

SE497
Project Plan

Kaya Oguz
Serkan Barut
Sevil Güler
Sakine Sule Metin

November 23, 2006

Contents

1	Overview	3
1.1	Overview	3
1.2	About The Game	4
2	Research Phases	5
2.1	Research Phase	5
2.1.1	Ottoman & Byzantine History	5
2.1.2	Real Time Strategy Games	7
2.1.3	Game Types	7
2.1.4	Programming Languages and Game Libraries	8
2.1.5	Development Environments	9
2.1.6	Artificial Intelligence	9
2.2	Getting Experience	9
3	Implementation Phases	10
3.1	Design Phase	10
3.2	Coding Phase	11
3.3	Testing Phase	11
4	Project Staffing	13
4.1	Software Development Staff	13
4.2	Game Graphics	14
4.3	Game Sounds & Music	14

5	Software Engineering Methods	15
5.1	Coding Standard Verification	15
5.2	Code Reviews	15
5.3	Coding Standard	15
5.4	Object Oriented Design	16
5.5	Software Life Cycle	16
6	Project Risks	17
6.1	The Risk Of Inexperience	17
6.2	Technical Risks	17
6.3	The Risk Of Scheduling	18
6.4	Quality Risk	18
7	Project Tools	19
7.1	Google Code Tools	19
7.1.1	Subversion Repository	19
7.1.2	Bug Tracker	20
7.1.3	E-Mail Lists	20
7.2	Software Tools	20
7.2.1	GNU GCC and MinGW	20
7.2.2	Bloodshed DevC++	20
7.2.3	Eclipse	21
7.2.4	SVN & TortoiseSVN	21
7.2.5	SDL	21

Chapter 1

Overview

1.1 Overview

This report is the *Project Plan* of the Real Time Strategy Game that we are going to develop in our Senior Project course. The project will last for two semesters.

This project will be developed by:

- Kaya Oguz
- Serkan Barut
- Sevil Güler
- Sakine Sule Metin

The game will be a 2D single player game. It will have scenarios inspired by the first centuries of the Ottoman Empire, until the Conquest of Istanbul in 1453, hence the name of the game 1453. The game will run on both Windows and Linux platforms.

We are planning to use the C++ programming language since it is fast, flexible and well supported. We will be using Bloodshed DevC++ IDE on

Windows and Eclipse IDE on Linux. On both platforms we will be using GNU GCC as compiler set. Along with GCC, we will be using SDL, Simple DirectMedia Layer, for graphics and sound of the game. SDL provides low level access to keyboard, audio, mouse and 2D video framebuffer.

1.2 About The Game

We are planning to call the game “1453” as it is an important year for both the world and the Ottoman Empire. The game will have a Turkish User Interface. The units in the game will also speak Turkish, thus giving the user a unique experience. For the music and graphics we will have outside help and will work with professionals.

We are planning to have sequential scenarios from the first years of Ottoman Empire as they take cities from Byzantine Empire one by one. We will focus on the conquest of Bursa, Edirne and Istanbul. During these conquests the player will play different missions to help the Ottoman Empire expand.

Chapter 2

Research Phases

2.1 Research Phase

Writing a game involves a lot of different skills. Before we start, we plan to research in areas which we do not feel comfortable or want to learn more.

We will be researching on the following topics:

- History of both Ottoman and Byzantine Empires
- Real Time Strategy Games
- Game Types (Single-player and Multi-player)
- Programming Languages and Game Libraries
- Development Environments
- Artificial Intelligence

2.1.1 Ottoman & Byzantine History

The Ottoman state began as one of many small Turkish states that emerged in Asia Minor during the breakdown of the empire of the Selçuk Turks.

The early phase of Ottoman expansion took place under Osman I, Orhan, Murad I, and Beyazid I at the expense of the Byzantine Empire, Bulgaria, and Serbia. Bursa fell in 1326 and Adrianople (the modern Edirne) in 1361; each in turn became the capital of the empire. The great Ottoman victories of Kosova (1389) and Nikopol (1396) placed large parts of the Balkan Peninsula under Ottoman rule and awake Europe to the Ottoman danger. The Ottoman siege of Constantinople was lifted at the appearance of Timur, who defeated and captured Beyazid in 1402. The Ottomans, however, soon rallied.

Byzantine Empire, successor state to the Roman Empire, also called Eastern Empire and East Roman Empire. It was named after Byzantium, which Emperor Constantine I rebuilt (A.D. 330) as Constantinople and made the capital of the entire Roman Empire. Although not foreseen at the time, a division into Eastern and Western empires became permanent after the accession (395) of Honorius in the West and Arcadius in the East. Throughout its existence the Byzantine Empire was subject to important changes in its boundaries. The core of the empire consisted of the Balkan Peninsula (i.e., Thrace, Macedonia, Epirus, Greece proper, the Greek isles, and Illyria) and of Asia Minor (present-day Turkey). The empire combined Roman political tradition, Hellenic culture, and Christian beliefs. Greek was the prevalent language, but Latin long continued in official use.

Both empires also have distinct armies with different types of soldiers. Ottoman Empire is clearly one of the strongest armies of its time, but to make the game fair, we will try to balance both sides.

2.1.2 Real Time Strategy Games

We will have to research and play a lot of RTS games like Age of Empires¹, Starcraft², Warcraft³ so that we can see the main characteristics of RTS games and to find common properties among them. We think that these properties will help us to determine rules and properties of our project.

We know that all games are either in 2D or 3D. The two-dimensional image is not just a representation of a real-world object, but also an independent artifact with added semantic value; two-dimensional models are preferred because they give more direct control of the image than 3D computer graphics.

3D are works of graphic art that were created with the aid of digital computers and specialized 3D software. The art of 3D modeling prepares geometric data for 3D computer graphics, while the art of 2D graphics is analogous to painting. However, 3D computer graphics rely on many of the same algorithms as 2D computer graphics. One advantage of full 3D models is easy animation: each animation frame is just a set of object-vertex positions. In most 2D games, frames are separate pictures, which takes more memory and disk space.

2.1.3 Game Types

There are two types of games: single-player and multi-player. Single-player refers to the variant of a particular game where input from only one player is expected throughout the course of the gaming session whereas in Multiplayer multiple people can play the same game at the same time. Single-player games mostly have a scenario or story built around, while popular multi-player games is usually about eliminating the opponent.

¹<http://www.microsoft.com/games/empires/>

²<http://www.blizzard.com/starcraft/>

³<http://www.blizzard.com/war3/>

Of course there are a lot of multi-player games with a story and which requires the cooperation of players.

2.1.4 Programming Languages and Game Libraries

The selection of the programming language is very important as the game should have certain properties like stability, speed and modularity. Most common language for game developing is C++, while higher level languages like Java, C# and Python etc. are becoming popular. The choice of the language is also bound to the game library. Although the libraries have bindings to several languages, it would be nice to choose a language and library which goes better together.

There are so much libraries for game programming. One of the most used library is DirectX⁴ which is a collection of APIs (Application Programming Interface) for handling tasks related to multimedia, especially game programming, on Microsoft platforms. It is widely used in the development of computer games for Windows, the Xbox (sixth generation era video game console produced by Microsoft,) and Xbox 360 (is the successor to Microsoft's Xbox video game console, developed in co-operation with IBM, ATI, Samsung and SiS).

Microsoft also released XNA Game Studio⁵, which runs on top of Visual Studio C# Express and enables users to develop Windows or Xbox games.

Another one is SDL⁶ (Simple DirectMedia Layer) which is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer. It is used by MPEG playback software, emulators, and many popular games. SDL supports many operating systems. (Linux, Windows, Win-

⁴<http://www.microsoft.com/directx/>

⁵<http://msdn.microsoft.com/directx/xna/>

⁶<http://www.libsdl.org/>

dows CE, BeOS, MacOS, Mac OS X etc.). Many game programmers use SDL because they found SDL quite easy to use and it supports many operating systems.

2.1.5 Development Environments

For a project like this, we will need a development environment for the upkeep of many files and libraries. There are many Integrated Development Environments (IDE) either commercial or free. This is also a little bit bound to the language and library selection. Among popular IDEs, there are Eclipse, Visual Studio, DevC++, CodeBlocks etc. We will probably choose one which is cross platform.

2.1.6 Artificial Intelligence

Even a simple game uses Artificial Intelligence (AI) in several parts of it. The simplest AI is done by random numbers. Other than that we will need popular algorithms for problems like shortest paths etc. There will also be an AI for the CPU player, which will play against the player.

2.2 Getting Experience

A Real Time Strategy game is not a good place to start writing a game for the absolute beginners like us. So, while we are researching, we will also try some libraries and develop simple games like Tic-Tac-Toe, Tetris, Pong etc. These games will give us clues while we will be designing the game and its flow.

Chapter 3

Implementation Phases

The implementation phases will have the basic Software Development Life Cycle. We will cycle through the steps until we reach our goal. The cycle will most probably take place in Coding and Testing Phases.

3.1 Design Phase

The design phase is the most important phase since it forms the backbone of the whole project. All types of failures, made during the design phase, will effect other phases.

This phase includes to find out the relationships between different classes and objects and how they will affect each other. Therefore it requires high attention during the design phase and not make many failures as it can be necessary to start the game from scratch.

While designing the game we will focus on making the whole flow and schema of the game:

- Requirements Analysis
- Creating Classes

- UML Schemas, E-R Diagrams etc.
- Process Schemas

During the design phase we will also use the information we have gathered from the research we have done so far. This will involve the historical background and game programming techniques we will have, by that time, learned.

3.2 Coding Phase

During the Coding Phase all schemas will be translated into codes. We will first implement the classes which came out in the design phase. If we need more, or don't need some, we will go back to the design phase and manipulate the design.

The coding phase will take most of our time as we will need a lot of issues to track of. During this phase we will use common programming tools like Subversion, so that all developers will be able to work simultaneously.

Since our game will be a single player game we will be needing AI as the opponent of the player. Aside from AI we will need Display and Sound Managers and a very stable game engine to keep the game going.

3.3 Testing Phase

A game is a very important software. If it has bugs or flaws, the players can give up quickly. So testing phase is as crucial as the other phases.

After the Coding has finished we have to test our game. Probably we will have many lines of codes. There will probably be a lot of bugs. So we have to test the functionality of the game, whether it runs smoothly and crashes. During this phase we will many times have to go back to the

coding phase. We will also test the playability and the balance of the game in this phase.

Chapter 4

Project Staffing

Current game market is very similar to that of movies. The graphics, music, sounds, user interface and game engine itself all belong to different types of professions. So, we have decided to also ask for outside help for those parts that we have no experience or training.

We also need help for the testing phase. Testing the project on four computers is not enough. So our friends from the faculty and families will probably play the game while it is in development phase and will contribute ideas and issues.

4.1 Software Development Staff

The software development staff are the members of this project:

- Kaya Oguz
- Serkan Barut
- Sevil Güler
- Sakine Sule Metin

As the game is a big project, all of us will contribute to the development of the software, which includes the research, design, coding and testing phases.

4.2 Game Graphics

The project members have neither experience nor training in graphics. This part of the game has an artistic style, so we decided to leave it to a professional friend of ours. Melih Önyer of Beyaz Kare Görsel Sanatlar¹ will be making our graphics (units, buildings, environment) for our game. He will also probably draw our User Interface Widgets (buttons, labels etc). He also will make a cover and logo for the project. We already appreciate his help.

4.3 Game Sounds & Music

In every strategy game you can hear the environmental sounds and units talking, walking or working. For the sounds of the units the project members and our volunteering friends will help us to read lines. We are planning to hire a studio to record the sounds clearly.

Another important part of a game, like graphics, is the music, which plays during the game or during game menus. For this, we will also ask the help of two professionals; Semih Önyer, a student of Composition and Conduction Main Branch in the Department of Music in Dokuz Eylül University School of State Conservatory, and Özcan Oguz, a long time freelance musician. They both will be composing music for the game.

¹<http://www.beyazkare.biz/>

Chapter 5

Software Engineering Methods

This section will provide an overview of what methods we will use to create our project and how we will perform quality assurance testing.

5.1 Coding Standard Verification

We will perform code checks to verify that the code meets the coding standard and catalog and fix any standards violations.

5.2 Code Reviews

As a team, we will meet every other week to conduct code reviews.

5.3 Coding Standard

We will follow a coding standard for this project that will increase readability and maintainability.

5.4 Object Oriented Design

We will use the object-oriented method for program design and keep this design updated with fixes for any potential problems we may catch. We will also perform design reviews throughout the project to ensure that the existing design will meet the criteria for the software.

5.5 Software Life Cycle

The period of time that begins when a software product is conceived and ends when the software is no longer used. The software life cycle typically includes concept, requirements, design, implementation, test, installation and checkout, operation and maintenance, and, sometimes, retirement. These phases may overlap or be performed iteratively.

We will use SLC method to give the best results from our project and finishing it on time.

Chapter 6

Project Risks

Game development has many risks, most of which can not be foreseen.

6.1 The Risk Of Inexperience

None of the project members have any experience in game programming, graphics or AI. The team will also be using new tools and will learning during production.

6.2 Technical Risks

There are different types of technical risks. During the development of a game, it is very important to use the right tools, since these tools enable the speed and flexibility. In addition, incorrect programming of the artificial intelligence may result in an uninteresting game. Another problem can occur if the speed of the game is not appropriate for many systems.

6.3 The Risk Of Scheduling

Game development requires a lot of time. Since it is our first game we have no experience about how long it will take to do every single process of developing a game. That is why we have to manage our time very carefully if we are to finish the game in time.

6.4 Quality Risk

Developing a game is not a trivial task. It is made up of a lot of parts, which are meant to work with each other smoothly. Each part is made up of a lot of lines of code. This increases the number of bugs. The time required to remove these bugs are important as they define the quality of the software.

Chapter 7

Project Tools

We plan to use the following tools and services:

7.1 Google Code Tools

Google Code¹ is a service provided by Google and is made up of a SVN server and a Bug Tracker. It offers a lot less than Sourceforge, which provides similar services, but it has two basic things we desperately need. Using Google Code Project Hosting is free and it links with our e-mail list, which is also a Google service. Our Google Project web address is:

<http://code.google.com/p/project1453/>

7.1.1 Subversion Repository

The goal of the Subversion project is to build a version control system that is a compelling replacement for CVS in the open source community. Google provides our project a SVN repository at the following address:

<http://project1453.googlecode.com/svn/>

¹<http://code.google.com/>

7.1.2 Bug Tracker

A bug tracker is a tracking system that is designed especially to manage problems (software bugs) with computer programs. Google Code provides us a simple, yet powerful “Issues” section to keep track of our bugs.

7.1.3 E-Mail Lists

An e-mail list is being used to communicate among the project members.

7.2 Software Tools

7.2.1 GNU GCC and MinGW

GCC², GNU Compiler Collection, includes front ends for C, C++ (and many more) as well as libraries for these languages. MinGW³ is a collection of Windows specific header files and import libraries combined with GNU toolsets that allow one to produce native Windows programs that do not rely on any 3rd party C runtime DLLs.

7.2.2 Bloodshed DevC++⁴

A free C++ IDE for Windows which uses the MinGW compiler. DevC++ has many advantages, like code completion, package server and a small memory footprint.

²<http://gcc.gnu.org/>

³<http://www.mingw.org/>

⁴<http://www.bloodshed.net/devcpp.html>

7.2.3 Eclipse

Eclipse⁵ is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. We will use C/C++ Development Tools extension (CDT⁶) for using Eclipse in a C/C++ development environment.

7.2.4 SVN & TortoiseSVN

TortoiseSVN⁷ is an easy to use SCM / source control software for Microsoft Windows. It is implemented as a Windows shell extension, which makes it integrate seamlessly into the Windows explorer. On linux we will be using the svn command for SVN⁸.

7.2.5 SDL

SDL (Simple DirectMedia Layer) is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer. It is used by MPEG playback software, emulators, and many popular games, including the award winning Linux port of "Civilization: Call To Power."

⁵<http://www.eclipse.org/>

⁶<http://www.eclipse.org/cdt/>

⁷<http://tortoisesvn.tigris.org/>

⁸<http://subversion.tigris.org/>