CENG 490
Senior Project

"A 3D – Massively Multiplayer Online Game"

Requirements Analysis Report

by

Ömer Akyüz e1347079
Önder Babur e1347186
Süleyman Cincioğlu e1347277
Güneş Aluç e1462670

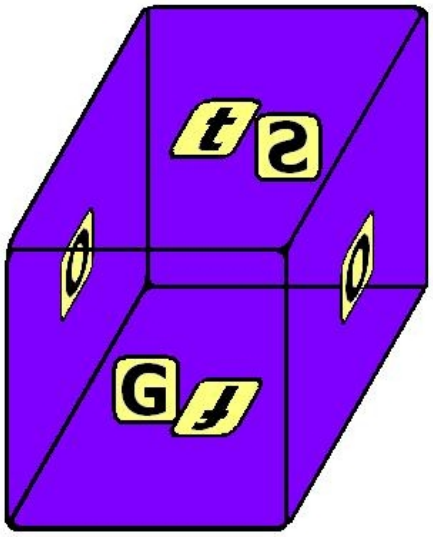


Table of Contents

Table of Contents

1 Introduction

1.1 Overview

This report provides an insight into the requirements analysis phase GOSOft went through for the "A Massively Multi-player Online Game with 3D Graphics" project topic. Initially the group members made some brainstorming to come up with a creative idea as to how the topic could be turned into a working product. After all, creativity plays an essential role in game development. Later on, the group members conducted a thorough research on the topic, including market analysis as well as some technical research. A brief summary of the useful information obtained during the research stage can be found in the relevant sections of the report.

The research data left the group members with several engineering problems to solve and decisions to make. Implementing a 3D-MMOG game consists of integrating various components such as Network, Game Engine, Computer Graphics and Artificial Intelligence. There are various methods to follow in implementing each of the four components. Consequently there are also some trade-offs to preferring one to the other. What the group did – which is presented more formally in this report – was first to determine the requirements that must be satisfied for a functional product. Besides its technical aspects, social (user friendliness) and economic (the MMOG market) factors were also taken into consideration while determining the requirements. Later on, the group members combined the research data with some of the requirements criteria to come up with some very vague design guidelines. They can also be found in this report.

The requirements analysis efforts were supported by several formal methods such as drawing Use-Case Diagrams and Data-Flow Diagrams. These can also be considered as preparatory efforts for the design phase. Additionally some risk management and quality assurance criteria as well as the process model have been set and agreed upon to lead to a smoother development process.

1.2 Project Definition

For our senior project, we will be implementing a 3D Massively Multi-player Online Game that is capable of simultaneously supporting hundreds of online players. The current Massively Multi player Online Game (MMOG) market is dominated by Role Playing Game (RPG) and First-Person Shooter (FPS) games. There are also a few examples of strategy games in this field. Since implementing multi player online games in a 3D environment is a difficult task, the examples of 3D MMOG games are limited, most of them being commercial. The reason why MMOGs have become so popular is that a person can play this game with other people and even his/her friends interactively. The 3D feature is an add-on to the environment. In this market, creative ideas are more likely to survive since they have a capability of leaving a stronger impact. We believe that our idea, whose details are provided below, is innovative in this respect, especially regarding the standard approaches.

The game to be implemented in this project should be capable of supporting more than a hundred online players simultaneously. This implies that a strong network architecture is essential for a working product. Thus, before proceeding with any design procedure, an in-depth analysis of existing architectures is essential.

Furthermore, the online players in the game are expected to interact with each other. This is actually the key point in producing a game that can attract a large number of people. For this purpose, the game should provide facilities in which one player's action should affect others'. However, the desired condition is that the scenario or the concept of the game **forces** the players to interact. Although, it is true that the server will suffer from this functionality, as it is a necessity, it cannot be neglected.

The game will be played in a 3D atmosphere that is rendered as realistic as possible. A good network architecture that separates graphics rendering from the game engine, or equivalently in this case the client module from the server, allows high-quality 3D graphics to be created and to be played without much of a network concern.

Although our initial view is that its integration to our game scenario will be difficult, the role of Artificial Intelligence cannot be neglected. Unlike in strategy games, our game – whose scenario details are given next – does not contain a solid set of rules that can guide the AI unit in resolving its search tree. The game to be implemented is mainly based on puzzle solving and for the AI player to become a part of it, some relations should exist between the puzzles. In reality they do not and this is the difficult part. To overcome this difficulty, the puzzles will be associated with each other by ontology mappings.

1.3 Project Scope

For this project, our goal is to implement an adventure game from the first person's point of view. The game scenario is based on the movie "The Cube[1]" in which the actors are trying to get out of a very large cube shaped building consisting of n*n*n – k rooms. The rooms are also cube shaped and they are similar in view. Furthermore, on regular intervals, the rooms in the construction change their absolute position inside the cube.

The primary goal of the players in the game is to eventually get out of the cube. This, as one might guess, is not an easy task. The rooms are filled with tricks, booby-traps and several puzzles to solve. Even if the players are very close to the exit location, just before they move out of the room, that room's absolute position might change. In our opinion, this will add some flavor into the game.

To safely move from one room to the other, the players need to collaboratively solve the predefined puzzles. It is different from the original movie in this respect. After all, in the movie, the actors try to avoid the booby-traps by solving the puzzles. On the contrary, in our scenario, the players unlock the doors by solving the puzzles. Of course, not all actions performed by the players lead to a solution. Additionally sometimes the actions that are performed might harm the players, thus they need to be very careful.

The puzzles can be solved directly or indirectly interacting with the objects in the environment.

1 Cube. Natali, Vincenzo. Movie. http://www.imdb.com/title/tt0123755/

Therefore, exchange of ideas between the players becomes very crucial at this point. By direct interaction we mean actions such as clicking on a wall item, using an object from the object inventory and various combinations of each. Indirect interaction refers to the personal experience gained in interacting with the environment objects. For example, there might be a written text on the wall that has clues as to how the puzzle for that room (or for some other room) can be solved. When a puzzle is solved, the players will be able to move to a different room. Although the players may prefer moving in groups, such a restriction is not imposed. Therefore, they might leave others behind. When they move inside the cube, the players will either encounter rooms with other players or some empty rooms.

As in almost every game, there will be some artificial intelligence present. However it will be implemented in a different way. In almost every occupied room there will be some (one or two, usually one) AI players present. They will mainly be moving with the group and performing actions together with them. The key role of the AI player will be to assist the other players in solving the puzzles. The AI players will usually have some background experience. After all, they will have moved with different people that have solved various puzzles. As the puzzles are solved, the AI players will learn from the actions that led to the solution. For this purpose, several predicate clauses will be associated with each of the actions that can be performed. As the AI player's search tree grows wider and deeper, it will be able to come up with better suggestions as to how the puzzle in the current room can be solved. Therefore, it will assess the objects and the actions that can be performed on these objects and later on evaluate its search tree to find if anything similar matches. The AI player can make a suggestion through the chat functionality or by answering questions asked by other players, a technique known as “interrogation”.

We now provide an example room definition to enhance the understanding of the game scenario. Suppose in one of the rooms there are four water valves that can each be turned on and off. Embodied under the valves, there exist four rectangular prism shaped cups and below them four drawers. On their own, the drawers are locked. The condition for unlocking the drawers is to make the cup above it filled with water and to leave all the other three empty. There is no other possible way of opening the drawers. The players will see a key in each of the drawers that they can use for unlocking the doors of the room. At this stage, the AI player can make the suggestion of emptying out some of the drawers through interrogation or if the players are lucky by some public chat message. This is just a simple example showing how a room can be like. Rooms can contain more than one puzzle and even booby-traps. Thus, deaths are inevitable. Fortunately though, when a player dies, she or he has the chance to continue playing, but in another room.

1.4 Goals and Challenges

Within the scope of the project the general goals can be classified as follows:

- Maintaining the game-state consistent,
- Ordering the events,
- Increasing interactive responsiveness,
- Providing realistic rendering of 3D components,
- Integrating artificial intelligence into the game.

The above functionalities can be used in setting up the roadmap for developing the MMOG for the senior project. Other goals are also present, but we will just provide them as challenges. In our design we will try to leave some open doors to the following items:

- Displaying animations,
- Scheduling computations across players,
- Providing authentication/authorization mechanisms,
- Implementing a cheat-proof design.

2 The Process

2.1 Process Model

Regarding the time constraints dictated on the project, it is essential to adopt a robust software process model. Hence, the waterfall model seems most suitable for the project with its linear sequential structure meeting the deadlines of the project phases, and its high reliability. But since a prototype of the product is expected right after the design phase, without thorough test and documentation concerns, the fixed structure of the model has to be disturbed in order to achieve the construction of the prototype. The modified model has the implementation phase with the initial construction of a working prototype at the beginning, meeting with the minimal requirements and integration concerns.

2.2 Team Organization

Although the nature of the project tends to favor Democratic Decentralized team structure, open to new ideas and creativeness, it is somewhat obligatory to adopt a more strict model to achieve timely release of documentation at the end of each process phase. Since it might be disastrous to employ too much control, it seems wiser to distribute the major tasks among the team in groups of two, where horizontal communication is encouraged. The groups are strictly responsible to project manager while cooperating within the group, thus achieving a Democratic Controlled structure.

Internal Organization

Project Roles:

- Game Engine: Önder Babur, Süleyman Cincioğlu
- Graphics Engine: Süleyman Cincioğlu, Güneş Aluç
- Network: Güneş Aluç, Ömer Akyüz, Önder Babur
- AI: Ömer Akyüz, Önder Babur
- UI: Güneş Aluç
- Art: Süleyman Cincioğlu
- Audio: Önder Babur
- QA&Testing: Önder Babur, Ömer Akyüz, Süleyman Cincioğlu, Güneş Aluç
- Puzzle design: Süleyman Cincioğlu
- Website design: Ömer Akyüz

Other roles are distributed among the whole team in a democratic manner.

2.3 Major Constraints

Time Constraints:
There exists strict deadlines for each phase of the process, so the constraint of meeting the deadlines is of utmost concern.

Attribute Constraints:
As a common denominator in game programming, it is best to use C++ as the main programming language as the server side component. The components to be used/integrated, hence the overall design is considered around the core C++ component.

Tool Constraints:
Using Graphics and AI Engines leads to limitations within the corresponding domain according to the implentation of functionalities. While engines reduce the burden of low-level programming, they introduce constaints to the capabilities of the project.

Network Constraints:
Since the aim is to be able to support a great number of players, the effective transmission of game data is exposed to network connection speed constraints, which greatly affects the project design.

Graphics Constraints:
A good frame rate is to be guaranteed in order to have a decent display and thus gameplay. This places constraints of to what extent the graphics module complexity can be pushed.

Personnel Constraints:
Having a small project team greatly limits the development of a larger scale project. The project is subject to balanced distribution of the limited workforce.

3 Research

3.1 Market Analysis

A **Massively Multiplayer Online Game** (**MMOG** or **MMO**) is a computer game which is capable of supporting hundreds or thousands of players simultaneously, and is played on the Internet. Typically, this type of game is played in a giant persistent world.

MMOs enable players to compete with and against each other on a grand scale, and sometimes to interact meaningfully with people around the world. Most MMOs require players to invest large amounts of their time into the game. Although MMOs are played by very large amounts of people, it is still one of the youngest game generations in the game industry and it is very rapidly developing and every day new things are added to satisfy the players.

There are a number of factors shared by most MMOGs that make them different from other types of computer games. MMOGs create a persistent universe where the game continues playing regardless of whether or not anyone else is. Since these games strongly or exclusively emphasize multiplayer game play, few of them have any significant single-player aspects or client-side artificial intelligence. As a result, players cannot "finish" MMOGs in the typical sense of single-player games. Some MMOGs, such as *Star Sonata*, do have an end condition that includes awarding a "winner" based on a player's standing in the game at the finale. But these examples are very rare and most of the times the players in the game do not intend to finish the game. They even like to feel as if they are living on another universe. For most of the MMOG players, the game they are playing has become a major part of their lives.

Most MMOGs also share other characteristics that make them different from other multiplayer online games. MMOGs host a large number of players in a single game world, and all of those players can interact with each other at any given time. Popular MMOGs might have thousands of players online at any given time, usually on a company owned server. Non-MMOGs, such as *Battlefield 1492* or *Half-Life* usually have less than 50 players online (per server) and are usually played on private servers. Also, MMOGs usually do not have any significant modes since the game must work on company servers. There is some debate if a high head-count is the requirement to be a MMOG. Some say that it is the size of the game world and its capability to support a large number of players that should matter. For example, despite technology and content constraints, most MMOGs can fit up to a few thousand players on a single game server at a time.

To support all those players, MMOGs need large-scale game worlds. In some MMOGs, large areas of the game are interconnected so that a player can traverse vast distances without having to switch servers manually. For example, *Tribes* comes with a number of large maps. A server plays in rotation (one at a time), but in the MMOG *PlanetSide* all map-like areas of the game are accessible via flying, driving, or teleporting.

There are also a few more common differences between MMOGs and other online games. Most MMOGs charge the player a monthly or bimonthly fee to have access to the game's servers, and therefore to online play. Also, the game state in an MMOG rarely ever resets. This means that a level gained by a player today will still be there tomorrow when the player logs back on. MMOGs often feature in game support for clans and guilds. The members of a clan or a guild may participate in activities with one another, or show some symbols of membership to the clan or guild. This is one of the most loved characteristic of the MMOGs, players like to belong to some clan or guild and compete with each other massively by these clans. There are especially a group of players who plays these games because of its clan feature, they want to fight against a group with his own group, these clan wars (as if wars between two countries) attract these group of people to MMOGs.

There are several types of massively multiplayer online games. Massively multiplayer online role-playing games, known as MMORPGs, are perhaps the most famous type of MMOG (*Everquest 2*, *Lineage 2*, *Toontown Online*, *World of Warcraft*, *Guild Wars* etc.). Browser Based Massive Multiplayer Online Role-Playing Games are MMORPGs played through an internet browser, saving the developer the hassle of creating a client for its game, and the players the hassle of downloading one (Starkingdoms, Bushtarion, Kingdom of Loathing, StarSphere etc.). Several MMOFPSs first-person shooters have been made (World War II Online, *PlanetSide* etc.). These games provide large-scale, sometimes team-based combat. A number of developers have attempted to bring real-time strategy games into the MMOG fray (*Mankind*, *Shattered Galaxy* etc.). Other types of MMOGs are not as popular as the ones we stated above[2].

3.2 Technical Research

3.2.1 Network

3.2.1.1 Overview

As the amount of information that must be transmitted over the network is quite extensive, a thorough analysis of the existing models is essential before implementing a network architecture that is capable of supporting a 3D massively multi-player online game. The research data shows that a good network architecture of a 3D MMOG should:

- maintain the world state consistent throughout the whole nodes on the network,
- allow interactive responsiveness to be as quick as possible,
- prioritize and order events,
- provide authentication and authorization mechanisms and prevent players from cheating,

2
"Massively multiplayer online game". Online. http://en.wikipedia.org/wiki/Massively_multiplayer_online_game.
"Game Ogre: The Online Gaming Mega Site." Online. http://www.gameogre.com/mmorpgs.htm.

- be highly scalable.

Although it is difficult to tackle all of the aforementioned challenges at once, a good design can – by reducing traffic over the network – eliminate most of them. The following methods can be applied to reduce traffic and thus increase the game performance:

- Static game data should be separated from the dynamic one and it should be transferred to the network nodes only once;
- Compression techniques should be used to reduce network load. However, this puts some extra computational load on the nodes;
- The nodes should receive only what they need to receive and not more. In other words, only the relevant information should be sent.
- If the state of an object remains unchanged, then the associated update message should be delayed.
- For cases when object updates cannot be accomplished all at once, update priorities should be considered.

In addition to the above statements, research findings indicate that not every change in the state of an object require an update message to be transmitted. In a first-person-shooter game for instance; unless the player changes his/her direction, no update messages are necessary. In other words, it would be useless to send the message: "player45 is still moving east", over and over again. Furthermore, complex techniques such as artificial intelligence can be utilized for reducing message complexities. For example, simply a "move to (x, y)" command can be sufficient in cases when AI can resolve it to simpler "move" commands with path-finding algorithms. However in this case, there is a trade-off between computational complexity and network load.

3.2.1.2 Available Models

During the research phase, we have come across the following models by which the network aspect of the 3D MMOGs can be implemented. Namely, and generally speaking, these models are:

- Client-server model,
- Publisher-subscriber model, and
- P2P architectures.

The publisher-subscriber model is sometimes known as the producer-consumer model. Although the basic idea behind them is almost the same, every model has some variants. For instance, the client-server model can be extended by using a group of machines, each assigned to a unique task, instead of a single one in the server side. Below you will find more detailed information on each approach.

The Client-Server Model

The client-server model has been in use for a long period of time in game development. The multi-player version of the Half-Life game can be given as an example to it. It is considered to be a small-scale multi-player game where the maximum number of concurrent players is around 60. In the client-

server topology, a machine that is powerful in terms of hardware and software capabilities is assigned the role of the server and all game-logic functionalities, the world state and other utilities such as authentication/authorization are implemented on it. The clients communicate with the game server to receive up-to-date information that enables the client application to make the game available to the players.

As in every model, the client-server model has its own advantages as well as drawbacks, which are outlined below:

Advantages:

- A single authority is used for the game engine and the communications, therefore, world data can be kept in a consistent manner,
- There is a central data repository, this is advantageous for consistency, processing efficiency and space,
- A network implemented with the client-server architecture is easy to secure.

Disadvantages:

- Unfortunately, the virtual world's complexity and size has to be limited by the functional capabilities of the server,
- The client-server model creates network traffic with a high-bandwidth,
- Usually in such architectures player-to-player interactions are limited,
- For a practical system, large amounts of money has to be spent on hardware.

During the research made on the Network aspect of 3D MMOGs, two different approaches for extending the client-server model were encountered. They both intend to reduce the load on the server and thus increase overall game performance. The first one is based on the idea of splitting the server into logical modules. For an MMOG, some possible logical groupings are:

- login server, game server, chat server, patch server or
- data repository, physics engine and artificial intelligence.

The second approach – namely the proxy-server model – is a consequence of the physical restrictions imposed by the network itself. Identical proxy servers each containing the same game state are situated in various countries around the globe and clients connect to the physically nearest station. With this approach, keeping the game state consistent in all of the proxy nodes becomes a challenge.

The Publisher-Subscriber Model

The client-server architecture allows one-to-one communications. In other words, although the server exchanges messages with a number of clients; the messages exchanged by each client are usually distinct. For the server, each client serves as a different, unique entity. On the other hand the publisher-subscriber model has been put forward to capture one-to-many and even many-to-many communication patterns. In this context, a “one-to-many” communication pattern implies that the server is associated with a group of clients. Although each group might have distinct characteristics, the members within a single group are treated as one entity. The messages exchanged by the members of a single group are

usually the same.[3]

The publisher-subscriber model is more suitable to cases where information needs to be sent to a large number of consumers (the model is sometimes referred to as the producer-consumer model). In this case, the clients subscribe themselves to entities called channels, along which the messages are transferred. Whenever a message appears in the channel all consumers can receive it. There are push/pull versions of the model, whose details are skipped in this report.

If the MMOG can be divided into logical (or even physical) sections such that it is sufficient that a client application knows what is relevant to it but nothing more, then the publisher-subscriber model can be applied. Splitting the map by means of hexagons and rectangles or separating environmental variables from the user interaction data are some examples of physical and logical divisions.

The P2P Model

The P2P model is relatively a new architecture and its utilization in MMOG implementations is still a hot research subject. Some researchers believe that it has various advantages over existing architectures and that it will lead to the development of MMOGs with a much higher interactive responsiveness.

In a P2P network, every node has the same responsibility and privilege as every other node on the network. Considering its utilization in MMOG development the it is indicated in several research papers that the P2P model has the following advantages:

- computation is spread over the whole P2P network, therefore the average workload is less than that of the server's in a client-server model;
- the consumed bandwidth is reduced: when there is a client-to-client communication in the client-server model, the message has to go through the server thus doubling the consumed bandwidth.

Unfortunately, the P2P model has the following disadvantages:

- security (authentication / authorization) issues become more difficult;
- maintaining world-state consistent becomes a major problem.

3.2.2 Graphics

3D graphics is one of the most important aspects of current game industry. It is the most rapidly growing part of the game development business, every day the 3D graphics in the games are getting better and better. They have become so popular that specialized APIs have been created to ease the processes in all stages of computer graphics generation. These APIs have also proved vital to computer graphics hardware manufacturers, as they provide a way for programmers to access the hardware in an abstract way, while still taking advantage of the special hardware of this-or-that graphics card. These APIs for 3D computer graphics are particularly popular: OpenGl, DirectX, RenderWare etc. Game developers develop their 3D game environment according to these APIs. But instead of using these

3 Fiedler, Stefan. Wallner, Michael. Weber, Michael. “A Communication Architecture for Massive Multiplayer Games.”

tools plainly, they use some powerful graphics engines to make their lives easier. These engines use different techniques to make games more than just games but like an artwork, for example Oblivion The Elder Scrolls 4 is a very good example of these games which was released by Bethesda[4].

Most of the time graphics engines are considered with game engines, since every game engine has a graphics engine inside, but there are also some independent graphics engines. The best examples of these engines are Ogre3D, Axiom, Power Render and Crystal Space. We have researched mostly on these graphics engines in order to use in the project. After the research we decided to use Ogre3D engine as our graphics engine. Here are some merits and flaws of Ogre3D and the reason why we choose it:

- Simple, easy to use OO interface designed to minimize the effort required to render 3D scenes, and to be independent of 3D implementation i.e. Direct3D/OpenGL.
- Extensible example framework makes getting your application running is quick and simple
- Common requirements like render state management, spatial culling, dealing with transparency are done for you automatically saving you valuable time
- Clean, uncluttered design and full documentation of all engine classes
- Proven, stable engine used in several commercial products
- Sophisticated skeletal animation support
- Direct3D and OpenGL support
- Builds on Visual C++ and Code::Blocks on Windows
- Generic models could easily be embedded into
- Although it supports skeletal animation, it is a bit hard to animate the generic models that were taken from some other source

Actually Ogre3D is the best open source graphics engine of the market. The Object Oriented aspect is also one of the most important fact why we choose Ogre3D. It is easy to use and very powerful[5].

3.2.3 Artificial Intelligence

Because of we have lots of question marks about AI before searching, understanding definition of AI is at first precedence for us. “AI is the ability of a computer or other machine to perform those activities that are normally thought to require intelligence.” This definition comes from *The American Heritage Dictionary of the English Language, Fourth Edition* (Houghton Mifflin Company).

4
http://en.wikipedia.org/wiki/3d_graphics
http://en.wikipedia.org/wiki/OGRE
http://www.ogre3d.org/

5
http://www.ogre3d.org/index.php?option=com_content&task=view&id=13&Itemid=62
http://en.wikipedia.org/wiki/Game_engine
http://www.openscenegraph.org/
http://www.gamasutra.com/

There are basically two types of AI. Strong and weak AI. Strong AI is the ability to solve a problem requiring intelligence if it were to be solved by a human is not enough; AI must also learn and adapt to be considered intelligent. Unlike strong AI, *weak AI* involves a broader range of purposes and technologies to give machines specialized intelligent qualities. Game AI falls into the category of weak AI.

Think about deterministic AI, non deterministic AI is the opposite side of deterministic AI. Non deterministic AI is unpredictable and indefiniteness. Of course AI method determines the rank of indefiniteness. For example if a non-player character learning to adopt to the fighting tactics of a player could be learned by neural network, genetic algorithm or Bayesian network which are nondeterministic AI.

Game AI can be divided into two categories: deterministic and non deterministic AI. In deterministic AI there is no uncertainty. Deterministic behavior is specified and predictable. If we want to give an example of deterministic AI chasing algorithm can be relevant. If you give x and y coordinate to character, character is moving along toward to x and y coordinate axes until it coincide with this x and y coordinate.

Deterministic AI techniques are the easiest part of AI. They are fast, easy to implement, test, debug and predictable. Deterministic AI does not make easy learning and evolving. Also it is predictable after a little game play.

Nondeterministic AI is related about learning and unpredictability. Nondeterministic method has behavior that is revealed without explicit instructions and can learn and extrapolate on their own.

For our project cheating which is deterministic AI and Bayesian network can be suitable. An example of a cheating, in war simulation game the AI player get all information about its opponents like the types, number and location of units without sending scouts. But cheating can be sometime useless. If the player understands that computer is cheating, the player thinks that his efforts is vain and can lose interest in the game. Additionally, if the cheating is unbalanced, the player can never beat the computer. In order to not lose players interest from the game cheating must be balanced very well. In our project there are players and AI players, in a room lets think four players and an AI player. Starting the game some AI player do not know anything so they can not solve puzzle but after a time human players will solve the puzzle and AI player in that room cheat how human player solve the puzzle. After solving many puzzles AI player knows more thing by cheating, it will combine the information by cheating and give some suggestions in order to help player to solving puzzle. Of course if AI player able to give suggestion, cheating is not enough it must have learning capability and this learning ability will be reached by some algorithm like Bayesian network.

In Bayesian network AI player makes decision when the state of game is uncertain. An example of a Bayesian network, in fighting game we want to predict the next strike the player will throw. By this strike the AI player either defenses itself and also strike according to players strike type. Player has more than one strike type, for example three; punch, low kick and high kick. If we are going to save this three strike combinations we can calculate probability for that strike given the previous two strikes. This will able to know three strike combinations. In our project for example in a room human player is exit from the room simply by opening the door. AI player is cheat that human player solve the puzzle

by opening the room and it will know that opening something (window, tab, and computer) may solve the puzzle. This ability is loading to AI player by us and if a room has window AI player knows that opening a window may solve the puzzle. In another room human player can exit the room by different way for example by just bouncing over the chair. In third room let's think there is a window which is not closed, the AI player give suggestion to human player that "lets exit the room from window by bouncing". This example may be not wonderful but it shows clearly that AI player able to combines the different puzzles. If there are more than one probability to solve the puzzle, AI player choose only one solution by looking up which solution is more used by others room by using probability.[6]

3.2.4 Game Engine

Being an integrated collection of software components of any computer game, game engine is the most important part in designing a game. On the top of the subcomponents game, it functions as the integrator of all these parts. Although it is used having the modularity feature, in this context it will be referred as the first definition: the backbone/core of any computer game.

The current market is populated with many powerful commercial game engines such as Unreal and NetImmerse engines, whereas there exist some satisfactory free MMOG middle-wares too. Since it is not allowed to use a game engine for this project, the purpose in researching the game engines is solely to have a better understanding of game design&development and to be able to divide the tasks for distribution among the team. From the whole research, the main parts of our game engine are as follows:

- 3D graphics module
- Basic physics module
- Collision detection module
- I/O module
- Sound module
- AI module
- Network module
- Database module
- GUI module

Usually, more functionality such as scripting is included in the game engine, but regarding the scope of the project, those seem to be negligible.
MMOG engines differ from regular game engines such that they are based on the network code and database management. 'Massively' feature requires that a huge amount of data is handled with huge number of players at the same time. With bandwidth being very expensive, network traffic optimization is one of the most important parts.
The free engines in the market such as Multiverse, Crystal Space and Realmforge GDK provide a good idea; but we have particularly been able to find a comparative overview of many engines, commercial and free, which can be found at Appendix A.2**.**

6 "AI for Game Developers". Online. http://safari5.bvdep.com/05960055555/ch00.

3.3 Interviews

Because of we have no experience about game development we arrange a formal meeting with Veysi İşler who is working at Modsimmer and a teacher at METU Computer Engineering Department, Faruk Polat who is a professor in Middle East Technical University Computer Engineering Department and Caglar Ata who was graduated last year from our department and he was a member of Anka Yazılım project group which was about a 3D educational software tool.

Meeting with Faruk Polat

We had a meeting with Faruk Polat on 16.10.2006 at METU Computer Engineering Department in his office. The meeting was very useful. After we explained about what is our project and our project scenario he gave valuable suggestions about artificial intelligent techniques. He mentioned that artificial intelligent is a very difficult and huge concept. He warned us to have enough knowledge about artificial intelligent and use AI techniques carefully. Especially he emphasized that learning part is very difficult in artificial intelligent. He did not keep secret that our task is hard because of time limit and we have not experience about game development before but he stressed that if we work systematically we will succeed in that project.

Meeting with Çağlar Ata

We had a meeting with Çağlar Ata on 16.10.2006 at METU Computer Engineering Department. The meeting was very useful especially he shared his experiences about group work and which tools we will use. Because of he was a senior student last year he worked in senior project and he advised us to share the work equally. He emphasize that we are lucky because of our group is forming by four people. In that project he said that being five people is very difficult according to four people. He mentioned that our task is very difficult and do not become low-spirited. He advised us that use OGRE tool while developing game and search the web carefully in order to get valuable information.

Meeting with Veysi İşler

We had a meeting with Veysi İşler on 18.10.2006 at Modsimmer in METU Teknokent. The meeting was very valuable. He shared his valuable experiences about game development. He was very glad to game development project was assigned to by our teachers. He got information about what we know about MMOG game and what is our scenario. He suggested that make a good risk analysis and do not make a huge target, reach our aims by a small step. He thought that for MMOG development we have not enough time. He advised that use a top-down approach, and solve the problems when we reach but he attracted our attention that set up framework very carefully. After we explained our project he shared his experience with us. He suggested us to use Torque for an engine. He added that OGRE is a very popular engine in fact he used last year, he encounter a problem but he could not remember the details of it. Also he said that Delta3D is a good engine but not good as Torque. For graphics rendering he suggested us to use DirectX or OpenGL. For network he used Torque and suggested us to use Torque. And for competition he advised that modeling is very important for visualization.

4 Project Requirements

4.1 Functional Requirements

4.1.1 Overview

To understand the needs of the project, assess feasibility, prevent ambiguity, and reach a valid project, the most important actions to be taken are determination and engineering of the requirements. The most important issues we considered during requirement specifications are:

- To make requirements consistent with the objective of the project.
- To specify requirements at proper level of abstraction.
- To state bounded and unambiguous requirements.
- To prevent conflicting requirements.
- To state technically achievable requirements.

The activities that we have performed for determining the requirements of our project are as follows.

- **Identifying Actors**

  Actors represent external entities that interact with the system. Actors basically help in defining the boundaries of the system. Identifying actors helps the developer see all the perspectives of the system.

- **Identifying Scenario**

  Scenario is, formally, a narrative description of what people do and experience as they try to make use of computer systems and applications. In our game, we have developed our scenario according to the game concept and game style. Scenario will have many different uses during the software life cycle.

- **Identifying Use Cases**

  In order to identify use cases, first we find all the use cases in the scenario that specifies all possible instances of what occurs when a player encounters an event in the game. After defining the requirements, we grouped them logically and added the necessary details.

4.1.2 Menu Requirements

These are the requirements that are related to the menu that is displayed in the game. This part is divided into two sections; one is about the general properties of the menu and the other one is about the properties and usage of the items that can appear in this menu.

a) General Requirements

• This menu is displayed from two contexts: first one is the entrance of the game (Main Access Menu) and the latter one is anytime whenever requested by the player during the playing of the game (Paused Access Menu).
• Menu is composed of items which are included in the menu according to the context that the menu is displayed from.
• The player can also use predefined shortcuts from the keyboard in order to select the menu items.

b) Menu Items Requirements

• Profile Selection
  o This item is displayed at very beginning of the game even before Main Access Menu.
  o The player can select his/her profile which stores the settings and games of his/her player.
• View Profile
  o This item is included in both Main Access Menu and Paused Access Menu.
  o The player can view his profile by this.
  o The information about the player and Total Elapsed Time In The Cube is written there.
  o Total Elapsed Time In The Cube is an important feature since the game has a Hall Of Fame section where the players who are able to get out of the cube are listed according to their elapsed time.
• Enter The Cube
  o This item is included in only Main Access Menu.
  o The player enters the cube with this button.
  o The player starts in a random room in the cube whenever he\she enters the cube.
  o We will try to group the players in 4-5 people, so that the player who enters the game will always interact with someone in the room and wont play alone.
  o The player who died in the game will not be able to enter the game for 15 minutes, so it is useless to try to enter the cube in this time interval.
• Resume Game
  o This item is included in Paused Access Menu.
  o The player is returned to his/her current game and continues playing from the exact position where the game is paused. But the game is actually never paused. Other players will continue to play.
• Help
  o This item is included in both Main Access Menu and Paused Access Menu.
  o Help section is displayed which includes general information about game contents, playing of the game and game controls.
• Credits
  o This item is included in Main Access Menu.
  o Information about the game developer company and its members are displayed.
• Hall Of Fame
  o This item is included in both Main Access Menu and Paused Access.
  o It is the table of most valuable players who are able to get out of the cube. The list is

made according to the minimum elapsed time in the cube.

- Leave The Cube
    - o This item is included in both Main Access Menu and Paused Access Menu.
    - o The player leaves the cube to the operating system.
    - o The player can select this item at anytime he/she wants.
    - o But the player must not forget that this game does not have a save option, and when a player leaves the game, he\she will not be able to continue from its last position. He\she will start in a random room in the cube if he\she wants to enter the cube again.

4.1.3 Game Flow Requirements

These requirements cover the period that the player is in the game actively. It includes environment, main character abilities, interaction between player and game and overall game logic during the game.

a) Game Logic Requirements

- The game has a homogeneous characteristic.
- The is game is played inside the rooms of the cube, and all rooms are treated equally.
- The player can change his location from one room to another.
- Most of the rooms will have some tricks and traps.
- Every room has a different trick, which means a different solution path.
- The traps can be deadly, so beware you can be wounded or die.
- Not all the rooms have to have a trick or trap, some of them can only be gateways.
- The rooms are not stable, they are moving inside the cube.

b) Environment

- The cube and Rooms
    - o It is the world that the game takes place in.
    - o It consists of only rooms. When you get out of the room, you either get out of the cube (if you are that lucky) or you get into another room.
- Objects
    - o Movable objects can be carried by characters and can be put to inventory.
    - o Most of the objects will be stable and can only be used.
- Inventory items
    - o They are objects that can be used by characters.
    - o A player can use only one item at one time.
    - o They have specific functionalities like for a key to open a door.
- AI Players
    - o They have artificial intelligence and/or scripted behavior.
    - o They can walk.
    - o They can jump.
    - o They can turn.
    - o They can climb.

    - They can speak.
    - They can get wounded.
    - They can die.

c) Human Player

- It has movement abilities.
    - Walk
    - Run
    - Jump
    - Crouch
    - Turn
    - Climb
- It has other abilities.
    - Pick up inventory items
    - Use inventory items
    - Drop inventory items
    - Use objects in the environment
    - Speak, chat with other human or AI players.
    - Get wounded, every player will have a specific health.
    - Die

d) Player-Game Interaction Requirements

- Player uses mouse and keyboard to supply input to the game.
- Game uses monitor and sound devices to supply output to the player.
- During the game the players can talk to each other or with AI players. This talk can be private or public.
- Player manages the his character's inventory items like selecting or dropping one by a menu.
- Player views the game from the first person's perspective.
- Player can view the health and inventory status during the game.
- Player can escape to the menu during the game. The game is not paused meanwhile.

4.1.4 Operational and Structural Requirements

These requirements form the base for the game. These are not controlled directly by the player but they work in coordination with each other to satisfy the integrity of the game.

a) Game Engine

- It provides coordination between the subcomponents.
- It processes the game flow based on game data and user responses.
- It provides the network architecture needed for the game.

b) Graphics

- It renders the scene in 3D perspective.
- It maps the models with the textures.

c) Sound

- Game has soundtracks playing in background.
- Player has the ability to control sound options such volume level.
- Game can deliver instant sound effects opening of a door.

d) AI and Scripting

- It is used in game controlled character behavior.
- It is used to control game flow.

e) Game Data

- Room State Information
- Models
    - Players
    - Rooms in the Cube
    - Objects and inventory items
    - Textures
    - Images
- Sound
    - Sound effects
    - Soundtrack music
- Scripts

4.2 Software Requirements

4.2.1 Overview

From the view of software capabilities, the requirements for the project can be assessed as follows:

Consistency

The game state should be kept in a consistent manner at all levels. Although over a large network with lots of interactions going on, it is difficult to achieve full consistency; logical assumptions should be made and efficient algorithms should be utilized in order to reduce game-state inconsistencies.

Node-to-node interaction

In order to support the functional requirements that are a consequence of simply being a massively multi-player online game, the network architecture employed in the project should:

- allow the maximum interaction among the players,
- enable a fast connection with a high bandwidth between the nodes on the network: client machines, the server (or servers), etc.

Client Side Processing Speed

In addition to the node-to-node interaction speed, the software modules deployed over the client machine should be able to process incoming data as fast as possible and minimize internal latencies. The same case applies to when the client machine processes information, before it is distributed over the network.

Realistic graphics rendering

The functional requirements discussed previously and the massively multi-player online game market imposes the game developers of this project to come up with a highly realistic graphics engine. Although a good deal of work has been conducted by other game developers over the past and the project members can make use of such tools; customizing them based on the project requirements and integrating the 3Dmodels, animation templates imposes further restrictions.

Artificial Intelligence processing

Both for interactive purposes and for adding some taste into the game, the AI module should be capable of resolving pre-defined predicates obtained from the game environment.

Some of the major items listed above are discussed thoroughly in the following sections.

4.2.2 Game State Consistency

The game to be developed in this project will be supporting hundreds of online players. Not only that but the current MMOGs on the market can support thousands and even tens of thousands of users simultaneously. Evidently, one of the most challenging tasks in developing a massively multi-player online game is to make sure that every player – to a high degree – has the same information about the game state. By game state we include all the information that is necessary for each client machine to fully functionally resolve and make the game available to the player. Although it is possible to include more information than what is actually *necessary*, such redundancy might lead to a weaker performance. Therefore before implementing strategies for resolving game state inconsistencies, the game state should be defined.

> In order to have a high performance, the game state to be utilized in this project should be kept minimal. Redundancies should be eliminated. However, such redundancy elimination should not put further load on the client machines in resolving the game state. The trade-offs should be well assessed.

Based on our scenario and the probable network architecture we will use in this project, the game state consistency problem can be evaluated in the following levels of abstraction, listed in decreasing priority in terms of time-constraints:

- Consistency on the client application,
- Consistency within the sub-worlds,
- Complete consistency.

Consistency on the client application

The priority should be given to resolving consistency problems inside the client application. If the players of the game cannot have a consistent view of the world (or sub-world) they are in, then unfortunately the whole purpose of the MMOG is gone. Suppose that the clients continuously receive information about the current world state. The client application should be able to:

- execute events in the their correct order:

  Due to physical factors affecting the speed of transmission over the network, the client application will not be able to receive every "game state-change" information in their correct sequence. The client application should therefore keep a list of "state-change" information based on their desired execution time and when the list is filled up such that a logical action can be obtained, the events should be executed in the correct order.

- resolve conflicts when necessary:

  Conflicts may and will arise during processing events in the execution list. Let us assume for the moment that the client application receives a "state-change" message that indicates that Player-A has started walking in the "north" direction. The client continues receiving messages indicating the current position of Player-A in regular intervals. Suddenly player A stops moving and a different message indicating the stop action is sent. If the stop message comes to the client with a relatively high delay, then the client

will still process Player-A to be moving in the "north" direction. Such conflicts should be resolved to a greatest extent in the client side.

- in case of work overload, take priority parameters into consideration:
  If the work load on the client is extensively large such that it **knows** it cannot execute all of the events in the required time interval, then the events with the highest priority should be executed before all the others.

Consistency within the sub-worlds

For our project, the game-state for the "sub-world" is limited to the information that is sufficient and necessary for simulating all interactions within a single room of the cube. Please remember that the game consists of hundreds of rooms that make a whole cube. In every room there will be several players interacting with each other and with the objects in the environment. Whenever the game-state within the "sub-world" changes, every player (or equivalently client application) should be notified, otherwise consistency cannot be achieved. Considering time issues, this problem has a priority level very close to that of resolving consistency on the client application. Yet it still has a lower priority: If the data repository associated with a single room is not updated so fast, the game state will still be consistent. It is just that the game will proceed in a slower fashion.

Complete Consistency

Although processing the interactions within a single room is more essential than processing the whole game-state data, a central repository is required to keep the game-state for the whole world consistent. Considering our scenario, players – upon solving the puzzles in a single room – will be able to move from one room to the other. Furthermore, the absolute locations of the rooms inside the cube will change on regular intervals. As the Artificial Component of the game also requires processing of the whole game-state data (statistics collection, etc.) complete consistency should be achieved.

"A supervisor (server) needs to know everything about the map but **does not need to know everything that is happening on the map**."[7]

The above statement clearly states what the central data repository's role and how its interactions with other software components should be like. It is not so important that the change in the "sub-world" state data be reflected as soon as possible on the central data repository. Furthermore not every single detail needs to be stored globally.

Various approaches can be taken for implementing the aforementioned requirement. A database management system seems suitable, however, such decisions are left to the design phase of the project.

4.2.3 Node-to-node interaction

The network architecture and backbone will be responsible for implementing the node-to-node interactions. Although a node-to-node interaction will be mainly between a client and a server, client-

7 Fiedler, Stefan. Wallner, Michael. Weber, Michael. "A Communication Architecture for Massive Multiplayer Games."

to-client interactions will be allowed for certain occasions such as implementing the chat functionality. In either case the network backbone should provide the maximum number of interactions possible.

Based on our research findings, the network architecture for a massively multi-player online should be such that:

- a high bandwidth is provided,
- latency is reduced to the minimum,
- security (authentication / authorization) issues are covered.

Keeping the bandwidth as high as possible and reducing the latency is our primary software requirement goal. Such requirements are critical in a game where interactive responsiveness is a major concern. Although security issues should be dealt with, due to overall project constraints, they will be addressed with a lower priority.

As discussed in the Technical Research section of this document, several models exist for implementing the network communications backbone. Taking the aforementioned requirements into consideration and evaluating the resources at hand, we have come to the conclusion that the "publisher-subscriber" model suits our project the best. The client-server architecture on its own seems incapable of supporting hundreds of online players. Since inherently the game is divided into so-called "sub-worlds" and the players do not really need to know about what is going on in other "sub-worlds", the channel approach discussed in the publisher-subscriber model seems more suitable. Although, the current trend is towards p2p architectures in MMOGs and a good amount of research is conducted on utilizing p2p models in the multi-player online games; due to consistency problems, we will avoid p2p models, at least for the core of the network backbone.

4.2.4 Client Side Processing Speed

One of the key steps in implementing a good massively multi-player online game is increasing interactive responsiveness, as discussed throughout the report. It is true that choosing a suitable network model and determining a solid representation of the game state are important steps taken towards reaching this goal. However, *reducing processing time on the client side is as important as the others*.

Suppose that consistency of the game state is preserved and that a strong network architecture is built for exchanging messages. Now each client application receives messages relevant to the channel they are subscribed to. For our scenario, every client will subscribe to the channel representing the room inside the cube. If a player inside the room turns on the lights for instance, this information must be made visible to every other client inside the same room. With the above consistency and network architecture conditions, the clients will receive the messages on time. However, there is no point at all if the clients cannot process them in a reasonable amount of time. In conclusion, consistency and timing considerations go together for such applications.

The same requirement holds for the reverse case: when the clients process user interactions (usually provided by some I/O devices) and send them to the server. However, for the reverse case it creates

additional problems. The desired (and of course expected) behaviour is that the clients process the user input and send them out with after a very short latency. If this were not the case, keeping the game-state consistent within the sub-world would also become a challenging task leading.

4.3 Hardware Requirements

P4 class processor or equivalent
64 MB of graphics card with OpenGL support
256 MB of memory
Some free disk space for installation
A monitor supporting at least 800*600 screen resolution
Sound card
Internet connection
Keyboard
Mouse

4.4 Non-Functional Requirements

Playability

In the game reviews and evaluations, the game is assessed generally in terms of four categories: game play, atmosphere, video and sound, while game play is considered as the most important feature of a game. According to this fact, the playability of the game is a major concern and requirement. The other features of the game are to be designed accordingly, with the ultimate aim to increase playability, thus player satisfaction. Regarding the tendency towards graphics ignoring game play in the current market further drives us to emphasize playability because we see it as a key concept in competing with high-budget MMOG's.

Portability

Since the current game market is overwhelmingly dominated by Windows platform, it is nearly obligatory to develop on that platform. Porting the game onto other platforms do not seem visible, therefore the project is planned to be compatible with only Windows platform.

Reliability

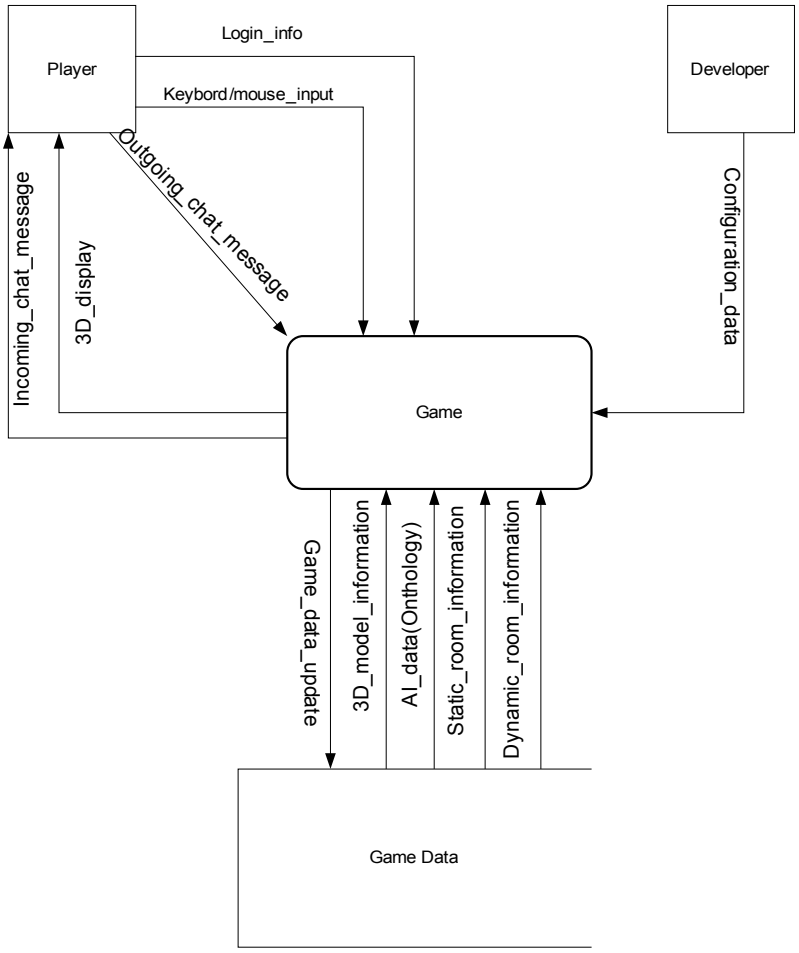
Being massively multiplayer, the game has highly distributed data operations, and for that reason a strong control of consistency and synchronization is to be achieved. This stands to be a very important topic in system robustness and thus user satisfaction.

5 System Analysis

5.1 Data Flow Diagrams

5.1.1 Level:0 DFD

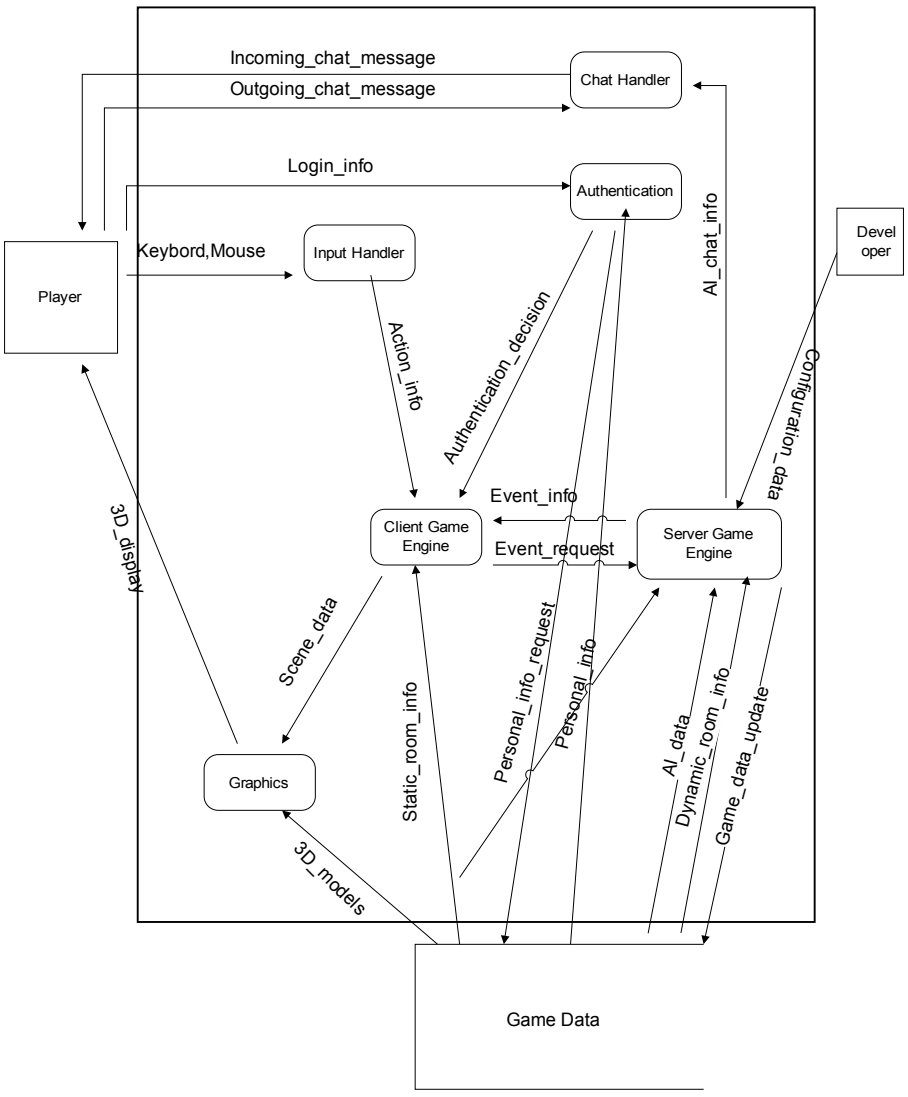
Level 0 DFD is the system in the highest level. The whole software is referred as ‘Game’ process with which player and developer interacts with. All the necessary game data is stored in the game data repository. Player interaction involves keyboard/mouse and chat message inputs, and display and chat message outputs. Game continuously queries 3D model information, AI data, static/dynamic room information and updates the necessary game data.



Level 0

5.1.2 Level:1 DFD “Game”

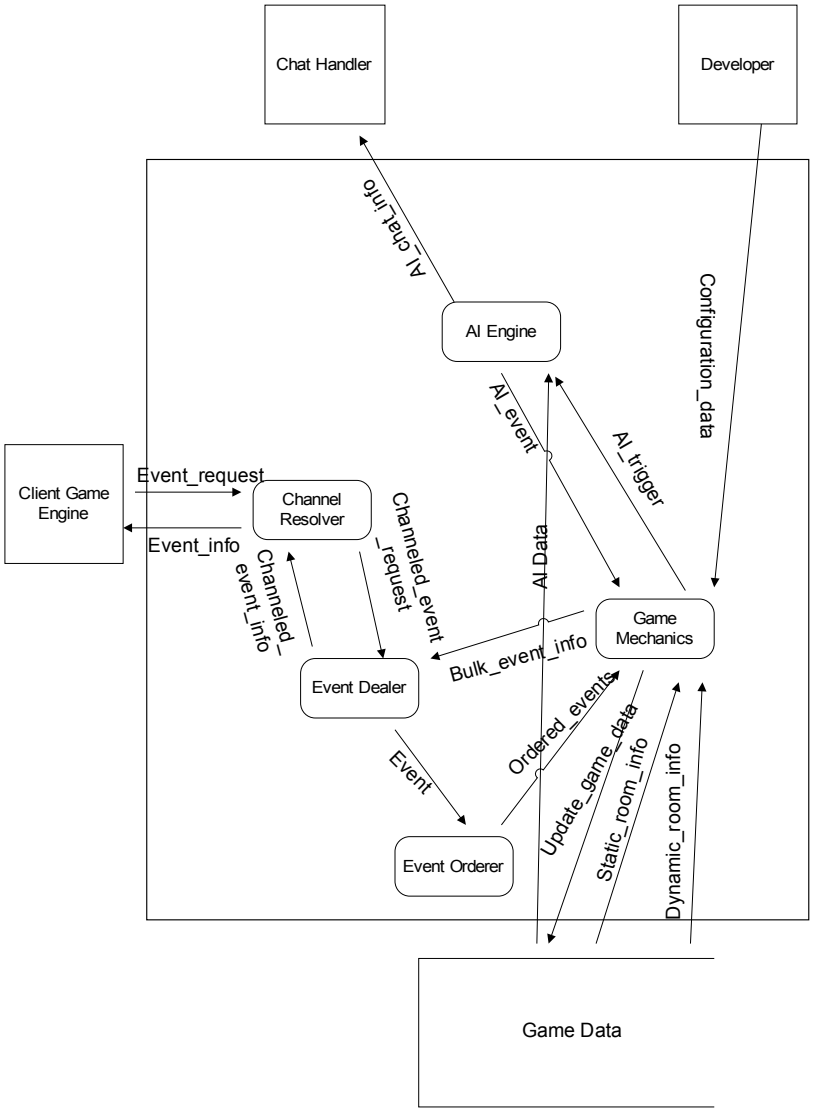
Level 1 DFD indicates the main functions of the game. Player provides the game with login info, which is authenticated and matched with the corresponding personal info of the player. This information is then passed to the Client Game Engine. Input Handler is the layer where player inputs are processed and converted into action information that is evaluated in Client Game Engine, and with the necessary room data from the repository, is combined to scene data. This is combined by Graphics module with 3D model data to construct the 3D scene to be displayed to the player. The actions performed by the player is transformed into events and sent to Server Game Engine by Client Game Engine, which also receives the incoming events from the Server Game Engine. Server Game Engine is the central part where events from clients are processed and distributed, highly using the Game Data Repository as well as the configuration data from Developer. There exists a further process of Chat Handler, which handles the chat message traffic among the human and AI players.



Level1 Game

5.1.3 Level:2 DFD “Server Game Engine”

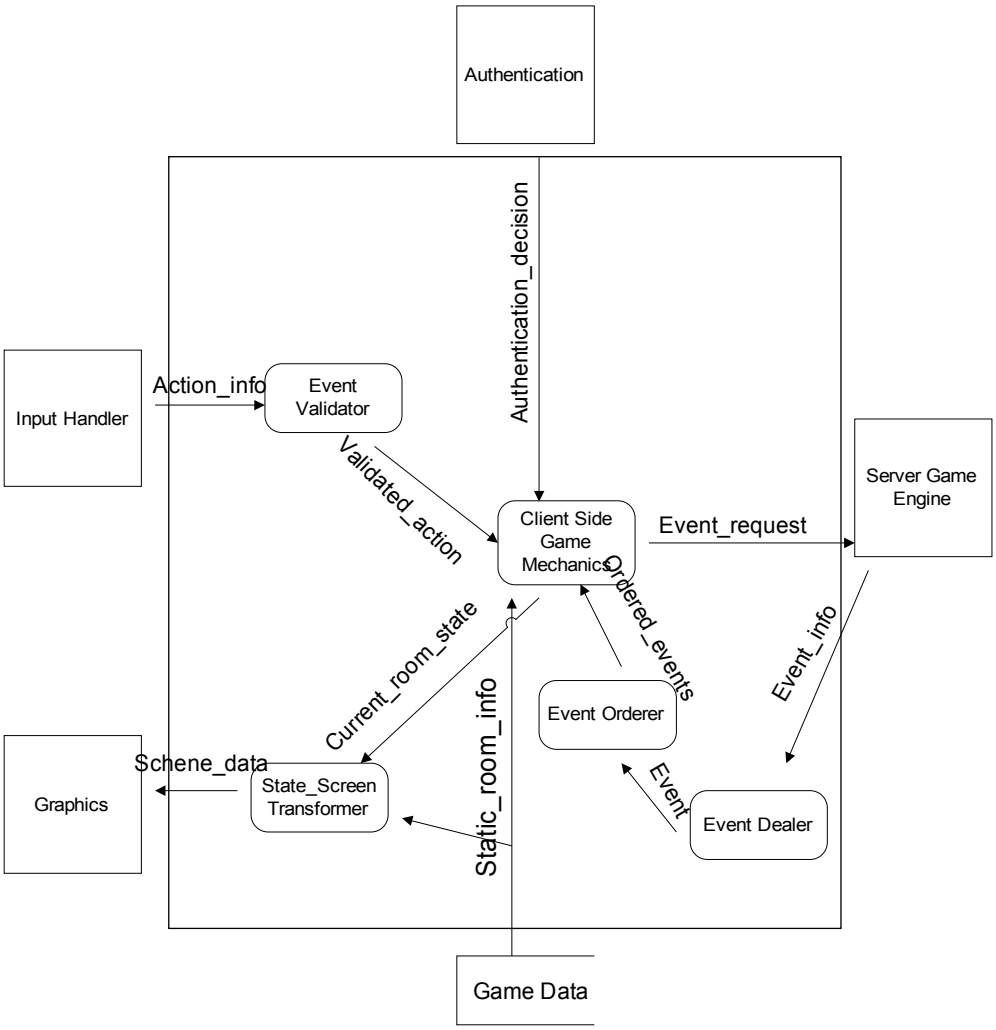
This L2 DFD introduces a detailed view of Server Game Engine, which is the core process of the game on the server side. First of all, Channel Resolver process is responsible for handling the incoming event request traffic by addressing the request to the relevant channels, which may be different for each client. Event Dealer is the part where event information is converted to actual events and sent to Event Orderer, where the events are ordered with respect to time and address constraints. Those ordered events, with the game data and room information from the repository, are processed in Game Mechanics module. The resultant event information is sent back to Event Dealer, and is eventually sent to the relevant clients. Game Mechanics process also interacts with AI Engine, sending and receiving event data while AI Engine is capable of sending messages to Chat Handler. Finally, configuration data from Developer comes directly to Game Mechanics and processed.



Level 2 Server Game Engine

5.1.4 Level:2 DFD “Client Game Engine”

This L2 DFD shows the inner dynamics of Client Game Engine, which is the core process of the game on the client side. Initially, authentication decision is sent to Client Game Mechanics. From then on, action info from the Input Handler is validated and sent to Client Game Mechanics. Being changed, current room state is sent to State Screen Transformer, combined with static room info from the repository and sent to Graphics. Events are handled similar to those in server side: event requests are made by Client Game Mechanics to Server Game Engine, and the incoming event information is received in Event Dealer. After being converter to events, they are sent to Event Orderer and ordered. Game Mechanics receives and processes the ordered events.



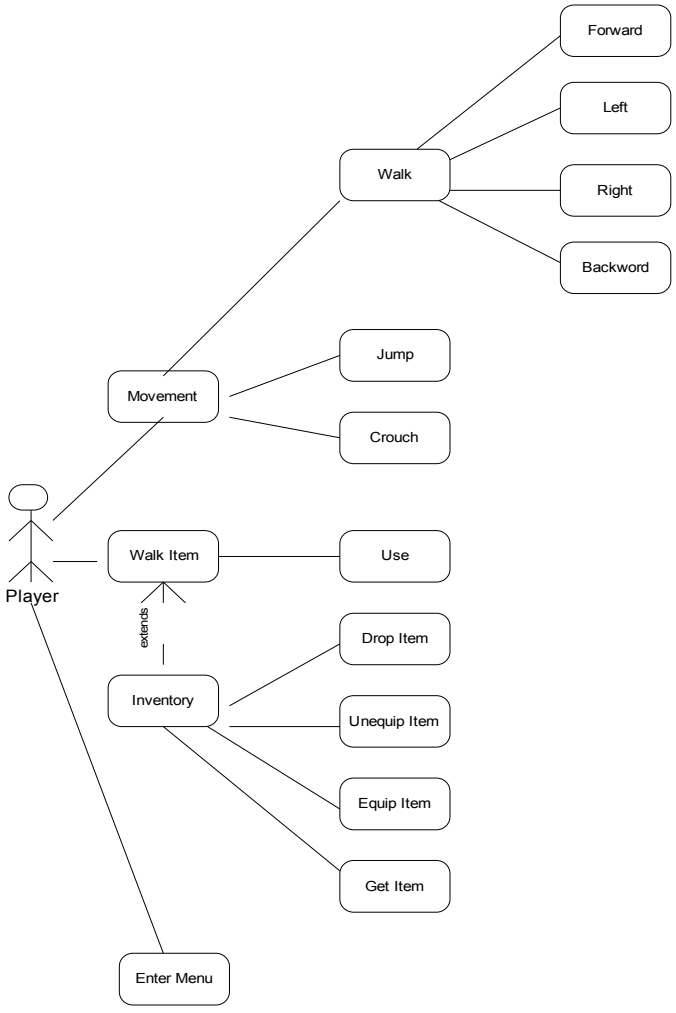
Level 2 Client Game Engine

5.2 Use Case Diagrams

The use case diagrams provided below go in direct correspondence with each of the functional requirements item discussed in the Project Requirements section of the report. The reader is encouraged to revisit that section, if necessary.

5.2.1 Game Play Use Case

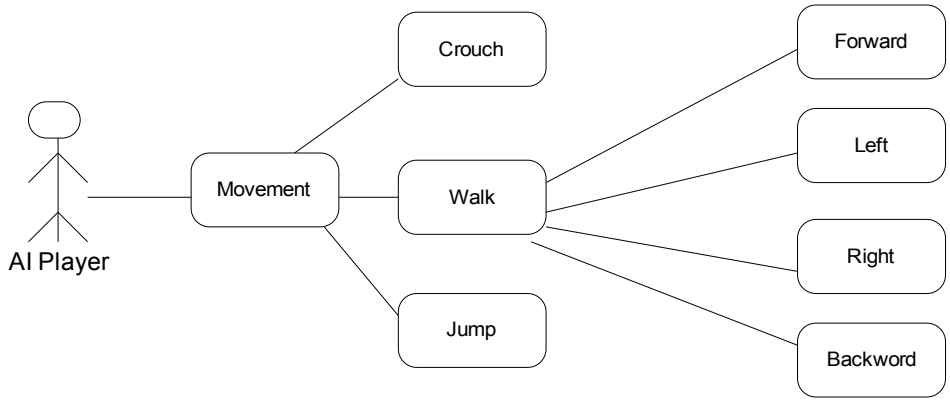
This use case displays the actions that the player can do in game play without interaction with other players. We grouped these actions into four: Movement, Camera, Items and Menu Entrance. Player can walk, jump or crouch in the game. Those movements can be executed forward, backward, left or right. Player can change the camera view in the game. Items is an important issue in the game. There are two types of items in the game: Wall Items are non-moveable items and can only be used, Inventory Items are moveable. An inventory item can be get, dropped, equipped or unequipped. Only one item can be equipped in a time and only the equipped item can be used. The last action is the Menu entrance in the game, player can enter the menu whenever he wants.



Gameplay Usecase

5.2.2 AI Player Game Playing Use Case

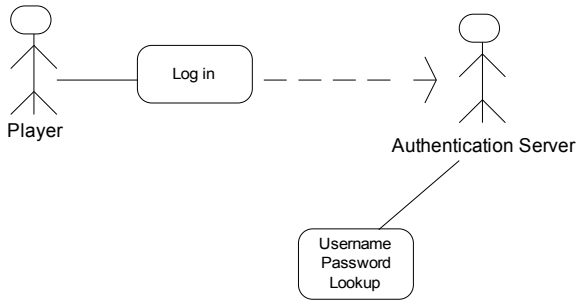
This use case displays the actions that AI player can do in the game. AI player in our game is a bit restricted, it can only have movement actions which are walk, jump or crouch. Of course the directions are forward, backward, left and right.



Gameplay Usecase(AI Player)

5.2.3 Login Use Case

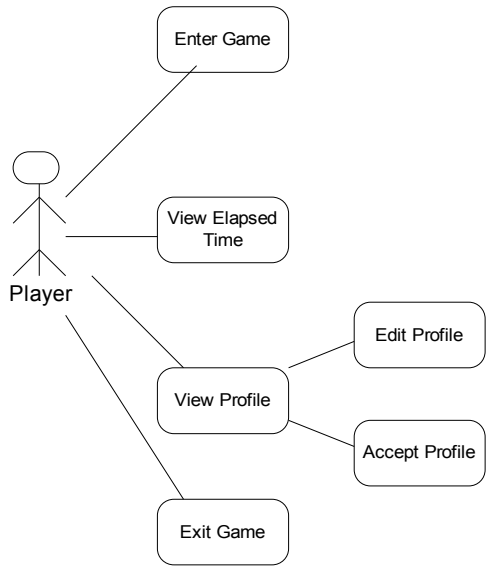
This use case displays the player's login to the game. Authentication server checks player's username and password and according to the check it either allows or disallows player's entrance to the game.



Login Usecase

5.2.4 Menu Interface Use Case

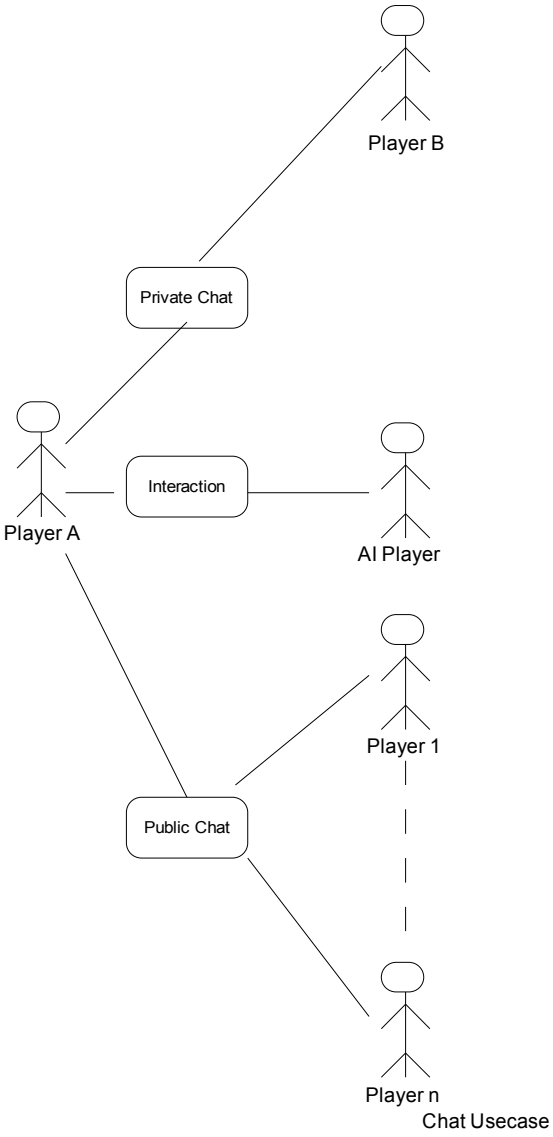
This use case displays the things that player can do in the Main Access Menu. He can view his profile, edit and accept it. He can also view his elapsed time in the cube. Other functionalities are entering and leaving the game.



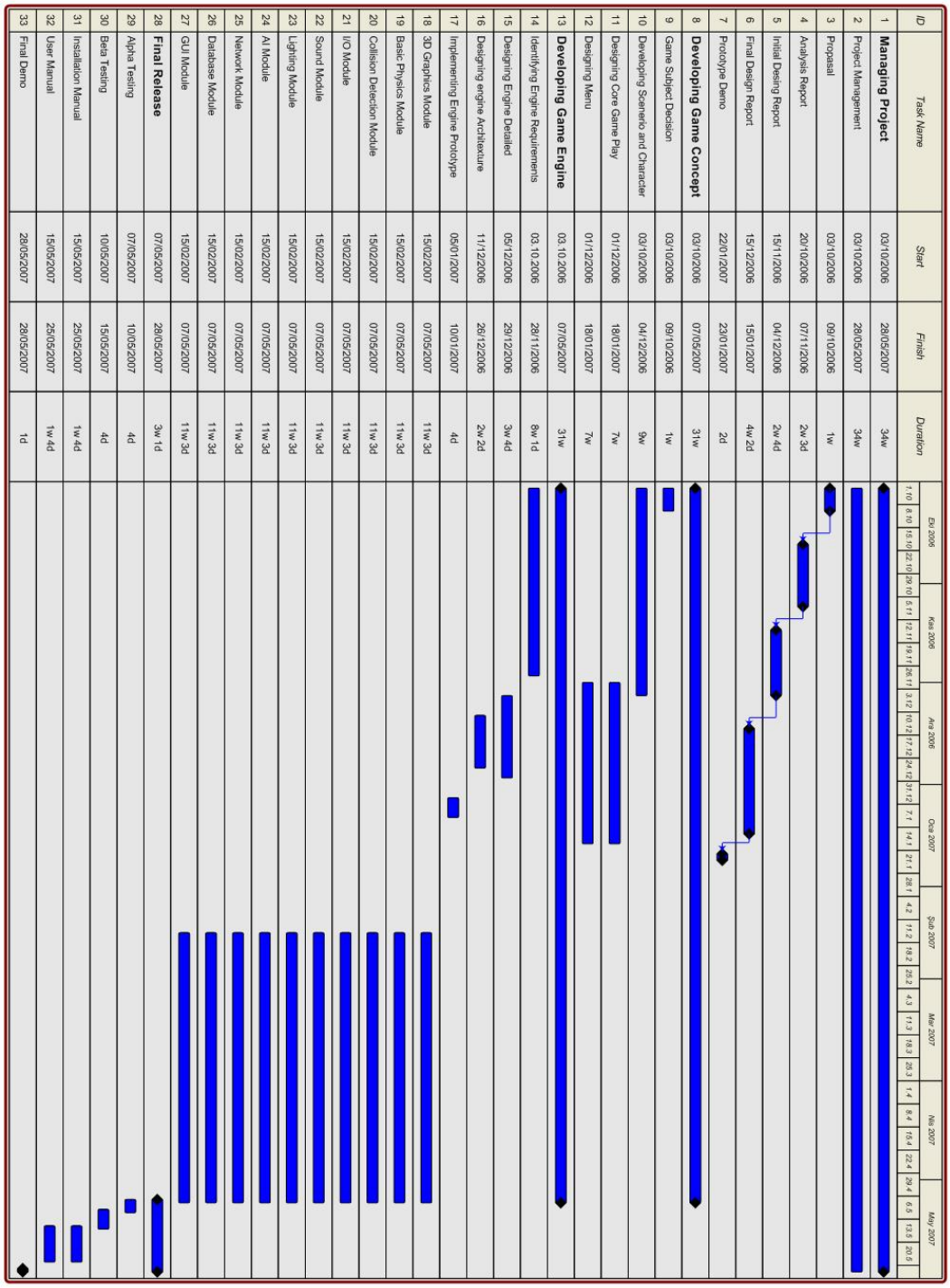
Menu Interface Usecase

5.2.5 Chat Use Case

This use case displays the chat usage in the game. Player can chat with other players in the game. This chat can be public, which includes all players in a room, or can be private, which can only be done by two players. Player can also interact with AI players in the game, this interaction is of course limited.



6 Project Schedule



7 Risk Management

Risk management is essential to be able to deal with strict deadlines. And since the project team have hardly any experience in game design and development, the risk management plan addresses a very important topic determining the success or failure of the project. The possible risks and the corresponding RMMM is explained in details, as a means of reducing the probability of failure.

7.1 Project Risks

Staff Size and Experience(ST):

- Given that the project relies upon individual effort, inexperienced staff, which is likely since the team consists of senior students with no domain expertise, results in inefficiency and extra training.
- Given that staff performance is very important in the quality of the project, degrading staff performance in time deteriorates the overall quality of the system.
- Given that team size is small, where no substitution or alternatives are easily tolerable, any staff turnover greatly jeopardizes the project's success, even though not leads to the termination of the project.

Customer Characteristics(CC):

- Given that the project has to have a good playability besides the technical attributes, end-users' dissatisfaction from the system by demanding enhancements of the other properties which are out of project scope forces reassessment of project objectives and scope, thus disrupting project layout.

Product Parameters(PP):

- Given that the strict deadlines, underestimated project size will increase total project completion time, thus making it impossible to meet the deadlines.
- Given that components are imported to satisfy the functionality of a module, any incapability of a component leads to either extension of the component(if possible), or substituting the component with another, in any case resulting in extra non-scheduled effort.
- Given that many components (e.g. engines) are integrated to realize the project, a possible disintegrative stemming from any component may lead to an unexpected gap in game design, thus introduce the reassessment of the design with a new component.
- Given that the 'massively multiplayer' feature is demanding on the network connection, and

'3D' feature on CPU& Video Card, a wrong estimation of hardware and network requirements end up destroying the practicality of the game with unrealistic/unfeasible requirements.
- Given that the task distribution is based on major roles, the discovery of too many minor roles that have been previously ignored causes an extra burden on the team members.

Development Process(DP):

- Given that a desired level of software quality is to be met, the deficiency of software quality assurance results of degrading quality of the product.
- Given that the process relies on the validity of the modules as well as the integrity of all, the absence of a detailed, systematic bottom-up testing approach makes it impossible to recover from the defects revealed too late.

7.2 Risk Table

Risks	Category	Probability	Impact	Risk ID
Inexperienced staff	ST	60%	3	R1
Degrading staff performance	ST	50%	3	R2
Staff turnover	ST	10%	1	R3
Customer dissatisfaction	CC	10%	3	R4
Underestimated project size	PP	30%	1	R5
Deficient components	PP	20%	3	R6
Lacking component integration	PP	40%	2	R7
Wrong estimation of hardware requirements	PP	20%	2	R8
Underestimated minor roles	PP	50%	3	R9
Lacking SQA approach	DP	30%	3	R10
Lacking systematic testing	DP	20%	3	R11

Where impact values are:

1. catastrophic
2. critical
3. marginal
4. negligible

7.3 Overview of RMMM

From the risk table, it can be seen that major problems arise in staff- and product-related parts. That is

natural due to the fact that the staff is small-sized, inexperienced and occupied part-time with the project, while the project is large, unfamiliar and to some extent chaotic in nature(where evolutionary design should be employed). Anyway, the general approach to risks is as follows:

- encouraging horizontal communication within groups
- keeping high motivation
- realistic (not idealistic) anticipation of the scope and purpose
- maintaining a good level discipline and professionalism

The detailed Risk Information Sheets related to RMMM can be found in Appendix A.1.

8 Software Quality Plan

Objective of quality assurance is to ensure that the product does not deviate far from the original design specifications. In case of deviations, future deviations will be prevented and previous deviations will be corrected. During the check for the deviations, 'Requirements Analysis' document will be the main criterion. At each stage of our work, it will be checked whether each task or subtasks are completed with respect to the original design.

Since we are developing a Massively Multiplayer Online 3D Game, total number of lines of code in the project will be in huge amount. Keeping track of such an amount will be very difficult. To make the code understandable and to ease debugging in case of error, a coding standard document will be prepared. Every group member will have to write their codes according to the code standard determined.

The project will have too many tasks, most of them will depend on each other and will most probably be assigned to different team members. As a result of this, in case of suggestion of an important change, it would take place only if every team member agrees on it. Also in case of efficiency decrease, we are planning to keep ready one member to help the main member, who is dealing with a task(graphics, network etc.).
The project will have many deliverables. For each deliverable specific properties will be checked. Such that

For the Game play:

- Does the game meet our network specifications? Can our architecture support the massively multiplayer online aspect?
- Does the game meet our 3d graphics objectives? Is it eye-pleasing?
- Does the game meet our playability constraints?
- Are the puzzles in the game prepared well? Are they too simple or too difficult to solve?
- Does the AI in the game meet our AI constraints?
- Are the interactions in the game enough?

For the source code of the project:

- Is the source code easy to read?
- Is the source code commented well?
- Is the source code highly modular?

The RMMM(Risk Mitigation Monitoring Management) will be used to prevent, monitor, and manage the risks. During the implementation, each member will be expected to do white box testing. At the end of the implementation of the project, alpha and beta testing will be made. Accordingly test cases will be prepared.

Appendix

A.1 Risk Information Sheets

Risk Information Sheet 1		
Risk ID : R1	Prob : 60%	Impact : 3
Description : Inexperienced staff		
Mitigation/Monitoring : 1. Do thorough pre-evaluation of staff to ensure capability 2. Establish frequent informal meetings to inquire about each member's knowledge about their tasks 3. Collect a detailed, technical assessment papers from staff as a means of self-evaluation		
Management : Increase horizontal communication among team members to increase information sharing, thus compromising total team inexperience. Train staff where possible to increase their capability.		

Risk Information Sheet 2		
Risk ID : R2	Prob : 50%	Impact : 3
Description : Degrading Staff Performance		
Mitigation/Monitoring : 1. Allocate initially balanced load to the staff to eliminate later disappointment and overburdening 2. Run regular performance tests to the staff to determine their performance level		
Management : Employ the sense of flexibility and keep motivation high to recover degrading performance.		

Risk Information Sheet 3		
Risk ID : R3	Prob : 10%	Impact : 1
Description : Staff turnover		
Mitigation/Monitoring : 1. Conduct initial interviews with the team members to guarantee their determination 2. Conduct informal discussion where any tendencies to turn over are uncovered		
Management : Increase and emphasize the roles of the members in coordination of the project and decision, thus increase satisfaction and responsibility.		

Risk Information Sheet 4		
Risk ID : R4	Prob : 10%	Impact : 3
Description : Customer dissatisfaction		
Mitigation/Monitoring : 1. Conduct detailed field research 2. Perform small-scaled polls to learn about the perspective of end-users		
Management : Inform the end-users well about the project, eliminating any conflicts due to misunderstanding. Then bend the project scope (at tolerant level, and where possible) due to user needs.		

Risk Information Sheet 5		
Risk ID : R5	Prob : 30%	Impact : 1
Description : Underestimated project size		
Mitigation/Monitoring :		
Management :		

Risk Information Sheet 6		
Risk ID : R6	Prob : 20%	Impact : 3
Description : Deficient components		
Mitigation/Monitoring : 1. Compare to similar projects in order to obtain a stronger estimation of size 2. Monitor schedule frequently and check its consistency with current situation		
Management : Check schedule goals and employ extra workload to keep up the milestones. Compromise the extra time due to any overestimated tasks with other tasks.		

Risk Information Sheet 7		
Risk ID : R7	Prob : 40%	Impact : 2
Description : Lacking component integration		
Mitigation/Monitoring : 1. Perform a detailed analysis of the components at the beginning. 2. Run small integration tests where basic functionalities of components are attached.		
Management : Retrieve comparative information about required components and be ready to have alternatives to be substituted for the inappropriate component.		

Risk Information Sheet 8		
Risk ID : R8	Prob : 20%	Impact : 2

Description : Wrong estimation of hardware requirements
Mitigation/Monitoring : 1. Use historical/statistical data of other projects to verify better estimation 2. Run performance tests to determine system requirements
Management : Apply necessary modifications throughout the project to compromise any components which have been tested to have high requirements. Reevaluate system features to accomplish the goal of low-requirement, adjust when needed.

Risk Information Sheet 9		
Risk ID : R9	Prob : 50%	Impact : 3
Description : Underestimated minor roles		
Mitigation/Monitoring : 1. Categorize and divide the roles considering the subparts of the modules. 2. Get workforce feedback to check if any major task spawns too many minor tasks.		
Management : Evaluate the workforce needed for each minor task and adjust the distribution of tasks among the team.		

Risk Information Sheet 10		
Risk ID : R10	Prob : 30%	Impact : 3
Description : Lacking SQA approach		
Mitigation/Monitoring : 1. Adopt a determined approach from the beginning of the project, and allocate enough workforce to SQA. 2. Get quality information through technical reviews, hold the statistical data.		
Management : Increase the amount of testing before release. Employ automated tests to improve quality.		

Risk Information Sheet 11		
Risk ID : R11	Prob : 20%	Impact : 3
Description : Lacking systematic testing		
Mitigation/Monitoring : 1. Adopt a determined approach from the beginning of the project, and allocate enough workforce to testing.		
Management : Employ a more wide range of testing, i.e. weigh beta testing with a sufficient group of players.		

A.2

Overview over engines:

Game Engines	Dark Basic Pro	Quake 1	Unreal	Halflife	Genesis	Nebula	Quake 2	Game Studio	Quake 3	Lichtech 2	Vulpine	Torque	Crystal Space	Power Render
Culling system	BSP	BSP	BSP	BSP	BSP	-	BSP	BSP	BSP	Portal	Portal	**BSP**	BSP	Yes
Mipmap	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	Yes	Yes
LOD	-	-	Discr.	-	Discr.	Discr.		Discr.	Discr.	Cont.	Cont.	**Discr.**	-	Cont.
Environment map	Cubic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	Yes	Yes
Lightmaps	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	**Yes**	Yes	Yes
Dynamic shadows	Yes	-	Yes	-	Yes	-	-	Yes	-	Yes	Yes	**Yes**	Yes	Yes
Mesh interpolation	-	-	Yes	-	-	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	-	Yes
Terrain	Yes	-	Yes	-	-	Yes	-	Yes	-	Yes	Yes	**Deform.**	Yes	Deform.
Particle system	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	-	Yes
Mirrors	Yes	-	Yes	-	Yes	-	Yes	Dev	Yes	Yes	Yes	**Dev**	Yes	Yes
Curved surfaces	-	-	Yes	-	-	-	-	-	Yes	Yes	Yes	-	Yes	Yes
Shaders	Yes	-	-	-	-	Yes	-	-	Yes	Yes	Yes	**Dev**	-	Yes
Bone animation	Yes	-	Yes	Yes	Yes	Yes	-	-	-	Yes	Yes	**Yes**	-	Yes
Multiplayer	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**	-	Yes
Multisession	-	-	Yes	-	-	-	Yes	Dev	Yes	Yes	Yes	**Yes**	-	Yes
Pysics Engine	-	-	-	-	-	-	-	-	-	Yes	Yes	**Dev**	-	?
Scripting language	-	Basic	C	Basic	C	C		TLC	C++	Java	C++	**C/pyth.**	phyton	C++
Price	$100	GPL	$10,000	?	$10,000		$10,000	$80	$250,000	$75,000		**$100**	GNU	$5,500

Data collected from[8]

8
http://www.garagegames.com/uploaded/GameEngines2.pdf

References

"AI for Game Developers". Online. http://safari5.bvdep.com/05960055555/ch00.

"AI Game Development: Synthetic Creatures with Learning and Reactive Behaviours". Online. http://safari5.bvdep.com/1592730043/ch01lev1sec2#X2ludGVybmFsX1NlY3Rpb25Db250ZW50P3htbGlkPTE1OTI3MzAwNDMvY2gwMWxldjFzZWMz.

"Multiverse Platform : An Overview". http://www.multiverse.net/platform/whitepapers/mv_overview.pdf

"The Game AI Page. Building Artificial Intelligence into Games". Online. http://www.gameai.com.

Amir, Gideon. Axelrod, Ramon. "Massively Multiplayer Game Development: Architecture and Techniques for an MMORTS." Online. http://www.gamasutra.com/features/20050613/amir_pfv.htm.

Bethke, Eric "Game Development and Production", 2003, Wordware Publishing, Inc.

Caltagirone, Sergio. Keys, Matthew. Schlief, Bryan. Willshire, Mary Jane. "Architecture for a Massively Multiplayer Online Role Playing Game Engine."

Fiedler, Stefan. Wallner, Michael. Weber, Michael. "A Communication Architecture for Massive Multiplayer Games."

Hammersley, Tom. "Viewing Systems for 3D Engines." Online. http://www.devmaster.net/articles/viewing-systems/

Hannah, Britt L. "Object-Oriented Game Design". Online. http://www.devmaster.net/articles/building-mmorpg/.

http://en.wikipedia.org/wiki/Game_engine

Massiv Project Website. http://massiv.objectweb.org/

Merabti, Majid. El Rhalibi, Abdennour. “Peer-to-Peer Architecture and Protocol for a Massively Multiplayer Online Game.”

Müller, Jens. Metzen, Jan Hendrik. Ploss, Alexander. Schellman, Maraike. Gorlatch, Sergei. “Rokkatan: Scaling an RTS Game Design to the Massively Multiplayer Realm.”

Privantu, Radu. “A Beginner's Guide to Creating a MMORPG.” Online. http://www.devmaster.net/articles/building-mmorpg/.

Rouse III, Richard. “Game Design Theory and Practice”, 2001, Wordware Publishing, Inc.

Simpson, Jake. “Game Engine Anatomy”. Online. http://www.extremetech.com/article2/0,3973,594,00.asp

Stang, Bendik. “Game Engines, Features and Possibilities”. http://www.garagegames.com/uploaded/GameEngines2.pdf

Yu, Peiqun. “Mopar: A Mobile Overlay Peer-to-Peer Architecture for Scalable Massively Multiplayer Online Games.”