AETD - Arch-Enemy Tower Defense
Group 6

Johan Gustafson
Jonas Hellgren
Erik Nordenhök
Olof Ol-Mårs
Felix Wallén

Contents

1 Preface

1.1 Expected readership of the document

This document will be read by the stakeholders that are involved in this project. It will be read to achieve an understanding of why this system is constructed, in what way it will be done and the exact functions that will be available when the system is finished.

The expected readers of this requirements document can be summarized as the following three kinds of people:

- The project supervisor
- The project members
- The potential customer

1.2 Version history

Date	Version	Summary
2007-12-07	0.1	First draft of the document.
2007-12-12	0.2	System Architecture added.
2007-12-17	0.3	System Evolution added.
2007-12-18	0.4	Spelling correction made.
2008-01-02	1.0	First version of the document compiled.

2 Introduction

2.1 Who are the users and what problem does it solve for them?

The users are mainly males between the age of 18 and 25. There are probably other users, i.e. girls and younger or older people, that will play the game and find it interesting, but we intend to build our game so that it may primarily fit males in that age group. The users are also used to playing computer games, but don't necessarily need to have in-depth knowledge about computers. Being able to connect to the Internet, download a file and start a program is enough to play the singleplayer version of the game. For the multiplayer part, knowledge about how to connect to a specific computer using its Internet Protocol (IP)-address is required.

This game is meant to be fun and therefore it solves the problem of boredom for the users. If the other users, mentioned above, are also bored and turn to our game to relieve them of their boredom, that's their choice and we are happy to provide the game for them. However, we aim only to make our intended target audience pleased with the game.

2.2 The main uses of the system

Arch-Emeny Tower Defense (AETD) is a multiplayer game where up to four players battle each other and various computer controlled monsters by building towers to defend themselves. Each player will have a playing field through which the monsters will try to pass. The main goal of the game is to stop them from reaching the opposite side of the field. To succeed in this the player has a variety of towers in their arsenal which he may build on the playing field. The towers are stationary and have different abilities to kill the monsters, such as slowing, high rate of fire etc. The players are supposed to collect gold which he can use to build or upgrade towers and send more monsters to his adversaries. The gold is acquired by killing monsters.

2.2.1 First usage narrative

Bill and Bob are sitting in school, studying for the big exam. They are both 21-years old and fairly used to playing computer games. They are tired, bored and in need of a break. They decide to play a game of AETD, which they both have already downloaded and installed on their laptops. They both start the game and since Bob has the best computer he hosts the game, allowing Bill to join his server by typing in his IP-address after selecting "Join multiplayer game" from the main menu. When both players are ready Bob start the game. Bill builds his towers by selecting them from a menu with the mouse and then clicking on the playing field where he wants them built. His towers kill the monsters very fast and from the gold he earns he chooses to buy and send extra monsters to Bob, who already has problems killing his own monsters. With the extra amount Bob can't handle them and he let too many through to the other side of his playing field. After a set number has made it through he loses and Bill leaves the fight in victory. They both shut down the game to continue with their studying, feeling relived and less stressed, from the fun break.

2.2.2 Second usage narrative

Ben is at home, watching TV, and is really bored. He is 24-years old and plays computer games on a regular basis. He already has the game installed on his computer. He doesn't know if anyone else is playing the game and decides to try the "Singleplayer mode". He therefore selects singleplayer mode from the main menu and the game starts. When playing single player he can't use the function "Send monsters" since there is no opponents to send them to. Ben builds the towers in his favourite way to counter the monsters, who comes from the left side of the field. He kills them all and gets enough gold to both upgrade his towers and build new ones. After a while of playing the game is over because Ben has let more monsters than allowed to the right side of the playing field. Ben, who really enjoyed himself playing the game, feels really satisfied and return to watching TV because his favorite show just started.

2.3 The context/environment in which the system is to be used

The game is meant to be played on a computer running Windows XP and that has the capability to use standard Open Graphics Library (OpenGL). We intend to write the source code in C++, a wide-spread programming language, which can be compiled and run on other operative systems than Windows XP but we can not guarantee that it will work as expected. OpenGL is a free source graphical library that contains functions to implement a graphic environment. To acquire the game, the user must have an Internet connection to be able to download the appropriate files from our web-page. Additionally to use the multiplayer functions of the game an Internet connection is also required. However once the user has the game installed on his computer he can play the singleplayer version of the game without having access to the Internet.

2.4 The scope of the system

Topic	In	Out
Multiplayer on same computer		X
Multiplayer over Internet and network	X	
Singleplayer	X	
Play directly from homepage, without downloading the game		X
Campaign, a single player mode where you follow a set storyline		X
Upgradable towers	X	
High-score		X
Map editor, ability for players to predefine new playing fields		X
Ability to send monsters to opponents	X	
Artificially Intelligent (AI) player		X
Different types of towers	X	
Different types of monsters	X	
3D graphics		X
Fullscreen mode	X	
Windowed mode	X	
Ability to save game		X
Ability to pause game		X
Player chat	X	
Platform independent		X

2.5 Main factors when designing and building the system

- If a host player disconnects or looses, we need to either reroute the hosting to a client player or close the current game session.
- We need to make sure that the response time of the network is low. Otherwise players will be frustrated and bored.
- The requirements for the performance of the game are high, since the players will find it boring if it runs slow. Therefore we need to use efficient algorithms and write efficient code.
- We want the game to work even if you do not have an advanced graphics accelerating card. Therefore we must try to use only the simplest of graphics functions found in the OpenGL.
- Since we have multiplayer and singleplayer game modes that are very similar we need to write the code in a way that lets us implement these game modes easily.

2.6 Technologies and risks

- In our project we are going to use the programming language C++ in order to build our system. We will make an object oriented system where every class is specified individually and have dedicated tasks.
- Since the language C++ doesn't have any garbage collector the risk of getting memory leaks are moderate.
- We are going to use the predefined library for networking included in C++ in order to host and join games.
- For the networking, we will use the User Datagram Protocol (UDP) instead of the Transmission Control Protocol (TCP) since UDP does not check that packages have arrived before sending new ones, which takes time. This lowers the risk of having high response time over the network.
- For our graphical engine we have decided to use OpenGL. Even though OpenGL supports 3D we are only going to use 2D.
- Using OpenGL in C++ should work smoothly since there are many sources of information about how to use the two together. The risk that these two technologies will have a problem interacting is therefore very small.
- All our code will be written in the environments Visual Studio or Gedit and compiled with the g++ compiler.
- For our textures we will use the program Adobe Photoshop. This will be used for all the graphics that we are to draw in any context.

3 Glossary

- **Actor**

  Someone or something that interacts in a use case.

- **Adobe Photoshop**

  Photoshop is a program developed by Adobe for editing photos and drawing pictures.

- **AETD**

  Arch-Enemy Tower Defense is the name of our game.

- **AI**

  Artificial Intelligence, a computer system that acts as if it could think.

- **Blocking**

  A tower is blocking if it is placed in such a way that monsters can find no path between their current position and the goal.

- **C++**

  A programming language that we will use to build our system.

- **Campaign**

  A single player game-mode where you follow a set storyline.

- **Client**

  One of the players in a multiplayer game that connects, rather than create, the game.

- **Client-computer**

  The computer of a client player.

- **Compiler**

  When compiling the computer translates the written code to a language that the computer can read.

- **Executable file**

  A program that can be executed on a computer.

- **Extensions**

  Alternative paths for a use case describing what happens when a use case does not behave as intended.

- **Functional requirement**

  A function that will be included in the final product.

- **G++**

  G++ is a compiler for C++ code.

- **Gedit**

  Gedit is an open source project that allows the user to write and edit text.

- **GUI**

  Graphical User Interface is the graphical interface that the system presents to the user.

- **Host**

  One of the players in a multiplayer game that creates, rather than connects to, a game. There is only one host for every multiplayer game.

- **Host-computer**

  The computer of the host player.

- **IP-address**

  An Internet Protocol-address is used for connecting computers over network. Much like how telephone numbers connects telephones.

- **Main path**

  The path of a use case describing what happens when everything works as intended.

- **Minimal guarantee**

  The minimal result promised for a use case by the system.

- **Monster**

  Object moving from one side of the playing field towards the other.

- **Non-functional requirement**

  A criteria for the system that will be followed.

- **OpenGL**

  Open Graphics Library is a free source graphical library that contains functions to implement a graphic environment.

- **Photoshop**

  See Adobe Photoshop.

- **Playing field**

  An area individual to each player where towers can be built and monsters move.

- **Precondition**

  Conditions that must be satisfied for a use case to be initiated.

- **Processor**

  The part of the computer that makes all the calculations.

- **RAM**

  Random Access Memory, the part of the computer that stores information essential for current running programs.

- **Stakeholder**

  A person or group that has an interest in the outcome or process of a use case.

- **Sub-folder**

  A folder in a computer system that is located inside another folder.

- **Success guarantee**

  The result promised for a successful completion of a use case.

- **TCP**

  Transmission Control Protocol, a specific set of rules used in networking.

- **Tower**

  A stationary object placed on a playing field that shoots at monsters in range.

- **UDP**

  User Datagram Protocol, a specific set of rules used in networking.

- **Use Case**

  A step-by-step description of a function of the system.

- **Visual Studio**

  Visual Studio is an environment developed by Microsoft for writing and compiling code.

- **Web-server**

  A storage space for files on the Internet. It allows users to download data from the server.

4 User requirements definition

4.1 Functional requirements

- **Download the program.**

  The program should be available to download from our homepage.

- **Install the program.**

  The user in our target audience should be able to install the program.

- **Configure game settings.**

  Alter settings such as player name.

- **Play singleplayer game.**

  A game-mode for only one player that does not require an Internet connection.

- **Host multiplayer game.**

  The user makes his computer the host computer of a multiplayer game that supports 2-4 players.

- **Join multiplayer game.**

  The user connects to a host computer of a multiplayer game.

- **Build towers.**

  To purchase and place a tower on the playing field.

- **Select a specific tower.**

  Select a tower already built on the playing field.

- **Sell towers.**

  It has to be possible to sell a built tower.

- **Upgrade towers.**

  All tower types have upgrades that make them more powerful.

- **View individual tower statistics.**

  See information about the selected tower such as damage per shot, rate of fire etc.

- **Monster levels.**

  At set intervals a fixed number of monsters will automatically be sent by the system to the monster starting point of all players playing fields. Each time they are sent, the monsters will be a higher level with increased health. Higher level monsters will give the player more gold once they are killed than lower levels.

- **Monster movement.**

  The monsters will automatically move from one side of the playing field to their goal, always choosing the shortest path available.

- **Kill monsters.**

  Monsters have a certain amount of health and when monsters walk in range of a tower, the tower will shoot it causing a certain amount of damage to it. When the monsters health reaches zero, the monster will die.

- **Receive information about monster's health.**

  The current and maximum health of each individual monsters should be available to view.

- **Send monsters to other players.**

  When sending monsters to another player, additional monsters will appear on an opponent users playing field. This will cost gold for the player sending the additional monsters and killing them will not yield any gold.

- **Chat with other players**

  The users can send text messages to each other.

4.2 Non-functional requirements

- **Performance**

  The computer the game is meant to be run on does at least have a 1.4 GHz processor and 512 MB ram.

- **Implementation**

  The code has to be written in C++ and the graphics must use OpenGL.

- **Space**

  The system should be no larger than 32 MB before installing.

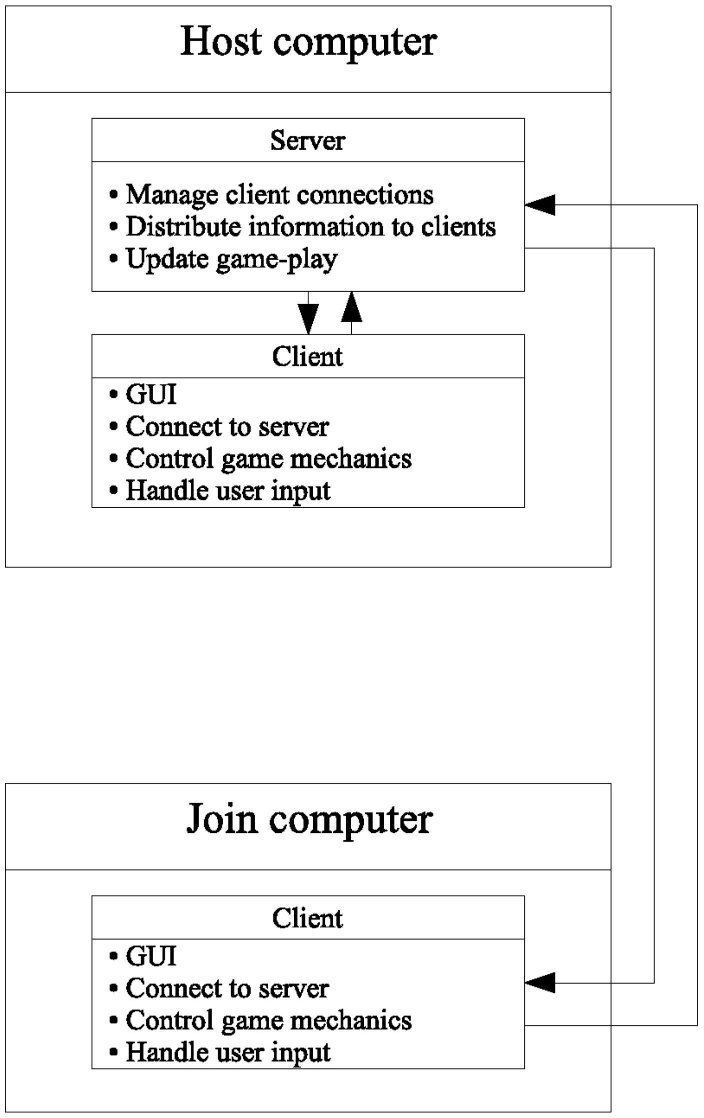
- **Reliability**

  The system will have to sustain the connection between host and clients.

- **Safety**

  No illegitimate access to host- and client-computer should be possible.

5 System architecture



5.1 Server

The server part of each network game manages the clients that connects and tries to connect to the server. For example, it denies clients to join if the game is already at its maximum number of players.

When the game has started the server is distributing information about actions that are undertaken by the different players. For example, when a player builds a tower his client version sends the information about the new construction to the server, which then passes it on to the other clients. Each one of the clients will then update their GUI corresponding to the changes that has been made.

5.2 Client

Each client can connect to a server by specifying the IP-address to the server. When connected, the client communicates with the server to send actions and to receive information about what the other players are doing. It also handles the GUI that is presented to the user, and verifies that the actions made by the user are legal according to the rules of the game.

5.3 Host and join computers

Each computer with an installation of the program AETD will be able to act both as a host computer or a join computer when playing network games. The type of computer is decided when a network game is initiated, the player who chooses to host the game will be the host computer and all other players will act as join computers. The host computer will start a server which he automatically connects to, acting as a client. When the server is running it will be open for connections from clients from join computers that are using an Internet connection. If a user decides to play a single player game his computer will act as a host computer with a server that only allows the client from the host computer to join the game.

6 System requirements specification

6.1 Detailed functional requirements

- **Download the program.**

  The program should be available to download from our homepage.

  The program will be available as an executable install file, which will be used to install the game.

  > Rationale: With a easily accessible program more users will use it.

- **Install the program.**

  The user in our target audience should be able to install the program.

  > Rationale: Installing the program to a specified folder ensures that all files required to run the game are in the correct sub-folders and the program will be able to run as expected.

- **Configure game settings.**

  Alter settings such as player name.

  > Rationale: Every user prefers their own individual configuration.

- **Play singleplayer game.**

  A game-mode for only one player that does not require an Internet connection.

  > Rationale: To provide the users a way to play alone.

- **Host multiplayer game.**

  The user makes his computer the host computer of a multiplayer game that supports 2-4 players.

  > Rationale: A host is required to allow other users to connect to it.

- **Join multiplayer game.**

  The user connects to a host computer of a multiplayer game.

  > Rationale: To play a multiplayer game, at least one user must join a host computer.

- **Build towers.**

  To purchase and place a tower on the playing field.

  Towers are stationary objects on the playing field that automatically shoots monsters that come into their range of fire. A tower built on the playing field prevents the monster from walking on the spot where the tower is built, thus forcing the monster to find a different route around it. A tower can not be built if the chosen spot is blocking, meaning the placement of the tower makes the monsters unable to reach the goal because of monsters not being able to move through towers.

  > Rationale: The user needs towers to kill the monsters, thus preventing them from reaching their destination.

- **Select a specific tower.**

  Select a tower already built on the playing field.

  > Rationale: The user needs to be able to select their towers to use the functions available such as upgrading, selling or seeing additional information.

- **Sell towers.**

  It has to be possible to sell a built tower.

  When selling a tower, the player will be refunded part of the cost of building and upgrading the chosen tower. It will also be removed from the playing field making the slot vacant allowing monsters to pass through and giving the possibility of building a new tower at the vacant slot. While a tower is being sold it does not fire at monsters.

  > Rationale: If the user is unhappy with the placement of one of his towers he needs to be able to remove it.

- **Upgrade towers.**

  All tower types have upgrades that make them more powerful.

  Every tower built on the playing field can be individually upgraded which gives the tower better abilities to kill monsters. Such as increasing the damage it does, increasing the rate of fire or giving it a longer range. While a tower is being upgraded it does not fire at monsters.

  > Rationale: Towers need to be upgraded to handle more powerful monsters.

- **View individual tower statistics.**

  See information about the selected tower such as damage per shot, rate of fire etc.

  > Rationale: Statistics about the different towers gives the user information about how efficient they are and this will help him to select the best towers to build.

- **Monster levels.**

  At set intervalls a fixed number of monsters will automatically be sent by the system to the monster starting point of all players playing fields. Each time they are sent, the monsters will be a higher level with increased health. Higher level monsters will give the player more gold once they are killed than lower levels.

  Making the monsters more difficult to kill forces and enables the players to continually build and upgrade their towers to make sure the monsters die before reching the goal.

  > Rationale: To make sure all players get to play, monsters must be sent to their playing fields.

- **Monster movement.**

  The monsters will automatically move from one side of the playing field to their goal, always choosing the shortest path available.

  Since monsters can't move through towers they will need to find alternate routes once the player has built towers on the playing field. The route will have to be recalculated whenever a tower is built or sold.

  > Rationale: The monsters need to move to get to their goal.

- **Kill monsters.**

  Monsters have a certain amount of health and when monsters walk in range of a tower, the tower will shoot it causing a certain amount of damage to it. When the monsters health reaches zero, the monster will die.

  > Rationale: The user must kill the monsters before they reach their goal. If he lets too many through he will loose.

- **Receive information about monster's health.**

  The current and maximum health of each individual monsters should be available to view.

  > Rationale: The monsters health gives the user an idea of how powerful his towers need to be to kill the monsters in time.

- **Send monsters to other players.**

  When sending monsters to another player, additional monsters will appear on an opponent users playing field. This will cost gold for the player sending the additional monsters and killing them will not yield any gold.

  > Rationale: Being able to send monsters to opponents gives the users a mean to defeat each other.

- **Chat with other players**

  The users can send text messages to each other.

  > Rationale: Being able to talk to the opponents is a fun addition to the game.

6.2 Detailed non-functional requirements

- **Performance**

  The computer the game is meant to be run on does at least have a 1.4 GHz processor and 512 MB RAM.

  It is very possible that the game will be able to run on computers slower than this but we will not be testing it and can therefore not guarantee that it will work.

  > Rationale: These specifications are the minimal system requirements because this is the slowest system we will be testing our game on.

- **Implementation**

  The code has to be written in C++ and the graphics must use OpenGL.

  > Rationale: We will use C++ because it is the programming language with which we have the most experience. The implementation of OpenGL in C++ is also well supported. The use of OpenGL is convenient because OpenGL is free.

- **Space**

  The system should be no larger than 32 MB before installing.

  > Rationale: We want a small game to make it easy and fast to download.

- **Reliability**

  The system will have to sustain the connection between host and clients.

  > Rationale: In a multiplayer game, it is important to not loose the connection. Making sure the connection works properly will be an important factor to make the game enjoyable and therefore fun.

- **Safety**

  No illegitimate access to host- and client-computer should be possible.

  > Rationale: If our program makes the computers vulnerable to malicious software attacks then players will be hesitant to install it.

6.3 Use Cases

All types of user inherits the Generic-users goal.

Actors	**Goals**
Generic-user	Play the game
Host-user	Host a game and play it
Client-user	Join a game
Singleplayer-user	Start a Singleplayer-game
Host-computer	Allow Client-computers to connect to a game
Client-computer	Connect to a game
Web-server	Distribute the game

6.3.1 Install the game

Primary actor: Generic-user.
Secondary actor: Web-server.
Stakeholders and interests:
Generic user - Wants the game to be installed on his PC.
Web-master - Wants to give the user an easy access to the game.
Preconditions: The user has a computer that meets the minimum hardware requirements with an Internet connection.

Minimal guarantee: Nothing unrelated to the game is downloaded or installed.
Success guarantee: The game is properly installed on the users computer.
Main path:

1. The user opens a web-browser of his choice and enters the adress of our website.
2. The user downloads the game to his computer and runs the executable install file.
3. The installation prompts for a folder to install in which the user enters.
4. The game is installed in the specified folder.

Extensions:

1a. Web-server unreachable.

    1. The user can't download the game and the use case ends.

6.3.2 Start singleplayer game

Primary actor: Singleplayer-user.
Secondary actor: None.
Stakeholders and interests:
Singleplayer-user - Wants to play the game in singleplayer mode.
Preconditions: Game installed on a computer that meets the minimum hardware requirements.
Minimal guarantee: The game starts in singleplayer mode.
Success guarantee: The game starts in singleplayer mode.
Main path:

1. The user starts the game on his computer.
2. The user specifies that he wants to play a singleplayer game.
3. A singleplayer game starts.

6.3.3 Host multiplayer server

Primary actor: Host-user.
Secondary actor: Host-computer, one or more Client-users and Client-computers.
Stakeholders and interests:
Host-user - Wants other players to connect to his server.
Client-users - Wants to join an existing server.
Network administrator - Wants the network to be safe and work properly for the users.
Preconditions: Game installed on a computer that meets the minimum hardware requirements with an Internet connection.
Minimal guarantee: Give Client-users the possibility to join a multiplayer-serer.
Success guarantee: The multiplayer-game gets hosted and starts.
Main path:

1. The user starts the game on his computer.
2. He specifies that he wants to play a singleplayer game.
3. The system now makes his computer the Host-computer and allows Client-users to connect.
4. When one to three Client-users has joined, he can start the game.
5. The multiplayer game starts.

Extensions:

5a. No Client-users connect.

  1. The Host-user can not start the game and the use case ends.

6.3.4 Join multiplayer server

Primary actor: Client-user.
Secondary actor: Host-user, Host-computer, other Client-users and Client-computers.
Stakeholders and interests:
Host-user - Wants other players to connect to his server.
Client-users - Wants to join an existing server.
Network administrator - Wants the network to be safe and work properly for the users.

Preconditions: Game installed on a computer that meets the minimum hardware requirements with an Internet connection.
Minimal guarantee: An attempt to connect to the given IP-address is made.
Success guarantee: The client user gets to join and play a multiplayer-game.
Main path:

1. The user starts the game on his computer.
2. He specifies that he wants to join a multiplayer game.
3. He is prompted for an IP-address and he enters the IP-address of the Host-computer of the server he wants to join.
4. He waits for the Host-user to start the game.
5. The Host-user starts the game and the Client-user gets to play.

Extensions:

3a. No Host-server exist with the entered IP-address.

   1. The Client-user is prompted to reenter the IP-address.

3b. The Host-server already has four players connected.

   1. The Client-user gets a message that the server is full.
   2. The Client-user is promted to enter a new IP-address.

6.3.5 Build towers

Primary actor: Generic-user.
Secondary actor: None.
Stakeholders and interests:
Generic-user - Wants to build a tower.
Preconditions: The Generic-user is currently playing in an active game.
Minimal guarantee: An attempt to build the tower is made.
Success guarantee: A tower gets built on the playing field.
Main path:

1. The Generic-user selects the type of tower he wishes to build.
2. He specifies the location on the playing field where he wants the tower to be built.

3. The system builds the tower as specified and debits the gold required.

Extensions:

2a. The specified spot is occupied.

   1. The tower is not built, no gold is debited and the user gets a message that the spot is occupied.

2b. The specified spot prevents the monsters from finding any path to the goal.

   1. The tower is not built, no gold is debited and the user gets a message that the spot is blocking.

3a. The user can't afford the selected tower.

   1. The tower is not built, no gold is debited and the user gets a message saying that he needs more gold.

6.3.6 Upgrade towers

Primary actor: Generic-user.
Secondary actor: None.
Stakeholders and interests:
Generic-user - Wants to upgrade one of his towers.
Preconditions: The user has built a tower on the playing field.
Minimal guarantee: An attempt is made to upgrade the tower.
Success guarantee: A pre-existing tower on the playing field gets upgraded.
Main path:

1. The user selects the tower on the playing field that he wishes to upgrade.
2. The user performs the task for upgrading the tower.
3. The gold needed to upgrade the tower is debited and the system starts the upgrade process for the selected tower.
4. After a set amount of time, the upgrade is completed.

Extensions:

3a. The user can't afford to upgrade the selected tower.

   1. The tower does not start the upgrade process and no gold is debited.

6.3.7 Send monsters

Primary actor: Generic-user.
Secondary actor: Other Generic-users.
Stakeholders and interests:
Generic-user - Wants to send monsters to an opponent.
Preconditions: The Generic-user is playing in a multiplayer-game.
Minimal guarantee: An attempt is made to send monsters.
Success guarantee: Monsters is sent to the opposite player.
Main path:

1. The user specifies that he wants to send monsters to another player.
2. The user is asked to specify which player to send the monsters to.
3. The gold needed to send monsters is debited and monsters are sent to the selected players playing field.

Extensions:

3a. The user can't afford to send monsters.

   1. No monsters are sent and no gold is debited.

6.3.8 Alter game settings

Primary actor: Generic-user.
Secondary actor: None.
Stakeholders and interests:
Generic-user - Wants to configure his game settings.
Preconditions: The game is installed on the computer.
Minimal guarantee: No unwanted settings are made.
Success guarantee: The chosen game settings are made.
Main path:

1. The user starts the game.
2. The user specfies that he wants to configure game settings.
3. The user sees the different settings - change name, etc.
4. The user changes his in-game name, accepts the changes and exits the menu.

6.3.9 Sell towers

Primary actor: Generic-user.
Secondary actor: None.
Stakeholders and interests:
Generic-user - Wants to sell one of his towers.
Preconditions: The user has built a tower on the playing field.
Minimal guarantee: An existing tower on the playing field is sold.
Success guarantee: An existing tower on the playing field is sold.
Main path:

1. The user selects the tower on the playing field that he wishes to sell.
2. The user performs the task for selling the tower.
3. The gold earned for selling the tower is received and the selling process starts.
4. After a set amount of time, the tower is sold and the spot empty.

6.3.10 Send in-game chat message

Primary actor: Generic-user.
Secondary actor: Other Generic-users.
Stakeholders and interests:
Generic-user - Wants to write and send a message to an opponent.
Other Generic-users - Wants to recieve a message from an opponent.
Preconditions: Users must be in a Multiplayer-game.
Minimal guarantee: Written messages is sent to the other players.
Success guarantee: Written messages is sent to the other players.
Main path:

1. The user specifies that he wants to send a chat message.
2. The user inputs his message.
3. The message is displayed on all the other users screens.

7 System evolution

7.1 Fundamental assumptions

- The user will have access to an internet connection.
- AETD is only required to operate on computers running the Windows XP operating system.
- AETD is only required to operate on computers with support for OpenGL.
- AETD is only meant to be used as an entertainment product.

7.2 Anticipated changes

7.2.1 Hardware evolution

Hardware standards are changing at an ever-increasing rate nowadays. However, since AETD has rather low hardware requirements, hardware evolution should not pose a problem. Neither should graphics incompabilities be a concern, as OpenGL is highly compatible and will almost certainly be compatible with future graphics hardware.

7.2.2 Changing user needs

Since AETD is meant only to be used for entertainment, it is difficult to speak of any tangible user needs. As with all computer entertainment products, as computer graphics and interfaces improve, the game will seem crude and lackluster in comparison to new products. However, since AETD is designed to entertain with its gameplay rather than with dazzling graphics, hopes are that it can enjoy longlevity.

7.2.3 Other

There is the possibility that game-balance issues will be discovered after the game is published. Thus, it may be required to make small changes to the code. As AETD will solely be distributed on the internet, new versions can easily be made available to users. There is also the possibility of expanding the game incrementally over time.

Index