



# **MAÇA YAZILIM**

**ONLINE VIRTUAL TEAM COLLABORATION PLATFORM  
WITH 3D GRAPHICS**



**CENG 491**  
**Initial Design Report**

**METU**  
**2007**

# Table of Contents

<b>1. INTRODUCTION .....</b>	<b>4</b>
<b>1.1 Purpose of the Document.....</b>	<b>4</b>
<b>1.2 Scope of the Document .....</b>	<b>4</b>
<b>2. PROJECT DESCRIPTION .....</b>	<b>4</b>
<b>2.1 Detailed Problem Definition .....</b>	<b>4</b>
<b>2.2 Project Features .....</b>	<b>5</b>
<b>2.3 Design Constraints and Limitations .....</b>	<b>6</b>
<b>2.4 Design Goals and Objectives .....</b>	<b>6</b>
<b>3. SYSTEM AND TOOL CHOICES .....</b>	<b>7</b>
<b>4. GRAPHICAL USER INTERFACE DESIGN .....</b>	<b>8</b>
<b>4.1 Initial Menu .....</b>	<b>9</b>
<b>4.2 Main Menu.....</b>	<b>9</b>
<b>4.3 Paused Mode Menu .....</b>	<b>11</b>
<b>4.4 Disconnect Menu .....</b>	<b>11</b>
<b>4.5 Final Statistics Menu .....</b>	<b>12</b>
<b>5. OVERALL ARCHITECTURE.....</b>	<b>13</b>
<b>6. DETAILED DESIGN.....</b>	<b>14</b>
<b>6.1 Data Flow Diagrams and Data Dictionary.....</b>	<b>14</b>
<b>6.2 Use Case Diagrams.....</b>	<b>21</b>
<b>6.3 Class Definitions and Diagrams.....</b>	<b>24</b>
<b>6.3.1 Simulation Module .....</b>	<b>25</b>
<b>6.3.2 Network Module.....</b>	<b>27</b>
<b>6.3.3 Graphics Module .....</b>	<b>28</b>
<b>6.3.4 AI Module.....</b>	<b>29</b>
<b>6.3.5 Physics Module.....</b>	<b>30</b>



6.3.6 Sound Module .....	31
6.3.7 Agents .....	32
6.3.8 Objects .....	33
6.4 State Transition Diagrams.....	34
6.4.1 Simulation States Diagram .....	34
6.4.2 Object Interaction States Diagram .....	35
6.4.3 Menu States Diagram.....	36
6.5 Activity Diagrams .....	37
6.6 Sequence Diagrams.....	41
6.7 ER Diagram .....	43
7. SYNTAX SPECIFICATIONS .....	43
7.1 File Naming Conventions.....	44
7.2 Classes .....	44
7.3 Method and Function Definitions .....	44
7.4 Variable Naming Conventions .....	44
7.5 Comments .....	44
8. PROJECT SCHEDULE .....	45
8.1 Current Stage of the Project .....	45
8.2 Future Work .....	46
8.3 Gantt Chart.....	47



## **1. INTRODUCTION**

The initial design is a constitutive process that identifies the future implementation phase and the justification of the main concerns related to the project in some basic perspective. That is what makes the initial design report so critical for the project. The design decisions mentioned in this report are not strictly unchangeable, but the points explained in the report can be seen as a guide to follow in the following phases of the project.

### **1.1 Purpose of the Document**

The purpose of this document is to explain initial design process of the project. In addition, the data types and the use of them in the context of the project are defined in detail to make the whole system more understandable and more concrete.

### **1.2 Scope of the Document**

The document comprises of simulation flow, environment, user interfaces, data design, architectural design, design constraints, development schedule, current design level and future works planned so far.

In the document, many diagrams are used to visualize the system components including the relations between them.

## **2. PROJECT DESCRIPTION**

The project is to develop an educative 3D virtual team collaboration platform. In the project, a ship emergency simulation is used to respond the project specifications. Users will have three different alternatives for character selection and different communication alternatives to strengthen the collaboration with other users. The users will take place in a scenario that is devised to be in benefit of the mankind.

### **2.1 Detailed Problem Definition**

In the simulation, users will cope with a fire on the ship. According to the character type they chose before initiating the simulation, they will decide what to do on that occasion. During the event flow, there will be exactly three persons using the program, each having a different role.



The available roles will be: (1) captain, (2) sea rescue chief and (3) first-aid chief. The user will choose one of them and think of him as if he is really having that emergency case, so he will gain experience from an imaginary event and will not repeat his simulation mistakes in real life. That is a realistic goal for our project because the event flow that will be provided in the simulation will not be very simple in order to prevent the users from memorizing the steps. Instead, for every event there will be different routes for users and these routes will give different results.

The fire is recognized by the fire alarm that is only given to the captain's central station not to make passengers panic or cause a chaos. Then the captain -the coordinator in the simulation- will communicate with his assistants (human resource) and then inform the other chiefs on the ship. The chief of rescue team will intervene in the fire and try to evacuate passengers. The crew will help him on his action. In the course of events, some passengers will get injured and the last character's mission is to help them as much as possible. Again, the crew will be working with him; they are obliged to process his orders.

As stated in the requirement analysis report the resources are shared among the characters like below:

- Resource of the coordinator (captain): assistants.
- Resource of rescue team chief: crew, extinguisher, cutting and piercing equipment, special protective outfits, lifeboats.
- Resource of "chief of first-aid": health officers, medical equipment, wheeled bed.

Each of the three characters will have the first person view while the facilitator has both the first person views of three characters and a third person view.

Two modes will be available for users in the simulation. The first mode will not require a computer experience background; however the second mode will require a basic level of experience with computers. The user interaction methods will be so simple that the program in the first mode will work with only mouse clicks and necessary devices for communication.

## **2.2 Project Features**

The features to be provided:

- 3D Computer graphics
- Text messaging



- Voice communication
- Simple, easily understandable user interface
- Evaluation of the simulation performance

## 2.3 Design Constraints and Limitations

Maca Yazilim is preparing this simulation project for senior project course of Computer Engineering Department, METU. This brings the most inevitable constraint – time. This project must be finalized before June; so development phase has duration of eight months, and two months have passed. Maca Yazilim team members are all senior students and have other projects, and courses that do not directly help this project, so time must be spent carefully. Beside the main deadline, there are many mini deadlines such as reports, presentations, demos that should be met.

Network security is one of the constraints that are underlined by the company. Encryption will be used in the further phase of the project, to provide the security of the network module. Since this property will be easily attached to the network module, it is not mentioned in this phase.

Performance is another constraint that must be concerned carefully in order to use resources effectively. In network module, only changed variables is sent to the server to inform the simulation module. Also in graphics module, frame per second rendering value will not go beyond the human vision capabilities in order not to waste resources.

Team members will use many libraries during development stage like torque, OGRE... These libraries will decrease the time that will spend in the implementation phase, but their limitations would directly affect the project, and become project's limitations. Team members will try to manage these limitations by using qualified libraries.

## 2.4 Design Goals and Objectives

Usability: The main concern for the simulation developing is the target user's computer capability. A simulation must be easily usable by a related person; however he did not know anything about the computers. As our purpose is to educate these people for the real life, they should be provided with the equipments they use in real life. Unfortunately, this is not a keyboard or mouse. The voice communication technique and a facilitator bring up an easily adaptable simulation environment.



Virtual Reality: In order to make users concentrate to the situation, it is inevitable to build up a realistic environment. The virtual reality should be supplied by not only graphical realism but also the appropriate physics rules or human behaviors. Therefore, a physics engine and a fire dynamics simulator were used. The virtual reality will surely increase the benefits of the simulation since it will be a close test of real life.

Reliability: The reliability requirement is always considered as a key requirement in the project team. Developing a bug free simulation is an important aim. Testing and debugging should be done very carefully in order to achieve this aim. The realistic environment of the simulation should not be sabotaged by the buggy codes.

Security: The online usage of simulation points out the importance of security. The carefully developed code should block out threats to the users during and after the simulation.

Portability: The simulation would not be working on a Linux operating system. However, the simulation will work on Windows operating system.

### **3. SYSTEM AND TOOL CHOICES**

System and tool choices can be grouped into four broad categories:

Operating System Choice: Windows XP Operating System.

Hardware Choice: P4 class processor or equivalent, 256MB of memory, Graphics card and Direct3D support, sound card, internet or network connection, devices for voice communication.

Open Source Engine Choices: OGRE as rendering engine, openTNL for networking and voice communication, ODE for physics engine, FDS (Fire Dynamics Simulator) to represent fire characteristics, emanation and smoke emission.

It should be noted that the team decided to use Torque Networking Library instead of DirectPlay. This modification was inevitable since Microsoft depreciated DirectPlay in the recent DirectX development kits. This depreciation resulted to an inconsistency between documentations and the SDK's. The team tried to develop simple applications with DirectPlay but these attempts failed due to these inconsistencies. The team decided to use DirectSound as the sound engine, however if it causes problems it will be directly replaced with OpenAL or Fmod sound engine.



## 4. GRAPHICAL USER INTERFACE DESIGN



Figure – 1: Main menu screenshot

This screenshot is prepared for giving a general idea about menus and overall graphics. This small graphics application is implemented in OGRE with the help of its built-in ocean application. Ship model (actually which is not a passenger ship) is taken from the internet. This ship model is created by 3D Studio Max and is in .3ds format. “.3ds” format is not supported directly by OGRE, which accepts mesh files. However OGRE provides a converter for this purpose (converts .3ds file to .mesh file). In further stages this tool can be helpful for converting the models that are created by 3D Studio Max. After integrating this ship model to the application, model and camera positions are adjusted with OGRE GUI. After that by using CEGUI, project’s Main Menu is created and positioned. As can be seen, this menu will consist of a title and buttons which are capable of changing simulation flow.



## 4.1 Initial Menu

Initial Menu is designed for, as the name implies, initial configurations of the simulation. This is the first menu that user is faced.

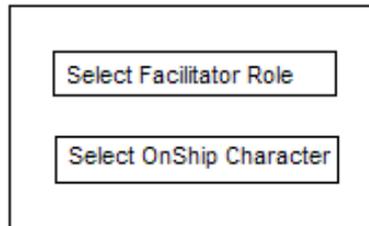


Figure – 2: Initial menu

If the user chooses facilitator mode, its simulator instance will behave as server and other users will connect to facilitator in order to involve the simulation. Facilitator cannot be able to choose simulation mode, start or replay simulation, so after selecting facilitator mode, another menu (different from main menu) will be shown. In this menu, facilitator will be able to see the on-ship characters state (ready / not ready).

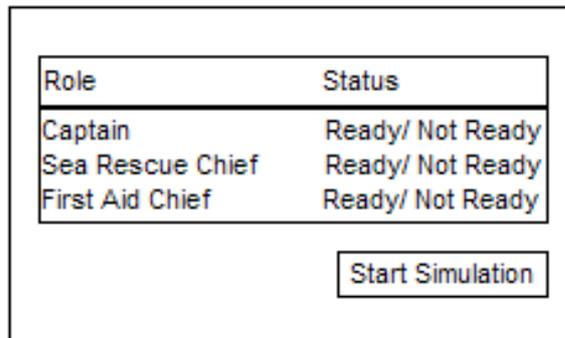


Figure – 3: Connection Status Screen

After on ship characters become ready, facilitator starts the simulation.

If the users choose on-ship characters, they will be directed to Main Menu.

## 4.2 Main Menu

Main Menu is shown after client / server attributes of the created instances are become certain and is related to client side application.



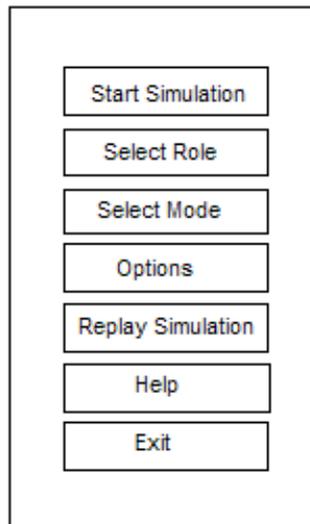


Figure – 4: Main Menu

Users that choose on ship characters must choose their certain character – captain, first aid chief, or sea rescue chief. To prevent from duplicated roles, previously chosen roles will not be available. This is handled by server-side.

Mode selection is – as stated before, decidable by the users and default mode is 1.

Options consisted of three fields: Graphics, Sound, and Key Board Controls. When Options is selected this menu will occur:

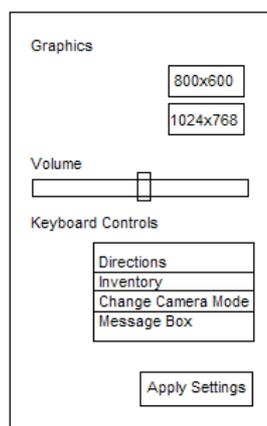


Figure – 5: Options Menu



Graphics session can be used for resolution settings, volume slider can be used for increase/decrease volume level and keyboard controls is used for assigning keys to certain tasks, direction (default 'w', 'a', 's', 'd') , inventory screen shortcut (default 'i'), change camera mode (default 'c'), message box shortcut (default 'm'). Change camera mode is used by facilitator only. Other users can use these keyboard controls only in mode 2.

Changes are activated by "Apply Settings" button.

### 4.3 Paused Mode Menu

This menu will be shown when the simulation is in the suspended state. In this state background is consisted of lastly rendered scene. On this background paused mode menu will be shown as follows:

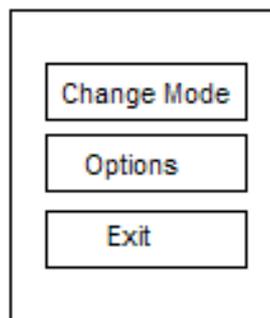


Figure – 6: Paused Mode Menu

The state switches to suspend state when the user pause the simulation or one/more users lost connection.

### 4.4 Disconnect Menu

Disconnect Menu is used when the user lost connection to server. Like Paused Mode Menu there will be a frozen background (lastly rendered scene) and this menu:





Figure – 7: Disconnect Menu

User can chose to reconnect or exit simulation totally.

#### 4.5 Final Statistics Menu

This menu will be shown when the time is up and simulation is ended. In order to give feed back to the users some statistics must be given. With these information users can compare their success between different simulations.

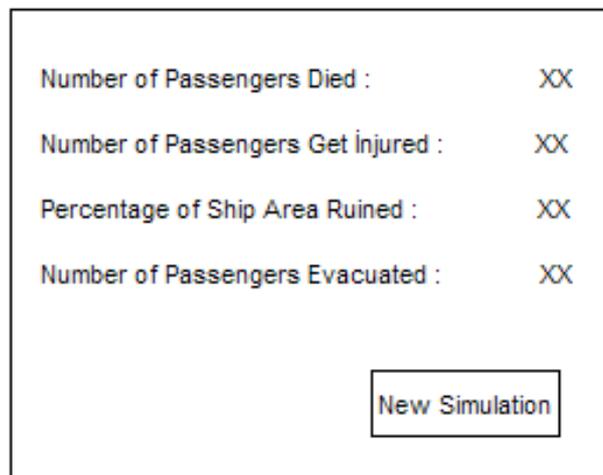


Figure – 8: Final Statistics Menu

New simulation button will start new simulation with the same team (users).



## 5. OVERALL ARCHITECTURE

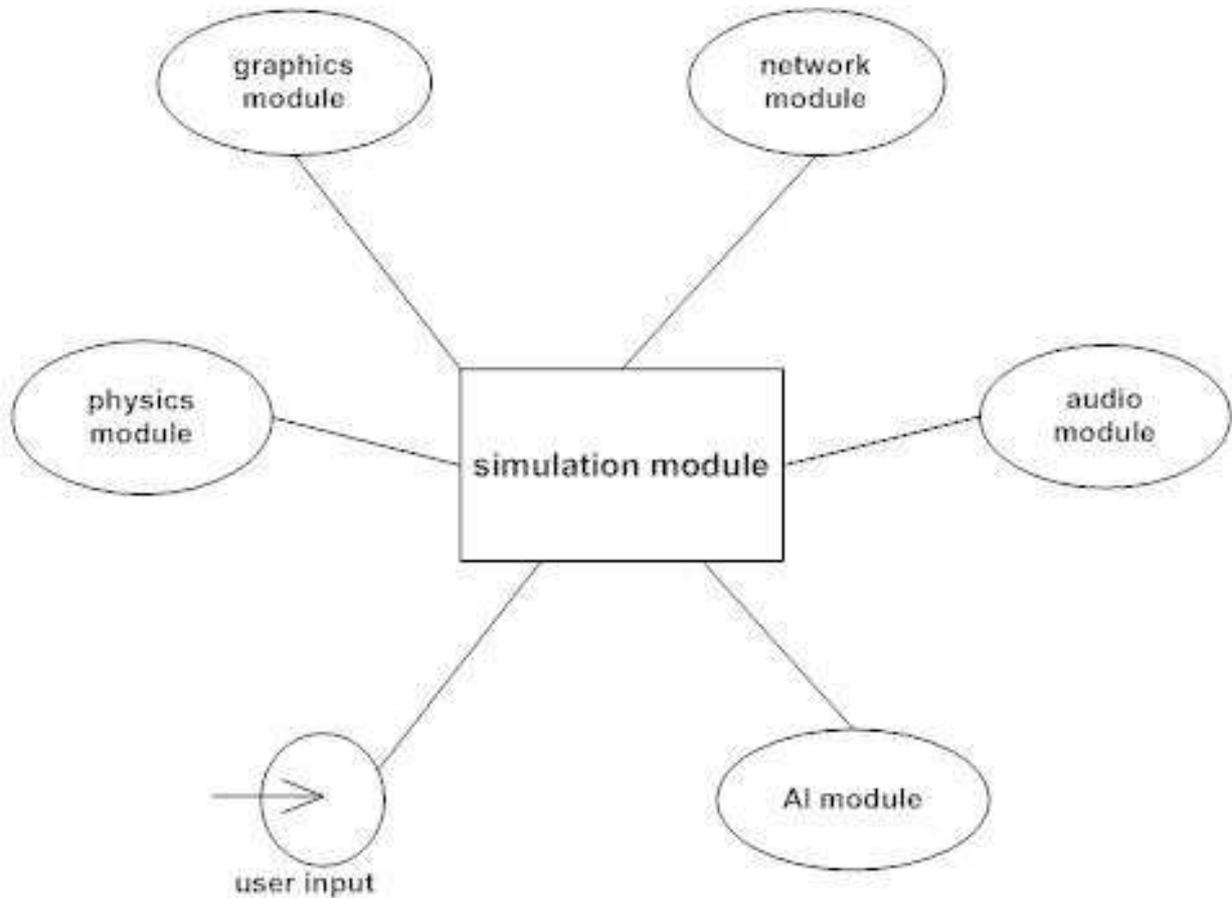


Figure – 9: Overall Architecture

The overall architecture of the project can be examined above. The main organizer part is the simulation module. It has the job of initializing and controlling other components in the simulation flow. Some of the modules have different behaviors for the server and client side; they will be spotted in the detailed design section.

In order to explain briefly:

Graphics Module: It will render the scenes of the player. The objects will be provided from the simulation module.



Network Module: The module will supply the data flow with a client/server approach. All communications will be done via the server. The communication types will be data packets for simulation flow and text or voice messages.

AI Module: The AI module will simulate the non-playing characters and fire.

Physics Module: The physics module will check the actions validity and detect the collisions. All actions will be evaluated in this module and the simulation module will notify the clients whether their actions are approved or not.

Audio Module: The module for playing audios and voice messages. The user will hear the audios that are appropriately selected by the simulation module.

User inputs: The keyboard/mouse inputs will trigger events on the simulation engine. This will not be implemented as a separate module however further considerations can be observed in the detailed design section.

## 6. DETAILED DESIGN

### 6.1 Data Flow Diagrams and Data Dictionary

Level: 0 DFD:

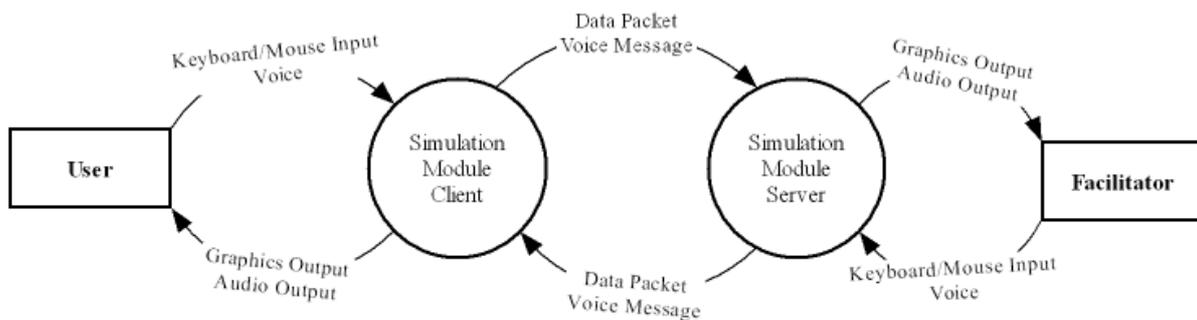


Figure – 10: Level 0 DFD



Level: 1 Simulation Client DFD:

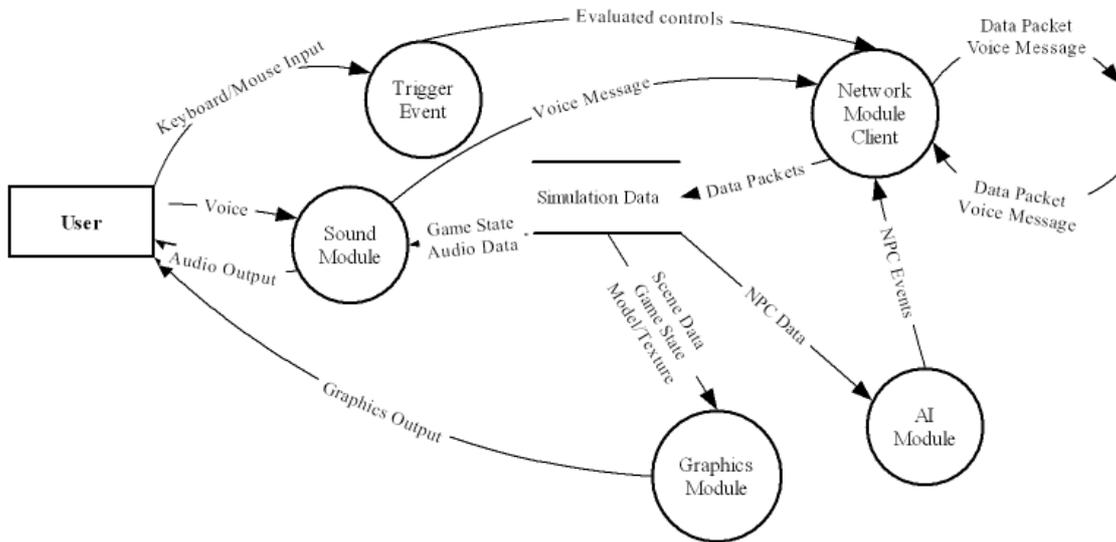


Figure – 11: Level 1 Simulation Client DFD

Level: 1 Simulation Server DFD:

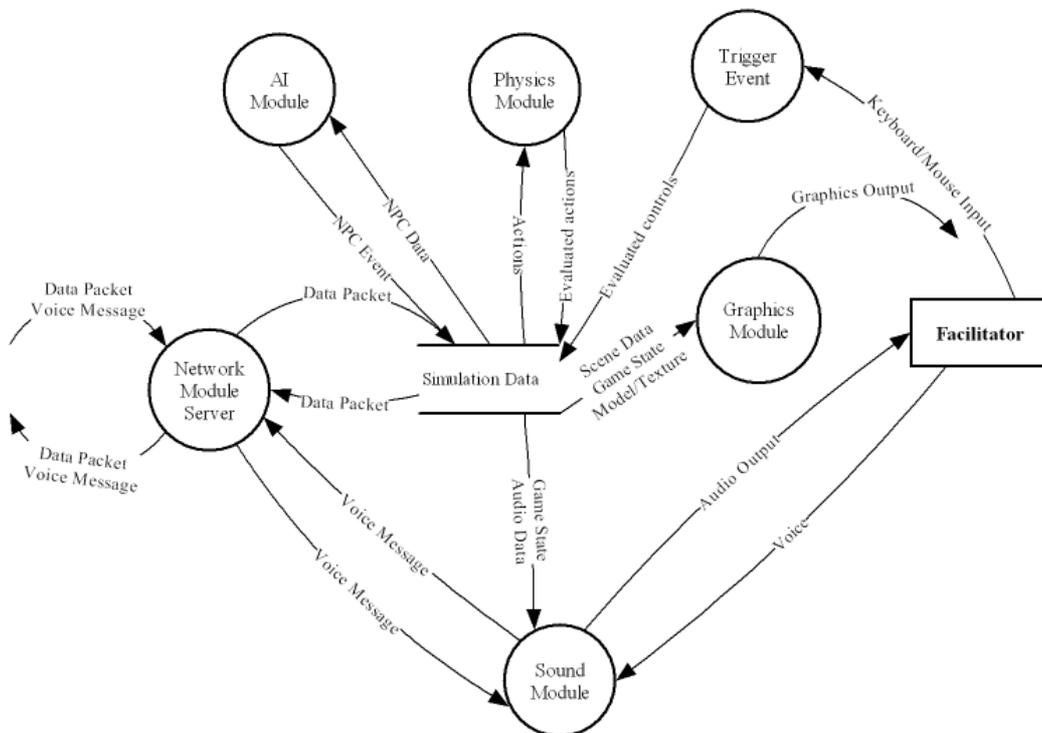


Figure – 12: Level 1 Simulation Server DFD



Level: 2 Graphics Module DFD:

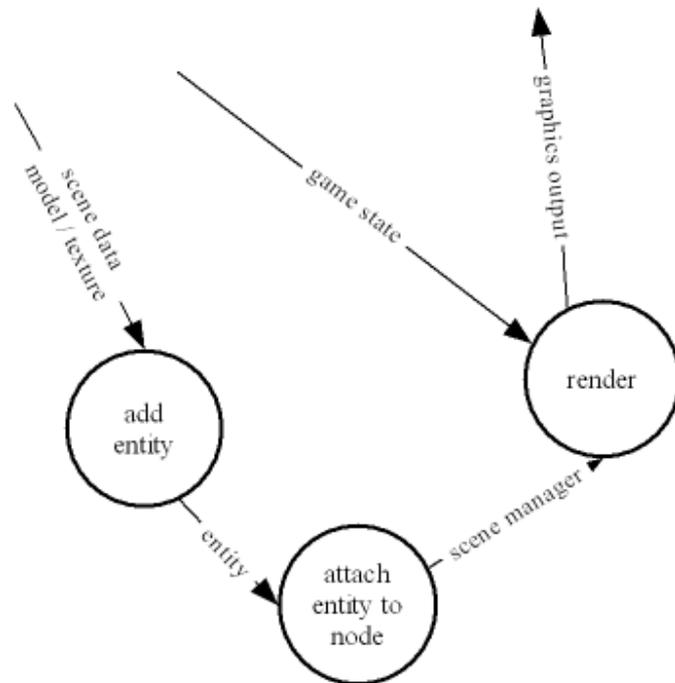


Figure – 13: Level 2 Graphics Module DFD

Level: 2 Network Module DFD:

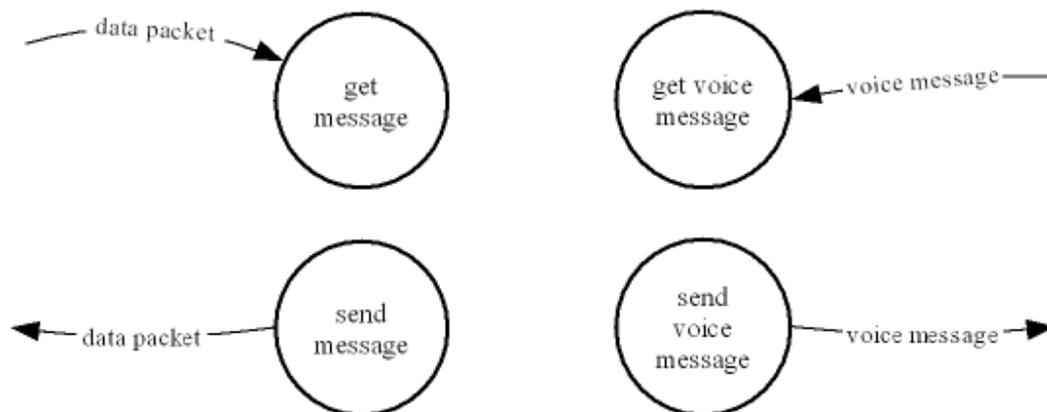


Figure – 14: Level 2 Network Module DFD



Level: 2 Physics Module DFD:

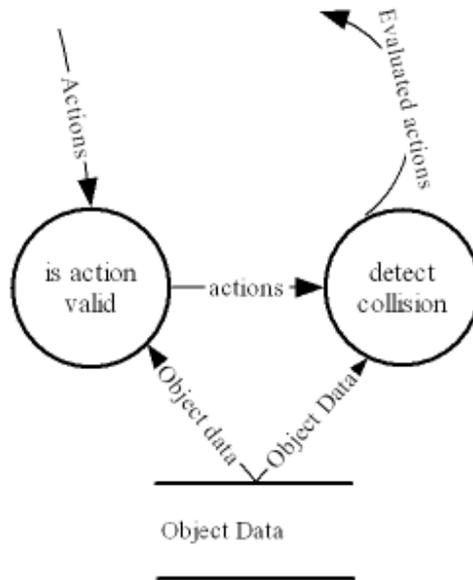


Figure – 15: Level 2 Physics Module DFD

Level: 2 AI Module DFD:

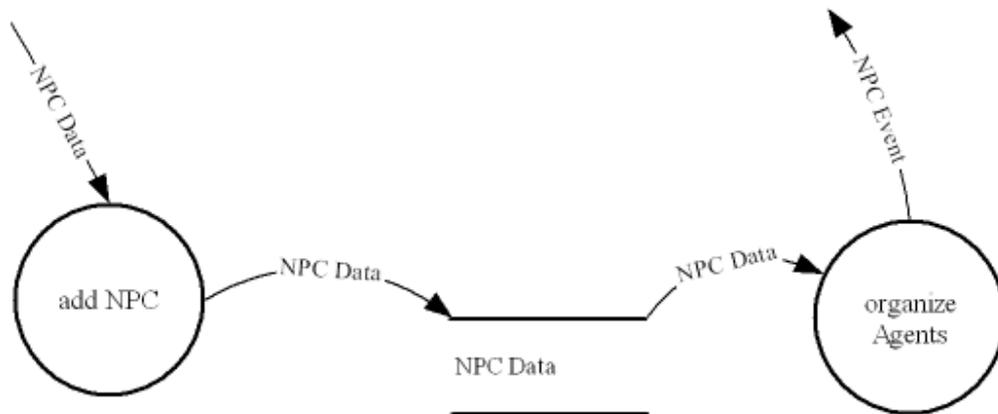


Figure – 16: Level 2 AI Module DFD



Level: 2 Sound Module DFD:

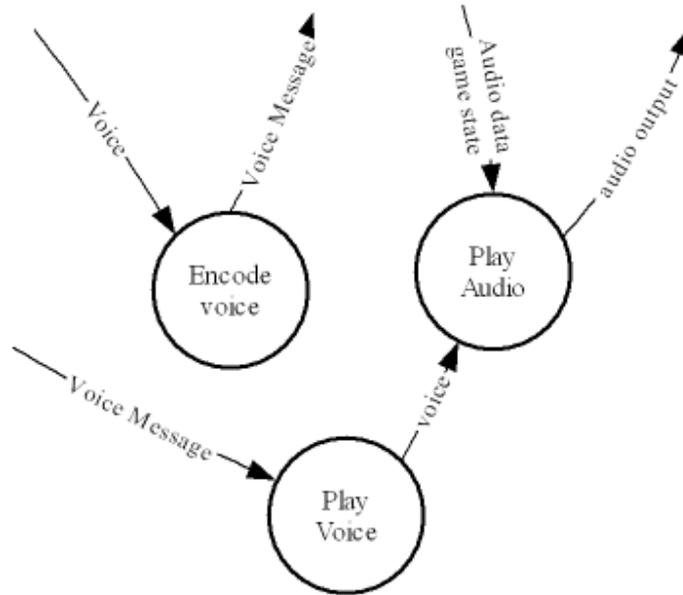


Figure – 17: Level 2 Sound Module DFD

Data Dictionary:

Name	Keyboard/Mouse Input
Where used	Output of User, input of SimulationModule (Level: 0)
Description	These are user input given during the simulation by keyboard or mouse

Name	Data Packet
Where used	Output of Simulation Client Module, input of SimulationModuleServer (Level: 0)
Description	The structure that transfers data between server and client

Name	Voice Message
Where used	Output of Simulation Module Client, input of Simulation Module Server (Level: 0)
Description	The byte buffer form of user voice



Name	Graphics output
Where used	Output of Simulation Module Server, input of Facilitator (Level: 0)
Description	The scenes rendered on the user display

Name	Audio output
Where used	Output of Simulation Client Module, input of SimulationModuleServer (Level: 0)
Description	The audios that the user hears

Name	Evaluated Controls
Where used	Output of User, input of SimulationModule(Level:1)
Description	These are user input given during the simulation by keyboard or mouse

Name	NPC Data
Where used	Output of Simulation Data, input of AI Module(Level:1)
Description	The various data of non playing agents

Name	NPC Event
Where used	Output of AI Module, input of Network Module Client(Level:1)
Description	Any action of a non playing agent

Name	Game State
Where used	Output of Simulation Data, input of Sound Module(Level:1)
Description	The game state that is stored in simulation data. General decisions about the loop and the menus



Name	Voice
Where used	Output of User, input of Sound Module(Level:1)
Description	The speech of the user

Name	Actions
Where used	Output of is action valid, input of detect collision(Level:2)
Description	The actions that will be controlled in the physics module

Name	Evaluated actions
Where used	Output of detect collision, input of Simulation Data(Level:2)
Description	The results of the actions evaluated by the physics module

Name	Audio data
Where used	Output of Simulation Data, input of play audio(Level:2)
Description	The audio files stored in the simulation data

Name	Scene data
Where used	Output of Simulation Data, input of add entity(Level:2)
Description	The environment objects data stored in the simulation data

Name	Model/texture
Where used	Output of Simulation Data, input of Graphics Module(Level:2)
Description	The model and texture files stored in the simulation data

Name	Entity
Where used	Output of add entity, input of attach entity to node(Level:2)
Description	An OGRE class used for rendering objects



Name	Scene manager
Where used	Output of attach entity to node, input of render(Level:2)
Description	OGRE class that renders the attached entities

Name	object data
Where used	Output of Object Data, input of is action valid(Level:2)
Description	The simple object class containing information about an object

## 6.2 Use Case Diagrams

Menu Use Case Diagram:

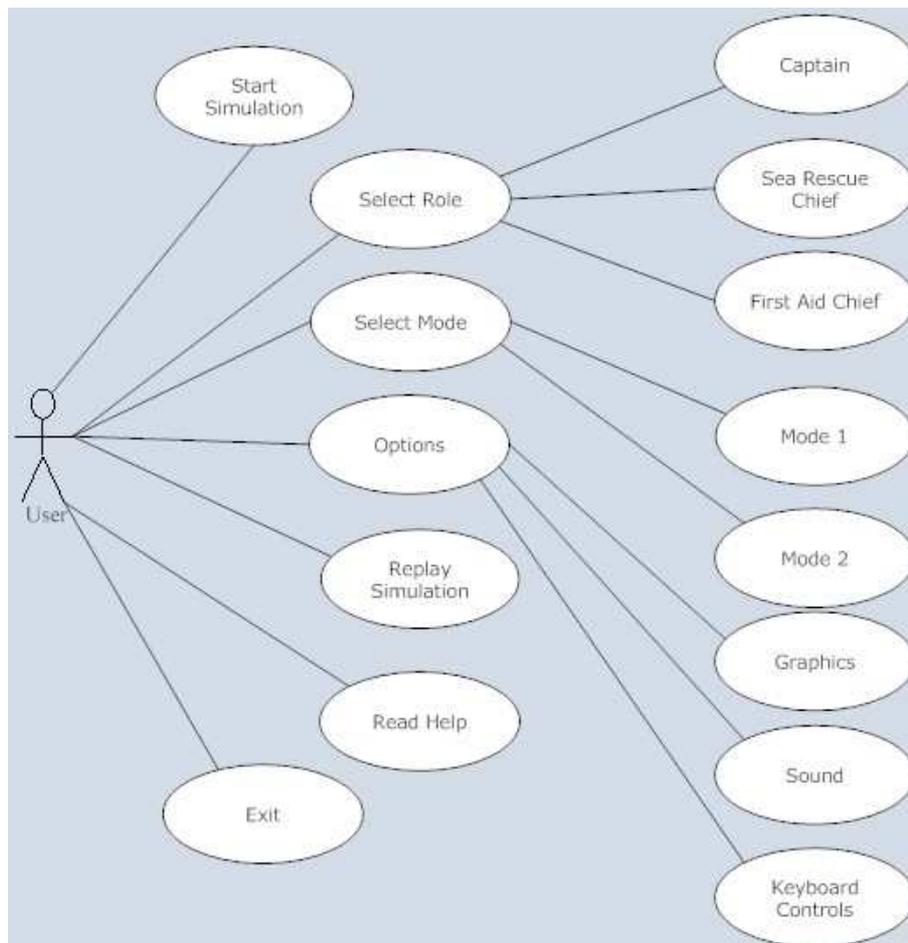


Figure – 18: Menu Use Case Diagram



Facilitator Use Case Diagram:

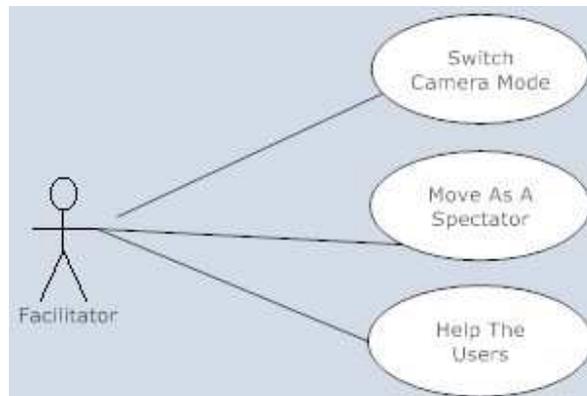


Figure – 19: Facilitator Use Case Diagram

Captain Use Case Diagram:

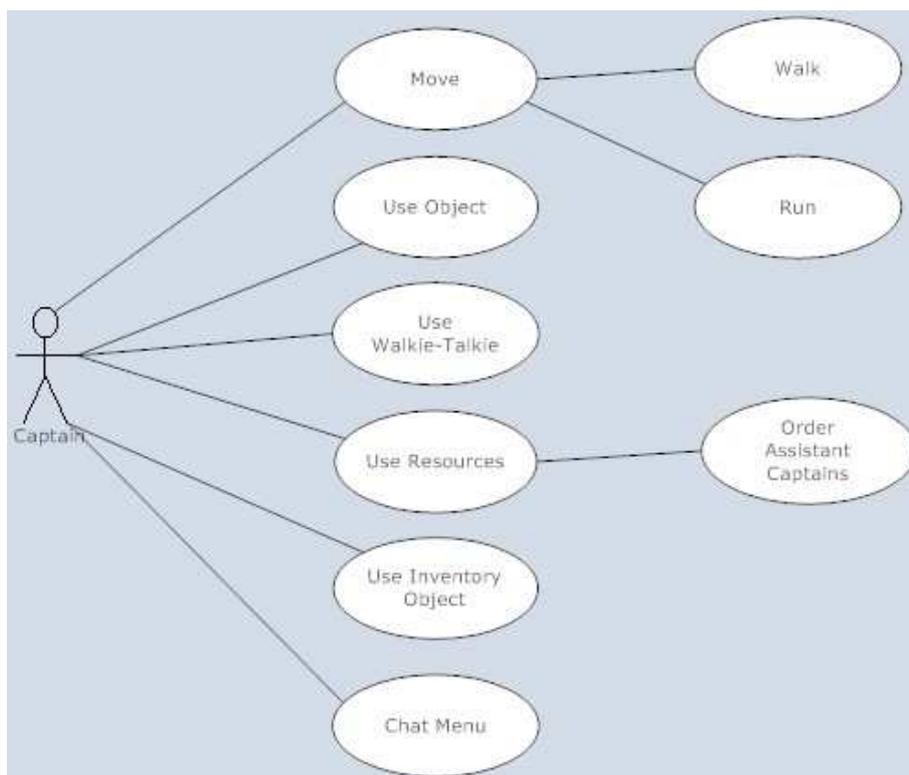


Figure – 20: Captain Use Case Diagram



First Aid Chief Use Case Diagram:

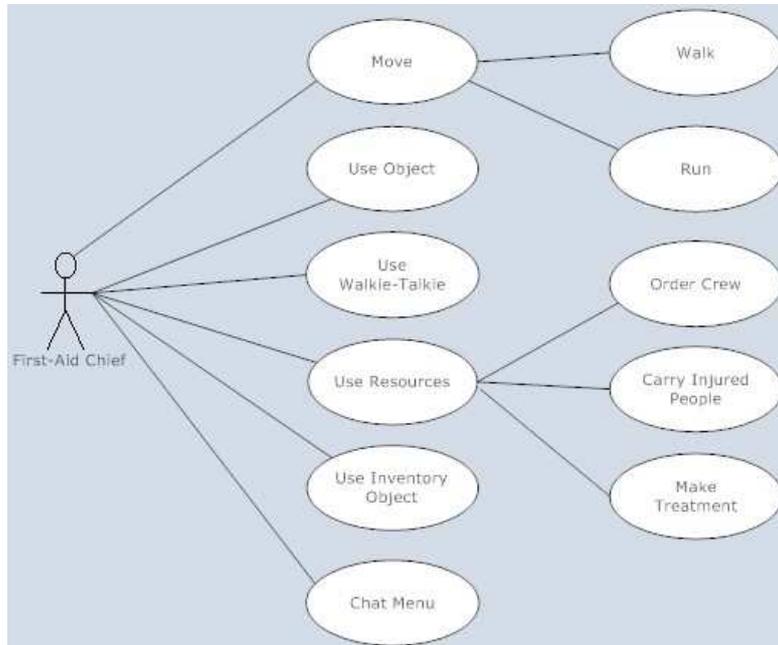


Figure – 21: First-Aid Chief Use Case Diagram

Sea Rescue Chief Use Case Diagram:

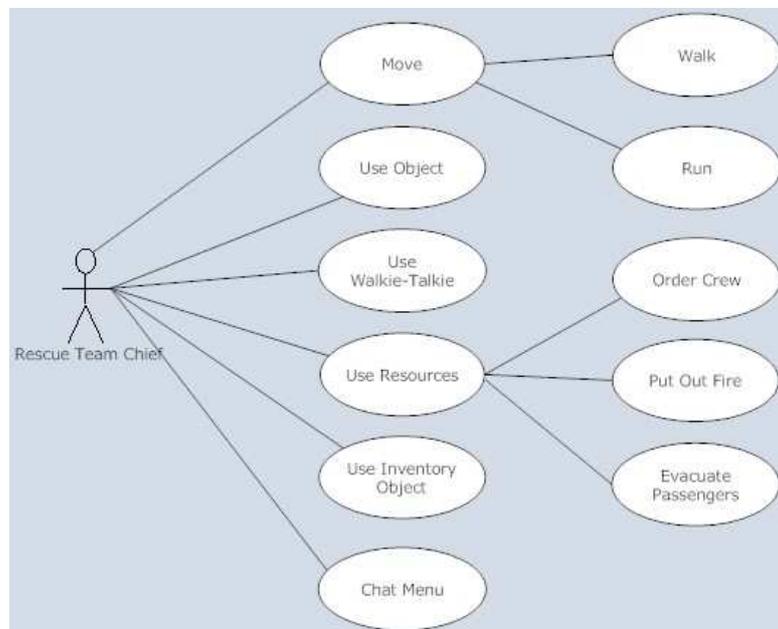


Figure – 22: Sea Rescue Team Chief Use Case Diagram



Passenger Use Case Diagram:

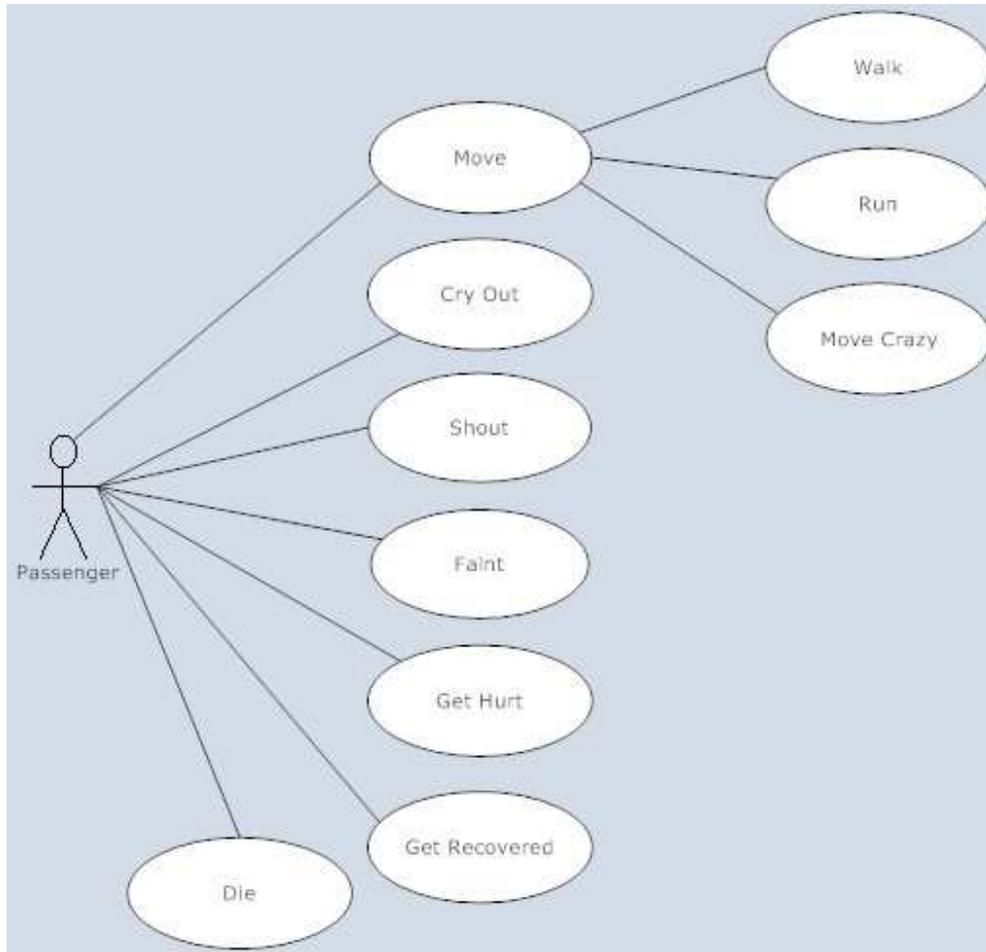


Figure – 23: Passenger Use Case Diagram

### 6.3 Class Definitions and Diagrams

Modules will be implemented in an object oriented paradigm. Specifically, the simulation engine will create instances of other modules. The ‘M’ character before the class names represents “Maca” as a convention of the team.

The relationship between the classes can be examined in the following sections. Some basic get/set methods of the class will be ignored in the diagrams.



### 6.3.1 Simulation Module



Figure – 24: Simulation Module Class Diagram

The simulation engine is the core component of the project that initializes the simulation and controls other modules. Therefore, the user will be provided with a consistent simulation flow. It contains the necessary information about the game state, the objects in the environment and the agents. The simulation engine has a special design which behaves different for server side and client side, therefore the simulation engine can be considered as Simulation Server and Simulation Client according to the user input isClient. In order to supply one application for both server side and client side, it was necessary to merge these two sides in one module. However, depending on the further conditions it can be considered to separate them into two different modules as client and server.

The simulation engine is created when the application starts. It mainly supplies different loops according to the game state like initialization, suspension and flowing. When the user enters



necessary information in the initialization state, the simulation engine creates instances of other modules. The state is switched to suspension on connection losses or pauses. Each module behaves according to the state.

The simulation module for clients initializes its graphics module, network-client module, AI module and sound module. The client firstly provides the connection to the server by networking module and waits for its state change to start simulation. It does not initialize the physics module since these controls will only be considered on server side. However, the AI module for clients organizes the behaviors of human resources of the user character.

The simulation module for server is instantiated for the user type of facilitator, and initializes all the modules. The network-server module creates a connection on the local host and accepts connections from clients. When all other clients connect and send ready message to the server, the server sets the state to flowing and simulation begins. In the flowing state, the game loop gathers information from the clients evaluates them in physics module, handles NPC agents like passengers. The network-server provides the delivery of voice or text messages between clients. At the end of loop the simulation server sends the evaluated events to the clients and clients are modified according to the changes received from the server.

The input handler module is omitted here since the keyboard/mouse inputs will trigger an event on OGRE and they can be handled. However an additional input handler module can be added to the simulation module in order to simplify the jobs done by simulation module.



### 6.3.2 Network Module

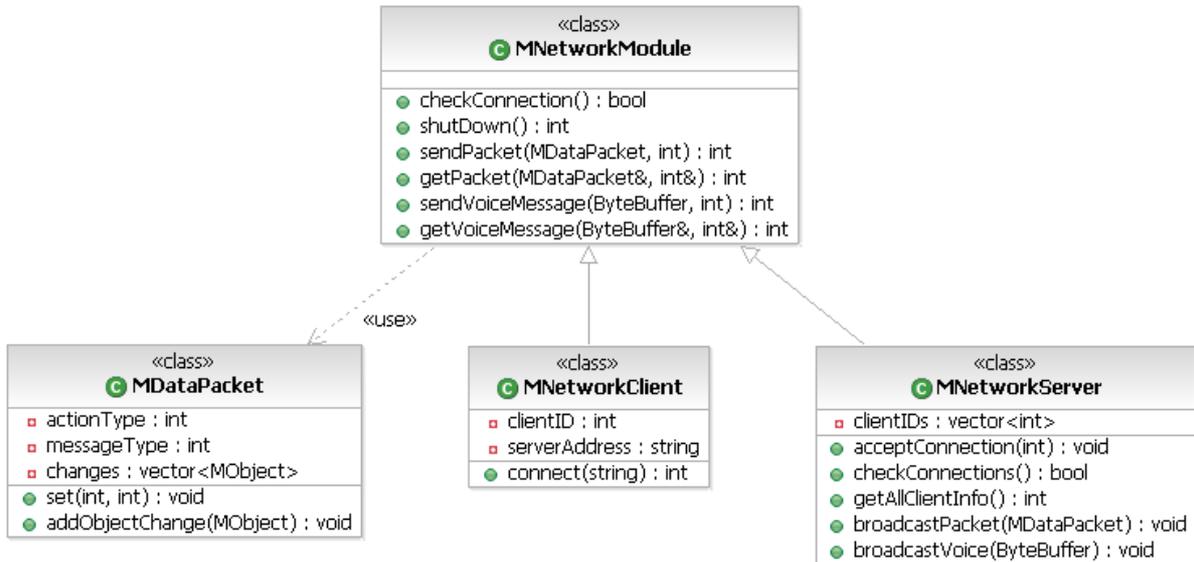


Figure – 25: Network Module Class Diagram

The network module is initialized by the simulation module according to the information given by the user. The network module is created as Network Client for client side and Network Server for the server side. However, both these modules extend a base class Network Module that has common properties for networking. The MDataPacket is the common packet used in communication. The packet is minimized in order to reduce the network traffic, transferring only the changes.

The openTNL library contains API's that provide passing primitive arguments and ByteBuffer type over the network connection. The DataPacket object will be converted to ByteBuffer or separated as arguments to transfer between sides.

The common methods of Network Module are transfers of packages and voice messages, checking the connection link is not broken and shutting down the connection.

The Network Client gathers server address information from the user and it is assigned a unique client ID by Network Server when the connection accepted by the server side.

The Network Server contains all client ids and provides the simulation server the ability of gathering information from the clients and broadcasting packets to them. Also the network server can broadcast a voice message to the all clients by the broadcastVoice method.



### 6.3.3 Graphics Module

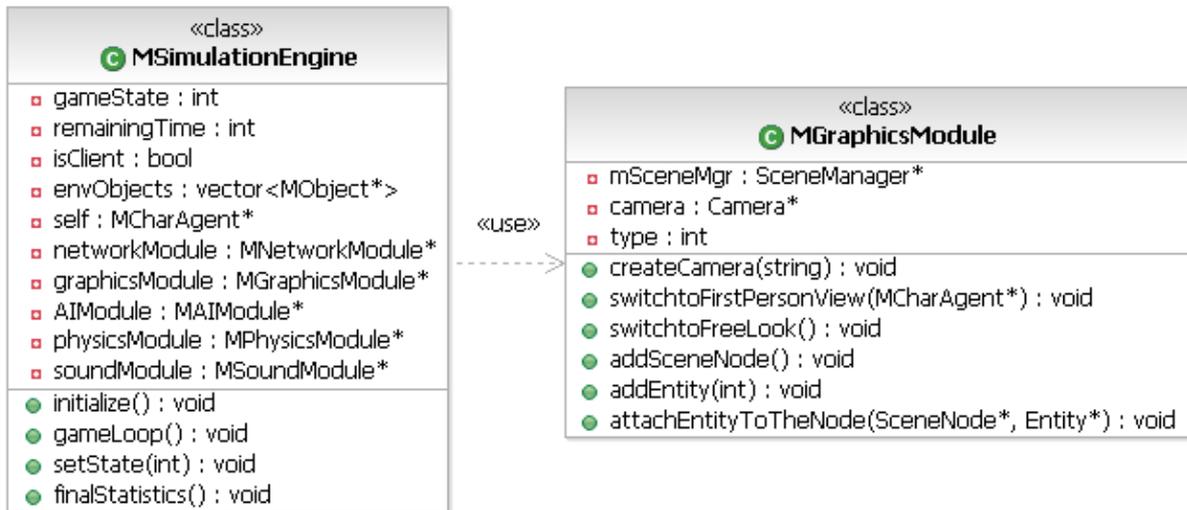


Figure – 26: Graphics Module Class Diagram

The Graphics Module will be developed using OGRE (Object-Oriented Graphics Rendering Engine) and renders the scenes of the users. The SceneManager is a class of OGRE containing the SceneNodes and the SceneNodes contains the Entities attached on them. Each object in the environment will be attached to the scene nodes as entities. The entities will be created using the ID's, models and textures of the objects.

The camera object will be attached to a character agent to provide a first person view. However, the facilitator can switch to free look mode, simply the third person view.



### 6.3.4 AI Module

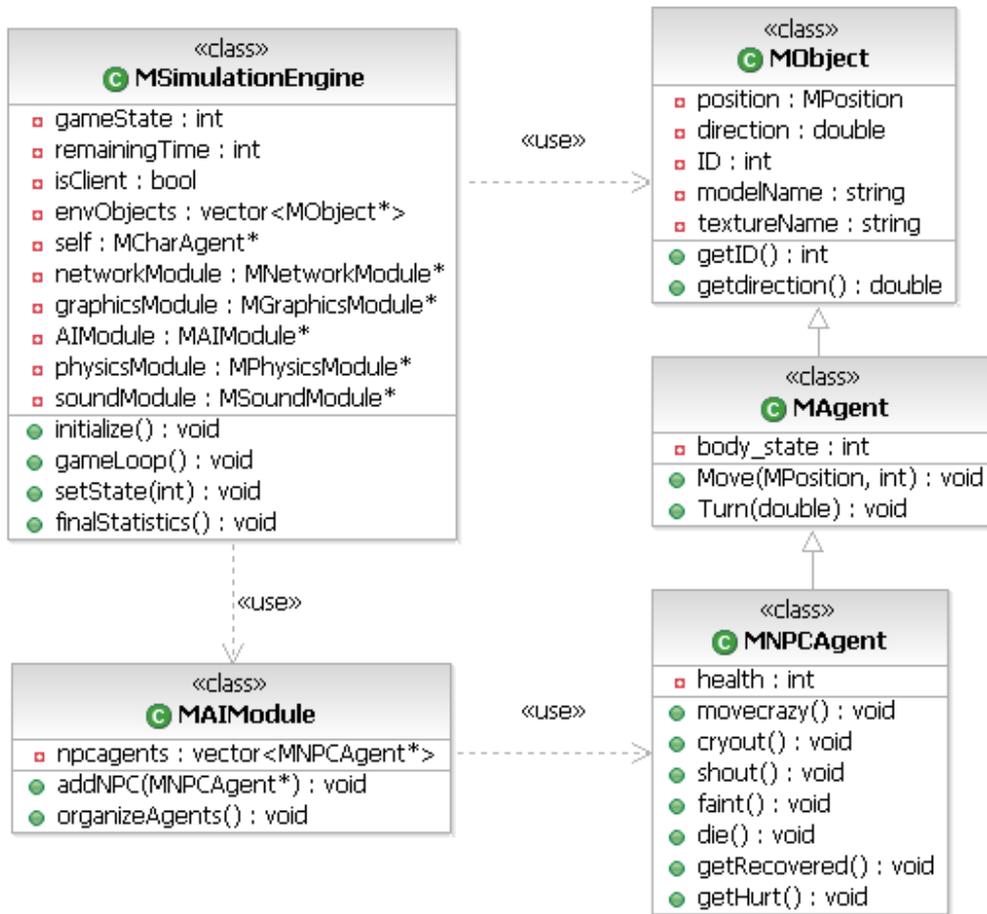


Figure – 27: AI Module Class Diagram

The AI module handles all the NPC agents' behaviors during the game loop. The NPC agent class derives from the Agent class which also derives from the Object class. The possible behaviors of a NPC agent are enumerated and the organizeAgents method of AI module will decide which action will be appropriate for the agent.

The AI module of a client will be responsible from the behaviors of human resources of a character agent. This design choice will be appropriate for improving the intelligence levels of human resources rather than improving all non-playing characters since the human resources may have specific knowledge about their job. The specific events should be handled in different module since a human resource can encounter complex events such as carrying a wheeled-bed with another human resource.



The AI module of the server will organize the all remaining non-playing characters in the main loop according to their current states. On the other hand, the fire will be evaluated with this AI module and the physics module.

### 6.3.5 Physics Module

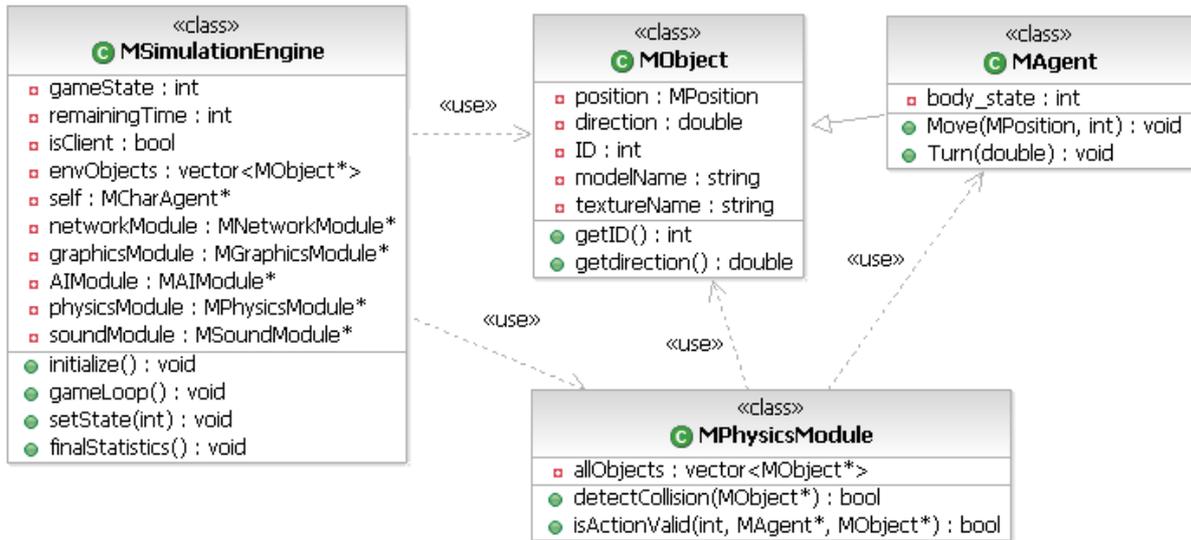


Figure – 28: Physics Module Class Diagram

The physics module, which is defined as an optional requirement for the project, will be developed with an open source library ODE. The main job of this module will be detecting collisions between objects and approving the actions depending on the physics rules. Each action handled in the simulation server will also be approved by the physics module.



### 6.3.6 Sound Module

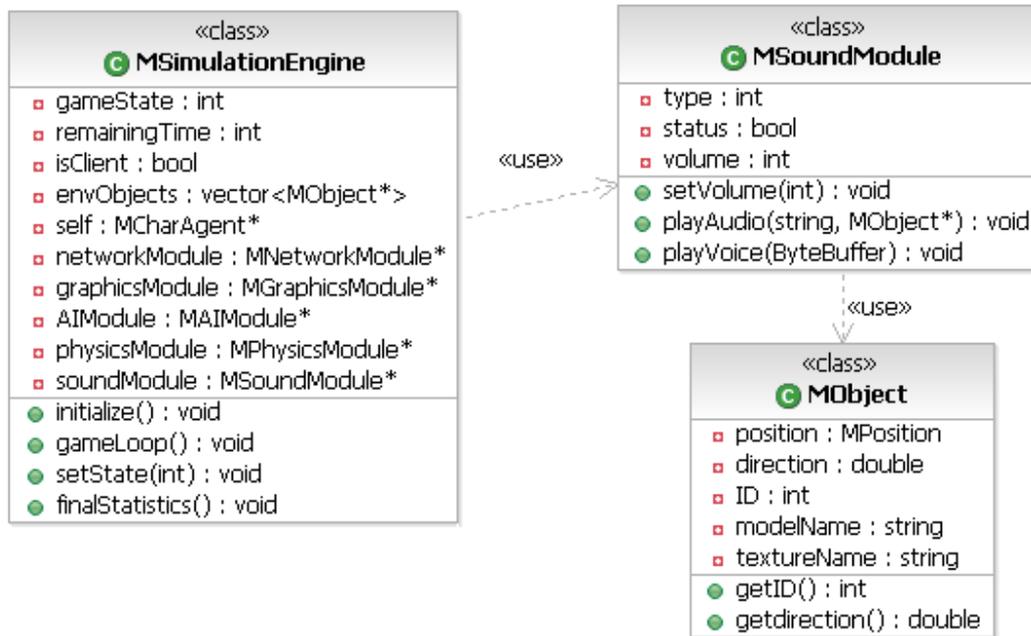


Figure – 29: Sound Module Class Diagram

The sound module will be developed using DirectSound, however as described on the section of system tools, there can be a risk of changing to another library like OpenAL or Fmod. The main role of the sound module is playing the audio files selected by the simulation module according to the state and playing the voice messages posted by the network module. It will decode the voice message and then play.

A powerful ability of the sound module is generating sound effects according to the position of the sound source.



### 6.3.7 Agents

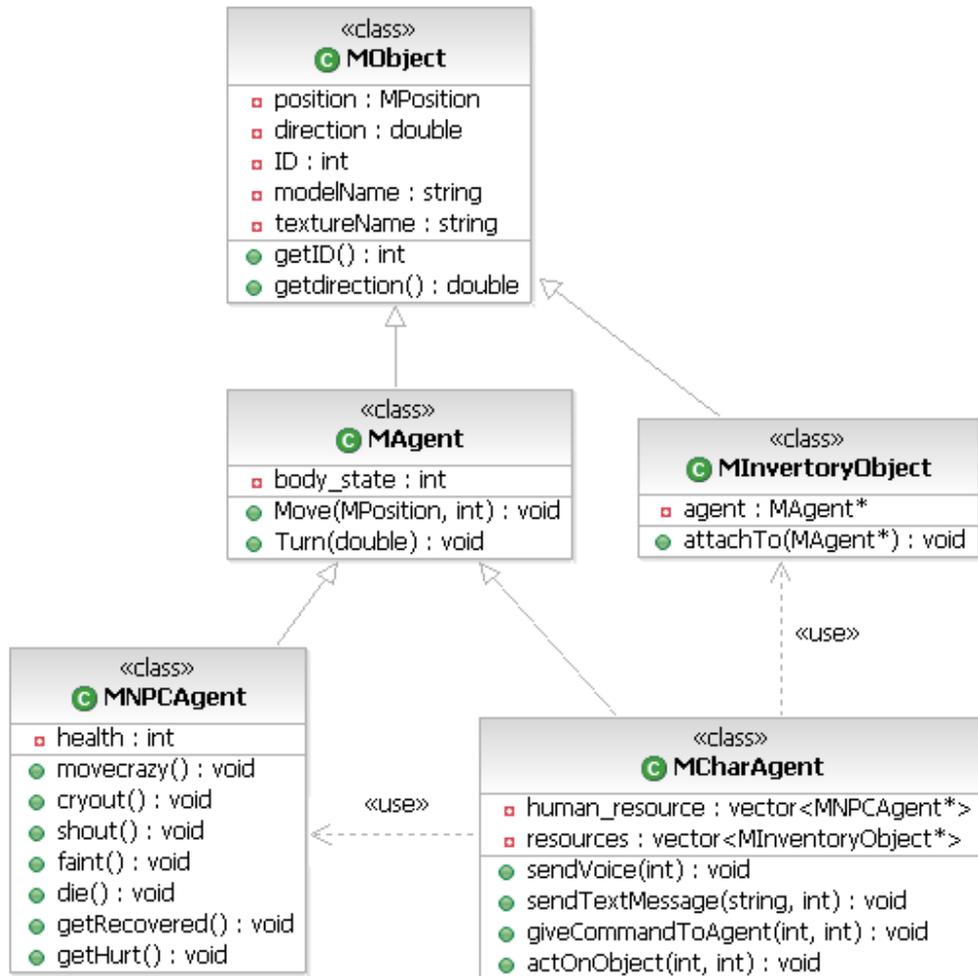


Figure – 30: Agents Class Diagram

The agents of the project will derive from an agent class which also derives from the object class. There are two types of agents: NPC agents and character agents. The NPC agents are the human resources and the passengers, briefly the AI agents. They have predefined enumerated actions and they are handled by the AI module in the game loop.

The character agents are the agent to represent the roles of the player in the simulation. They have NPC agents as human resources and inventory objects. The inventory objects are attached to the characters while initializing the characters.



The character agent uses actOnObject method in order to encounter a predefined action on an object. In order to control its human resources, the method giveCommandToAgent will be used. This method will also be used for directing other non-playing characters for evacuation team. The non-playing characters will be in tendency to obey these commands.

The communications between the character agents will be directed to the network module by the methods sendTextMessage and sendVoice.

### 6.3.8 Objects

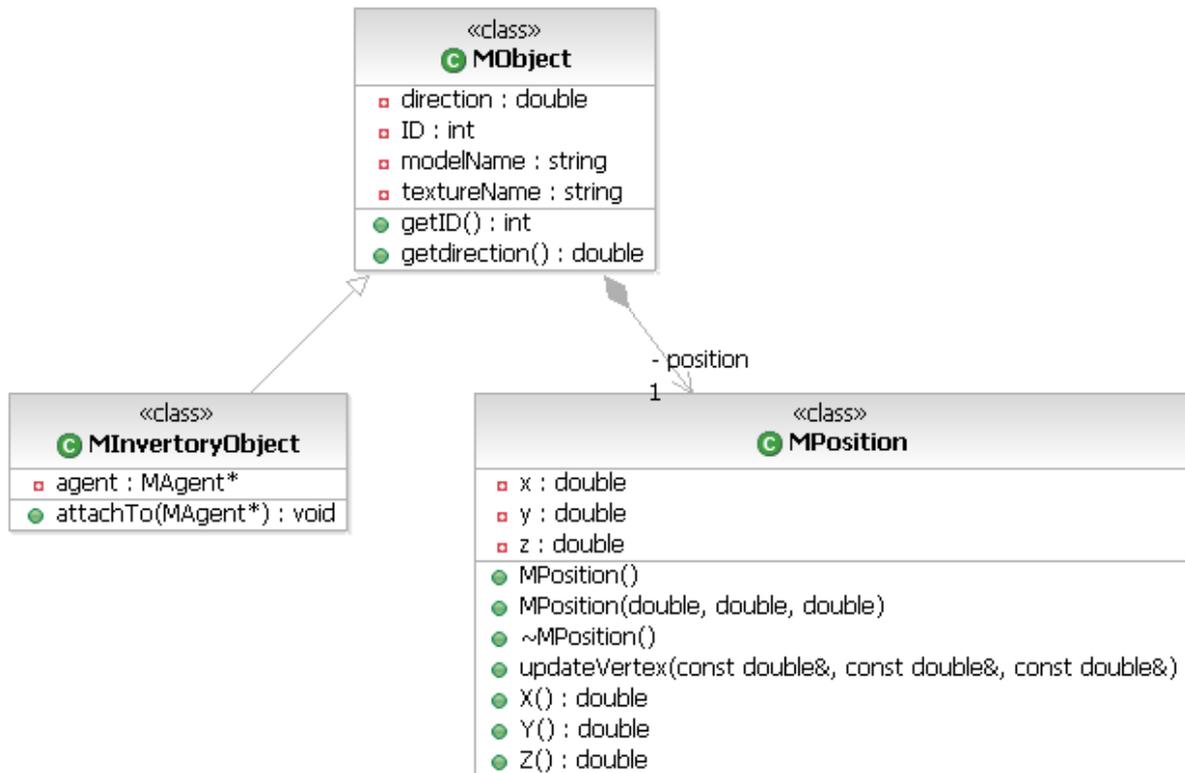


Figure – 31: Objects Class Diagram

The base class of the simulation is the object class. Most of the classes derive from the object class. However most of these derivations are used since the derived classes have common attributes like position, id, direction, model file and texture file. The inventory objects are special objects that can be attached to an agent.



## 6.4 State Transition Diagrams

### 6.4.1 Simulation States Diagram

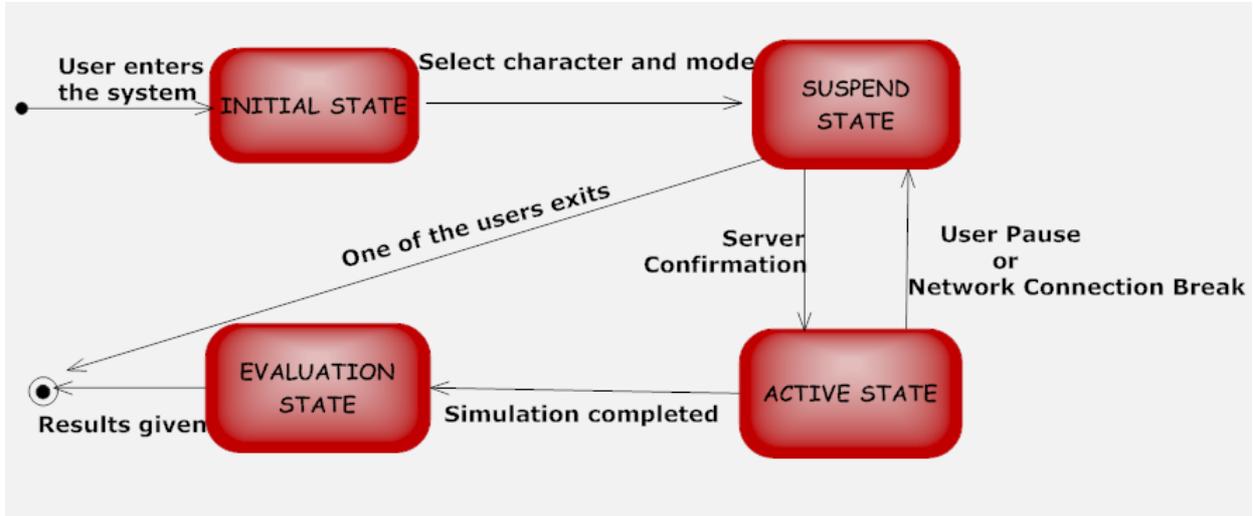


Figure – 32: Simulation States Diagram

The simulation will be in initial state before all the users that will take place in the simulation select mode1 or mode2 and one of the available character alternatives. The user who made the selection will make a transition to the suspend state and wait for the other users to complete their selections. If the user is the last one making these selections then the simulation is ready to start, else it will state in suspension until all the necessary selections are made. In active state, the normal simulation flow continues. If the simulation flow is corrupted by a connection break or one of the users pauses the simulation all the users go by the suspension. If the simulation completes, the users make a transition to the evaluation state where the users are informed about their simulation performance results.



## 6.4.2 Object Interaction States Diagram

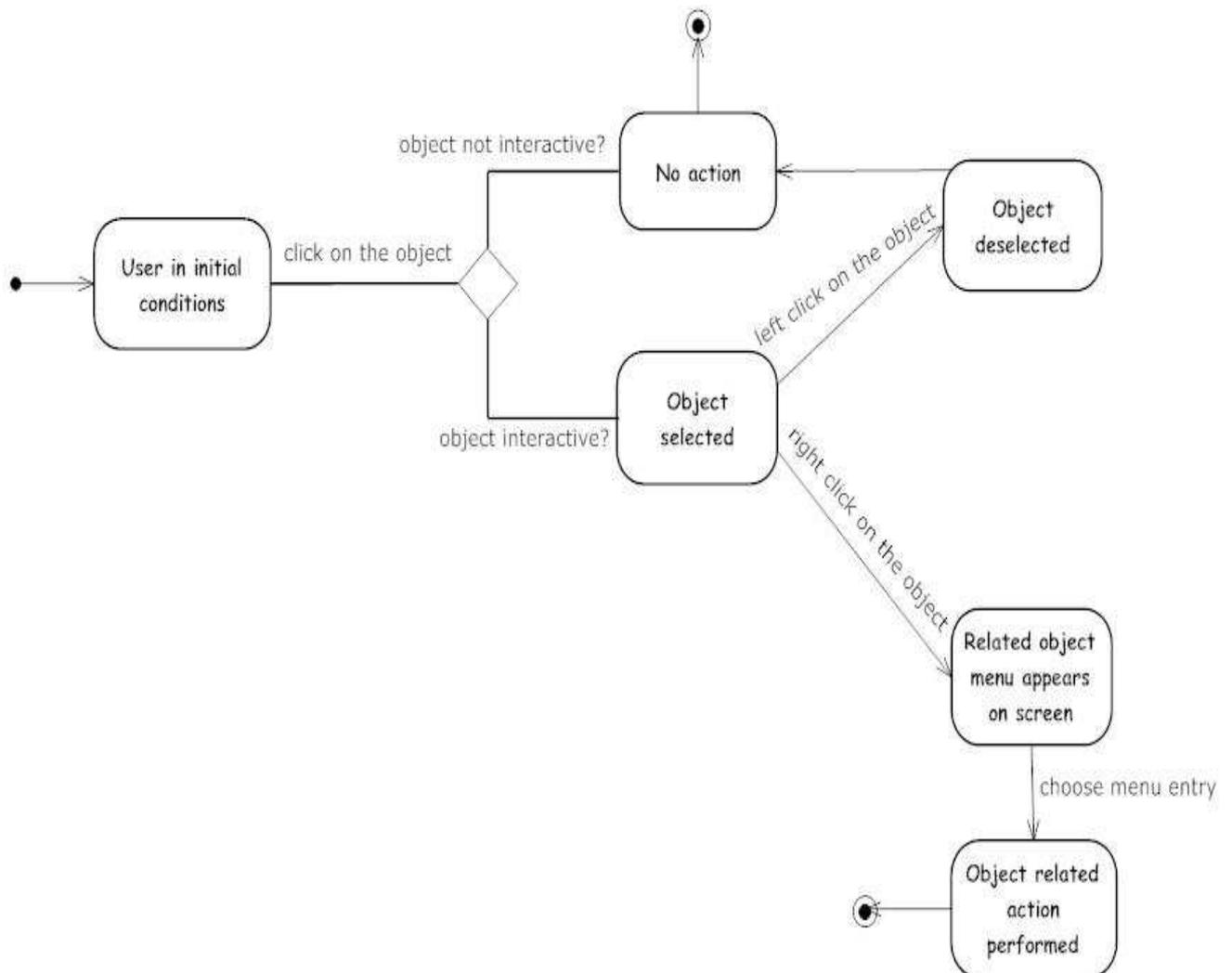


Figure – 33: Object Interaction States Diagram

In the simulation environment, there are objects used as resource. These objects are specific to the character types. If one user clicks on an object, then it is controlled if he can use it. For an object being usable points out being interactive for that user. An interactive object becomes selected when the user clicks on it. An object menu is opened with a right click provided that the object was in selected mode. The user can either select a menu entry to perform or close the menu without choosing an action.



### 6.4.3 Menu States Diagram

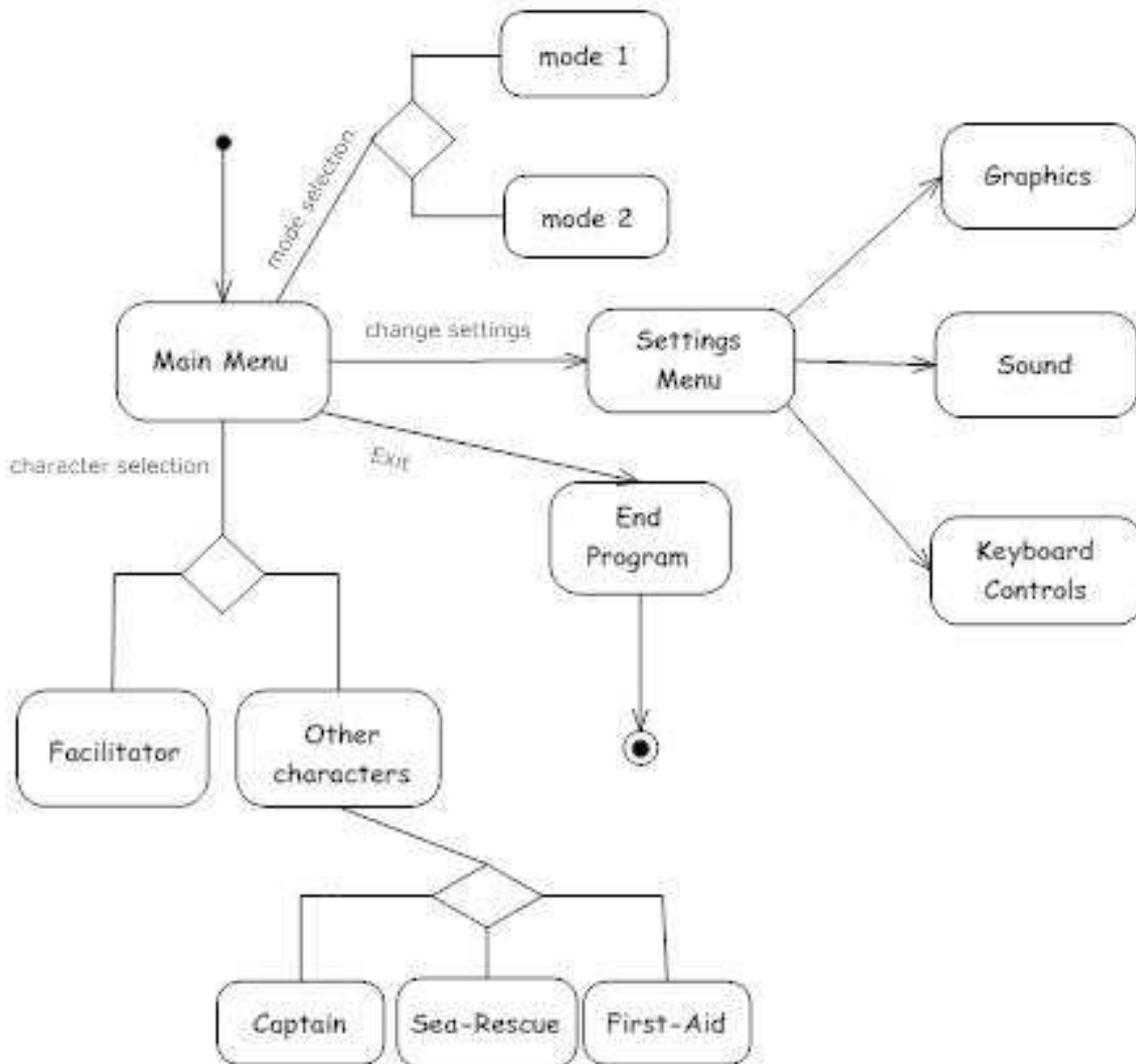


Figure – 34: Menu States Diagram

Simulation starts with main menu. User selects character type and mode type in main menu before starting simulation. He can also change the settings in main menu and quits by selecting Exit in the menu.



## 6.5 Activity Diagrams

User Movement Activity Diagram:

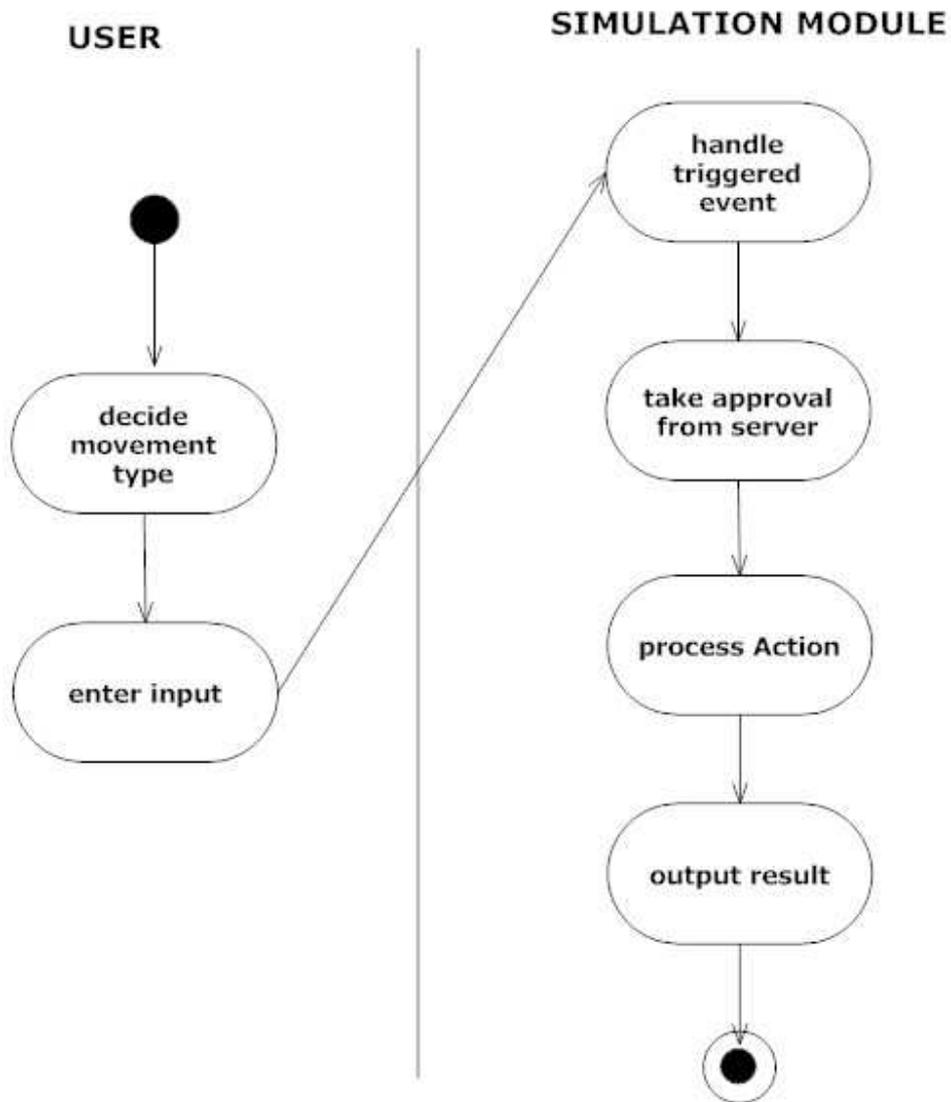


Figure – 35: User Movement Activity Diagram



User Command Activity Diagram:

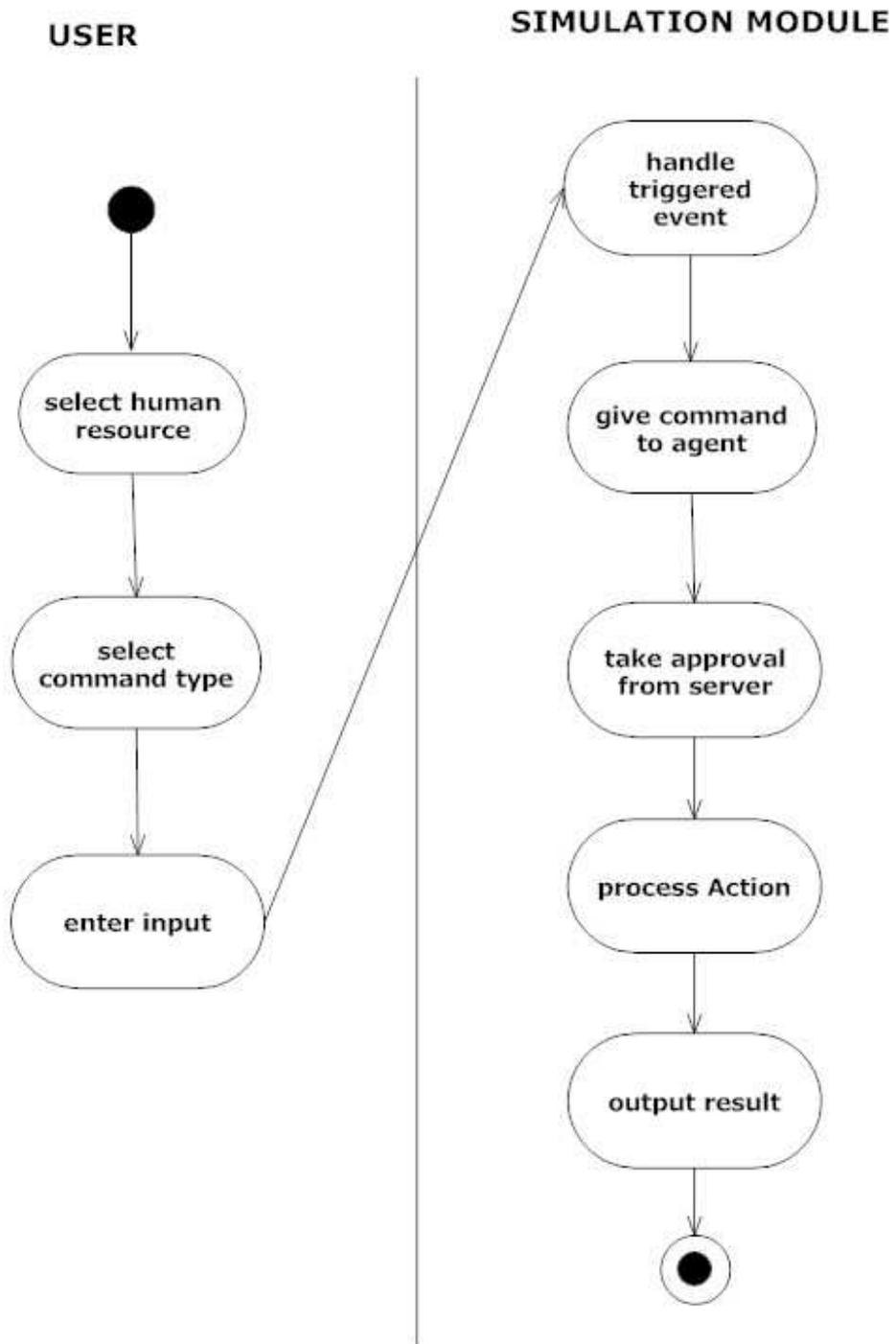


Figure – 36: User Command Activity Diagram



Menu Activity Diagram:

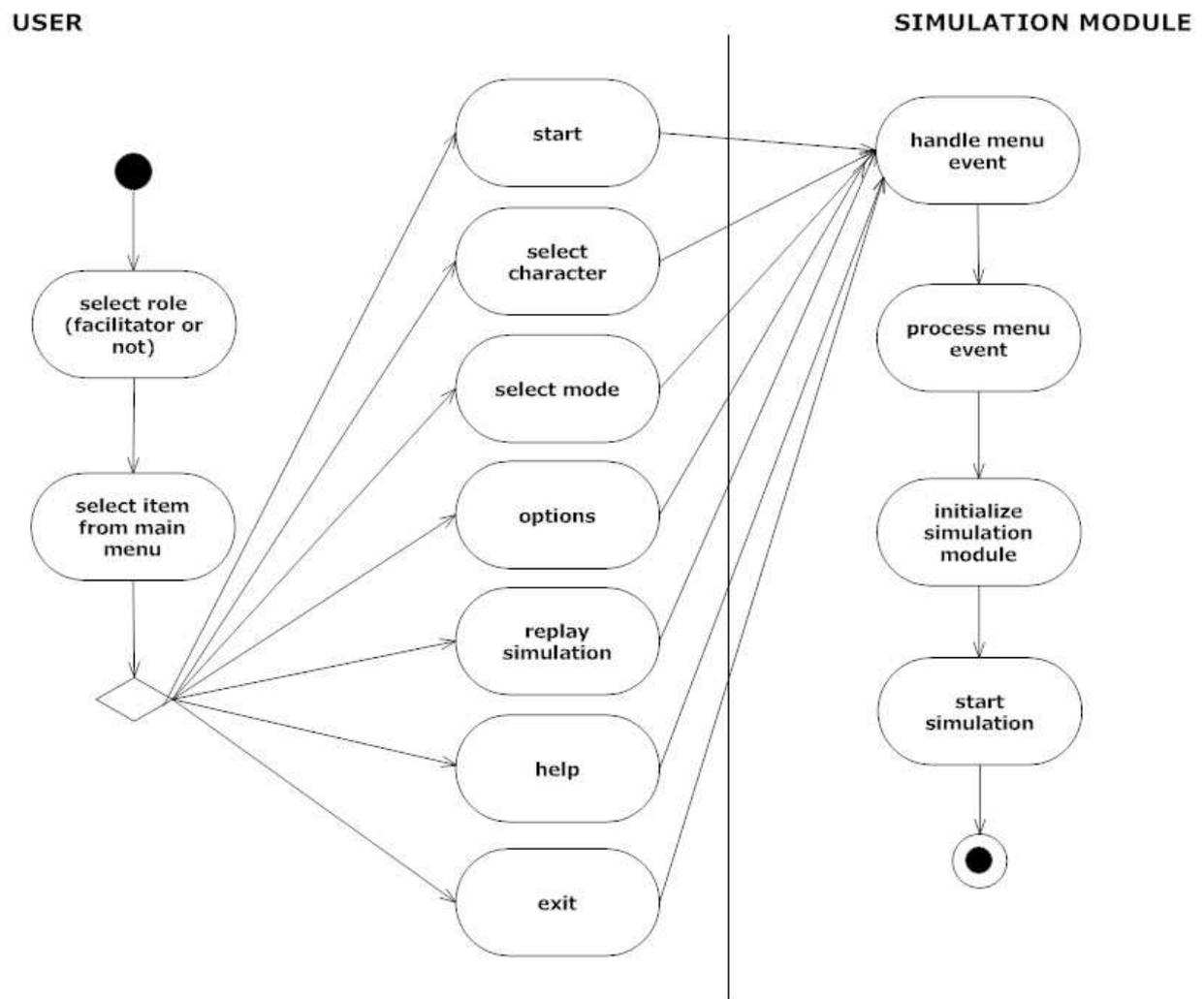


Figure – 37: Menu Activity Diagram



Manage Inventory Activity Diagram:

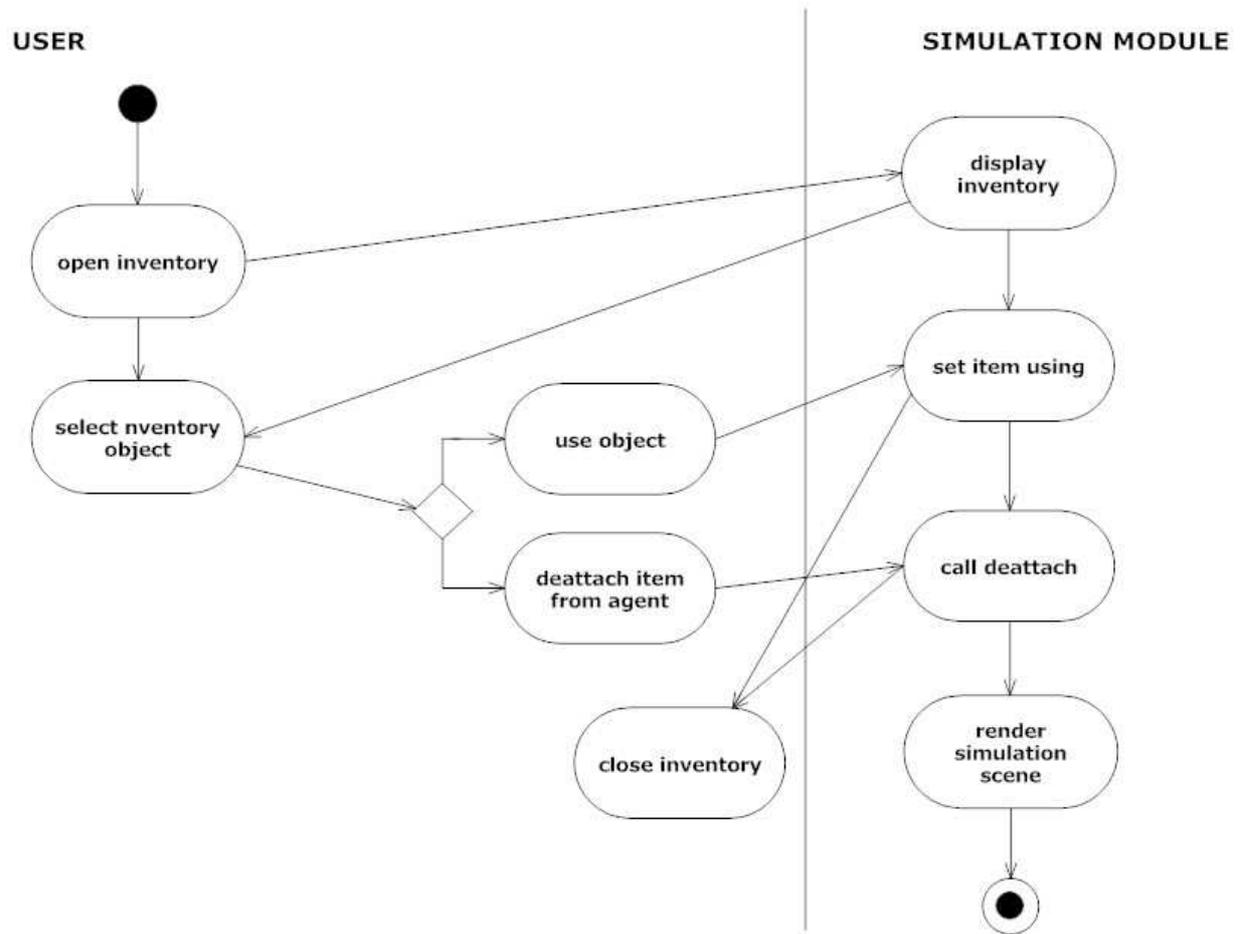


Figure – 38: Manage Inventory Activity Diagram



## 6.6 Sequence Diagrams

### Simulation Sequence Diagram:

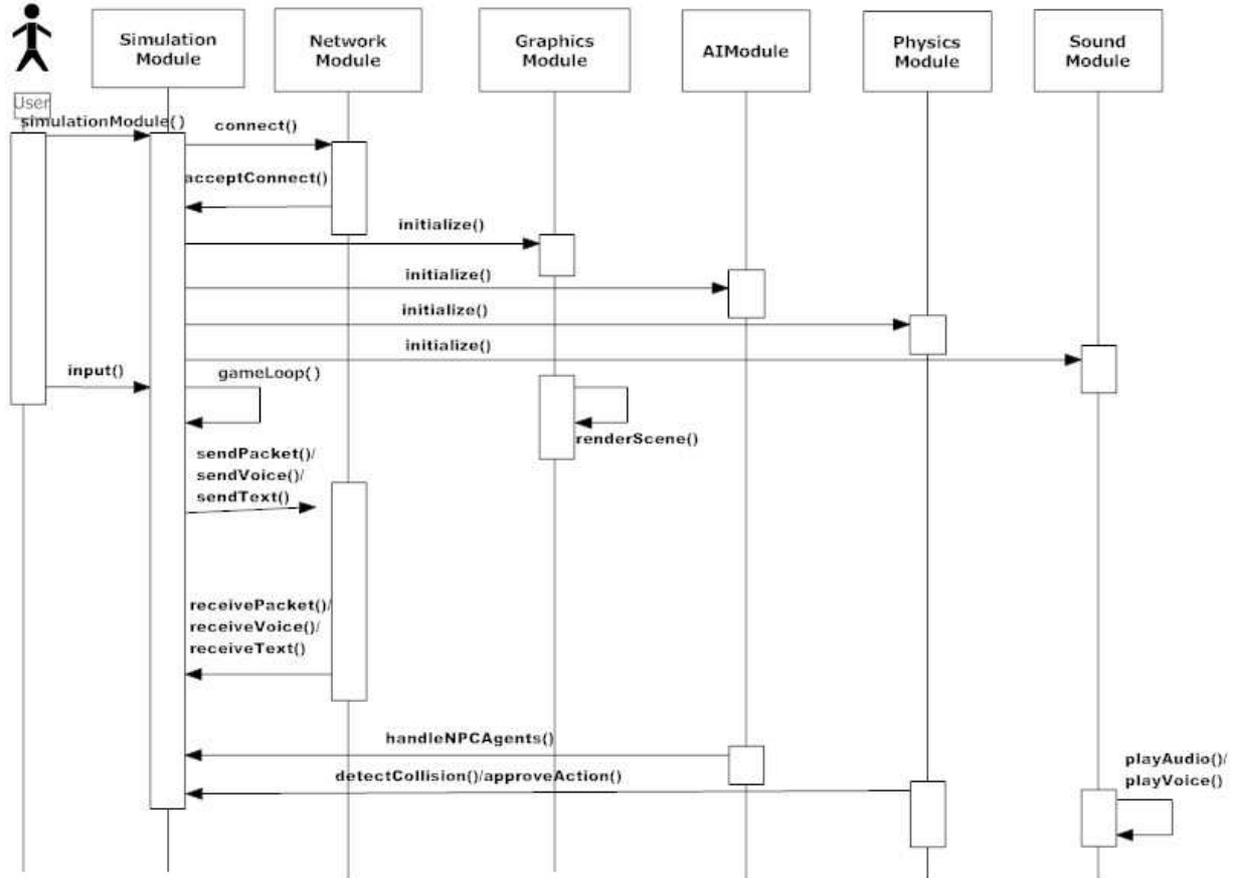


Figure – 39: Simulation Sequence Diagram



Menu Sequence Diagram:

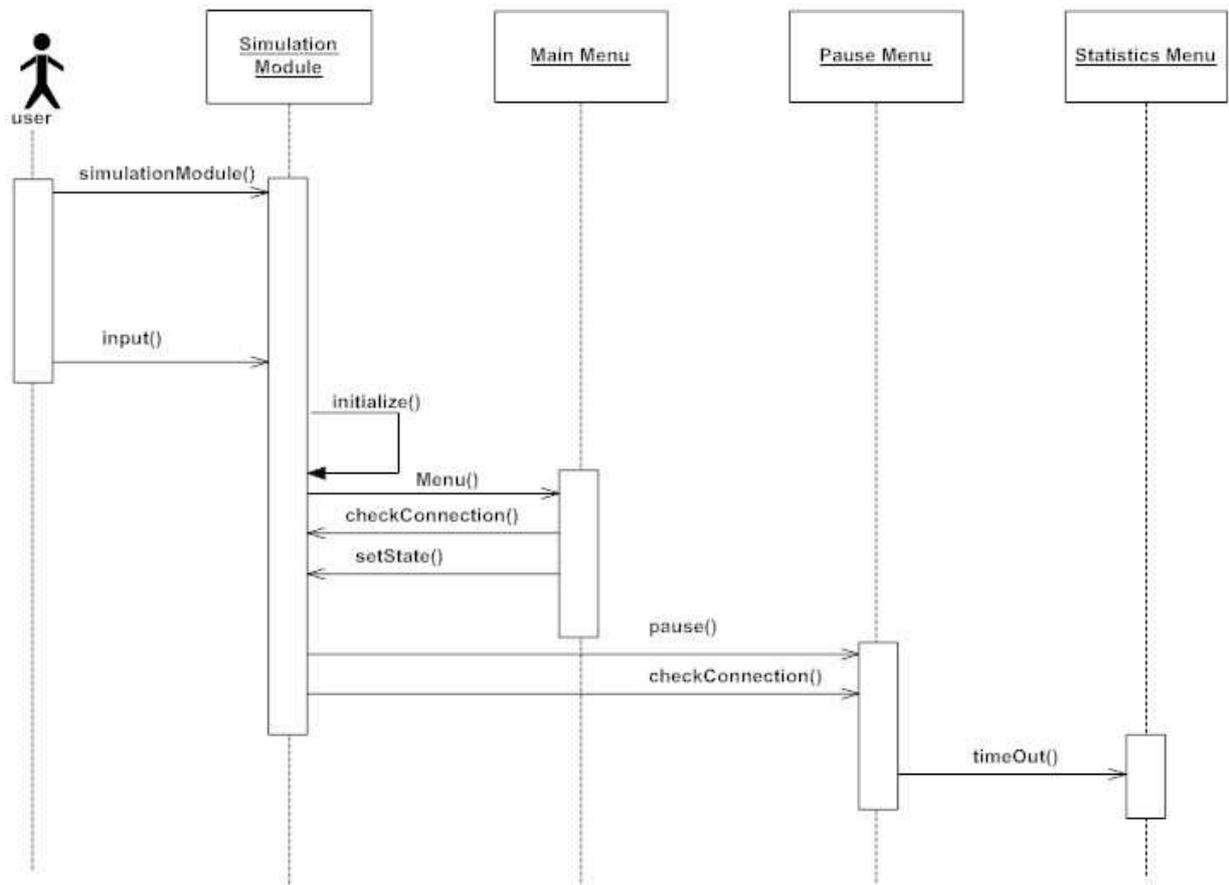


Figure – 40: Menu Sequence Diagram



## 6.7 ER Diagram

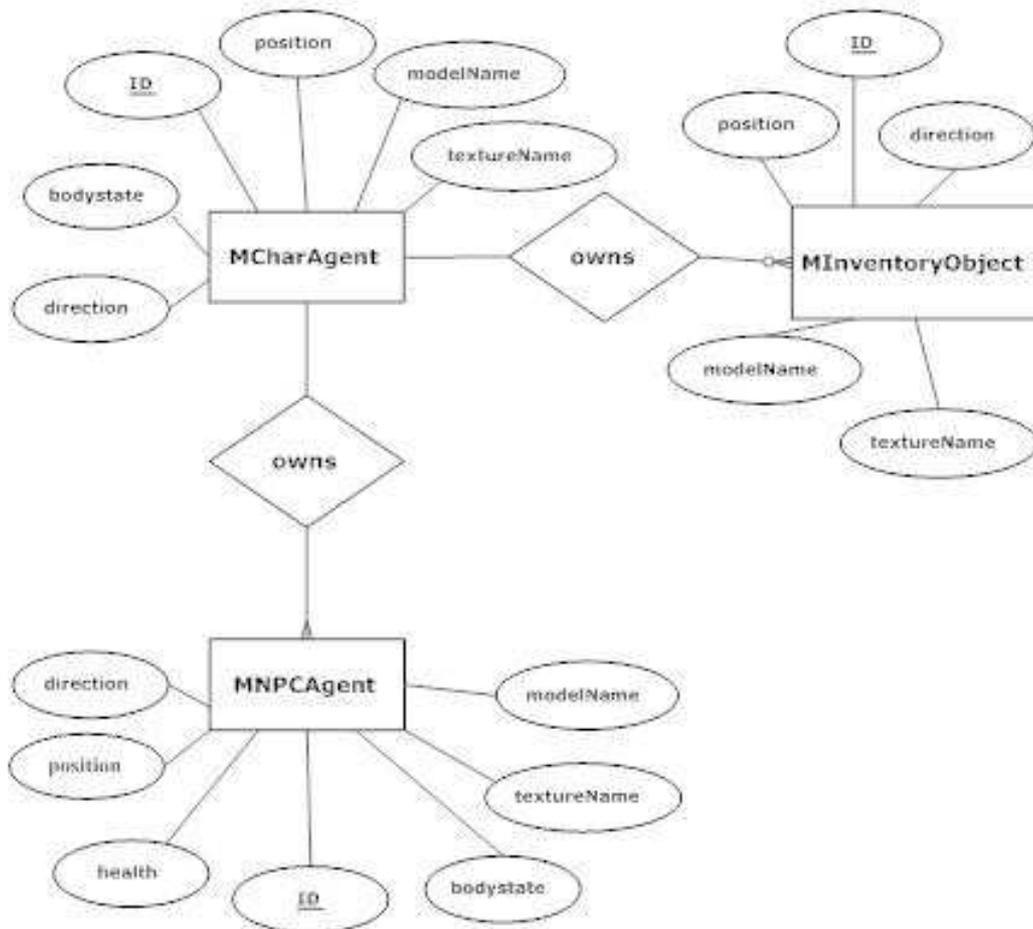


Figure – 41: ER Diagram

Character agent entity represents the characters in the simulation, while NPC agent corresponds to the human resource of main characters and inventory object corresponds to the other resource. In the simulation, characters have human resource and other resource, so the relationship owns stands for the ownership of characters. The relationships are one-to-many type since the characters can have more than one resource; in contrast a resource can belong to only one character.

## 7. SYNTAX SPECIFICATIONS

As every software company has its own syntax, Maca Yazilim has defined own syntax specification for the project. Having programming guidelines before starting the implementation



phase will be very useful in a large scaled project. This is a vital issue in order to ease the readability of the codes developed by separate team members. The common syntax specification will be helpful for integration and debugging phases.

The general specifications of Maca Yazilim will be stated in the sections below.

## **7.1 File Naming Conventions**

The header files of the classes will have the same name with the class it contains. The mesh files that are used in OGRE will have names simply representing the model it provides. The audio files will have clear names describing the state when it is played. The documentation files will have a prefix showing that they are documentation files.

## **7.2 Classes**

As a Maca Yazilim convention, the class name starts with an uppercase M stand for Maca Yazilim, the remaining part will contain words starting with an uppercase letter. The classes will first declare the private members and then public members. The classes will simply use set and get methods in order to avoid use of public variables.

## **7.3 Method and Function Definitions**

The methods except simple get/set methods will be preceded with comment blocks briefly explaining the main usage of the function. The functions will be given meaningful names related to their jobs. Their names will start with a small letter and followed by words starting with an uppercase letter.

## **7.4 Variable Naming Conventions**

The variable naming will not be very critical except the global variables. The use of Visual Studio will help the coder to find the variables automatically. However, a convention similar to Win32 API's can be helpful for further progresses.

## **7.5 Comments**

Comment writing will be helpful for understanding the codes written by other team members, for later modifications and debugging. On the other hand unnecessary and long comments will decrease the readability of the code. Team members decide to write comments for class descriptions, hardly understandable methods, and complex implementation parts of the project. Comments will be clear and informative.



## 8. PROJECT SCHEDULE

This section covers detailed information about the overall development and the remaining future work plans. The development section mainly contains how the prototypes have been developing.

### 8.1 Current Stage of the Project

The prototype that will be demonstrated on January 18, 2008 will include basic graphics about the simulation where the clients connect, do some basic actions like moving and perform a voice communication with the other clients.

In order to achieve this aim, the team has started to implement network module and graphics module. The detailed information about these implementations can be found as follows:

The network module was firstly planned to develop using DirectPlay library. During the implementation of the network library the team faced with problems caused by the inconsistency of the documentation and version of the DirectPlay. The methods defined in the API's did not exist in the SDK's. The corresponding methods for them were not doing the same job correctly. The inconsistencies of DirectPlay were caused by the depreciation of this library by Microsoft. Therefore the team switched the network library to an open source networking library: openTNL (torque networking library).

First of all, a simple network module developed for proving connection over IP addresses. The server simply created a connection to the local host and the clients directly connected to the server using the IP address of the server given as an input from the command line. After that the methods used for transferring strings between the server and the clients are implemented. In order to get rid of parsing strings, the methods for transferring primitive arguments implemented. As the next step the team started to the implementation of methods passing the structure types that will be used in the project as data packets. The method for this job is converting the objects to the predefined type of `TNL::ByteBuffer`. Instead of parsing the strings, these methods will convert the byte buffers into objects when they received byte buffers. Also they will convert the object into byte buffers for sending.

The prototype for the networking works in an infinite loop and in every time step the clients randomly generates x and y coordinate values and sends them to the server. The progress of sending packets is still continuing.



As the second part of the prototype, the team has developed a simple application with OGRE and the graphical user interface library CEGUI. The start point of the development inspired from a sample application of OGRE named ocean application. The team has found models for the ship via internet and integrated it to the application. The models were developed by 3D Studio Max so that they were in a file type of .3ds. However OGRE library did not support using a 3ds file directly in the application but it accepted mesh files. Therefore the team made use of a converter application that converts 3ds files into mesh files. The converter application will be very useful in the following steps of the project. At this step, the team achieved the success of rendering a ship sailing on an ocean with sunset view.

The next step for this prototype was implementing the graphical user interfaces with CEGUI. The team examined the XML codes of the sample applications and tutorials of CEGUI. Implementing a simple XML code brought us a main menu that can be seen in the graphical user interface section. The further progress of this prototype will be adding simple agent models and other necessary menus.

## **8.2 Future Work**

The main flow of the project can be examined in the Gantt chart section. In this section, the near future works for the prototype implementation will be discussed.

The target prototype will be developed by integration of separate prototypes. The team has currently developed some parts of the networking and graphics module. As a future work the team needs to develop an application for transferring voice messages.

The voice communication will be implemented after the network module becomes strongly successful on transferring any type of messages. In order to play the transferred voices, the need of sound module comes up. Therefore the team will continue the researches about the sound module.

The graphics module will be able to render agents and some basic objects on the ship or an empty room.

The final prototype will be an application that the users connect to a server, has a rendered character in the environment, can make basic movements and send voice messages to other clients via server. This prototype will be an easy progress if the separate prototypes are correctly developed.



Although it is not visible directly, the final prototype mainly builds the backbone of our project.

### 8.3 Gantt Chart



