CENG 491

Fall 2007 - SpongeSoft Initial Design Report

Online Virtual Team Collaboration Platform with 3D Graphics

Duygu ATILGAN 1394667
Nilgün DAĞ 1394857
Aslı ÖZAL 1514868
Bahadır ÖZDEMİR 1395342

December 3, 2007
# Table of Contents

1. Introduction ........................................................................................................................................... 5  
   1.1. Project Description ......................................................................................................................... 5  
   1.2. Project Features .............................................................................................................................. 5  
   1.3. Purpose of Document ..................................................................................................................... 5  
2. Design Considerations .......................................................................................................................... 6  
   2.1. Design Constraints .......................................................................................................................... 6  
   2.1.1. Scheduling ................................................................................................................................. 6  
   2.1.2. Performance ............................................................................................................................... 6  
   2.2. Design Goals & Objectives ............................................................................................................. 6  
   2.2.1. Portability ..................................................................................................................................... 6  
   2.2.2. User Interactivity ......................................................................................................................... 6  
   2.2.3. Reliability ..................................................................................................................................... 6  
   2.2.4. Realistic Environment ................................................................................................................. 6  
   2.3. Development Tools and External Resources ................................................................................ 7  
   2.3.1. Software Choices ......................................................................................................................... 7  
3. Game Content ......................................................................................................................................... 8  
   3.1. General Scenario ............................................................................................................................... 8  
   3.2. 3D Environment of the Game ......................................................................................................... 10  
   3.2.1. Buildings ..................................................................................................................................... 10  
   3.2.2. Fires ............................................................................................................................................... 11  
   3.2.3. Civilians ....................................................................................................................................... 12  
   3.3. Players and Their Tasks .................................................................................................................. 13  
   3.3.1. Search and Rescue Unit Chief ................................................................................................. 13  
   3.3.2. Fire Fighting Unit Chief ............................................................................................................. 13  
   3.3.3. Medical Unit Chief ..................................................................................................................... 14  
4. General Game Flow ............................................................................................................................... 14  
5. Graphical User Interfaces ..................................................................................................................... 15  
   5.1. Main Menu Design .......................................................................................................................... 16  
   5.2. Pause Menu ..................................................................................................................................... 17  
   5.3. In-Game Screen Design .................................................................................................................. 17
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>System Architecture</td>
<td>18</td>
</tr>
<tr>
<td>6.1</td>
<td>Overall System Architecture</td>
<td>18</td>
</tr>
<tr>
<td>6.2</td>
<td>Data Flow Diagram Level 0</td>
<td>19</td>
</tr>
<tr>
<td>6.3</td>
<td>Data Flow Diagram Level 1</td>
<td>20</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Data Flow Diagram Level 1 for Server Game Engine</td>
<td>20</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Data Flow Diagram Level 1 for Client Game Engine</td>
<td>21</td>
</tr>
<tr>
<td>6.4</td>
<td>Data Flow Diagram Level 2</td>
<td>21</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Data Flow Diagram Level 2 for Server Game Engine</td>
<td>22</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Data Flow Diagram Level 2 for Client Game Engine</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>Data Design and External File Structure</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>Detailed Design</td>
<td>24</td>
</tr>
<tr>
<td>8.1</td>
<td>Packages, Subsystems and Their Classes</td>
<td>24</td>
</tr>
<tr>
<td>8.1.1</td>
<td>Game Data</td>
<td>24</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Menu Package</td>
<td>27</td>
</tr>
<tr>
<td>8.1.3</td>
<td>Rendering Package</td>
<td>27</td>
</tr>
<tr>
<td>8.1.4</td>
<td>Network Package</td>
<td>29</td>
</tr>
<tr>
<td>8.1.5</td>
<td>Sound Engine</td>
<td>31</td>
</tr>
<tr>
<td>8.2</td>
<td>Collaboration Diagrams</td>
<td>33</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Game Play – Server Collaboration Diagram</td>
<td>33</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Game Play – Client Collaboration Diagram</td>
<td>34</td>
</tr>
<tr>
<td>8.3</td>
<td>Sequence Diagrams</td>
<td>35</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Game Play – Server Sequence Diagram</td>
<td>35</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Game Play – Client Sequence Diagram</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>Project Schedule</td>
<td>37</td>
</tr>
<tr>
<td>9.1</td>
<td>Project Management</td>
<td>37</td>
</tr>
<tr>
<td>9.2</td>
<td>Development Tools Analysis</td>
<td>37</td>
</tr>
<tr>
<td>9.3</td>
<td>Game Concept Development</td>
<td>37</td>
</tr>
<tr>
<td>9.4</td>
<td>Game Resource &amp; Game Art Production</td>
<td>38</td>
</tr>
<tr>
<td>9.5</td>
<td>Graphics &amp; Audio Development</td>
<td>38</td>
</tr>
<tr>
<td>9.6</td>
<td>Network Development</td>
<td>38</td>
</tr>
<tr>
<td>9.7</td>
<td>Game Logic Development</td>
<td>39</td>
</tr>
<tr>
<td>9.8</td>
<td>Game Deployment and Testing</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>References</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>Appendices</td>
<td>41</td>
</tr>
</tbody>
</table>
11.1 Appendix A – Screenshots from Prototype ................................................................. 41
11.2 Appendix B – In Game GUIs ..................................................................................... 42
12 Gantt Chart .................................................................................................................... 43
1 Introduction

1.1 Project Description

‘Savior’ is a type of computer simulation program that simulates an abstract model of a real-life situation; “evacuation after a disaster”. The program will serve as a bridge between theory and experiment. It will provide conditions that are experimentally difficult so that users will have chance to develop their skills.

The content of program is developed following the properties of a simulation program. Several units – medical unit, firefighting unit and search & rescue unit - will try to accomplish given tasks using limited resources and in specified time by collaborating with each other. They will be challenged by different conditions and situations. Dynamic feedbacks will be given to the players in response to their actions throughout the game.

1.2 Project Features

Features that will be included in the project can be given as:

- **3D Graphics**: To simulate a real situation in computer, highly-configurable, software and hardware-accelerated graphics libraries are used. These libraries also support sophisticated animation so that animations within project will be realistic and smooth.

- **Dynamic Environment**: At each run-time, users may be faced with different environment items and different situations. Such a dynamic environment will catch user’s attraction.

- **Audio Communication**: The project includes a sound engine and a mechanism to transfer audio records between users over network.

- **High-quality Sound Effects**: By using high-level libraries, sound effects will be created as natural as real world.

- **Exchangeable Roles**: Having three main actors and three different tasks, users are given chance to improve their skills at different professional areas.

- **Script**: Using a scripting language, control of program flow will be done in run-time. Results for user actions will be shown dynamically and user satisfaction will be improved.

1.3 Purpose of Document

Purpose of this document is to express initial design issues about project. This document explains how project is designed to meet requirements specified in requirement analysis report. System components and the relations among them are described by using functional, structural and behavioral modeling methods.

Document contains the design issues about:

- **Game Concept**
  - Game Scenario
  - Game Elements
  - Game Play

- **System Components**
  - Overall System Architecture
- Structures of System Components (Game engine, Network Engine, Sound Engine, and Script Engine)
- Data Designs (modules, classes and their relations)

V Interfaces
- Game User Interfaces

V Tools, Libraries & External Resources

2 Design Considerations

2.1 Design Constraints

2.1.1 Scheduling

In order to reach the aim of finishing the project by the end of year, very detailed time schedule has been prepared. This schedule will be followed by team and at certain intervals, reports about finished tasks will be prepared. Project time schedule is given in section XX.

2.1.2 Performance

At certain intervals, project performance tests will be performed. Tests are aimed to identify the latencies caused by rendering and network, and to eliminate these latencies so that other phases of project are not affected. In such cases, number of polygons and the data amount transferred through network will be tried to be decreased.

2.2 Design Goals & Objectives

2.2.1 Portability

Although developing a platform-independent project would increase the workload of developers, this project is aimed to work on different operating systems. Choices for libraries and external tools are made considering this goal.

2.2.2 User Interactivity

Users of the project are not expected to have high computer knowledge. Therefore, one of the main goals of the project is to be user-friendly. To accomplish this goal, it is essential to make users' interaction with program as simple and as understandable as possible.

2.2.3 Reliability

Elimination of possible bugs and the consistent execution of program are two crucial issues for the success of project. For this purpose, the development phase of project will be completed as soon as possible and several tests will be done before release.

2.2.4 Realistic Environment

As the project has simulation features, it is intended to create the reality of actual events in the program. The quality of graphics, models, and sounds will be high so that they would be similar to real environment. Also, by generating input responses, users will observe a similar realistic environment.
2.3 Development Tools and External Resources

2.3.1 Software Choices

Since the project is aimed to work on different platforms, main concern while choosing the libraries was to be platform-independent.

After a short market research, it is observed that Java and C++ are the most preferable programming languages in game programming, and the libraries for these languages are huge in number. The choice of programming language is decided to be made considering the libraries for that language. Therefore, researches for libraries in both languages are made, and after a detailed research, C++ is decided to be used in the project. In following subsections, libraries chosen for the project, their brief descriptions and the reasons for the choices are given.

2.3.1.1 Graphics Library

After the analysis of many libraries, the main reason behind choosing Ogre as graphics library is Ogre’s high-qualified features compared to others. An evidence for its quality is that it is being used in many commercial games. Other than this, Ogre has rich documentation, tutorials and support forums for developers. As graphical Ogre-compatible user interface Crazy Eddie’s GUI System (CEGUI) will be used.

After Ogre is chosen as the graphics library, other libraries are chosen considering their compatibility with Ogre.

2.3.1.2 Sound Library

OpenAL is preferred as sound library. Its similarity with OpenGL, the experience of team members with OpenGL, being open source, having a sound recording feature, and its support in 3D sound system are the reasons beneath this preference. Reference documents of the library are available. Similar to Ogre, it has been used in many commercial games.

2.3.1.3 Input Library

OIS (Object-Oriented Input System) library is chosen to gather input from keyboard and mouse. Its compatibility with Ogre is the reason for this choice.

2.3.1.4 Network Library

Although there are many network libraries for Java, considering limited number of network libraries for C++ used in game programming, little documentation support, and the doubts about quality of libraries, network library is the hardest choice that has been made. After serious researches, RakNet network library is chosen. Being high-leveled, more documentation support compared to others, and being used by some game developers are the main reasons.

2.3.1.5 Scripting Tool

Considering the experience of team members with Pyhton, it is decided to be used as script programming language.
2.3.1.6 Software Tools

After researches that have been made 3D Studio Max will be used in Windows platform and Blender will be used in Linux platform as modeling tools. Other than these, free models with low polygon number are decided to be used in the project.

To manipulate images and textures, Adobe Photoshop in Windows platform and Gimp software in Linux platform will be used. For audio manipulation, Sony Sound Forge in windows platform, and in Linux platform Audacity software are going to be used. Microsoft Visual Studio in windoes and Eclipse in linux are chosen as integrated development environments for the project. Also, CELayoutEditor is used in design of graphical user interface.

3 Game Content

3.1 General Scenario

The scenario takes place in the campus of Middle East Technical University. The sketching of map of the program is given in Error! Reference source not found.. There are several buildings and open areas in the campus with different properties. Dormitories, sports hall, mall, culture and convention center, and departments are places where many people are found. Departments have classrooms where many students are being educated. There are people in the stadium watching a game of school team, or doing sport at Baraka Sport Hall, or attending to “Ben Anadolu” play at CCC. Also, some people are enjoying their meals at mall, or having rest at their rooms.

Suddenly a disaster happens. There are fires in several places i.e. in wood and in stadium. Several buildings have been collapsed, or are in danger of collapsing. There are many injured people and many are dead. People are in panic and running unconsciousness all around. All in all, after the disaster, the people are needed to be rescued and transported to the safety regions. Our heroes appear here and are in charge of everything in the campus. They are expected to rescue people, extinguish fires, and take control of the area.

Players are the chiefs of three main units within the program: first-aid and medical unit, firefighting unit, and search & rescue unit. Each chief has to accomplish his tasks in certain time but needs others’ help. The players are expected to collaborate with each other vocally and to perform the following tasks in general:

- To have situational awareness (to know what is going on so that to be able to figure out what to do.)
- To decide the order of areas to be interfered. (Areas should be interfered considering their strategically importance – i.e. presidency - , their possible dangers - i.e. gas stations, laboratories - and the number of people within - i.e. stadium, dormitories.)
- To rescue as many people as possible, and to minimize the total damage.
- To finish his own tasks in a reasonable time.

Before explaining the details of the players’ units’ tasks and expected actions, more information about the area and environment items are given in following subsection.
Figure 1 - Map of METU Campus
3.2 3D Environment of the Game

Environment items are designated as buildings, fires and civilians. Since our game has an educational purpose and needs to be as realistic as possible, players are expected to make the best decision at each situation. During the game, players will have to face different situations of each item and they should take properties of these into consideration to complete their tasks successfully. Therefore, characteristic properties of the environment items are listed and these properties will be shown to players during the game.

3.2.1 Buildings

Buildings are main structures in the environment. After disaster, there are some side effects. Fires in the buildings and the collapsing of buildings are some examples for these effects. Before interfering a building, units need information about it. Building-specific data can be stated as:

- **Identifier**: gives information about the utility and the importance of the building. For example, library, presidency, CCC, dormitory1, computer engineering department ... etc.
- **Population**: number of people inside or around the building.
- **Number of stocks**
- **Structure**: the material that the building is made of. It should be considered while entering the building in case of collapsing, or while choosing the extinguisher for a possible fire. Examples for structural properties are: reinforced concrete, wooden, adobe, steel, half-timbered.
- **Damage ratio**: can be grouped as slightly-damaged, moderate-damaged, and highly-damaged.
- **Possible dangers in the building**: fire in the building or the existence of explosives, chemicals, can be listed as possible dangers.

![Figure 2 - Engineering building with its properties](image-url)
3.2.2 Fires

For a firefighter, fire properties are crucial while dealing with fire. In order to end the fire as quick as possible and with minimum loss, further movements should to be determined considering all constraints and all available resources. Properties of fires that may be constraints for a fire are as follows:

- **Fire class**: There is a universal system to describe different types of fires. This system incorporates the use of letters, colors, and symbols to help users select an extinguisher suitable for the type of material involved in the fire. According to flammable substance, fires are classified as A, B, C, or D.
- **Fire kind**: Fire may be a building-fire, wood-fire, vehicle-fire, or may be caused by gas explosion.
- **Burning substance**
- **Direction and magnitude of wind**: has a reasonable effect on fire. Wind is one of the most constraints for fire growth rate.
- **Temperature**
- **Elapsed time**: Time elapsed from the beginning of fire may affect the growth rate of fire and the temperature in the area.

Since METU campus has large wood areas, wood-fire may be an important environment item in the game. Other than general properties of a fire, a wood fire has some extra features. These can be summarized as: *kind of trees in wood, age of trees, weather*.

![Figure 3 - Wood fire with its properties](image-url)
3.2.3 Civilians

People in the game are divided into two categories: main actors and figurants. Here, figurants are the ones without movement capabilities. Civilians in the area of disaster have an important role in the scenario. Main aim of the players is to treat injured ones and transport all civilians to safety areas. Here, some attributes of civilians are listed that should be considered to decide the treatment of them.

- **Health state:** To identify which civilians that need to be treated and which are not, health state is the major constraint. Some possible states are: healthy, in shock, unconscious, cerebral hemorrhage, bleeding, to have broken bones, to be burnt, smoke-poisoned.
- **Urgency:** Civilians should be grouped into three by medical unit to clarify the ones whose situations are critic and who should be treated first. Groups are: slightly injured, badly injured, and almost dead.
- **Level of consciousness:** helps the medical unit to understand the civilians situation and to decide what needs to be done e.g. open-conscious, close-conscious, and changeable.
- **Rate of respiration, Pulse rate, Body temperature, Age:** these properties determine the way of treatment.

![Figure 4 - Injured woman with properties](image)
3.3 Players and Their Tasks

3.3.1 Search and Rescue Unit Chief

Search and rescue unit chief has the most responsibility after a disaster. He has the responsibility to manage other units, to collect data about buildings and civilians within the area, and to use time and resources efficiently so that as many people as possible are evacuated to safety regions. Also, he is the authority in emergency situations to carry out orders, and use judgment in the event of disaster to evacuate area.

In the game, as a chief of search and rescue unit, player has to act carefully and perform certain steps in evacuation plan in minimum time. His main tasks can be grouped as:

- To rescue civilians from buildings those have collapsed or have risk of collapsing.
- To note building damages, fires, people trapped inside, and injuries.
- To collaborate with other units and make sure that related unit is informed about the situation being faced.

User will be evaluated in the game not only by the number of people he has rescued; also by the steps he had performed to perform his task. Possible steps in a search and rescue situation are explained below.

When the chief arrives to area, he should collect information about structural property of building, number of stocks, its utility, its damage and number of people inside. This information is then factored with other considerations to determine the initial urgency and scale of the mission. Then he should take safety precautions like turning off (electrical, gas, plumbing, or central heating) system to prevent further damage. After analyzing the situation, search phase should begin. A search by staff often aided by dogs should be performed to identify the positions and situations of the civilians. To make searching more efficient, task distribution between staff teams would increase points of team chief. When an injured civilian is found, medical unit should be notified about his situation and his health state. When a civilian is rescued, or civilians are present in the area, chief should keep them from running and direct them to the assigned evacuation location areas with vehicles. Search and rescue in an area should go on until it is certain that no one is left.

3.3.2 Fire Fighting Unit Chief

After a disaster, fires are encountered generally. In the game, as a chief of firefighting unit, the player is tested by his knowledge about situation but mainly his endurance to cope with the stress of the situation. He has the mission to take charge of people around, and to deal with urgent situations. Since fire may spread around, he has to be quick and has to direct his staff effectively. His main tasks are to extinguish fire, to take precautions to prevent fire, and to notify other units – especially medical unit in case of injured civilians.

When the chief arrives at the area, he needs to investigate about the area. He should collect information about the fire area such as if a building is in fire, its structural properties, possible substance that is burning...etc. should be gathered. Similarly, for a wood fire, constraints like kind of trees, wind properties, intensity of trees in the wood, and the weather details should be viewed by the player before any action. Type of fire (A, B, C, or D), the time elapsed from the beginning of fire, direction and magnitude of wind, and the growth rate of fire should be all observed. After gathering information, the chief should divide tasks that need to be done among his staff. He should assign jobs to staff carefully and if necessary, inform other units about the situation. Then, he needs to decide his actions. He should carefully identify the location to start extinguishing, and the materials that will be used to extinguish. As a firefighter, he has clothing, extinguishers, oxygen tubes and vehicles as
resources. These should be used carefully and only if necessary, because they are limited during the game. Considering each fire will have different properties, each one requires different materials to be used, and chief should have knowledge about the kinds of extinguishers and their usage areas. While extinguishing a fire, the chief should be careful about:

- To take wind on the back, to prevent injuring himself.
- To use correct extinguisher for that type of fire.
- Not to leave the area until fire is extinguished totally.

At each fire, certain conditions will be different, so actions need to be decided carefully. Any wrong decision will decrease the players' point and will get the current situation more complicated.

### 3.3.3 Medical Unit Chief

Disasters cause financial, structural, and human losses. Among these, the human loss is the most important issue. To minimize the number of people suffering from a disaster, medical unit has to do its tasks properly.

When medical unit comes to the area, there may be injured and dead people all around. As chief of medical unit, player is expected to:

- Apply triage to all injured people in the area.
- Apply treatment to people that are classified to have urgent health state.
- Ensure that injured people are transported to a safe region with an ambulance.

While performing these tasks, chief should be careful before making decisions and taking actions. Each wrong action or decision will worsen patient’s health state, and may even lead to death of him. Similar to chiefs of other units, his actions will be evaluated at each step, and game will continue according to results of these actions.

In an emergency situation, medical unit should enter the area after safety precautions are taken. After entering the area, to use time efficiently, chief should begin with the nearest patient. He should get information about consciousness, respiration rate, and circulation information of the patient. Then, patients should be classified according to their situations. While classifying the patients, triage should be applied to each patient. Chief should determine the number of doctors that will take further care of the patient, and the materials that will be used in further treatment. After triage and distribution of tasks between staff have finished, the chief can start treating the patients; starting with the ones that are labeled as urgent. If necessary conditions are held, first-aid should be done.

### 4 General Game Flow

Program starts with an introductory video about disasters and their effects. This video will be shown to give user an impression about real situations. Then, main menu will be displayed. User should type his user name and his password to enter a game. User may then click the new game button and create a new game as a facilitator or join an existing game. (Error! Reference source not found.) If the user prefers joining an existing game, he should enter his IP address and the role in the game. If the player chooses an existing role to join the game, he is given an error message and returned to join game menu. When the player clicks to I’m ready button on the left-bottom, he is directed to intro scene of the game until all other players are joined.

If user is the facilitator of the game, he has opportunity to choose the level of the game, and is able to decide the players joining the game. He can also use chat box to give details of the game to
players before the game, and he can start the game when all players are joined. If he starts the game before all roles are occupied, he is faced with an error message.

![Figure 5 - Join Game](image)

During game, player can do certain movements specified according to his role, i.e. only the medical unit chief may do artificial respiration. However, all players may use a resource or view the properties of the environment items around. At any instant, player will be given the possible actions and he may be able to decide on one.

All players are able to give tasks to his staff or decide to use a resource to do some action. Each player can see his resources on the right of his screen. Types and number of resources available are displayed all through the game so that he may decide on the amount and time of their usage.

When the player pushes the escape button, he is faced to pause menu. This menu is very similar to main menu, only it has a button to view resources details, and to resume game. If not, he can exit game.

Screenshots of the prototype are given in Appendix A.

5  **Graphical User Interfaces**

Graphical user interfaces of the program can be grouped into menu interfaces and in-game interfaces. Menu interfaces are main menu and pause menu, and in-game interfaces are different for each player according to his role in the program. These interfaces are implemented with Ogre-compatible CEGUI library. Details about the interfaces are explained in following subsections.
5.1 **Main Menu Design**

Main menu is designed considering the actions that need to be done in the beginning of the game. As explained in the general game flow, user needs to enter a user name and password either to join a game or to create a new game. For the easiness of user, a combo box with all existing usernames will be shown and he will be asked to enter a password for that username as seen in Figure 6.

![Main Menu](image)

*Figure 6 - Main Menu*

“Learn to Play” button is designed to especially increase user interactivity. Having a simulation utility, user may need to know some details about game to improve his skills. Player is informed about possible emergency situations in the game and some hints to handle such a situation with minimum loss. For example, a firefighter should gather information about wind direction before starting extinguishing action. Other than these, options for hardware (sound, microphone, speaker...etc.) can be adjusted by clicking options button. Statistics of each user are held in the database so that each user may compare his previous results and see his personal improvement or observe his teams’ success. Statistics button leads user to this screen. Last option for user is to exit the game and to terminate the program execution at that machine.

Menu types (both main menu and pause menu) have a tree structure. Each has their submenus as a child, and they may have common leaves. Tree structure of the menus provided easiness while traversing between menus and submenus. Details of the menu module are explained in coming sections.
5.2 Pause Menu

Pause menu is just as same as the main menu, only with a few differences like resources button. This button shows the details of remaining resources of the player. Another difference is that the statistics button here shows the current statistics/performance of user in the current game. After the game is finished, current statistics are also added to database so that user may view his total statistics in the main menu.

![Pause Menu]

Figure 7 - Pause Menu

5.3 In-Game Screen Design

During the game play, screen is designed so that player will not be distracted from the game. It is designed to be as simple as possible.

On the right column of the screen, the resources at that instant are displayed. This design was made considering that management of resources is a really important issue in the game. For each player in the game, the types of available resources are differentiated. Figures for these screens are given in Appendix B. Player may hide or show this sub-window in the game.

As stated in game flow, available actions will be displayed to players. Choices of these actions are designed to be displayed at the bottom of the screen. However, this will be clarified until the design report after a sample implementation for prototype.
6 System Architecture

6.1 Overall System Architecture

The project consists of several subsystems each of which is responsible for a different task. It is designed in a way so that it has a modular structure which makes it easier to maintain. Main principles of object-oriented approach are considered in the design phase. Coupling is tried to be reduced by minimizing the dependencies between the modules. Each package is aimed to do only its job.

The main element of the project is the game engine which controls other engines and manages game state. The other components that game engine manages are rendering engine, audio engine, network engine, script engine and input handler. Graphics engine is responsible for organizing the game scene and menu output of the system. This engine uses Ogre (Object-Oriented Graphics Rendering Engine) library for accessing graphics hardware. Ogre provides several high level features such as dynamic shadowing, billboards and particle systems. Those features are aimed to be used in the advanced implementation of the project. Audio engine is responsible for playing environmental game sounds and other players’ recorded voices; it also captures user’s own voice and directs this to the network engine. Similar to graphics engine, this engine uses OpenAL library for accessing audio device. Input handler module gets user’s keyboard and mouse inputs via OIS (Object-Oriented Input System) library. Network engine is responsible for sending and receiving packets. These packets include game commands and recorded voices. Communication and synchronization with other players are provided by network engine. Thus, it plays the most critical role in the application. To access network card, RakNet library is used which provides an API easy to maintain. Finally, script engine is the decision maker structure of the project. It makes decisions according to current game state and rules in the saved scripts.

![Overall architecture diagram]

**Figure 8 - Overall architecture**

As a multiplayer application the game has two sides: Server and Client. Game Flow is the same at both sides except the network management part. Initializations of all of the modules are done by the game engine which starts with the help of the booser module. After the initialization, the
application is started after choosing the desired settings. The application mainly consists of a message receiving and rendering loop. Rendering is done continuously depending on the Renderable Object’s properties. On the other hand, message receiving part is asynchronous and thus harder to handle. We built our network architecture upon an event driven approach called Publish-Subscribe method. The details of this method are explained in the Network Package. To explain generally, client generates events according to the changes of states of its objects. These events are listened by the EventListener application which tracks the Dispatcher objects tracking the state of objects. Actually every action or change of property corresponds to an event in our approach. Thrown events are both processed by the client itself and sent to the server. In the client, if those events are related to the sound they are directed to the sound engine either generating sound effects, or recording the player’s sound. In the same manner, if the events are related to the animation of a character, those are directed to the rendering engine. Rendering issues other than animation are not considered since it is a continuous loop rendering the objects according to their properties. Events are processed by the EventProcessor in the Client side. On the other hand thinking about the server side, it receives the messages sent from the client asynchronously. Then, it converts the messages to events and orders the events according to their timestamp and priority. There are predefined rules in the server which maps an event to other events. Those rules are defined according to the game logic. In this manner, the actual game state and logic is in the server. With the help of these rules server deduce new events and sends those events with the first event to other clients. Clients process these events and update their game state accordingly. With this approach general architecture is aimed to be robust and synchronized.

6.2 Data Flow Diagram Level 0

Game gathers input from keyboard and mouse. Also, sounds are recorded and sounds are transferred to/from other clients through network. Data related to game are also sent between clients over network. What is more, game fetches data from repository or writes data into repository, according to requests.

![Figure 9 - DFD Level 0 Diagram](image-url)
6.3 Data Flow Diagram Level 1

In this level, game is divided into several engines, each of which is responsible of different part of game flow. Game engine is the centre where information defining the game state is held, and where the instructions to other engines are sent. Only difference between client and server is the existence of script engine. The reason for this difference is; real game state is held only in the server and the decisions about next states are made in the server. Clients, however, have approximate game states and they update these states according to data coming from server.

6.3.1 Data Flow Diagram Level 1 for Server Game Engine

![Data Flow Diagram Level 1 Diagram at Server Side](image-url)

Figure 10 - DFD Level 1 Diagram at Server Side
6.3.2 Data Flow Diagram Level 1 for Client Game Engine

Figure 11 - DFD Level 1 Diagram at Client Side

6.4 Data Flow Diagram Level 2

In this diagram, sub modules for the engines in level 1 can be seen. Encode and decode modules in network engine are responsible for conversion of events and sound data to messages and the vice versa. Decoded voice data is sent to audio engine, and the message is sent to game engine. Game engine has a decision maker structure other than the controllers of other engines. This decision maker structure can be explained as arranging the events of servers' itself and the events coming from clients according to their priorities and processing them with the help of script engine.
Also, the event itself and the events triggered by this incoming event are sent to other clients. Clients update their game states according to these coming events.

### 6.4.1 Data Flow Diagram Level 2 for Server Game Engine

![DFD Level 2 Diagram at Server Side](image-url)

Figure 12 - DFD Level 2 Diagram at Server Side
6.4.2 Data Flow Diagram Level 2 for Client Game Engine

Figure 13 - DFD Level 2 Diagram at Client Side
7 Data Design and External File Structure

Since the data that will be stored is not complex, an external DBMS is not required; instead of this, a XML file is sufficient to store the data. For processing the file, an XML Parser API will be used. Furthermore, the configuration file will be stored in XML format. It includes entities about especially graphics and network options. For example; video mode, rendering device, colour depth, display frequency, sound volume and IP address. Moreover, there will be other xml files for indexing game sound files, images, videos, model and texture files.

![E/R Diagram](image)

Figure 14 - E/R Diagram

8 Detailed Design

8.1 Packages, Subsystems and Their Classes

8.1.1 Game Data

There are a lot of objects in the program. Each object has different properties and different kind of data. Considering the workload of network, data classes are needed to be defined well-structured.

8.1.1.1 Design Decisions

Considering total amount of total data in the program, excess data are not stored in the system. To add, graphics part of the program is taken into consideration while forming the object
classes. To clarify this renderable objects have very distinct properties compared to non-renderable ones, so these objects are defined in different classes.

Figure 15 - Class Diagrams No.1
Figure 16 - Class Diagrams No.2
8.1.2 Menu Package

Figure 17 - Class Diagram of MenuObject

CeGUI will be used in the project as the Graphical User Interface Library, which is quite a flexible library integrating into Ogre easily. Also CeGUI fits the menu logic that will be used in the project. In this library, different from most GUI systems that is displayed is a subclass of the Window class, and a window can have any number of children windows. This means that when you create a frame to contain multiple buttons, that frame is a Window. The menu objects in the project will have a window as its property. The children of the menu objects will have window objects which are children of the parent menu object’s window object. In this way, correspondence between the menu objects and their interfaces will be maintained. Also the information of whether this object is activation, selection or visualization... etc. will be taken from the window object.

8.1.3 Rendering Package

8.1.3.1 Description of Rendering Classes

Renderable Class handles all renderable data in the game being a base class for the Renderer class. Renderable class has the Entity attribute which represents the mesh of the rendered object.

Renderer Class is an abstract class, having an ObjectDispatcher in which the node to render the object is provided. Since we use Ogre as the graphics engine, we follow a scene-related approach. Each renderable object is attached to a parent node. A child node moves related to its parent node. This makes it quite easier to implement the transformation of the displayed objects.

AnimableRenderer and NonAnimableRenderer Classes inherit from Renderer. Actually, animable is used in the meaning of movable and/or animable here. Animable objects in the game
include the actors, visual effects of some environment related events (such as fire, or storm) and some of the resources. Nonanimable objects are the MenuObjects and some of the items.

MenuObjectRenderer class is responsible for rendering the menu items according to their type and value.

8.1.3.2 Design Decisions for Rendering Classes

Graphics package is one of the largest components of the design. Ogre has also directed this package’s design. As a camera can be attached to any renderable object in Ogre, there was no need to implement a class to deal with the camera position. Also as the scene is rendered by rendering all of the nodes, transformation will be easier. In this manner, while designing the classes extended features such as animation are considered. This package is designed to be extensible by making the Renderer classes as generic as possible. Also providing different Renderer classes gives the opportunity to handle rendering and animating issue in a much better way.

![Class Diagram of Rendering Classes](image)

**Figure 18 - Class Diagram of Rendering Classes**
8.1.4 Network Package

8.1.4.1 Description of General Network Classes

*ConnectedUser:* There is a ConnectedUser class which Server and Client inherit from. It consists of the common attributes and methods of server and client such as an instance of a Properties class which has the connection information in, send () and receive ()...etc. methods. Client and Server classes have also own properties specific to themselves which will be explained below. ConnectedUserState class is the main class including the dynamic data of the execution. Client and Server both have a ConnectedUserState describing the dynamic properties of the objects which change according to game play. All of the Client, Server and ConnectedUserState classes implement the singleton pattern. Also a ConnectedUser instance needs a MessageEncoder and MessageDecoder instance to convert events to messages and vice versa.

*Server and Client Classes:* The Server class makes up the core of the server to be implemented in the project. The main issue considered while designing the server class implementation was synchronization. Server class includes the global game state, server properties. Also singleton instance of EventOrderer is included in the server. Global game state is the actual game state maintained in the server. It is represented by a ServerState object. The clients’ game states’ being as much identical to the global game state as possible is important to provide synchronization between users. GameState class mainly deals with the general parts of the game and the interaction among the players and items. In the GameState class there are dispatcher classes which are responsible for holding the dynamic data for the whole game. Such dynamic data includes the position, orientation and state of items and players.

Whenever a data item’s dynamic property is changed it causes an event. In other words a change at an object is turned into an event. This event is resolved by the EventListener and processed by the EventProcessor, on the other hand it is also sent to the server. An event is converted to a message by putting into a general packet structure by the MessageEncoder. In the server the message is converted to an event by the MessageDecoder. MessageDecoder sends the event to the EventOrderer. In the event orderer events are ordered according to its timestamp and priority. (Actually we are considering using an API which sends the events in the correct order. However timestamps may also be used in case) In the ServerGameState there is a hash map consisting of the rules which map one event to many events. This is the difference of the server from a client. Rules are predefined mappings of the events. For example a wrong treatment to the fire is an action so it is also considered as an event. This event causes another event: fire’s becoming larger. This mapping is expressed in a rule. So an event automatically causes another event in the game loop. There will be a continuous chain of events in the application as input comes from the player and other clients. Events deduced from the rules in the server, are sent back to all clients together with the first event. So they are aware of the client’s action that has sent the event.

This approach makes the application more deterministic and clever. If all possible events are specified and implemented in detail, the application will behave accordingly. The application’s being deterministic in such a way makes it a better education tool.

8.1.4.2 Network Design Decisions

Network architecture was the most critical part while designing the system. It affects both how the application is working and whether the application is working and results in a catastrophic impact on the project if not designed correctly. Therefore, it is important to decide on the architecture at the beginning and design the remaining system accordingly. In the requirements analysis and design phase, many other applications and networking paradigms used with those are investigated. Although harder to implement in some cases, event-driven approach seemed to be the
most useful and appropriate approach for this system. A paradigm called publish/subscribe paradigm is planned to be used on top of this approach. This paradigm is an asynchronous messaging paradigm where senders of messages are not programmed to send their messages to specific receivers. Rather, receivers express interest in one or more classes and senders send messages without specifying a receiver. The reason that this paradigm is chosen is the fact that we aim to maintain a dynamic network topology, which also provides synchronization. Also it is known that separating the data that a client is relevant is quite a difficult part of the implementation. With the help of this approach, the clients don’t have to specify the objects they are interested with. Combining this paradigm with the event driven approach, it will be enough for the clients to specify the events they are interested with.

**Figure 19 - Class Diagram of Network Classes No.1**
8.1.5 Sound Engine

There are two classes related with audio, which are inherited from audioEngine class. AudioEngine class only includes status information and functions about status and initialization. One of the inherited classes is responsible for playing audio and the other is responsible from capturing. AudioCapture class has functions for starting, stopping capture and getting recorded audio. Recorded sound is put into a buffer by OpenAL and by getRecordedAudio function the address of the buffer can be got. AudioPlay class is responsible for playing audio. This class can play recorded sound, generate sound effects and also play environmental sound. For playing environmental sound, giving id of the sound is sufficient because the class can reach indexed and stored sounds on hard disk.
Figure 21 - Class Diagram of Audio Classes
8.2 Collaboration Diagrams

8.2.1 Game Play – Server Collaboration Diagram

Events coming from the client and events deduced by the server are both sent.

Asyncronous message comes from the client.

EventManager defines whether events from the given event and predefined rules
8.2.2 Game Play – Client Collaboration Diagram
8.3 Sequence Diagrams

8.3.1 Game Play – Server Sequence Diagram
8.3.2 Game Play – Client Sequence Diagram
9  Project Schedule

In our requirements analysis report, main tasks and subtasks of our project were identified. Relationships between subtasks were defined and each task was given a start and a finish date. Team members were assigned to these tasks according to their interests and skills.

Our tasks are grouped under eight work packages. Following sections describe these work packages and explain what has been done so far and what is planned to be done as future work.

9.1  Project Management

“Project management” package includes tasks related to documentation of our project. Following tasks are involved in this package:

− Proposal Report
− Requirement Analysis Report
− Initial Design Report
− Final Design Report
− Web Page

Proposal report, requirement analysis report and initial design report have been delivered so far. Final design of the project is planned to be completed by the end this semester. Group members will be working on the final design in following weeks. Also the web page will be kept up-to-date with new documents and downloads as they are released.

9.2  Development Tools Analysis

“Development tools analysis” package includes tasks related to research and analysis of the software tools that are going to be used in the implementation phase of the project. Following tasks are involved in this package:

− Open Source Graphics Engine Analysis
− Open Source Network Library Analysis
− Open Source Audio Library Analysis
− Modeling Tool Analysis
− IDE Analysis

The tasks under this package have been completed at this initial design phase of the project. Open source graphics engines, network libraries, audio libraries are analyzed in detail. The software tools that will be used in the implementation are determined accordingly.

9.3  Game Concept Development

Tasks related to the conceptual design of the game are grouped under the “Game Concept Development” work package. Following tasks are involved in this package:

− Game Subject Decision
− Scenario Development
− Information Gathering about Emergencies, First Aid, Firefighting, Search & Rescue, Evacuation
− Character Roles & Resources Identification
− Game flow Design
− User Interactivity/Control Decisions
So far game subject has been decided, game scenario has been developed, information about emergencies, first aid, firefighting, search & rescue and evacuation have been gathered. Also character roles and resources have been identified, game flow has been developed and some sketching has been done. Although these tasks are mostly completed; a few minor changes can be made until the final design report.

User interactivity & controls and storytelling will be determined in the following weeks and will take its final form in the final design. Graphical user interfaces such as main menu, game creation menu, game sign-in menu and pause menu has been designed so far, whereas in the following weeks, options menu, user controls menu and sound & audio settings menu design should be completed. For game map, the buildings, places, areas in the Metu campus that will take place in the game story should be specified.

**9.4 Game Resource & Game Art Production**

Tasks related to production of game resources such as 3d models, textures, sound effects are grouped under the “Game Resource & Game Art Production” work package. The tasks under this package include:

- 3D Objects Modeling
- 3D Characters Modeling
- Texture Creation
- 2D Map Modeling
- Sound Effects & Music Generation

These tasks are scheduled to begin in the second semester.

**9.5 Graphics & Audio Development**

“Graphics and Audio Development” package contains tasks related to implementation of 3D graphics, graphical user interface and sound effects. The tasks under this package include:

- Surface Texturing
- Graphical User Interface Implementation
- 3D Object Loading and Rendering
- 3D Environment Implementation
- Illumination Implementation
- Sound Effects Implementation
- Animation

Implementation tasks are scheduled to begin in the second semester.

**9.6 Network Development**

“Network Development” package contains network related tasks. Following tasks are involved in this package:
- TCP Based Low-Level Protocol
- High-Level Server-to-Client Protocol
- High-Level Client-to-Server Protocol
- Packet Creation
- Packet Parsing
- Packet Transfer
- Server Network Implementation
- Client Network Implementation
- Send and Receive Modules
- Server Multi-threading Implementation
- Synchronization
- Compressing
- Encryption

By now, high level server-to-client and client—to-server protocols have been determined. Network classes are designed and the format for data transmission, data encoding and decoding functions are specified.

9.7 Game Logic Development

“Game Logic Development” package includes tasks related to implementation of game logic related issues such as the change of game state in response to a user input or time. Following tasks are involved in this package:

- Game Logic Implementation
- Game Scripting
- Database Implementation
- Game Save/Load Implementation

So far, development of game logic has been mostly completed. Response of program to user input and events that occur in regular time intervals has been decided. In the following weeks, the logic will be specified in details. For example, the growth rate of fires per minutes will be determined clearly.

9.8 Game Deployment and Testing

This package contains the tasks about “Game Deployment and Testing” given as:

- Optimization
- User Manual
- Installation Manual
- Demo Portfolio
- Prototype Demo Implementation
- Final Demo
- Test Cases Design
- Alpha Testing
- Beta Testing and Getting Feedback
- Debugging
- Upgrades/Patches

Tasks in this package are planned to be done in next semester.

10 References

11 Appendices

11.1 Appendix A – Screenshots from Prototype
11.2 Appendix B – In Game GUIs
12 Gantt Chart