# D.U.N.E.

Group 11
Klas Flodin
Kaj Sandberg
Anders Ljungqvist
Mikael Nilsson
Erik Nikkola

# 1. Preface

## 1.1. Expected readership of this document

This document is meant to be read by the stakeholders of this project, where stakeholders are the project team members and any possible investors.

## 1.2. Version history

2008-01-02 Version 1.0 The first version

# 2. Introduction

## 2.1. Target demographic

The gaming community is an aging one. According to Sveriges branschförening för multimedia Dator- & TV-spel(MDTS) the average age is about 29 years and rising. A lot of these players remember and still talk of the old games that have made a big impact on their lives. This is evident in the rising amount of remade games and forums about them. For example the Retrogaming Roundtable forum has over a million posts and around 20000 users.

Dune 2 was one of the first modern Real Time Strategy (RTS) war games when it came out in 1992 on the DOS platform. It was a huge success at the time. However it featured only single player mode and the control scheme only allowed the player to control one unit at a time.

We want to create a Dune 2 inspired game that features the more advanced functions that has arisen in gaming technology since the original release of the game, such as multiplayer gaming and control schemes. Our target demographic is nostalgic players who have played Dune 2 and felt it wasn't all it could have been. Today it would require an emulator such as DosBox to play Dune 2, tools which functions aren't always obvious to the average user. We want to solve this by making a game that is compliant with Windows XP and hope that these modernized components will make the game a more pleasant experience. Of course we recognize that there might be other possible users than our intended demographic but we will intentionally target only the people who have experienced the original game first hand.

## 2.2. The main uses of the system.

D.U.N.E is an attempt to create a game inspired by the old classic Real Time Strategy game Dune 2 but with a touch of the more modern features in the RTS genre. The project will try to keep the same simple graphical concept but add another depth to the original game by adding features such as more flexible tech tree, that is to say a tree with additional technologies the players can research for additional advantages during the game, and a limited possibility to create customized units[1].

[1] Customized units imply that the player has limited ability to affect the speed, weaponry and armor of these units during game play.

## 2.3. User scenario 1

Donny had run into his old buddy Malcolm which he hadn't met for several years. They used to hang out back in the early nineties, a time when they called themselves D.Burns and Mr.Madcow on carefully selected IRC-channels. So now he invited Malcolm to a night of remembrance and nostalgia. They started by opening a couple of beers and when they were past the first moments of catching up they turned on two computers and to play a multiplayer game.

They both agree that when they get joyful chills when they think of Dune 2 but they also agree that the magic is partly gone now that they are older. But Donny kept smiling because he had some news for Malcolm. D.U.N.E - the new Dune 2 inspired game that had been modified with more functionality and a wider range of possibilities and with that extra spice it made the magic better than ever.

They both start up the game on each computer, Donny hosts a network game. He then chooses a map on which to play and then the speed of the game. They are both a little rusty so he selects a slow game pace. Finally he starts the game and the game goes into “wait for other players”-mode. Malcolm then joins the multiplayer game.

They both begin by building the essential buildings that are needed to produce units. Donny sends his units to patrol the perimeter of his base so that he won’t be surprised by Malcolm’s attack. Malcolm concentrates on building defensive buildings. Donny is so happy that he doesn’t realize the amount of beer he drinks and after a while he really needs to go to the bathroom. But the game cannot be paused while in multiplayer mode. When he comes back Malcolm has produced an incredible amount of troops and Donny’s demise is inevitable, Malcolm crushes his opponent in one swift strike and wins the game. Mr.Madcow is back!

## 2.4. User scenario 2

Larry sits back down in front of his computer after a long lunch break. He has a lot of work to do but he doesn't feel up to it at all really. He needs something to get him back in good mood and it comes to him instantly. His new favorite game, not the old game Dune 2 that he played fifteen years ago or so, but a new refreshed clone.

One of the best things about it is that it is so easy to get up and running, no need for DosBox or any other emulator, just start the game and play. He starts up the game and it only takes a second even though his office computer has been with him for quite a few years. He loads a game that he had saved quickly earlier this morning when his boss had come running into his office and instantly he is back in a world of his own, the world that brought him back in time before life got too serious.

Resuming the game, he had forgot that he was in the middle of a battle and it doesn't take many seconds before he dies so he reloads the saved game. It might be called cheating, but what the heck. He realizes this time that he can select multiple units by drawing a box around them and thereby more quickly move his troops, this being a feature that Dune 2 never had. Unfortunately he had not managed to collect enough resources this morning to produce enough units to win. Both armies lose all units. But the battle took place just outside Larry’s base and left a lot of dead unit debris. Larry

sends out one of his harvesters to recycle the debris and suddenly he has enough resources to produce a vast army. After that he wins the battle in a few strikes and he's jumping up and down and shouting in euphoria in his office chair, when he remembers where he is and, now in a much better mood, returns to this afternoons work.

## 2.5. The context/environment

In which context/environment is the system to be used?

This game can be used whenever the player has spare time and an available computer. It can be played at home, at work as an after work activity or on the bus if you have a laptop. However the game will require some time and attention to play.

The game will be coded in Java and use the OpenGL graphics engine and the OpenAL sound library, so both of these have to be installed in the environment, however we plan to include these with the final game package.

The game will run on Windows XP. It will require a graphics card with support for OpenGL 2.0, a sound card, a CPU in excess of 600 MHz, a minimum of 256MB of RAM and a network connection of at least 256 kb/s for network access. Controls required will be keyboard and mouse. The game supports only one player per computer.

We would like to expand the game so it will be playable on multiple platforms such as Linux etc but such things will not be added in this release.

## 2.6. The scope of the system.

| Topic | In | Out |
|---|---|---|
| 2D graphics | X | |
| 3D graphics | | X |
| Artificial intelligence | X | |
| Cheat detection | | X |
| Cinematic movies | | X |
| Configurable keyboard | | X |
| Custom Soundtrack Playback[2] | X | |
| Customized units | X | |
| In-game help | X | |
| Keyboard input | X | |
| Keyboard shortcuts | X | |
| MINA[3] | X | |
| Two button mouse | X | |
| Multiplayer | X | |
| Multiplayer chat | X | |
| OpenAL[4] | X | |
| OpenGL[5] | X | |
| Original Soundtrack | X | |
| Other input devices | | X |
| Pause multi player | | X |
| Pause single player | X | |
| Predefined maps | X | |
| Randomly generated | X | |
| Real time action | X | |
| Recycle dead units[6] | X | |
| Save/Load | X | |

[2] The user can specify a custom folder from where the system will randomly play back any MP3, OGG or WAV files.
[3] MINA is a network framework developed and maintained by the Apache Software Foundation which extends the built-in Java network capabilities.
[4] OpenAL (Open Audio Library) is a free software cross platform audio API.
[5] OpenGL (Open Graphics Library) is a standard specification for writing applications that produce 2D and 3D computer graphics.
[6] A special unit can be built that will gather resources from the wrecks of destroyed units and buildings.

| | | |
|---|---|---|
| Script based events | | X |
| Set game speed | X | |
| Space environment | | X |
| Story mode | | X |
| Technology tree | X | |
| Terrain elevation | | X |
| Unit experience | | X |
| Statistics after game | | X |

## 2.7. Design factors

What are the main factors that need to be taken into account when designing and building the system?

- The learning curve of the game. The game interface must be intuitive and easy to use.
- The game must provide a continuing motivation to play.
- It must appear to be challenging but still winnable to the experienced player as well as the beginner.
- The game needs to be balanced. Both in consideration to the enemy as well as unit balance.
- The AI must be challenging but not too much so.
- Game pace. Players are smart but the computer is much faster at performing simple tasks.
- Hardware performance must be considered. The game must run smoothly on the system requirements stated in section 3.
- The markets needs and wants must be considered.
- If a player dies or disconnect in multiplayer and there are more than one players left the game will continue.

## 2.8. Technologies and Risks

What technologies are to be used in the project and what are the risks of using them?

- **OpenGL**
  We will use OpenGL to present graphics.

  Risk: OpenGL is a widely used and implemented specification and should therefore present very little risk. Only one member in the project team has extensive knowledge in OpenGL which might present a risk.

  Solution: Since one of the project members is well versed in OpenGL he will educate the other members where needed.

- **OpenAL**
  OpenAL is a widely used and implemented specification maintained by Creative

  Risk: Because OpenAL is maintained by Creative there is very little risk of any major bugs within the library. But since we have only little previous experience with OpenAL there is a certain risk we might run into problems during development.

  Solution: We will appoint a project member to immediately read up on the library and its use with Java, he will later educate the rest of the project team as needed.

- **Java**
  The main development language of the game. Most of the core functions will be written in Java.

  Risk: Java is a stable and mature language that is largely used and supported by a large part of the computer developing community. All project members are very experienced with Java which makes the risk to use Java for development very small.

  Solution: The risk is so small no special actions will be taken to minimize it.

- **MINA**
  MINA is a network framework developed and maintained by the Apache Software Foundation which extends the built-in Java network capabilities.

  Risk: MINA is a part of the Apache Software Foundation supported frameworks and is therefore relatively safe to use. The knowledge of and experience with MINA in the project team is very limited.

  Solution: Every group member is experienced in the Java network core that MINA builds on and the Apache Software Foundation homepage has a great deal of documentation and a number of tutorials to assist us.

- **Eclipse**
  Eclipse is the main development CASE tool that we have chosen.

  Risk: Eclipse is a very widely used CASE tool and all team members have a large experience with it, therefore the risk is negligible.

  Solution: None needed.

- **Soundtrack**
  We plan to support the OGG, MP3 and WAV audio formats for soundtrack playback.

  Risk: The three listed formats are mature technologies and the playback of such files is part of OpenAL and Java support libraries, the risk is therefore negligible.

  Solution: None needed.

# 3. Glossary

**Attack, unit**
A basic unit-to-unit interaction or unit-to-structure interaction. The attacking unit will attempt to cause damage to, and eliminate, it's target.

**Bandwidth**
A measure of a network connection's information throughput. Measured in bit/s or kbit/s.

**Bit**
The smallest information package in a computer.

**Byte**
A small collection of bits in sequence, generally the smallest usable information structure of a computer program.

**Cheat**
An exploit of a weakness of the system to gain an advantage inside the game that was not intentionally designed.

**Client-server architecture**
A networking architecture where one computer system acts as the manager or referee of the game.

**CPU**
Abbreviation for Central processing unit. The core mathematical unit of the computer.

**Database**
An information repository used to store and retrieve data to be used by the game.

**Disconnection**
A user can get disconnected from a network. Disconnection from a network means that communication over the network is no longer possible.

**Environment lock-up**
If the computer freezes and can no longer accept commands, it is said to be locked up.

**Execution**
Generally, starting a process such as the game or an action in the game.

**Flag (marker)**
A variable attached to an object in the code to signify a state the object is in.

**Harvest**
Gather a resource inside the game.

**Host system, multiplayer**
The computer which acts as the host in a client-server architecture during a multiplayer game.

**I/O device**
A physical device attached to the computer used by the computer to display information or to give the computer commands. For example a keyboard, mouse or monitor.

**I/O stream**
A way of the computer to receive or transmit information to i/o devices or from files on the computer.

**Indestructible, unit**
An indestructible unit cannot be removed from the unit database through direct actions of a player, normal units can by large not affect it. However, normal units can often be affected by the indestructible unit.

**Initiation of**
The start of an action in the game

**Input**
Either: (a) information from an i/o device to the computer or (b) key information used by one of the game's process'

**Interface**
The collective name for the menus, map displays, and other representations that the user will see on his or her screen.

**Java**
A programming language.

**Java Runtime Environment**
See Java Virtual Machine

**Java Virtual Machine (JVM)**
A platform which enables Java programs to be executed similarly on all computers that has a JVM installed.

**kbit/s**
Kilobit per second. A measure of amount of information transferred over time through a network connection.

**Map file**
A file on the computer which describes a game environment.

**MINA**
A network framework

**Multiplayer**
A game mode where several players compete with or against each other in the same game.

**Neutral unit**
A unit that is not controlled by one of the competing players in the game.

**OpenAL**
Abbreviation for Open Audio Library. An open source sound solution used for handling 3-dimensional sound.

**OpenGL**
Abbreviation for Open Graphics Library. A solution for 2- and 3-dimensional graphics.

**OS**
Abbreviation for Operating System. The base program of a computer, eg. Microsoft Windows XP

**Packet**
A package of bytes used in network communication to better ensure and control that no information is lost during transfer.

**Packet loss**
A concept of where a packet is lost or altered in transfer over the network.

**Player**
Ambiguous: a competitor for victory inside the game. Generally: a human user of the game.

**Quit game**
The action of terminating the game to operating system or terminating a game session to the game's main menu.

**RAM**
Abbreviation for Random Access Memory. A work memory used by the computer to store temporary data.

**Research pool**
The collection of research a player has done during a game session.

**RTS**
Abbreviation for Real Time Strategy. A game with strategic elements that does not pause waiting for actions during normal game play.

**Save game**
The concept of saving a game session to later return to the same game session regardless of whether the computer has been intermittently turned off.

**Session, game**
A unique game. Similar to how chess pieces are reset to their starting positions between sessions, different computer game sessions are reset to their starting values.

**Singleplayer**
A game type where the user competes against players not controlled by other users.

**Specification, unit**
The specification of a unit determines how it performs in its interactions in the game world.

**Structure, game**
A game structure is a representation of a unmovable object in the game world.

**Synchronization**
To ensure that the system during a multiplayer game is represented equally with all participants.

**System, ambiguous**
The operating system.

**System, game**
A general reference to the game.

**System state**
A general reference to the data the game system currently uses.

**TCP/IP**
Transmission Control Protocol and Internet Protocol. One of the most broadly used all-purpose protocols for networking over the internet.

**Team, multiplayer**
A group of players competing for mutual victory during a multiplayer game.

**Technology, game**
A concept of technology in the game with no physical game world representation, functions like a key to unlock abilities or new units, structure or technologies in the game.

**Unit, game**
A game world representation of a mobile entity used to interact with the game world.

**Unit ID**
A concept for keeping track of individual units by the game system.

**Variable**
An information representation that is changeable by nature.

**Weapon, unit**
The concept of a unit's offensive capability in interaction with other units.

# 4. User requirements definition

## 4.1. Functional requirements

### 4.1.1. Game session Control

*4.1.1.1. Starting a new game*

The user shall be able to start a new game by either running a pre made map or by using a built in function for creating a randomly generated unique map.

*Rationale:* To increase game longevity, a random map generator should be supplied so the user does not get bored of the game.

*4.1.1.2. Resuming an old game*

The user shall be able to resume a previous unfinished game if this game state has been saved.

*Rationale:* Some maps or battles could take a long time to finish so a load function to resume an old game shall be supplied

*4.1.1.3. Saving a game*

The user shall be able to save.

*Rationale:* Some maps or battles could take a long time to finish so a function to save the game state shall be supplied

*4.1.1.4. Pausing a game*

The user shall be able to freeze the current game state.

*Rationale:* For whatever reason the user might feel the need to leave the game. Therefore the user shall be able to freeze the game

### 4.1.2. Production

The user shall be able to construct units, buildings and initiate research.
In order for a user to improve in strength regarding army and base the user is able to build new buildings and produce new units, predefined and custom ones, as well as initiate research to gain new technology for use in production.

*Rationale*: This is part in the strategy in the RTS (real time strategy) genre, for any user to be able to build up his strength in units, buildings and research. This is to aid in the goal of winning the game.

### 4.1.3. Economy

The system shall have one form of resource which is continuously required to produce more units, buildings and commit research. The resource shall be called credits. The amount of the resource available to the user will vary depending on the user's actions, this includes using certain units and buildings to get increase his resource amount.

*Rationale:* By forcing the user to amass a resource for any form of progress we can control the pace of the game play by adjusting the speed of the replenishing of the resource. The decision to keep it at only one resource was for simplicity and to keep it in spirit with the original game.

### 4.1.4. Improvements

Any user shall be able to research improved technology for his units and buildings. To be able to improve units and buildings over time the user can research new technology at the cost of in-game resources.

*Rationale*: To make the game more challenging this adds to difficulty instead of having everything available from the beginning.

### 4.1.5. Factions

The system shall offer the user a selection of different unique factions. The factions are different approaches to the game, each offering unique traits and special units.

*Rationale:* This is a very common concept amongst all forms of similar games. By creating different factions which encourage different styles of play the game gets a further level of variety ensuring that the user will not get bored and continue to use the system.

### 4.1.6. Unit Design

The user shall be able to modify certain base units to create custom units.
Selected ground and air based vehicles are able to be modified by the user, this allows for further enhancing their uses and strengths. The customization can only be done through combining certain predefined unit parts.

*Rationale:* By adding customized units to the game some more depth can be achieved. This makes the user to adapt each starting faction's play style more to his liking and will make his experience with the system better. However, we limit it to certain types and a module based system as to not make it too complex to scare the user away.

### 4.1.7. Unit/building handling

The system shall allow the user to correctly control all units and buildings by using only a mouse and keyboard.
The mouse shall have two buttons at least. All forms of unit interaction by the user will be handled like this.

*Rationale:* The most common way to control any similar game is by using the mouse to select units and keyboard shortcuts to assist in the various tasks. This is a very flexible way for the user to work his units and ensures that the user does not feel strained trying to use the system.

### 4.1.8. Combat

The system shall handle real time combat. (as opposite of turn based combat with the opponent ) and provide a computer controlled opponent for single player games. To make the game playable in single player mode there shall be an AI controlling the user's opponent. During multiplayer, to get a more real feeling to the combat it shall be in real time that is not dependent on who made the last move.

Rationale: This is to make the game more interesting by today's standard and it makes strategic thinking and planning more important.

### 4.1.9. Network

The system shall provide multiplayer option for the users to be able to play against each other over the network. One user shall be able to act as a host and let other users connect. Once all users are connected they shall be able to start a game.

*Rationale:* Multiplayer option is a requirement due to a design choise. We focus on this because it helps increase the user base and helps increase the longevity of the game.

### 4.1.10. Configuration

The system shall provide an interface for configuring the players system.
The user shall be able to change the settings in game without having to configure the game externally by editing a configuration file.

*Rationale:* Only allowing the user to select specific options helps prevent errors that may have been caused by the user by the user choosing using settings that aren't compatible.

### 4.1.11. Ending game

Any user participating in a game shall be able to quit the game at any time and the system shall end the game when one or more users are victorious. When the game ends, regardless if a user quit on his own behalf or if someone won or lost the game, user will be returned to first input screen.

*Rationale*: The end game function is needed so that the game can be restarted, reconfigure or exited at any time.

## 4.2. Non-functional requirements

Product requirements

### 4.2.1. Performance requirements - Minimum Specification

The system shall deliver required or above performance only on a specified level of minimum hardware and software. System performance will not be guaranteed on hardware and/or software not meeting this requirement.

*Rationale:* As it is impossible to deliver the same performance on all type of computers a minimum level of performance has to be specified on a minimum level of hardware and software requirements that have been set by the developer. If the environment the system is to run in is below this level the system performance might be affected.

### 4.2.2. Space requirements - Required available memory

The system shall require a specified amount of available permanent memory for installation and temporary memory for execution. The system shall not install or execute if less than required memory is available.

*Rationale:* Since the system will contain a large amount of graphical elements and system files it will need to install on the user's environment and a certain amount of hard disk drive space needs to be available for such an installation to be possible. Furthermore, during execution some of these files will be loaded into the temporary memory for use by the system and a certain amount of RAM memory thus needs to be available.

### 4.2.3. Efficiency requirements - Time constraints

The system shall take no longer than a specified amount of time to perform the initialization, save, load, random map generation and pause functions. The time is measured from the moment the user interacts to start the function until the next user interaction is available.

*Rationale:* If the user has to wait a longer time for any major system function the user will lose patient and consider the system inefficient. This might limit further uses of the system by the user which is not desirable, therefore acceptable limits for the most important functions have been set.

### 4.2.4. Reliability requirements - Mean time between critical failures

The system shall at most have one critical failure per 20 executions. When such a failure occurs the game will have no saved information of its current game except for any previous saves the user has done. The system must have at least have the minimum required amount of available memory or this requirement might not be upheld.

*Rationale:* If the game continuously crashes the user would see it as unreliable and not worth wasting the user's time on. However, aiming for a bug free game that never crashes is very unrealistic, so a limit of 20 launches between each failure is seen as acceptable.

### 4.2.5. Learnability requirements - Time to learn to play

The system shall be quick to learn for a player that has played other games in the same genre. Games in the same genre are generally called real-time strategy (RTS) games and largely share the same concept. They usually consist of battles between opposing forces occurring in real time where the users control their own units via mouse and keyboard. Some famous games in this genre are Command & Conquer, Red Alert and Warcraft – furthermore Dune 2 is often seen as the original definer of the genre. Learning the game is seen as being able to play it will enough that the user can win a game.

*Rationale:* A user that previously has played any RTS game should find the game play similar enough that he within a few hours can learn all of the features of the system, and therefore win a game.

### 4.2.6. Usability requirements - Limit of required interactions

Every command and action in the game shall be designed so they can be performed with a few interactions by the user. An interaction is defined as a mouse button click or a keyboard key press by the user.

*Rationale:* If the user had to use a longer sequence of interactions to perform an action the user might soon get annoyed or strained. However, if too few interactions are needed the game will become too simple and the user will not see it as challenging enough to play again. To ensure that the game is kept easy to use yet challenging to play the goal is to set a reasonable amount of interactions per user action.

### 4.2.7. Scalability requirements – Multiplayer

The system multiplayer function shall be scalable enough to allow several players without requiring a fully dedicated system to controlling the game.

*Rationale:* A common concept in the genre is to not have a central server arrange the multiplayer games, instead one of the users act as host and the other users connect to the host user over the network. This solution to multiplayer games is not as scalable but much cheaper as there is no costs for a dedicated server to always be on.

Organizational requirements

#### 4.2.8. Implementation requirements - Development language

The system shall only be developed in one development language.

*Rationale:* To make development easier all of the system will be developed in the same language. This not only ensures that it's an easier development process, it also removes the problem of interoperability between languages.

External requirements

#### 4.2.9. Safety Requirements – Multiplayer security

The system shall ensure that no harmful data can affect the user's environment through the game during a multiplayer session. The system will use a network for communicating with other users during multiplayer sessions.

*Rationale*: Security is always an issue when a network is involved as this leaves the user's environment open for foreign attacks of various kinds. All that the system can do is try to limit malicious intents by ensuring all data it reads over the network is valid.

### 4.3. Use cases

For use cases see section 8 - appendices

# 5. System architecture

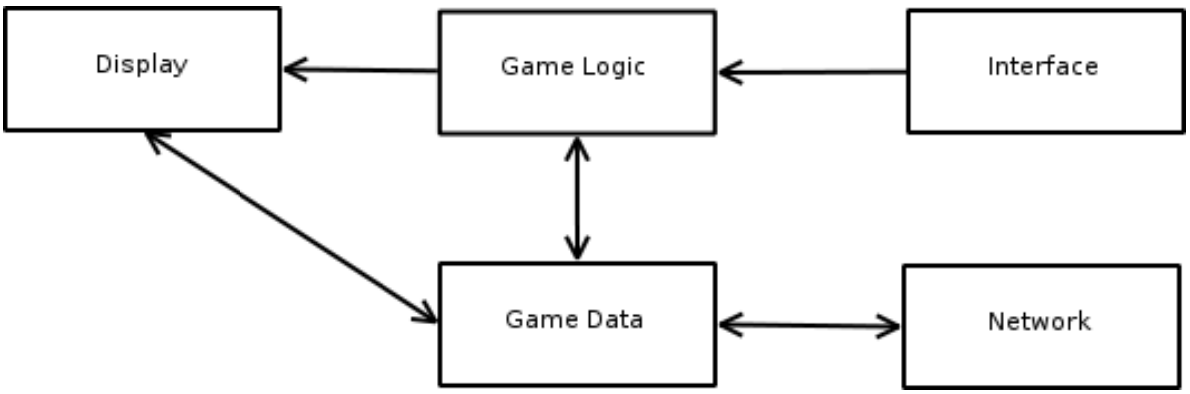
**Game data**
All data and information is loaded and stored in a database so that it can be used by other parts of the system without having to duplicate data.

**Display**
Renders all data it receives.

**Interface**
Handles all input from the user and sends out events to the game logic.

**Network**
Handles all network communication to other clients.

**Game Logic**
Handles all logic for the game.



# 6. System requirements specification

## 6.1. Functional System requirements

### 6.1.1. Game session control

#### *6.1.1.1.**Starting a pre-made map***

| | |
|---|---|
| Function | Starting a pre-made map |
| Description | Starts a game session using predefined defaults from a file. |
| Inputs | Map file |
| Source | Game system menu |
| Outputs | Map Geometry - the terrain details of the map<br>Starting Position - each player's starting position<br>Neutral Units - position of each neutral unit on the map<br>Resource Fill - the location, spread, and quantity of the map's resources. |
| Action | Load map geometry from the file. Loads starting position values, neutral units and resource values. |
| Requires | User sequence initiation. Existing valid map file. |
| Pre-condition | The game is currently in the game system state. |
| Post-condition | The game state is changed to a game session. |
| Side effects | Any changes that have not been saved in the game system menu are discarded. |

#### *6.1.1.2.**Starting a randomly generated map***

| | |
|---|---|
| Function | Starting a randomly generated map |
| Description | Starts a game session with randomly generated map values. |
| Inputs | Values for: impassable terrain, clear terrain, and resource density |
| Source | Game system menu |
| Outputs | Map Geometry - the terrain details of the map<br>Starting Position - each player's starting position, on a clear terrain tile<br>Neutral Units - position of neutral units on the map<br>Resource Fill - the location, spread and quantity of each of the map's resources |
| Action | Takes the inputs from each of the three variable values and computes a random map with these as seeds. |
| Requires | User initiation |
| Pre-condition | The game is currently in the game system state. |
| Post-condition | The game state is changed to a game session |
| Side effects | Any changes that have not been saved in the game system menu are discarded. |

#### *6.1.1.3.**Load***

| | |
|---|---|
| Function | Load. |
| Description | Load a previously saved game state from the hard drive |
| Inputs | A saved game file \saves* on the hard drive where * is a user specified file |
| Source | Game system menu |
| Outputs | Map Geometry - the map terrain<br>Unit Position - each player's unit and structure position<br>Neutral Units - each neutral unit's position<br>Resource Fill - the position of resources on the map |
| Action | Loads the file's geometry and units into the main game control. |
| Requires | The specified saved game file exist and is of valid format |
| Pre-condition | The game is not in a multiplayer state. |
| Post-condition | The main game control state is returned under the control of the user |
| Side effects | Changes in the current game state or game system menu are all discarded. |

#### *6.1.1.4.**Save game state***

| | |
|---|---|
| Function | Save game state |
| Description | Saves the current game state to a file on the hard disk. |
| Inputs | An on-going game session. Current map geometry, resource spread, position and type of each player's units and structures. |
| Source | Game system menu |
| Outputs | A file on the hard disk in the \saves\ directory |
| Action | Writes the game geometry to the file i/o-stream, then the unit identity, ownership, and position. |
| Requires | A running game session. |
| Pre-condition | The hard disk is accessible |
| Post-condition | None. |
| Side effects | The different unit's current orders are not saved. |

#### *6.1.1.5.**Pause***

| | |
|---|---|
| Function | Pause |
| Description | Holds the current game session in a frozen mode where no orders can be given to units or structures, and no units may take any stored action. |
| Inputs | None. |
| Source | Main game session |
| Outputs | None. |
| Action | Holds any and all orders given to the units and structures. The game state is, effectively, temporarily saved. |

| | |
|---|---|
| Requires | An on-going main game session. |
| Pre-condition | The current game session is no in a multiplayer state. |
| Post-condition | The game session is not paused. |
| Side effects | The game menus and map views are not touched and may be freely navigated. |

### 6.1.2. Production

#### *6.1.2.1.* ***Building construction***

| | |
|---|---|
| Function | Building construction |
| Description | Constructing a building |
| Inputs | A building construction site is placed on the map by the a user |
| Source | Game session UI. |
| Outputs | A construction site representation for the building type until the construction is completed when it shall be replaced with a representation of the finished structure. |
| Action | When a construction site is placed on a specified clear terrain area, start a timer on the construction site that depends on the number of construction structures and construction sites currently active. This timer shall be updated if the conditions change. |
| Requires | The, for the building type being constructed, appropriate construction building, available technologies, sufficient resources to pay for the construction, and sufficient clear terrain to for the site to be placed on. |
| Pre-condition | A running game session |
| Post-condition | The game session's unit list is updated with the structure. |
| Side effects | The terrain the structure is placed on is made impassable. |

#### *6.1.2.2.* ***Unit construction***

| | |
|---|---|
| Function | Unit construction. |
| Description | Creates a unit at the main production facility. |
| Inputs | The unit production queue. |
| Source | Game session UI. |
| Outputs | The session's unit list |
| Action | When a unit construction is engaged, add this unit to the unit construction queue for the unit's type. Once the timer on the unit is finished, place a new unit of that type and configuration near the building specified by the Primary production facilities requirement. |
| Requires | The, for the unit specified, appropriate construction building, available technologies, and sufficient resources to pay for the unit. |
| Pre-condition | A running game session. |
| Post-condition | The game's unit list is updated with the unit. |
| Side effects | The unit may be unable to place on the map when it is |

ready, the unit will wait to be placed until such a time as there is place for it.

#### *6.1.2.3.* ***Primary production facilities***

| | |
|---|---|
| Function | Primary production facilities. |
| Description | Primary production facilities specify the preferred exit point for newly constructed units. |
| Inputs | None. |
| Source | Game session UI. |
| Outputs | XxxPrefFacility variable where Xxx is the unit type. |
| Action | None. |
| Requires | An available production facility for the unit type. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | If the production facility is destroyed, units will be placed near an unspecified production structure for that type that the user controls. If the specified preferred facility is blocked, units will be placed near an unspecified production structure that the user controls. |

#### *6.1.2.4.* ***Unit types***

| | |
|---|---|
| Function | Unit types. |
| Description | Unit types specify basic behavior and production facility for a specific unit. |
| Inputs | None. |
| Source | Unit database. |
| Outputs | None. |
| Action | None. |
| Requires | Each unit is associated with a unit type. The unit type in turn is associated with a production facility and a certain production queue. |
| Pre-condition | The structure type exists. |
| Post-condition | None. |
| Side effects | None. |

#### *6.1.2.5.* ***Production shortcuts***

| | |
|---|---|
| Function | Production shortcuts. |
| Description | Shortcuts from the i/o devices to make production selection go faster. |
| Inputs | Key press from a i/o device |
| Source | Keybindings.ini a file which i/o input is bound to what action. |
| Outputs | None. |
| Action | Adds a unit to the construction queue or activates a building's construction sequence as defined by the keybindings.ini |

| | |
|---|---|
| Requires | i/o device |
| Pre-condition | The production facility for the shortcut is available to the user. |
| Post-condition | None. |
| Side effects | Some keys may be bound to multiple actions, priority shall be given to the selected unit or building. |

### 6.1.3. Economy

#### *6.1.3.1.* ***Currency***

| | |
|---|---|
| Function | Currency |
| Description | The game uses one currency which is used as universal payment for buildings, units and research. |
| Inputs | None. |
| Source | Player Balance variable. |
| Outputs | Display the current currency balance in the UI. |
| Action | Whenever a resource is gathered the currency balance is increased. Whenever a production is initiated, the balance is reduced by the amount required for the production. |
| Requires | The balance is always either 0 or positive |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | A production project may require more currency than the user currently has, this project will then not be initiated for the user and no currency will be deducted. |

#### *6.1.3.2.* ***Harvestable resources***

| | |
|---|---|
| Function | Harvestable resources |
| Description | Harvestable resources may be found across the map. Users will be able to gather these and convert into currency. |
| Inputs | Resource worth. |
| Source | The game map information. |
| Outputs | Game map information. |
| Action | When a unit gathers the resources, the map shall be updated and remove the representation of the resource. The resource amount harvested is added to the user's economical balance variable. |
| Requires | A unit type that can gather the resources. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | None. |

#### *6.1.3.3.* ***Other resources***

| | |
|---|---|
| Function | Other resources. |
| Description | The game provides other means of gaining resources than harvesting them from the map's resources. |
| Inputs | Resource worth. |
| Source | None. |
| Outputs | None. |
| Action | These resources are added to the user's economical balance variable. |
| Requires | Access to the resource which depend on the resource in question. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | Access to this resource may be lost, or it may be temporary in nature. |

#### *6.1.3.4.* ***Salvaging resources***

| | |
|---|---|
| Function | Salvaging resources |
| Description | The process of salvaging can be done on wrecks and ruins on the map by any user who fulfill the game criteria for this. |
| Inputs | The salvaged target's worth. |
| Source | Unit and structure database |
| Outputs | Currency gained from the salvage procedure. |
| Action | The output value depends on the salvaged target's worth linearly, the resulting worth is added to the user's economical balance variable. |
| Requires | A unit with the ability to salvage near a wreck or ruin. |
| Pre-condition | None. |
| Post-condition | The targeted wreck or ruin is removed from the map. |
| Side effects | None. |
| Dependencies | Unit destruction; Building destruction. |

#### *6.1.3.5.* ***Salvaging own units and structures***

| | |
|---|---|
| Function | Salvaging own units and structures. |
| Description | Salvaging is possible to discard unwanted units or structures without having to wait for someone to destroy them first. |
| Inputs | The unit or structure's worth |
| Source | Unit and structure database |
| Outputs | Currency gained from the procedure |
| Action | When a unit is salvaged, a great portion of its initial worth is added to the user's economical balance variable and the unit is removed from the table. |
| Requires | A unit capable of salvaging near the building or unit about |

| | |
|---|---|
| | to be salvaged. |
| Pre-condition | None. |
| Post-condition | The targeted unit or building is removed from the map. |
| Side effects | Removing some buildings has impact on research or production. |
| Dependencies | Salvaging resources |

### 6.1.4. Research

#### *6.1.4.1.**Research***

| | |
|---|---|
| Function | Research |
| Description | The user shall be able to research improved technology for units and buildings. By researching, the user may also unlock previously unavailable structures, units, and upgrade options. |
| Inputs | Non researched technology |
| Source | Research pool |
| Outputs | Researched technology |
| Action | When activated, the technology is put into a research queue. A timer starts on the research menu indicating how far the research has proceeded. Speed of research depends on number of research buildings the player has. The timer shall be updated if the conditions change. |
| Requires | Research capable building and correct amount of resources |
| Pre-condition | There is unresearched technology available |
| Post-condition | Researched technology is made available for construction |
| Side effects | None |

#### 6.1.4.2. *Unlocking research*

| | |
|---|---|
| Function | Unlocking research |
| Description | There are two different kinds of research, unlocking and upgrading. Unlocking research provides new units or new buildings. |
| Inputs | Non researched technology |
| Source | Research pool |
| Outputs | Researched technology |
| Action | When activated, the technology is put into a research queue. A timer starts on the research menu indicating how far the research has proceeded. Speed of research depends on number of research buildings the player has. The timer shall be updated if the conditions change. |
| Requires | Research capable building and correct amount of resources |
| Pre-condition | There is unresearched technology available |
| Post-condition | Researched technology is made available for construction |
| Side effects | None |

#### 6.1.4.3. *Upgrading research*

| | |
|---|---|
| Function | Upgrading research |
| Description | There are two different kinds of research, unlocking and upgrading. Unlike unlocking research, upgrading research doesn't provide new units or buildings but improves upon already existing technology. |
| Inputs | Non researched technology |
| Source | Research pool |
| Outputs | Upgraded technology |
| Action | When activated, the technology is put into a research queue. A timer starts on the research menu indicating how far the research has proceeded. Speed of research depends on number of research buildings the player has. The timer shall be updated if the conditions change. |
| Requires | A previously researched upgradable technology and special buildings associated with the particular research |
| Pre-condition | The technology in question is upgradable and not at its maximum level. |
| Post-condition | All units or buildings using the technology question are automatically and immediately upgraded to conform to the new specifications the upgraded technology brings. |
| Side effects | None |

### 6.1.5. Factions

#### *6.1.5.1.* ***Faction selection***

| | |
|---|---|
| Function | Faction selection |
| Description | At the start of each game, the user shall be given the choice which faction the user will play as. |
| Inputs | The user selects faction when starting a new game on a pre-made or randomly seeded map. |
| Source | The user interface. |
| Outputs | The faction number |
| Action | The selected player faction is added as a variable for the user. |
| Requires | The initiation of a new game. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | None. |
| Dependencies | Starting a pre-made map; Starting a randomly generated map |

#### *6.1.5.2.* ***Faction differences***

| | |
|---|---|
| Function | Faction differences |
| Description | Each faction gives the user certain advantages that the other factions do not have access to. |
| Inputs | Player Faction variable. |
| Source | Research database. |
| Outputs | Research pool. |
| Action | At game start, add the faction's basic technologies to the research pool. |
| Requires | None. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | None. |
| Dependencies | Starting a pre-made map; Starting a randomly generated map; Load; Research; Unlocking research. |

### 6.1.6. Customization

#### *6.1.6.1.* ***Design dialogue access***

| | |
|---|---|
| Function | Design dialogue access. |
| Description | The design dialogue will allow a user to customize his own unit to maximize the user’s strategy effectiveness. |
| Inputs | UI interaction. |
| Source | Game system menu. |
| Outputs | Unit database. |
| Action | The unit shall be saved to the database with a flag notifying the system that the unit is a custom unit and a list of research dependencies. |

| | |
|---|---|
| Requires | None. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | The unit database might become large and heavy with too many custom design units. |

#### *6.1.6.2.**Designing units***

| | |
|---|---|
| Function | Designing units |
| Description | Parameters for vehicle design. |
| Inputs | Engine, weapon, armor, and special item. |
| Source | Design dialogue. |
| Outputs | Unit database object. |
| Action | Store the unit into the unit database with the design marker flag. |
| Requires | None. |
| Pre-condition | None. |
| Post-condition | The unit is available for use as soon as the user fulfills the research dependencies for the unit. |
| Side effects | The unit cost and dependencies are calculated when the database load the unit. |

#### *6.1.6.3.**Design budget***

| | |
|---|---|
| Function | Design budget. |
| Description | The limitation of what may be put on a unit. |
| Inputs | Unit parts and foundation. |
| Source | Research database. |
| Outputs | Budget cost of the unit. |
| Action | Compute the current cost of the unit and compare it to the unit's foundation (or chassis) to see if it is legal. |
| Requires | None. |
| Pre-condition | None. |
| Post-condition | The total budget may not be transcended. |
| Side effects | None. |

#### *6.1.6.4.**Multiplayer designs***

| | |
|---|---|
| Function | Multiplayer designs. |
| Description | How to handle custom designs in multiplayer. |
| Inputs | Customized units from the unit database. |
| Source | Each client's unit database. |
| Outputs | An updated temporary database for each participating player. |
| Action | None. |
| Requires | A multiplayer game session. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | None. |

### 6.1.7. Unit handling

#### *6.1.7.1.**Selecting a single unit or building***

| | |
|---|---|
| Function | Selecting a single unit or building |
| Description | The game shall let the user select a single unit to control its behavior |
| Inputs | Unit ID. |
| Source | I/O device |
| Outputs | Unit ID. |
| Action | When a unit or building is selected the object shall be flagged and a visual confirmation shall be presented to the user. |
| Requires | Mouse click |
| Pre-condition | There must be a unit or building under the cursor |
| Post-condition | The system shall provide the user with visual feedback of the selected unit. |
| Side effects | Control of previously selected units is relinquished. |

#### *6.1.7.2.**Selecting a group of units***

| | |
|---|---|
| Function | Selecting a group of units |
| Description | The game shall provide a selection system that lets the user multiple units or a single building. Multiple units shall be selected with a selection box while buildings can only be selected by specifically selecting it. |
| Inputs | Unit ID. |
| Source | I/O device |
| Outputs | List of unit IDs |
| Action | By using the selection box to envelop the desired units the player selects multiple units. The units are flagged and a visual confirmation is presented to the user. |
| Requires | Mouse |
| Pre-condition | Player owned units must exist within the boundaries of the selection box |
| Post-condition | The system will provide the user with a visual feedback of the selection area |
| Side effects | Control of previously selected units is relinquished. |

#### *6.1.7.3.* ***Controlling units with mouse***

| | |
|---|---|
| Function | Controlling units with mouse |
| Description | The game shall provide a system that let the user control the behavior of a single or multiple units. |
| Inputs | Hardware command action and target area information |
| Source | Mouse |
| Outputs | Unit command |
| Action | The system shall determine the most appropriate command based on whether the selected destination contains an enemy, friend, or empty terrain and carry it out. The system shall also provide the user with visual and audio confirmation of the action taken. |
| Requires | Mouse |
| Pre-condition | One or more units are selected |
| Post-condition | The unit has its command list updated with the command information. |
| Side effects | Previous commands are all over-written. |

#### *6.1.7.4.* ***Controlling units with keyboard***

| | |
|---|---|
| Function | Controlling units by keyboard |
| Description | The system shall provide shortcuts to let the user access certain functions directly via predefined keyboard commands. Keyboard shortcuts are meant to increase command efficiency for experienced players by allowing them to control the behavior of the selected units with multiple input units. |
| Inputs | Keyboard device feed and unit ID. |
| Source | Keybindings.ini a file which i/o input is bound to what action. |
| Outputs | Unit command. |
| Action | When one or more units are selected the player shall be able to use the keyboard to temporarily change the behavior of the mouse clicks or active certain unit specific abilities. |
| Requires | Keyboard |
| Pre-condition | One or more units selected |
| Post-condition | Unit carries out command. |
| Side effects | Previous commands are all over-written. |

### 6.1.8. Combat

#### *6.1.8.1.* ***Controlling units in combat***

| | |
|---|---|
| Function | Controlling units in combat |
| Description | The total unit count can easily overcome what is manageable for the player. Controlling each unit individually will easily become an impossible task. Therefore the system shall be able to manage combat. Still, the player shall be able to alter the behavior of a unit or withdraw the unit from combat if necessary. |
| Inputs | Unit ID and target unit ID. |
| Source | The game's main system |
| Outputs | Unit ID, target unit ID, and attack action. |
| Action | If an enemy unit is within range of a player controlled unit, the player controlled unit shall attack the enemy without the players express command. The player shall be able to counteract his unit's behavior via keyboard or mouse. |
| Requires | The attacking unit has a weapon to attack with. |
| Pre-condition | Enemy unit in range of player controlled unit |
| Post-condition | No current unit commands must be altered. |
| Side effects | None. |

#### *6.1.8.2.* ***Defensive buildings entering combat***

| | |
|---|---|
| Function | Defensive buildings entering combat |
| Description | The game shall provide a combat system for defensive buildings. Defensive buildings should automatically attack enemy units of their intended type. The behavior can be altered by the user by selecting the building and modifying it. |
| Inputs | Structure ID and target unit ID. |
| Source | The game's main system. |
| Outputs | Structure ID, target unit ID, and attack action |
| Action | The defensive building shall automatically attack any enemy units that enter its range. |
| Requires | The building is capable of attacking the unit entering its range. Unit is not on player's team. |
| Pre-condition | Enemy unit is in range. |
| Post-condition | None. |
| Side effects | The buildings behavior can be changed by the player.<br>The building is not be able to attack the unit type in question and therefore ignores the unit. |

#### 6.1.8.3. *Computer controlled opponent*

| | |
|---|---|
| Function | Computer controlled opponent |
| Description | If no other human players are available or no network connection is available the player may engage in a single player game. A single player game is a mode where there is one human player and his opponent is controlled by the computer. |
| Inputs | Computer generated unit and building commands. |
| Source | Game's main system |
| Outputs | Command lists and queues for entities controlled by the computer player. |
| Action | The computer handles all actions of the opposing player. The computers behavior changes during the game based on the opponent's action and the environment. The changes are based on the preselected difficulty level |
| Requires | None. |
| Pre-condition | Only one human player is running the game |
| Post-condition | None. |
| Side effects | None. |

#### 6.1.8.4. *Indestructible computer controlled neutral units*

| | |
|---|---|
| Function | Indestructible computer controlled neutral units |
| Description | Indestructible units are objects that do not belong to a faction. These are added to reflect that it's not always possible to solve every hazard with violence. Storms or tornados for instance are not possible to shoot down and should therefore be indestructible. |
| Inputs | Unit ID and behavior list. |
| Source | The game's main system. |
| Outputs | Unit ID and action for that unit. |
| Action | The indestructible units shall have a predefined list of behavior that is randomly selected and may be altered during the course of the game. |
| Requires | None. |
| Pre-condition | None. |
| Post-condition | None. |
| Side effects | The players cannot influence the indestructible units but the indestructible units can affect the players units or buildings |

### 6.1.9. Network

#### *6.1.9.1.**Starting a Multiplayer game***

| | |
|---|---|
| Function | Starting a Multiplayer game |
| Description | The system shall provide means of hosting a multiplayer game and let other users connect over the internet or Local Area Network. |
| Inputs | Connection requests. |
| Source | Network. |
| Outputs | Game information |
| Action | Host accepts connections until such time that player decide to start game. When game commences no more connections are accepted |
| Requires | Network/internet connection |
| Pre-condition | Player starts hosting a game, i.e. accepting connections |
| Post-condition | Multiplayer game initiates. |
| Side effects | Certain parts of the game are modified versus the single-player version. |

#### *6.1.9.2.**Request Multiplayer team***

| | |
|---|---|
| Function | Request Multiplayer team |
| Description | The system shall provide means of letting the users select teams in multiplayer mode prior to multiplayer game initialization. |
| Inputs | Team number |
| Source | Client |
| Outputs | Team number |
| Action | Client enters a request for a certain team number to the host. Host replies acceptance or non acceptance. |
| Requires | More than two players |
| Pre-condition | Team is empty |
| Post-condition | Client is awarded team number. |
| Side effects | Team is full. Client is not awarded a team number. |

#### *6.1.9.3.**Multiplayer chat***

| | |
|---|---|
| Function | Multiplayer chat |
| Description | The game shall provide means of communication in the form of an instant message chat during multiplayer game. |
| Inputs | Text message.<br>Variable to track whether message is public or intended for team members eyes only. |
| Source | Keyboard |
| Outputs | Text message.<br>Variable to track whether message is public or intended for team members eyes only. |
| Action | Player enters text, chooses destination and sends message. |

| | |
|---|---|
| | Message is received by affected clients and displayed. |
| Requires | Keyboard, initiated multiplayer game. |
| Pre-condition | Multiplayer game is started |
| Post-condition | Message is displayed to the correct clients |
| Side effects | Message is sent to the wrong clients. Strategy revealed. Game over man. |

#### *6.1.9.4.**Multiplayer cheat control***

| | |
|---|---|
| Function | Multiplayer cheat control |
| Description | The system shall only let the users create valid units in a multiplayer game. |
| Inputs | Custom unit list. |
| Source | Client |
| Outputs | Validation variable for the unit configuration. |
| Action | Client sends unit specifications to host for validation. Host validates specification. |
| Requires | Initiated multiplayer game. A unit that conforms to standards set in terms of what kind of technology this unit can wear/use. |
| Pre-condition | Player creates unit |
| Post-condition | Player is allowed to remain in game. |
| Side effects | Player has created an invalid unit. Host disconnects player. |

### 6.1.10. Configuration

#### *6.1.10.1. **Setting video options***

| | |
|---|---|
| Function | Setting video options |
| Description | The user shall be able to set video options in order to match users display capabilities.<br>Modern monitors support a number of different resolutions, but most of them have an optimal resolution where the monitor performs the best. |
| Inputs | Video settings |
| Source | Configuration file |
| Outputs | Configuration file |
| Action | Player enters desired video setting. |
| Requires | Monitor compatible with desired video settings. Settings are tested. Settings are saved. |
| Pre-condition | User enters video settings |
| Post-condition | Video settings are saved and implemented. |
| Side effects | If no resolution has been set a default resolution shall be used |

#### 6.1.10.2. *Setting audio volume*

| | |
|---|---|
| Function | Setting audio volume |
| Description | The user shall be able to set audio volume.<br>As most computers have different models of speakers/headphones their base volume will vary. This as well as the fact that the user will want to vary the game volume depending on his current desires means the in game audio effects and the music soundtrack volume should be able to be set after the users wishes. |
| Inputs | Desired music volume<br>Desired audio effect volume |
| Source | I/O device |
| Outputs | Confirmation message and modification to the configuration file. |
| Action | Player enters desired audio settings. Settings are tested. Settings are saved. |
| Requires | Soundcard |
| Pre-condition | User enters audio settings |
| Post-condition | Audio settings are saved and implemented. |
| Side effects | If no volume settings are defined a default value shall be used. |

#### 6.1.10.3. *Custom soundtrack folder*

| | |
|---|---|
| Function | Custom soundtrack folder |
| Description | The user shall be able to specify a folder on his computer that has custom audio files in it, these shall be played in random order by the system as the in-game soundtrack. |
| Inputs | Folder address |
| Source | I/O device |
| Outputs | Configuration file |
| Action | Player enters folder address containing custom audio files.<br>Game selects random file and plays it. |
| Requires | Music files are in correct format |
| Pre-condition | A correct folder address has been entered. |
| Post-condition | Game plays random music file from folder |
| Side effects | If no directory has been picked the default original soundtrack shall be played in game. |

#### 6.1.10.4. *In-game name*

| | |
|---|---|
| Function | In-game name |
| Description | The user shall be able to choose an in-game name that shall be displayed on every players screen to represent him in chat and other relevant situations. |
| Inputs | Name string |
| Source | I/O device |
| Outputs | Character string variable to the host. |

| | |
|---|---|
| Action | Player enters desired name. Name is checked with server. |
| Requires | None. |
| Pre-condition | A unique name is selected |
| Post-condition | Name is associated with client |
| Side effects | Another client already has said name associated. The system adds a numeric identifier to the end of the name to make it unique. |

### 6.1.11. Ending game

#### *6.1.11.1. Quit the game*

| | |
|---|---|
| Function | Quit the game |
| Description | The user shall be able to at any given time choose to quit the current game in progress and end the system. |
| Inputs | Quit request |
| Source | I/O device |
| Outputs | Connection reset command<br>Game quit command<br>Visual confirmation |
| Action | User is immediately disconnected from any games in progress and the full system shall quit to the original state of the computer as it was before the game started. |
| Pre-condition | Game is running |
| Post-condition | Game is not running |
| Side effects | None |

#### *6.1.11.2. Victorious game by disconnection*

| | |
|---|---|
| Function | Victorious game by disconnection. |
| Description | If all clients are disconnected from the host the system ends the game and declares the host victorious. |
| Inputs | List of client connection status. |
| Source | The game's main system. |
| Outputs | Victory message |
| Action | System displays victory message and ends current game |
| Requires | None. |
| Pre-condition | No multiplayer connections exist |
| Post-condition | Game ends |
| Side effects | None. |

#### *6.1.11.3. Victorious game by mass conquer*

| | |
|---|---|
| Function | Victorious game by mass conquer |
| Description | If only one user team has any buildings or units alive said team has won the game. |
| Inputs | List of victory variables. |
| Source | The game's main system. |
| Outputs | Victory message |
| Action | If the client determines that the list of variables is complete enough for a victory, display the victory message and end the game for the player. Broadcast to host. |
| Requires | None. |
| Pre-condition | Only one teams units and buildings remain |
| Post-condition | Victory is awarded to the only remaining player team. |
| Side effects | The players are returned to the main menu. |

#### *6.1.11.4. **Lost game by disconnection***

| | |
|---|---|
| Function | Lost game by disconnection |
| Description | If a user is disconnected from the game host in a network multiplayer game due to network failure he loses. |
| Inputs | Connection status to host. |
| Source | The game's main system. |
| Outputs | Error message |
| Action | Client loses connection to Host. |
| Requires | None. |
| Pre-condition | Player is disconnected for whatever reason |
| Post-condition | Any remaining units of the player shall be removed from the game.<br>Any remaining participants in the game shall be presented with a status message showing the disconnection of the user.<br>Player shall be returned to the main menu. |
| Side effects | None. |

#### *6.1.11.5. **Lost game by annihilation***

| | |
|---|---|
| Function | Lost game by annihilation |
| Description | If all of a users buildings are destroyed he loses the game. |
| Inputs | Structure list. |
| Source | Structure list. |
| Outputs | Loss message |
| Action | When the user no longer has any buildings left, the client broadcasts this to the host. |
| Requires | |
| Pre-condition | Player has no remaining buildings |
| Post-condition | Any remaining units of the user shall be removed from the game.<br>Any remaining users in the game shall be presented with a status message showing the loss of the player. |
| Side effects | The players system is returned to the main menu. |

## 6.2. Non-functional System requirements

### 6.2.1. Performance requirements

#### *6.2.1.1.* ***Minimum Hardware Specification***

The system shall run with the required performance in a hardware configuration matching the minimum hardware specification. The minimum hardware required is set assuming that no other non-essential background processes are running or interfering with the system. System performance will not be guaranteed if all minimum hardware is not achieved.

*Rationale:* To ensure that the system runs with required performance a minimum level of hardware has to be set. By choosing this level slightly above the level recommended by Sun for the Java Virtual Machine we further ensure our performance will be adequate to meet the required performance.

Requirement: The system shall meet the required performance tests on a hardware configuration that matches the minimum hardware specification.

Dependency:
*Minimum Hardware Specification:*
600MHz CPU or better
392MB RAM or more
OpenGL 2.0-compatible graphics card
OpenAL compatible sound card
Network Interface Card with TCP/IP-support
256kbit/s network connection
Two button mouse or better
Keyboard

#### *6.2.1.2.* ***Required Software Specification***

The system shall run with the required performance in a software environment meeting the required software list. Successful execution of the system will not be guaranteed if all software requirements are not installed.

*Rationale:* The system will need certain software installed to ensure that it can execute and operate properly.

Requirement: The system shall execute in a software environment where all required software is installed correctly.

Dependency:
*Required Software:*
*OS:* Windows XP (SP2 or better) or Windows 2003 or Windows Vista
*Java:* J2SE 5.0 or later
*OpenGL:* 2.0 or later
TCP/IP protocol installed

#### *6.2.1.3.**Space requirements***

Required hard disk space

The system shall be small enough to fit on older computers and not be of inconvenience to the user.

*Rationale*: The system requires a number of files containing graphics and other system related information to be installed. Space is not unlimited and therefore a limit is set to ensure that the system will not inconvenience the user with unreasonable space requirements.

Requirement: The system shall be designed to occupy no more than 100MB of hard drive space excluding the Java Runtime Environment, soundtracks and saved content.

#### *6.2.1.4.**Required available RAM***

The game shall require available RAM during execution to store system state and other essential information. The amount of used RAM will not always be as high as required available. System performance will not be guaranteed if required amount of available RAM is not met.

*Rationale:* The system needs to store a lot of information in the internal RAM memory. To ensure that performance will not be affected by excessive swapping to page file the system will be designed to have a set amount of RAM always available and this amount made a requirement.

Requirement: The system shall at most use 128MB of RAM beyond the usage of the Java Virtual Machine and its support libraries.

### 6.2.2. Efficiency Requirements

#### 6.2.2.1. ***Start-up time***

Starting the game shall take an acceptable amount of time on the minimum level of hardware. The time is measured from when the user has requested initial execution of the system until the next possible user interaction.

*Rationale:* If the user is required to wait more than one minute from starting the system until his first interaction the user will likely be annoyed or assume the system has stalled. To ensure that the user will keep using the system such annoyances needs to be minimal.

Requirement: The system shall take at most one minute on the minimum level of hardware to correctly initialize.

#### *6.2.2.2. Time to generate map*

Generating a random map shall take an acceptable amount of time on the minimum level of hardware. The time is measured from when the user has requested generation of the map until the next possible user interaction.

*Rationale:* If the user is required to wait a longer time on map generation he will likely get impatient and consider quitting the system or never use the random map feature again. To ensure that the user will continue using the system a limit has to be set, and two minutes is the longest acceptable value.

Requirement: The system shall take at most two minutes on the minimum level of hardware to correctly generate a random map.

#### *6.2.2.3. Time to load game*

Loading a previously saved game shall take an acceptable amount of time. A previously saved game contains all the information to restore the system state and resume the game at a later time. The time is measured from when the user has requested the function until the next possible user interaction.

*Rationale:* If the user is required to wait a longer time for the previous game to be restored the user will lose patience and not use the save and load features often. To ensure that the use will continue using the system the acceptable limit has been set to one minute.

Requirement: The system shall take at most one minute on the minimum level of hardware to correctly restore a previous system state and initialize it as a new game.

#### *6.2.2.4. Time to save game*

Saving a currently played game shall take an acceptable amount of time. Saving a game means storing the current system state in such a way that it can be restored at a later time and the game resumed. The time is measured from when the user has requested the function until the next possible user interaction.

*Rationale:* If the user is required to wait a longer time for the game to save the user will not save the game often which might result in the user not returning to the game due to lost games due to crashes or other problems where no saves were done. Ten seconds is seen as an acceptable limit and shall be used.

Requirement: The system shall take at most ten seconds on the minimum level of hardware to correctly store the current system state.

#### *6.2.2.5.Time to pause game*

The pause command shall freeze the current game in an acceptable amount of time. To issue the pause command makes the current game stop its game play, putting it in a dormant state until the user decides to remove the pause. The time is measured from when the user has requested the function until the next possible user interaction.

*Rationale:* If pausing the game took longer than one second some event might occur in the game that requires the user interaction and the user might be annoyed that such things happen as the user is waiting for a pause to occur.

Requirement: The system shall take at most one second on the minimum level of hardware to correctly pause the currently ongoing game.

### 6.2.3. Reliability requirements

#### *6.2.3.1.Failure to launch*

The system shall not fail to launch more than an acceptable amount. A failure to launch is defined as the system never reaching the state of the first user interaction after being initialized by the user. A return to the state before initialization and a freezing of the system on initialization is both considered a failure to launch.

*Rationale:* Whenever the system doesn't launch as expected this causes a big inconvenience for the user as the use cannot be required to understand what went wrong and might not be able to handle the situation that occurs without performing a complete environment shutdown. Therefore a very limiting number is set as maximum of failed launches.

Requirement: The system shall at most one time out of 200 fail to enter a correctly running state after initialization.

#### *6.2.3.2.Unexpected failure*

The system shall not unexpectedly quit or crash due to critical system failures more than acceptable. All forms of unexpected returns to desktop, complete system freezes or environment lock-ups count as an unexpected failure. Upon occurrence of such a failure no information of the current game will be saved.

*Rationale:* If the system suddenly throws an error and freezes or quits the user will be most inconvenienced as there will be no saved data from the system state right before the error occurred. If the errors were persistent and crashes occurred repeatedly the user would stop using the system due to its unreliability.

Requirement: The system shall at most terminate the current state incorrectly one time out of 20 executions of the system.

### 6.2.4. Learnability requirements

#### *6.2.4.1.* ***Ease of learning***

The system shall follow the established standards of other systems in the same genre to ensure that the user is able to learn to use the system quickly. The proper use of the system is defined as being able to end a game successfully in the users favor.

*Rationale*: By using key bindings, controls and other similarities that resemble many of the other games in the same genre the system can ensure that the user feels familiar straight away if he has previous experience in the genre.

Requirements: The system shall be able to be learned to use in two hours if the user has previous experience with similar systems.

### 6.2.5. Usability requirements

#### *6.2.5.1.* ***Command efficiency***

The user shall be able to correctly perform any action available during normal system state with only a few interactions. Any form of usage of mouse and keyboard counts as one interaction, a movement of the mouse pointer from current point to a new point counts as only one interaction.

*Rationale:* To ensure that the game is easy enough to play that the user doesn't find it annoying a limit will be set to any form of interaction during normal gameplay. However, the game should not be too easy to play either as the user will get bored fast if the game is too simple without challenges.

Requirements: The system shall with no more than five correct sequential user interactions support all available user actions during normal system usage.

### 6.2.6. Scalability requirements

#### *6.2.6.1.* ***Hosting limit***

Multiplayer games will be organized based on a modified Client-Server (CS) architecture with a limited number of clients. The modification means that one of the users can act as a server with his own system; this means that any user can at any time start a multiplayer game as a host.

*Rationale:* In CS, players exchange periodic updates through a central server, the game host that is also responsible for resolving any state inconsistencies. The CS architecture is not very scalable with the number of players due to a large bandwidth requirement at the server

and therefore the limit of possible multiplayer users will be set to 8 as to not have unreasonable bandwidth requirements.

Requirements: The system shall limit multiplayer to seven other connected systems to one host user.

#### *6.2.6.2.* ***Hosting bandwidth***

The system shall only require a certain amount of bandwidth for multiplayer games. During multiplayer sessions a lot of control packages and several game states being sent over the network. When bandwidth is limited below the minimum level game performance will deteriorate fast as all users will have to wait on control packages and game states to be fully transferred to all clients from the server.

*Rationale:* For the multiplayer function to work at all the host system needs to be able to transmit partial and full game states as well as control packets constantly to all players. This traffic increases very fast as the amount of remote users rise, thus the host user will have a higher bandwidth requirement. Since any users can be host at any time this requires that all users have a faster bandwidth.

Requirements: The system shall be able to host seven remote users without any loss of game performance within the minimum hardware requirements.

#### *6.2.6.3.* ***Multiplayer synchronization***

There shall be a limit to how often resynchronization events occur. The event occurs whenever a client goes out of synchronization with the server; this means that the client's local copy of the system state is inconsistent with the server's which results in problems with the normal usage. To solve this problem the server transmits a new full or partial system state to the client to try and correct its synchronization error.

*Rationale:* When a client goes out of synchronization with the host a certain amount of data needs to be retransmitted containing the full or partial system state to get the client back in synchronization. If several users go out of synchronization at once due to network congestion or similar issues the amount of new system states the server needs to send out must be limited or it would end up resulting in a denial-of-service situation.

Requirements: The system shall at most try to resynchronize the full game state every 30 seconds with a client during multiplayer sessions should desynchronization occur.

#### *6.2.6.4.**Multiplayer error handling***

The system shall drop any multiplayer clients that are not reachable over the network over a certain amount of time. The host has to constantly send control packages over the network to communicate with all the clients during a multiplayer session. During network failure these packages will get lost resulting in packet loss and desynchronization, the client system is deemed unreachable.

*Rationale*: To prevent game time loss and wait time for bad connections the system should drop desynchronized clients after a certain amount of time without being able to successfully resynchronize them.

Requirements: The system shall disconnect any multiplayer client when more than 15 consecutive seconds of 75% or more packet loss has occurred.

Operational requirements

### 6.2.7. Implementation requirements

#### *6.2.7.1.**Development language***

The game shall be developed in a certain language.

Rationale: In an effort to increase interoperability we have decided to develop this game in Java and OpenGL. This will ensure that future expansion to more supported platforms will be possible.

Requirements: The game shall be written in Java using OpenGL and OpenAL as supportive libraries.

External requirements

### 6.2.8. Safety requirements

#### *6.2.8.1.**Multiplayer security***

The system shall not interpret any packages over the network that are not part of the current multiplayer session. Since the multiplayer sessions use a network for communication non system related packages could be sent with malicious intents to the ports the system listens to.

*Rationale:* Any user would be very upset if the system could compromise the user's environments security so the system must at all times ensure safety. The system should not handle any unknown packages because the packages could be malicious. However, no encryption will be done as this would require too much extra work.

Requirements: The system shall verify that all packets received over the network are proper system packets or simply ignore them.

### 6.3. Use Cases

For Use cases see section 8 - Appendices

# 7. System evolution

## 7.1. Fundamental assumptions

All hardware platforms meeting the minimum hardware requirements will be able to run the required software.

The Java engine will be updated by its developers and patches made readily available for system users, the patches will be backwards compatible with the parts used by the system.

The Windows Operating System will be updated by its developers and patches made readily available for the system users, the patches will be backwards compatible with the parts used by the system.

The required ports for networking through TCP/IP are unrestricted, any firewall, router or other systems that have the ability to block ports will have been properly set up by the user to allow traffic of the system.

## 7.2. Anticipated changes

As the required software gets developed to newer versions the backwards compatibility should still be there, if it is lacking in some area the system uses and it is deemed feasible a minor patch will have to be released to address the issue. However, the support of the system will not cover an unlimited amount of time, at any time further updates of the system can be discontinued without prior notice.

Hardware development will not affect the system negatively in any way in the foreseeable future as the system uses the Java Virtual Environment. This runtime environment keeps the system from direct low-level interaction with the hardware and the Java engine will be updated by its developers to handle changes in the hardware architecture while still maintaining the same environment for the system to run in.

As any system users gets more apt at using the system and use the system for very long times he might feel the system is not balanced in a way to his liking and might want further forms of the system not supported in the original version. To prolong the use of the core system the developers can issue future patches to enhance the system and add more content. The system will also be developed in such a way any user with the required knowledge and intent will be able to modify it to create a custom version of the system with altered forms of usage to further prolong the use.

# 8. Appendices

## 8.1. Summary

### 8.1.1. List of technologies that will be used

- OpenAL
- OpenGL
- Java
- Apache MINA
- Eclipse

### 8.1.2. Short summary of Non-functional requirements

- **Performance** - System performance will not be guaranteed on a system that does not meet the System Requirements
- **Space** - The program shall not require more than 100mb HDD space of itself.
- **Efficiency** - The system shall take at most two minutes to perform any start up or loading action and at most 10 seconds to perform other system menu related action.
- **Reliability** - The system shall not have more than one critical failure per twenty executions
- **Learnability** - The system shall be recognizable by players with past experience of games in the same genre
- **Usability** - The system shall require at most five sequential actions to perform an action during normal game play.
- **Scalability** - The system shall be able to handle up to eight players in a multiplayer game
- **Implementation** - All systems shall be written in the same programming language.
- **Safety** - No harmful data shall be transmitted between systems during multiplayer.

## 8.2. Minimal system requirements

- Windows XP (SP2 or better) or Windows 2003
- Java: J2SE 5.0 or later
- OpenGL: 2.0 or later
- TCP/IP protocol installed
- 600MHz CPU or better
- 392MB RAM or more
- OpenGL 2.0-compatible graphics card
- OpenAL compatible sound card
- Network Interface Card with TCP/IP-support
- 256kbit/s network connection
- Two button mouse or better
- Keyboard

## 8.3. Use Cases

### 8.3.1. UC1: Game player starts the game

**Primary Actor**:
Game player
**Stakeholders and Interests**:
Game player: Wants to get the game up and running.
**Preconditions**:
Game player uses a PC with minimum requirements[7].
Game player runs Microsoft Windows XP as Operating System.
Java 2 Platform Standard Edition 5.0 or later is installed.
Game player has successfully downloaded the game from the Internet or has by some other means ended up with the game installation file on his computer.
Game player has installed the game on his computer.
**Success Guarantee**:
Main game program starts flawlessly.
Game player chooses configuration and successfully starts the chosen type of game.
**Minimum guarantee:**
Game player's computer returns to its previous state in case the success guarantee could not be granted.
**Main Success Scenario:**
1. Game player starts the main game program and comes to first input state.
2. Game player chooses configuration for graphics, a screen name and desired faction[8].
3. Game player chooses one of the integrated soundtracks in the configuration.
4. System logs options from configuration.
5. Game player chooses a map to play on.
6. Game player chooses single-player mode.
7. Game player starts the game-play.

**Extensions:**
2-3a. Game player changes his mind and exits the game.
5-7a. Game player changes his mind and exits the game.
5-7b. Game player loads and resumes a previously saves game.
3b. Game player changes from integrated soundtrack to custom soundtrack.
    1. Game player chooses a custom soundtrack folder from a list over his local hard drives.
6c. Game player chooses to host a game in multiplayer mode.
    1. System waits for a certain amount of time for connections from other game players.
    2. When everyone is connected or time runs out, the game-play starts.
6d. Game player chooses to join a game in multiplayer mode.
    1. Game player enters IP-address of the game host.
        1a. Game is on hold until every player is connected and then game-play starts..
        1b. Game-play starts

**Special Requirements:**
At most 100 sound files will be loaded from the custom soundtrack folder.

**Technology and Data Variations List:**
2b. Only alpha numeric characters in screen name in configuration.
3b1a. Only wav, mp3 and ogg sound formats accepted from the custom soundtrack folder.

---

[7] CPU faster than 600MHz
RAM > 392MB
Free HDD space > 100MB
OpenGL 2.0-compatible graphics card
OpenAL compatible sound card
Network Interface Card with TCP/IP-support
At least 256kbit/s network connection
Two button mouse or better
Keyboard

[8] By choosing a certain faction the player starts with certain basic technology and certain traits.

5b. Map can be either one of the predefined maps or a unique randomly generated map.

Frequency of Occurrence: Whenever the game player is in the mood for game-play.

### 8.3.2. UC2: Game player builds base and army.

**Primary Actor**:

Game player

**Stakeholders and Interests**:

Game player: Wants to build a strong base and army.

**Preconditions**:

Game is started, either in single player mode or multiplayer mode. UC1 was successful.

**Success Guarantee**:

Game player manages to build a strong base.

Game player successfully produce an army.

**Minimum guarantee:**

Game player can at any time exit the game if he finds it hard to understand, boring or don't feel up to it.

**Main Success Scenario:**

1. Game player produces a construction yard.
2. Game player produces a desired building and places it on a valid tile on the map.
3. Game player produces a desired vehicle and it appears by the relevant building in which it was produced.
4. Game player produces a desired unit and it appears by the relevant building in which it was produced.
5. Game player sends units to search for resources and harvesters to harvest resources.
6. Game player chooses new technology to research from a list of possible researchable objects.
7. System provides game player with researched technology.

**Extensions:**

* At any time game player decides to quit the game.

2-6a At any time resources are low and game player needs to harvest more to be able to produce a building or infantry unit or to research new technology.

3b. Desired vehicle cannot be built because the relevant building is not built yet.

1. Game player produces relevant building and places it on a valid tile on the map.
2. Game player produces a desired vehicle and it appears by the relevant building in which it was produced.

4b. Desired infantry unit cannot be built because the relevant building is not built yet.

1. Game player produces relevant building and places it on a valid tile on the map.
2. Game player produces a desired infantry unit and it appears by the relevant building in which it was produced.

4c. Game player designs a custom unit to produce.

1. System verifies that the custom unit is ok.
   1a. System does not verify the custom unit as ok.
   1b. Game player redesigns custom unit.
2. Custom unit appears by the relevant building in which it was produced.

5b. Game player has no units available to search for resources.

1. Game player produces a desired infantry unit and it appears by the relevant building in which it was produced.
2. Game player sends units to search for resources and harvesters to harvest resources.

5c. Game player has not yet produced any harvester.

1. Game player produces a desired vehicle and it appears by the relevant building in which it was produced.
2. Game player sends units to search for resources and harvesters to harvest resources.

**Frequency of Occurrence**: Could be continuous during game-play.

### 8.3.3. UC3: Game player destroys an opponent's unit/building

**Primary Actor**:
Game player
**Stakeholders and Interests**:
Game player: Wants to destroy opponent's unit or building.
Opponent: Wants to destroy game player's unit and not have his own unit/building destroyed.
**Preconditions**:
Game has successfully been started.
Player has at least one opponent.
Enemy units are within game player's visibility range.
**Success Guarantee**:
Opponent's unit/building is destroyed.
**Main Success Scenario:**
1. Game player moves units into attack range of opponent's unit/building.
2. Game player successfully attacks opponent.
3. Game player defeats opponent's unit/building and it is destroyed.

**Extensions:**
* At any time game player decides to quit the game.
1a. Opponent moves outside player's visibility.
1b. Opponent attacks player.
2a. Game player's attack fails.
2b. Opponent attacks player.
3a. Game players unit/building is destroyed.

**Variations List:**
* Opponent is either some other game player or AI (if in multiplayer or single player mode)

**Frequency of Occurrence**: Could be continuous during game-play.

### 8.3.4. UC4: Player defeats an opponent

**Primary Actor**:
Game player
**Stakeholders and Interests**:
Game player: wants to defeat the opponent.
Opponent: Wants not to be defeated.
**Preconditions**:
Game has successfully been started.
Player has one opponent.
Opponent has one unit left.
Enemy units are within game player's visibility range.
**Success Guarantee**:
One player is removed from the game.
**Main Success Scenario:**
1. Game player moves units into attack range.
2. Game player successfully attacks opponent.
3. Game player defeats opponent's last unit and the opponent is destroyed and removed from the game.

**Extensions:**
* At any time game player decides to quit the game.
1-2a. Opponent moves outside player's visibility.
1b. Opponent attacks player.
2b. Game player's attack fails.
2c. Opponent attacks player.
3a. Game players unit is destroyed.

**Variations List:**
* Opponent is either some other game player or AI (if in multiplayer or single player mode)

**Frequency of Occurrence**: Could be continuous during game-play.

### 8.3.5. UC5: Game player wins

**Primary Actor**:
Game player
**Stakeholders and Interests**:
Game player: Wants to win the game.
Opponent: Wants to win the game.
**Preconditions**:
Game has successfully been started.
Player has one opponent.
Player has one unit left.
Opponent has one unit left.
Enemy units are within game player's visibility range.
**Success Guarantee**:
Game ends.
**Main Success Scenario:**
1. Game player moves units into attack range.
2. Game player successfully attacks opponent.
3. Game player defeats opponent's unit and wins the game.

**Extensions:**
* At any time game player decides to quit the game.
1-2a. Opponent moves outside player's visibility.
1b. Opponent attacks player.
2b. Game player's attack fails.
2c. Opponent attacks player.
3a. Game players unit/building is destroyed and he loses the game.

**Variations List:**
* Opponent is either some other game player or AI (if in multiplayer or single player mode)

**Frequency of Occurrence**: Once at the end of the game.

### 8.3.6. UC6 – Player saves game

**Primary Actor**:
Game player
**Stakeholders and Interests**:
Game player: Wants to save the game.
**Preconditions**:
Game has successfully been started.
Game is currently in single player mode.
**Success Guarantee**:
Game is saved.
**Minimum guarantee:**
Game continues.
**Main Success Scenario:**
1. Game player saves the game.
2. Game player resumes the game.

**Extensions:**
1a. Saving the game fails.
    1. Player is returned to the game.

**Frequency of Occurrence**: At any time during game-play.

# 9. Index