) access IM clients and server data, (b) give users the possibility to control these clients, (c) create client to access the AIM network. On the EVE server side, an extension was also necessary to provide the user data required by the clients.

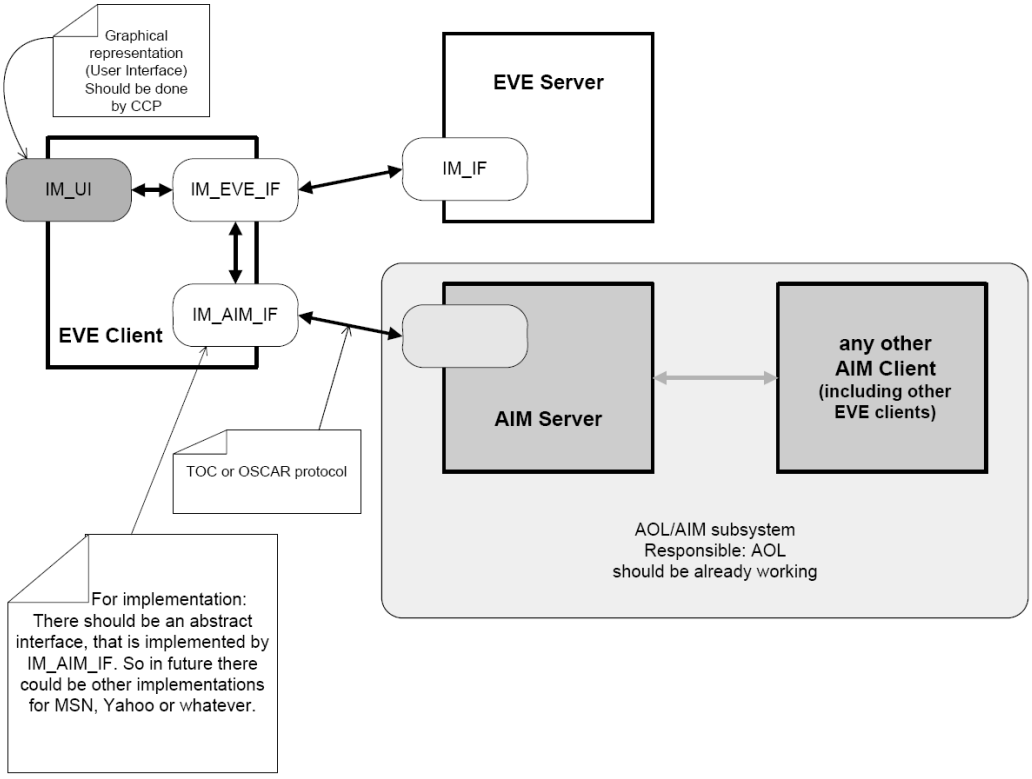


Figure 9.11. Final solution for the IM integration. By making use of the OPIS server, the name mapping problem can be solved easily. On the EVE server side, no extension is necessary. All IM functionalities are implemented on the EVE client side. The connection between the EVE server and client is only the standard connection used for the actual game.

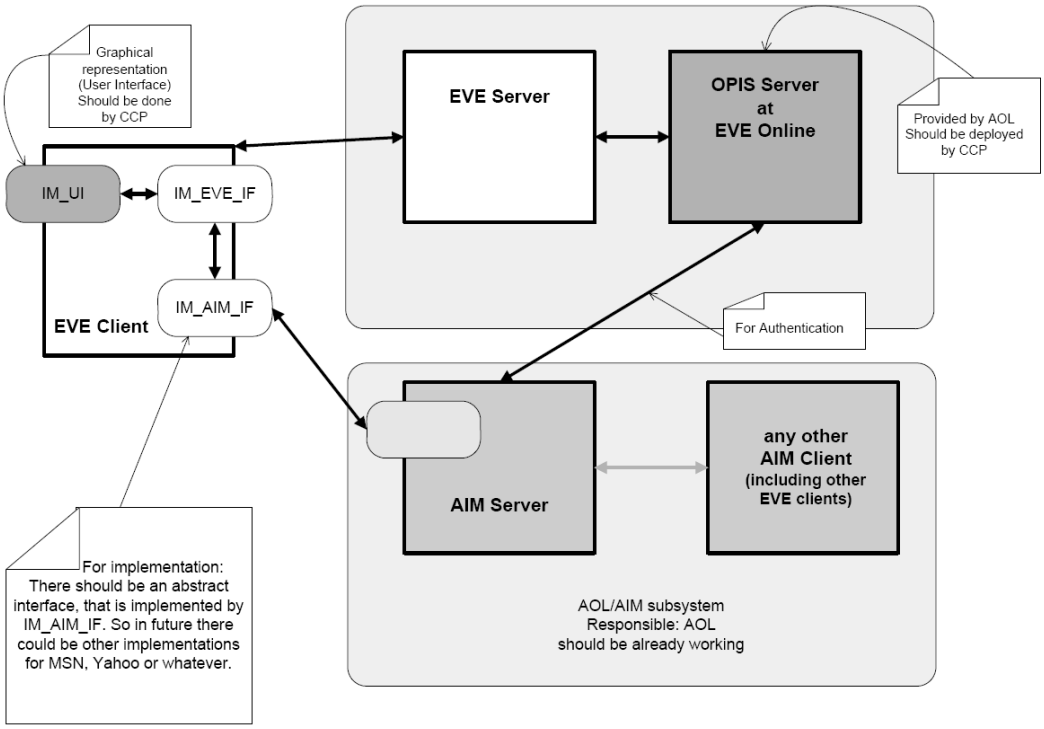


Figure 9.12. The IM framework's component structure. The prototype developed is the IMInterface, the communication core to provide Instant Messaging. EVE Client symbolizes any GUI to use the IMInterface, an it can be substituted by another GUI. AccCoreLib is the section of the AIM SDK that enables communication with the AIM network. Additional sub-systems would be required to support multiple protocols.

Figure 9.13. Class overview of the IMInterface. The central point and main access point from the GUI is the ClientManager that can manage multiple clients (implementations of the ClientInterface). The BuddyList internally represents the consolidated contact lists of all connected clients and thus manages multiple contacts (Buddys). The AimClient implements all methods of ClientInterface and uses the AccCoreLib of the AIM SDK to access the AIM network. Currently, the MsnClient does not exist and is only shown for demonstration purposes. The ClientInterface and its implementations have safeguarded access and can only be accessed from within the IMInterface. Certain global lists have been defined for several purposes. Only the key methods are shown for the classes, although more actually exist.

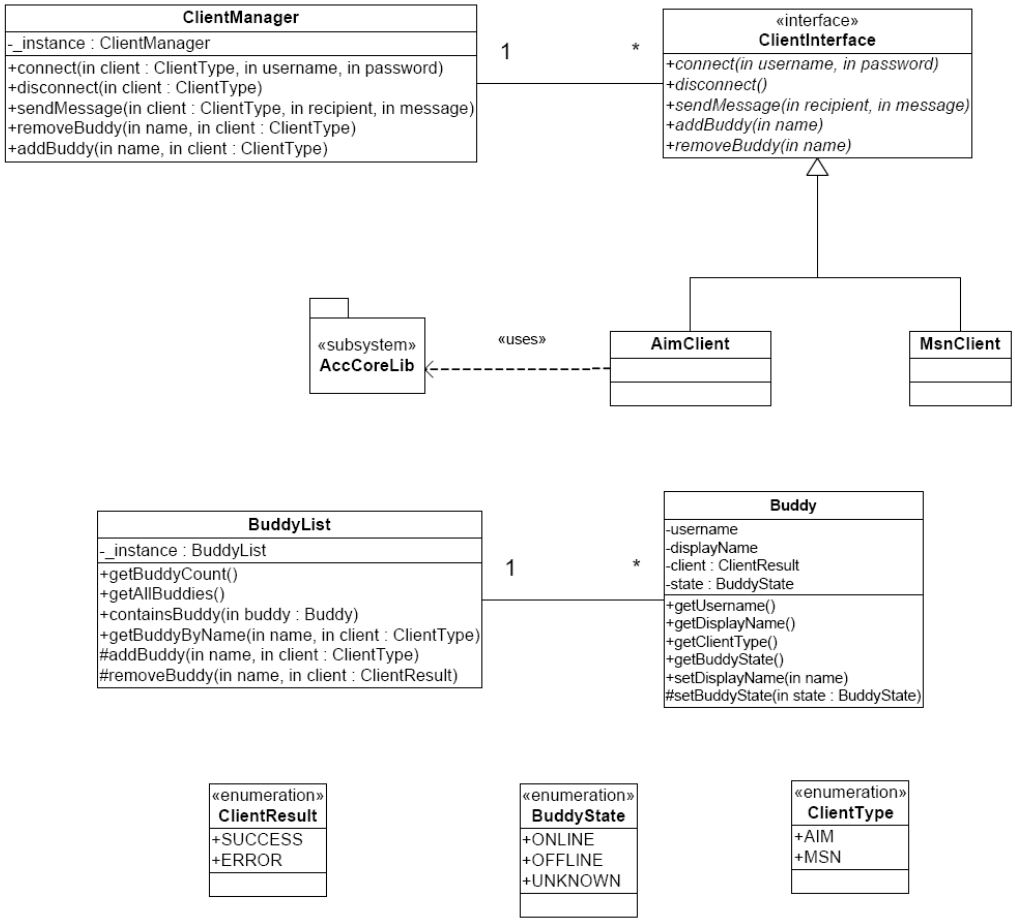


Figure 9.14. Sequence diagram for the connection process in the IM framework.

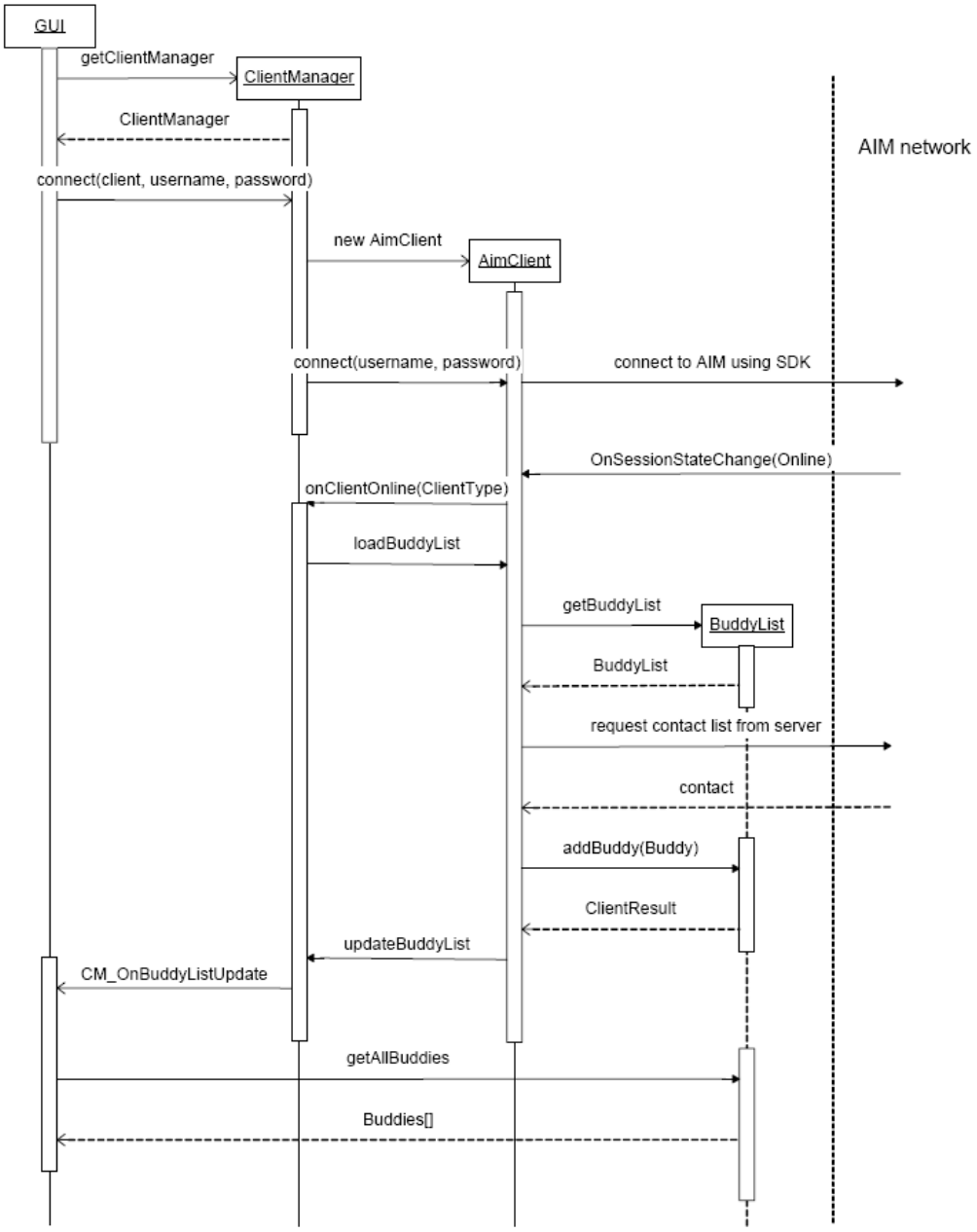


Figure 9.15. Methods of the ClientManager in the IM framework.

Method	Parameters	Return value	Use
Public static methods			
getClientManager	-	ClientManager	Get the Singleton instance
evalReturnCode	code:ClientResult	string	Returns a textual description of the given ClientResult
mapNameToClientType	name:string	ClientResult	Maps name (e.g. "AIM") to client type (e.g. "ClientType.AIM")
mapClientTypeToName	Client:ClientType	string	Maps client type (e.g. "ClientType. AIM") to name (e.g. "AIM")
Public methods			
isBusy	-	boolean	Returns whether ClientManager is busy
connect	client:ClientType user:string password:string	ClientResult	Connects to given client with user-name and password
disconnect	client:ClientType	ClientResult	Disconnects the given client
getUsername	client:ClientType	string	Returns the username that associated with the given client, null if client is not connected
isOnline	client:ClientType	boolean	Returns whether client is online
sendMessage	client:ClientType recipient :string text :string	ClientResult	Sends an IM to the given recipient using the given client
sendMessage	buddy:Buddy text:string	ClientResult	Sends an IM to the given Buddy
removeBuddy	buddy:Buddy	ClientResult	Removes the given Buddy from the contact list
removeBuddy	name:string client:ClientType	ClientResult	Removes contact with given name from the contact list of the given client
addBuddy	name:string client:ClientType	ClientResult	Adds a new contact with the given name to the contact list of the given client
Internal methods (only accessible from within IMInterface)			
onClientOnline	client:ClientType	-	Perform necessary actions when client goes online
onClientOffline	client:ClientType	-	Perform necessary actions when client goes offline
onClientError	client:ClientType	-	Signal client errors
receiveMessage	client:ClientType sender:string text:string	ClientResult	Receive an incoming IM, called by the clients
sentMessageResult	client:ClientType recipient:string result:ClientResult	-	Called by clients when a sent IM has been acknowledged with ack or error
sentMessage	client:ClientType recipient :string text :string	-	Called by clients when IM was sent. No information about success or failure given here.
updateBuddyList	-	-	Called when BuddyList was changed
OnBuddyRemoved	client:ClientType name:string	-	Called by clients when contact has been removed. Removes the contact from internal BuddyList, too.
OnBuddyAdded	client:ClientType name:string	-	Called by clients when contact has been added. Adds the contact to internal BuddyList, too.

Figure 9.16. Methods of the ClientInterface in the IM framework.

Method	Parameters	Return value	Use
connect	username:string password:string	ClientResult	Connect the client with given login information
disconnect	-	ClientResult	Disconnect the client
isOnline	-	boolean	Returns whether client is online
isBusy	-	boolean	Returns whether client is currently busy
getUsername	-	string	Get the username that is associated with this client
loadBuddyList	-	ClientResult	Request contact list from server and store contacts in internal BuddyList
unloadBuddyList	-	-	Remove contacts of this client from internal BuddyList (not from server). Used when client goes offline.
sendMessage	recipient:string text:string	ClientResult	Send an IM to the given recipient
addBuddy	name:string	ClientResult	Add a contact to the clients contact list (on server, not in BuddyList)
removeBuddy	name:string	ClientResult	Remove contact from clients contact list (on server, not from BuddyList)

Figure 9.17. Methods of the AimClient in the IM framework.

Method	Parameters	Return value	Use
Public methods			
Implementation of ClientInterface, so same method here			
Event handlers (for events triggered by AIM) (private)			
s_OnStateChange	Called when state of the AIM session (AccSession) has changed (e.g. client goes online or offline)		
s_OnSecondarySessionStateChange	Called when state of a secondary session (e.g. IM session) changes		
s_OnImReceived	Called when an incoming IM was received		
s_OnSendResult	Called when sent IM was acknowledged (ack or error)		
s_OnImSent	Called when IM has been sent		
s_OnUserChange	Called when property of a user (contact) has been changed (e.g. online status)		
s_OnBuddyAdded	Called when contact has been successfully added		
s_OnBuddyRemoved	Called when contact has been successfully removed		

Figure 9.18. Methods of the BuddyList in the IM framework

Method	Parameters	Return value	Use
Public static methods			
getBuddyList	-	BuddyList	Get the Singleton instance
Public methods			
getBuddyCount	-	int	Returns the number of Buddies in the BuddyList
getAllBuddies	-	ArrayList	Returns all Buddys in BuddyList as an ArrayList of Buddys
containsBuddy	buddy:Buddy	boolean	Returns whether given Buddy is contained in BuddyList
containsBuddy	name:string client:ClientType	boolean	Returns whether contact with given name for given client is contained in BuddyList
getBuddyByIndex	position:int	Buddy	Returns the Buddy at the given position in the BuddyList
getIndexByBuddy	buddy:Buddy	int	Returns the position of the given Buddy within the BuddyList
getIndexByName	name:string client:ClientType	int	Returns the position of the Buddy with the given name
getBuddyByName	name:string client:ClientType	Buddy	Returns the Buddy with the given name
Internal methods (only accessible from within IMInterface)			
addBuddy	buddy:Buddy	ClientResult	Adds the given Buddy to the list
addBuddy	name:string client:ClientType	ClientResult	Creates new Buddy with given name and adds it to the list
clear	-	-	Removes all Buddys from the list
removeBuddy	buddy:Buddy	ClientResult	Remove the given Buddy from the list
removeBuddy	name:string client:ClientType	ClientResult	Remove the Buddy with the given name and client from the list.

Figure 9.19. Methods of the Buddy in the IM framework

Method	Parameters	Return value	Use
Public methods			
Buddy	username:string displayname:string client:ClientType	-	Constructor. Username and client type have to be given, display name may be NULL (is set to username then)
getUsername	-	string	Returns the username
getDisplayName	-	string	Returns the display name
getClientType	-	ClientType	Return the client type this Buddy belongs to
getBuddyState	-	BuddyState	Return Buddy's online state
setDisplayName	name:string	-	Set a new display name. If name is NULL, displayname is set to username
Internal methods (only accessible from within IMInterface)			
setBuddyState	state:BuddyState	-	Set the Buddy's online state when it has changed.

Figure 9.20. List of the survey questions from how hardcore are you.

Demographic aspects:	Gender, age, marital status, graduation, leisure time, income per month, nationality, number of relatives playing
Questions concerning the game behaviour:	Which MMORPGs are you playing at the moment? How many different MMORPGs have you played so far? How many days per week do you play MMORPGs? How many hours a day do you play MMORPGs?
How often do you participate in following activities:	PvP, PvE (except for raids), raids, role playing game, farming (business) Which one of the mentioned activities do you like most?
How often do you participate in activities around the game:	Websites, forum, communication tools (ICQ,MSN, Ventrilo, Teamspeak), making/watching videos (about the game)
Further questions:	Do you eat at the computer in order to not miss out on anything important in-game? Is the MMORPG relevant for your daily schedule (such as raid time)? Do you schedule your day around the MMORPG? Have you ever cancelled a date because of a spontaneous online event? Do you neglect your job for the MMORPG? Did you ever call in sick from work to have more leisure time for the game? Do your friends/family know how much you play? Do you want to be the very first person to have new game items? Is MMORPG the number one topic number when you talk to others?

Figure 9.21. List of the survey questions from virtual fragmentation

Demographic aspects:	Gender, age, marital status, graduation, leisure time, income per month, nationality, number of relatives playing
Questions concerning the game behaviour:	Which MMORPGs are you playing at the moment? How many different MMORPGs have you played so far? How many days per week do you play MMORPGs? How many hours a day do you play MMORPGs?
I would classify myself as someone who …	... talks to other people quite often. … spends whole evenings with just a good dialogue. … has a friendly attitude. … tries to establish a dialogue with others. … talks to other people frequently. … guides others with my greater experience and overview. ... tries to attain a leading position. ... is rapidly enthused. ... puts a lot of energy into something that is really important. … is capable of listening to others. … has a friendly attitude towards others. … seldom has a hard time with strangers. … always tries to listen to my friends' problems. … can listen to very personal topics with friends. … receives a lot of trust from other persons. ... likes to help other people. ... helps friends during tough times. … always tries to solve problems in a team. … forgives mistakes. … is a reliable worker.

	… has a difficult/challenging job.
	… is rather lazy.
	… works on an important task until it gets done.
	… schedules dates and work efficiently.
	… already knows what the day will be like.
	… makes plans and sticks to them..
	… thinks about the consequences of my own action.
	... thinks first and then acts.
	… keeps promises/dates.
	… is often depressed.
	… gets angry very quickly.
	... has a hard time trying to totally relax.
	… doubts my own decisions.
	… doesn't have strong nerves.
	… is often afraid.
	… takes criticism far too seriously.
	… handles stress well.
	… worries a lot.
	… is emotionally stable.
	… is interested in art, music and literature.
	… is interested in science.
	… has creative hobbies.
	… often comes up with new ideas.
	… has a good imagination.
	… totally dislikes routine jobs.
	… is interested in new inventions.
	… reads a lot.
	… wants to have serious discussions with others.
	… has a wide range of interests.

	... loves to chat with other players.
	… has at least just as much fun chatting as I do playing the game.
	… often visits different places and carries out different quests.
	… seldom uses harsh or rude language when chatting.
	… often communicates with others inside the game.
	... never stays quiet in the group chat.
	... leads a player group or guild.
	... has problems when others take the lead.
	... spontaneously explores new quests and instances with others.
	... follows the game with above average motivation.
	... is capable of realizing other players' situations.
	... is friendly to most of the other players.
	… does not judge new players prematurely but rather, I treat them in a friendly manner.
	... can interact well with other players.
	... knows secrets about other players but I keep them to myself.
	... isn't afraid to discuss personal topics online.
	... doesn't only aim to achieve my personal advantage.
	... also tries to help out with social problems in the group/guild.
	... prefers group play over solo play.
	... would play with other players who make mistakes again.
	... takes guild and game schedules seriously.
	… takes the game seriously.
	... is relaxed and just waiting for things to happen online.
	… plays the game and the quests seriously.
	... solves multiple quests in parallel to the deadline.
	... has strict goals for the current game session.
	... reads a lot of related topics to coordinate the game better.
	… worries about my online reputation.

	... thinks first and then acts.
	... keeps my promises.
	... also plays even when I don't really want too.
	… gets upset about game events very quickly.
	... always tries to reach the same game goal.
	... is always worried about doing something wrong in the opinion of other players.
	... cannot enjoy the game in a relaxed atmosphere.
	… is worried about the consequences of my own action.
	... cares too much about what others think of me.
	... can handle stress well.
	... worries a lot.
	... doesn't have any trouble solving game-related issues in a peaceful manner.
	... tries to understand the game mechanisms.
	... tries to help develop the game with my own ideas.
	… enjoys level design and monster analysis.
	... frequently shares my own ideas with fellow players.
	... has a good imagination.
	... always looks for new game content and dislikes doing the same things twice.
	... automatically reads the patch news.
	... analyzes the game in detail.
	... is capable of talking about more than just the game with others.
	... has more background game knowledge than others.

Figure 9.22. Illustration of the Join() function in the mobile lobby [Lehm 2006].

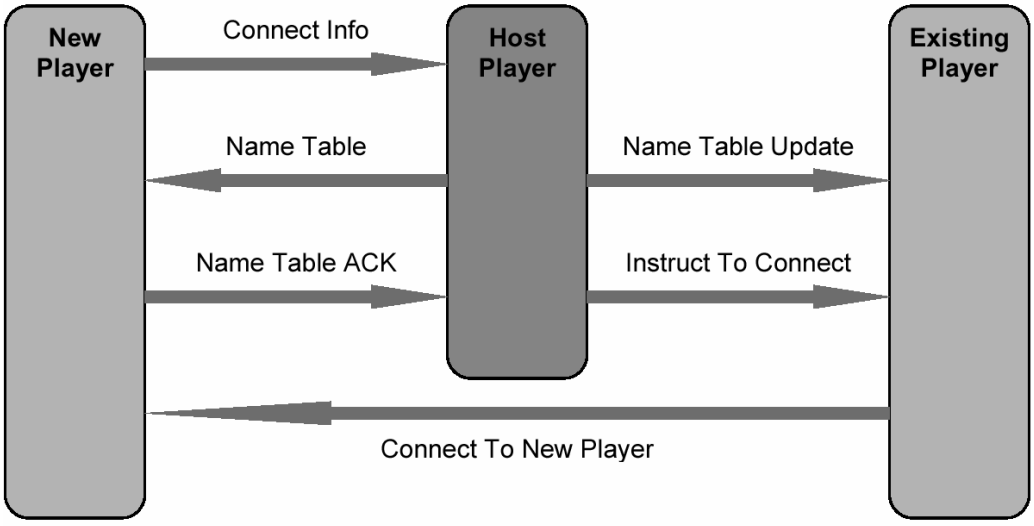


Figure 9.23.Distribution of the educational level in [Frit 2007.2].

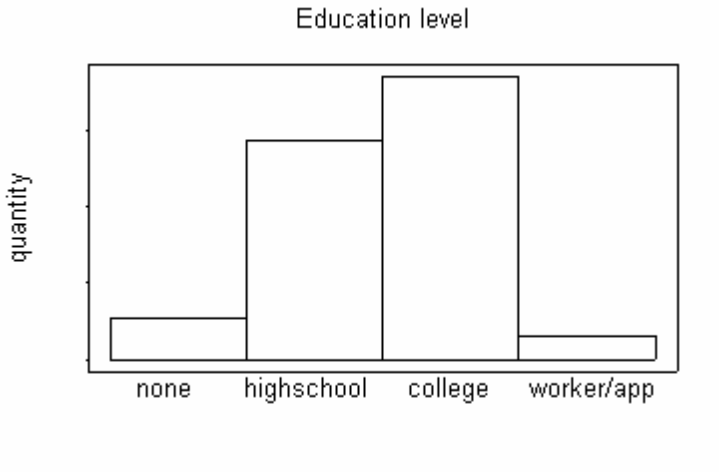
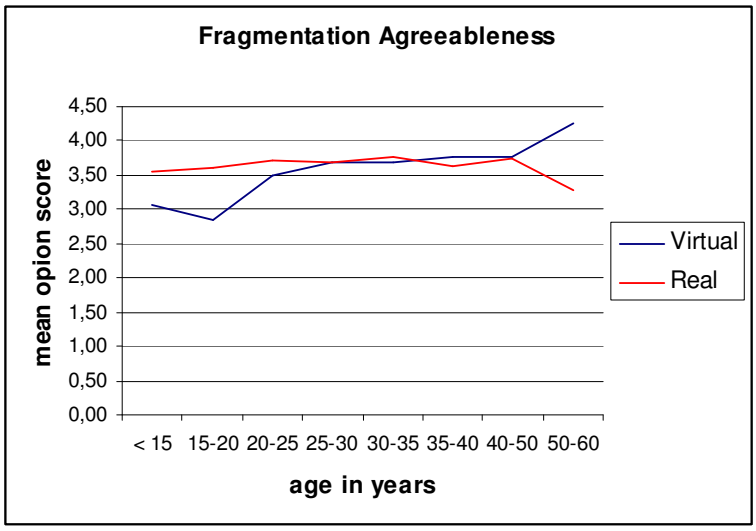


Figure 9.24.Virtual fragmentation for the factor agreeableness



10. References

"One person with belief is equal to a force of 99 who have only interests", John Stuard Mill.

[ADM 2001] Arbeitskreis Deutscher Markt- und Sozialforschungsinstitute e.V. : „Standards zur Qualitätssicherung für Online-Befragungen", In Proc. of ASI 2001.

[AOF] Age of Fantasy: http://www.panasiagames.com. (last checked o 9/20/2007)

[Agga 2005] S. Aggarwal, H. Banavar, S. Mukherjee, S. Rangarajan. "Fairness in Dead-Reckoning based Distributed Multi-Player Games". *In Proc of Netgames05*, Hawthrone, USA, October 2005.

[Akka 2004] A. Akkawi, S. Schaller, O. Wellnitz, L. Wolf. "A Mobile Gaming Platform for the IMS". In Proc. of Netgames04, Portland, USA, September 2004.

[Akka 2004.2] A. Akkawi, S. Schaller, O. Wellnitz, L. Wolf. "Networked Mobile Gaming for 3G-Networks". In Proc. of International Conference on Electronic Commerce 2004, Eindhoven, Netherlands, 2004.

[Assi 2006] M. Assiotes, V. Tzanov. "A Distributed Architecture for MMORPG". In Proc. of Netgames06, Singapore, October 2006.

[Bamf 2006] W. Bamford, P. Coulton, R. Edwards. "A Massively Multi-Authored Mobile Surrealist Book". In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Bare 2006] A. Barella, C. Carrascosa, V. Botti. "JGOMAS: Game-Oriented Multi-Agent System based on Jade". In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Bart] R. Bartle. “Player Taxonomy: Hearts, Clubs, Diamonds, Spades: Players who suit MUDs”, 1996, URL: http://www.mud.co.uk/richard/hcds.htm. (last checked 9/20/2007)

[Bart 2004] R. Bartle. “Pitfalls of Virtual Property”. Published by the Themis Group, 2004. http://www.themis-group.com/uploads/Pitfalls of Virtual Property.pdf (last checked on 9/20/2007)

[Beig 2004] T. Beigbeder, R. Coughlan, C. Lusher, M. Claypool. “The Effects of Latency on User Performance in Unreal Tournament 2003”. In Proc. of Netgames04, Portland, USA, September 2004.

[Bern 2001] Y. W. Bernier. “Latency Compensation Methods in Client/Server In-Game Protocol Design and Optimization”. In Proc of Games Development Conference 2001, Australia, February 2001.

[Bert 2006] C. Bertelsmeyer, E. Koch, A. H. Schirm. “A new approach on wearable game design and its evaluation”. In Proc. of Netgames06, Singapore, October 2006.

[Bliz] Blizzard Battle Net: http://battle.net. (last checked on 9/20/2007)

[Bogo 2004] I. Bogost. “Asynchronous Multiplay: Futures for Casual Multiplayer Experience”. In Proc. of Other Players Conference 2004, Copenhagen, Denmark, December 2004.

[Boul 2006] J. Boulanger, J. Kienzle, C. Verbrugge. “Comparing Interest Management Algorithms for Massively Multiplayer Games”. In Proc. of Netgames06, Singapore, October 2006.

[Cast 2002] M. Castro, P. Druschel, A. Kermarrec, A. Rowstron. “SCRIBE: A Large-scale and Decentralized Application-level Multicast Infrastructure”. In Proc. of IEEE Journal on Selected Areas in communications (JSAC), 2002.

[Cast 2001] E. Castronova. “Virtual Worlds: A First-Hand Account of Market and Society on the Cyberian Frontier”. In Proc. of CESifo Working Paper Series #618, 2001.

[Casu 2007] Analysis of the Casual Player Market: http://www.casualconnect.org/newscontent/11-2007/CasualGamesMarketReport2007_Summary.pdf (last checked on 9/20/2007)

[Ceci 2004] F. Cecin, M. G. Martins, R. O. Jannone, J. L. V. Barbosa. "FreeMMG: A Hybrid Peer-to-Peer and Client-Server Model for Massively Multiplayer Games". In Proc. of Netgames04, Portland, USA, September 2004.

[Cham 2005] C. Chambers, W. Feng. "Patch Scheduling for On-line Games". In Proc of Netgames05, Hawthrone, USA, October 2005.

[Cham 2006] C. Chambers, W. Feng, W. Feng. "Towards Public Server MMOs". In Proc. of Netgames06, Singapore, October 2006.

[Chan 2005] A. Chandler, J. Finney. "On the Effects of Loose Causal Consistency in Mobile Multiplayer Games". In Proc of Netgames05, Hawthrone, USA, October 2005.

[Chen 2006] K. Chen, K. Lei. "Network Game Design: Hints and Implications of Player Interaction". In Proc. of Netgames06, Singapore, October 2006.

[Chen 2006.2] H.Chen, H. Duh, P. P. S. Koon, D. Z. Y. Lam. "Enjoyment or Engagement: Role of Social Interaction in Playing Massively Multiplayer Online Role-playing Games (MMORPGS)". In Proc. of International Conference on Electronic Commerce 2006, Cambridge, UK, 2006.

[Chen 2006.3] K. Chen, C. Huang, P. Huang, C. Lei. "An Empirical Evaluation of TCP Performance in Online Games". In Proc. of ACE06, Hollywood, USA, June 2006.

[Chen 2006.4] K. Chen, J. Jiang, P. Huang, H. Chu. "Identifying MMORPG Bots: A Traffic Analysis Approach". In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Clay 2005] M. Claypool. “On the Turbulence of Nintendo DS and Sony PSP Handheld Network Games”. In Proc of Netgames05, Hawthrone, USA, October 2005.

[Cons] Console wars, online document: http://www.economist.com/business (last checked on 9/20/2007)

[Coul 2006] P. Coulton, O. Rashid, P. Garner. “’Getting it up’ with mobile virtual graffiti”. In Proc. of International Conference on Electronic Commerce 2006, Cambridge, UK, September 2006.

[Coun] Counter-Strike: http://www.counter-strike.net (last checked on 9/20/2007)

[Cram 2004] D. Cramer, D. Howitt. “The Sage Dictionary of Statistics”, 2004, ISBN 076194138X.

[DFC] DFC Intelligence: http://www.dfcint.com/game_article/ june04article.html (last checked on 9/20/2007)

[Diab] Diablo 2: http://www.blizzard.com/diablo2/ (last checked on 9/20/2007)

[Digi] Digitalgamedeveloper: http://www.digitalgamedeveloper.com/2002/11_nov/features/dlmmogd112602.htm (last checked on 9/20/2007)

[Digm 1990] J. M. Digman. “Personality structure: Emergence of the five-factor model”. Annual Review of Psychology, No. 41, pp. 417-440, 1990.

[Dire] Direct Play:

http://www.microsoft.com/presspass/press/1996/apr96/drctplpr.mspx (last checked on 9/20/2007)

[Dmoz] Online survey supplier: http://dmoz.org/Computers/Software/ Marketing/Surveys (last checked on 9/20/2007)

[Drus 2001] P. Druschel, A. Rowstron. “PAST: A Large-scale, Persistent Peer-to-Peer Storage Utility”. In Proc. of IEEE Journal on Selected Areas in communications (JSAC), pp. 75-80, 2001.

[Duch 2004] N. Ducheneaut, R. Moore, E. Nickell. “Designing for Sociability in Massive Multiplayer Games: an Examination of the Third Places of SWG”. In Proc. of OtherPlayers04, Copenhagen, Denmark, 2004.

[EA] Electronic Arts: http://www.ea.com/ (last checked on 9/20/2007)

[ElNa 2006] M. S. El-Nasr, S. Yan. “Visual Attention in 3D Video Games”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Ency] Encyclopaedia Britannica: http://www.britannica.com/ (last checked on 9/22/2007).

[Endo 2006] K. Endo, M. Kawahara, Y. Takahashi. “A Proposal of Encoded Computations for Distributed Massively Multiplayer Online Services”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Ente] Enter the Matrix: http://www.enterthematrixgame.com/ (last checked on 9/20/2007)

[EveO] Eve Online: http://www.eveonline.com (last checked on 9/20/2007)

[Ever] Everquest: http://www.everquest.com. (last checked on 9/20/2007)

[Ewen 1998] R. B. Ewen. “Personality: A topical approach”. Mahweh, NJ: Erlbaum, 1998.

[Fant] Fantasy World Rhynn: http://www.awaredreams.com. (last checked on 9/20/2007)

[Fran 2006] I. Frank, N. Sanbou, K. Terashima. “Some Positive Effects of Online Gaming”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Frit 2005] T. Fritsch, J. Schiller, H. Ritter. “Overviewing Scientific Research for (Mobile) Gaming”. Freie Universität Berlin, 2005. http://page.mi.fu-berlin.de/~fritsch/Overviewing_Scientific_Research_for_(Mobile)_Gaming.pdf (last checked on 9/20/2007)

[Frit 2005.2] T.Fritsch, H. Ritter, J. Schiller. „The Effect of Latency on MMORPGs“. In Proc of Netgames05, Hawthrone, USA, October 2005.

[Frit 2006] T. Fritsch, B. Voigt, J. Schiller. “Distribution of Online Player Behaviour. (How Hardcore Are You?)”. In Proc. of Netgames06, Singapore, October 2006.

[Frit 2006.2] T. Fritsch, H. Ritter, J. Schiller. “CAN mobile gaming be improved?“. In Proc. of Netgames06, Singapore, October 2006.

[Frit 2006.3] T. Fritsch, H. Ritter, J. Schiller. “Mobile Phone Gaming (A Follow-up Survey of the Mobile Phone Gaming Sector and its Users)”. In Proc. of International Conference on Electronic Commerce 2006, Cambridge, UK, September 2006.

[Frit 2006.4] T. Fritsch, H. Ritter, J. Schiller. “User Case Study and Network Evolution in the Mobile Phone Sector (A study on current mobile phone applications)”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Frit 2007] T. Fritsch, J.Schiller, C. Magerkurth. “Next Generation of In-Game Message Interfaces”. In Proc. of IUI 2007, Hawaii, USA, January 2007.

[Frit 2007.2] T. Fritsch, B. Voigt, J. Schiller. “Personal Behaviour and Virtual Fragmentation“. In Proc. of International Workshop on Intercultural Collaboration 2007, Tokyo, Japan, January 2007.

[Frit 2007.3] T. Fritsch, C. Magerkurth, B. Voigt, J. Schiller. “4MOG – Massive Multiplayer Middleware for Mobile Online Games“. In Proc. of International Workshop on Intercultural Collaboration 2007, Tokyo, Japan, January 2007.

[Game] GameDev discussion thread. (last checked on 9/20/2007)

http://www.gamedev.net/community/forums/topic.asp?topic_id=319003

[Gams] Gamespot: http://www.gamespot.com (last checked on 9/20/2007)

[Garn 2006] P. Garner, O. Rashid, P. Coulton, R. Edwards. “The Mobile Phone as a Digital SprayCam”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Gaut 1998] L. Gautier, C. Diot. “Design and Evolution of MiMaze, a Multi-Player Game on the Internet”. In Proc. of the IEEE Conference on Multimedia Computing and Systems, p.233, 1998.

[Gonz 2005] J. Gonzalvez. “Mobile Radio: Industry Forecasts and Surveys”. In Proc. of the IEEE Vehicular Society News, Vol. 52, No.1, pp. 29-30, August 2005.

[Hamp 2006] T. Hampel, T. Bopp, R. Hinn. “A Peer-to-Peer Architecture for Massive Multiplayer Online Games”. In Proc. of Netgames06, Singapore, October 2006.

[Hand] Handango: http://www.handango.com. (last checked on 9/20/2007)

[Harm 2004] P. Harmjanz. “Conceptual Evaluation of Current Handheld Multiplayer Games”, diploma thesis, Freie Universität Berlin, 2004.

[Hend 2001] T. Henderson. “Latency and User Behaviour on a Multiplayer Game Server”. In Proc. of Networked Group Communication: Third International COST264 Workshop, London, UK, November 2001.

[Heis] Heise online report from Ericsson: http://www.heise.de/newsticker/meldung/86908/ (last checked on 9/20/2007)

[Higg 2001] J. Bartlett, J. Kotrlik, C. Higgins “Organizational Research: Determining Appropriate Sample Size in Survey Research”. URL: http://www.osra.org/itlpj/bartlettkotrlikhiggins.pdf (last checked on 6/20/2008)

[Hsia 2005] T. Hsiao, S. Yuan. “Practical Middleware for Massively Multiplayer Online Games”. In Proc. of IEEE Internet Computing Vol. 9, Issue 5, pp.47-54, September 2005.

[ID] ID software: http://www.idsoftware.com/ (last checked on 9/20/2007)

[IPho] Apple iPhone: http://www.apple.com/iphone/ (last checked on 9/20/2007)

[Izai 2006] T. Izaiku, S. Yamamoto, Y. Murata, N. Shibata. "Cheat Detection for MMORPG on P2P Environments". In Proc. of Netgames06, Singapore, October 2006.

[Jako 2003] M. Jakobsson, T. Taylor. "The Sopranos Meet Everquest". In Proc of DAC03, Melbourne, Australia, 2003.

[Jane 2005] A. Janecek, H. Hlavacs. "Programming Interactive Real-Time Games over WLAN for Pocket PCs with J2ME and .NET CF". In Proc of Netgames05, Hawthrone, USA, October 2005.

[Jamb] Jamba: http://www.jamba.de (last checked on 9/20/2007)

[JavaD] JavaDoc: http://java.sun.com/j2se/javadoc/faq/index.html (last checked on 9/20/2007)

[Kabu 2005] P. Kabus, W. W. Terpstra, M. Cilia, A. P. Buchmann. "Addressing Cheating in Distributed MMOGs". In Proc of Netgames05, Hawthrone, USA, October 2005.

[Kane 2005] Y. Kaneda, H. Takahashi, M. Saito, H. Aida. "A Challenge for Reusing Multiplayer Online Games without Modifying Binaries". In Proc of Netgames05, Hawthrone, USA, October 2005.

[Klas 2006] L. Klastrup. "Death Matters: Understanding Gameworld Experiences". In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Kuma 2000] V. Kumar (2000): International Marketing Research, New Jersey: Prentice Hall.

[Lazz 2006] N. Lazzaro. "Why We Play Games: Four Keys to More Emotion Without Story". In Proc. of International Conference on Electronic Commerce 2006, Cambridge, UK, September 2006.

[Lee 2006] K. J. Lee, S. C. Ahn, H. G Kim. “Using a Mobile Device as an Interface Tool for HMD-based AR Applications”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Lehd 2005] V. Lehdonvirta. “Real Money Trade of Virtual Assets: Ten Different User Perceptions”. In Proc. of Future Play05, Michigan, USA, October 2005.

[Lehm 2006] S. Lehmann. “Studienarbeit: Finden von mobilen Spielen“, Freie Universität Berlin, 2006. URL: http://page.mi.fu-berlin.de/fritsch/Studien.pdf (last checked on 9/20/2007)

[Li 2006] S. Li, C. Chen. “Interest Scheme: A New Method for Path Prediction“. In Proc. of Netgames06, Singapore, October 2006.

[Liu 2006] L. Liu, H. Ma. “Wireless Sensor Network Based Mobile Pet Game”. In Proc of Netgames06, Singapore, October 2006.

[Lu 2006] F. Lu, S. Parkin, G. Morgan. “Load Balancing for Massively Multiplayer Online Games”. In Proc. of Netgames06, Singapore, October 2006.

[Mann 2000] T. Manninen. “Interaction in Networked Virtual Environments as Communicative Action: Social Theory and Multiplayer Games”. In Proc. of CRIWG - 6th International Workshop on Groupware, Madeira, Spain, 2000.

[Max] Max Payne: http://www.rockstargames.com/maxpayne/ (last checked on 9/20/2007)

[McCr 1996] R. McCrae, P. Costa. “Toward a new generation of personality theories: Theoretical context for the five-factor model of personality”. From Theoretical Perspectives (pp. 51-87), New York, Guilford, 1996.

[Merr 2006] K. Merrick, M. L. Maher. “Motivated Reinforcement Learning for Non-Player Characters in Persistent Computer Game Worlds”. In

Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[MMOR] MMORPG.com. http://www.mmorpg.com (last checked on 9/20/2007)

[Mobi] Mobile Games Blog: http://www.mobilephoneblog.org/2007/03/mobile-game-revenue-in-us-hits-151.htm (last checked on 9/20/2007)

[Mönc 2006] C. Mönch, G. Grimen, R. Midtstraum. "Protecting Online Games against Cheating". In Proc. of Netgames06, Singapore, October 2006.

[Muel 2006] F. Mueller, L. Cole, S. O'Brien, W. Walmink. "Airhockey Over a Distance – A Networked Physical Game to Support Social Interactions". In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Myer 1990] D. Myers. "Computer Game Genres". In Proc. of Play&Culture 3, pp. 286-301, 1990.

[Mysi 2006] S. Mysirlaki, F. Paraskeva, N. M. Sgouros. "Socio-cognitive learning perspectives on the impact of violent videogames on Greek adolescents". In Proc. of International Conference on Electronic Commerce 2006, Cambridge, UK, September 2006.

[Neve] Neverwinter Nights: http://www.neverwinternights.com (last checked on 9/20/2007)

[Nint] Nintendo DS: http://www.nintendods.com/ (last checked on 9/20/2007)

[Noki] Nokia Webpage: http://www.nokia.de (last checked on 9/20/2007)

[Open] Open Mobile Alliance: http://www.openmobilealliance.org. (last checked on 9/20/2007)

[Okan 2004] P. Okanda, G. Blair. "OpenPING: A Reactive Middleware for Construction of Adaptive Networked Game Applications". In Proc. of Netgames04, Portland, USA, September 2004.

[Park 2006] Park-Associates survey:

http://www.parksassociates.com/press/press_releases/2006/gaming_pr4.html (last checked on 9/20/2007)

[Pell 2003] J. D. Pellegrino, C. Dovrolis. “Bandwidth requirement and state consistency in three multiplayer game architectures“. In Proc of Netgames 2003, Redwood City, USA, 2003.

[Pell 2005] R. Pellerin, F. Delpiano, F. Duclos, M. Simatic. “GASP: an open source gaming service middleware dedicated to multiplayer games for J2ME based mobile phones”. In Proc. of CGAMES’05, Angouleme, France, 2005.

[Petr 2005] L. Petrak, O. Landsiedel, K. Wehrle. “Framework for Evaluation of Networked Mobile Games“. In Proc of Netgames05, Hawthrone, USA, October 2005.

[Rows 2001] A. Rowstron, P. Druschel: “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”. In Proc. of 18th IFIP conference on e-commerce, Heidelberg, Germany, 2001.

[Seha 2006] K. Sehaba, P. Estraillier. “Game execution control by analysis of player’s behaviour”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Shai 2004] A. Shaikh, S. Sahu, M. Rosu, M. Shea. “Implementation of a Service Platform for Online Games”. In Proc. of Netgames04, Portland, USA, September 2004.

[Shel 2003] N. Sheldon, E. Girard, S. Borg, M. Claypool. “The Effect of Latency on User Performance in Warcraft III”. In Proc. of Netgames03, Redwood City, USA, May 2003.

[Shel 2004] N. Sheldon, E. Girand, S. Borg, M. Claypool. “The Effect of Latency on User Performance in Warcraft III”. In Proc. of Netgames04, Portland, USA, September 2004.

[Smed 2002] J. Smed, T. Kaukoranta, H. Hakonen. “Aspects of Networking in Multiplayer Games”. Tukuru Centre for Computer Science, 2002.

[Smed 2004] J. Smed, H. Niinisalo, H. Hakonen. “Realizing Bullet Time Effects in Multiplayer Games with Local Perception Filters”. In Proc. of Netgames04, Portland, USA, August 2004.

[Smit 1982] B. C. Smith. “Procedural Reflection in Programming Languages”. PhD Thesis, MIT, MIT Laboratory of Computer Science Technical Report 272, Cambridge, 1982.

[Sing 2004] A. Singh, A. Acharya. “Using Session Initiation Protocol to Build Context-Aware VOIP Support for Multiplayer Networked Games”. In Proc. of Netgames04, Portland, USA, September 2004.

[Sims] The Sims: thesims.ea.com (last checked on 9/20/2007)

[SoE] Sony Online Entertainment: www.soe.com (last checked on 9/20/2007)

[Sony] Sony PSP: http://www.sony.com/ (last checked on 9/20/2007)

[Stat] Statistik Austria, Internet access distribution in Germany: http://www.statistik.at/web_de/static/haushalte_mit_internetzugang_2007_022207.pdf (last checked on 9/20/2007)

[Tang 2005] L. Tang, J. Li, J. Zhou. Z. Zhou. “FreeRank: Implementing Independent Ranking Service for Multiplayer Online Games”. In Proc of Netgames05, Hawthrone, USA, October 2005.

[Taru 2006] H. Tarumi, K. Yokoo, S. Nishimoto. “Open Experiments of Mobile Sightseeing Support Systems with Shared Virtual Worlds”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Thee 2007] The ESA, American game market analysis: http://www.theesa.com/facts/index.asp (last checked on 9/20/2007)

[Trib] Tribes 2: http://www.planettribes.com/tribes2/ (last checked on 9/20/2007)

[Unde] Undercover 2: Mercenary Wars:

http://www.undercover2.com/main.php (last checked on 9/20/2007)

[Ulti] Ultima Online: http://www.uo.com. (last checked on 9/20/2007)

[Vill 2006] N. Villar, K. M. Gilleade, D. Ramduny-Ellis, H. Gellersen. “The VoodooIO Gaming Kit: A real-time adaptable gaming controller”. In Proc. of International Conference on Advances in Computer Entertainment Technology 2006, Hollywood, USA, June 2006.

[Vik 2006] K. Vik, C. Griwodz, P. Halvorsen. “Applicability of Group Communication for Increased Scalability of MMOGs”. In Proc. of Netgames06, Singapore, October 2006.

[Vlee 2005] B. Vleeschauwer, B. Bossche, T. Verdickt, F. Turck. “Dynamic Microcell Assignment for Massively Multiplayer Online Gaming.” In Proc of Netgames05, Hawthrone, USA, October 2005.

[Wiec 2002] H. Wiechers. “Fallstudien zur organisatorischen Gestaltung der Softwareentwicklung bei deutschen Computerspielherstellern“. Diploma thesis in computer and business science, University of Cologne, 2002.

[Yama 2005] S. Yamamoto, Y. Murata, K. Yasumoto, M. Ito. “A Distributed Event Delivery Method with Load Balancing for MMORPG”. In Proc of Netgames05, Hawthrone, USA, October 2005.

[Yan 2005] J. Yan, B. Randell. “A Systematic Classification of Cheating in Online Games”. In Proc of Netgames05, Hawthrone, USA, October 2005.