Online Virtual Team Collaboration Platform With 3D Graphics

Requirement Analysis Report

Incredibles

Salih Ahi
Kamil Nematli
Mustafa Önder
Abdulkadir Yazıcı

2007

1. Introduction

This document is prepared by Ceng490 students in order to analyze this project's details and requirements.

1.1. Project Description

1.1.1. Purpose

The purpose of the project is:

- To simulate real world scenarios in a virtual environment.
- To train squad leaders each specialized on its own area and improve their skills in their areas.
- To improve collaboration between different squad leaders.

1.1.2. Scope

The project will be developed for leaders of squads mentioned below:

- police squad
- fire squad
- bomb defuse squad

In addition, this simulation could be useful for anyone who wants to increase their team collaboration skills.

1.1.3. Objectives

Project will have the following features:

- The scenarios are going to be designed in a realistic manner.
- There will be a facilitator who manages the simulation from the server and can intervene anything in the scenario.
- Facilitator will be able to choose among several different scenarios.
- Users can communicate with facilitator and other users via the voice chat.
- Facilitator can observe everything in the simulation from different view angles.
- Users follow the scenario from the first person view.
- Users will have limited resources and tools. These resources will be both equipments and people.
- The squad leader's subordinates are capable of accomplishing given tasks.

The program will run in two modes: passive and active mode. The features above will be same for both modes, however there will be some differences:

Passive mode:

- Users have restricted interaction with computer.
- Users manage their teams via the facilitator.

Active mode:

- Users have active interaction with computer.
- Users manage their teams using user interfaces on their own computers and via the facilitator.

1.2. Constraints

There is not much constraint given by the company. Also there is not any restriction to project platform or development environment. The constraints are:

- The project must have been finished by June 2008.
- The project must be done by 4 Metu-Ceng senior students.
- The project schedule must be synchronous with the course schedule.
- The trainee user interface must not be complex.
- Trainees' view perspective must be first person view.
- Facilitator must not affect the flow of scenario.

1.3. Management

Although the lack of an experienced programmer with leadership capabilities will require us to deploy Democratic Decentralized team management, that model may cause latencies in deadlines which should be strictly avoided. The estimated number of lines of code is large enough to pull the slider to a Controlled Centralized structure. Yet the modularity of the problem is not realy high, and a structure as strict as CC might result in finger-pointing and loss of confidence when a member fails to complete his task successfully. Considering the cases above, we will use a Democratic Centralized team structure. We will solve the main problems as a group, whilst implementation of tasks will be done individiually.

1.4. Scenario

Prologue : The scenario takes place in a large public building (like a big shopping center as Armada) in present time. In the building several minor bombs have exploded and as a result fire has started in some areas. And it has been reported that several unexploded bombs may still exist.

Main goal : Take all of the civilians to a safe area before they get hurt and defuse all of the bombs.

Teams : Firefighters, police squad and bomb defuse squad

Tools & resources :

Firefighers : Water is used as both tool and resource. They have two sources of water one of which is infinite and the other one is finite. Also they have a fire engine.

Police squad : A maul will be used as a tool for breaking the doors. Also a resource will be used for taking out the civilians from the high levels of the building.

Bomb defuse squad : A trained dog will be used as a tool for finding the bombs. Also detonators will be supplied as a resource for deactivating the bombs and exploding the doors.

Subgoals :

Firefighters: Their task is to prevent the fire to spread over the building. Also they must distinguish the fire in specific areas so that police and bomb defuse squads will be able do their task on these areas.

Police squad : Their task is to find and take out the civilians from the scene. Also they must help bomb defuse squad enter the rooms by breaking the doors with their maul.

Bomb defuse squad : Their task is to find and deactivate the bombs. Also they must help police squad enter the rooms by blowing up the doors with their detonator.

Colloboration Analysis:

If a team does not exist;

Fire fighters : The fire will spread over the building. As the result police and bomb defuse squad wil not be able to operate on areas which are under fire, also civilians will be hurt by the fire.

Police squad : Not all the civilians will be found and the found ones will be taken out of the scene in an unorganized way. Also bomb defuse squad may have some problems on entering some rooms since no doors will be broken by the police squad.

Bomb defuse squad : The bombs will not be defused and will damage the building and may damage the civilians in the evidence area. Also the police squad will not be able to enter some rooms since no door will be blowed up by the bomb defuse squad.

If a team isn't collobrating with the others;

Firefighters : Police and bomb defuse squad wil not be able to operate on areas which are under fire. And they may be trapped by the fire. Also civilians will be hurt by the fire.

Police squad : There will be time loss for bomb defuse squad for waiting the door to be broken. Also firefighters may try to distinguish fires in unnecessary places.

Bomb defuse squad : Police squad or civilians under their control may be hurt by bomb defuse squad's detonator. There will be time loss for police squad for waiting the door to be exploded. Also firefighters may try to distinguish fires in unnecessary places.

Scenario Specifications:

Bombs defuse squad is able to blow up any door with their detonators, but police squad is able to break only thin doors. Since bomb defuse squad has a limited number of detonators, they are not enough for blowing up all the doors. So they need police squad assistance for passing through the thin doors. Likewise police squad needs bomb defuse squad assistance for passing through the though doors.

2. Requirements

2.1. Functional Requirements

2.1.1. Trainee Interface

The trainees are assumed to be inexperienced users. Thus the interface must be simple, yet have a high functionality. After configuring the settings in the main menu, trainees will be directed into the simulation.
In passive mode, since all orders are to be given by voice chat, almost no user interface element will be visible except the chat panel. That panel will let the user figure out who is talking at the moment. Also current amount of resources can also be seen from the screen.
In active mode, users will have more responsibilities and are required to do some tasks themselves. In addition to the chat panel and the resources, there will be buttons that enable the trainee to give direct commands to their squads such as moving to an area or defusing a bomb.
In both modes, users will be able to check the status of objectives.

2.1.2. Facilitator Interface

In contrast to trainees, a complex user interface will not be much of a problem for the facilitator. Especially in the passive mode, a great deal of orders is to be executed by the facilitator. As the facilitator selects an object or person on the map, the task panel on the screen will show available commands for that object. From that panel, orders such as move, rescue, diffuse can be executed or the assets like amount of detonators can be changed.

2.2. Software Requirements

2.2.1. Developer Software Requirements:

- MS Windows XP/Vista for OS
- MS Visual Studio for IDE
- MS .NET Framework 2.0 SDK
- Python and IDLE for scripting
- MS Word, Excel, Office Project and Visio for documentation
- 3ds max for 3D modeling and animation
- Photoshop/Paint Shop Pro for textures and images
- Graphics, physics, AI (if any), network (if any) and sound engines SDKs or APIs
- TortoiseCVS for CVS

2.2.2. User Software Requirements:

- MS Windows 98/XP/Vista for OS
- MS .NET Framework 2.0

2.3. Hardware Requirements

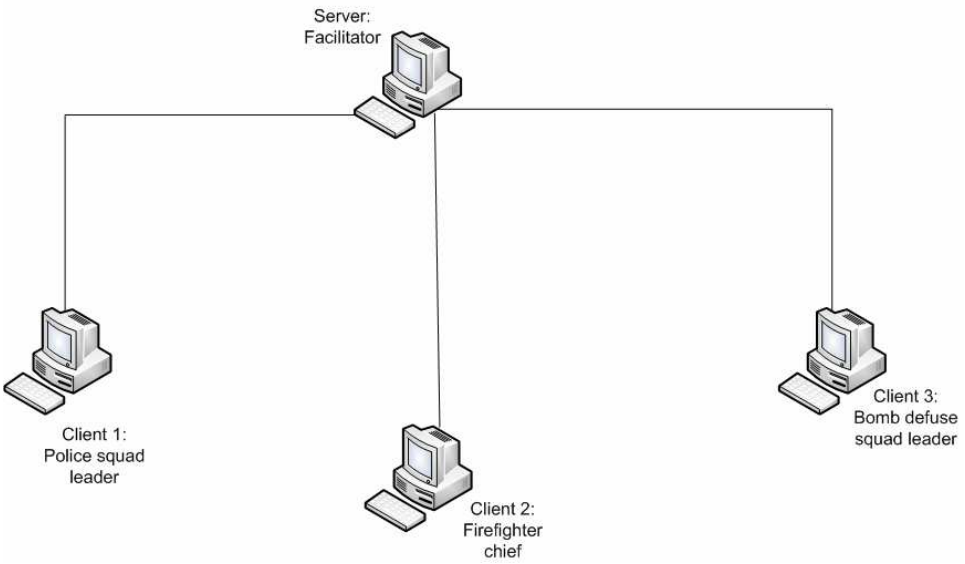
These are for both user and developer.

- Intel Pentium3 or Equivalent Processor
- 3D accelerated graphics card
- 256 MB RAM
- Sound card
- Network card (modem, Ethernet card...)
- Keyboard, mouse, speakers and microphone

2.4. Technical Requirements

2.4.1. Networking

The project will be a real time online multi-user simulation. Users will connect to the system via network connection. This requires a good server/clients network architecture. There will be three clients and a server connected to each other. It is simply as follows:



This scheme makes following network requirements necessary:

- Use TCP/UDP protocol
- Handle and process messages that server receives from clients
- Handle and process messages that clients receive from server

- Multi-threading for handling clients' connections
- Synchronize movements and actions of clients

Generally, there are two issues network engines should consider about: latency and synchronization.

Latency: Since this project is a real-time simulation system, latency should be as low as possible. Users expect low latency from the simulation. In server/clients architecture number of nodes, distance and network hardware affect latency over network. As the system has three users and facilitator the latency will not be a problem.

Synchronization: In real-time systems synchronization has crucial importance and should be handled with care. Server should sent actions to the clients in a synchronized way that all clients must receive packets at the same time without any losses.

2.4.2. Sound

Sound engine is the part of the system that makes it more realistic. All the sound effects and voice communication will be supported by this part of the project. The engine has the following requirements:

- Simulate sound effects of the environment such as, opening door sounds, screams of people, dog barking etc.
- Encode the voice incoming from microphone.
- Decode the voice in order to send to speaker.
- Create and use a voice stream channel to exchange the encoded voice data.

2.4.3. Graphics

Although graphics quality is not a major case in the evaluation of the project, users must have satisfactory graphics on their screen to understand the surronding environment. In active mode, user will be able to interact with some objects directly. Whereas in passive mode, interaction will be restricted and commands will be given through the facilitator. This made it necessary to design two different interfaces, one for the users and one for the facilitator, which are described in detail above.

To get rid of low level implementation details, a graphic engine (irrLicht) and a level editor (most probably irrEdit) will be used. This will strictly seperate graphic details from the game logic.

2.4.4. Simulation Engine

The core part of the simulation engine must satisfy these requirements:

- Timer (for synchronization of connected simulations)
- Managing and transmitting objects between itself and graphics, physics and ai engines

- Applying simulation logic
- Resource management
- Read and apply scripts
- Import maps from external files
- Set the simulation's state (which are menu, in-simulation and paused states)

2.4.5. Physics

The simulation needs to apply simple physics rules. With the needs of the scenario, these requirements are:

- Physical objects which has properties as mass, volume (which is bounding box/sphere) and velocity.
- Collision detection (for avoid passing through physical objects and moving physical objects by force)
- Breakable physical objects (for explosions)

2.4.6. Artificial Intelligence

Since the user's subordinates will be non-playing characters, an ai engine will control them. The ai engine's requirements are:

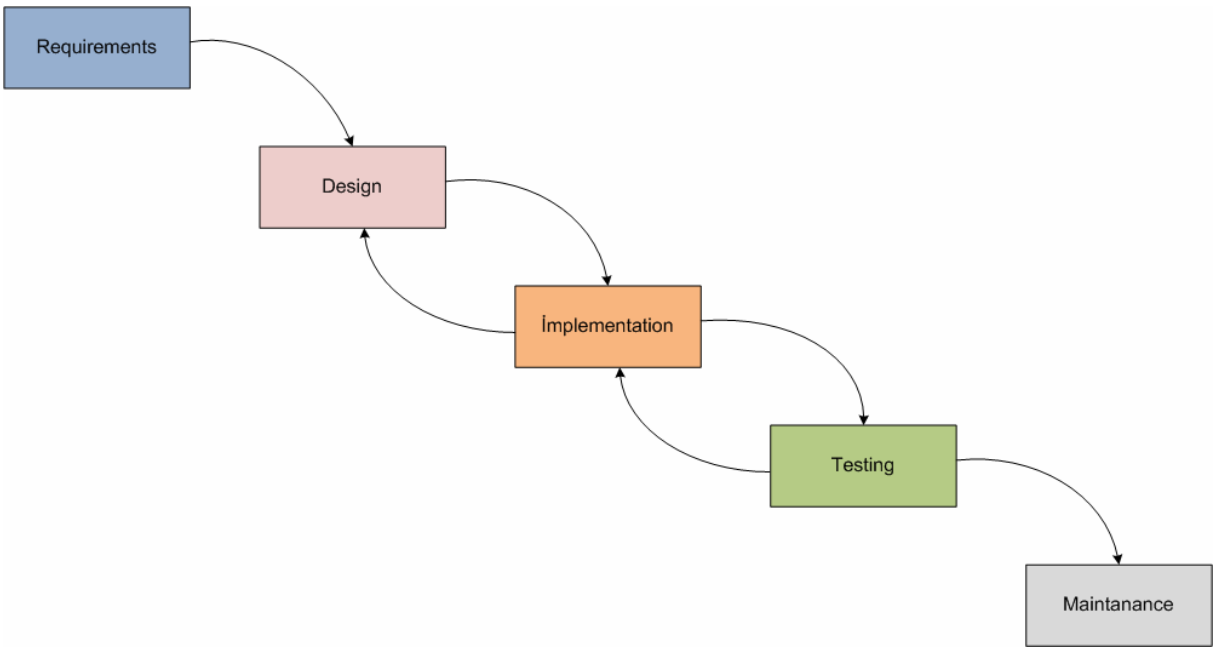
- Shortest path algorithm (so NPCs will move in a realistic way)
- Simple reaction based behavior model (such as escaping from fire)

3. System Analysis

3.1. Software Model

The steadiness of the requirements makes the Waterfall model the most suitable software development process. The simple and disciplined structure of the model will help us follow a concrete roadmap. Another advantage of this model is that it has discrete, easily understandable phases and marked milestones. At certain places during development, we will prepare prototypes which will ensure correctness of our progress and will help us determine the defects in the timeline.

Since there is no space for feedback in the original model, we –as many who use this model do- have modified the process a bit. The results in testing may require some changes in the implementation, and it is probable that changes may be realized to be necessary during the implementation that cause some changes in the design. Such feedbacks can be seen in the figure below

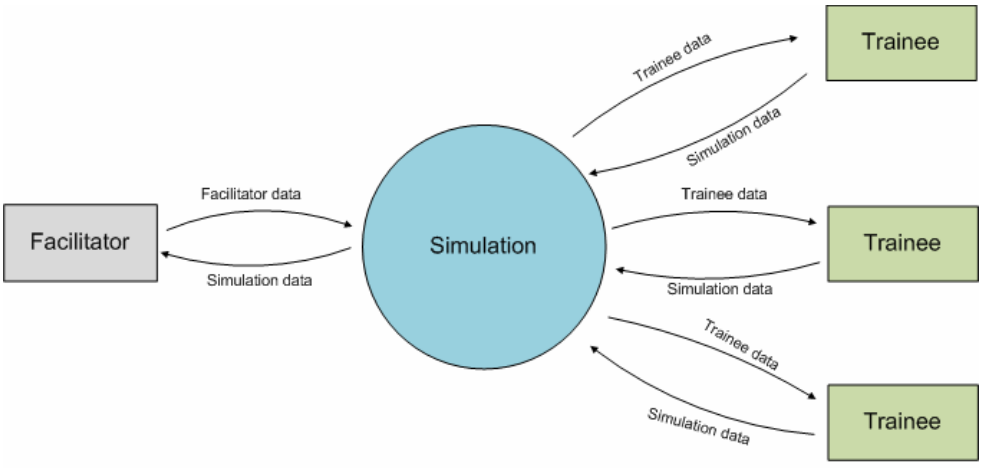


Waterfall Software Model

3.2. Data Flow Diagram

3.2.1. DFD Level-0

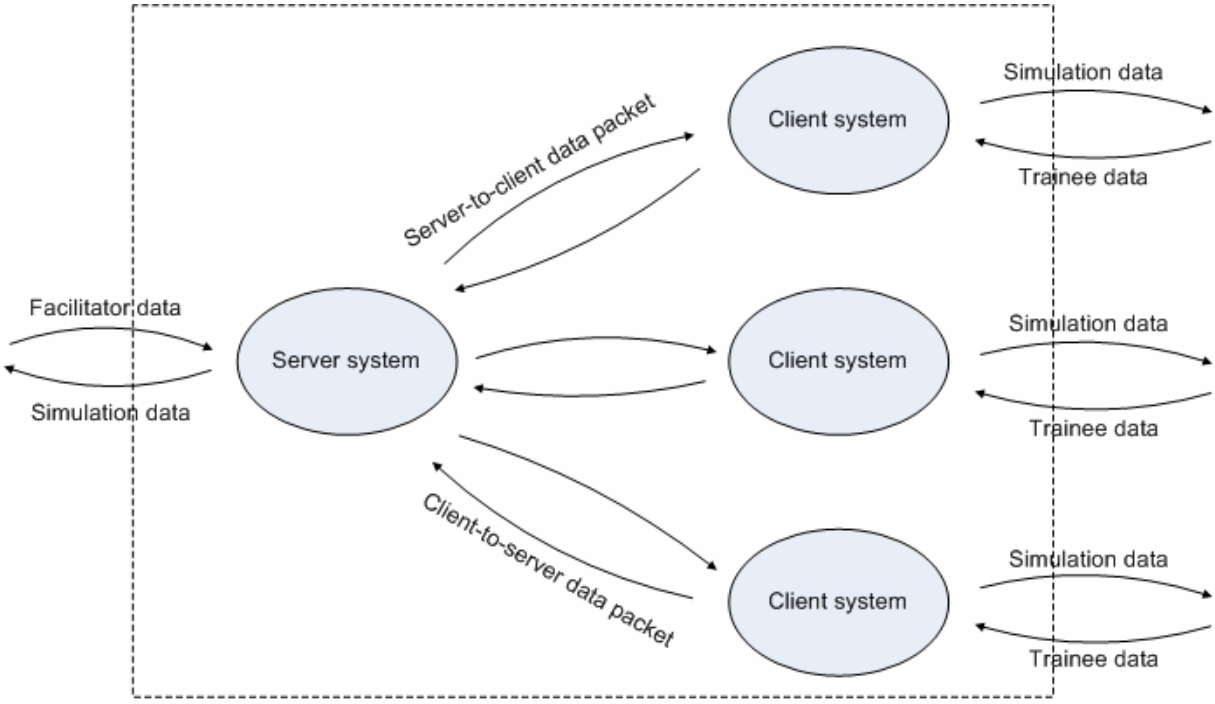
This diagram explains Data Flow Level-0 of the simulation. Facilitator and trainees sends commands via mouse, keyboard and microphone. They receive graphical representations of the simulation environment and listen to other users from their speaker.



DFD: Level 0

3.2.2. DFD Level-1

This diagram explains Data Flow Level-1 of the simulation. Client-to-server data packets include information such as user's outgoing voice and user commands. Server-to-client data packets include information such as other users' incoming voice, environment objects' modified properties.



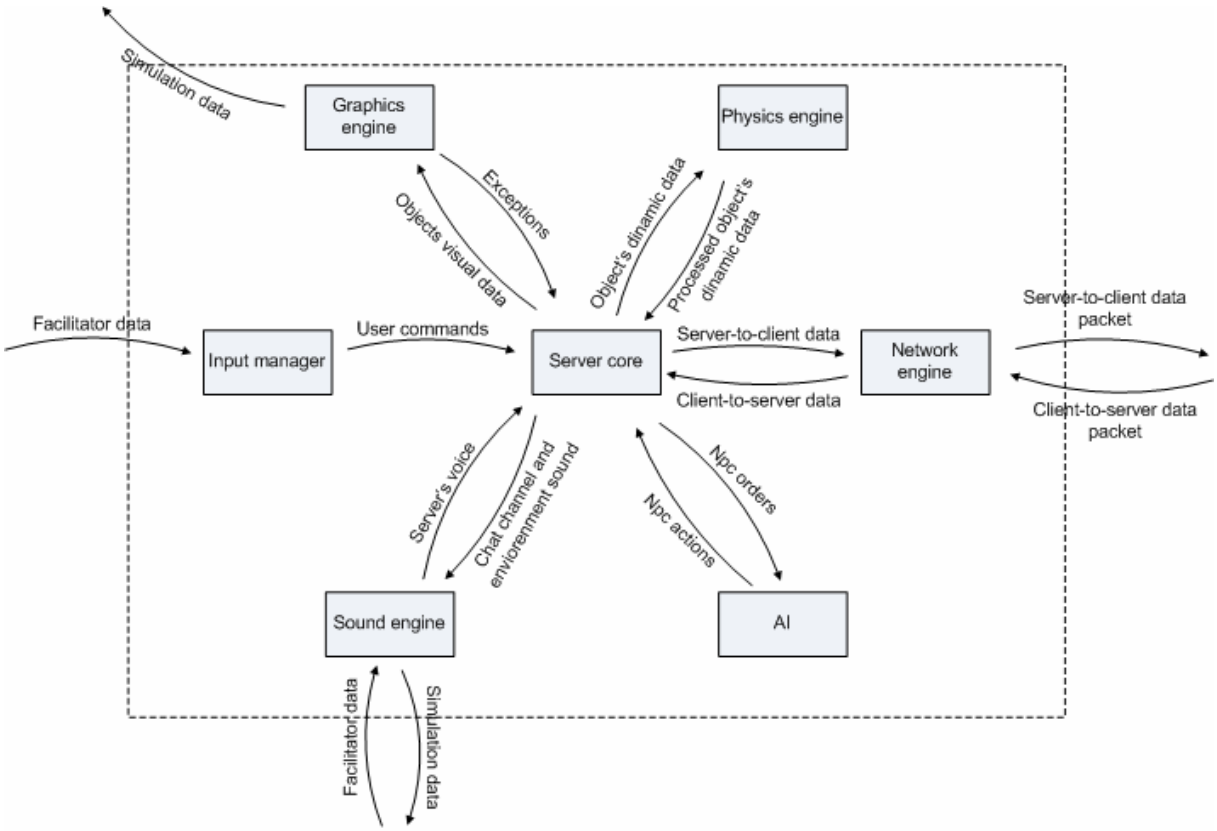
DFD: Level 1

3.2.3. DFD Level-2

These diagrams explain Data Flow Level-1 of the simulation.

3.2.3.1. Server Core

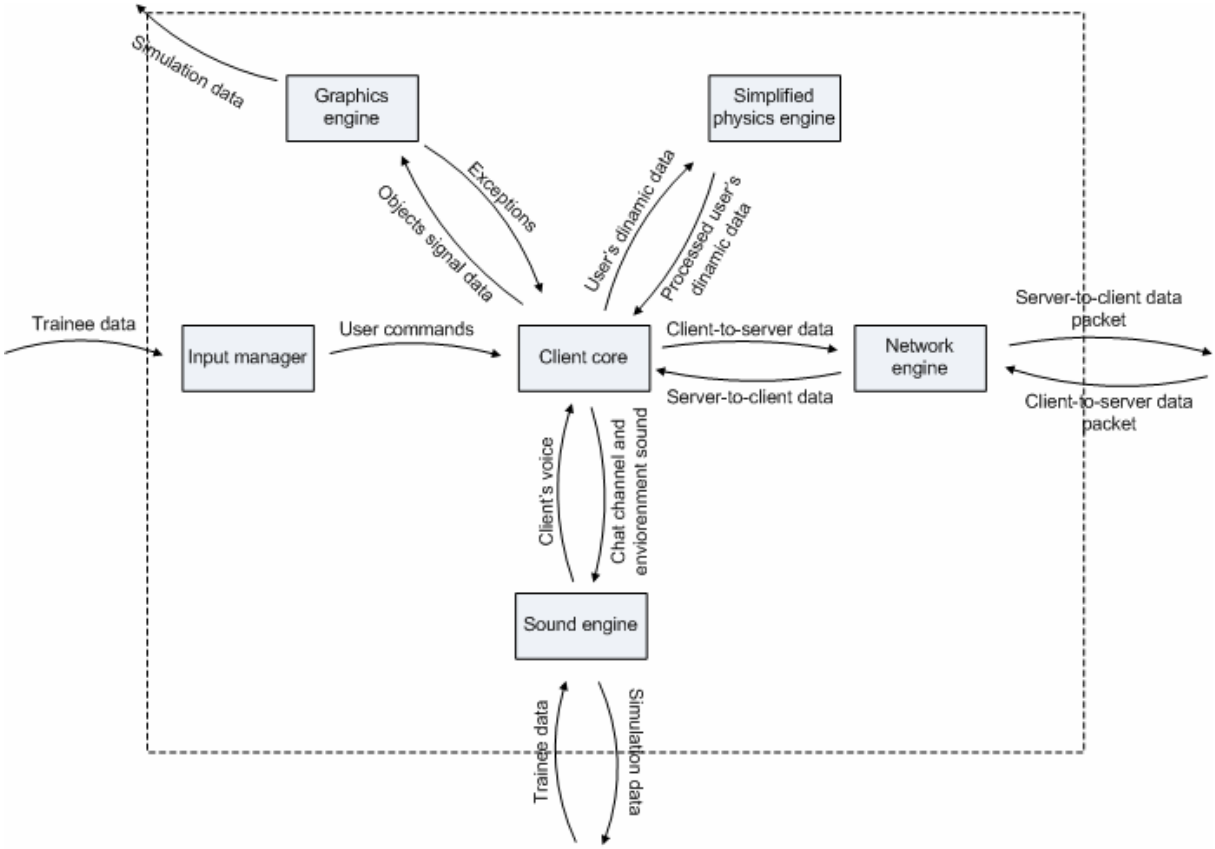
- Object's visual data contains visible objects information.
- Graphics engine returns only exceptions to server core.
- Object's dynamic data contains physical object information.
- Processed object's dynamic data consists of physical object's modified information.
- NPC orders consist of given to non-playing characters.
- NPC actions consist of non-playing characters' reaction to given orders.
- User commands are commands generated by the user inputs.



DFD: Level 2 Server core

3.2.3.2. Client Core

- Object's signal data contains visible objects information.
- Graphics engine returns only exceptions to server core.
- User's dynamic data contains user's character's physical information.
- Processed user's dynamic data consists of physical object's modified information
- User commands are commands generated by the user inputs.

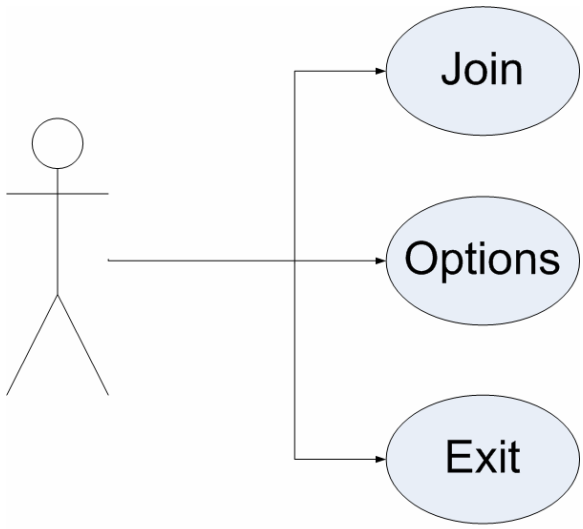


DFD: Level 2 Client core

3.3. Use Case Diagrams

3.3.1. Client: Menu State Use Case

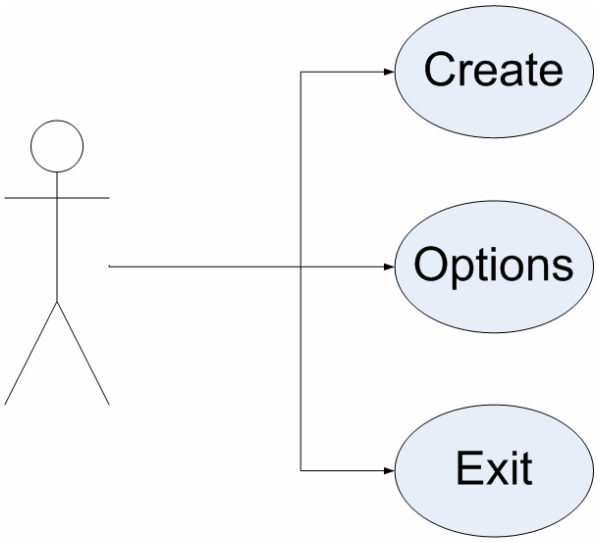
This is the main screen the client will face when starting the program. The screen has 3 buttons.
Join: Connect to the server at the given IP address.
Options: Set program options such as graphics, audio and controller configurations.
Exit: Terminates the program.



Client: Menu state

3.3.2. Server: Menu State Use Case

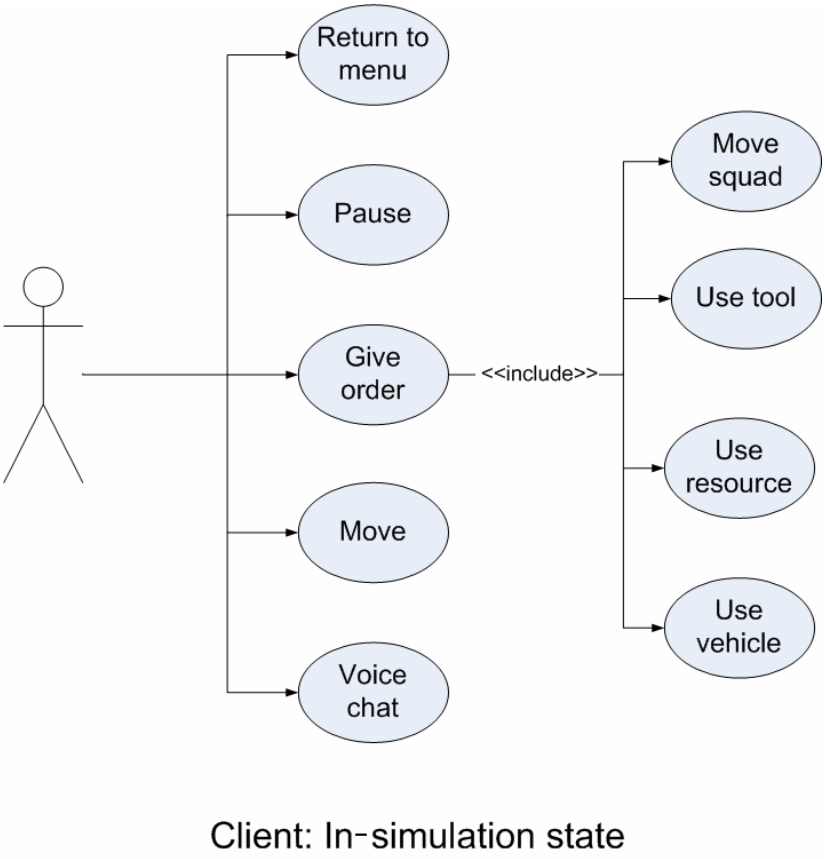
This is the main screen the server will face when starting the program. The screen has 3 buttons.
Create: Selecting this button will initialize the simulation. The clients then will be able to connect the server by entering the IP.
Options: Set program options such as graphics, audio, controller and simulation options.
Exit: Terminates the program.



Server: Menu state

3.3.3. Client In-Simulation State Use Case

This diagram explains what the client can do while the simulation is running.
Return to Menu: Selecting this will switch the current state to the user menu state.
Pause: Selecting this will send a pause request to server and the simulation's current state will be switched to pause state.
Give Order: The client will be able to give commands such as: Move squad, use tools, use resources and use vehicle. If the simulation is at "Passive Mode", these commands shall be given to the facilitator via voice chat and are executed by the facilitator. If the simulation is at "Active Mode", these commands can be given by either voice chat or user interface.
Move: Moves the user on the map using the keyboard and the mouse.
Voice chat: Users will be able to communicate with the facilitator and other users by voice chat.

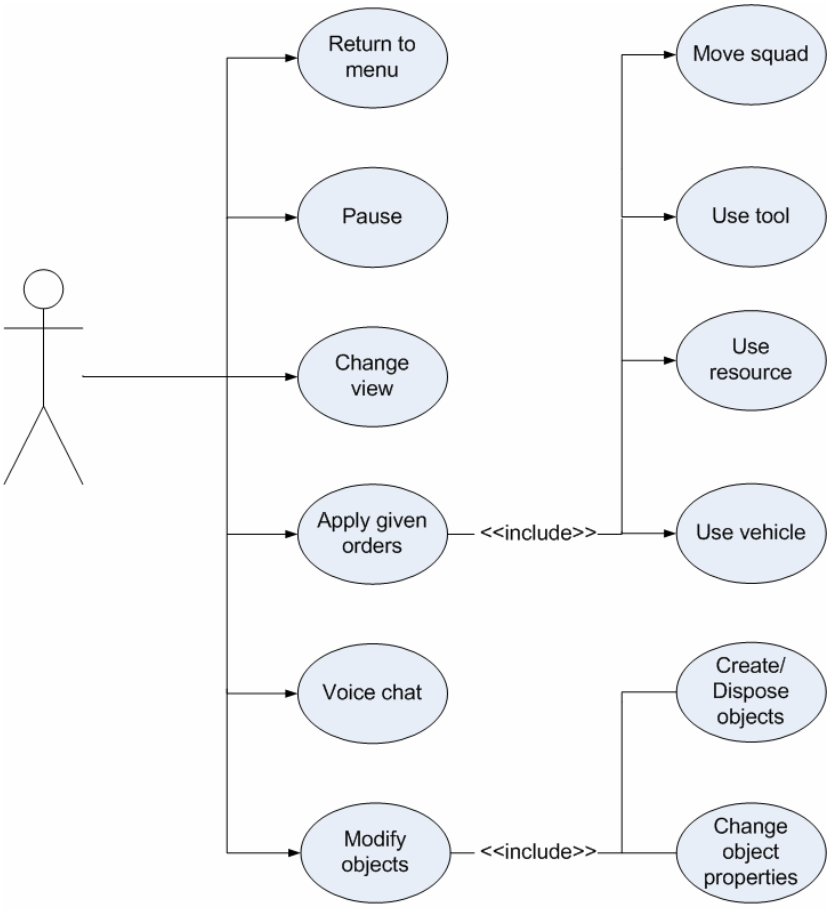


Client: In-simulation state

3.3.4. Server: In-Simulation State Use Case

This diagram explains what the server can do while the simulation is running.
Return to Menu: Selecting this will switch the current state to the server menu state.
Pause: The simulation's current state will be switched to pause state.
Change View: By default, facilitator will begin at "Free View", in which he can move in any direction without any constraint. He also can view the scene directly from any client's view at "User View". Another view mode is the "Map View", in which the facilitator sees the clients as little symbols on a full screen map.

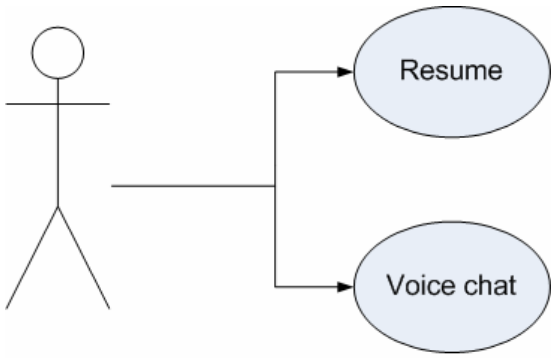
Apply given orders: Facilitator can, at any time, execute orders such as: Move squad, use tools, use resources and use vehicle. At "Passive Mode", these orders may be executed only by the facilitator.
Voice chat: Facilitator can communicate with the clients via voice chat.
Modify objects: Depending on the flow of the scenario, facilitator can create or dispose objects, or modify attributes of an object.



Server: In-simulation state

3.3.5. Server and Client: Pause State Use Case

This diagram explains available actions for the server and the user when the simulation is paused.
Resume: Selecting this will switch simulation's current to in-simulation state.
Voice chat: During the Pause State, users will be able to voice chat.



Server and Client: Pause state

4. Risk Management

4.1. Probable Risks

Development tools, libraries : In time, some bugs or insufficiencies in libraries may be realized, or tools used for development (e.g. Visual Studio, level editors..) may lack required capabilities.

Large project size: Due to developers' lack of experience, the size of the project may be estimated smaller than it should be. This can lead late submissions and the deadlines may need to be pushed forward.

Integration: During the integration of components, some parts of the project may not work as expected. Depending on where the inconsistency occured, the regression of the project may be more than affordable.

Badly designed test systematics: Improper testing may not cover some existing bugs. For the stability of the project testing should be done according to a well designed systematic.

Customer disagreement: Due to wrong requirement analysis, the final product may not be exactly what the customer wanted.

Loss of staff: Extreme level of disagreement may cause a member of the group to leave, which may lead a catastrophic results in the overall progress.

4.2. Risk Table

Risk	Mitigation/Monitoring	Management	mpact Rate
Development tools, libraries	Sufficient research should be done for required tools	If possible, try alternate solutions for incompatible components	2
Large project size	Good observation of the whole project	Increase work time	1
Integration	Sufficient research should be done for required tools	Modify buggy parts of the code	2
Badly designed test systematics	Good knowledge of testing process	Turn back and re-implement bugs	1
Loss of staff	Good communication, team spirit	Rearrangement of task among members	3
Customer disagreement	Well defined specifications	Turn back and redesign intended parts	1

Impact Rates

1: Negligible

2: Critical

3: Catastrophic

5. Schedule

	Task Name	Duration	Start	Finish
1	Scenario Design	15 days	Mon 15.10.07	Fri 02.11.07
2	Concept Research	15 days?	Mon 15.10.07	Fri 02.11.07
3	Requirement Analysis Rep	15 days?	Tue 16.10.07	Sun 04.11.07
4	Server Core Design	30 days?	Mon 03.12.07	Fri 11.01.08
5	Client Core Design	30 days?	Mon 03.12.07	Fri 11.01.08
6	Graphics Design	46 days?	Sun 04.11.07	Fri 04.01.08
7	Network Design	46 days?	Sun 04.11.07	Fri 04.01.08
8	AI Design	46 days?	Sun 04.11.07	Fri 04.01.08
9	Physics Design	46 days?	Sun 04.11.07	Fri 04.01.08
10	Graphics Demo	15 days?	Mon 24.12.07	Fri 11.01.08
11	Network Demo	15 days?	Mon 24.12.07	Fri 11.01.08
12	Physics Demo	15 days?	Mon 24.12.07	Fri 11.01.08
13	AI Demo	15 days?	Mon 24.12.07	Fri 11.01.08
14	Audio Demo	15 days?	Mon 24.12.07	Fri 11.01.08
15	Map Design	55 days	Mon 17.12.07	Fri 29.02.08
16	Models Design	65 days?	Mon 03.12.07	Fri 29.02.08
17	User Interface Design	25 days?	Mon 10.12.07	Fri 11.01.08
18	Initial Design Report	1 day?	Fri 30.11.07	Fri 30.11.07
19	Integration Demos	11 days?	Mon 19.11.07	Mon 03.12.07
20	Final Design Report	1 day?	Fri 11.01.08	Fri 11.01.08
21	Prototype Implementation	10 days	Mon 31.12.07	Fri 11.01.08
22	Prototype Demo	0 days	Fri 18.01.08	Fri 18.01.08
23	Graphics Implementation	55 days?	Mon 21.01.08	Fri 04.04.08
24	Client Core Implementation	60 days?	Mon 21.01.08	Fri 11.04.08
25	Server Core Implementatio	60 days	Mon 21.01.08	Fri 11.04.08
26	Input Manager Implementat	10 days?	Mon 21.01.08	Fri 01.02.08
27	User Interface Implementat	15 days?	Mon 17.03.08	Fri 04.04.08
28	AI Implementation	45 days?	Mon 04.02.08	Fri 04.04.08
29	Physics Implementation	17 days	Mon 03.03.08	Tue 25.03.08
30	Server Network Implement	60 days	Mon 21.01.08	Fri 11.04.08
31	Client Network Implementa	60 days?	Mon 21.01.08	Fri 11.04.08
32	Voice Chat Implementation	30 days?	Mon 03.03.08	Fri 11.04.08
33	Alpha Testing	21 days?	Mon 31.03.08	Mon 28.04.08
34	Beta Testing	27 days?	Tue 29.04.08	Wed 04.06.08
35	Quality Assurance	14 days	Tue 13.05.08	Fri 30.05.08
36	Documentation	15 days?	Mon 12.05.08	Fri 30.05.08
37	Final Product Presentation	10 days?	Mon 19.05.08	Fri 30.05.08
38	Final Product Release	6 days?	Fri 23.05.08	Fri 30.05.08