BACKGROUND

Since the dark ages of game development, users interests evolved video games into a real application frame for computer science. Many fields of computer science especially networking, artificial intelligence and geometrical 3D graphics, use game business as a test area for validating the success of new techniques. Nowadays interactive games introduce the cheapest and absolutely the most attractive tool for killing time. Game developers work on a large basis to fulfill preferences of various user ranges. In the market one can find games of all styles, starting from classical racing, fighting and arcade games, to the new powerful-menu multiplayer strategically-formed scenarios.

The significant improvements in the data communication technologies in the last decade have led to a new form of social structure, MMOGs (Massively Multiplayer Online Games). These games not so surprisingly captured the throne of previously dominant games, non-massively multiplayer online games. The reason is apparent; the more player controlled characters are online at the same time in the same virtual world, the more realistic a game becomes. That did not take long for the leading companies of the gaming industry to realize the potential of MMOGs.

Beside this approach, there exist plenty of experts in the field seeing the game industry as a future basis for many purpose entertainment applications, including virtual reality and specific meta-physical simulations. In the light of the above evidences we think game development will pay big dividends to the rest of our careers.
DEFINITION

MMOGs (Massively Multiplayer Online Games) are surely the new trend in computer entertainment industry. A great potential source of income for the companies having a significant amount of share in the computer entertainment market.

The lack and deficiency of the resources about MMOGs is the target problem. That is surely a hinderence preventing the independant developers and small companies to express their original ideas regarding MMOGs.

In the project that CORE5 is going to carry out, the main goal of the project is to present a partially open source MMORPG (Massively Multiplayer Online Role Playing Game) featuring an original story-line with the latest technical improvements in the computer entertainment industry, and direct light to the paths of those without enough resources.

GOALS AND SCOPE

During the development phase our basic goals are:

Develop a well-structured object oriented design and implementation
Provide a flexible multi purpose 3d graphics implementation
Create a user-friendly, large range audience game

For the basic research phase on the field we focused on the following aspects:

Specific data structures that best fit client-server paradigms
Common methodologies on related software engineering tips and tricks
Best network interaction techniques, as a primary scope
Embedded physical engine packages and their interfaces
Possible ideas on artificial intelligence implementations: machine learning
The previously mentioned ideas will contribute to the our main scope, which is basically defined as:

- Developing of a full documented design process
- Analysis of previous networking games
- Analysis on artificial intelligence methods, including neural networks
- Design of a modern full-featured graphics application

**Problem Analysis**

After analyzing popular MMO games, the following propositions have been reached:

Server-client model is decided to be adopted for the network structure, that in the end led to the following structure of the game.

The game is decided to be divided into two parts:

- Web Software
- Client Software
- Server Software

The Client Software can be divided into these major components:

- Graphics Engine
- Physics Engine
- Sound Engine
- Network Engine
- Core Game Engine

The Server Software can be divided into these major components:

- AI Engine
- Physics Engine
- Network Engine
- Core Game Engine
On the Client Side:

*Graphics Engine* takes care of the need of the 3d representation of the game. Only exists on the client-side since there is no need for a 3d representation on the server. All the calculations are performed on the client.

*Physics Engine* normally can be on either the server or the client. But restricting Physics calculations only to server or client becomes problematic.

**Case 1:**

Physics Engine only on the client. Because the client is updated with the properties of the physical entities frequently, at first it may seem to work without problem. Also because the physical calculation performed on the client is restricted to a small region, a complex physical model can be used to simulate object-object interactions. But with a detailed analysis it can be easily seen that this model makes the server critically dependant on the clients to gather physical properties of the entities which in turn prevents the correct projection of the physical world onto a time frame. Informally this situation can be expressed as “the client does whatever it wants in terms of physical behaviour”. This can lead to even worse cases that any illegally modified client software can make devastating effects on the game. These deficiencies renders this approach inappropriate for a MMOG.

**Case 2:**

Physics Engine only on the server: Server carries out the physics calculations and updates the clients appropriately. There are some old MMOGs that uses this model but the main problem is that if the latency of the network is directly reflected on the client, and makes the user feel the latency directly. This situation has strong negative effects on the user.
As stated in case 1 and case 2, it is inappropriate to use Physics Engine on one side. A better approach solves the problems apparent in the 2 cases mostly. Physics Engine on both the client and the server works as follows: Client calculates physical entities and if latency reaches a certain threshold it uses its own calculations, otherwise uses the calculations performed by the server. Server does not accept the calculated values. Physical calculations performed on the client render the latencies on the network nearly transparent to the user up to a certain threshold. Despite this model, long latencies are still disappointingly perceptible. When this model is applied to situations where the interaction is between a moving and a non-moving object, the best results are gathered. Interactions between two movable objects can cause not realistic results since physical calculations are applied on spatial properties which are defined by time. Since no interaction between two movable objects is intended to be handled, that situation is not predicted to cause a major problem. Despite all these justifications, the proposed model is not perfect since the result of the calculations performed on the server and the client can differ in perceptible amounts even if no modifications are made on the client the server is aware of. The main cause of this is the difference in the platforms the calculations carried out. Floating point precision differences can be given as an example to make the situation more concrete. But since the project is intended for only one platform, the problem is of no concern of the project.

**Sound Engine** is planned to take care of the necessity of providing position dependent sound calculation. As in real life the distance from the sound source make the sound to be perceived differently. And the position of the sound source with respect to the target plays an important role in simulating the 3d behaviour of the world. This feature is used in most 3d games, not restricted to online games. Sound Engine resides only on the client side since there is no need for a sound solution on the server.

**Network Engine** on the client is responsible for sending requests to the server and getting updates about the world. Despite this brief explanation, most important part of the
Core Game Engine glues all the other components to work in harmony and carries out content related rule calculations. Some examples may help to visualize the issue about content related rules, filtering the skills or spells the user character have ,allowing the user to perform an action an object and not allowing on other object. The Core Game Engine acts as a bridge role between other components. The Server Side Core Game Engine is different than this one, the client side one.

On the Server Side:

AI Engine is responsible for simulating the behavioral existence of the non-player characters. The main reasons that this engine is not included also in the client software like the physics engine is the possibility that non-deterministic algorithms that may be used in this engine which may cause different results different times.

Physics Engine takes place both in server and client as stated in client-side Physics Engine.

Network Engine on the server software sends the updated info to the clients when the client needs. If no visible change on the client takes place, no update is made. That approach is to conserve bandwidth, but in worst case, when every client has visible change, all the clients are informed. This feature is dependent on the information provided by the Core Game Engine in that it provides existence of the visible change on the client. Another responsibility of the Network Engine is to get the requests the clients have sent and send them to the Core Game Engine to further processing.

Core Game Engine on the server side is like the client side Core Game Engine, but it differs in that this engine performs some calculations that are not handled by the client side look-alike. This engine performs all the inter-character calculations, all the battle simulations are performed in this engine. The bridge feature of this engine is different from that of the client side Core Game Engine in that this engine communicates with AI Engine, Physics Engine and Network Engine, not Graphics or Sound Engines.
FUNCTIONAL REQUIREMENTS

Here the functional requirements are presented:

- Players can change their profile using the web services
- Players can register and create accounts using web services
- A menu with enough functionality providing the user character creation, selection, configuration change, login and logout features exists to carry out these purposes
- Players can choose their character among 3 races
- All the player attributes, containing world location and personal stats, are saved upon user logoff or a world-save that is performed periodically
- Different parts of the terrain have different effects on the player stats.
- Terrain system allows realistic landscapes: Mountains, lakes, dungeons
- Trees and different plant elements exist
- Various collectible items exist, some collected flora in time
- Players can walk
- Players can strafe
- Players can sneak
- Players can run
- Players can get tired
- Players can attack with weapons
- Players can attack with free hands
- Players can die
- Players can cast spell
- Players respawn in a predefined zone when they die
- Players can be teleported to other zones
- Players can trade items with other players and non-player characters
- A simple economical model exists to control the flow of currency in the game
- Players can be resurrected when they die
- Players can pick-pocket other players
• Players can pick some locks
• Players make different types of sound as a result of their actions.
• Players can not be looted by other players when killed.
• Players stats can be affected by the items carried
• Players stats can be affected by the items equipped
• Players can consume some items to take permanent or temporary effects.
• Players stores a limited number of objects
• Player combines items to form new items
• Player has extendable list of skills
• Player skills can be static or dynamic
• Game rules support level-ups
• Players can respond upon death
• Death has penalty on player characteristics' values
• Players can lose hit points
• Players can lose mana points
• Players can regenerate mana points
• Players can recover hit points
• Non Player Characters can walk
• Non Player Characters can attack
• Non Player Characters can feed
• Non Player Characters can die
• Non Player Characters can get tired
• Non Player Characters can cast spells
• Non Player Characters can use and consume items
• Non Player Characters can respawn
• Non Player Characters can run
• Non Player Characters can fly
• Non Player Characters have inventory that can be looted by the killer
• Players can form party
• Players can chat with other players
- Players can attack other player
- Players can not loot other players
- Players can take quests to complete
- Quests have level-limitations
- Quests have race-limitations
- Players can send messages to the admin
- Admin can kick players
- Admin can send message to players
- Admin can transport player to predefined locations
- Admin can chat
- Admin can ban player
- Admin can monitor player
- Admin can view world

NON-FUNCTIONAL REQUIREMENTS

**Usability**

The user should be able to control the game without constraint. Since visual and audible aids are important for the player to comprehend the game, these aids should be attractive and understandable. Menus should be designed as clear as possible in order to prevent user lost in menus. Game flow and game scenario should be smooth in order to make the game adaptable.

**Reliability**

Breakdowns should be removed. The game is reliable as far as it is error free so almost all the bugs should be debugged. Since multithreading will be used, all the possible deadlocks should be determined and then prevented. Network connection should not be dropped unexpectedly. Uncontrolled cheating will be prevented in order to provide justice.
**Security:**
The network security should be accomplished faultless.
In order to ensure the money transfer safely, it is necessary to contact with professional and reliable corporations.

**Supportability:**
The game will be up to change and the updates will be distributed via internet.
Production defects will be compensated.
Performance:
In order to decrease network latencies several optimizations will be carried out.

**Software Requirements:**
Windows XP operating system

**Hardware Requirements:**
The main target PC configurations the project is to be developed for is as follows:
Programmable GPU.
High speed internet connection.
512 MB of RAM.
Pentium 4 class CPU or better.
DFD 0

Here the data flow diagram level 0 is presented:
ADMIN USE CASE

- Kick Player
- Communicate
- Transport Player
- Ban Player
- Monitor
- View world
RESEARCH

In this section research part of the analysis is presented, the main aim of the research was to find what is valuable in the current computer online gaming industry and to find out what makes a project more valuable when the final consumer is considered.

The following chart shows the overall distribution of massively multiplayer online games according to their genre:

According to the research it is obvious that massively multiplayer online fantasy RPG games lead the game market. The other type of online multiplayer games holds a small part of the market share in their hands. This leads the gaming industry to an ever-growing competition and making the success rate of the projects drop. But with a well-designed strategy successful results can be obtained. This research result is a reason to select fantasy RPG game for project.
The following chart demonstrates the fact that when satisfying 3d graphics, smooth game-play due to well-structured networking software design and clever marketing meet trust-worthy support, success is unavoidable. World of Warcraft has all these qualities.
Time Chart: