

SE497  
Requirements Document

Kaya Oguz  
Serkan Barut  
Sevil Güler  
Sakine Sule Metin

December 13, 2006

# Contents

<b>1</b>	<b>About The Project</b>	<b>4</b>
1.1	1453 . . . . .	4
1.2	Development Platforms . . . . .	4
<b>2</b>	<b>Requirements Of A Game</b>	<b>6</b>
2.1	Game Development Library . . . . .	6
2.2	Story . . . . .	6
2.3	Graphics . . . . .	7
2.4	Sound and Music . . . . .	7
2.5	Documentation . . . . .	7
2.6	Technical Requirements . . . . .	8
2.6.1	Screens . . . . .	8
2.6.2	Graphical User Interface . . . . .	8
2.6.3	Levels . . . . .	8
2.6.4	Sprites and Animations . . . . .	9
2.6.5	Game Loop . . . . .	9
2.6.6	Settings and Preferences . . . . .	9
2.6.7	Saving and Loading . . . . .	9
2.6.8	Artificial Intelligence . . . . .	10
2.6.9	Game Play . . . . .	10
2.6.10	Physics . . . . .	10
2.6.11	Multiplayer Games . . . . .	10
2.6.12	Winning and Losing . . . . .	10
2.6.13	Scoring . . . . .	11

<b>3</b>	<b>Our Project Requirements</b>	<b>12</b>
3.1	Game Development Library: SDL . . . . .	12
3.2	Story . . . . .	13
3.3	Graphics . . . . .	13
3.4	Sound and Music . . . . .	13
3.5	Documentation . . . . .	14
3.6	Technical Requirements . . . . .	14
3.6.1	Screens and Graphical User Interfaces . . . . .	14
3.6.2	Levels: Scenarios and Maps . . . . .	14
3.6.3	Sprites and Animations . . . . .	14
3.6.4	Game Loop . . . . .	15
3.6.5	Settings and Preferences . . . . .	15
3.6.6	Saving and Loading . . . . .	15
3.6.7	Artificial Intelligence . . . . .	15
3.6.8	Game Play: Real Time . . . . .	16
3.6.9	Winning, Losing and Scoring . . . . .	16
<b>4</b>	<b>Playing 1453</b>	<b>17</b>
4.1	Non-playing Screens . . . . .	17
4.1.1	Main Screen . . . . .	17
4.1.2	New Game and Scenarios . . . . .	19
4.1.3	Load Game Screen . . . . .	19
4.1.4	Settings Screen . . . . .	21
4.1.5	Credits Screen . . . . .	21
4.2	Game Playing Screen . . . . .	21
<b>5</b>	<b>Conclusion</b>	<b>24</b>

# List of Figures

4.1	Main Screen Of The Game . . . . .	18
4.2	A scenario screen . . . . .	19
4.3	Load Game Screen . . . . .	20
4.4	Game Settings Screen . . . . .	21
4.5	Credits Screen Scrolling . . . . .	22
4.6	Game Play Screen . . . . .	23

# Chapter 1

## About The Project

### 1.1 1453

1453 is a Real Time Strategy Game about the first centuries of the Ottoman Empire. It will be written in C++ using various libraries starting with SDL<sup>1</sup>: Simple DirectMedia Layer. The game will run on both Windows and Linux systems. Here are some planned features of the game:

- 2D Single Player Game
- Scenarios and Missions revolving around the new Ottoman Empire
- Original Graphics, Music and Sounds
- Completely in Turkish (troops will speak Turkish)
- A Challenging Artificial Intelligence

More features can be added during the project.

### 1.2 Development Platforms

The game will be developed and tested on both Linux and Windows systems. We are planning to use a generalized build system called GNU

---

<sup>1</sup><http://www.libsdl.org/>

Make. On Windows systems we will be using Bloodshed DevC++<sup>2</sup> and on Linux we are planning to use Eclipse<sup>3</sup> IDE with C/C++ Development Tools<sup>4</sup>.

The game will be written in C++ around the popular and stable multimedia library SDL. SDL is a mature and commercial quality multimedia library which allows easy access to the multimedia capabilities of a computer. Along with SDL we will be using some derived libraries such as SDL\_image, SDL\_ttf and SDL\_mixer which we will discuss through out this document<sup>5</sup>.

---

<sup>2</sup><http://www.bloodshed.net/devcpp.html>

<sup>3</sup><http://www.eclipse.org/>

<sup>4</sup><http://www.eclipse.org/cdt/>

<sup>5</sup>Libraries are available at SDL home page

# Chapter 2

## Requirements Of A Game

Games have a lot in common. Since the day we have our project we researched in detail what we will need and how we will manage to develop the game. It will give a more broad vision if we discuss these common traits of the games.

### 2.1 Game Development Library

Most games today use DirectX as the games are targeted at the popular Windows platform. There are many game development libraries out there such as SDL, ClanLib, etc. These libraries make the game development easier and save the programmers a lot of time as they don't write the same code all over again. Also the code becomes more mature as more games use it. The bugs are fixed and more features are implemented. This is why a game development library is the first place to start for developing a game.

### 2.2 Story

No game is fun when you do not know why you are playing it. You may be saving the world, or trying to save yourself in a story of the game. This gives the player a chance to put himself into the scenario and give him a

virtual world to have fun. Some of the games have such good stories that they are made into a movie; like Resident Evil, Silent Hill, etc.

## **2.3 Graphics**

Every game has an interface and graphics which suits its own theme. The graphics are done with professional artists. Some major games go through this step like a movie. They have design themes which work on the game art and develop digital images according to it. The games may have their own design of weapons, buildings, aliens and etc.

## **2.4 Sound and Music**

The sound and the music are the other important parts of a game. Mostly original music is composed and recorded for the games. Some game musics have their own soundtracks. The in-game sounds, like explosions, ambients sounds are also recorded specially for the game. The voice actors are also important as their acting give depth and reality to the game. Some popular games have popular artists do the voice acting, especially if the game is developed after a movie. The movie's leading stars mostly do the voice acting of their computer game parts.

## **2.5 Documentation**

Just like every software, the game needs a good documentation which describes general game play, game rules and alike. The documentation is mostly a simple booklet which comes with the game. It may also include a trouble-shooting guide.

## **2.6 Technical Requirements**

These technical requirements involve the programming requirements. Not every game may have all these requirements, but most of them have a lot of them.

### **2.6.1 Screens**

Even the simplest game have a lot of screens. Screens are like windows on a desktop program. When you first run a game you see a welcome screen. After clicking, let's say "Options", you go to the Options screen. Very much like opening a new window but of course the illusion is different.

### **2.6.2 Graphical User Interface**

Screens will need some widgets on them to operate. Using operating system widgets in a game makes the game boring. Mostly developers roll their own widgets and give the game a unique interface. Mostly these widgets are common widgets from our desktops but they look better, they are styled and have cool effects on them.

### **2.6.3 Levels**

Each game progresses as the player keeps on playing. This brings up the concept of "levels". Player can see his progress with these levels. Levels mostly have increasing difficulty, which makes the Artificial Intelligence stronger, or the number of enemies higher.

Levels are sometimes considered as new areas of playing, like new maps for some games. Having different maps means the game should read a pre-written file and load it. Some games have auto-generating maps while some have stored maps.

## **2.6.4 Sprites and Animations**

In a 2D game most of the animations are done with sprites. Sprites are the frames of an animation which is all in one image. The sprites are mostly generated by some specialized software, like a 3D renderer. Animations are achieved with displaying the certain part of the sprite, giving the illusion of movement.

## **2.6.5 Game Loop**

This is the infinite loop where the game is run until the game is exited or the game has come to an end. This infinite loop checks for user input, update game status and draw the game on screen. It's one of the most important parts of the game as this loop has to be flawless.

## **2.6.6 Settings and Preferences**

Settings are the choices for the general running of a software, where Preferences are the choices of the user, like how he wants to customize the software. Games use Settings for Video Modes, Color Depths and etc. Preferences are mostly used for customizing keyboard shortcuts, mouse speed etc. The real problem here is where and how to save and load these settings when you are developing a multi-platform application.

## **2.6.7 Saving and Loading**

In some games, you are given a tip to save often. Saving a game is recording the state of the game to a file to load it at a later time. Some games let you save at certain points through the game, some give you a limited number of saves. Saving and loading require some binary and encrypted format to stop the player to edit the save file and cheat.

### **2.6.8 Artificial Intelligence**

Artificial Intelligence can be as simple as a random number generator, or a complicated neural network. In most of the games, if it's not a board game, competing with an artificial intelligence is not easy. The computer is mostly accurate in its calculations, so in some games they make the artificial intelligence a less dangerous competitor with randomizing its actions. Artificial Intelligence is one of the key elements of a game as it gives the feeling of facing a real-life opponent.

### **2.6.9 Game Play**

The game play is how the game is played; whether it is a turn based game or a real-time game. In fact all games are turn-based. When it's real-time each second there are so many turns that it gives an illusion of real-time.

### **2.6.10 Physics**

The game engines of some games, like car racing, sports and etc. need to have a very good physics engine to make the player feel like what is happening on the screen is real. In simpler games, just an acceleration may be implemented for giving a better movement feeling.

### **2.6.11 Multiplayer Games**

Some games follow the idea of a real opponent instead of an artificial intelligence. Even the simplest games are more fun when they are played in multiplayer mode. This is because your opponents is a friend of yours and nothing is more fun than crashing your friend's car, for example in a car racing game.

### **2.6.12 Winning and Losing**

It is no good if you do not have the chance to win or lose. The game should have a mechanism for winning and losing conditions. This may be a part

of a scenario, or a mission in a scenario, or just completing a simple level.

### **2.6.13 Scoring**

This is also as important as Winning and Losing. Even if you have lost, you may have beaten the top score for the game, which feels almost as good as winning.

# Chapter 3

## Our Project Requirements

As we have discussed what a game needs, we can discuss what we have chosen and why we did choose them.

### 3.1 Game Development Library: SDL

What we needed was something simple, multi-platform, well-documented, easy to develop and easy to manipulate and fit our needs.

SDL is a library written in C. It runs with C++ natively which allows us to derive our own classes to encapsulate SDL functions written in C. It already has basic access for graphics, sound and input. It's just a few lines of code to create a window (screen), and a few more to display an image on the window. The image can be loaded from a bitmap or SDL can draw it pixel by pixel. It has an event loop which has a queue for keyboard, mouse and joystick events. It also supports playing uncompressed wave and PCM files.

What we needed was to get more out of SDL with libraries which are mature like SDL itself. With `SDL_image` we can use common image formats like PNG, GIF and JPG. With `SDL_mixer` we can use common sound formats like MP3 and OGG. With `SDL_ttf` we can use True Type Fonts to write on screen. With all these extra libraries we can manage most of what we need.

SDL is a 2D library, although it supports 3D with OpenGL. Our game will be a 2D game, so we will not use OpenGL or another 3D library. This means that we will need a GUI system for our game. There are a few GUI libraries written for SDL but we could not find one which suits our needs. Most of them want to take control of the game loop as they will need to check for events. So we decided to write our own GUI.

## **3.2 Story**

The game is about the Ottoman Empire, so we are thinking of commanders who will be in charge of some Ottoman troops while important events are taking place in the empire at the background. The game will probably have small short stories during scenarios and missions.

## **3.3 Graphics**

As we stated in our previous report, we are getting professional help in graphics so that we can concentrate on developing the game engine. Beyaz Kare<sup>1</sup> will be producing our game art and graphics.

## **3.4 Sound and Music**

We will have original music composed for the game. Semih Önyer and Özcan Oguz, who are both professional musicians, will compose music for the game.

The sounds will be recorded in a studio with probably more friends acting in different roles. The ambient sounds will either be recorded, or will be found on the internet.

---

<sup>1</sup><http://www.beyazkare.biz/>

## **3.5 Documentation**

With project reports we already have some of the project documented. We also run a website for developers<sup>2</sup>. The final game will probably have a manual which will explain the game play and rules.

## **3.6 Technical Requirements**

We won't implement all of the requirements listed above as they will not be needed by our game. Some of the requirements have not yet been decided as we do not have experience on game developing. They will be decided during the development of the game.

### **3.6.1 Screens and Graphical User Interfaces**

We will have a GUI system which will have screens and widgets (buttons, labels, etc). These will be implemented with basic SDL functions and will have states, binded functions and will give the user a similar experience.

The UI will be used on both screens and during the game, on top of the map, as an interface to unit and building commands.

### **3.6.2 Levels: Scenarios and Maps**

The single player game will have scenarios which are related with certain maps. We will develop a format for both of these structures.

### **3.6.3 Sprites and Animations**

The graphics we will have from Beyaz Kare will be in a sprite sheet. We will probably have a sprite class for each animated unit or image in the game. This is will be in SDL, too.

---

<sup>2</sup><http://bindortyuzelliuc.wordpress.com/>

### **3.6.4 Game Loop**

The game loop is one of the important parts of the game. As an RTS game, we have a map which is larger than the screen and we will have a view port where shows the part of the map which the user is playing. So in the game loop:

- We will draw the map and everything on it (units, buildings, etc.) which is in the view port
- We will draw the UI on top of it
- We will check for user input
- We will process the user input and
- We will let the game's AI make its move

### **3.6.5 Settings and Preferences**

In a multi-platform application we need to support every platform, we will also develop a format for saving and setting preferences in the game.

### **3.6.6 Saving and Loading**

We have not yet decided yet whether to save the game at each scenario or during the gameplay. We will need to develop a format for saving and loading the game which supports both platforms.

### **3.6.7 Artificial Intelligence**

This is crucial for our game as it will serve as the opponent for the player. We are working on neural networks, fuzzy logic and other artificial intelligence topics to get it right. There will also be AI for the player's units. The units will choose how to behave when certain conditions occur; when enemy approaches to him, he decided whether to attack or go ask for help.

### **3.6.8 Game Play: Real Time**

We have chosen this right at the beginning. The game will be a real time game, instead of a turn based game.

### **3.6.9 Winning, Losing and Scoring**

The game will be complete when all scenarios are beaten. We did not yet think about a ranking system.

# Chapter 4

## Playing 1453

This chapter has mock-up screens for the game. They are subject to change and the look will be stylized and more graphical. These are just templates. Some of the images on the screens are from different sources. They will all be changed in the final game, with our own graphics and images.

The images are put in figures and they can be seen in following pages if not under the title.

The widgets, buttons, labels, list-views, scrollbars etc. will be of our own GUI system. They will be implemented using raw SDL functions.

### 4.1 Non-playing Screens

These screens involve the screens where the game loop is not running.

#### 4.1.1 Main Screen

This is the screen where the user has the options to start a new game, load a previous game, change the settings, see credits or simply quit the game. This is the initial screen of the game.

# 1453



- Yeni Oyun
- Oyun Yükle
- Ayarlar
- Emeği Geçenler
- Çıkış

Figure 4.1: Main Screen Of The Game

# 1453



Figure 4.2: A scenario screen

## 4.1.2 New Game and Scenarios

These are the screens when new game is selected and the first scenario appears. The scenario screen has a mission briefing, missions to complete and start the game. A button also enables the player to return to the main screen. The scenario screens will load scenario details and start the game, so a similar screen will be used for each scenario with new data.

## 4.1.3 Load Game Screen

This is the screen when selected from the main menu. Because we have not yet decided how and where to save the game, we are planning that the saving screen will also be similar to this one.

# 1453



Figure 4.3: Load Game Screen

# 1453

## Oyun Ayarları

Oyun Zorluk Derecesi

Fare Kaydırma Hızı



Kaydet

Ana Menü

Figure 4.4: Game Settings Screen

#### 4.1.4 Settings Screen

These are the settings for the game play. There may be more options, or these options can be changed.

#### 4.1.5 Credits Screen

This screen shows the people who have contributed to the project.

### 4.2 Game Playing Screen

This is the screen while the game is played. There is a bar on the right which includes a mini map of the map. The mini map also shows which part the view port currently shows.

# 1453



Emeđi Geenler

Yazılım:  
Kaya Ođuz  
Serkan Barut  
Sevil Gler  
Sakine Őule Metin

Grafik:  
Melih nyer (Beyaz Kare)

Ana Menu

Figure 4.5: Credits Screen Scrolling

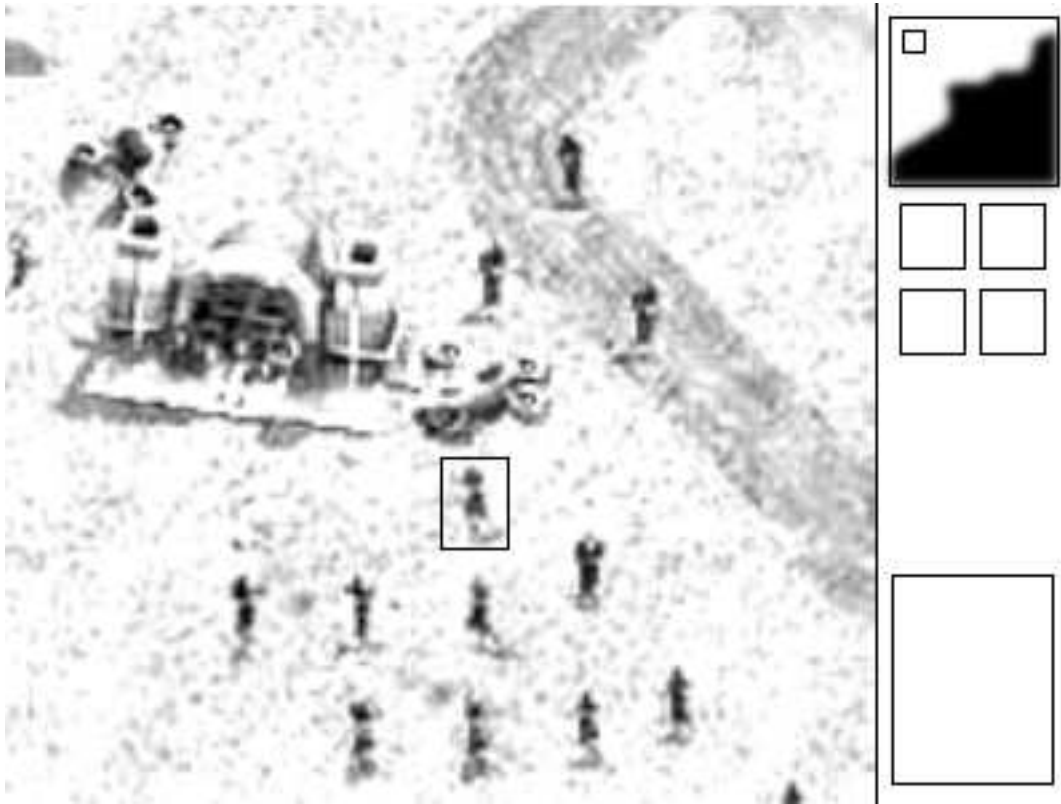


Figure 4.6: Game Play Screen

When a unit or a building is selected its commands are displayed under the mini map.

Under these commands, right at the bottom, there is an info panel which tells about the current status of the player's game, like his amount of resources etc.

# Chapter 5

## Conclusion

The game development has become very clear. We will be using tools and libraries which will make the development of the game less painful. Every decision we have had are subject to change as we are all unexperienced in developing a game.