The Architecture of Massive Multiplayer Online Games

Sladjan Bogojevic, Mohsen Kazemzadeh

September 8, 2003

Department of Computer Science
Lund Institute of Technology, Lund University
Box 118
S-221 00 Lund
Sweden

Abstract

The purpose of this master thesis has been to investigate the architecture of massive multiplayer online games, MMOGs. As broadband is spreading around the world as well as powerful PCs are available, more computer graphics can be applied in MMOGs. The users of today want to see more graphics, i.e. more graphics features results in more users involved in the game.

The established knowledge from the investigation was used for implementing a part of the International Football Manager game, the IFM game. The implemented part of the IFM game consists of a graphical platform for showing the highlights from a certain football match.

Design techniques concerning MMOGs are discussed, including the game balancing for MMOGs. The investigation regarding networking aspects of MMOGs is addressed, where main areas have been inquired such as the client/server operations needed for the programming of a MMOG.

Computer security aspects as well as the importance of database systems of MMOGs are brought up. Industry analysis added by a market research for MMOGs are discussed in detail. Recommendations for future work regarding the nature of this master thesis are made.

Acknowledgments

We would like to thank our supervisor, Professor Lennart Ohlsson, for his continual encouragement, help and guidance throughout this work.

We would also like to thank the administrator and developer of the International Football Manager game, Berhan Karagöz, for sharing his knowledge and for his constant support during our master thesis work. Berhan Karagöz is also the director of the BK Data Company and responsible for the release of the International Football Manager game, the IFM game.

Furthermore, we wish to thank Martin Stankovski and Sanja Bogojevic for giving us feedback on the report.

We are very thankful to "EA SPORTS FIFA Football 2003" game for giving us the inspiration needed for our feature implementation within the IFM game.

Finally, we would like to thank our girlfriends Sladjana and Heli and especially our parents for their support and understanding.

Sladjan Bogojevic, Mohsen Kazemzadeh
Lund Institute of Technology
September, 2003

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Purpose

MMOG stands for Massive Multiplayer Online Game. MMOG were first named as MMORPG, i.e. Massive Multiplayer Online Role Playing Game and many companies still use that term. This is the kind of online game that allows thousand of players to play simultaneously, from all parts of the world. Players which play these kinds of games are able to stay in the game over longer time of period which is typically 6-12 months. During this time they are able to develop their own game style. Even if a player stops playing the game the servers will still be online with other players. Online games are a blooming market with many opportunities as well as challenges for game developers.

The purpose of this thesis project has been to investigate the architecture of MMOGs. We have both studied general principles for games already which exist on the Internet and as a concrete case chosen to study the International Football Manager game, the IFM game, in more detail. To gain insight in how animated graphics should be included in a MMOG, we also implemented a graphical platform for a new feature in the IFM game, showing highlights from a certain football match.

Design techniques concerning MMOGs are being discussed in this thesis project, where it is clarified what audience of different kind, expect out of MMOGs. How these should be designed according to the audience type is also discussed. Under the very same topic the architecture of the MMOGs in general is being presented along with the architecture of the IFM game where a comparison between these two can be made.

Networking aspects regarding MMOGs is addressed where main areas as the client/server operations, needed for the programming of a MMOG, are revealed. Issues like bandwidth limits and refresh frequency are brought up, as they play an important part in the design of a MMOG itself, affecting the maximum amount of data that can be sent over the Internet.

The importance of computer security aspects of MMOGs are also brought up. According to the current situation in the world, where hackers are the dominating threat to the computer based industry overall, security is a pressing issue for those who are interested in the development of a MMOG. Vulnerabilities of MMOGs are therefore addressed, as effective solutions concerning these are given.

As database systems are used for storing and maintaining data for MMOGs a separate chapter is used for illustration of their functionality and usage. The importance of a structured database is also given as it could be extremely time-saving.

Industry analysis added by a market research for MMOGs are being discussed in detail as it is of great importance having an idea of how much a MMOG costs to develop and for example what game categories that one should consider having its game developed within.

In the history of MMOGs, the first games were of role playing type with poor graphical presentation. This was not because of the non interest from the developers' point of view, but simply because of the bandwidth limit. Computing power had also become cheap enough, in the mid 1990's, so that companies could build enough powerful PCs, devoted to computer graphics.

Nowadays, when broadband connections are increasing by number around the world as well as powerful PCs available for reasonable prices, more computer graphics can be applied. This is yet another reason why the MMOGs can be found in additional genres.

Even back in the days when the arcade games were most popular the main thing that attracted players was the graphics in the game. Even then a simple rule was applied among players, the better graphics within the game the more players will the game attract.

As the situation is at the IFM game today, no animation graphics are available, generally because it is a manager game which mainly consists of statistics and tables. During our education at Lund Institute of Technology we have learned how to use computer graphics in real time games and the investigation itself gave us a knowledge overview on why graphics is important in MMOGs. The users of today want to see more graphics, i.e. more graphics features results in more users involved in the game. Experimental implementations within the IFM game have been done. The accomplished implementation includes a platform for future graphical representation of the feature, for showing the highlights from a certain football match. Everything from physics needed to the artificial intelligence that has been implemented in the IFM game feature is being described.

A comparison between 3D programming tools is also made, where the established result can be used as a guidance for those who someday would like to develop a MMOG on their own. Finally, recommendations for future work regarding the nature of this master thesis project are made.

1.2 Disclaimer

This master thesis was created and written by Sladjan Bogojevic and Mohsen Kazemzadeh. The information within the master thesis was obtained from publicly available sources, including company Websites, company annual reports and news sites dedicated to games.

First of all, only 3 weeks of the total amount of 20 were used for the IFM game feature implementation. We have, according to our own opinion, managed to produce very satisfying results. It is obvious that we would be able to produce much more as a matter of programming code if we would have all the 20 weeks for programming to our disposal. We shouldn't forget though, that our research has given us tremendous insight in what in the end of the 20 week period came out to be a wide scientific area.

Chapter 2

MMOG Design Techniques

2.1 Creating and Designing MMOGs

When creating and designing MMOGs it is necessary at the early stages of development to get an overview and thereby also good understanding of what the goal should be. General topics as whom the game should be designed for and what the main purpose of the game should be are a few questions that are going to be discussed in this chapter. Concurrently, the IFM game is going to be used as a case study for comparing these aspects.

The IFM game as a MMOG can be classified as something between sport strategy and multiplayer role game. The IFM game could contain several game worlds but the status of the IFM game today is that there exists only one world. A game world includes 200 countries and their international teams where each country has 4 divisions with 12 teams in each division. This results in 9.600 teams totally. As 9.600 teams are the maximum capacity of one world, a new world could be created by the IFM game if this would be necessary. Another solution would be additional divisions for each country, in case of user overload.

2.2 Encouraging Cooperative Play in MMOGs

The expression "building community" has become one of the key goals of a successful MMOG design development over the past several years. There are three reasonable arguments regarding encouraging cooperative play which are listed below:

- *Firstly*: Cooperative play reduces system overhead. If a server can handle a maximum of 3.000 concurrent players and each player can fight one monster alone, the server must be capable of handling 3.000 concurrent beasts. This includes also for example their AI routines, collision detection etc. If the number of players that are required to defeat one enemy instead increases to three, only 1.000 concurrent monsters must be provided.
- *Secondly*: Cooperative play modes increase a game's variability which in turn leads to longer time where players will reduce the likeliness of being bored. The longer time the players stay in the game, the more monetary gain the game will collect. Another thing that is worth mentioning is that players will probably bring more friends and family to their game world, which will also benefit the game economically.
- *Thirdly*: People playing MMOGs in general often build social bonds, which eventually will spread outside the game. Even if the players quit playing a certain MMOG the people they used to play with usually remain their friends outside the online society.

Unfortunately in the IFM game there are no monsters to kill. The IFM game's system capability has to face another type of scenario. This scenario is about taking care of the football match simulations which need to be calculated and presented as quick as possible. This involves calculations needed for every match played in the world.

These calculations are going to take approximately 15 minutes independently of how many players that are playing in the current world.

Concerning the second and the third argument mentioned earlier, players often build social bonds outside the game. Every successful game on the market has some sort of chat system. The developer's of the IFM game have already in the early stage of development thought of having a chat room that would let players communicate with each other and thereby create social bonds and even exchange experience concerning the game.

Yet another chat room slightly different in design, is to be developed for the IFM game. This chat room will be of virtual type and referred to as a virtual pub where managers could meet and discuss tactics. In the virtual pub area a table could be reserved within the private chat room, where up to ten managers could chat with each other. The managers in question are going to be seen on the screen in form of 3D characters, representing themselves.

These added features will hopefully, in the end, result in more players playing the IFM game which will also mean that there will be a economical benefit to the game in addition to creating a pleasurable atmosphere for all of those playing the game.

As the IFM game is implemented today, only one player can manage a certain team. In the future there are plans on letting 20-30 players manage one team together in the IFM game. This will of course lead to more frequent communication between players where each and everyone are going to have a special role in the team they choose to represent. Those roles could consist of being a trainer, marketing representative, financial manager etc. This improvement of the game will fulfill the third argument already described earlier.

2.3 Building MMOGs for the Masses

Creating a MMOG has some practical challenges. Since there are different types of players existing one should keep in mind also that there are basically two different extremes of gamers when designing a MMOG. The two different types are:

- *The mass audience*: Gamers who often are not skilled or experienced e.g. kids.
- *Hardcore gamers*: Gamers with experience and good game knowledge.

The games for a mass audience should be pleasant and fun to play. But the truth is that to create an interesting game, some kind of conflict is required. The challenge is to make the conflict meaningful and important to players, but not so intense that it alienates crucial segments of the audience.

Since hardcore gamers have a higher level of patience, the excitement when playing the game is not a primary factor, i.e. the difficulty level should be challenging from the start. Hardcore gamers' preconceived thinking about the implications is also different comparing with the mass audience gamers.

If the complexity is initially hidden from the mass audience and the difficulty level is sufficiently low at the start, the players will support long playing game designs.

The philosophy of the development for the mass audience games should be never to let the framework get in the way of a player having fun playing the game. The following illustrate some helpful rules regarding the application of game principles for mass audience games.

- Let the excitement begin right away
- Avoid forcing a player to make uninformed choices
- Game play should dictate geography
- Play session can and should be short
- Allow and encourage players to be spectators for learning tactics
- Teamwork is often more rewarding than individual accomplishments

It is important to keep who the players are and to not forget the design principles while designing the games. The IFM game is developed for players with experience of manager games. So, those players with experience who join this game will know what to expect. However, there is still hope for newcomers. There is a fully documented online manual and tutorial for the game so even beginners are able to play. After a while anyone should be able to manage the game but of course the strategy of how to play is something that comes with some serial game playing.

In IFM's case the players who participate in the game enthusiastically are going to be rewarded for it. Since the IFM game has its own newspaper, every user is able to write his/hers own articles and possibly get them published in the newspaper. Every article is studied by the administrator where only the best ones get published. For every published article the player gets awarded with one point. At the end of a season the three best players who have the highest amount of points, get to play another season for free.

2.4 Game Balance for MMOGs

Balancing in an MMOG environment is a process that begins at the early stages of development. MMOGs should theoretically be played hundreds or even thousands of hours by each player. Action of one player in MMOGs is capable of affecting others which means that imbalance in the game itself could indirectly affect the entire community. This makes balancing an extremely important issue that should be handled at the early stages in development.

Yet another important part of effective game balancing is game metrics. Game metrics are methods by witch game values are measured and recorded. By using these resulting data decisions, on how future balance of the values should be set, can be made.

The following illustrates common factors to track in MMOGs for balancing purposes:

- *Player advancement*: By measuring a player's progression through the game the designer is enabled to determine if advancement is occurring at the expected rate.
- *Item use*: The item use measurements are very useful for checking the balance of items such as weapons, armors etc. Having a situation where one certain item is used 100 times more than the subsequent one, there is a clear hint that the items are unbalanced.
- *Actions*: What do players spend their time on while playing MMOGs? Is it fighting monsters, recovering from previous losses or do they spend time on something completely different? It is therefore important to track players for representing the best possible balance.

When it comes to the IFM game an administrator is set to monitor the action and thereby correct the balance if necessary. The IFM game does not have a monitor program as a tool for helping the administrator to keep the balance in check. However, a monitor program is planned to be implemented in the near future.

An unbalanced situation that could occur in the IFM game is that a statistically stronger team would continuously beat a statistically weaker team. On the other hand, an inexperienced player that keeps loosing while playing with more experienced players is not considered as an unbalanced situation. In this situation the inexperienced player needs to learn more about the game tactics. This could be done by exchanging tactics tips with other players playing the game, within the IFM game's chat room or simply go through the online manual once again.

It is common to add new features to an online game after launch. These additions are first of all an excellent way to maintain and even increase interest in the game. They also carry the same dangers as rebalancing. Even a small feature that seems to be of small importance can have a large effect as it interacts with all of the other game systems. The feature that will show highlights of a football match will have a very small effect when it interacts with other parts of the game. This feature will increase game pleasure but it will not disturb the balance of the game as such.

2.5 The Architecture of MMOGs

Developing an MMOG is a task which is often underestimated. The architecture of a MMOG often becomes quite complex. It is important to notice that there is not any single best or standard solution. In the figure 2.1 a functional, highly simplified MMOG architecture is shown.

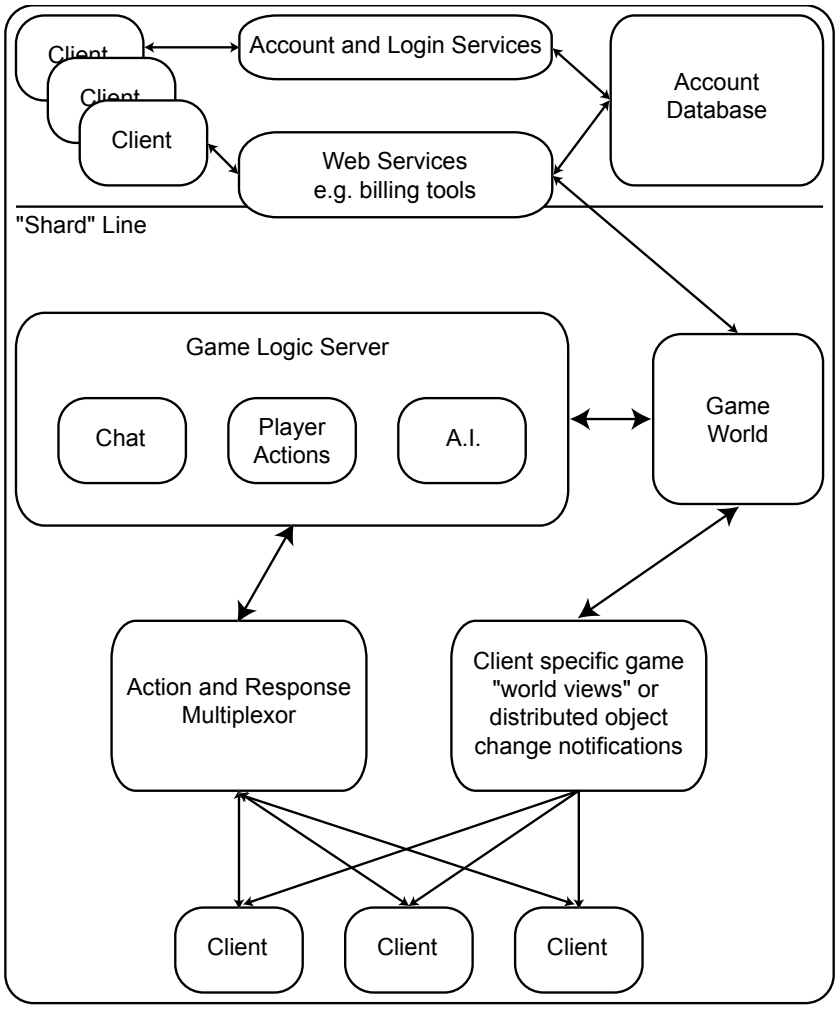

Figure 2.1: Simplified MMOG architecture

Having the highly simplified example of a MMOG architecture represented in figure 2.1 at our disposal, the figure is going to be explained in general terms, from bottom up.

The first layer consists of clients who request different actions from the Game Logic Server segment. The clients can be seen as input to a MMOG where they are ported to the Game World segment.

When a client performs an action it is queued by the Action & Response Multiplexor segment which is located in the second layer. This data will be forwarded to the Game Logic Server segment. The information exchange is performed similarly, when responses and confirmations are returned from the Game Logic Server segment. The information exchange can also stream via the client's view of the game world.

The third layer in the figure 2.1 includes the Game Logic Server and the Game World segments. Depending on what kind of implementation has been applied, memory, database or some combination of these two, it can be used to store the Game World data. As there is different kind of data, like the one describing important transactions or the one describing non critical data different techniques needs to be applied. Non critical data, e.g. player skills should be stored in an SQL database while important transactions, e.g. positions and health should be hold in memory to reduce processing overhead. Considering the Game Logic Server segment as such, it mainly provides tasks associated with the Game World segment and the Action & Response Multiplexor segment. One of the tasks is managing the game world model and the other one is providing additional services to players such as chat, artificial intelligence that is AI, NPC, etc.

The fourth layer handles the Web interfaces for account management. This layer is becoming standardized as more and more games are using the Web services to deliver information such as customer service tools, consisting of client code and billing to the Game World segment. It is worth mentioning that billing and other customer service tools are hard tasks to manage and are unfortunately often neglected until late in development.

2.6 The Architecture of the IFM game

In the IFM's case clients login through the so called Account and Login Services to access the game environment, i.e. Game World Visuals. Game environment uses the Game Database in turn, to display relevant data from the Game Database. Then there are the Game World Update Tools that keep the data up to date and control the live matches which can be seen using the Game World Visuals.

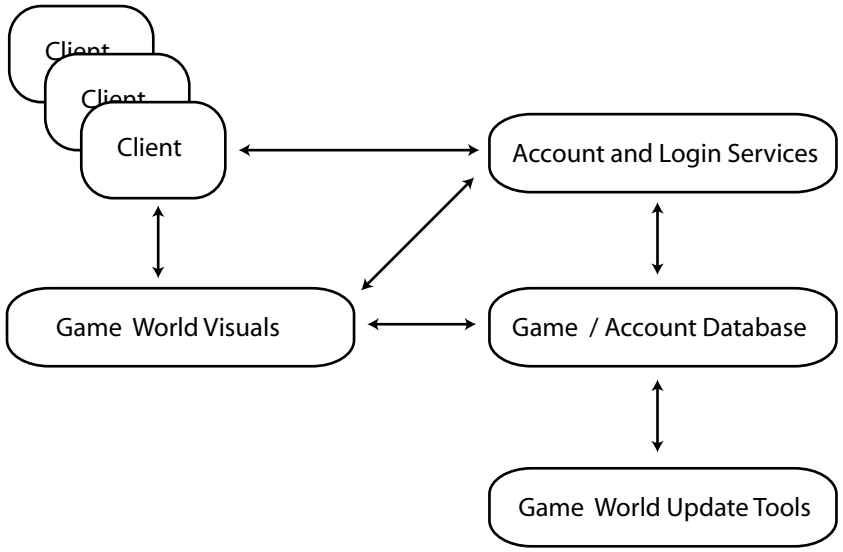

Figure 2.2: Simplified architecture of the IFM game

If the demand for playing the IFM game would increase so that the existing capacity of 9.600 clients would not be enough, new game worlds would need to be set up. In that case the IFM game could use a common account login server and so called "director" to refer the clients to appropriate game world servers.

2.7 Platforms for MMOG Development

Most MMOGs on the market today are what is called platform specific. The games have, for example, been developed using the C++ language for Windows. Standard programming environments such as Visual C++ along with DirectX have been used.

On the other hand programming languages such as Java and the increasingly used Python are worth considering, especially if a development team has an established expertise, or if a chosen middleware is based on these platforms. It is important to notice that the difference in language performance is minimal when compared to the differences that efficient design and careful optimization can make to performance.

To accelerate MMOG development and at the same time make it a bit easier, there are companies offering complete MMOG network solutions and software infrastructure, i.e. tools and middleware.

A few examples of companies that provide network solutions and software infrastructure are presented below in alphabetical order:

- Big World Technologies
- Butterfly.net
- Global Gaming Innovation
- Nevrax
- Quazal
- Silver Platter Software
- Terraplay
- Touchdown Entertainment
- Turbine
- Twisted Matrix Laboratories
- Zona

None of the above mentioned companies have yet launched a commercial MMOG game based on their technology except for Turbine. One reason could be the fact that the above mentioned companies have not been on the market for a long time as it takes approximately four years for a MMOG to be developed.

Chapter 3

Networking Aspects of MMOGs

3.1 Introduction to Networking

The worlds largest network, Internet, is the network that MMOGs take advantages of. For using its capacity efficiently, one needs to investigate the theory of the Internet and how the Internet works.

In a network two or more computers use a language to communicate and share information with each other. The language in this network is called the protocol of the network. This means that computers who want to communicate with each other must use the same protocol standard e.g. Transmission Control Protocol/Internet Protocol, TCP/IP or User Datagram Protocol, UDP. The protocol that is used in Internet communications is the TCP/IP.

A protocol defines the format of the data and the rules for communication. This implies that different computers that even run different types of operating systems can share information as long as they are connected to the network.

3.2 Differences between TCP and UDP

The TCP is a reliable protocol that guarantees packet delivery and is for that matter responsible for verifying the correct delivery of data from client to server. The risk is that data sent from client to server can be lost in the intermediate network. The protocol adds support for retransmitting the data until either a timeout condition is reached or until successful delivery has been achieved. If the sending computer is transmitting too fast for the receiving computer, the protocol will activate the flow control mechanisms to slow down the data transfer. It is worth mentioning that these are just some of the services that the protocol has to offer.

The UDP protocol does just about as little as a transport protocol can, in other words it offers only a minimal transport service. The protocol is mainly used for applications that do not want TCP's flow control and wish to provide their own. The protocol is an unreliable one in a way that it does not guarantee delivery nor does it require a connection. All error processing and retransmission, on the other hand, must be taken care of by the application program. The UDP protocol is generally better for real time interactions.

The IP handles the movement of data between host computers. It provides among other things error reporting of information. It is important to observe that the IP only delivers the data packets between host computers. It's up to another protocol, for example, the TCP to put them back in the right order. Both TCP and UDP are connected to the IP protocol.

3.3 Client/Server Programming

The server's task is to wait for an incoming request of connection from client/clients. Once a connection is established the server proceeds by serving the client/clients in question. The structure and functionality of a server and a client are very much different and will therefore be discussed in this section.

3.3.1 Server Methods

If there are no client/clients to serve, then there is not much for a server to do. It is therefore very important to keep in mind when creating a server that it sometimes can be a very passive application. It is simply a waste of CPU time letting a server constantly run an empty loop when there are no client connections.

There two types of servers, iterative servers and the concurrent ones. The main difference, between these two types, is that the iterative ones can only serve one client connection at a time. However, iterative servers are not very useful.

Assume that there is an iterative server handling a MMOG. This would not be so appropriate since this type of server can only handle one client connection at a time, which may put the rest of the clients on hold.

A concurrent server handles several clients at a time in an appropriate way. When this kind of server is started, its first task is to listen for incoming connections requested by clients. Now, the thread for listening for incoming connections requested by clients will instantly create a child process, a thread. Thereafter the main thread will continue to listen for incoming connections.

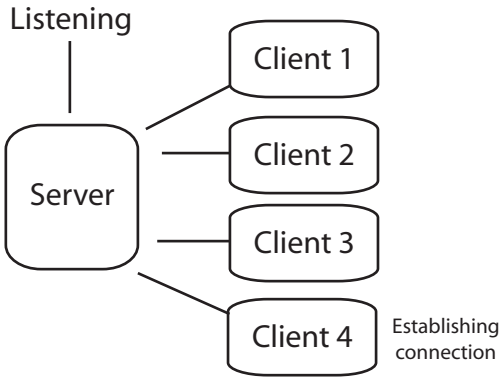

Figure 3.1: Simplified representation of a concurrent server

The figure 3.1 shows how the concurrent server works. Depending on the state of the application it will successively accept all the connections requested from the clients. This will continue until it reaches an upper limit of possible clients that it can take care of.

In the figure 3.1 one can see that clients 1-3 have already established a connection with the server. Client 4 is on the other hand still establishing one.

It is important to add that a server uses a socket for listening and in figure 3.1 this socket is already free and has started the listening process for more connections. A socket is basically a pipe which data flows through.

3.3.2 Different Types of Sockets

In the industry two kinds of sockets are available, A stream socket is a connection oriented socket which means that a connection must be established before data exchange.

A datagram socket is on the other hand a connectionless socket which means that whenever there is any data to send, the data in question is sent to a predefined address. This implies that data in a datagram socket is transmitted in datagrams. If one of the hosts crashes the others will not be able to notice this. This fact makes it more challenging for the developers to create the application.

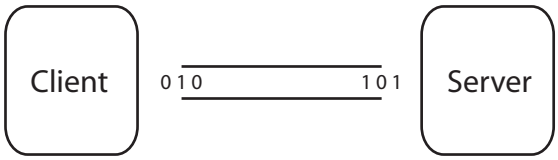

Figure 3.2: A stream socket connection

The stream socket types are used in the TCP technique and the datagram sockets types are used in the UDP technique.

3.3.3 Clients

Clients are the ones initializing the connection to the server. Clients only wait for data from the server to be received. In MMOGs the clients are normal users which can be seen as part of the game itself. The clients are only interested in knowing the address and the port number of the server. In some MMOGs this information is built in to the game, so that the players need for setting up the connection is taken care of automatically. This is implemented in the IFM game, i.e. when a player/client uses the login and password, the connection to fsdata Company is established automatically. The fsdata Company is a so called Web hotel that provides the IFM game concerning computer security, database management, data storage area, etc. for a low monthly cost.

3.4 Client/Server Operations

In this section a few TCP and UDP client/server operations are going to be discussed.

3.4.1 An Easy Transport Protocol

TCP is an easy transport protocol for developers to manage as it is for creating TCP servers as well. The only thing that has to be done is for the servers to listen for incoming connections and accept them. After that, data between the server and client units can be exchanged. At the end, if a client requests to close the connection to the server, a request is sent to the server and it is thereafter confirmed by the server. The figure 3.3 illustrates these procedures in time.

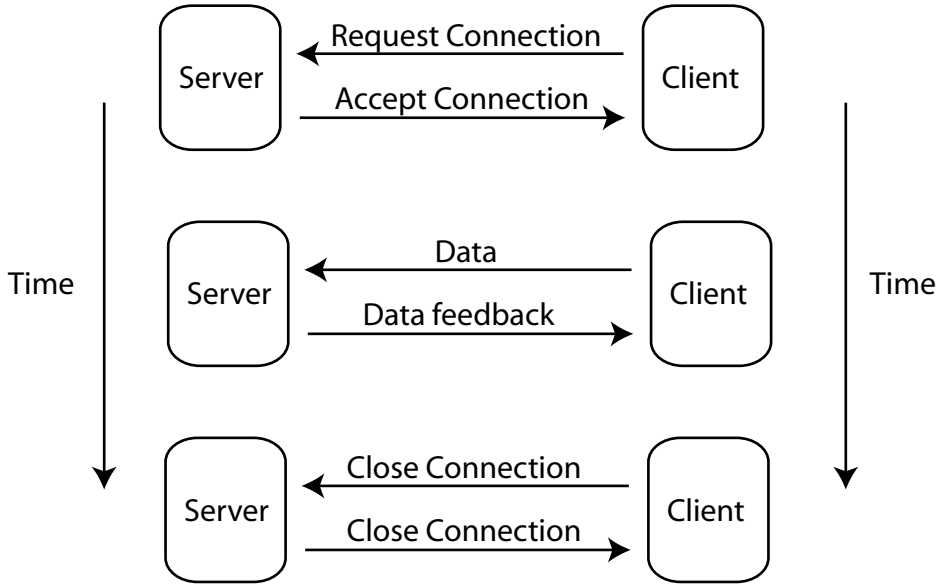

Figure 3.3: A normal TCP client/server operation

3.4.2 A Connectionless Protocol

As mentioned earlier, TCP servers have to listen for incoming connections. Dealing with UDP servers, one does not need to listen for incoming connections since UDP is a connectionless protocol.

This type of model just reads the incoming datagrams and acts depending on what information it holds and how the programmer wants it to act. A client host could simply notify a server by sending a datagram that holds relevant information about the client.

This kind of system is called a knocking system where a "knock" message is sent in the beginning. When the server receives this message it will update its list of clients or just create a new child process depending on the design. A problem may occur when there are lots of clients who want to connect to the server. Then the socket may be overloaded which would lead to delays on the client sides. That is why the child process technique is to be preferred.

Dealing with UDP servers, one server only uses one socket to interact with the clients. To be able to exchange data, the hosts must know about their address information. The figure 3.4 shows a normal UDP client/server operation.

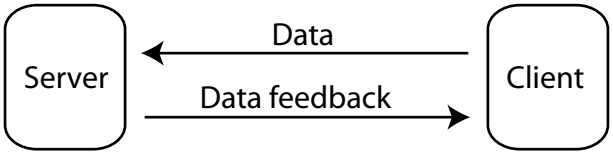

Figure 3.4: A normal UDP client/server operation

3.5 Server Side Development

In this section the basic issues considering the development of MMOGs on the server side will be discussed. The development on the server side is extremely important and should be considered through early stages of game design. There are many options, procedures and potential solutions that will have consequences throughout the life of the game.

3.5.1 Seamless Servers

MMOGs usually have to manage thousands of players concurrently. Because of this, they require a server environment that should be distributed in such way so that it is able to spread all game related computation across not only one but several host machines. One of the most usual ways of doing so is by splitting the game world into different regions and thereafter letting them be managed by different server processes.

In an environment such as the one where different regions are managed by different server processes, the game world itself can be categorized as either seamless or zoned. The difference is in whether the server process boundaries are directly observable inside the game or not.

One could say that the seamless world is one where a player without knowing may interact with objects that are in turn handled by several game processes. Zoned world on the other hand are geographically independent areas. Between these areas there is often very little or no contact at all. In that case players interact only with other objects on the very same server.

The decision on whether to implement a seamless or zoned world will have effect on every single part of a game's development and is therefore of extreme importance.

3.5.2 Partitioning the Server Load

Breaking the game world up into pieces simply results in better balance because it spreads various types of game processing equally across server machines. By doing so, it is going to be easier to add more game content without impacting current server loads.

There are several different solutions of partitioning the server load. To split the game simulation along the geographical lines and thereafter assign them to multiple servers is one of the most common solutions.

As there are other solutions, what other ways of partitioning the server load for a certain game design could be obtained? The answer to this question is that it all depends on the game design itself and the number of players populating the MMOG in question. These factors are the ones controlling the amount of computations being executed.

The following solutions are some of the possible ones:

- Splitting physics and game computation
- Letting artificial intelligence run free
- Spend more money

The alternative solutions that are going to be discussed in more detailed form, correspond to methods for speeding up the CPUs.

3.5.3 Splitting Physics and Game Computation

Latest MMOGs are most often 3D worlds. Handling a 3D world involves large amount of time updating the character movement, executing collision detection and updating other physics related activities in a 3D world as well.

One possibility would be to separate the updating of the character movement and the performance of collision detection from one another. These could thereafter be handled in separate severs.

As it is shown in the figure 3.5, what is chosen to be called the physics server would handle the movement and collision and thereafter update what is chosen to be called the game server.

A scenario could contain clients connecting to a front end server. The very same server would then redirect the packets to either the physics or the game server. Redirection is depending on the type of the data packet in question.

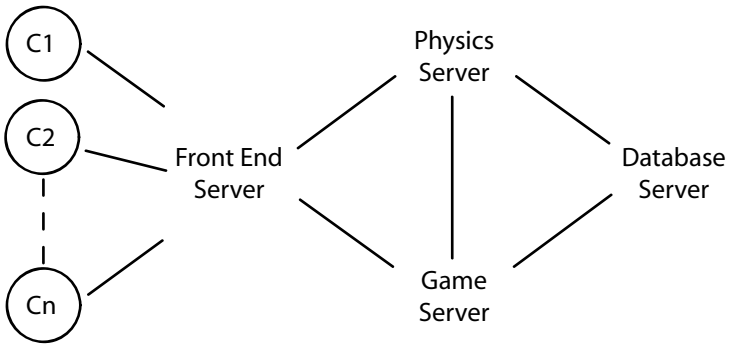

Figure 3.5: The architecture of a server with separate physics and game servers

3.5.4 Letting Artificial Intelligence Run Free

One could partition the server load by simply letting the Artificial Intelligence, AI processing be handled on its own server. Unfortunately this also causes some negative effects which must be handled.

First of all, data must be replicated since AI decision making most often relies on the properties of, for example, player characters and other game objects.

Assume a scenario where the decision making of a character is based on the position of another game object. If the position of the game object in question is not updated concurrently because of badly scaled bandwidth, the AI processing will get out of synchronization. Synchronization must be applied as well as increased communication between the servers.

If replication is not carefully managed, obviously the code in question could get out of synchronization. The last mentioned facts will most certainly lead to abnormal behaviour and bugs in the code that would be very hard to find.

3.5.5 Spend More Money

At the end one could always invest more money in better and more powerful server machines. By doing so, one could make use of multiprocessor boxes and multiple threads. Making use of multiprocessor boxes is equivalent to increasing the CPU power and the processor clock speed.

3.5.6 Benefits of Seamless World

As time goes by the world will become more populated. In a seamless world the servers can be adjusted to handle the increased load. There are two different types of server boundaries, the dynamic and the static ones. The dynamic server boundaries are able to perform the balancing of the increased load at run time, by itself. As a matter of static boundaries the balancing of the increased load can be adjusted during down time.

Another great benefit of the seamless world is its potential for higher reliability. The server boundaries can be adjusted to spread the load over the remaining servers if one of the server machines would break down or a server process would crash.

An interesting detail is that a dynamic server boundary implementation might automatically spread the load over the remaining servers when it detects a server crash.

The time required for map loading is greatly decreased as smaller parts of the world are loaded in, only when required which in turn has a great affect on the clients in the end.

3.6 What Data Needs to be Sent?

When a player, that is a client gets his/hers login and password, there will be several types of data sent forward and backward between him/her and the server. This also counts for the rest of what could be thousands of MMOG players. All kind of data sent between the client and the server needs to be taken care of differently, depending on how time critical it is.

Some examples of common interactions in a MMOG, generated by players or by non player characters, NPCs could be combat, movement, chat or trade. Taking combat as a triggered interaction it is far more time critical and highly prioritized, for that matter, than chat or trade interaction. To be able to give a player a good sense of continuity and game feeling as well, movement information does not need to be more than medium prioritized interaction. This is why interactions like the combat one, should constitute a large portion of a data stream.

Taking the IFM game as a case study along with the feature comprehending the visualization of the football match highlights, interactions slightly differs from other MMOGs. In IFM's case the client/clients use so called "pull technique" for retrieving the information needed. When a client wishes to perform an action, a request is sent to the server which will thereafter serve the client in question.

3.7 Bandwidth Limits

There are two reasons why one kilobyte per second, which games on the market all share, is a good bandwidth measurement per player. The first one is that even a low speed modems like those that has a rate of 28.8k are able to keep up with the data rate of one kilobyte per second per player. The second one is, considering one kilobyte per second bandwidth measurement of which MMOGs are charged for leads to the fact that even if players would keep on playing a MMOG constantly for a month, the game would still be profitable. In other words there would be money left from their monthly subscription for paying MMOG maintenances as well as ongoing expenses.

3.7.1 Refresh Frequency

The refresh frequency itself affects the average amount of transferred data between the hosts. Independent of what network delivery protocol is used, TCP or UDP, a header will be attached to the data packet. These headers vary in size, which are 28 bytes for a UDP package and 72 bytes for a TCP package.

Assume that a frequent refresh is being made. This means that the extra header data will also be sent along with the rest of the package. Ideally, the sending of a package should include all the available information concerning a player. Even if the packet in question should include several different types of information concerning a player, it would be ideal to send it over the network as one packet.

As mentioned earlier the TCP is used for exchanging the information between the hosts via the Internet while the IFM game is making use of this functionality. A scenario could appear where both TCP and UDP techniques are used in combination at a same time for sending data packages. In this case, data that needs to be sent should be broken up in two subpackages. One sub package should include all TCP data merged into one package and the other send should include all UDP merged data into another package.

As a developer of a MMOG one will always attempt to send the information needed from server to its clients as often as possible. This might seem like a good idea but in some cases this consumes extra bandwidth unnecessarily. By doing so one would not only consume extra bandwidth but one would make it easy for the players to notice lag, i.e. delays, when they occur during playing a MMOG. However, these lags could be used in an inappropriate way by cheating clients.

On the other hand, not sending the data packages frequently emphasizes some serial visualization issues. These issues could include visual popping and warping of the characters. Warping could be recognized from a client point of view if an object hops from location to location instead of, for example, gliding. Yet, another issue regarding infrequently sending data packages is that these will increase in size.

Consider an example which explains what happens when a large amount of data needs to be sent over the network.

This large amount of data could for example be split up in four smaller data amounts that could each be put into a data packet. After sending these from, for example, a server to a client, assume that one gets lost. Losing a part or any part of a great amount of data, that could be representing some kind of a message, would result in this whole message being thrown away and resent again. This will in turn once again result in unnecessary bandwidth consumption.

3.8 Data Storage Required for Each Player

To be able to maintain and manage the clients belonging to a specific server, the server must first of all have a predefined storage area per client. In IFM's case there is information such as last time a client logged in for playing the MMOG, last time a client won a football match, last time a client bought a certain player, etc. This information has to be stored somewhere on the server. This is where the supervision administrator keeps track of the game history. This information must be managed in a proper way.

In IFM's case, thousands of players could be involved in the game. Some players are active ones and others are not. As there could be thousands of players to manage, proper use of every space of memory available becomes extremely important.

Assume that a certain client's subscription period has expired. Any detailed information about this client is now unnecessary data to keep stored. This data can now be handled in several different ways. It could be stored separately from the server's storage area or recycled within it. Before these actions can be executed a certain procedure must be followed.

A client whose subscription has expired will be contacted via an email. The client is going to be notified of the expired subscription and at the same time be encouraged to pay for another one. If the player after being contacted does not reply to the email sent to him/her after a predefined period of time, the account of the client in question will be closed. The supervisor will then save the data information by removing it outside the server memory as the initial condition and structure of the team is being reset. The information being removed consists of the team history during the period the client was an active user of the MMOG.

In order to save memory space at the chat handling part on the server, a compress technique will be needed for minimizing the storage data. One solution could be applying a method of encrypting every word to a "key_id" consisting of letters and numbers.

3.9 Handling of Game Events

Because of the great number of users causing all sorts of things to happen at any given time, the events are more important in MMOGs than ever. Event based programming handles concurrency in processes without using threads.

Comparing threads with events one can say that threads are more difficult to use because other processes might interrupt the current thread at any given time. This requires more out of the application itself, e.g. mutual exclusion which means that only one process can access the shared data at a time. Threads are also hard to debug and they are often hard to schedule.

On the other hand using events, one does not need to worry about any process interrupting the current one. Additionally neither the mutual exclusion needs to be considered. When the application invokes the events they execute and run until they have completed their task. Event based programming is therefore much easier to understand, debug and schedule. It is important to observe that using event based programming does not exclude the usage of threads for other parts of the MMOG.

In IFM's case the usage of events is applied. There is a main loop whose purpose is reading changes in the output of the game engine. The game engine creates among other things also actions and events depending on a team's current status. Pseudo code for the main loop already described, which also could be found in the IFM game could look like the following one:

```
mainloop()
{
  while(game_is_running)
  {
    // wait for incoming requests
    waitForRequests();

    // handle requests
    processRequests();
  }
}
```

Once an event is generated it will be stored in the game server register. As a client requests to visualize a certain highlight of the game, the event is broadcasted to him/her. There are about thousands of different events. An event can, for example, contain information about player A at position P1 passing to player B at position P2 where from player B scores by performing a so called "shoot with his right foot" action.

The broadcasted event is then used for creating the 3D environment showing the requested highlight. The creation of the environment is actually an activeX component which is done in the clients RAM memory.

3.10 Movement Prediction

This section essentially discusses techniques for movement prediction on client side in MMOGs. Although this subject is not applied in the IFM game, the prediction itself is a broad and important issue to consider.

The players' movement perception of the characters in a MMOG will have an affect on how deep the players will involve themselves in the game. An example of problems that may occur is when characters move through closed doors or boxes. These problems are common in all networked games, nevertheless in MMOGs.

Client side movement prediction has to be considered from two points of views. First of all, the players' requirement of immediate feedback has to be considered as well as the variance of network delays. An obvious thing is that players have a small waiting tolerance when they perform an action until the action itself is displayed on their screen. Clients, in most cases, do not care to think about that every action performed by a player has to be computed at the server and rebroadcast to the player in question.

There are lots of ways to deal with immediate user feedback. A simple solution might be to give an audio signal to the client which indicates that the order has been received while the order itself is passed to the server. This signal is used to camouflage the time delay.

Another trick is to initiate a command delay into an action signal at the client side. The predefined delay is used to reduce the expectation of immediate user feedback at the client side. However, this solution is considered to be disturbing for hard core gamers.

Although the above mentioned solutions can be considered as proper ones, there are factors that might not be controlled. In most cases for MMOGs, the client side is being constructed in such way that it is expecting to be updated in a constant rate, normally 4 to 20 times per second.

Since the Internet uses TCP protocol for data package delivery where the delays are not predictable, a situation may occur where a package gets delayed or lost. In case of a lost package, the package will be redelivered. If the data package in question contains position data of an object, extreme amounts of variance in the time between position updates will contribute to warping.

3.11 Movement Prediction Techniques

Following subsections will illustrate some useful techniques concerning wrapping and popping of characters. As mentioned earlier these problems are generated because of network latency. However, these techniques are presented below:

- Command time synchronization
- Extrapolation
- Reversible simulation

3.11.1 Command Time Synchronization

In certain MMOGs where objects have to be moved to specific positions based on path finding requests, the command time synchronization technique is suitable.

These MMOGs are in general of real time strategy type. The problem appears when the client A wishes to move an object to a specific position. This request is first of all sent to the server. Due to network latency rest of the clients will receive this information when the client A's object has already been moved a bit.

To be able to compensate for this, the other clients must receive client A's request as well, so that the moving object gets to it's final location at the right time based on the latency of the move command from the server.

Sending the message "Would you be kind and move object O to position P" from server to all of its clients, it will result in a scenario where object O will reach its final position differently depending on how far on the average, the clients are located from the server.

A better solution would be applying the command time synchronization technique which serves the clients by sending the message "Would you be kind and move object O to position P at time T in the future nearby". Assuming that the time is synchronized between the hosts and that the speed of the object in question can be regulated in such way that it reaches its final position at each client in correct time.

3.11.2 Extrapolation

In the figure 3.6 a line and a curve are shown. The straight line represents what the client knows about the traveling of the object in question. The curve, on the other hand, represents the turn that the object is actually making on the server.

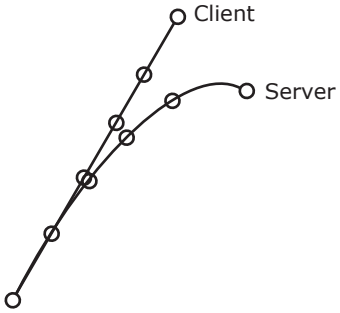

Figure 3.6: Server's view and client extrapolation of an object position

Each dot in the figure 3.6 represents a unit of time. If the client does not receive the updated position within a predefined time range, the extrapolation technique is going to be applied. The equation is simple, the longer the server waits until sending the updated position to a client, the further out of the position the two objects will be.

A technique that uses extrapolation method is the "dead reckoning" technique. Dead reckoning technique is a form of client side prediction. This technique is widely used in MMOGs that distribute positional data and it is a way of compensating for latency. Dead reckoning as such, could be described as a technique that will decide where an object should be positioned "now" based on where it has been positioned "before" and which velocity it had.

Making it more understandable, assume for a moment that it is time to move an object and that the updated position for the very same object has not arrived yet. The dead reckoning technique then simply estimates the new position. It is important to notice that this last mentioned estimate of the new position is often based on where the object was last time and how fast it was moving. The acceleration at the previous position is yet another fact that is sometimes used for estimating the new position of the object.

The usage of the dead reckoning technique requires that the designers and programmers of a MMOG work together to be able to make the best choices getting out the most of the technique. It will for example make a great difference in movement of an object, which is clearly going to be seen at the end the final result is presented, if the simulation time is used in the calculations of dead reckoning technique, or not.

So, what could happen if simulation time is not applied? Well, because of the latency, the updated position for the object in question could be in the past. The position at that time could already be estimated. This will result in the object suddenly popping to the position that has been already guessed, in the past. Figure 3.7 shows an example of this where it clearly can be seen that the object will suddenly pop, so to say, from guessed position to its true one.

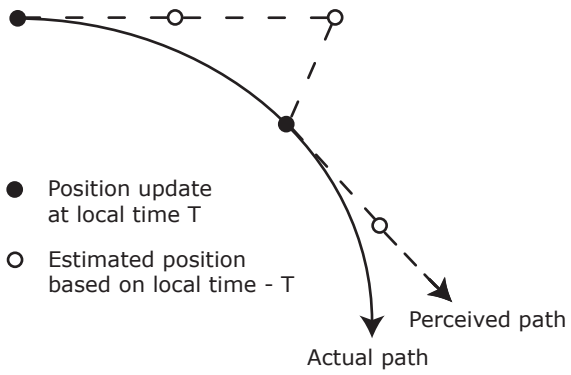

Figure 3.7: Basic dead reckoning

Now, considering the difference between the time of the update and the current simulation time, the scenario described in figure 3.7 could be avoided. In figure 3.8 it can be seen that the displayed path does not get back on the actual path. Instead, the perceived path is repredicted based on the difference between the time of the update and the current simulation time. This will result in getting smoothness of the object movement.

It is on the other hand depending on the type of the MMOG in question what kind of dead reckoning technique is more suitable. In some cases the true path of an object is of absolute importance, then considering the difference between the time of the update and the current simulation technique is not to be preferred.

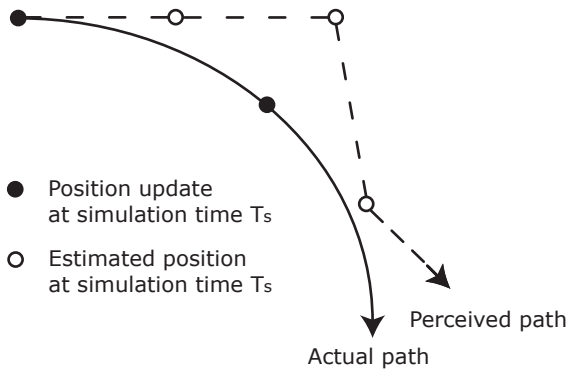

Figure 3.8: Dead reckoning with time compensation

Not only does it depend on what type of the MMOG designers and developers are working on, every object in the game does not need to play by the same rules. This is why considering the usage of the dead reckoning technique is not just a technique that is implemented by the programmers. It requires that the designers and programmers of a MMOG discuss and work together before applying it at all.

3.11.3 Reversible simulation

This technique is used in MMOGs where shooting a target occurs. In these games a scenario might occur when a client shoots at an object that it thought had at its sight. Network latency would delay the shooting command getting to the server. This targeting object may now be at another position on the server by the time the trigger was pulled. What happens is that the client misses the target, even though the target was in its sight. The figure 3.9 will help visualize this scenario.

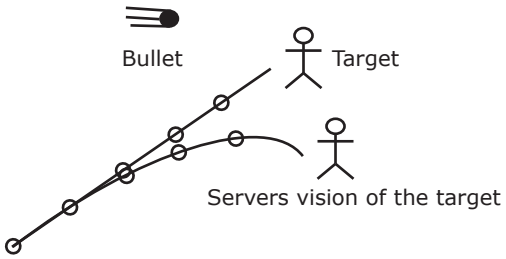

Figure 3.9: Differing client and server views leads to missed shot

Taking advantage of reversible simulation will however, lead to fair play. The idea is that the server goes back to the last acknowledged command from the targeting object i.e. the position where the object was insight when being shot. The server will then start to extrapolate what would have happened as if the target had not moved at all.

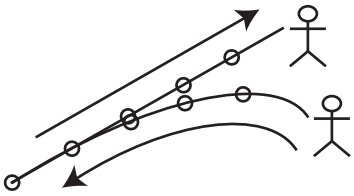

Figure 3.10: Using reversible simulation to change from server to client view

The server will simulate a scenario were commands are received in the right order from the clients. The figure 3.10 shows that in this case the target is placed at the correct position when being shot simultaneously as the client makes it's shooting correct. This is then confirmed by the server who then eliminates the target.

Chapter 4

Computer Security Aspects of MMOGs

4.1 Computer Security in General

In short, one can state that security is all about the protection of assets. This type of definition implies that one has to have an accurate evaluation of the assets that he/she have to protect. There are three different types of rough classifications of protective measures that have to be considered. These protective measures are:

- *Prevention*: taking measures preventing assets being damaged, for example locking a doors and window bars, thus making it more difficult for a burglar to break into ones home.
- *Detection*: allowing one to detect when an asset has been damaged, how it has been damaged and who has caused the actual damage.
- *Reaction*: enabling one to recover his/hers assets.

4.2 To be Able to Play MMOGs

To be able to play MMOGs requires that you first of all have an approved CD Key, username and a password. Once you got your username, password and you have a correct CD Key you will then have to pay for a monthly subscription.

The information about people who would then like to play MMOG will, apart from personal information such as approved CD Key, username and password, be stored on the company's server for necessary verification.

4.3 Why Would Security Services be Hacked?

When it comes to MMOGs and security, implementations of these can be realized in two different ways. One solution would be having the company developing the MMOGs take care of all the security aspects by itself. The other option is letting a company, who is already specialized in computer security for MMOGs, secure the computer system in question.

The latter solution is often preferred from a developer company's point of view. Then companies do not need to take care of security implementation at all but instead their time can be spent on e.g. implementation of game logics.

It is essential to add that the service for the security of MMOGs is often not very expensive. This conclusion is based on the fact that when comparing the amount of time developers at a certain company need to spend on programming from scratch simultaneously as well as maintaining the system by themselves, it is much cheaper to allow a company specialized on computer security to set up the security system for the intended program.

Following is a list of possible motives for hackers to compromise a security system:

- Companies which provide the security services withhold private information about people/players.
- People often pirate games, which they would like to play online. To play online a client is most often forced to give an approved CD Key to be able to do so. CD Key has to be approved so that the server would be able to set up a clients account, therefore they are obviously one of the goods that hackers are keen to put their hands on.
- If a client, as a player, wants to join a certain MMOG he/she must pay a fee for playing the online game in question for a predetermined time period. To be able to pay the fee the client will probably use a credit card number, which is another reason why companies need to provide security services so that this information does not get uncovered by hackers.
- Hacking the companies that provide the security services also means obtaining valuable information so that hackers can cheat ahead.
- To be able, for example, to delete other peoples characters etc.

4.4 The Cryptography Issues of MMOGs

Cryptography is the science of secret writing. To encrypt information that is sent from a player when he/she is logging in to play a MMOG, is one thing that has to be done. To thereafter encrypt every little message that is sent between a player and the server, that holds MMOG information would be devastating. Full encryption encapsulation would therefore be detrimental to the industry.

A part of the nature of MMOGs is that there are thousands of users who are always online. These players are not interested in how the implementation of the MMOG that they are playing is done. Cryptography exaggeration would lead to a poor updating rate in the game which in turn will result in players abandoning the game.

4.5 Vulnerabilities of MMOGs

MMOGs in general are often exposed to client hacks. Players are constantly looking for ways to take advantage of the system. Using "how to cheat" sheets posted on different Websites, cheaters are stealing accounts, hacking into the servers, duplicating items, weapons and even whole characters.

In the following subsections vulnerabilities of MMOGs are pinpointed followed by a description and possible solution.

4.5.1 Server Masquerading

Problem

One variety of a server masquerading is "server spoofing". Server spoofing is when attacker poses as a game server. This allows attacker to get usernames and passwords and the attacker may also be able to get CD Keys.

Yet another attack is called "man in the middle" attack and implies that attacker accepts the connection and passes credentials on to the server. This will allow the attacker to take part of valuable information as mentioned in the case of server spoofing. When information is sent from client to server, it will usually be sent via several different computers before reaching its final destination. The "man in the middle" attack is in most cases caused by lack of server authentication protocols.

Solution

Digital certificate system should be introduced so that no one else but the server, to whom the information is devoted, could take part of the information. The digital certificate system must be implemented in such way so that it has to be changed often.

In general use, a certificate is a document that attests to the truth or ownership of something. A digital certificate is a digital document that serves the same purpose.

Assume Client A using his/her private key, decrypting a message that has been encrypted by his/her public key. Client A can therefore prove that he/she owns the public key. The digital certificate confirms that the owner of that particular public key is Client A. Thus the digital certificate binds Client A and his/hers public key in a sort of digital ID Card. There are three main practical uses for digital certificates:

- to prove identity for purposes of electronic commerce
- to prove identity for purposes of access control
- to prove identity to prevent spoofing

4.5.2 Client Masquerading

Problem

Client masquerading is when a player has the role of an attacker acting as a valid user. This case allows the attacker unauthorized access to service and to obtain or modify client information. This problem is for the most time a result of poor password requirement of service.

Solution

Solution to this problem would be to introduce stricter password requirements. In combination with these stricter requirements, letting people who want to play a MMOG make notice when choosing a password is also part of a solution. To choose a word from dictionary to have as password is inadvisable, as the password then becomes an easy prey for the attacker. By forbidding dictionary passwords, dictionary attacks are effectively prevented.

Another solution to the problem would be letting the company provide each user with a password, which they can guarantee is relatively hard to crack. Giving password to a certain user should not be based on this user's username, password, or CD Key.

4.5.3 Distributed Denial of Service, DDOS

Problem

A "denial of service" attack is characterized by an explicit attempt by attackers to prevent legitimate users of a service from using that service. For those players who have bought a game for playing online the game server will be unavailable.

The main reason why MMOGs are vulnerable to this attack is because they must accept connections from any client at any time. A "denial of service" attack could therefore result in huge monetary losses to companies.

Solution

A solution to this problem would be to introduce routers. Routers would come to play a role when attackers are preventing a user from using a service. Their responsibility would be to direct the user to another direction so that the usage of a certain service could be possible.

A solution to the problem could be downloading patches that automatically, by installing them on a user's machine, could enable the user to get a service that he/she required. Introducing network monitoring tools is a solution where a server is simply supervising the whole network in case sudden changes would appear, like for example sudden disappearances of many users etc.

4.5.4 Cheating

Problem

One could ask oneself what the expression cheating actually means and involves. The expression cheating could be defined as using a service in an unintended way.

For example making oneself invincible and progressing in the game without completing certain steps is considered cheating. Those players that might witness this kind of cheating will most certainly complain until cheating is fixed which will certainly cause expensive problem solving. If the cheating is not fixed players could decide to leave the game, which leads to a loss of income.

Alongside the outright cheating and hacking, MMOGs have also been having problems with issues concerning gray market activities. One of the side effects that arise is grey market activities providing monetary gains for the player who cheats by selling game items obtained in inappropriate ways. These game assets or so called virtual game assets could be swords, magical spells or characters, which are sold in open Internet auctions, such as on eBay. Internet auctions are places where the gray market activities are taking place and "virtual goods" can be sold for real money.

A key argument against Internet auctions are for example when someone buys the account of a powerful character but does not really understand how to play the game effectively, then he/she can get himself/herself into trouble and may ruin the play experience for others.

This could lead to an unbalanced game. Cheating is a vulnerability that is caused by various bugs in the game and lack of testing during game development.

And then there is the case of fraud, where a player buys a virtual item with real money and does not receive the item. The player usually attempts to have the people running the service to help them, which makes it impossible tracing the fraud, since they have no way to verify the transaction.

MMOGs are also vulnerable to system attacks not only for the normal type of hackers that most often want to cheat themselves through the game but also for malicious hackers. A definition of a malicious hacker is simply a hacker who wants to destroy the game for all other players playing the same game as the hacker himself/herself.

When it comes to the IFM game, cheating scenario could occur when a client of the IFM game for example has registered himself/herself as owner of two different teams. To own two different teams doesn't really constitute a problem itself. Problem occurs when a client tries to build up one of the teams he/she owns by selling football players to the other team that has the very same owner for an extremely low price. After this procedure he/she could sell these players to the teams he/she does not own and in that way build up a strong team and earn money while doing it.

Solution

Developers must make security a top priority, ensuring the effectiveness of firewalls and securing the databases containing personally identifiable information on subscribers.

Storing critical information on server is one solution and to not allowing client to modify directly is yet another. To implement efficient patching system and fix bugs as soon as they are found is also a part of solution.

Developers must pay special attention to this kind of cheating and develop better ways to control the situation. One of the latest ideas is that as a developer support a blacklist of troublemaking players shared on an industry wide basis.

Solution to the cheating problem at the IFM game could be solved by having a monitor program, which the administrator uses, for signaling when heavily unbalancing is taking place. By noticing this signal the administrator can easily solve the problem by suspending the player who from then on is seen as a cheater.

4.5.5 Server Control of Client Systems

Problem

How much of the control should a server be able to have over the client systems? Problems that are security related may occur when a game server forces a client to perform action without user consent.

For example, a server could have an auto download of client updates. Then a situation could occur where server is causing client software to scan user's machine for other software etc. The phenomenon mentioned is caused because of implicit trust of game servers by clients.

Solution

File verification is one possible solution and includes a maneuver of file checking and asking the user of the machine before performing any actions without user consent. Server masquerading issues should be handled by limiting implicit trust of servers in general as well as implementing an intruder detection system, which would be able to give away a signal when intruders get detected.

4.5.6 User Modified Client Software

Problem

The case where attacker modifies client software and adds functionality to it could appear as well. This could lead to a scenario where the attacker can be able to exploit flaws in server system.

Solution

Even this problem contains storing critical files on server i.e. moving client files to secure servers. Introducing client verification would reduce the vulnerability as well. Then there are potential privacy issues, how much of the information about the user should be revealed anyway?

4.5.7 Computer Security Aspects of the IFM game

The IFM game is using a Web hotel, managed by the fsdata Company, for enabling its players to play their game.

Advantages with a Web hotel are presented below:

- The IFM game does not have to pay the gigantic cost, which it would have to pay had the game been loaded and driven from the IFM game itself.
- The IFM game does not have to worry about the firewall. Firewall is used as a common name for any security system protecting the boundary of an internal network.
- The IFM game does not have to worry about backing up the system constantly.
- The IFM game leaves all the computer security aspects regarding MMOG itself to the fsdata Company who is specialized in security issues and offers its services for a low monthly cost.
- The IFM game uses FTP for communication between its development computer and the Web hotel at the fsdata Company for updating and diverse adjustments.
- The IFM game can set up optional number of users with individuals usernames. This is done when a user registers himself/herself and has paid his/hers fee.

Chapter 5

Database Systems of MMOGs

5.1 Databases in General

A database is nothing more than a collection of information arranged in some meaningful way that exists over a long period of time, often many years. Databases are today essential to every kind of business. Databases are often used to maintain internal records, to present data to customers and clients on the World Wide Web and to support many other commercial processes. Yet, another area for using databases is for storing and maintaining data, for MMOGs.

Knowledge that has been developed over several decades and that today is representing the true power of databases has been embodied in a specialized software called Database Management System, DBMS, or in everyday language a database system.

A DBMS is a software program that via other programs handles the administration and all contact within the database. This means that the access to the database has to happen via the DBMS. It is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist safely over long periods of time.

A DBMS is also, what is called a general purpose program, which means that it is not just written for a certain application or database. It could be used to create and administrate several different databases as well. These systems are among the most complex types of software available.

The capabilities that a DBMS provides the user with are:

- *Persistent storage*: A DBMS supports, like a file system, the storage of very large amounts of data. This data exists independently of any process that is using the data. A DBMS, also provides flexibility and finesse that goes far beyond those of a file system. These are, for example, data structures that support efficient access to very large amount of data.
- *Programming interface*: A DBMS allows the user or an application for that matter, to access and modify data, using a powerful query language. It is important to observe that the advantage of a DBMS over a file system is not only the flexibility of being able to manipulate stored data, but doing so in much more complex ways than reading and writing of files.
- *Transaction management*: A DBMS supports simultaneous access by many transactions at once. To avoid some of the undesirable consequences of simultaneous access, the DBMS supports isolation so that it appears that the transactions execute one at a time. The DBMS supports atomicity which means that the requirements that transactions execute are either done completely or not at all. A DBMS also supports durability, the ability to recover from failures or errors.

A key element of all databases created from a certain DBMS is that, they have a predetermined structure which allows users to access all databases in the same way.

To be able to get data from a database and modify it, there is a procedure called manipulation of data. A database language which supplies these functions is called a Data Manipulation Language, DML. Changing the structure in a database is yet another ability that has to exist. A language for being able to do so is called Data Definition Language, DDL.

Both of these languages can be represented as one database language in a DBMS. Other types of database management systems could have one language for definition and one language for manipulation. It is important to observe that DDL and DML are not different examples of database languages, but a description of what functions the language itself contains.

It is not necessary to use a general database management system. In that case developers are forced to specially design their database system which will lead to writing all of the programs and functions required for administrating the database.

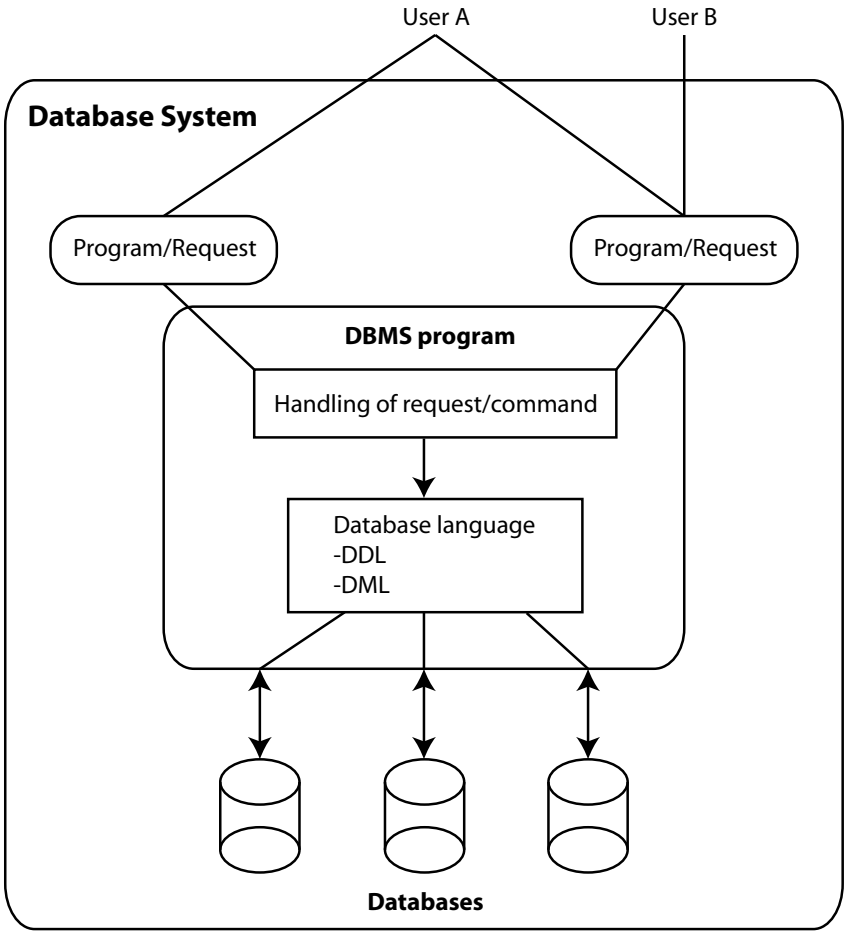

Figure 5.1: The database management system

5.2 The Designing Process

The process of designing a database begins with an analysis of what information the database must hold. This analysis includes also the relationship among the components of that information. The structure of a database is called the database schema, which is specified in one of several languages or notations suitable for the design in question. A database takes its physical existence after the design has been committed into a form called the Database Management System, DBMS.

There are several popular design approaches for a database. Some of these popular approaches are going to be discussed in the following subsections.

5.2.1 Entity-Relationship Data Model

This model, also referred as the E/R model, is the most common model of abstract representation of a database schema. The representation of the structure is performed graphically. The abstract E/R design is converted to a schema in the data specification language of some DBMS. Most commonly this DBMS uses the relational model. There are no DBMS that uses E/R model directly. The reason is that this model does not support an efficient data structure which is motivated in database manner.

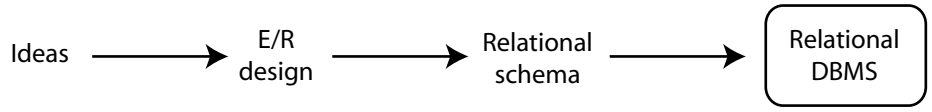

Figure 5.2: The database modelling and implementation process

5.2.2 Relational Data Model

The relational model provides a single way to represent data. This representation of data is done as a two dimensional table called a relation. It is often easier to design databases using the E/R notation at first, and then translating them into relational notation. Relational models have a design theory of their own. This theory is called normalization which is in MMOGs case discussed later on in this chapter. The relational model is very flexible and supports a very high level programming language such as the Structured Query Language, SQL.

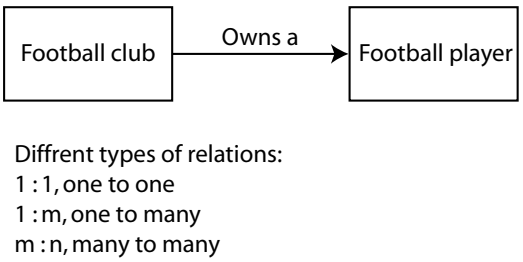

Figure 5.3: A simple approach of some possible relations

5.2.3 Object Oriented Data Model

This approach is extended from the concept of object oriented programming languages such as C++ or Java. In the database world the idea is used for database design and for extending relational DBMSs with new features. Object Definition Language, ODL is a standardized language specifying the structure of databases in object oriented terms. Object oriented database is most suitable for applications that work with more complex data structures such as CAD system drawings.

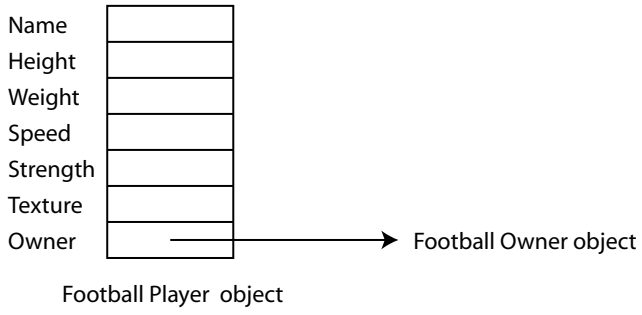

Figure 5.4: An object representing a football player

5.2.4 Object Relational Model

This model is an extension from the relational model to the object oriented model. The reason is to take advantage of the most common object oriented concepts, such as classes, object identities, attributes etc. This model itself is a part of the most recent SQL standard called SQL-99.

5.2.5 Semi Structured Data Model

This model is suitable for integration of databases. In other words it describes the data contained in two or more databases that contain similar data with different schemas. It serves as a document model in notations such as XML as well. The last mentioned is used to share information on the Web and is explained in the following subsection.

The structure of the data representation is very similar to a hierarchical one but with the difference that a parent node in this case could refer to another parent node, see figure 5.5. The hierarchical structure itself is described in a top to bottom approach.

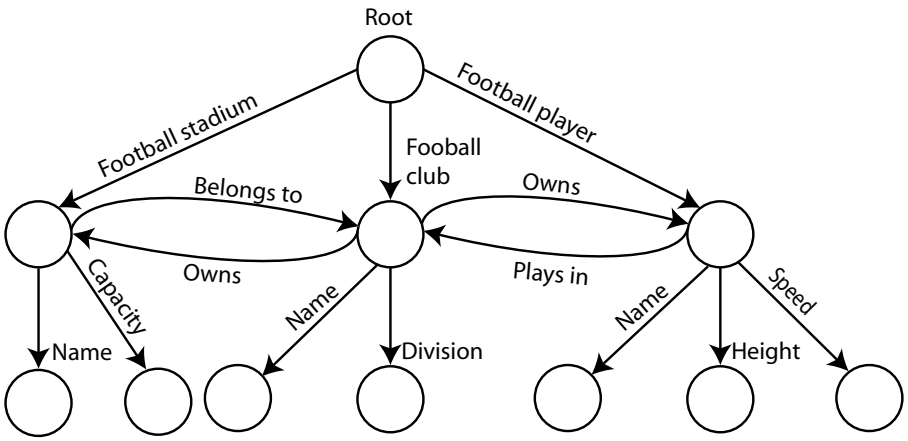

Figure 5.5: Semi Structured Data Model representing a football stadium, a football club and a football player

5.2.6 Extensible Markup Language

Extensible Markup Language, XML is a tag based notation for marking documents. A XML document contains characters and the XML notations are very similar to HTML. However, while an HTMLs tag describes the representation of the information contained in the document for example font style, XML tags describes the meaning of substrings within the document. The XML model in MMOGs case is going to be discussed later on.

5.3 Database Management in Data Driven Systems for MMOGs

A database management system is powerful piece of software which takes care of a heavy workload in industrial applications. There are many MMOGs on the market already, taking advantage of database management systems to provide better service for their customers. The purpose of data driven systems is to separate the source code from the game rules. Yet, this separation could be managed in more appropriate ways.

Most game rules in MMOGs can be summed up in different types of objects, behaviours etc. Following examples are game objects that can be represented by data:

- Weapons
- Spaceships
- Character skills
- Types of terrain
- Planets

Some examples of the IFM's game objects could be following:

- Football player
- Football manager
- Football player skills
- Football stadium
- Football club

Not choosing to have data driven systems, major problems could occur. In the IFM game we have thousands of players. Without data driven systems all players would have to be defined in the source code. This would result in a scenario where recompiling the entire code would have to be done for every little change made.

Lower costs of development, maintaining and extending are other benefits that data driven systems results in. There are different media that can be used for storing the data used for data driven systems. Each type of data source has different characteristics and complexity. Following subsections are descriptions of some of the ways that data for driving the systems of a MMOG can be stored.

5.3.1 Relational Databases

Relational databases can be used as data sources. A Relational Database Management System, RDBMS is a program that is used for creating, updating and administrating a relational database. RDBMS also organises the data and gives users the possibility to find the information again.

The service that the software RDBMS is designed to provide, is implemented via a standardized language, SQL. As software, SQL is the one which is de facto standard in business applications throughout the corporate world. There are many software trademarks providing RDBMS, it is worth mentioning that there are open source options such as MySQL also.

In MMOGs, one might find tables representing a Player, a Character, a Monster and a Bank. A character table could, for example, contain attributes as ClassId, Name, Height, Weight, etc. These attributes are on the other hand not relevant for the games in general, which means that they are different from a game to another.

This example shows SQL code for creating a table to store different types of players as well as inserting data into the created table.

```
CREATE TABLE playertypes
(
  player_id  int           PRIMARY KEY,
  name       varchar(32)   NOT NULL,
  height     int           NOT NULL,
  weight     int           NOT NULL,
  speed      int           NOT NULL,
  strength   int           NOT NULL,
  texture    varchar(32)   NOT NULL,
);

INSERT INTO playertypes VALUES (1, "shortPlayer", 160, 65, 9, 9,
                                "shrtP.bmp");

INSERT INTO playertypes VALUES (2, "tallPlayer", 185, 85, 6, 8,
                                "tallP.bmp");
```

A football player table in the IFM game's database could look like the following one:

Id	Name	Height	Weight	Speed	Strength	Texture
1	shortPlayer	160	65	9	9	shrtP.bmp
2	tallPlayer	185	85	6	8	tallP.bmp

Table 5.1: A football player table

Data in relational databases is organized into units called tables. A table contains columns of various data types and each entry in a table is called a row.

A table will usually have a primary key. This key is a combination of one or more columns that uniquely identifies a single row in the table. This unique index, created as the key of the table, is typically the optional method of accessing a single row from the table. The profit that comes with relational databases is a more flexible model which can be modified in the future.

The IFM game organizes its game data by using a relational database. The software that is used as a standardized language for this purpose is MySQL. The main reason why MySQL is used in the IFM game, is that the software is an open source project. Yet another reason is that relational database systems are suitable for data driven systems and games. They are also more practical when it comes to manipulating data, searching for data, storing data etc.

RDBMS offers robust tools for viewing, manipulating, defining and managing schemas. Administration can however be very complex to install and maintain. The IFM game continuously updates the game database application on fsdata Company's Web hotel. This is done by using fsdata Company's FTP services.

5.3.2 Text Files

Software development is all about choosing the right tools for the job. Using text files as data source solution is certainly the simplest way. Text files are also often referred to as a tab delimited or colon separated files. Text files can at best simulate the concept of a database, although text files are commonly incorrectly referred to as databases.

It is important to not forget that databases are created for handling and storing data, so using a database is considerably easier then for example using text files. The main reason why databases are easier to use is because they take care of the information details for the users.

One could let SQL do the work instead of writing routines to parse text files. Managing large amounts of text files for data can become difficult using text files. As parts of, for example, a MMOG get more complex during the development, it is better to use a database instead of functions to operate on text files. If one chooses to stick to text files despite large amounts of data the solution would be to use a parser, written by the developer, for creating tables.

RDBMS, as already mentioned, is built for eliminating the complexity of information handling. This fact is the main advantage for using a real database model. One could describe a MMOG as a highly dynamic data area, an area where things change a lot. Database performance improvements will therefore likely be significant. Not only is the server going to perform fewer steps, but each of those steps will also be a lot faster to execute.

One should not go around and think that plain text files are useless for using as a data source solution. Text files and text file operations are good candidates for managing data sources for MMOGs. The structure of a text file is arbitrary and not predefined. This forces the programmer to write and handle syntaxes, relationships between rules in a way so that the information in a file could be validated.

Following example shows how a colon separated text file is parsed into a table.

```
# Class: Name: Height: Weight: Speed: Strength: Texture
CShrtP: shortPlayer: 160: 65: 9: 9: shrtP.bmp
CTallP: tallPlayer: 185: 85: 6: 8: tallP.bmp
```

# Class	Name	Height	Weight	Speed	Strength	Texture
CShrtP	shortPlayer	160	65	9	9	shrtP.bmp
CTallP	tallPlayer	185	85	6	8	tallP.bmp

Table 5.2: An alternative way of representing a football player table which is generated by a parsed text file

5.3.3 XML

XML allows developers to easily describe and deliver structured data from any application in a consistent way. It is a little bit hard to understand at first, but XML does not actually do anything. XML was created to structure, store and to send information.

An XML database has several advantages over other type of databases. XML data can be inserted directly into the database. The game data does not need to be manipulated or extracted from a document in order to be stored. When data gets inserted into the database, most parts of an XML document, including white space, are maintained exactly. Queries return XML documents or fragments. This means that the hierarchical structure of XML information is being maintained.

In general, XML databases allow programmers to handle XML data and store it quickly. This also includes performing the last mentioned task with the minimum of programming time required. Eliminating the need for converting XML into other data structures is an advantage.

Looking at the disadvantages of XML, the lack of good tools can be mentioned. However, there are more advantages than disadvantages with XML. Developers are often forced to build custom tools to manipulate the game data anyway.

XML data is a little bit harder for people to read than other data sources. Following examples shows how data for different type of players could be presented by XML.

```
<\<>object name = "shortPlayer">
  <property name = "height"    value = 160>
  <property name = "weight"    value = 65>
  <property name = "speed"     value = 9>
  <property name = "strength"  value = 9>
  <property name = "texture"   value = "shrtP.bmp">
</object>
```

```
<\<>object name = "tallPlayer">
  <property name = "height"    value = 185>
  <property name = "weight"    value = 85>
  <property name = "speed"     value = 6>
  <property name = "strength"  value = 8>
  <property name = "texture"   value = "tallP.bmp">
</object>
```

5.3.4 Scripting Languages

Other types of techniques that are used for data sourcing are scripting languages. Possible languages that are used are Python, TCL, Lua, Java or languages written by the developers.

So, why use a scripting language? Well, using scripting languages simply puts most of the game logic scripted for functionality, rather than coding it as part of the engine. By using these scripting languages there would be no need for developers to write a data parser and the integration with the system will get uncomplicated as well. Thus scripting languages becomes tremendously powerful if the game engine uses it for storing or manipulating data. From a developers point of view the stored data gets easy to access and manipulate by using tools made by the developer.

As an example, think about loading or initializing a level. When the level gets loaded, one would maybe like to cut a scene to play or maybe show some game credits, etc. With a scripting system, one could get individual game entities to do specific tasks. This means that several entity activities could be executed at the same time, in other words in parallel.

Yet, another useful area is using scripting languages for AI. Take for example a Non Player Character, NPC, it needs to know what to do. Coding each NPC in the game engine each at a time, could be a frightening task in a matter of time the task would need to be taken care of. If one would change a behavior of a certain NPC, one would have to recompile the whole system. Using a scripting system on the other hand, one could interactively change the behavior and save the time required for recompiling.

Following example shows how Python code could be used for defining different types of players at the IFM game.

```
InitialPlayerTypes = [
       ["shortPlayer", 160, 65, 9, 9, "shrtP.bmp"],
       ["tallPlayer", 185, 85, 6, 8, "tallP.bmp"]
]
```

5.3.5 Database Services Offered by the fsdata Company

The IFM game organizes its game data by using a relational database. The work that involves continuous updating of the applications does get a lot easier by using the services that the company fsdata Company offers. There is no point reeling off all services that fsdata Company has to offer, only those that are relevant for the IFM game are going to be discussed here.

The Web based administration at fsdata Company contains several functions, which one easily can control via a Web based interface. At an allotted Web page one can adjust how the system should work and correspond to commands. In that way one can take advantage of its finesse without completely understanding how a Web server works in detail. The understanding of how a server works, users of an fsdata Company's Web hotel avoid functions that might take programmers days or weeks to develop. Using fsdata Company's Web administration, it becomes very easy:

- to change passwords
- to protect Web pages by passwords
- to create Open DataBase Connectivity, ODBC connections
- to control the usage of disc space, etc.

ODBC is an Application Programming Interface, API which has a series of functions that other programs can use to make the operating system do the so called hard work. ODBC is actually a DBMS program, which has been discussed earlier, that fsdata Company has developed. Using Windows APIs for example, a program can open windows, files, and message boxes as well as perform more complicated tasks by passing a single instruction.

If one would like a program to talk to for example three different databases using three different scripting languages one would need to code a program with three different database scripting languages. This can be a quite frightening task. When programming to interact with ODBC one only needs to talk the ODBC language and the ODBC Manager will figure out how to contend with the type of database that is targeted. The only thing that one needs to do is to install an ODBC driver that is specific to the type of database that will be used.

Another service that fsdata Company has to offer is for updating a database, this could be done by sending the updated files to one's allotted domain along with the usage of FTP software.

The server supports a large number of scripting languages that are embedded in HTML. Scripting languages as PHP 4, SSI, Embperl and ePerl are available for usage. Possibility to access powerful software as MySQL is another advantage. One is able to create over 60 databases and there is also a Web based administrator for using MySQL.

The company is able to offer very detailed statistics. The generated statistics can, among other things, give answers, to questions such as:

- what Web page has most visitors
- where the Web page visitors comes from
- at what time people do visit one's Web pages
- what operative system and Web browser the users use, etc.

All this information can be used to improve a certain Web page or in IFM's case, improving its game in different ways.

5.4 Types of Data Driven Game Architectures

There are many types of game architectures. In the following subsections, three possible game architectures that can be driven by data are going to be discussed.

5.4.1 Generated Classes

Using data for generating source code in this architecture, results in a programming language class for each type of game objects that the game contains. Using a module as a back bone of this system one can generate class definitions from a set of data for language that is used in the game.

Generating these class files from data and compiling the class files beside the source code is a two step process. It is important to observe that the generating class files step must be performed before the compilation of the class files. This must be done in mentioned order due to header files that must be available when the non generated code is compiled.

Following are some of the advantages of using the above mentioned procedure:

- It is more efficient because all data is at the class level, not at the instance level.
- The code is consistent, as it is generated by the same source.
- It is also more efficient because there is no loading time, all data is presented as class attributes.

Following are some of the disadvantages of using the above mentioned procedure:

- The fact that compiling is a two stage process.
- The fact that code must be recompiled for even the smallest change.
- It might be a bit hard for people to read the generated code.

Following example shows how a generated C++ header files code for different types of players might look like in the IFM game's case.

```
class CShrtP : public CPlayer
{
static const string      mLabel = "shortPlayer";
static const int         mHeight = 160;
static const int         mWeight = 65;
static const int         mSpeed = 9;
static const int         mStrength = 9;
static const string      mTexture = "shrtP.bmp";

CShrtP(void);
virtual ~CShrtP();
};

class CTallP : public CPlayer
{
static const string      mLabel = "tallPlayer";
static const int         mHeight = 185;
static const int         mWeight = 85;
static const int         mSpeed = 6;
static const int         mStrength = 8;
static const string      mTexture = "tallP.bmp";

CTallP(void);
virtual ~CTallP();
};
```

5.4.2 Dynamic Properties

In this architecture different type of game objects are completely separated from the programming language model, where these game objects have a dynamic set of priorities that are entirely driven by data.

Even this system is implemented as a module where game objects, as the architectures name implies, have a dynamic set of attributes where each of these has a value. At the programming language level, only one class can be found. So, comparing different types of game objects we come to a conclusion that these are all instances of this single class but with different attributes.

Below are some of the advantages, of using the above mentioned procedure, presented:

- There are certainly a small number of classes, as there is only one.
- On the other hand this architecture provides high flexibility as every single instance of data can be different.
- It is also allowed to use dynamic run time modification of object attributes.

Below are some of the disadvantages, of using the above mentioned procedure, presented:

- There is only instance data and no class data.
- This type of architecture requires loading time creating all objects.
- It can also be difficult to determine the type of game objects as it is common to have instances of the same type that in turn also have the very same attributes.

Following examples shows how code for constructing player prototype objects with properties that have different values might look like in the IFM game's case. These prototype objects are then used as bases to copy from when creating actual game instances of these types.

```
GameObject shortplayerTemplate = new GameObject();
shortplayerTemplate.addProperty("height", 160);
shortplayerTemplate.addProperty("weight", 65);
shortplayerTemplate.addProperty("speed", 9);
shortplayerTemplate.addProperty("strength", 9);
shortplayerTemplate.addProperty("texture", "shrtP.bmp")

GameObject tallplayerTemplate = new GameObject();
tallplayerTemplate.addProperty("height", 185);
tallplayerTemplate.addProperty("weight", 85);
tallplayerTemplate.addProperty("speed", 6);
tallplayerTemplate.addProperty("strength", 8);
tallplayerTemplate.addProperty("texture", "tallP.bmp")
```

5.4.3 Category Objects

In this case an instance of a category object for each type of game object can be found. There is also a set of category classes for the categories of types of game objects. This type of architecture can be seen as a compromise solution between the two previously mentioned architecture solutions.

This system is also implemented as a module but here a hierarchy of category classes can be found instead. These classes represent a wide area consisting of categories, which are different types of game objects. These types of game objects could have differing behaviour or data requirements. A big difference in architecture design in relation to previously mentioned dynamic properties system is that each category class here, is a corresponding category instance class. Every category instance object has in turn a category class attribute. This attribute holds the data that is specific to its category.

The advantages of this system include:

- There are a small number of classes. Only one category of game objects exists.
- Category classes can be created at run time.
- By changing the category object instance, all instances of a category can be changed.

The disadvantages of this system include:

- This type of architecture requires loading time for creating all objects.
- This type of architecture requires a parallel class hierarchy for category types and instance types.
- Some data must be accessed from the category instance rather than the object instance.

In this chapter different types of architectures that developers have available have been discussed. By summing all these up, a wide variety of choices when implementing game systems for MMOGs is given.

Following example shows how a category class for types of players and the corresponding game object class for player instances in C++ might look like in the IFM game's case.

```
class CPlayerType
{
  CPlayerType(string label, int height, int weight,
              int speed, int strength, string texture) :
              mLabel(label), mHeight(height), mWeight(weight),
              mSpeed(speed), mStrength(strength),
              mTexture(texture) {}

  virtual CPlayerType();
```

```
  const string   mLabel;
  const int      mHeight;
  const int      mWeight;
  const int      mSpeed;
  const int      mStrength;
  const string   mTexture;
};

class CPlayer
{
  CPlayer(type) : mType(type) {};
  Virtual ~CPlayer();

  const CPlayerType &mType;
};

//Creating Type instance objects
CPlayerType shrtp = CPlayerType("shortPlayer", 160, 65,
                                9, 9, "shrtP.bmp");

CPlayerType tallp = CPlayerType("tallPlayer", 185, 85,
                                6, 8, "tallP.bmp");
```

5.5 Managing Game State Data Using a Database

With MMOGs tending to massive number of players and with worlds growing larger and larger, it is very important to prepare for the enormous amount of data a game will generate. A MMOG could easily have thousands of players and a millions of game objects with these. Several questions such as "Which of these objects should be stored?", "How will the objects live on?" need to be answered. To properly manage these large volumes of data that a MMOG will generate, continuous planning is going to be required.

Substantial growth of the MMOGs size and complexity is a matter of time. During a phase when a database is reaching a state where it is considered as large, attention to their design and implementation is required.

As a developer using powerful databases, one is constantly tempted to use database features in wrong ways. Below are the most common pitfalls, everyone under their own subsection.

5.5.1 Indexing

Why do databases use indexes after all? Well, indexing is simply speeding up the access to the records that are stored in a table. Record is another term for a row in a table. One could request for example a database to retrieve all the players that have scored more than 20 goals during the last football season.

The information could later be used for raising the transfer sum for the player in question. This will in turn benefit his team and further the team's manager which in the IFM's case is a player of the IFM game.

But one of the greatest benefits with indexing in databases is its capacity of retrieving a record table based on multiple conditions. Using the IFM game as a case study, a scenario might occur where the database is to retrieve all the players that have a yellow card assigned in each of the two previously played matches. This information for example, could later on be used for suspending the player/players from the next coming match. The whole idea with indexes is to provide a way for the database system to speed up data retrieval.

It is important to observe that indexing does not only have a positive impact. Indexing will slow down the updates to the tables in question. This will occur because the database system must not only update the data in question when it is being modified, but also all the corresponding indexes as well.

A possible scenario concerning this situation would be having a tournament where a couple of players have received one yellow card each, in a group play. If the team they are playing for should qualify for the play offs, their yellow card status would need to be restored to zero yellow cards. In database terms this means that the database must first off all modify the card status for the players in question that are going to join the play offs and thereafter modify the database. The reason why the last mentioned step must be executed is because players in question must be assigned new indexes as they no longer belong in "the players with one yellow card" segment. This is what is meant by claiming that the database must modify all the corresponding indexes as well.

5.5.2 Normalization and Referential Constraints

Normalization and referential constrains will be under the very same topic because these two concepts are closely connected with one another. Normalization means that a table should only contain such data that is relevant to the key of the table. Thus, it is the practice of reducing the redundancy of data within a database. Referential constrains on the other hand are used when a player table itself must contain a "team_id" that in turn exists in the team table.

So, what could happen if the database would not be well normalized? Assume that all the information is put in one table. Then, the table could grow to be very large. Splitting up this large table into smaller ones, not using the normalization guideline will result in needing to update all tables also when only one table changes so as to keep the data synchronized. So, splitting up a large table in smaller ones and not using the normalization guideline is not recommended from efficiency point of view, as more work is needed maintaining the databases.

Too much normalization is not ideal either. The retrieval of data itself if it requires data from many tables, multiple table joins, could result in slower retrieval since data is not located in one place. The retrieval could be so slow that retrieving data from a single large table would take less time.

In this case certain parts should be carefully denormalized for obtaining high performance for that certain retrieval.

5.6 Misused Techniques

Topics that are going to be discussed in the following subsections are some of the techniques frequently misused when working with MMOGs.

5.6.1 Database in a database

The concept of this technique is choosing a string for representing data. Database in a database technique is a very tempting one to use in a MMOGs database, which is to represent more than one piece of information in a single string column.

Suppose an eight byte string is used for information transferring. The structure of this string column could be written so that the first byte is the colour of the player's hair, the second byte could represent the players power ability to kick a football, etc. By choosing this technique the information will be packed into a smaller space. Thereafter there would be no need for additional columns to the table.

As with everything else in this world, there is a positive and a negative aspect of using this technique. It is almost impossible to index such a column which leads us to the fact that referential constraints cannot be applied. As already mentioned this technique may be very tempting to use, but the data integrity is placed in the hands of the application itself and not in the database, where it also should be.

5.6.2 Multipurpose Columns

This is a technique where a basic column is made for storing one type of data for one record and storing another type of data for another record, at the same time.

An example is having a column named "slot1_data". In this case the goals scored could be stored in one record and the location of the player object at another. This technique reduces the number of columns. If one would like to add new records for a certain object, this could be done easily, since these records are reusing the existing table structure.

Multipurpose columns is on the other hand a technique where indexing of the columns is difficult to manage. It is possible, but the performance of such indexing is unpredictable, as data on these columns may be unrelated.

It is impossible for the database to know what data "slot1_data" column contains as it is different from record to a record. The application itself is in control of that, instead of the database. Integrity is in this case also being placed in the hands of the application itself and not in the database, where it should be.

It is almost impossible for the database systems to apply referential constraints, as several records could use the same column for different sort of things.

5.7 Considerations Concerning Data System

Concerning data system as a whole there are both internal and external performance influences. Few of those are presented in following subsections.

5.7.1 Performance

It has already been discussed what performance loss that huge tables, extensive joins and indexing could cause. The performance consequences of the overall database system have however not been discussed.

The maintenance of a massively huge table is catastrophically time consuming. Should a database containing such a table, crash or be affected by a data virus, it would be extremely difficult to recover the lost data.

Heavy performance losses can also occur if table joins are used, where for example, a 10 table join is required to retrieve the information desired. When it comes to MMOGs a join consisting of maximum two to three tables should only be allowed. Even table joins of this size could generate high performance losses if they are used with high frequency.

MMOGs are most often using queries and updates as database utilities. In addition, at some point of time, deletes are going to be required, hence they are very expensive to perform. These deletes should not be executed during high activity but during maintenance of the database.

5.7.2 Networks

The fact that database systems allows an application to send or retrieve huge amounts of data are just some of the advantages showing the great flexibility of a database system. Unfortunately, these benefits of database systems are not always used in a proper way which can be rather problematic.

No matter what object is studied, it is simply, a waste of bandwidth sending all attributes of that certain object, if only one of the objects attribute is being requested to be retrieved.

Using the IFM game as a case study, assume that a player object has an attribute representing its position on the football field and an attribute for how many goals he has scored in a current tournament, among others. Consider the feature at the IFM game for showing highlights from a chosen match already played. However, it would be completely meaningless to send information about a player scoring goal including his entire tournament attends that he has made since he started playing football.

So, instead of very large requests it would be much better to break it up into smaller ones.

Using this technique, only the data that is absolutely required are sent. The last mentioned lowers the bandwidth and improves database system performance in general.

5.7.3 Transaction Loads

A typical MMOG game contains millions of records of valuable information. Imagine how hard a database system would need to work if it did not use indexing. The database system would literally need to scan every record that exists in the table.

A shortcut to a solution to this problem is to take a closer look at what conditions the request itself is imposing. For example, in IFM's case, assume that the information presenting all the players who scored in the first ten minutes of a game that also has a yellow card assigned is wanted.

There should be indexes on goals scored for a certain player, yellow card status for a certain player, red card status for a certain player, goals scored at certain time for a certain player etc.

Also, there should be indexes presenting all these four conditions together. In this way it is possible to request the database system on information fitting the player who both has scored in first ten minutes and who has a yellow card assigned. If these conditions were indexed, this will facilitate the query and the answer to the request will be given faster. Reducing the time for each request will reduce the overall transaction load of the database system.

5.8 Alternative Ways of Managing Data

Even if a developer chooses to follow all the suggestions given, the database system in question can still be overloaded and slowed down. In the following subsections some additional topics discussing things that consider MMOG database systems will be discussed.

5.8.1 Data caching

Using data cashing technique will gain database system performance. The big difference between the usual data caching inside a database system and the one that is going to be described here is that this one is external. It is an additional process placed between the application and the database system itself, as the figure 5.6 shows.

In this model all requests made from the database in question will come through the data cache. After arriving at the data cache the request will be passed on to the database.

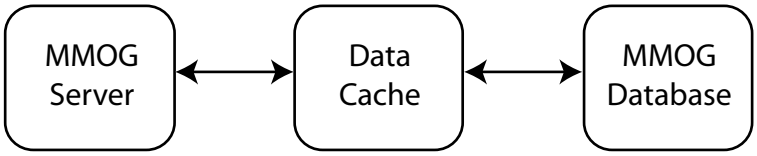

Figure 5.6: A data cashing process

If the request in the future would be for the same information this would be returned from the data cache memory instead of the database system. Using this technique could significantly decrease the load on the database system. Concerning update requests, the process is done in reverse. If the data already exists in the data cache it is going to be modified and sent on to the database later.

According to what has been said here about the data caching technique it has its own so called challenges. Dramatic reduction of load on the database and better performance to it makes the additional work to set this process up worthwhile.

5.8.2 Data Caching in Practice

The game design architecture of a MMOG must be such that no direct line between the user and the database should be available. Assume just for the moment, that there is a direct line between the user and the database. Then, the users would have a direct performance impact on the database. If a user would click at a button that generates a request to the database several times, it would cause some serious problems.

A typical MMOG game might have thousands of users. If every user would press a button that generates a request to the database, that database would obviously be completely occupied. A solution to this problem would be to filter the requests which would then be executed only after certain time delay.

The previously mentioned data caching system could also help to ease the total workload of the database system.

5.8.3 Maintenance and Connectivity

After releasing a MMOG, the number of users will continuously grow with time. As time goes by, the maintenance on regular basis of the MMOGs database gets more and more important. Being able to get the most out of the MMOGs database, to improve its performance, the database should be periodically brought down for maintenance.

Regarding connectivity when it comes to database systems, they are usually able of handling several connections simultaneously. Instead of having only one connection handling and taking care of all the requirements coming in a database system should have a pool of connections to the database.

Having a pool of connections where every one of them could be managed as a separate thread with its own database connection creates an advantage. This advantage signifies the fact that as requests come in, they can be handed off to an available thread and handled immediately. By finding a balance of connections and threads, no request should need to wait to be able to communicate with the database.

Dynamic connections, the ones that are shut down after completing of the request should be avoided as they are time consuming operations. Hence, these need to be avoided as the database system has a direct impact on the performance of a MMOG.

Taking a closer look on what have been discussed in the last subsection a conclusion of managing game state data in a database really comes down to understanding the following:

- What the requirements are
- What must be stored
- How much will be stored
- Choosing how to store it
- Applying the strategies and techniques for the each phase

Chapter 6

Industry Analysis and Market Research of MMOGs

6.1 The Economic Aspects of MMOGs

The MMOGs have been around for a couple of years. The basic idea with this type of game is gathering thousands of players in one place, making them interact with or against each other. At the same time as the interaction between the players is taking place it will also result in simultaneous game play.

The question is, how big is the MMOG industry going to be? According to a detailed industry analysis "State of MMOG 2002: A New World in Electronic Games", written by Zona, Inc. and Executive Summary Consulting, Inc., the estimated number of players in the year of 2002 was approximated to six million players, playing the MMOGs worldwide. The very same amount of players would result in a subscription revenues of USD$494 million, in the year of 2002 alone. By the year of 2006, the very same industry analysis is estimating that MMOG subscription revenues will generate USD$2.7 billion.

One could, of course, as sceptics who do not predict any bright future for the MMOGs, view these future estimated results as an extreme exaggeration. Being able to answer to this scepticism a closer look will be taken at the game software company NCSoft in South Korea. Referring to the industry analysis, the software company NCSoft made about USD$100 million in the year of 2001. This profit was mainly made off one of the NCSoft products which shortly was described as a massive multiplayer fantasy game. This MMOG had an estimated USD$1.2 million of paid monthly subscribers in the year of 2002.

The final conclusion is that if one company alone in a small country can make about USD$100 million on one game, which could easily be described as a fast growing market worldwide, one can widely claim that there is a multibillion industry out there. As it is mentioned in the research itself, the remaining question is how much time it will take before this market will start to get fully used, and for whom.

It is important to notice though, according to the industry analysis, that MMOGs were already tremendously popular in South Korea when the game software company NCSoft broke through with their massive multiplayer fantasy game. This is only five years since the genre of MMOGs first appeared on the market.

On the U.S. market there are more than 100.000 monthly MMOG subscribers, paying USD$10-13 per month. This large number of subscriber payments represents the top ten games which have more than 100.000 monthly subscribers each. In Asia on the other hand there are subscribers paying up to USD$20 a month for playing a MMOG. At the moment, there are more than 50 MMOG titles active around the world. More than 40 MMOG titles are known to be under development and the tension is rising constantly, with people waiting to see which of the MMOGs that will gain more than one million subscribers worldwide.

Even the MMOG sceptics can not disregard the fact that the MMOG phenomenon as such is spreading fast around the world. MMOGs are gaining subscribers all across Europe, in the U.S., Canada, Japan, Taiwan, Singapore, Australia and even China.

6.2 Different Type of Gamers Worldwide

There are different types of people playing games, called gamers, worldwide. In this section several types are going to be discussed and they are:

- PC Gamers
- Arcade Gamers
- Mobile Phone Gamers
- Console Gamers
- Handheld Gamers
- Casual Online Gamers
- Massive Multiplayer Online Gamers

It was back in the year of 1964 that the game software company Sega introduced the game named Periscope. This was the first of its kind, an arcade game. In the year of 2001, almost 30 years after, the electronic game industry generates more than USD$30 billion in consumer spending worldwide per year. This is nearly double what people spend on movie tickets worldwide per year.

6.2.1 PC Gamers

Having an amount of 500 million people owning PCs in total worldwide it is certainly a platform that has more people playing electronic games than any other game platform. Independent of game kind, the PC owning people playing games will constitute to 40% of the total amount of 500 million people owning a PC.

6.2.2 Arcade Gamers

These types of games are not as popular as they were at their peak in the late 1980s. Because they are still distributed in the arcades worldwide, the arcade game playing is not yet to be dismissed.

6.2.3 Mobile Phone Gamers

Concerning the European and the Asian market, mobile game play is extremely popular and the amount of those playing mobile phone games continues to grow. There has not been any real indication of MMOGs for mobile phones to date yet. The market of North America is on the other hand not as developed as the European and Asian ones concerning mobile phone games. It's depending on the fact that the mobile phone gaming is fast growing but still relatively small worldwide.

6.2.4 Console Gamers

It is estimated that 40% of all U.S. households have game consoles. One should notice that the number of gamers using their home consoles continues to grow and that the amount of these kind of players, are usually growing in proportion to release cycles of new platform upgrades. Three of the biggest game console companies, Sony's PlayStation2, Nintendo's GameCube and Microsoft's Xbox sold more than 40 million of their systems in the year of 2001 alone. As these three previous companies continue to sell their game consoles, Sega as well as Sony and Nintendo continue to service millions of users with new game titles, who use their older console systems.

6.2.5 Handheld Gamers

Nintendo's game device Game Boy holds a significant market share of the total handheld game device market. Personal Data Assistants, PDAs such as Sony's Clie, Palm and Handspring are growing in popularity as many of these could also be connected online. This possibility makes it possible to play online games. Due to the fact that the Game Boy device itself does not connect wirelessly online it does not look like there will be a large number of players soon anyway.

6.2.6 Casual Online Gamers

It has been estimated that in the year of the 2002, people playing games on the Internet would include as many as 85 million people worldwide. One should notice though, that the majority of online game play is free to users.

6.2.7 Massive Multiplayer Online Gamers

The amount of players who have played MMOGs have been estimated to 6 million people worldwide in the year of 2002. MMOGs have been around since the beginning of the 1990's. According to the industry analysis global subscriber figures will rise to 19 million by 2006.

6.3 High Profit Economics of MMOG

The main reason for the enormous popularity of the MMOGs among publishers is their huge potential for profit. The profitability of the MMOGs depends on the number of monthly subscribers.

As it has been mentioned earlier, the monthly subscription fees lie in the range of USD$10-13 in the U.S. and in the range of USD$20 in Asia. In some popular game cafes in Asia one hour of playing the MMOGs could lie in the range of USD$1-2 which suddenly makes the price of an average monthly subscription fee in Asia of USD$20 a reasonable price to pay for playing a MMOG.

These subscription fees are often associated with newly arrived hot titles on the market that will potentially be able to have up to 200.000 subscribers per month. For smaller independent games the monthly subscription fees usually lie in the range of USD$5. In the IFM's case the subscription fee for one season is USD$5.

So, what about the costs of developing a MMOG, launching it on the market and the time needed before making it profitable? Well, without getting too involved in details, the development itself of a MMOG game costs about USD$4 million. The marketing of the MMOG will cost about USD$3 million and another USD$2 million will be needed for the capital investment and integration of the server network. It will in other words cost about USD$9 million in total before the game is even turned on.

It is extremely important to notice that these figures concern MMOGs which are assumed to have a base of monthly subscribers that will eventually reach 200.000, two years after being launched on the market. It is also assumed that the monthly subscribers fee will be USD$12 per month. The total cost of USD$9 million includes the MMOG in question having 75.000 monthly subscribers in the first quarter which is a realistic figure only for a really huge MMOGs.

First of all, there are some huge differences between diverse kinds of MMOGs. There are MMOGs on the market, like the one mentioned earlier, where millions of dollars have been invested. The equation is simple, if one chooses to invest a large amount of money in a MMOG, the chance of making a great profit will certainly increase. But investing large amounts of money will lead to a greater risk as well.

Then there are MMOGs where the publishers have not been involved in the marketing of them and where the total investment is considerably less than in the example mentioned above. Obviously, every investors dream would be investing as little as possible and still be able to have a monthly subscriber amount of one million people. Smaller companies developing MMOGs which are not able of investing large amount of money would therefore certainly satisfy themselves with a considerably lesser amount of profit, rather than those which have invested several million of dollars.

The IFM game belongs to the genre of smaller independent game companies which do not have a publisher marketing their game. Instead, the marketing of the game is managed by registration of the game in several search engines such as Google, Yahoo, etc. The game development team consists of three software engineers and a 3D artist.

Assume a MMOG which has one million subscribers per month where the subscription lies in the range of USD$10-13 per month. The figure 6.1 represents the accumulative monthly subscriber's fee over one year period having a subscription fee equal to USD$13. The MMOG in question would then achieve USD$156 per year and player which can be seen in figure 6.1, as well.

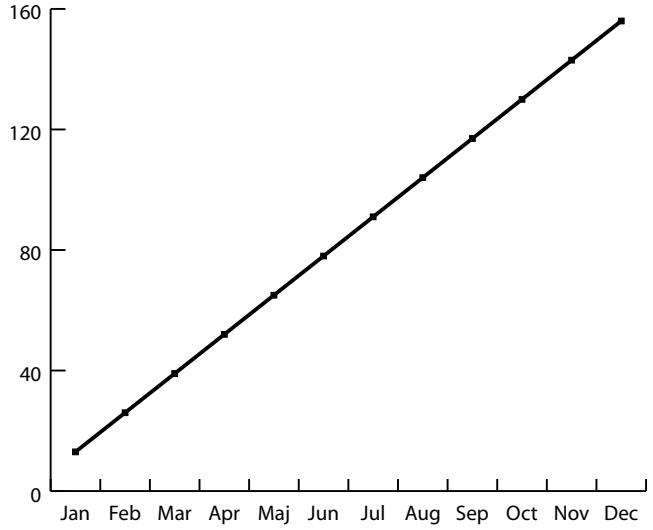

Figure 6.1: Accumulative monthly subscribers fee over one year period

This means that a MMOG will be able to achieve more than USD$150 million in gross profit in a year. In three years, the gross profit will be USD$450 million. Selling an ordinary PC game that costs USD$50 to one million buyers through the retail channel, the achieved sum will be equal to USD$50 million in gross profit in a year.

Some MMOGs on the market are played by installing the software on the users PC. This software can either be bought or downloaded for free. In case of buying the software, the price lays in the range USD$15-50. In the IFM's case there is no need for any software installation at all. This is of great advantage, because one does not need to pay for any initial purchase as well as avoiding software installation. When playing the IFM game, one could still be able to join the game from different PCs, while not needing any initial software on these at all.

Considering that there is a huge market out there to conquer as well as no standardized way of receiving payment, there are a few companies envisioning and at the same time pursuing alternative revenue strategies. Some of these are hourly pricing, prepaid coupons, and payment via mobile phones.

According to the industry analysis, selling the initial purchase in stores reduces the barrier of gamers entering the game. This concerns only the MMOGs where a player must make an initial purchase to be able to play a game. There is therefore no question of replacing a retail channel but to add another channel online.

There are expenses on the operating side which have to be included as server costs, that are estimated to be USD$1.25, as well as bandwidth expenses which are estimated to be USD$1.50 per user and month. The costs on the server side include warranty costs, network operation, firewall maintenance, data backup, database administration, etc. In the IFM's case all of the previous mentioned services are handled by the fsdata Company which is specialized in these kinds of services.

According to the industry analysis model, another USD$1.4 million are invested into ongoing development expenses as there is a customer related technical support cost estimated at USD$1 per subscriber per month. By the eighth quarter the model is estimated to have 250.000 monthly subscribers. After a two year period, it is assumed in this model that no increase in subscription fees will occur as there are no revenues from sales of initial game setup software. In the figure 6.2 it is shown that the model breaks even by the sixth quarter and returnes a cumulative profit of USD$15.9 million in two and a half years.

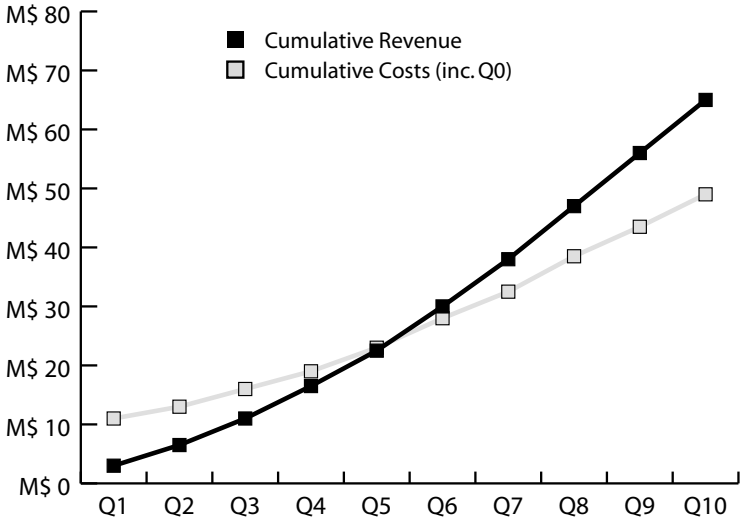

Figure 6.2: Two year revenue versus expense growth

A MMOG that achieves 100.000 users within two years, according to this model, will turn a profit in the eighth quarter. For a MMOG that reaches one million monthly subscribers within 18 months, this model generates a cumulative profit of USD$100 million after two and a half years.

For a MMOG to be able to achieve one million monthly subscribers, its graphical and audio richness, speed of action and technical sophistication has to be better in comparison with the standalone games.

MMOGs have their downsides as well. A PC or console game takes less than two years to develop in comparison with four years or more for a MMOG. Despite four years or more of development, up to USD$10 million in development expenses, the MMOG as such will return a tremendous profit if it breaks through on the market as a title hit.

6.4 MMOGs versus Hollywood

As the movie industry makes a great profit on their movies, the MMOG industry sees Hollywood as an unexplored market with great potential profitability. In order to generate a great profit return in the movie industry, the audience dose not need to be as large as in the MMOG industry. A costumer usually pays USD$7 in average for seeing a movie while a MMOG player pays USD$13 for 12 consecutive months.

According to an industry analysis "The Challenges of MMOG Development: Learn more about the issues that MMOG developers face with their projects" written by Zona, Inc. and Executive Summary Consulting, Inc., the storylines of the games must be rich and compelling enough to keep gamers engaged for hundreds of hours of play over several months. Many games now feature regular updates of content, almost more like TV programs. For standalone games, developers need to program the game only once. MMOGs must be rich enough in content to sustain gamers for hundreds of hours of players. Players will not settle for less than what they can get from standalone games in terms of story complexity, server stability and multiplayer interactions.

According to the Internet Movie Database, there have been about 30 movie successes that have grossed USD$450 million in history, over their life time. However, an average movie costs about USD$50 million to make in comparison with the development of a MMOG which costs about USD$10 million. This fact makes the movie industry in Hollywood to invest in MMOGs. Some famous Hollywood movies that have also become MMOGs are:

- Star Wars
- The Matrix
- Crouching Tiger Hidden Dragon
- Minority Report
- Lord of the Rings
- Harry Potter

6.5 MMOGs Around the World

MMOGs have in the past years grown into a million player industry worldwide in Asia, Europe and North America. An interesting case worth mentioning is South Korea that has more than four million players which corresponds to nearly 10% of the population. It is not a coincidence that MMOGs are a huge industry in South Korea. The country has broadband connections in more than 50% of all households making it one of the leading countries in broadband field. According to Business Week magazine, South Korea has over 20.000 gaming cafes which had an estimated cost of USD$862 million in year 2000 alone.

According to the article "MMOG - Game On!" written by Michael Singer, there are 25% of Internet households in the U.S. with broadband connections. In Europe the MMOGs are a blooming market as the number of developers and gamers is increasing across the continent, to be precise in France, Germany, Italy, Russia and Scandinavia. In Germany, Japan and Netherlands, more than 40% of Internet homes have broadband. The Internet connection status in these countries increases every day and today it is more than enough to reach the expected amount of subscribers in the future.

There are significant differences in the European market according to "State of MMOG 2002: A New World in Electronic Games", written by Zona, Inc. and Executive Summary Consulting, Inc. Martin Rogard, an employee in a French game development firm which produces strategy based MMOG with 150.000 subscribers worldwide, claims that: "Real-time strategy games are more popular for the European market than the role-playing types of games that dominate the American market," he says. "Lots of European players do not want to be a character in the game, they prefer strategy. This is very important in Germany, for example."

In the current state of the world, the U.S. market with its 1.2 million active MMOG players is the most significant market for the MMOG sector. From the developers point of view the MMOG market has a bright future. Several MMOG projects are to be implemented around the world such as the U.S., South Korea, Japan, France, Norway and Iceland.

6.6 Pirating MMOGs

Pirating MMOGs is an impossible task to manage because of one simple reason. Even if an initial purchase of a MMOG is cracked down and copied, the user cannot play the MMOG without paying the monthly subscription fee.

This is on the other hand a huge problem concerning single games. The game pirating business in Asia is a huge industry which publishers around the world have condemned. According to the Interactive Digital Software Association, IDSA, the publishers lost USD$3 billion on these kinds of game revenues in year 2001 worldwide. But despite the aforementioned attempt to pirate the initial game setup CDs of MMOGs, publishers do not lose nearly as much as when the single games are pirated.

6.6.1 MMOG Payment Systems

A vulnerability concerning MMOGs is that the players most often pay via credit card or bank transfer across the Europe and U.S. This indirect vulnerability of MMOGs is not a concern of the publishers. There are no standardized ways of paying monthly subscription fees as of yet. The Asian market has alternative ways of paying the monthly subscription fees such as cash by the hour at game cafes, coupons and mobile phone payments.

PayPal is an account-based system that lets anyone with an email address securely send and receive online payments using their credit card or bank account. Along with the service that PayPal provides, the IFM game is capable of accepting online credit card payments.

It does not cost the IFM game to sign up for PayPal. The company charges modest fees for its members to receive payments. Players playing the IFM game are not charged any extra when paying their subscription fees.

To manage the payment without involving other parts such as PayPal in the process, in order achieve as high profitability as possible, the developers of the IFM game plan to come up with optional ways for players paying their subscription fees. One solution might be managing the payment process within the game itself. Other solutions could be paying via mobile phones.

6.7 Global MMOG Subscription Revenue

The MMOGs and similar online games may conquer a quarter of the electronic game market by the end of this decade. Referring to industry analysis 60% of all Americans, i.e. nearly 170 million people are playing electronic games on a regular basis.

	2001	2002	2003	2004	2005	2006
Europe	$29	$46	$74	$118	$189	$302
North America	$93	$156	$267	$414	$700	$1.085
Asia	$200	$290	$421	$610	$884	$1.282
Elsewhere	$1	$2	$4	$8	$16	$32
Global	$323	$494	$765	$1.149	$1.789	$2.701

Table 6.1: Estimated Consumer Spending on MMOG Subscription Worldwide in Millions of Dollars

As can be seen in table 6.1, it is estimated that global MMOG subscription revenue will hit USD$2.7 billion by 2006. In order to achieve USD$1 billion in subscription revenue in the U.S. market, only 4.4% of all existing gamers, i.e. 7.5 million people, need to play MMOGs.

The Themis Group, an international provider of marketing and management services for developers and publishers of online PC and console games, estimates that online gaming revenues for year 2003 are going to be USD$635 million in the U.S. alone.

DFC Intelligence, a strategic market research and consulting firm which focuses on interactive entertainment, forecasts that 114 million people worldwide are expected to be playing online games by the year 2006.

There is as of yet no standardized way of distributing MMOG initial game setup software. In Europe and Asia the free model which means that the MMOG initial game setup software is given away for free online is likely to be more prominent.

In the U.S. on the other hand it is still to be decided if the free model as the one applied in Europe and Asia is to be used. First of all, this is mainly because there is a possibility for some so called hybrid approaches. These hybrid approaches include, for example, charging for the CD in stores while giving the software away online or requiring the setup software to be purchased retail. Secondly, there are huge numbers of offline channels which in the end will result in preserving the physical sale distribution on the U.S. market.

Independent of what model that will be applied in the different MMOG markets worldwide the retail revenues will still be small in comparison to monthly subscription revenues. By assuming 50% of the 19 million MMOG subscribers worldwide by the year 2006 in assumption that will be privileged of not paying for any initial game setup software, only less than 10% of the total profit will be lost. This estimation assumes an average price of initial game setup software to be USD$25 which will lead to USD$237.5 million in total.

According to the article: "Trends in MMOG development" written by Mirjam Eladhari researcher at the Zero-Game Studio Interactive Institute in Sweden, there were 51 MMOGs available and about 120 MMOGs in development by April 2003.

6.8 MMOG Genre Trends

Considering MMOGs one could think that game types in the future will be a combination of different game genres, such as real time strategy or first person shooter combined with role playing. One could think of larger number of game worlds having unique themes as well.

Looking at the figure 6.3 one can see that world theme genres for games in development is about the same as for existing ones. An observable change that is not clear from what is presented in the figure 6.3 is the increase of world themes inspired by either comics or movies.

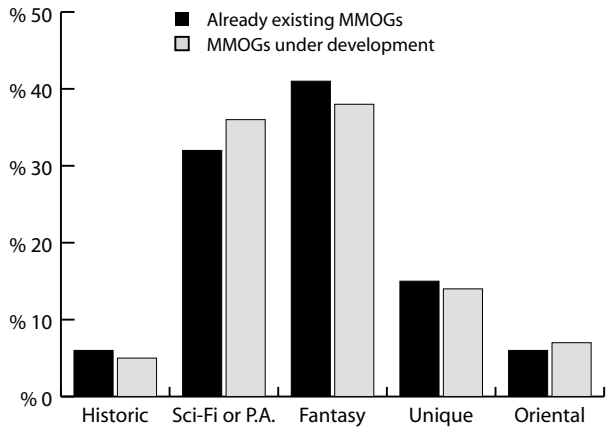

Figure 6.3: Balance among themes of game worlds

As there were 51 different MMOGs available worldwide by April 2003, the fantasy theme genre dominated these which can be observed in figure 6.3. It should be noticed that the percentage numbers representing the already existing MMOGs in figure 6.3 are based on 34 different games out of 51 in total. For the MMOGs under development the percentages are based on 68 different games out of 114 in total. The rest of the games are not of interest as they could for example be vehicle games.

The abbreviations Sci-Fi or P.A. stands for science fiction or post apocalyptic. The description post apocalyptic refers to a scenario in a world where a catastrophe has occurred, e.g. such as world wide atomic war.

Comparing genres of world themes between game titles, those available versus the ones being under development, the percentages are roughly the same. By looking at the figure 6.4, a significant difference can be observed, since the number of real time strategy games is much lower.

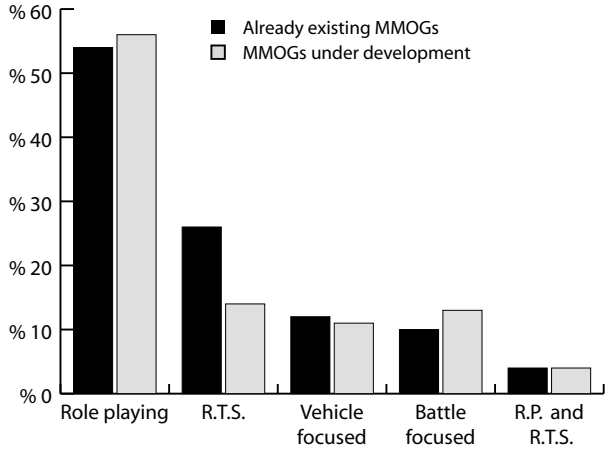

Figure 6.4: Balance among themes of game worlds

The shortenings R.P. and R.T.S. stands for role playing and real time strategy. The percentage numbers for the MMOGs under development are based on 94 different games out of 114 in total. The remaining 20 games which are not included in the figure 6.4 are too early in production to form an opinion of.

6.9 Risk Management

While MMOGs can be a tremendously profitable investment, they are overwhelmed by technical and competitive risks as well. Implementation of the MMOGs is significantly more complex than the implementation of the single player games and actually requires skills that in the existing games industry are not yet very common.

According to a detailed industry analysis, to be able to reduce the risk of investing in a MMOG "IGDA Online Games White Paper 2nd Edition - March 2003" written by the International Game Developers Association, IGDA, Online Games Committee, careful prototyping and an incremental development process that exposes the game to players early and often should be applied.

MMOGs are like any game and software project difficult to schedule accurately. Despite considerable, large financial risks associated with MMOGs which often make the publishers nervous, publishers should not jeopardize entire investment by forcing the game to a schedule which could lead to poor results in the end.

To avoid heavy MMOG competition like Star Wars: Galaxies, EverQuest 2, Middle-Earth Online, etc. new developers should seek after alternative genres for increasing their chances of succeeding.

6.10 Market Analysis for the IFM game

As mentioned previously, the IFM game's subscription fee is USD$5 which will result in achieving USD$60 per year per player which can be observed in figure 6.5. Comparing the total subscribers fee over a one year period for the example represented in the figure 6.1, with the one represented in figure 6.5, there is a difference in USD$96.

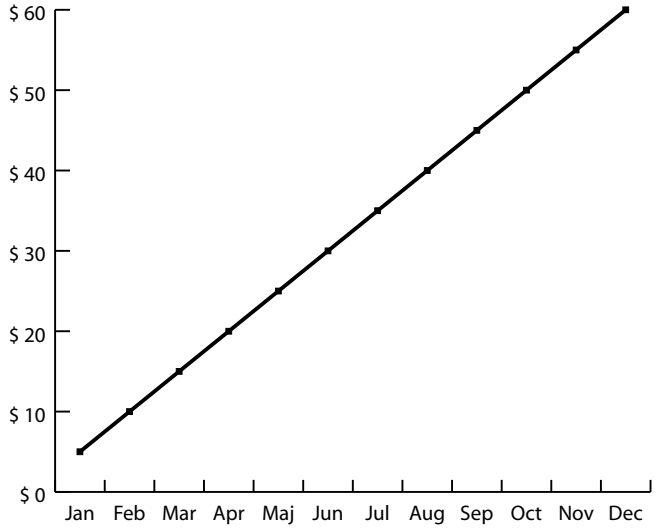

Figure 6.5: Accumulative monthly subscribers fee over one year period for IFM

It should be observed that the amount of money invested in the MMOGs in question differs as the company behind the IFM game has not been able to spend millions of dollars in development and marketing. Hence the IFM game's way of attracting the clients of playing their game is by not charging as much for the monthly subscription fee as the giants in the MMOG market does.

Should the game attract more players than expected, new features will be implemented in the game as well as an increase of the subscription fee.

It took almost three years of development to create the IFM game. Since the start of he development phase, approximately USD$7.500 have been invested in the game. The game itself is the creation of a couple of enthusiastic football game players who have been designing and developing this game during their free time.

Hence, the reason why the amount of money invested is not greater than USD$7.500 is that no salaries have been given to the designers and the developers during the implementation period. The expenses included in the amount of money invested in the IFM game, computer hardware and software investment, telephone and Internet bills, monthly rent for premises, etc.

In the model represented in the figure 6.6, it has been estimated that the increase in the number of clients is going to be exponential by the factor of 35% in the first six quarters whereas a linearly increase in the number of clients is to follow. In this, which in our opinion represents a realistic model, an amount of 500 clients at the point of releasing the game and a maximum amount of 5.000 clients at end of the tenth quarter, have been estimated.

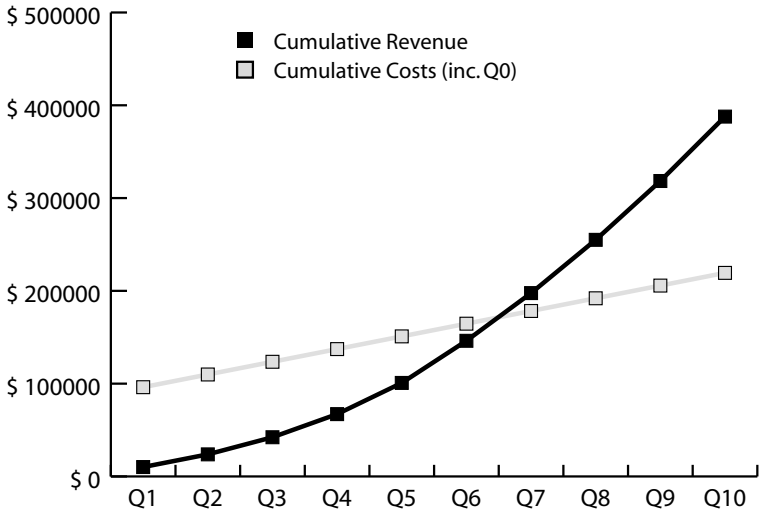

Figure 6.6: Two year revenue versus expense growth for the IFM game

According to this model a cumulative profit of USD$168.500 after two and a half years could be obtained. This profit is calculated out of expenses regarding the development, which includes a developer working full time at USD$9.400 and two developers working part time at USD$3.800 during a quarter of a year. The amount of USD$380 is used for the maintenance of the game, simplified as cumulative costs. In the figure 6.6, these two quantities versus cumulative revenues during a period of two and a half years are represented.

Using the very same model with 40% client increasing factor, 1.100 clients at the point of releasing the game and 9.600 clients at end of the tenth quarter instead, results in a cumulative profit of USD$633.500 after two and a half years. This profit is calculated out of expenses regarding the development, which includes two developers working full time at USD$18.800 and one developers working part time at USD$1.900 during a quarter of a year. The amount of USD$380 is used for the maintenance of the game, simplified as cumulative costs.

Being able to invest millions of dollars in developing and marketing a MMOG, the developers and publishers of these games are taking great risks as they are expecting huge profits at the same time. Despite the huge difference in the money that has been invested between these to models, a sufficiently smaller risk is taken by investing in the IFM game. For that reason the profit made out of the IFM game's model is more than satisfying for the investors.

If the demand of playing the IFM game should exceed the 9.600 maximum available player positions, second team players could be introduced in every team as the junior player activity. Beside these extensions, new servers in the most demanding countries could be set up as well as new worlds within the IFM game could be created which would result in sufficient satisfaction of client accounts.

Chapter 7

MMOGs versus Computer Graphics

7.1 Computer Graphics in MMOGs

Why is graphical representation within the MMOGs of importance? In the history of MMOGs, the first games were of role playing type with poor graphical presentation. This was not because of the non interest from the developers' point of view, but simply because of the bandwidth limit. As computing power became cheap enough, in the mid 1990's, the companies could build powerful PCs, devoted to computer graphics.

As already mentioned, in the IFM game no animation graphics are available. By adding a visualization program for showing the highlights of a football match, the IFM game would be able to market the game in a whole new perspective.

The answer to the question "Why is graphical representation within the MMOGs of importance?" is that users of today want to see more graphics, i.e. more graphics features results in more users involved in the game. This is also a reason why the MMOGs can be found in additional genres.

7.2 Graphical Platform Implemented in the IFM game

The additional graphics feature in the IFM game might not bring additional hardcore manager gamers, but for beginner gamers it would perhaps be the main reason for playing the game. People today are keen on to see action, i.e. animation graphics which would lead to completeness of a game.

Figure 7.1: One animation sequence of a player running with the ball

Figure 7.1 shows a football player running in a football field with a ball. The player itself consists of a texture, se figure 7.2, and a MD2 model that has 21 different animations such as running, shooting etc.

MD2 is the 3D model file format used by the Id Software Company which created the Quake games. In the first Quake game the Id Software Company used the .MDL extension for 3d models where MDL were short for MoDeL. For Quake2 they used the extension .MD2 where MD means MoDeL and the 2 refers to Quake2. MD2 supports key frame animations and this format is the easiest way to use animations. For the game Quake3, Id Software Company uses the .MD3 extension.

Figure 7.2: A texture of a football player

The football field consists of a flat box with a texture applied on it and the ball is a 3DS MAX model without any intelligence, which is simply a polygon net colored black and white.

The challenging task is to arrange these entities, a football player, the football field and the ball to perform a sequence of realistic animations for an event generated by the IFM's game engine. In the following sections, subjects such as physics and AI regarding computer graphics animation in a football game will be discussed. This might, for example, be of interest for ordinary football player gamers who often do not have knowledge of what really is going on behind the screen. It would also bring out the complexity of such a task for making it realistic.

7.3 Game Physics Used in the IFM game's Visualization Program

In the IFM game's visualization program, the challenging part of the implementation is focused on physics. The movement of the ball is defined through two types of paths. Following subsections will discuss the linear and the projectile path of a ball, followed by the ball physics.

7.3.1 Linear Motion

To start simulating a linear movement of the ball, the definition for a line in the three dimensional space, often referred to as 3D space, is used and implemented. The figure 7.3 shows the orientation of the ball in a 3D space.

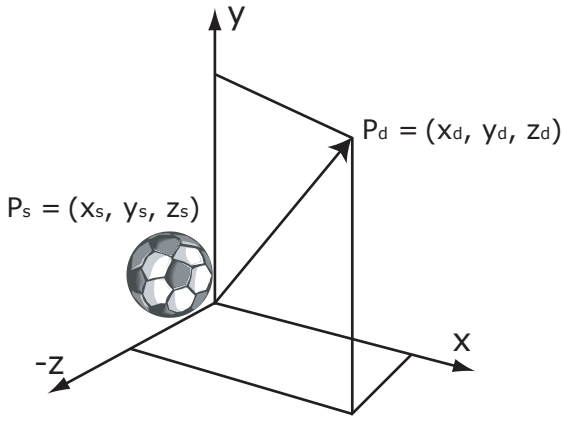

Figure 7.3: The linear motion of a ball in the 3D space

To be able to move the ball, there has to be two data points known. These are the actual position of the ball P_s and the target position P_d, which the ball should reach.

The position of the ball (x, y, z) given a specific time t along the line in the 3D space, due to the above description, is represented below:

$$\begin{cases} x &= x_s + t(x_d - x_s) \\ y &= y_s + t(y_d - y_s) \\ z &= z_s + t(z_d - z_s) \end{cases}$$

Once these data points are known, the parameterization of the line is to be considered. Different parameterization will affect how fast the ball will move from its start position to its destination. If the parameterization is constant during the whole period of time, i.e. constant acceleration, the movement is not going to be realistic. Therefore, some experiments with the parameterization have to be done.

7.3.2 Projectile Motion

To be able to calculate a projectile motion of the ball some simplifications have to be done. For instance, the aerodynamic drag of the ball and the earth curvature and rotation has to be neglected. Figure 7.4 illustrates a projectile movement of a ball which is going to be used when a player attempts to perform a long pass.

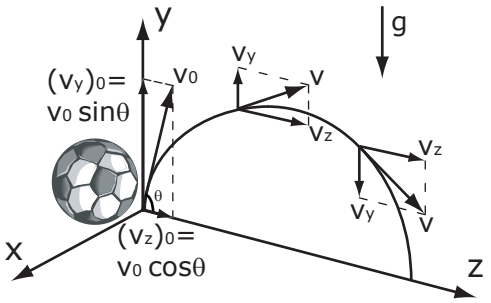

Figure 7.4: The projectile motion of a ball in the 3D space

To be able to determine the path of the ball in a projectile motion way, there has to be four data points known. These are the actual position of the ball P_s, the initial velocity v_0 and the initial angle to the ground θ, of the ball. The direction of the ball is also needed, which is simply determined by knowing the targeting position P_d.

Since this theory is applicable in the 2D space, yet another problem has to be solved, considering 3D simulation of the ball. One solution that was used was applying a `"LookAt"`, function to the ball with the targeting position as the point to redirect after.

The position of the ball (y, z) given a specific time t along the curve, in the 3D space due to the above description, is represented below:

$$\begin{cases} z &= z_0 + (v_z)_0 t \\ y &= y_0 + (v_y)_0 t - \frac{1}{2} g t^2 \end{cases}$$

Even in this case it is very important to choose the parameterization in a way to make the movement realistic.

Next step is to decide the angle and determine the initial velocity of the ball, so that the ball arrives to the desired position. This is simply done by setting up some fixed angles for different kind of passes and then calculating the velocity. For example, a 20 meter long pass should have an angle of 20°. Then the velocity is determined by eliminating t in path equations. The result of the last mentioned is given below.

$$v_0 = \sqrt{\frac{g(z_d - z_s)^2 \cdot (1 + \tan^2 \theta)}{2(y_s - y_d) + 2(z_d - z_s) tan\theta}}$$

Once the velocity for different positions and angles is defined, the only thing that remains beside the parameterizations is to make the ball spin along the projectile path. This is done by changing the x-magnitude in a proper way. The thing one has to consider is that the x-magnitude at the start and finish has to be equal. Figure 7.5 illustrates this scenario.

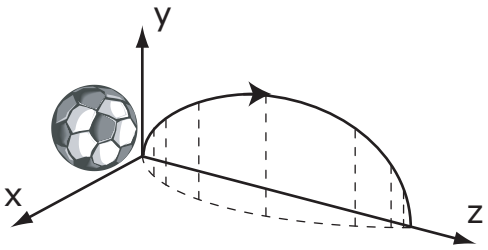

Figure 7.5: The projectile spinning motion of a ball in the 3D space

7.3.3 Ball Physics

Figure 7.6 shows the path of the ball with a 35° angle shot. One can realize that when the ball hits the football field, a bounce motion has to be applied to the ball.

Figure 7.6: An illustration of a football's bouncing pattern

The physics of the ball is much more complex then one can imagine. There are several parameters that affect the ball during its motion. The flight of the ball is determined by Newton's second law of motion:

$$Force = mass \cdot acceleration$$

In the general case there are three forces acting on the ball. These are:

- The force of gravity
- The drag force
- The Magnus force

The drag force and the Magnus force are forces that arise from interaction with the air. The drag force acts in the opposite direction to the ball's velocity and the Magnus force acts in the presence of the spin. With a spin about the horizontal axis the Magnus force can provide a lift of the ball, and with a spin around the vertical axis the flight of the ball is made to bend. To be more realistic in modeling the drag force one has to consider the effect of the wind as well. In the model that is implemented, in the graphical platform for the IFM game, the Magnus force and the effect of the wind are neglected.

During a bounce, the ball initially undergoes an increasing deformation as the bottom surface is flattened against the ground. The deformation is among other factors depending on the ball's velocity and its angle before it hits the ground. There are several parameters that have to be considered regarding the resulting force. The item list below illustrates some of these parameters.

- The initial velocity and angle of the ball
- The ground friction
- Air pressure in the ball
- The area of contact
- The size and elasticity of the ball
- Angular momentum
- The spinning of the ball

In the model that is implemented, in the graphical platform at the IFM game, the bounce of the ball is modeled with the initial velocity and angle of the ball as well as the ground friction. This means that every time the ball hits the ground the initial velocity of the ball in all directions is subtracted by a ground friction constant and then the resulting force is reapplied to the ball at the opposite incoming direction.

Combining the previous mentioned gravity and drag force with the simplified bouncing model one can obtain a surprisingly realistic motion of the ball.

7.4 Artificial Intelligence

The field of game artificial intelligence, AI, has existed since the beginning of video games back in the 1970s. AI in games is usually used for creating the player's opponents. Applying AI in games is equal to rescuing the player from the boredom of repetition and letting him/her focus on the interesting aspects of the game.

Surely, there are games on the market where the opponents AI could be written in a better way. There are two possible scenarios which will result in a player quit playing a game.

The first one is if it is too obvious that the opponents are not playing well or if the opponent AI's are too good so that they are continuously winning. Getting the game balanced is not an easy task.

Because of understanding people's decisions we perceive other people as intelligent. Opponents in a game could end up as emotionally boring characterless creatures sliding around the screen. To make computer opponents challenging one has to try to make a computer think like or better than a human being. Achieving the previously mentioned claim a player has to be provided with more insight considering their actions, intentions, thoughts and emotions for perceiving them as intelligent.

A couple of games worth mentioning which proved the potential of AI were the shooter game Half-Life developed by the Valve Software company and the so called Sim games, such as SimCity. Another Sim game title that is particularly worth noting for the depth of personality of its AI agents is the MMOG Sims Online.

The result of the fact that the developers are nowadays taking the game AI a bit more seriously than before is the great success of recent games such as The Sims. Development of the game AI used to be a last minute rush job that developers implemented in the final phase. Yet another reason for not having well written AI earlier was the fact that the graphics rendering traditionally required a huge part of the available CPU power leaving little memory for it.

7.5 Different Types of AI Techniques

There are many interesting AI techniques that have been developed over the past decades on AI field of research. As the history of the evolution of this field is outside the scope of this section, only a couple of AI techniques that are considered relevant to present and future AI games are going to be discussed.

- *Random Methods*: A simple example of an opponent that uses random methods could be a bee that flies from one part of the screen to another in an impulsive and random fashion. This will of course not result in something of a humanistic nature, but is used quite often in most shooter types of games.

It does not take much to realize that this method is not highly intelligent which will lead the fact that a player would not have any problem of fighting this bee unless there were many of them.

- *Deterministic Methods*: An example of an applied deterministic method is the brown turtle in the Mario Series games that would walk in circles. The brown turtle will move in an x-axis direction until it meets a barrier or a pitfall, where it will finally stop and turn. This action is deterministic in behavior because of the predefined positions in which the turtle cannot go over and turns it to move in a new direction.

- *Tracking Methods*: Applying this method the opponent will respond to the player behavior in a tracking type of way. In other words, the opponent will follow the player which in this case is the object being tracked. An example of an object with applied tracking method is a heat seeking missile or opponent objects that fly towards a player in a game.

- *Predefined Paths*: Suppose that opponents have paths that are fixed such as starting in the top left corner, move a number of steps right, then down, then left and etc. Instead of writing in this fashion, it is possible to write, for example, an array of moves representing the definite path in which the opponent will move. For each increment of the game step the opponent follow the instruction by reading a new element in the predefined array.

- *Finite State Machines, FSM*: As a game object can have a high number of different behaviors, applying the FSM to it simply means that its FSM will occupy exactly one state at any moment. This will generate a game object that will be easier to control in such a way that it would not act unexpectedly, from a developer's point of view.

- *Decision Trees*: An opponent's decision, for example, is being based on a set of inputs by starting at the root of the tree and at each node selecting a child node based on the value of one input. There are several basic search tree methods that can be used in programming the AI in games such as Depth First Search, Breadth First Search, etc.

- *Fuzzy Logic*: This is a method that has been used in AI quite often lately. Simplifying the explanation of the method, one can say that the method is an applied form of statistics in AI. Consider two persons that may find 40°C respectively 30°C, to be very hot temperatures. Since different people have different perceptions, there is actually no right or wrong answers. Fuzzy Logic method deals well with this kind of reasoning, that is non monotonic reasoning.

- *Multi agent systems*: The agents in multi agent systems are considered to be autonomous entities. Their interactions can be either cooperative or competitive. That is, the agents can share a common goal, e.g. an ant colony or they can pursue their own interests.

- *Artificial life*: Refers to multi agent systems that attempt to apply some of the universal properties of living systems to AI agents in virtual worlds.
- *Flocking*: This is a subcategory of artificial life that focuses on techniques for coordinated movement such that AI agents maneuver in remarkably lifelike flocks and herds.

7.5.1 Limitations and Other Considerations

It is important to notice that there are many limitations and other considerations that the developers have to consider when implementing AI in a game. As the computer resources have to be shared with a number of different modules this represents the main limitation in a game. Applying methods such as deterministic methods or predefined paths will certainly help lower processing requirement. Good and robust AI for games is a huge challenge which differs depending on type and genre of games in question.

7.6 Artificial Intelligence in the IFM game Implemented Feature

Managing game AI is one of the most challenging parts of the visualization program that is going to be used for showing the highlights of a football match in the IFM game. The game AI at this stage contains the movement of all team members in a structured formation. The movement of the team should be determined by the position of the ball on the football field.

There are several solutions to obtain a fairly realistic behaviour of the players. A very complex and sophisticated solution might be obtaining more information of players, and add more computation. Yet, another solution might be introducing distributed behavioral models, like flocks, herds, and schools in artificial life.

In the following subsection a very famous simulated flocking algorithm designed by Craig Reynolds in 1987 is illustrated.

7.6.1 Craig Reynolds Flocking Algorithm

The flocking behavior in the Craig Reynolds flocking algorithm consists of two opposing factors. The first one is a desire of each flock member to stay close to the flock. The second one is a desire to avoid collision within the flock. This behavior can be simulated by the following three rules:

- *Collision avoidance*: avoid collisions with nearby flock mates
- *Velocity matching*: attempt to match the velocity of nearby flock mates
- *Flock centering*: attempt to stay close to nearby flock mates

A short overview considering the final result after implementing the Craig Reynolds flocking algorithm is that flock members tend to stay near one another which is the same as flock centering. Flock members always maintain cautious separation from their neighbors which is the same as collision avoidance. They are heading in approximately the same direction at approximately the same speed which is the same as velocity matching. When they change direction they do it in synchronization.

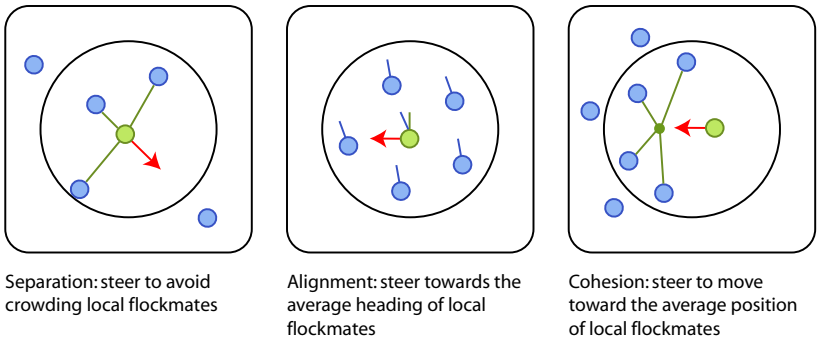

Figure 7.7: Craig Reynolds flocking algorithm

7.6.2 Formation Considered AI Used in the IFM game

By reviewing the Reynolds flocking algorithm in the previous subsection, one realizes that it is not suitable for the movements of a football team because of the formation that the team must hold. Therefore an algorithm for the movement, which considers the formation of the team members, is developed. The consideration of a traditional 3-4-3 formation is managed by dividing the football field in 35 equally squares where each player's position, depending on the position of the ball, is predetermined. The figure 7.8 shows the distribution of the squares on the football field.

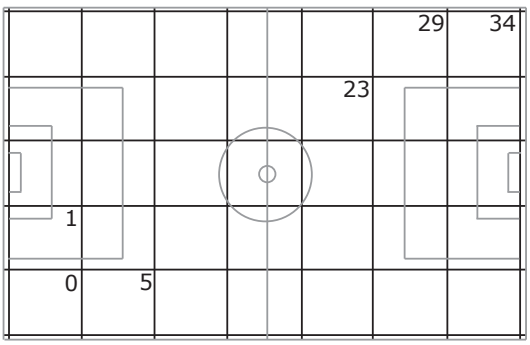

Figure 7.8: The distribution of the squares on the football field

The algorithm uses the main loop, i.e. a simple thread, which makes it non complex. The pseudo code below illustrates the idea behind the algorithm.

```
for each frame {
  determine the position of the ball

  determine on which square the ball is positioned

  for player 1 to 10 {
    determine the target position of the player -
    in comparison with the ball

    calculate the movement step on each axis

    if players position near target position
      do not move
    else
      move the player one step
  }
}
```

The goalkeeper's AI is at this stage considered as a simple observer that follows the ball on the goal line.

7.7 Blitz 3D

Blitz 3D always seems to be mentioned in comparisons to other programs of the same genre as soon as 3D programming is being discussed. Blitz Basic was released in October 2000 and Blitz 3D in September 2001. Blitz 3D was developed by Amiga games programming legend Mark Sibly and this programming language has a history of being stable and judging by the Forums it will have a large following of faithful users in the future.

Our impression during the time period where the feature concerning the IFM game has been developed is that the programming language is easy to learn. The truth is that a majority of 3D games available on the market are built around the C++ language, but Blitz 3D is easier to grasp and can handle 3D programming much easier than C++.

Blitz Basic itself is written in Visual C++. Many of its commands are implemented just for the 3D environment which makes them more efficient than one might expect.

Blitz 3D provides its users with the possibility using many of the DirectX capabilities and full Network TCP/IP and UDP support, as well. As far as 3D graphics is concerned, following features are available in Blitz 3D:

- Mipmapped Textures
- Transparent Textures
- Animated Texture Support
- Multi Textures
- Deformable Meshes
- Level of Detail Terrain
- High Speed Collision Engine
- Directly load of .X, .3DS or .MDL files
- Support for MD2 Animation
- Vertex Coloring
- A Flexible Entity System
- Reflection Effects
- Multiple Cameras

According to the online Forums what puts Blitz 3D slightly ahead of the other 3D programming languages is its speed. Being satisfied with the programming code implemented one is always keen on seeing the created graphical results. Blitz 3D's compile time is one of the fastest for programs in this genre.

Blitz 3D uses a single runtime library for everything, which is compiled directly into an executable file, this makes the executable file fairly large, but it does also mean that one does not have to distribute DLLs with the game created.

This is done partly to speed up development, as Blitz Basic focuses on program speed and stability before RAM and disk storage space, which is generally not much of an issue these days.

Blitz Basic also has the Maplet program which can be added. With Maplet one can easily create 3D indoor levels that can then be implemented into the Blitz 3D applications.

7.8 Blitz 3D in Comparison to Other Programming Tools

During our education at Lund Institute of Technology we have been in touch with OpenGL and Crystal Space.

Well, first of all it is important to understand that Blitz 3D is a programming language. Blitz 3D is not an engine, it will do things such as load 3D objects and textures, but it is up to the programmer how he/she will get to use them in a game. Blitz 3D contains instructions that one can use to write your game/program, such as `LoadMesh("model.3ds")`, which will load in an entire 3D model.

Crystal Space on the other hand is an open source 3D game engine. In general, open source refers to any program whose source code is made available for use or modification as users or other developers see fit. Open source software is usually developed as a public collaboration and made freely available. Crystal Space works like a sort of a pre-built game. One could think of it as a sort of source for ones game ideas.

Beside from the fact that Blitz Basic and Crystal Space are used for creating 3D games they also share the fact that they are both written in Visual C++.

Both OpenGL and DirectX are graphics APIs and are used in the games one is able of buying today, depending on which API one would like to use. As already been explained in "Database Systems for MMOGs" chapter, API stands for Application Programming Interface. API can be used for creating and managing graphic images and multimedia effects in applications such as games.

Blitz 3D is written for the DirectX and it is a kind of bridge between a user and DirectX. It allows one to use the full power of DirectX without having to fully understand how, or what is going on under the hood of it, so to say. There is also a possibility of using OpenGL in Blitz 3D by simply getting an OpenGL plug-in.

As Blitz 3D is written for the DirectX it is only possible to run it on Windows platforms with an official version of DirectX 7 or above installed. This means that Windows 9x, Windows Me, Windows 2000 or Windows XP can be used as a platform. Windows NT4 is not supported by Blitz Basic or Blitz 3D, as it only has DirectX versions up to 3.

A great advantage in using Crystal Space lies in the fact that it is able to run it on GNU/Linux, general Unix, Windows, Windows NT, and MacOS/X platforms. Crystal Space can also optionally use OpenGL on all mentioned platforms as well as others.

One can probably get a very simple game up and running in fewer than 20 lines of Blitz 3D code. To do that using OpenGL or for that matter Crystal Space, one would need lines and lines of code.

7.9 Established Implementation Overview

Our final result over what could be described as solid foundation for those interested in developing a football game is summarized in seven different game demos. These demos illustrate what we have been able to do. The first demo contains following parts:

- *Running Player Animation*: The ability of handling a running player animation is represented. With this knowledge we are hereafter able to handle all type of different player animations.
- *Ball Handling*: A ball is positioned close to the player. The ball follows the player as a player can be moved around the football field using the arrow keys on the keyboard.
- *Manual Camera Positioning*: A camera is installed and can be regulated manually in x-, y- and z-axis.

The second demo contains following parts:

- *Automatic Camera*: To be able to have a feature that is capable of automatically handling the graphical representation of the highlights in a football match an automatic camera management is needed, which is exactly what we have established in this demo.
- *Camera AI*: The camera management is programmed in such way that it sees the football field as a field divided in several minor segments. A finite state machine, FSM, have been used for modeling this kind of AI.

The third demo contains following parts:

- *Team AI*: An illustration of a team AI is represented. One is able to move the ball with the arrow keys on the keyboard which will result in specific individual movement among players, within the team, towards the ball.
- *Automatic Camera & Camera AI*: The camera AI developed in the previous demo is used here for following the ball around the football field.

The fourth demo contains following parts:

- *Ball Physics*: A representation of physics considering the ball are shown in this demo. It is worth mentioning that our ball physics parameters are fewer by number in comparison with the number of parameters in reality. Despite this fact, our model is behaving like a realistic one.
- *Shooting Animations*: Being able of managing the ball physics we chose to demonstrate this by letting our player kick the ball in different directions and letting it bounce against the football field.

- *Different Shoot Angles*: Adding more excitement to this demo, the ability of choosing the shoot angle on your own is added.

The fifth demo contains following parts:

- *Ball Finding AI*: First of all, this demo contains two players. After having a player kicking the ball in a certain direction the other player's AI triggers an automatically ball finding algorithm.
- *Manual Passing*: After having a player reaching the ball destination one is now once again able of shooting out the ball which the other player automatically will aim to reach.
- *Player - Ball Collision*: Using our selves of invisible collision spheres, which could also be referred as bounding volumes, attached at the players' bodies and at the ball's centre, we are able of detecting a collision between a player and the ball.

The sixth demo contains following parts:

- *Automatic Ball Passing*: This demo also includes two players. The big difference here is that this demo is automatically run by itself. We have, in other words created a visual representation of what could be a certain scenario in a football match.
- *Player Turning AI*: In this demo we added a turning sequence which will be executed as soon as a player gets in touch with the ball. This adds an extra bit of realism in this demo.
- *Sound Effects*: A demonstration of sound effect handling is shown in this demo as one is able of hearing a sound as a ball gets kicked by a player as well as the ball bounces against the football field.

The seventh demo contains following parts:

- *Automatic Ball Heading*: Here we have a scenario where we have a player who is about to perform a free kick. He is positioned at a certain distance from what is representing a player blocking wall. As soon as he performs a free kick, the player blocking wall, consisting of four players, are going to try to block the ball by jumping a bit from the ground and perform a forward heading.
- *AI for Finding Jump Position*: An algorithm is triggered for finding the right position for performing the heading. Goalie AI is also available in this demo as he follows the position of the ball continuously, goalies "saving the ball" animation is added as well.

- *Sound Effects*: A demonstration of sound effect handling is demonstrated in this demo as well. Besides from what has been implemented in the previous demo, a background sound is added presenting the attending crowd at the stadium. This is accomplished by having different sound channels managing different sound actions. A heading sound is also added as the player, having a role of a forward, heads the ball towards the goal. The sound of a cheering crowd appears as the ball passes the goal line.

7.10 Current and Future Work

Concerning what have been established during this 20 week period there are huge posibilities for future work. First of all, our thesis truly constitutes an introduction for those who are intreseted in the MMOG field and it reflects all the steps one needs to take in consideration when creating a MMOG. Everything from designing techniques concerning MMOGs to industry analysis are discussed.

Advantages and disadvantages of the programming language, Blitz 3D that we have been using is also brought up. This will definetelly make it easier when it comes to choosing the programming language that one would like to use for his/hers game.

The Blitz 3D code that we have produced is public, which means that anyone interested can take part of what we have acomplished.

Chapter 8

Bibliography

8.1 Literature

[Gollmann99] D. Gollmann, "Computer Security", John Wiley & Sons., pp. 245-264

[Alexander03] T. Alexander, "Massively Multiplayer Game Development", Charles River Media, Inc., pp.1-90, pp.211-362, pp.365-406

[Watt-Policarpo01] A. Watt, F. Policarpo, "3D Games", ACM Press, An Division of the Association for Computing Machinery, Inc. (ACM), pp.581-584

[Mulholland-Hakala02] A. Mulholland, T. Hakala, "Developer's Guide to Multiplayer Games", Wordware Publishing, Inc., pp.181-241

[Silberschatz-Galvin99] A. Silberschatz, P. Galvin, "Operating System Concepts", John Wiley & Sons, Inc., pp.89-235

[Tanenbaum96] A. Tanenbaum, "Computer Networks", Prentice Hall, Inc., pp.1-77

[Garcia-Molina02] H. Garcia-Molina, "Database systems", Prentice Hall Inc., pp.1-188

[Zona02] Zona, Inc., "State of Massive Multiplayer Online Games 2002: A New World in Electronic Gaming, A detailed industry analysis and market projection by Zona, Inc. and Executive Summary Consulting, Inc.", pp.1-50

[IGDA03] The International Game Developers Association, "IGDA Online Games White Paper - 2nd Edition, Written by the IGDA Online Committee", pp.1-140

[Lengyel02] E. Lengyel, "Mathematics for 3D Game Programming & Computer Graphics", Charles River Media, Inc., pp.271-293, 1-53

[Wesson02] J. Wesson, "The Science of Soccer", IOP Publishing Ltd, pp.45-68, 143-185

[Watt-Policarpo01] A. Watt, F. Policarpo, "3D Games", ACM Press, An Division of the Association for Computing Machinery, Inc. (ACM), pp.484-515

[Bourg02] D.M. Bourg, "Physics for Game Developers", O'Reilly & Associates Inc., pp.101-120

8.2 Homepages

www.internationalfootballmanager.com

www.cert.org/tech_tips/denial_of_service.html

www.cert.org/advisories/CA-1996-21.html

www.cert.org/advisories/CA-1996-01.html

www.itsecurity.com/dictionary/certificate.htm

www.blizzard.com/support/?id=msi0504p

islab.oregonstate.edu/koc/ece478/proj/2002RP/KZ.pdf

www.digitalgamedeveloper.com/2002/11_nov/features/dlmmogd1126023.htm

www.digitalgamedeveloper.com/2002/10_oct/features/dlmmog101102.htm

www.whatis.com

www.xml.se/xml/REC-xml-19980210-sv.html

www.w3schools.com/default.asp

www.game-research.com/art/_trends/_in/_mmog.asp

www.themis-group.com/view/_news.phtml?id=13

siliconvalley.internet.com/news/article.php/1479321

www.dfcint.com/news/prjune252003.html

www.digitalgamedeveloper.com/2002/11/_nov/features/dlmmogd112602.htm

www.zona.net/whitepaper/

www.igda.org/online/online/_whitepaper.php

www.sharewarejunkies.com/03zwd2/blitz_3d.htm

www.blitzbasic.com/

www.openskies.net/files/Openskies_Network_Architecture.pdf

www.openskies.net/files/Openskies_MMPOG.pdf

www.openskies.net/files/Openskies_Scalability_Whitepaper.pdf

www.cert.org/tech_tips/denial_of_service.html

news.com.com/2100-1040-276911.html?legacy=cnet&tag=mn_hd

alive.znep.com/ marcs/passport//

news.com.com/2100-1017-238900.html?legacy=cnet

online.securityfocus.com/archive/75/190653

www.stratics.com/content/mmogweekly/mmog.shtml

www.red3d.com/cwr/boids/

www-cs-students.stanford.edu/ amitp/gameprog.html

ai-depot.com/GameAI/Design.html

ai.fri.uni-lj.si/ aleks/FT/

www.mmog.com

www.mmorpg.net/

www.mpogd.com/