Thunder Tactics

Technical Report

Supervisor
Lector Dr. Adrian Iftene

Students
Paul NECHIFOR
Tiberiu PASAT

Index

- Problem representation
- State of the Art
  - Idea
  - MMORPGs
  - TBSs
  - Tools and Resources
- Solution
  - Server
  - Client
- Results, Evaluation
- Comparison with other solutions
- Future work
- Conclusions
- Bibliography

Problem representation

We plan to create a strategy game that will combine two different types of games: an MMORPG game and a turn-based strategy game.

Players will be able to navigate trough a map with a hero where can find shops (to buy different things to help him in fights), safe places (where cannot be attacked, e.g. a town), mobs (are monsters in game that players can attack to level up).

Players can fight with other players.

This is a MMORPG game, users will have to register, and will fight with other players only when they are logged in game.

Also is a turn based game so players will be moved into an arena when the fight starts and each one will have a turn (this is where strategy appears).

Each player has a hero, that can be leveled up by defeating mobs. With each level the hero grows stronger.

After the hero wins a battle it wins experience and money. With money the player can buy items that will give extra power or can buy other creatures (as army) that will fight with him.

This creature can't level up, but can have different abilities. Some of them fight with sword, others have bows or spells.

The fight ends when a hero dies and the other one is declared as winner.

Actors and objectives

Player: have fun playing with multiple players and create strategies alone or in teams to defeat their enemies.
Client: to be able to provide an attractive graphical interface (3D). To be usable without install requirement. Can be accessed from multiple operating systems.
Provide to player possibility to enter credentials to be sent to server in order to be registered or logged in.
Server: to provide functionalities for client as: register, login, logout, being able to control his character in game world, in order to have fun, socialize and fight with others.
Administrator: can moderate illegalities in game.

State of the Art

Idea

We plan to create a strategy game that will combine two different types of game: a massively multiplayer online role-playing game (MMORPG) and a turn-based strategy game (TBS).

Since it is a multiplayer game, the users will have to register with a server. The game is browser-based which means that the user can play from anywhere without needing to install anything. The only requirement is a modern browser which supports WebGL.

Each player has a hero, that can be leveled up by defeating mobs or other players. With each level, the hero grows stronger. In free mode, the player is only represented by his hero (not all his troops) and can navigate the game world in real-time. This is where he can encounter shops, other players, mobs and others.

At shops, they can hire troops or buy items that will help them in fights. Shops are usually in safe places, these are zones where players cannot fight (e.g. a town, or the original spawn site).

Players in free mode. Screenshot of users chating in Metin2.

Mobs are non-player characters (NPCs) which means they are controlled by the computer and

respawn after being defeated. They can be made up of hostile troops, which attack the player upon sight, or neutral troops, which will only fight if they are attacked. Mobs are of varying levels and a player can choose fight weak mobs in order to advance his level more easily.

In non-safe places, players can choose to engage other players in fights or they can be attacked themselves without warning.

Since it is also a TBS, when a fight starts players will be moved to a grid based arena. Here, all of the player's troops will be visible and the players take turns to attack each other (this is where strategy appears).

The fight ends when a hero dies and the other one is declared as the winner.

Example of arena mode battle. Screenshot from Heroes 3, a turn-based game.

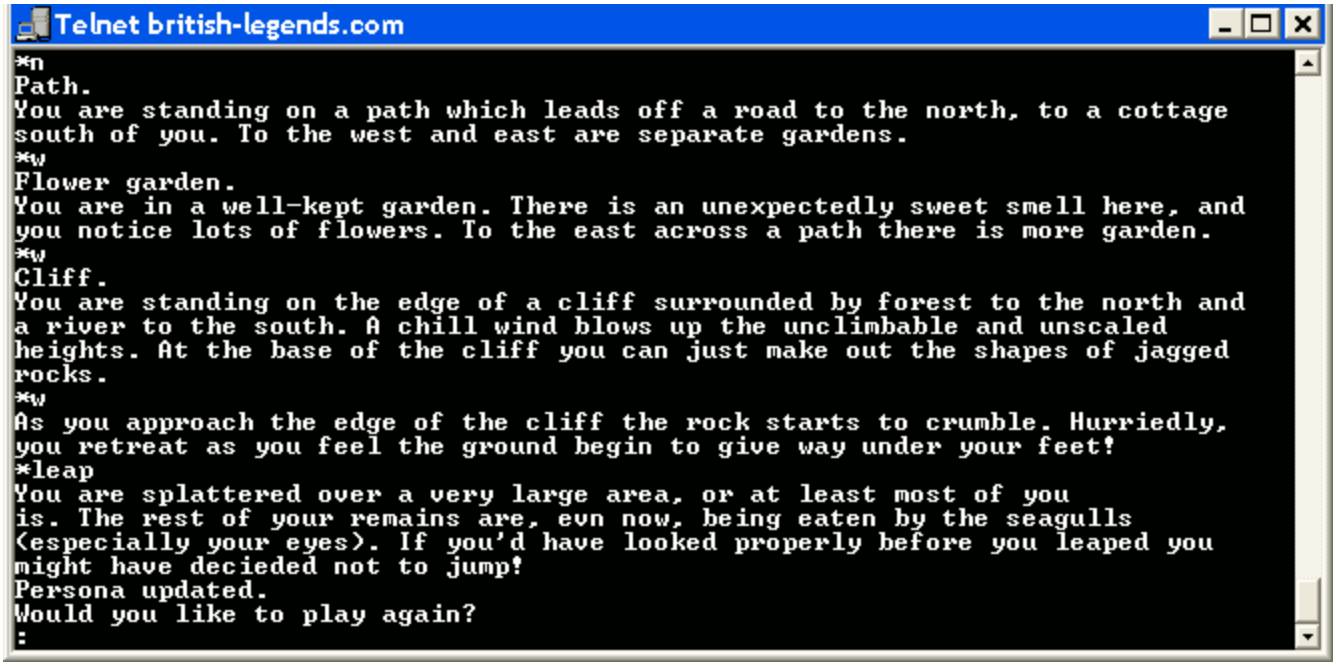
After a player wins a battle, he gains experience and can collect the defeated player's gold and items. With the gold, the player can buy items that will give him extra power or he can hire other creatures to enter his army and fight for him. If he doesn't have enough gold, he can sell some items or troops.

Troops can't level up, but they can have different abilities. Some of them fight with swords (melee units), others have bows or cast spells from afar (ranged units).

MMORPGs

The term MMORPG was coined by Richard Garriott, the creator of Ultima Online, in 1997. Previous to this and related coinages, these games were generally called graphical MUDs.

MMORPGs can be seen to have roots in the earliest multi-user games such as Mazewar (1974) and MUD1 (1978).



Screenshot of MUD1 played online with Telnet on British Legends.

Most popular MMORPG games these days are World of Warcraft, Lineage II, Mu Online, Metin2, and Diablo III.

World of Warcraft

As with other MMORPGs, players control a character avatar within a game world in third- or first-person view, exploring the landscape, fighting various monsters, completing quests, and interacting with non-player characters or other players.

Screenshot with a fight in World of Warcraft.

World of Warcraft requires the player to pay for a subscription, either by buying prepaid game cards for a selected amount of playing time, or by using a credit or debit card to pay on a regular basis.

Lineage II

To begin playing Lineage II, players create a character as their avatar in the game's medieval-style virtual world. All the races start off at Talking Island. Players can choose from either fighter or mystic professions at the start, except for Dwarves and Kamael which are only able to select the fighter profession; this choice acts as an archetype for later profession options.

Each race has its own set of classes, even if humans, elves and dark elves have a lot of classes that are very similar to their counterparts in the other two races.

As players kill non-player character monsters, they accumulate experience points and skill points (SP). As experience points accumulate, the character's level increases, meaning various attributes of the character are augmented. Players purchase and then upgrade their character's skills using SP. Players can play alone or as part of a group to fight monsters and complete quests for new skills, experience points, and items. Player versus player (PvP) is a significant portion of the game. The game provides many social, political, and economic aspects which are developed through the community and by the actions, in-game, of single players.

On November 30, 2011 Lineage II adopted a free-to-play model in Lineage II: Goddess of Destruction, with all game content being free save for "purchasable in-game store items and packs".

PvP fight in Lineage II game.

Mu Online

MU Online was created in December 2001 by the Korean gaming company Webzen. Like in most MMORPGs, players have to create a character among five different classes and to set their foot on the MU Continent. In order to gain experience and thus to level up, a players needs to fight monsters (mobs). MU is populated by a large variety of monsters, from simple ones like goblins and golems, to frightening ones such as the Gorgon, Kundun or Selupan. Each monster-type is unique, has different spawn points, and drops different items.

Every character has four basic statistics: Strength, Agility, Vitality, and Energy. When a player gains enough experience points to level up they can distribute 5 points freely among these stats. Magic Gladiator and Dark Lord characters gain 7 points instead of 5. When a character reaches level 220 they may take on a special quest so that every level up will get them 6 points instead of 5. In addition, for every level beyond 220 they will receive 1 point, so there is no waste. After doing this quest, at level 380 a player can start a new quest to get their character getting 7 points each level instead of 6.

- **Strength** - This is a character's physical strength. The more invested in this stat, the more damage dealt when striking with a melee weapon and the less chance of *missing* an enemy. Higher strength allows a character to equip stronger armor and weapons.
- **Agility** - This measures a character's dexterity. Agility increases a character's hitting accuracy, attack speed, defense and evade rate (known in the game as defense rate). While Strength determines melee damage, Agility determines damage from a Bow or Crossbow. Some weapons and equipment require large numbers of Agility points, and every piece of Elf armor requires some.
- **Vitality** - This is a character's toughness. Vitality gives a character extra HP so that it can survive more damaging hits.
- **Energy** - This is a character's capability to use skills and spells. Dark Wizards and Energy Elves need large amounts of Energy to learn new spells, while Dark Knights mainly require it to use their weapon skills more often. Investing points in Energy increases maximum mana, mana recharging rate and "AG" as well as increasing the damage that a skill does.
- **Command** – This is a special stat that only Dark Lords have. It measures the ability a Dark Lord has to handle the pets of nature. By increasing command, a Dark Lord may learn new spells (apart from Fireburst, which only requires energy, and Force, the spell given to new DL's). A Dark Lord that has raised a lot of command can also hold the maximum capacity of 80 guild members if he's a Guildmaster.

Fight scene from MuOnline game.

TBSs

The first turn-based game was Empire which was released in 1977. It was created by Walter Bright and was inspired by board games. The most popular turn-based games of all time are the Heroes of Might and Magic series. The first was released in 1995 and the latest in 2011.

In TBS games players fight by commanding their units (which represent multiple soldiers) on a turn by turn basis. These place more emphasis on thought as opposed to fast action like in real-time strategy games.

TBS games usually feature a unit representing the hero and his army which can move in real time on a map. There are various objects (like buildings, bridges etc.) with which the hero can interact and there are various hostile armies navigating the map as well. When two armies get too close a turn-based fight is started.

The players have multiple types of units which they position at the beginning of a fight.

The hero has a leadership attribute which determines how many soldiers will fight for him, an amount of gold for buying troops, spells, armor and others.

Heroes of Might and Magic series

The Heroes series is within the genre of turn-based strategy. The titular heroes are player characters who can recruit armies, move around the map, capture resources, and engage in combat. The heroes also incorporate some role-playing game elements; they possess a set of statistics that confer bonuses to an army, artifacts that enhance their powers, and knowledge of magical spells that can be used to attack enemies or produce strategic benefits. Also, heroes gain experience levels from battle, such that veteran heroes are significantly more powerful than inexperienced ones. Experienced heroes may persist through a campaign, but generally do not carry over between scenarios.

Whenever a player engages in battle, the game changes from the adventure map display to a combat screen, which is based on either a hexagonal or square grid. In this mode, the game mimics the turn-based tactics genre, as the engaged armies must carry through the battle without the opportunity to reinforce or gracefully retreat. With few exceptions, combat must end with the losing army deserting, being destroyed, or paying a heavy price in gold to surrender. Surrendering allows the player to keep the remaining units intact.

Creatures in an army are represented by unit stacks, each of which consists of a single type of creature, in any quantity. A limited number of stacks are available to each army, varying by game. Players generally maneuver their stacks attempting to achieve the most favorable rate of attrition for themselves. The games also have an automatic combat option that allows the computer to make tactical choices for a player. Heroes participate in battle as well: passively by granting bonuses to their army, and actively by engaging in combat and casting spells. In most of the games, heroes do not act as units, and cannot be harmed. However, in Heroes IV they do act as regular units and can be "killed"; these dead heroes are transferred to the nearest town's dungeon where they can be freed if their team captures the town.

Combat is affected by several random factors. In addition to simulating dice rolls to determine damage, a variety of influences including hero abilities and special bonuses determine a unit's luck and morale ratings, which affect the likelihood of those units triggering a bonus during combat. A unit that triggers good luck deals more (or receives less) damage, and a unit that triggers high morale receives an extra turn. In some other games, luck and morale can also be negative, with opposite corresponding effects. Luck and morale can be improved by hero abilities, artifacts, and spells. Morale may suffer with overwhelming odds in combat or by mixing incompatible unit types (e.g. Chaos with Order.)

Screenshots from different versions of the game:

Heroes 2 fight.

Heroes 3 fight.

Heroes 5 battle screenshot

King's Bounty series

The first King's Bounty game was released in 1990 and is considered to be the inspiration for the Heroes of Might and Magic series. In it, the player is given the task of finding a hidden artifact by the dying king.

The player can choose a hero from four classes: barbarian, knight, paladin, or sorceress. He has to locate the artifact before the king dies by finding the 25 pieces of a map scattered through the four continents.

He does so by exploring the world with his army, which he has to pay on a weekly basis. He will encounter wandering armies and can siege castles and will be rewarded upon defeating them. Each region has specific creatures available for hiring and each unit type has a disposition towards other types and this increases or decreases the morale of the army which affects it's strength.

King's Bounty screeshot.

In 2007 the series was restarted with King's Bounty: The Legend which has had two expansions. The new games feature a hexagonal grid. Besides the usual attributes of the hero (like leadership which controls his maximum army size, mana which controls the spells he can cast), a new feature is the Chest of Rage which includes four spirits of rage, each with their special ability.

Example of a fight in King's Bounty: The Legend showing the hexagonal grid.

Sieging a castle with an army of dragons in King's Bounty: Armored Princess.

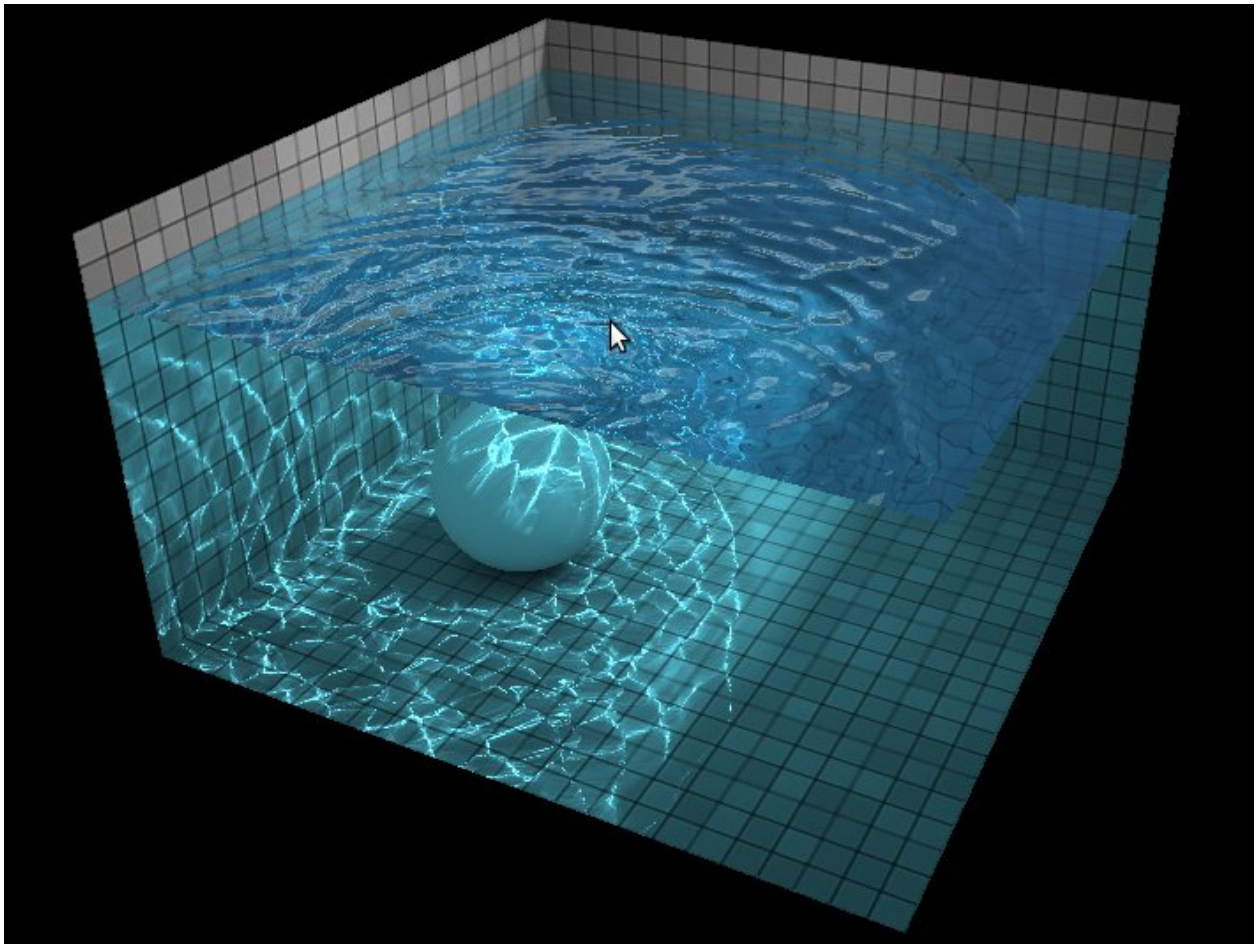
Tools and Resources

WebGL

The idea to put 3D in the browser was first realized in 1994 with Virtual Reality Markup Language, but it did not gain widespread use do to multiple factors: slow internet speeds at the time, limited programmability, few computers with dedicated graphics hardware and others.

In 2009, Khronos Group with Apple, Google, Mozilla and others started working on WebGL and in 2011 the first version was released. WebGL is based on OpenGL for Embedded Systems and is now supported by all major browsers (except IE, of course). WebGL is a standardized, cross-platform, plugin-less (except for IE, of course) and versatile way of working with 3D withing a browser.

As a new technology, there are so far few tools and working applications but there are many demos showcasing it's potential. The following water simulation demonstrats reflections, refractions, ambient occlusion, heightfield water simulation, soft shadows, and caustics all being done in real-time in the browser.



WebGL Water simulation by Evan Wallace.

Three.js

Three.js is a lightweight cross-browser JavaScript library/API used to create and display animated 3D computer graphics on a Web browser. Three.js scripts may be used in conjunction with the HTML5 canvas element, SVG or WebGL. The source code is hosted in a repository on GitHub.

L3DT - Large Terrain Generator

L3DT is a Windows application for generating terrain maps and textures. It is intended to help game developers and digital artists create vast high-quality 3D worlds.

L3DT does quite a lot. A brief list of features would include:

- **Generating and editing height maps** (Designable height fields using a high-level 'design map' in which you can specify land altitudes, roughness, lakes, climate, etc.)
- **3D display** (Real-time 3D rendering and editing of maps in the Sapphire plug in, as well as support for Terragen, VTP Enviro, and other popular renderers.)

- **Calculations and optimisations** (An automated calculation pipeline that allows you to queue-up some or all of your calculations, set the relevant settings, and then walk away while L3DT does its thing. No program baby-sitting is required)

WebSockets

The WebSocket specification—developed as part of the HTML5 initiative—introduced the WebSocket JavaScript interface, which defines a full-duplex single socket connection over which messages can be sent between client and server.

The WebSocket standard simplifies much of the complexity around bi-directional web communication and connection management.

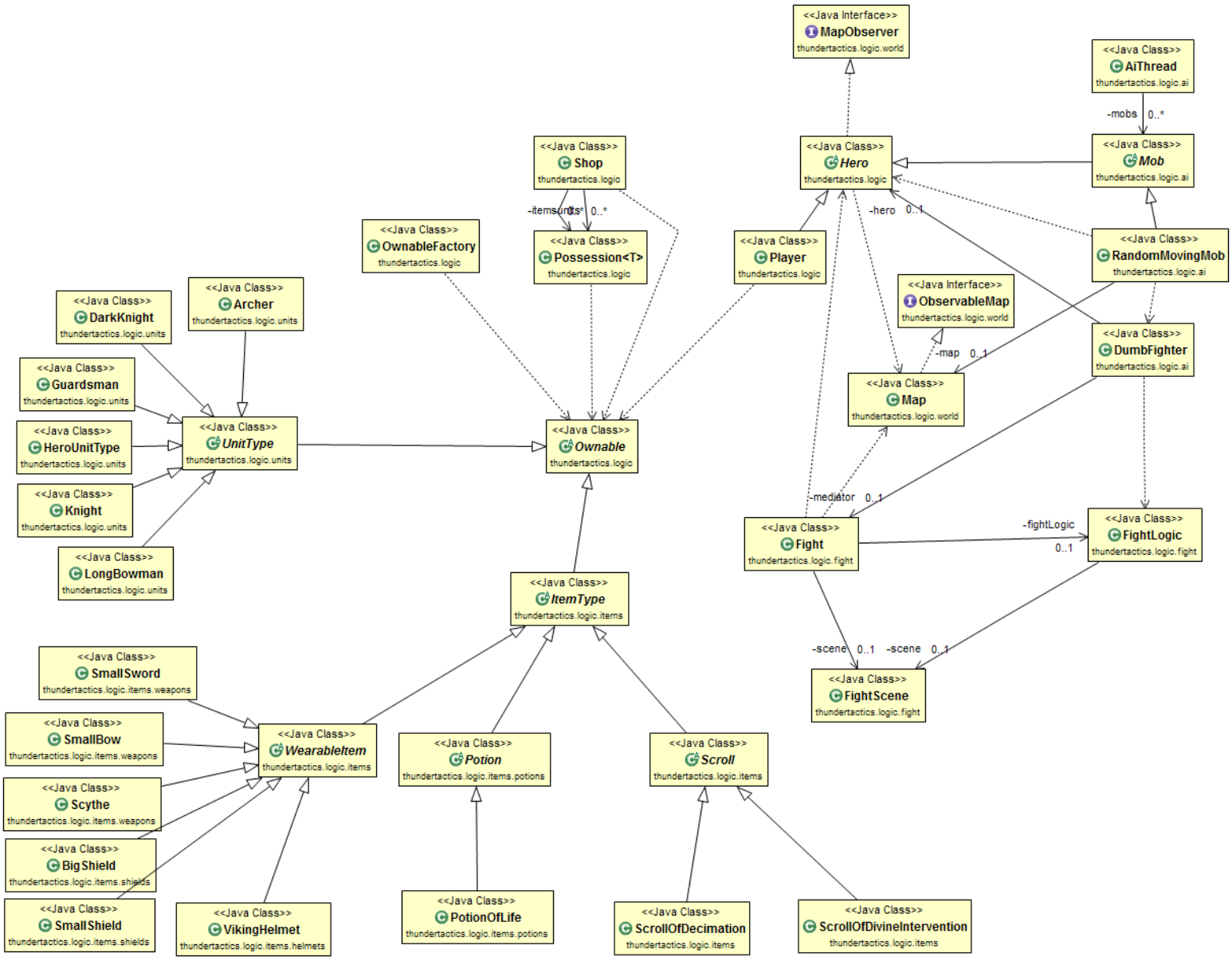
WebSocket represents the next evolutionary step in web communication compared to Comet and Ajax. However, each technology has its own unique capabilities.

Solution

Server

The users play the game by accessing two programs: the **HTTP server** which provides the game resources (models, textures, scripts etc.) and the **Thunder Tactics server application** which is business logic of the game. The players connect to it through the WebSocket protocol from the browser and it connects all users together through functionalities such as moving through the game's world, chating with other players, fighting other players, etc.

It also serves the purpose of validating player actions so that cheating can be elimitated. Providing extensive checks can be computationally costly, but we think we have reached an agreeable level of verifications in order to eliminate the more troublesome cheating while still being able to allow the server to be run on a single averagely built computer.



Class diagram for the server application.

The main components of the **server application** are:

- **WebSocket Server** - handles users connections and users messages
- **Threaded player proxy** (through WebSocket conn.)
- **Fight mediator**
- **AI module**
- **Database storage module**
- **Chat**

WebSocket server

It handles the protocol requirements for communicating through the WebSocket protocol. The

client-side scripts open a WebSocket connection which looks like a HTTP connection with an Upgrade request and then continues with a packet-based bi-directional communication with very little overhead. This is what makes the protocol appealing for real-time web games.

The received packets are deserialized into the game specific classes. In order to make it easy for the client-side JavaScript code, we have settled on using JSON for the format of the messages. Theoretically we could use a binary format for certain messages with few server-side changes, but testing has shown that the bandwidth requirements are acceptable for the number of supported threaded connections.

Threaded player proxy

We have settled from the start on using a threaded model of communication as opposed to Java NIO. While NIO supports a larger number of concurrent connections, that is desirable with protocols such as HTTP where connections are inactive most of the time (keep-alive connections). But in real time games connections are very active and a threaded model offers faster overall response times with the penalty of larger memory requirements and lower number of connections. The computational needs for every user also means that tens of thousands of connections couldn't be maintained anyway.

The game logic is modeled through interfaces which are agnostic of the nature of the communicator (web player or server AI). So the player proxy converts messages received from the WebSocket protocol into method invocations and invocations of the proxy's methods back into messages for the client.

Fight mediator

This component is responsible for fights in game. When an user starts a fight against someone all messages are handled in this component. They stop being notified about other players than those inside the fight.
After each fighters move all others are notified about that move and results of that move(if fighter attack other fighter).
All moves are validates on this component so fighters cannot move when it's not their turn, or do other tricks to cheat the other player and win.
After entering in a fight, players can decide to finish the fight peacefully. So they all are declared winners. This means a fight can have multiple winners, for instance: in a fight enters 3 fighters, 2 decide to ally against the other one and after defeating him to split the loot captured from the looser, by ending the fight peacefully.

AI module

The game contains NPC (non-player characters) which need an ai-module for fight.
AI implemented for NPC is not a very clever one. Meaning that a NPC has 3 options when it is it's turn:

- Attack the opponent hero(if can reach to him)

- Attack the opponent units(if there are units that can be attacked)
- If none of the above move closer to opponent hero.
- If cannot move or attack just defend himself.

Database storage module

All information about users are stored in a **Database.**
When an user tries to login credentials are searched in **Database** and if valid accept user. Also when an user registers his account details are saved in **Database.**
Database is also used to save character info: level, experience, gold, etc.

Chat

There are multiple types of chat messages:

- **Global** - All online players receive the message
- **Public** - Only closer players receive the message
- **Private** - Only a specific person receive the message

Cheat detection & debug

In order to power this components we used:

- **Aspect Oriented Programming**
  - **Profiler** - we store how much time it takes for every method called in every module
  - **MessageStatistics**
  - **SQLExceptionLogger** - it logs every exception thrown for future maintenance
  - **ChatMessageCensor** - censor offensive words before sending to other users
  - **MessageCache** - it cache common fight messages
- **Monitoring-Oriented Programming**
  - **PositionMonitor** - while a player is moving this monitor is storing his position, and when he travels too much in a very short time, it is logged to be inspected by admin
  - **FightMonitor** - it monitors fights to change turns correctly.

Client

Screenshot from game: **Free Mode**

Main components of the **Client application** are:

- **World.js** - a container that contains all graphical elements
    - **Map.js** - ground of the world
    - **Fight.js** - mediates arena fights for players
    - **Player.js** - contains information about the player and also the graphical interface.
    - **ui.js** - creating user interface as html
- **WebSocket** - used to communicate with server

World

Use **WebSockets** for communication with server. The main functionality of this component is to mediate communication between **client components** and/or **server.**

Map - ground of the World

Also, contains other graphical elements like waters and trees.
The ground has been generated with **L3DT.** Trees and waters has been added using **Autodesk 3ds Max**.

Fight.js - mediator for fights

Screenshot from game: **Arena fight**

Contains graphical interface of the arena, in which players are fighting (there are two arenas created using **Autodesk 3dsMax**), and squares on which players can move.
The functionality of this component is mediate the fight (it receives/sends messages to server and show the result to all fighters)

Player.js

Contains graphical interface(with animation) for players(or mobs) and other information about that player, such as level, name, etc.
Graphical interface was created with **Autodesk 3ds Max**.
Players can have different appearance(clothes or body aspect) and can have different equipment (sword, bow, shield, arrows)

ui.js

ui.js is used to create and handle html elements for displaying user interface elements such as minimap, hero statistics, hero controls(attack,open map, defend, etc).

Results, Evaluation

In the table below, we presented which features were expected to be implemented, which were implemented and how, in comparison with expectation.

Feature	**Expectation**	**Result**
Authentication		
Explore free mode		
PvP mode		
Shops		
Inventory		
Banning users		
Clans		
3D Client Interface		
Player leveling		
Stealing gold from defeated opponent		

Green - 100%| Red - none | Yellow - partially

Comparison with other solutions

Games, unlike other software engineering products, are hard to compare in an objective manner. For that reason we have chosen more technical criteria for comparison.

For our comparison we will consider World of Warcraft (a MMORPG) and Heroes of Might and Magic (a TBS game).

Platform

One of our main advantages is that we've build a game which is independent of platforms like: hardware, operating system and browser (although it has to support WebGL and WebSockets). On the other hand, traditional games that are stored locally can be more graphically detailed and faster to load.

After a user enters the world, the game resources are stored in browser's cache (which is limited) and if a user clear his browser cache they will have to be downloaded again. On the other hand, this means that a user can play the game on any computer and he doesn't have to worry about hardware failure, since this will not result in the loss of the game state.

Game play features

Feature	Thunder Tactics	WoW	Heroes
Authentication required			
Realtime fight			
Turn based fight			
Shops			
Items			
Free explore			
Multiple units			
Single player			
Multiplayer			
Infinite game play			
Diplomacy			

Green - available | Red - not available | Yellow - partially

Future work

In the future we could improve the game by adding features which we could not implement due to time constraints and lack of free adequate resources. Some of these features are:

- **Clans**: Users should be able to associate in small or large groups. They should have the posibility of chatting with their teammates, helping each other with advices and items and even fight together.

- **More items**: As of now, the items included in the game are: 2 shields, 3 weapons, 1 helmet and 1 potion. But we would like to add more of these types of items and also others like: boots, gloves, armor.

- **Items drop**: After defeating an mob, it should drop the items it has so that others can pick up and use or sell them.

- **More characters**: Now there are only 2 character models with multiple textures for each, but it would be great to have more models including female characters and non-human characters. We have been unable to include more since 3D models of acceptable polygon counts which are free and come with the required animations are hard to find and we prefered to concentrate on the gameplay as opposed to the looks.

- **Events:** At certain intervals, it should be possible to spawn **special** types of mobs which can have a bigger experience or drop **special items** (which cannot be found in another ways).

Conclusions

The objective of our project was to create an open-source game which is platform independent (by being web based), requires no installation, could be played be multiple users at the same time and which would create a shared experience in the same world that allows them to socialize.

By using software engineering techniques we developed a game with a very dynamic engine which can be maintained and improved upon very easily. We have used aspects and monitors that have helped us to find bugs and exploits and evaluate our project by multiple criteria. This allowed us to improve our project faster.

The game we created actually exceeded our expectations and has almost all of the features planned in the begining.

Some of the gameplay can be previewed in these videos:

- http://www.youtube.com/watch?v=iBoa-jr8kI8
- http://www.youtube.com/watch?v=c9snRkjMrtk

Bibliography

1) WebGL Up and Running, O'Reilly Media (August 2012)
2) WebGL: http://en.wikipedia.org/wiki/WebGL
3) Three.js on GitHub: https://github.com/mrdoob/three.js/
4) Three.js Wiki: http://en.wikipedia.org/wiki/Threejs
5) L3DT official website: http://www.bundysoft.com/L3DT/
6) WebSockets: http://www.websocket.org/
7) MMORPGs: http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game
8) TBS: http://en.wikipedia.org/wiki/Turn-based_strategy
9) World of Warcraft: http://en.wikipedia.org/wiki/World_of_Warcraft
10) Lineage II: http://en.wikipedia.org/wiki/Lineage_II
11) Mu Online: http://en.wikipedia.org/wiki/Mu_Online
12) Heroes: http://en.wikipedia.org/wiki/Hero_Games