MAÇA YAZILIM

ONLINE VIRTUAL TEAM COLLABORATION PLATFORM WITH 3D GRAPHICS

CENG 491

Detailed Design Report

METU

Table of Contents

1. INTRODUCTION

Introduction part of this document is about the detailed project definition, project features, and general design properties.

1.1 Purpose of the Document

This document is the final design report of the project. It covers the implementation related plans and the complete specification of features. The issues defined in the initial design report will be given in further detail.

1.2 Project Definition

In the simulation, users will be confronted with a fire on a passenger ship. In accordance with their character type, they will decide which action to take. During the event flow, there will be exactly three persons using the program, each having a different role.

The available roles will be: captain, sea rescue chief and first-aid chief. For every event there will be different action alternatives for users and these alternatives 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. When the alarm is active, the captain will see the red light emitted by the alarm and hear the sound of it. Then the captain -the coordinator in the simulation will communicate with his assistants (human resource) and then the other chiefs on the ship will be informed. Other chiefs can be awakened by using voice communication. The chief of rescue team will intervene in the fire and try to evacuate passengers. The fire will be easier to extinguish at the beginning, and it will grow as time passes. The crew will help the rescue-team chief on his action. The chief can select equipment that is in the non-human resources and a group of crew, so assign a task to that group. To exemplify, he gives fire-extinguisher to a specific crew unit and send them to the position he wants. In the course of events, some passengers will get injured and the last character's mission is to help them as much as possible. There will be several types of injuries. Depending on the seriousness of the injury, the chief will have more difficulty with the injured passenger. The treatment of a seriously wounded passenger will be obviously more challenging. If the treatment is not proper or the injury is too serious, the wounded passenger will die. Health officers will be working with the first-aid team chief to process his orders as human resource.

As stated in the previous reports the resources are shared among the characters like below:

- Resource of the 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.

1.3 Project Features

The features to be provided:

- 3D Computer graphics
- Text messaging
- Voice communication
- Simple, easily understandable user interface
- Evaluation of the simulation performance: Mainly four criteria will be used for the evaluation process.

  1) Number of passengers died
  2) Number of passengers injured
  3) Percentage of ship area ruined
  4) Number of passengers evacuated

The weight of responsibility on these topics will be different for each character. Since the captain has the coordinator role in the simulation, he is responsible for all the topics equally. Rescue-team chief has more responsibility for “Percentage of ship area ruined” and “Number of passengers evacuated”. Finally the chief of first-aid team chief is responsible for “Number of passengers died” and “Number of passengers injured”.

1.4 Design Constraints and Limitations

Maça Yazılım is preparing this simulation project for the senior project course of Computer Engineering Department, METU. This brings the time constraint. This project must be finalized before June which means the development phase has a duration of eight months, so time should be used carefully.

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 are 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 are using many libraries during development stage like openTNL, 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.

1.5 Design Goals and Objectives

- The main concern for the simulation is the target user's computer capability. The simulation must be easily used by an ordinary user even he did not know much about the computers. The voice communication technique and a facilitator bring up an easily adaptable simulation environment.

- In order to make users concentrate to the situation, it is necessary to build up a realistic environment. The virtual reality should be supplied by both graphical realism and the appropriate physics rules or human behaviors. Therefore, a physics engine and a fire dynamics simulator will be used. The virtual reality will surely increase the contribution of the simulation to the users since it will be very similar to real life.
- 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 will be done very carefully in order to achieve this goal.

- The communication via network points out the importance of security. The carefully developed code should block external threats to the users during and after the simulation.
- The simulation will be working on Windows operating system.

2. SYSTEM AND TOOL CHOICES

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

Operating System Choice: Windows XP Operating System.

Hardware Choices: 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:

- Graphics Rendering Engine: OGRE, one of the most popular rendering engines, is chosen for the project. Most important features of OGRE that affected the choice were its design quality, flexibility and clear documentation. As the sample programs using OGRE were really helpful in developing the project, using OGRE brought some additional advantages. In addition, advice of computer engineers who have worked on some graphics projects before, contributed to our decision. The choice of OGRE affected the decisions of input handler and GUI library directly. The OIS (Object Oriented Input System) and CEGUI respectively for the input handler and GUI library are chosen for their compatibility with OGRE.

- Network Engine: OpenTNL (Torque Network Library) is chosen as the network engine. Before the final decision is made, there were some other alternatives like DirectPlay and Raknet. At first, the alternative that was most likely to be chosen was DirectPlay but some inconsistencies occurred between documentations and the SDK's because of the depreciation. The team tried to develop simple applications with DirectPlay but these attempts failed due to the inconsistencies. Consequently the team reached an agreement on OpenTNL. It was allowing the developer to transfer the objects as argument, string, byte buffer, bit stream or vector. It has a supportive documentation and samples.

- Sound Engine: OpenAL and DirectSound will be used for sound. Since the team had indecisions about this issue, implementations using both of them for the prototype of the

project have been made. Finally, OpenAL was preferred for playing the sounds on the other hand DirectSound was preferred for recording the voices.

- Voice Codec: LPC10 of Hawk Voice was chosen to encode and decode the recorded sounds for its high performance on compression of the voice. However, the GSM codec was also integrated in the prototype in order to observe the tradeoff between voice quality and network latency.

  (The table below which is taken from *www.hawksoft.com/hawkvoice/codecs.shtml* represents the test results of codecs in hawkvoice.)

  CPU cycles per second for 8 KHz sample rate sound.

  Compression % is compared to 16 bit PCM.

	encode	decode	compression
u-law:	42K	40K	50%
ADPMC:	407K	330K	75%
GSM:	2.0M	950K	89.7%
LPC:	2.5M	1.0M	96.3%
CELP 4.5K:	24-52M*	4.4M	96.5%
CELP 3.0K:	25-47M*	4.0M	97.7%
LPC-10:	6.4M	3.5M	98.1%
CELP 2.3K:	24-45M*	3.8M	98.2%
OpenLPC 1.8K:	2.9M	1.8M	98.6%
OpenLPC 1.4K:	2.9M	1.9M	98.9%

- FDS (Fire Dynamics Simulator) was chosen to achieve a higher level of virtual reality. It shows the fire characteristics, emanation and smoke emission very realistically and no other open source option can be found that is doing the same task.

- Physics Engine: ODE is widely used with OGRE, so we have decided to use it as the physics engine.

3. 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.

3.1 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.

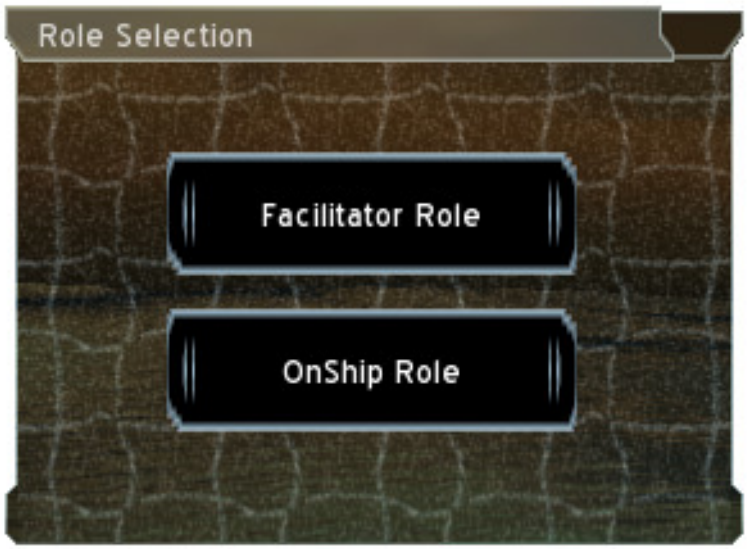

Figure – 2: Initial menu

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.

3.2 Main Menu

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

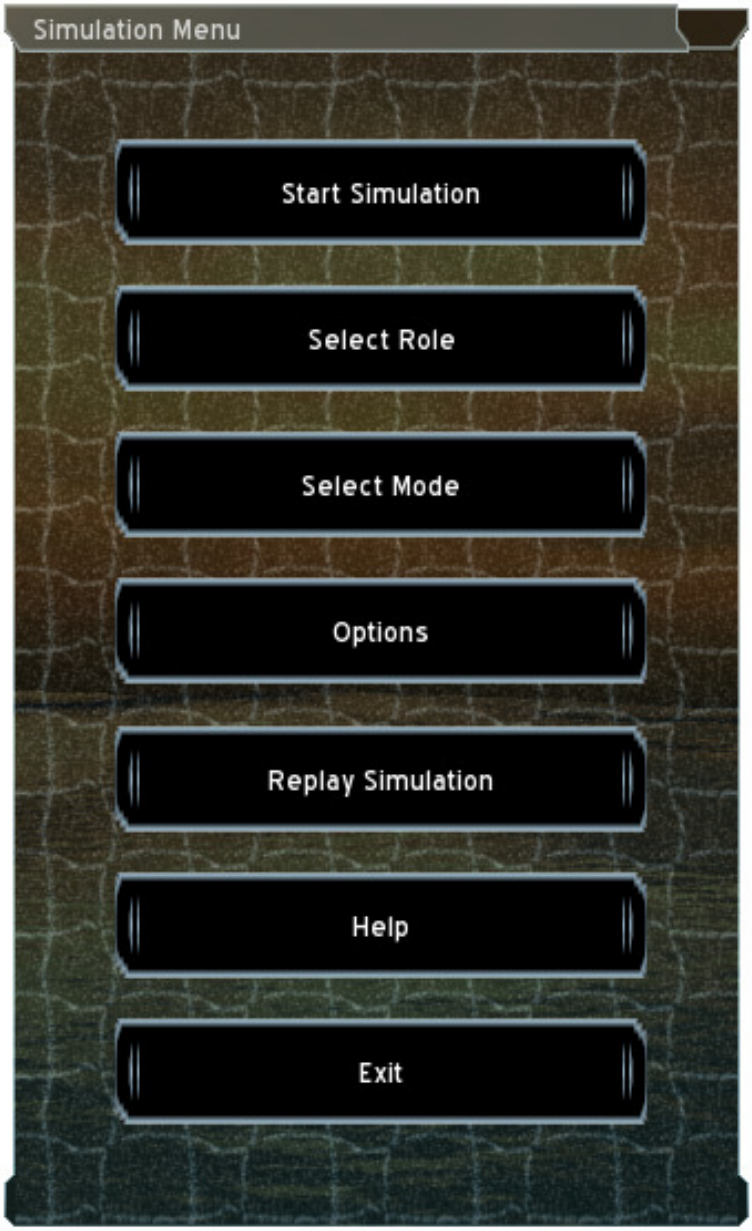

Figure – 4: Main Menu

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.

3.3 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:

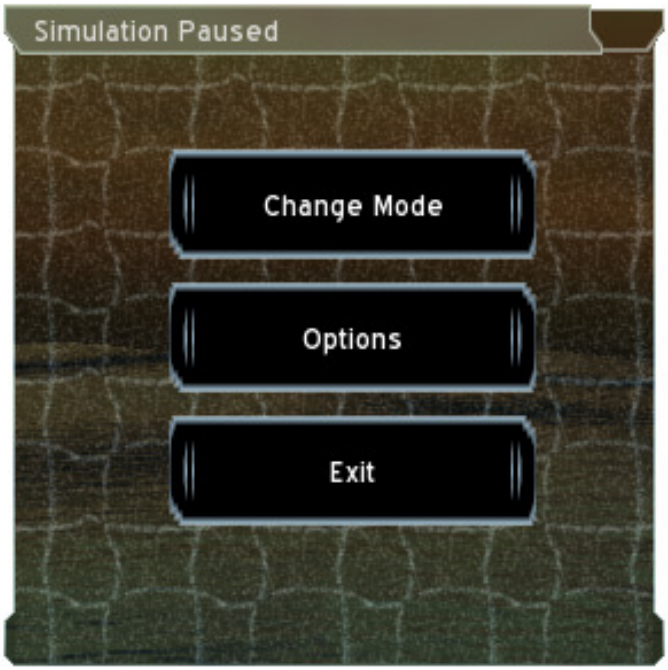

Figure – 6: Paused Mode Menu

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

3.4 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.

3.5 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).

4. OVERALL ARCHITECTURE

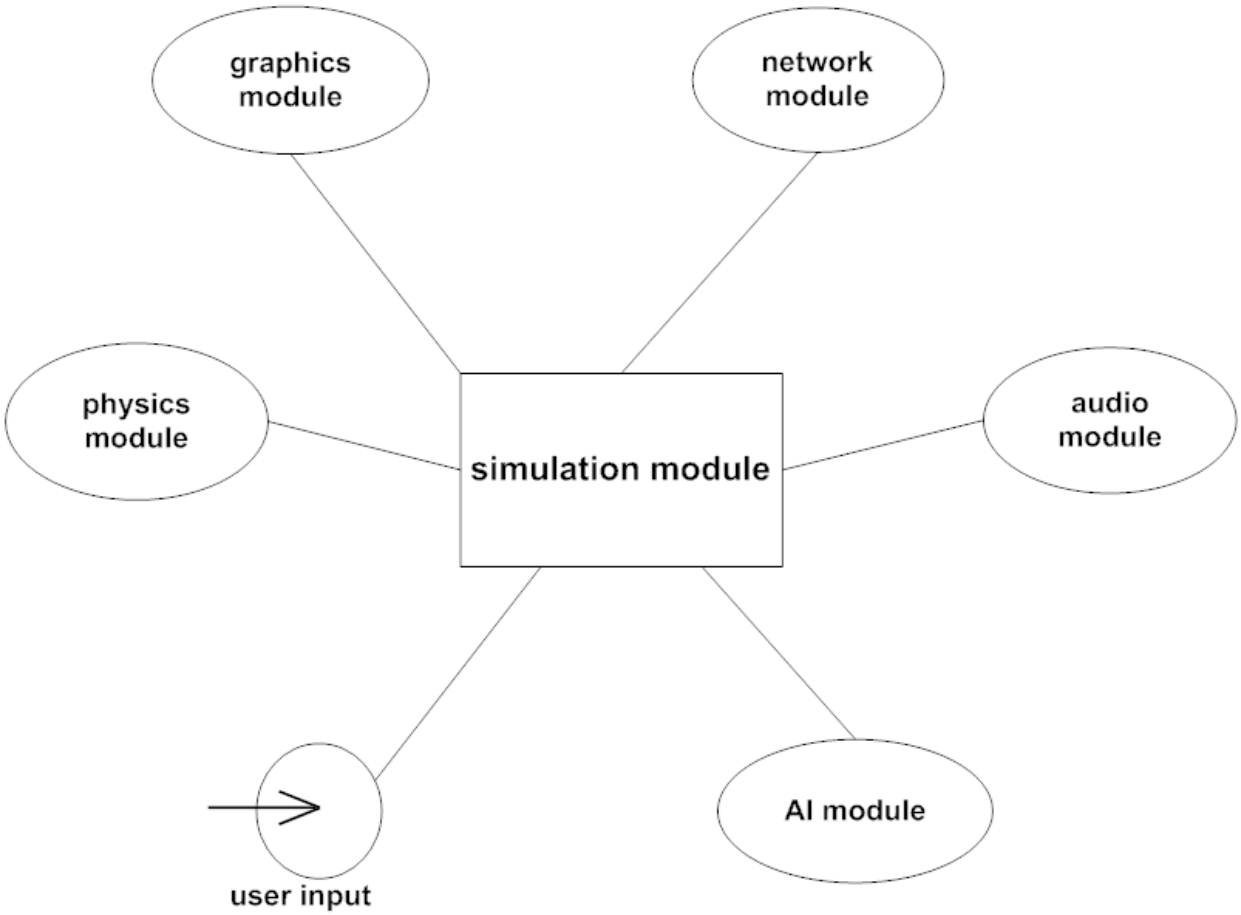

Figure – 9: Overall Architecture

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.

5. DETAILED DESIGN

5.1 Data Flow Diagrams and Data Dictionary

Level: 0 DFD:

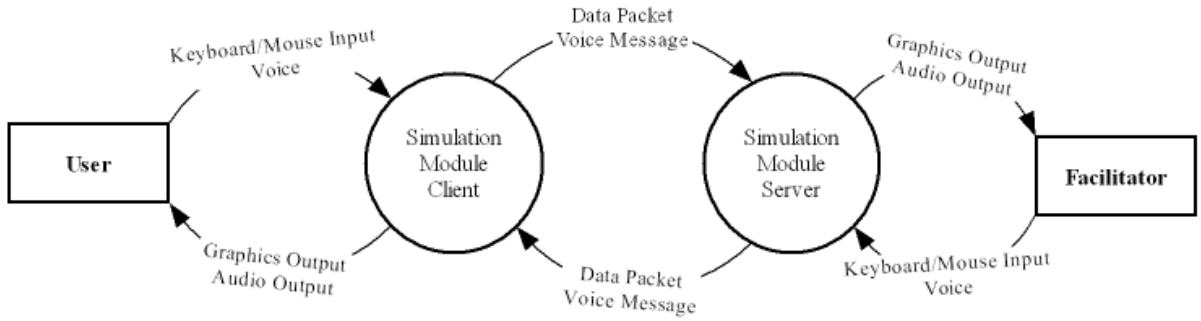

Figure – 10: Level 0 DFD

Level: 1 Simulation Client DFD:

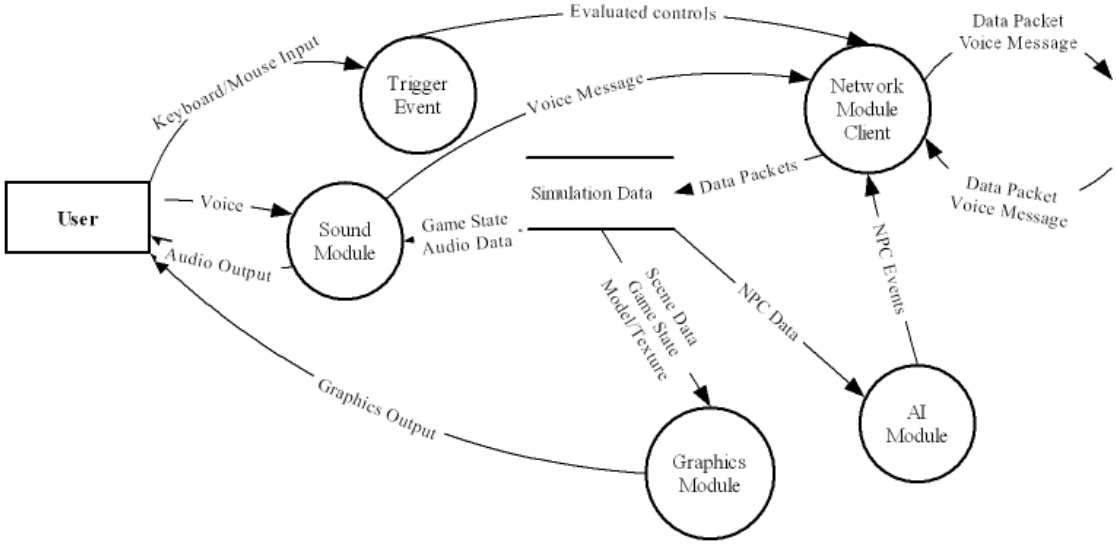

Figure – 11: Level 1 Simulation Client DFD

Level: 1 Simulation Server DFD:

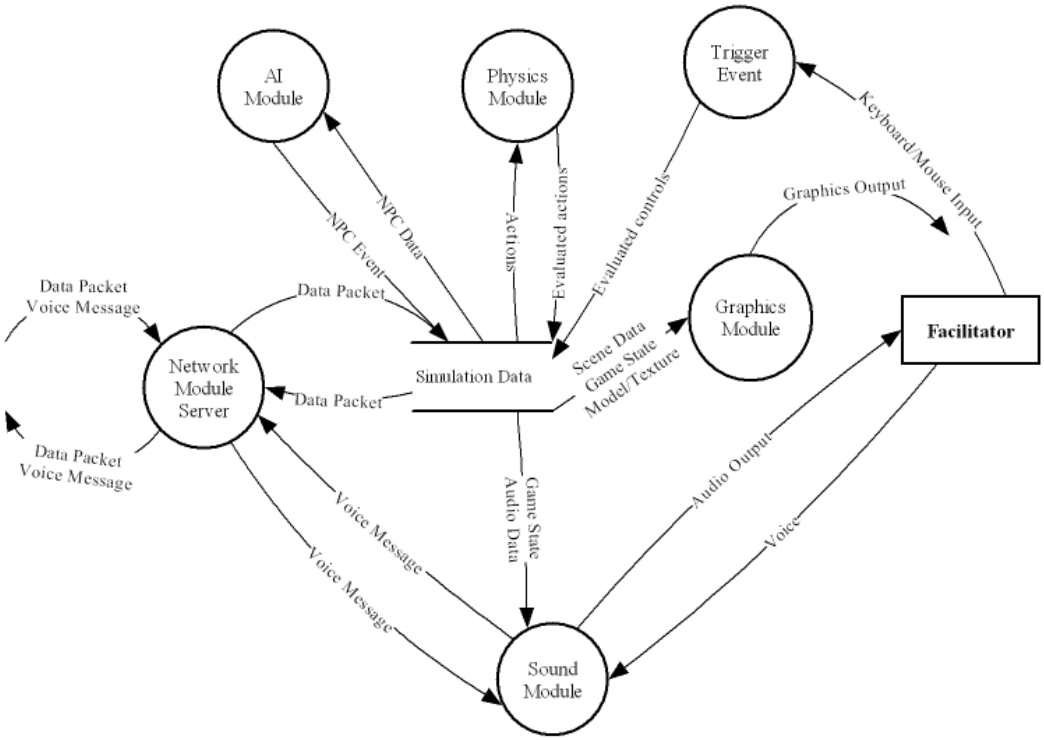

Figure – 12: Level 1 Simulation Server DFD

Level: 2 Graphics Module DFD:

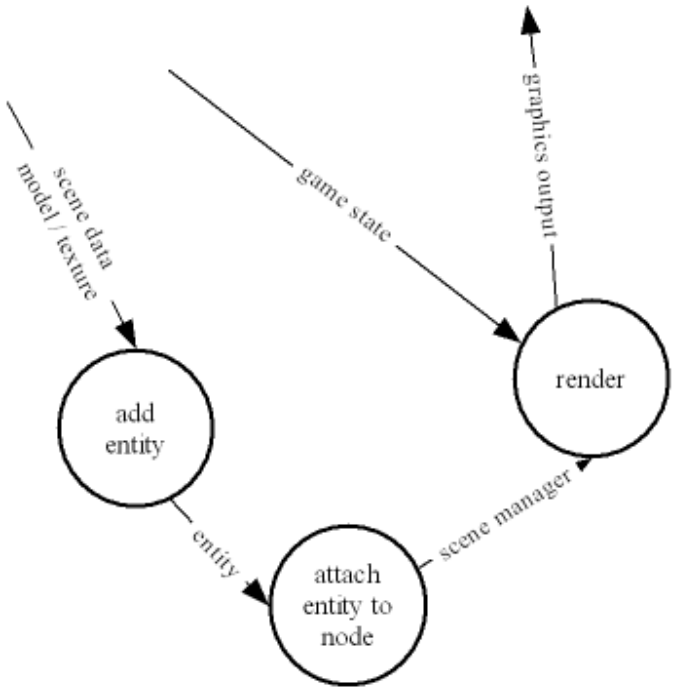

Figure – 13: Level 2 Graphics Module DFD

Level: 2 Network Module DFD:

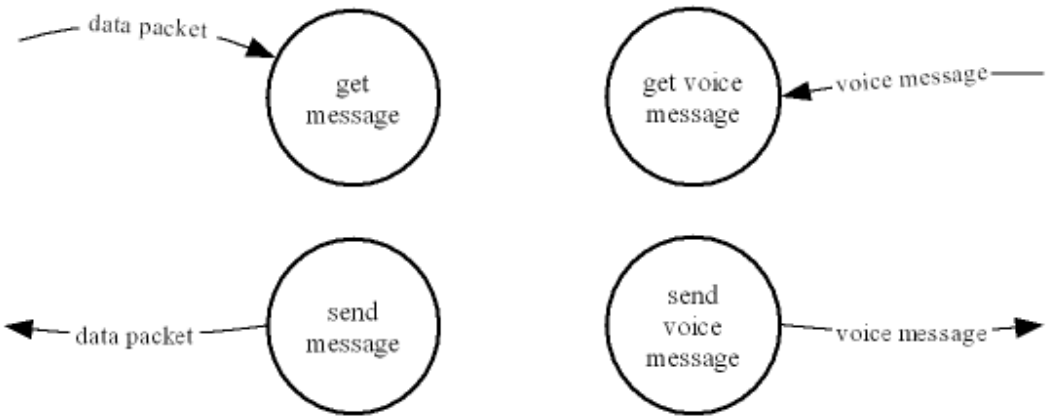

Figure – 14: Level 2 Network Module DFD

Level: 2 Physics Module DFD:

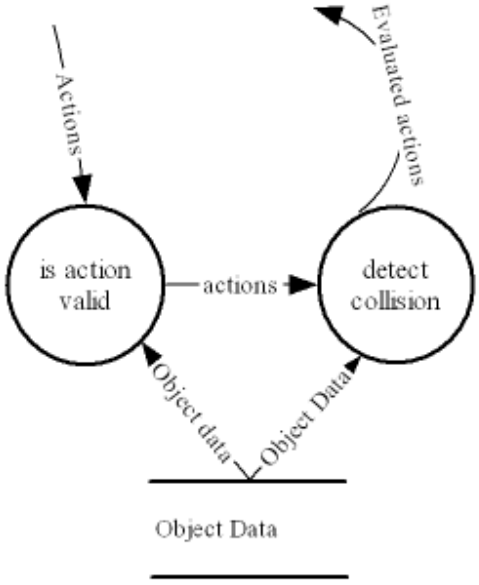

Figure – 15: Level 2 Physics Module DFD

Level: 2 AI Module DFD:

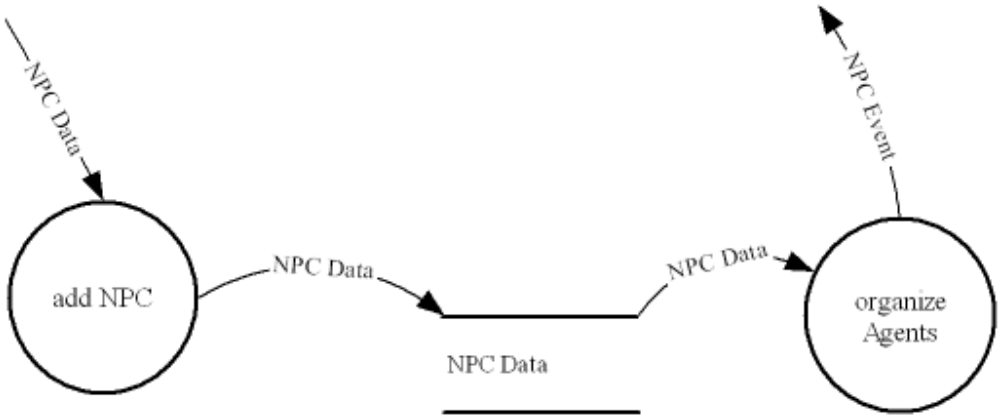

Figure – 16: Level 2 AI Module DFD

Level: 2 Sound Module DFD:

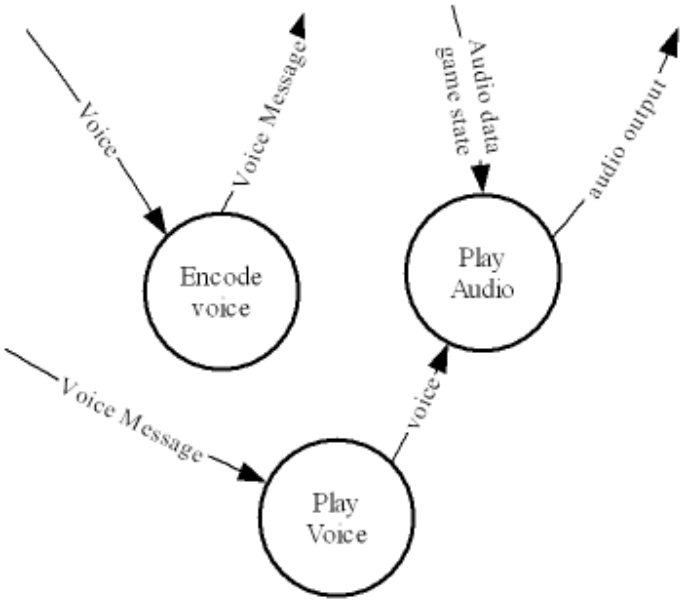

Figure – 17: Level 2 Sound Module DFD

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

5.2 Use Case Diagrams

Menu Use Case Diagram:

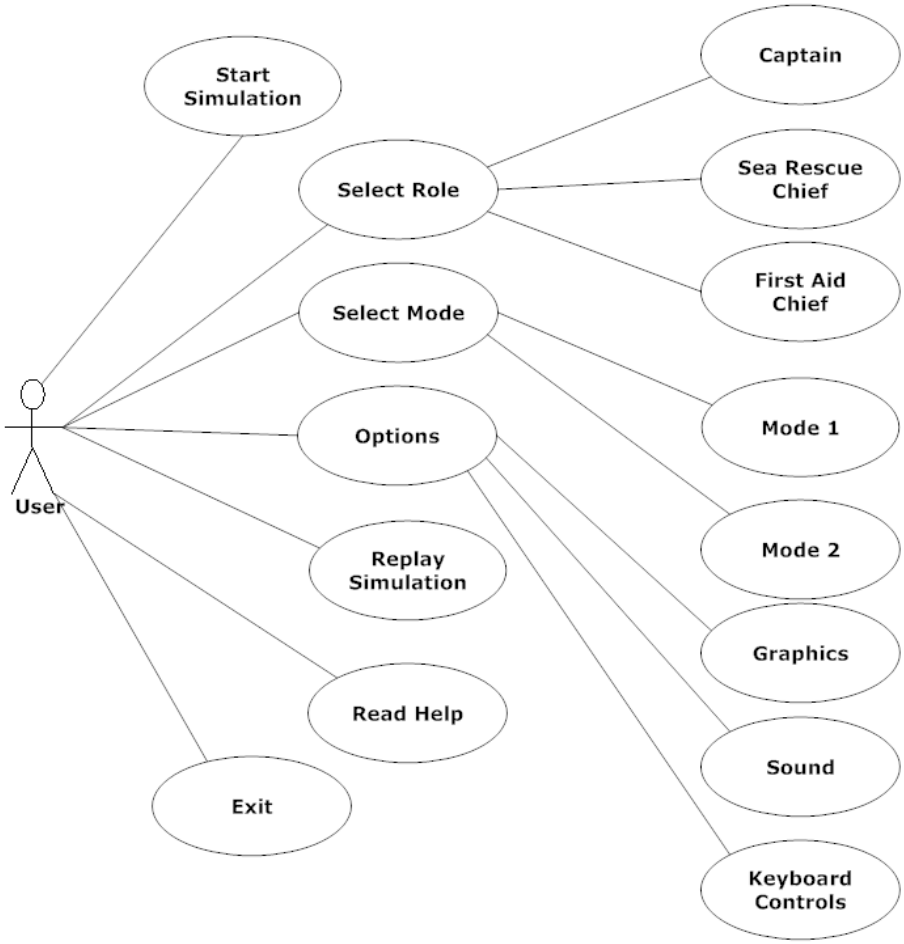

Figure – 18: Menu Use Case Diagram

Facilitator Use Case Diagram:

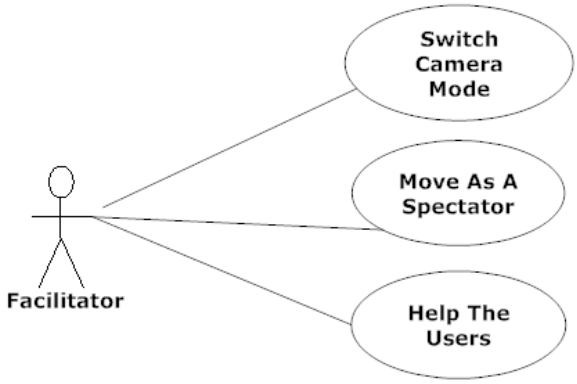

Figure – 19: Facilitator Use Case Diagram

Captain Use Case Diagram:

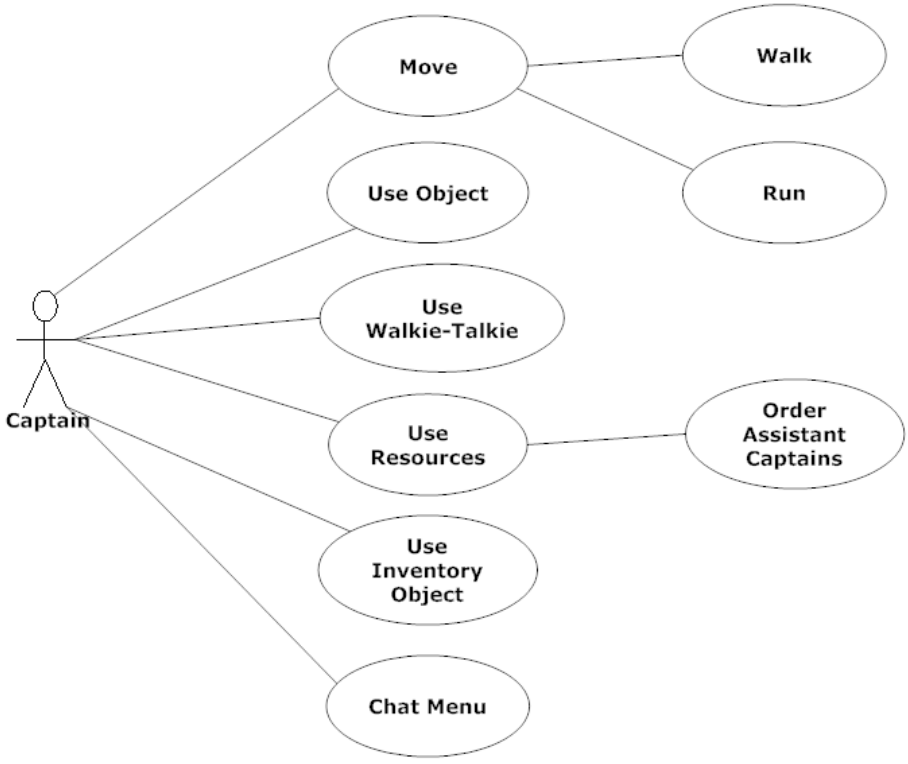

Figure – 20: Captain Use Case Diagram

First Aid Chief Use Case Diagram:

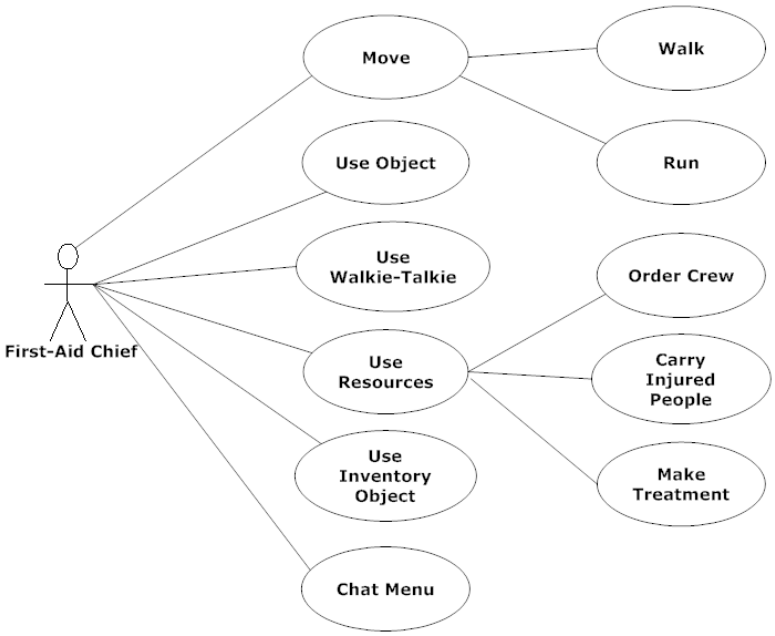

Figure – 21: First-Aid Chief Use Case Diagram

Sea Rescue Chief Use Case Diagram:

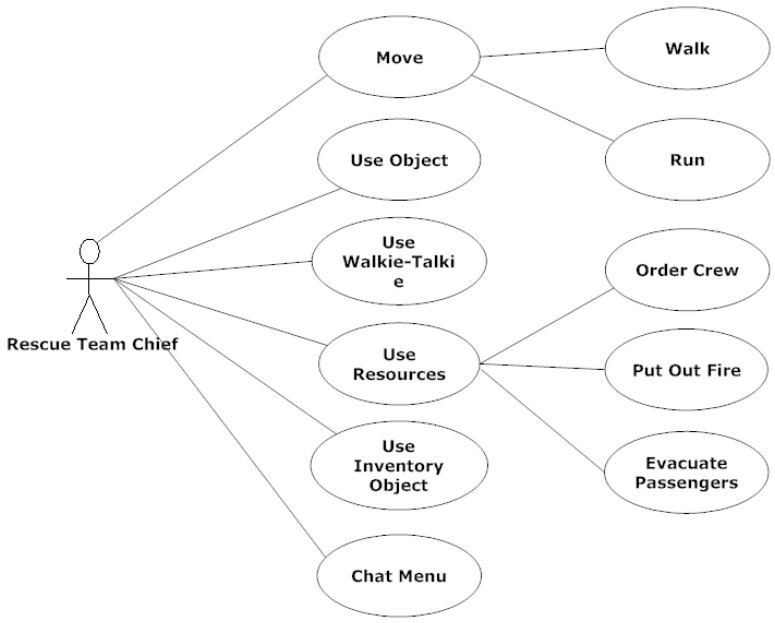

Figure – 22: Sea Rescue Team Chief Use Case Diagram

Passenger Use Case Diagram:

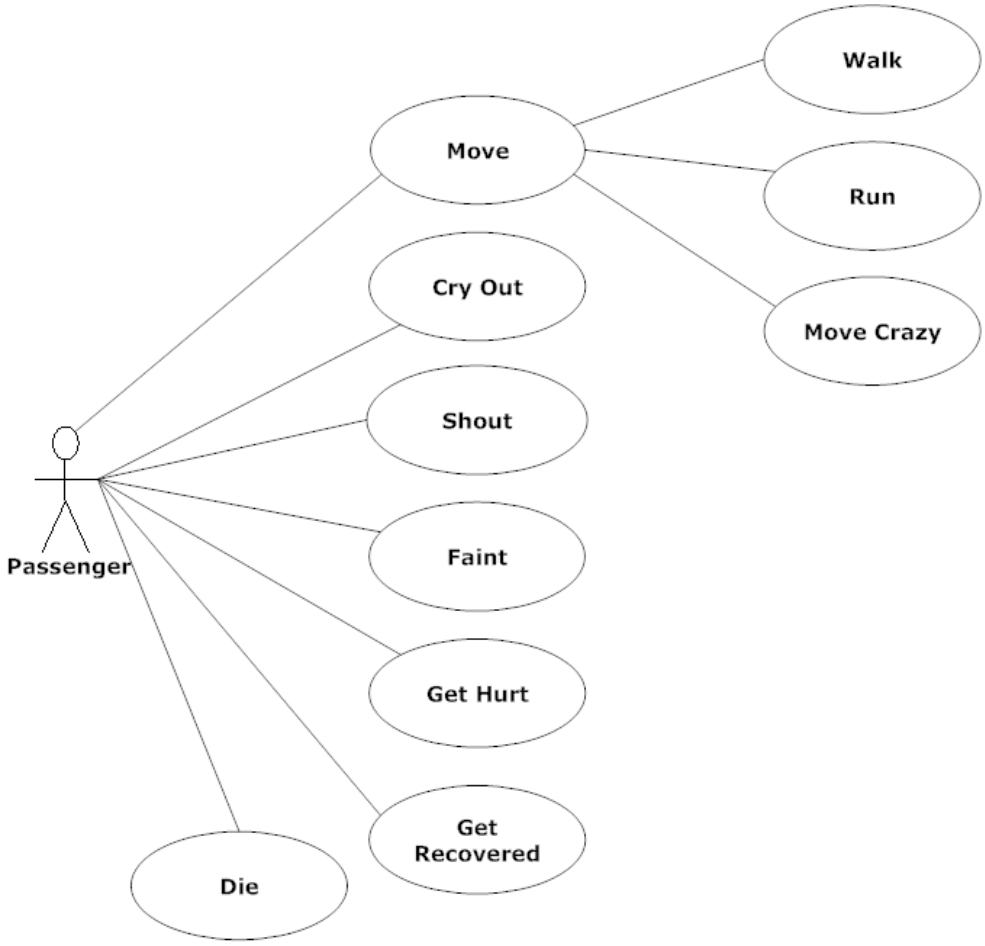

Figure – 23: Passenger Use Case Diagram

5.3 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.

5.3.1 Simulation Module

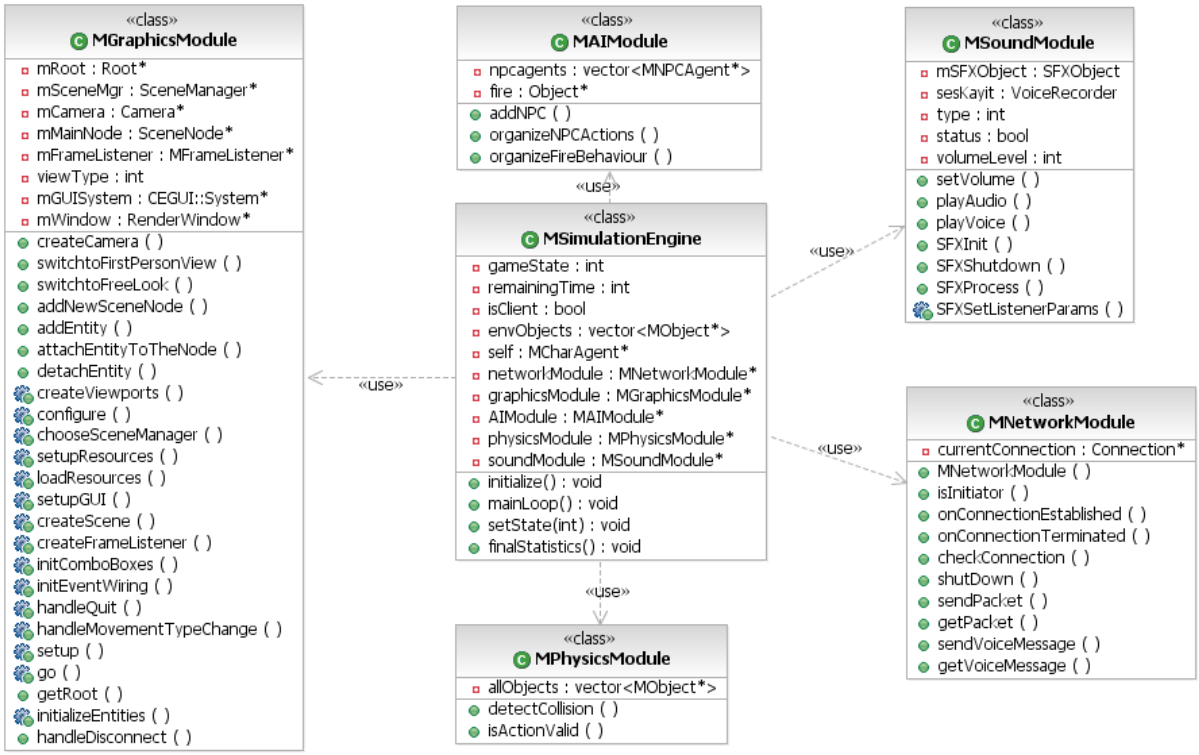

Figure – 24: Simulation Module Class Diagram

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

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 network 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. This choice lightens the heavy load of the server, and consequently fastens the simulation loop.

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 simulation 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 each 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 in main design since the keyboard/mouse inputs will trigger events on OGRE and they will be handled using OIS library (Object Oriented Input System).

5.3.2 Network Module

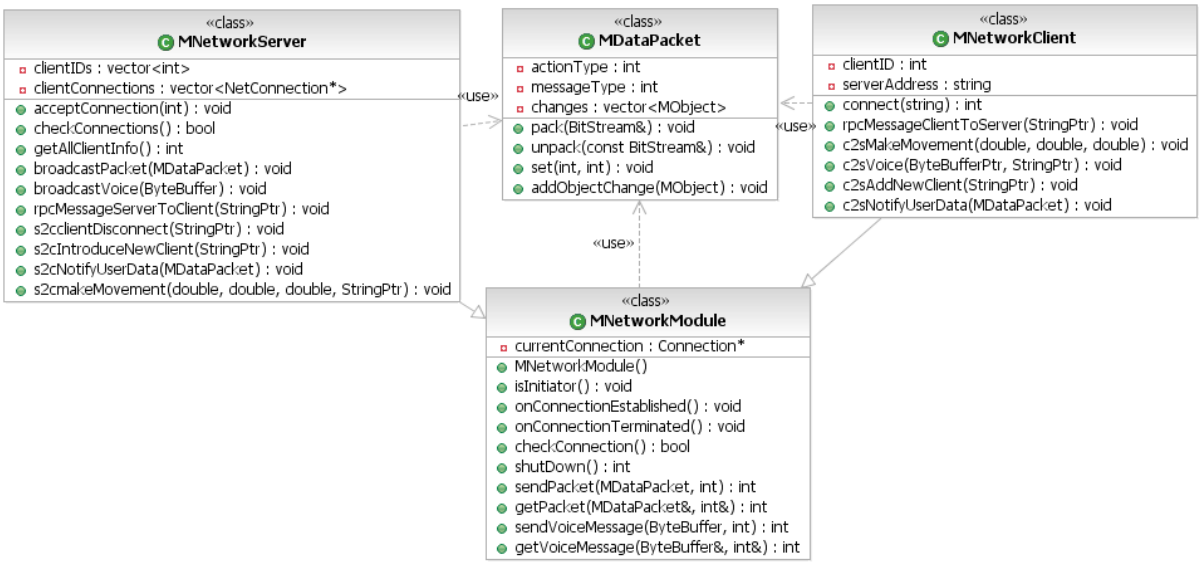

Figure – 25: Network Module Class Diagram

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, Bit Stream or separated as arguments to transfer between sides.

The common methods of Network Module are used for transfers of packages and voice messages, checking the connection link to see whether it is not broken and shutting down the connection. The initialization of the connection will be done when the *onConnectionEstablished* method is called. The *onConnectionTerminated* method is called when a client or server directly disconnects or loses connection with the remote side.

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 client may raise events on the server side using the c2s methods. The c2s is a convention for client-to-server and s2c for server-to-client. The *c2s* methods are special methods used for corresponding event notifications to 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. The network server has *s2c* methods that broadcast the events of c2s methods to the remaining clients. The s2c methods can be called directly as a reply to a c2s method or it can be called by the server in order to raise an event on the client side.

5.3.3 Graphics Module

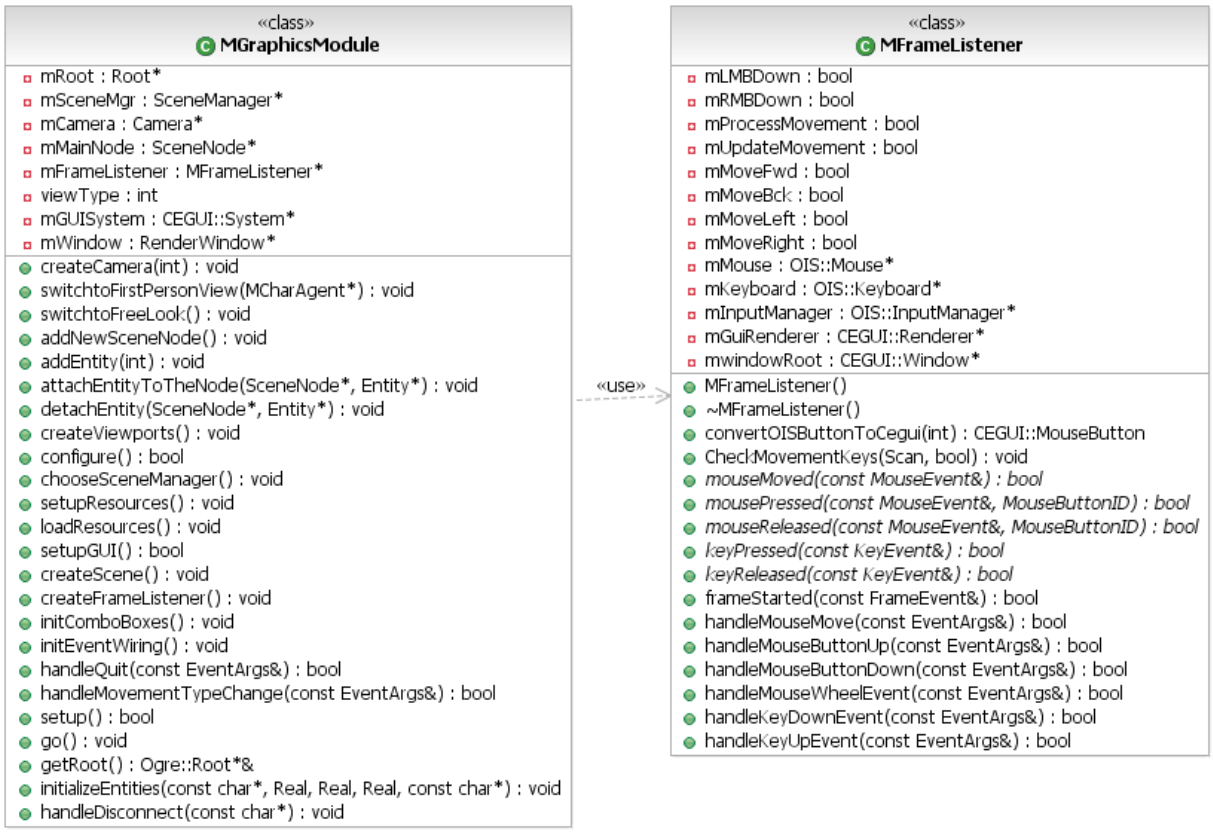

Figure – 26: Graphics Module Class Diagram

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 module uses *configure, createCamera*, *createViewports*, *setup*, *createScene* and go methods for the initialization steps. The *loadResources* and *setupResources* methods are used for loading *mesh* files 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.

The rendering loop of OGRE is modified in order to integrate it with other modules. The jobs done in OGRE rendering loop are carried to the simulation loop.

The input handler module is integrated in the graphics module. The *FrameListener* class handles the keyboard and mouse actions on the viewport and sets the attributes according to the triggered events. The *checkMovementKeys* method calculates the position of the object.

The OIS library (Object Oriented Input System) will be used for developing the FrameListener class which can be easily integrated to OGRE.

The *setupGUI* method initializes the menu interfaces developed using CEGUI. The rendering of these menus are done by the graphics module and the events will be handled by the FrameListener.

5.3.4 AI Module

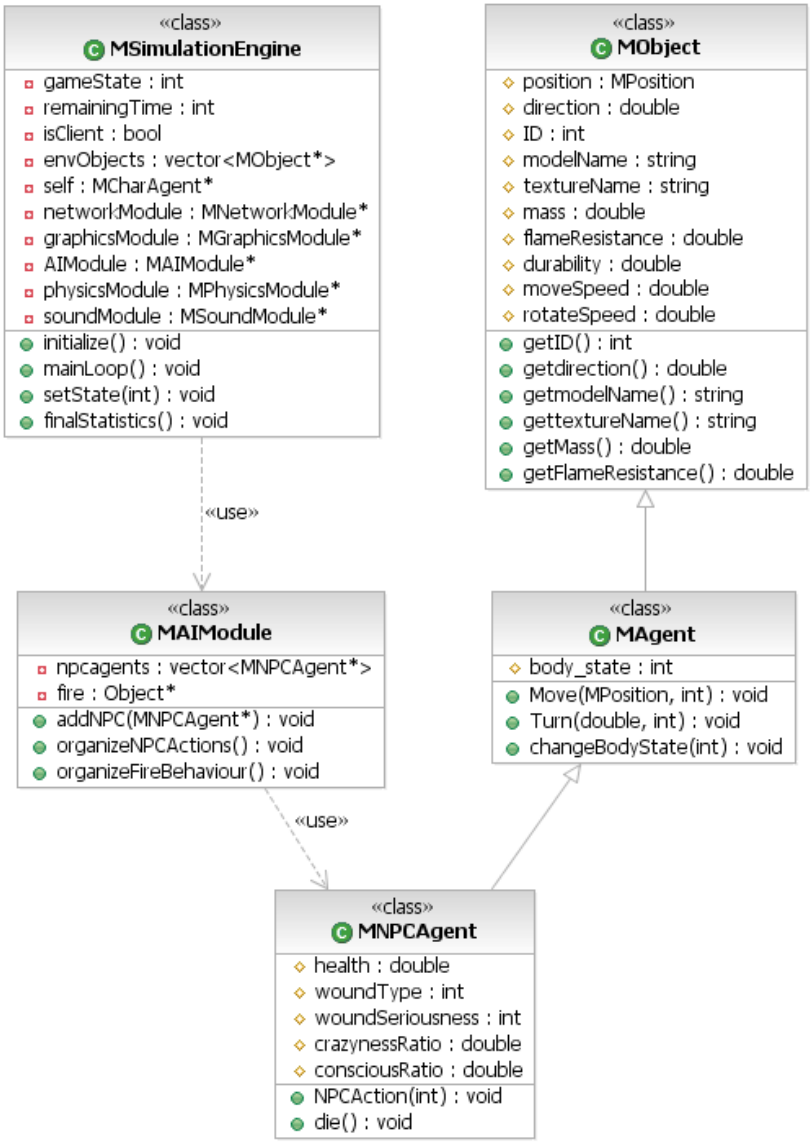

Figure – 27: AI Module Class Diagram

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. In addition to that, the load of the server AI will be distributed to the clients.

The AI module of the server will organize the all remaining non-playing characters in the main loop according to their current states. The *crazynessRatio* and the *consciousRatio* attributes of MNPCAgent class together effects the behaviors of passengers. As the *crazynessRatio* increase, the passenger will move around in an uncontrollable way and the situation will be harder for a user in the simulation. In contrast, a rise in *consciousRatio* can be considered as a sign of imperturbability for that passenger. The attributes *woundType* and *woundSeriousness* will make a difference between each wound that the passengers can have.

The fire will also be evaluated with this AI module and the physics module.

5.3.5 Physics Module

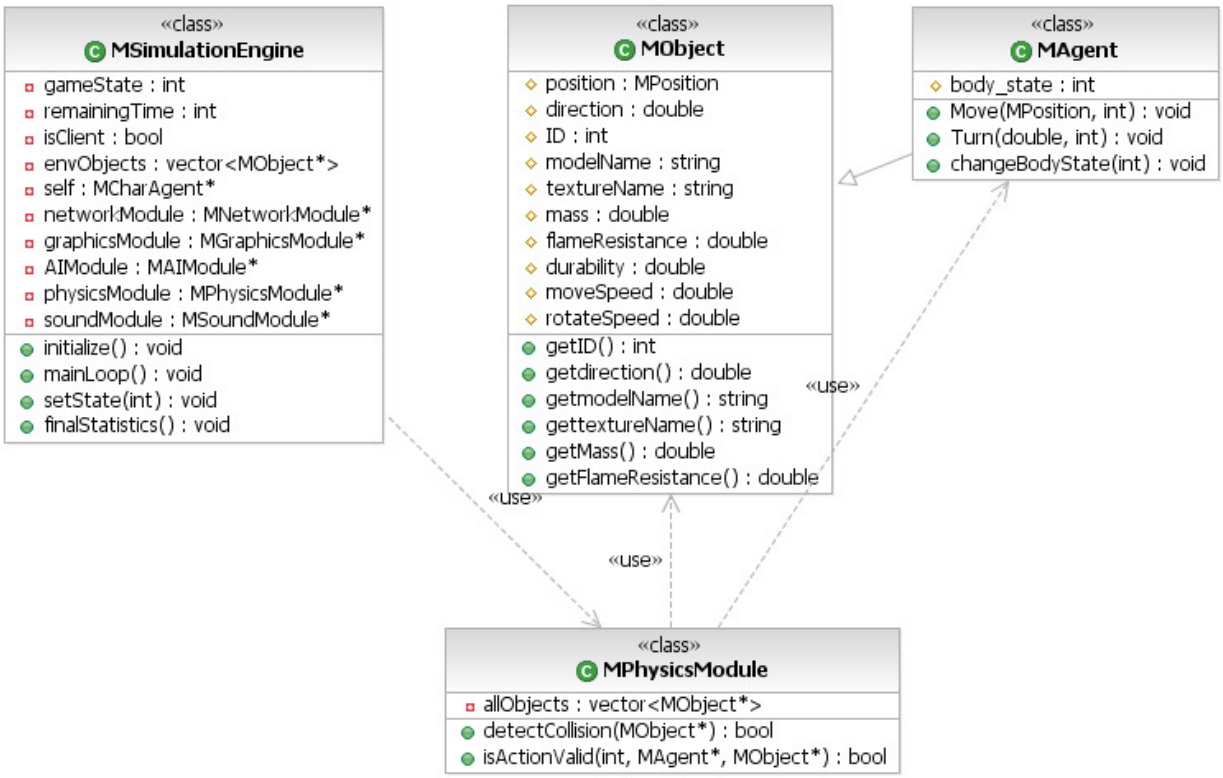

Figure – 28: Physics Module Class Diagram

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.

5.3.6 Sound Module

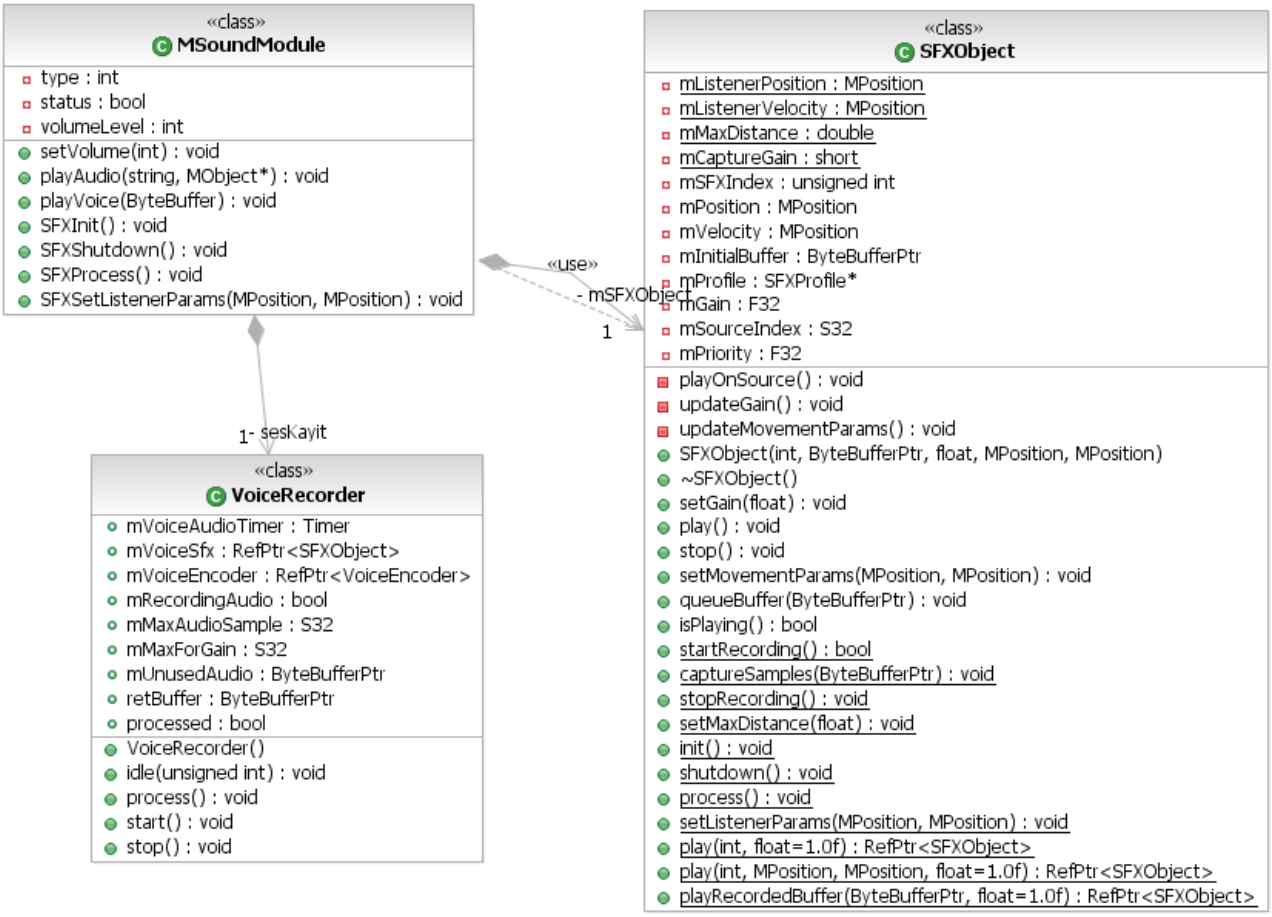

Figure – 29: Sound Module Class Diagram

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. The sound module will be developed both using DirectSound and OpenAL. The audio playing part of the module will be implemented using OpenAL and capturing the voice messages will be done using DirectSound.

A powerful ability of the sound module is generating sound effects according to the position of the sound source. The *playonSource* method provides this ability. The maximum distance that the sound reaches will be set by *setMaxDistance* method.

The sound module initializes the SFXObject for playing audios and voice messages and VoiceRecorder class for encoding the speech of users.

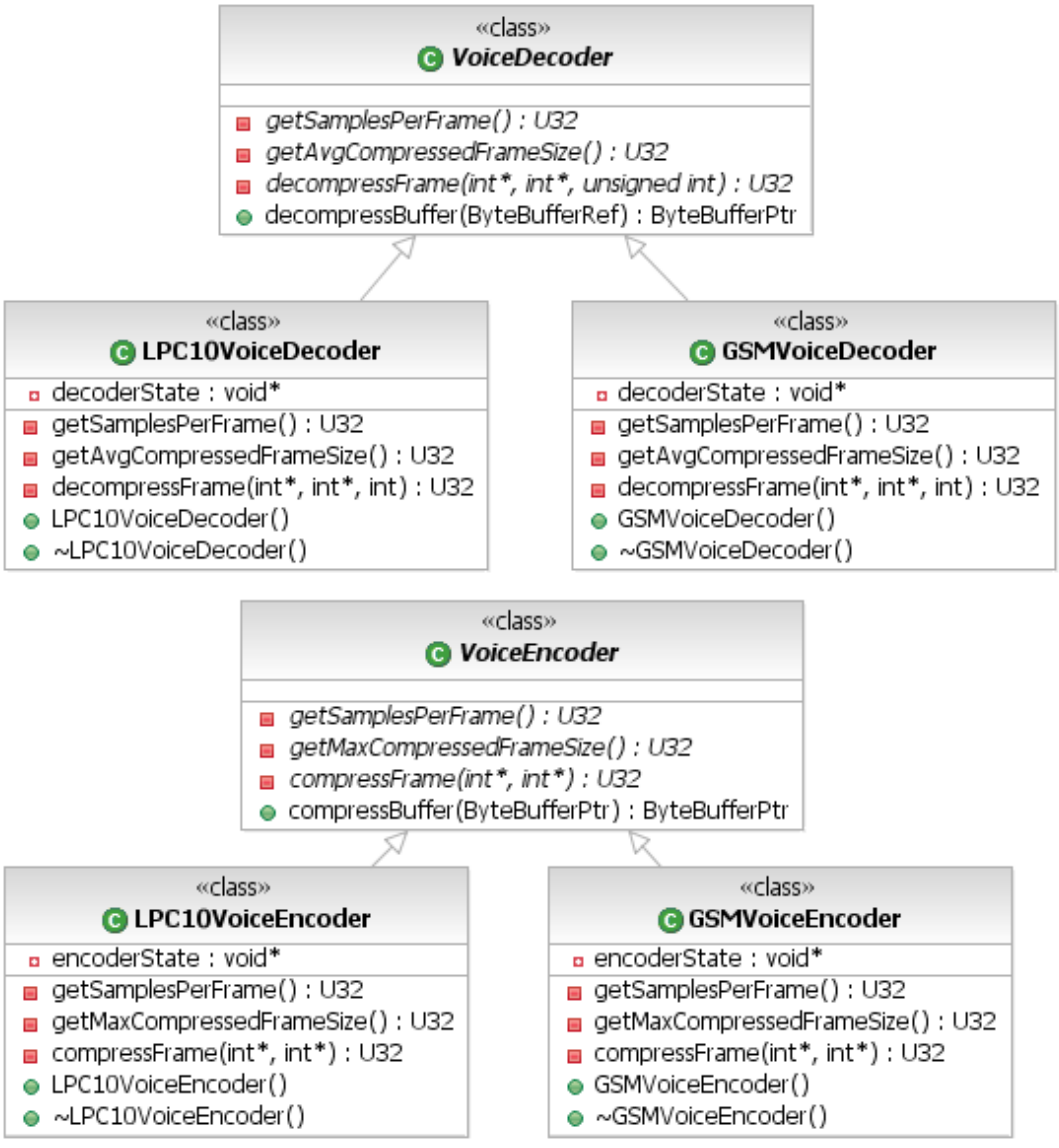

Figure – 30: Voice Codec Class Diagram

The voice encoder and decoder classes will be developed using libraries HawkVoice Direct Interface library. The library provides two different codecs for encoding and decoding voice messages. The LPC10 codec uses VBR(variable bit rate) algorithm and GSM uses a low-level speech compression algorithm. The LPC10 codec is currently integrated in the project which performed better compression then the GSM codec.

5.3.7 Agents

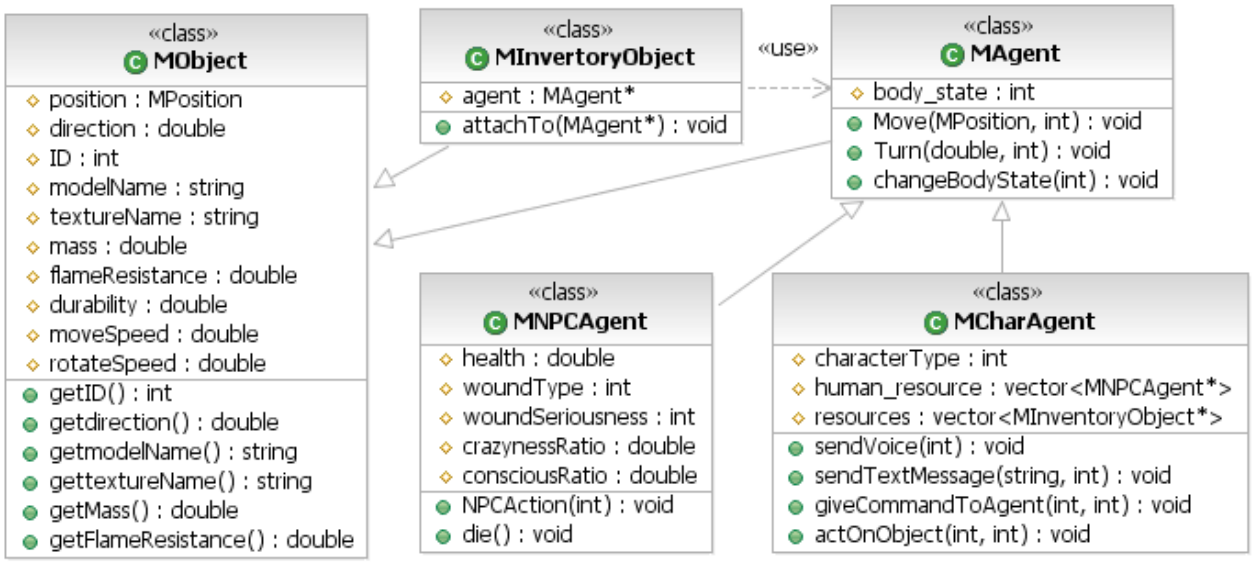

Figure – 31: Agents Class Diagram

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 simulation loop.

The character agents are the agents 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.

5.3.8 Objects

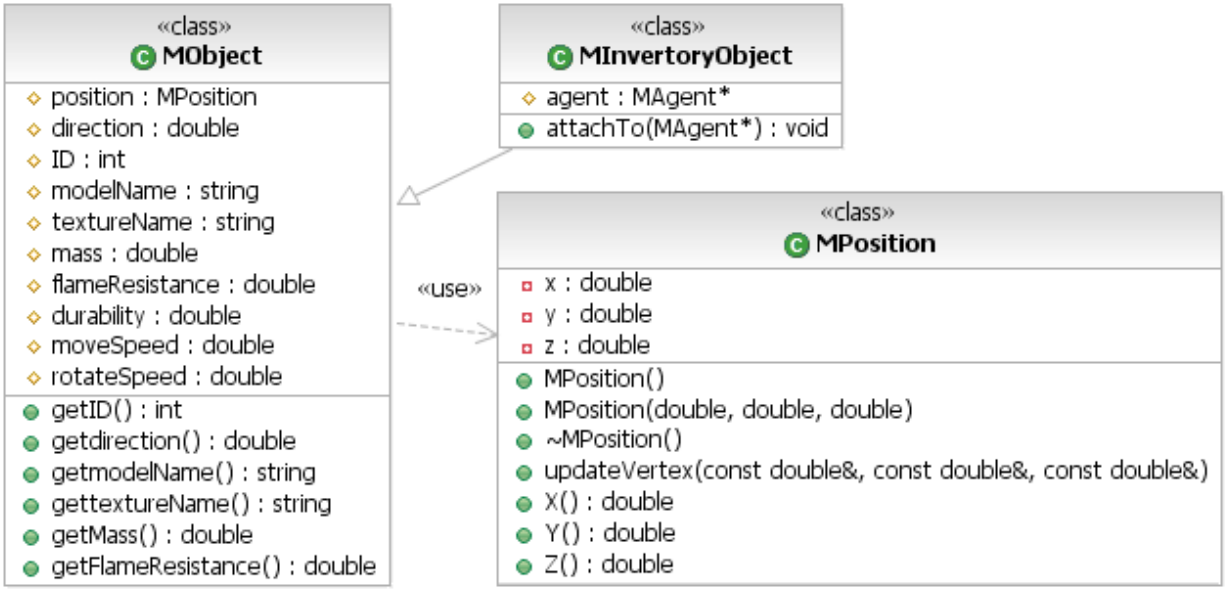

Figure – 32: Objects Class Diagram

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.

The object class has flameResistance, mass, durability attributes for the physics module and AI module. The modules will make calculations depending on these values and evaluate the actions related with these objects.

5.4 State Transition Diagrams

5.4.1 Simulation States Diagram

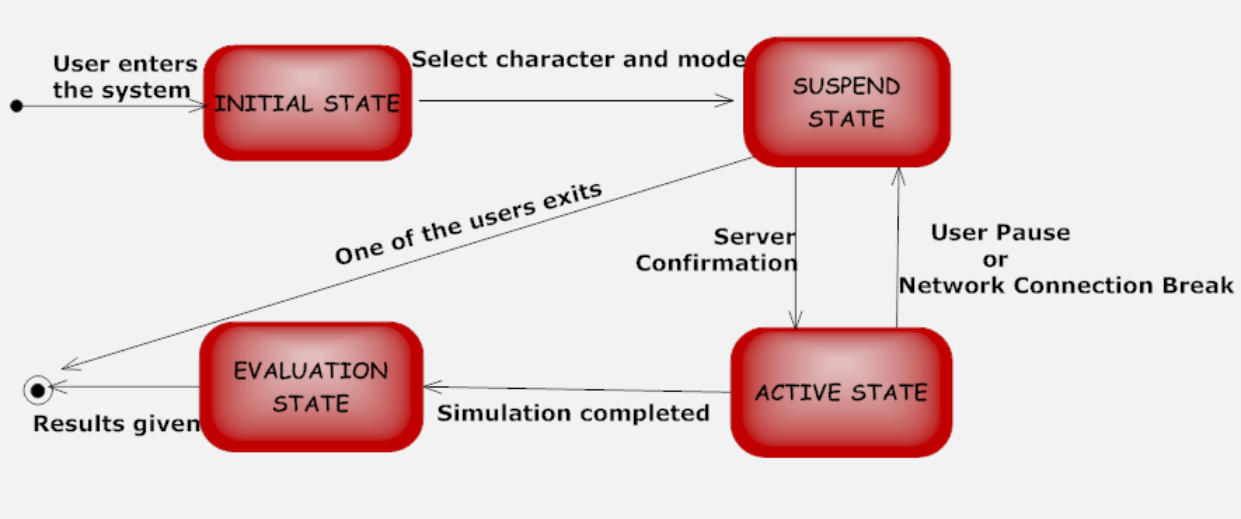

Figure – 33: Simulation States Diagram

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.

5.4.2 Object Interaction States Diagram

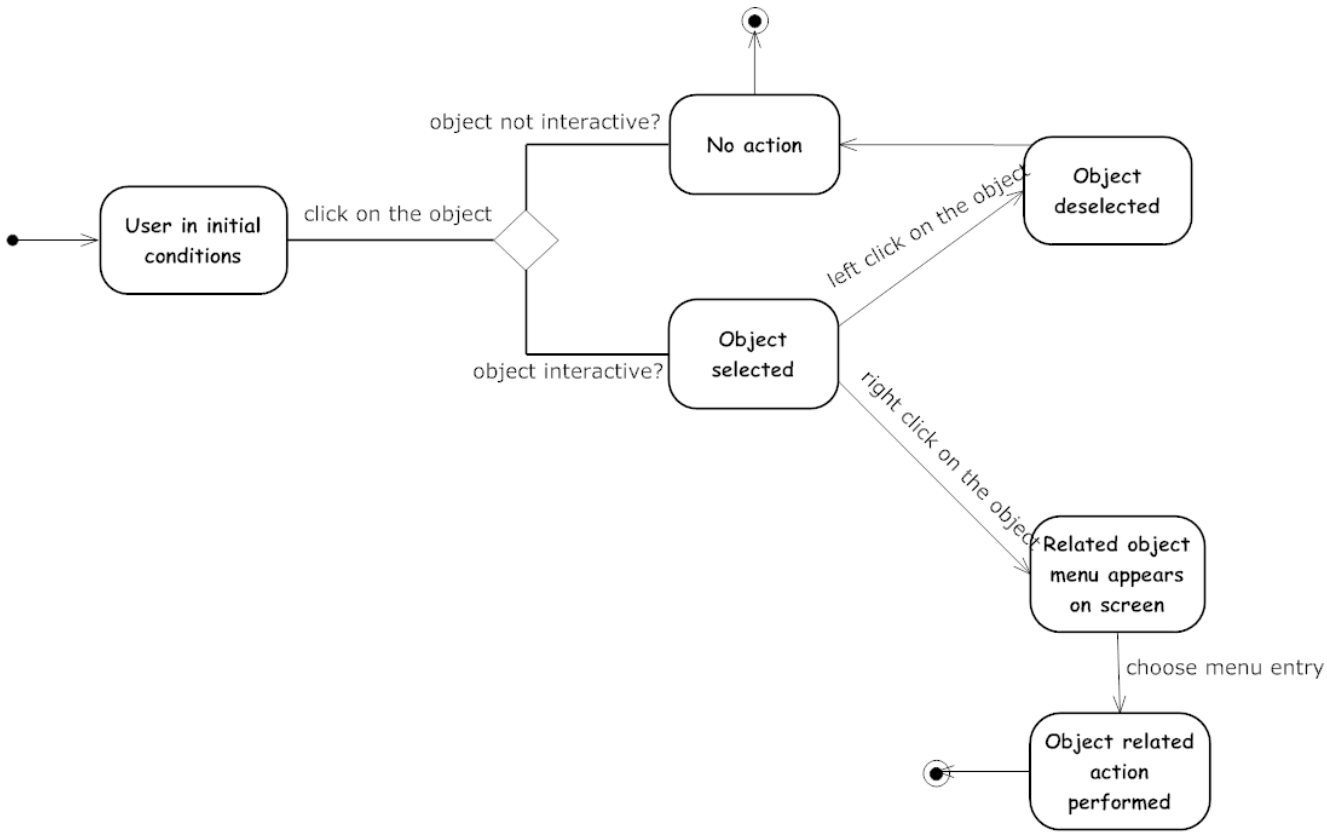

Figure – 34: Object Interaction States Diagram

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.

5.4.3 Menu States Diagram

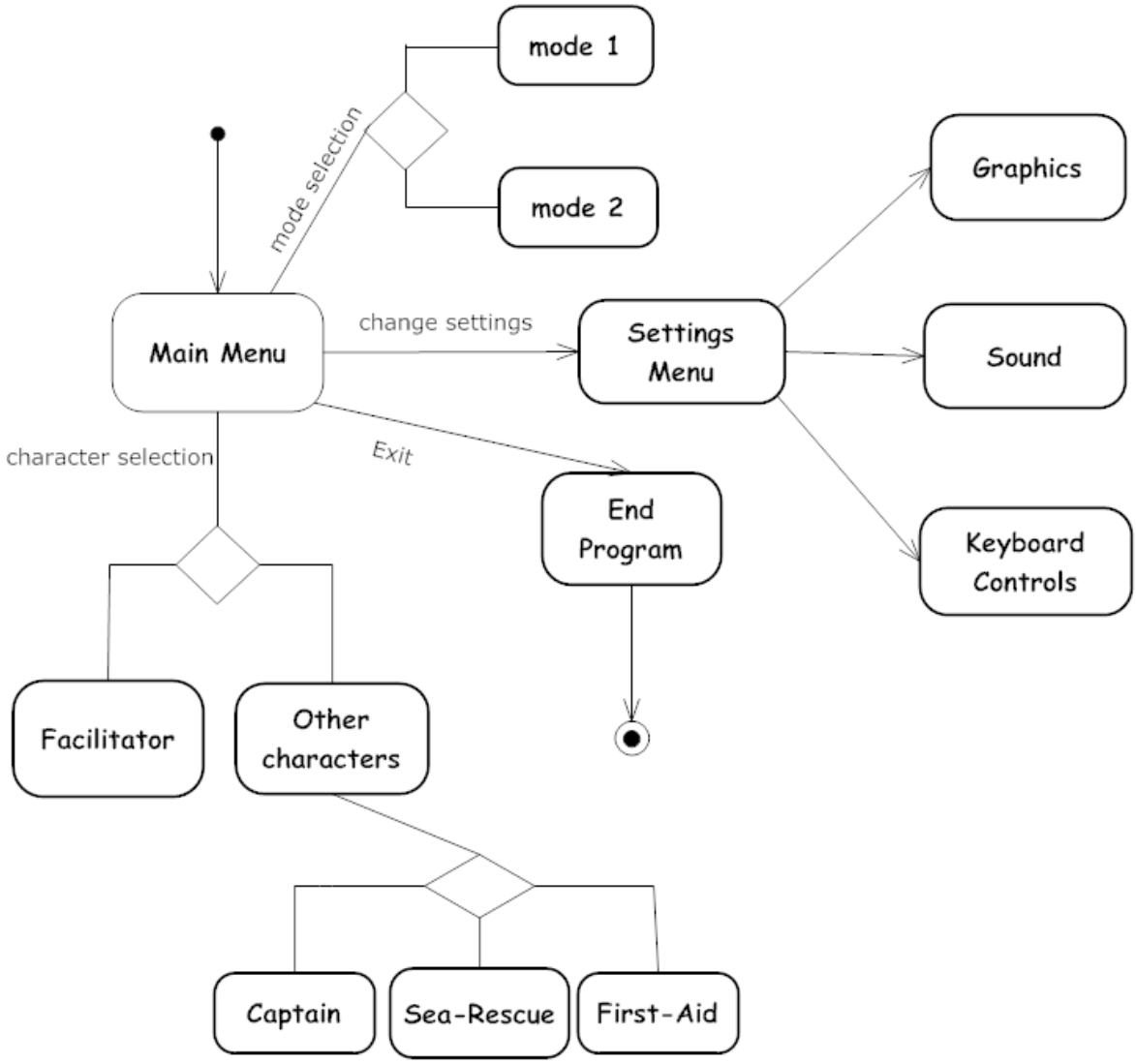

Figure – 35: Menu States Diagram

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.

5.5 Activity Diagrams

User Movement Activity Diagram:

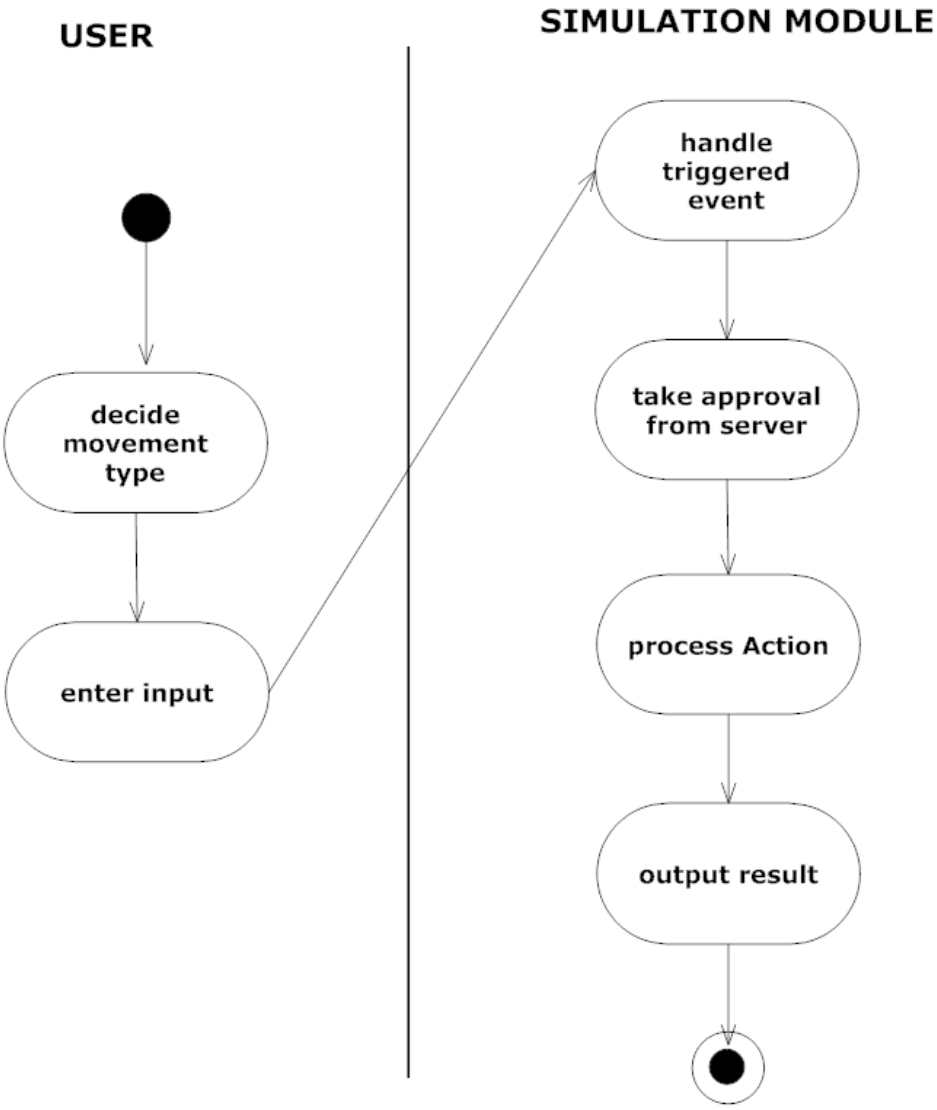

Figure – 36: User Movement Activity Diagram

User Command Activity Diagram:

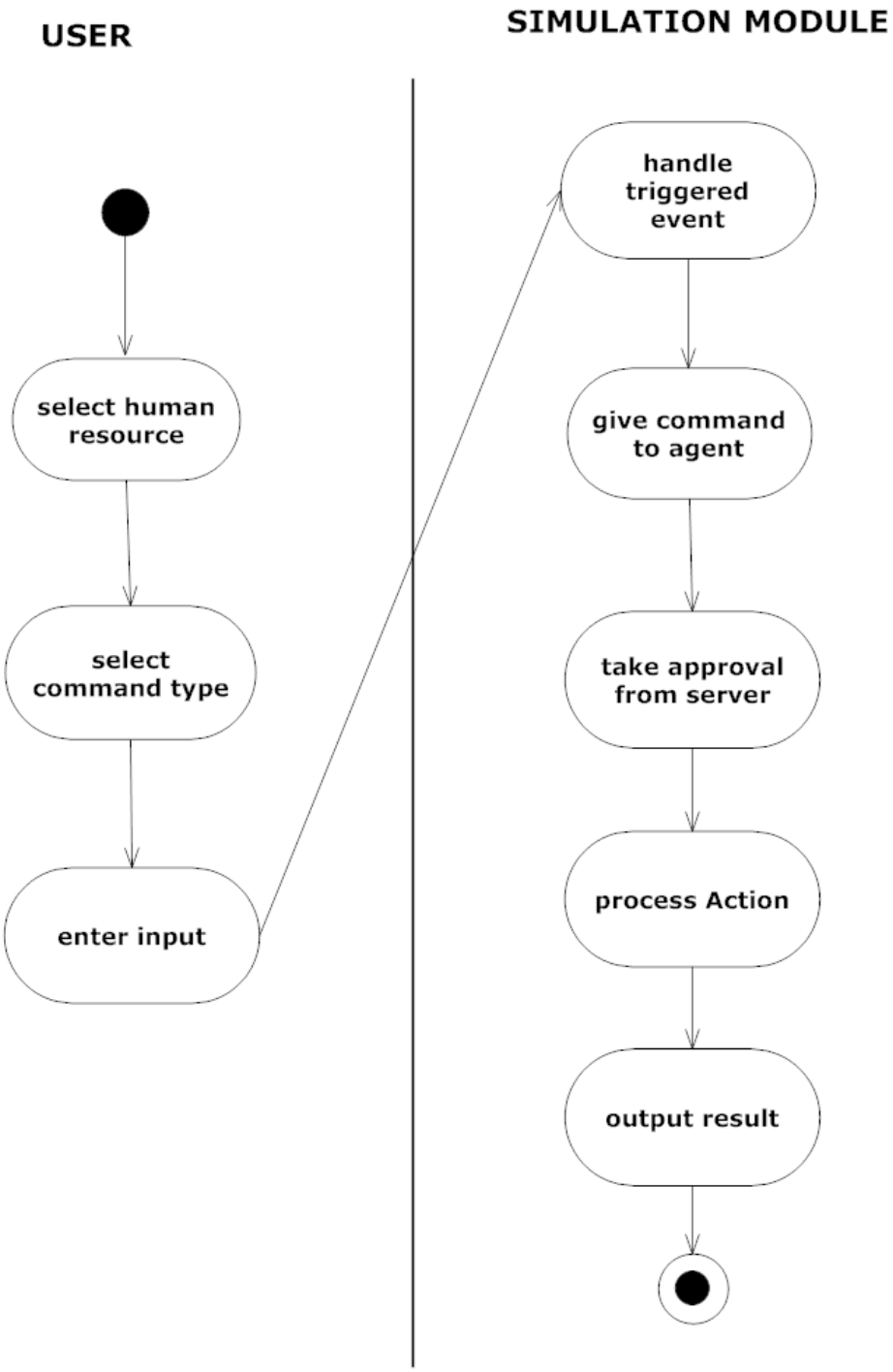

Figure – 37: User Command Activity Diagram

Menu Activity Diagram:

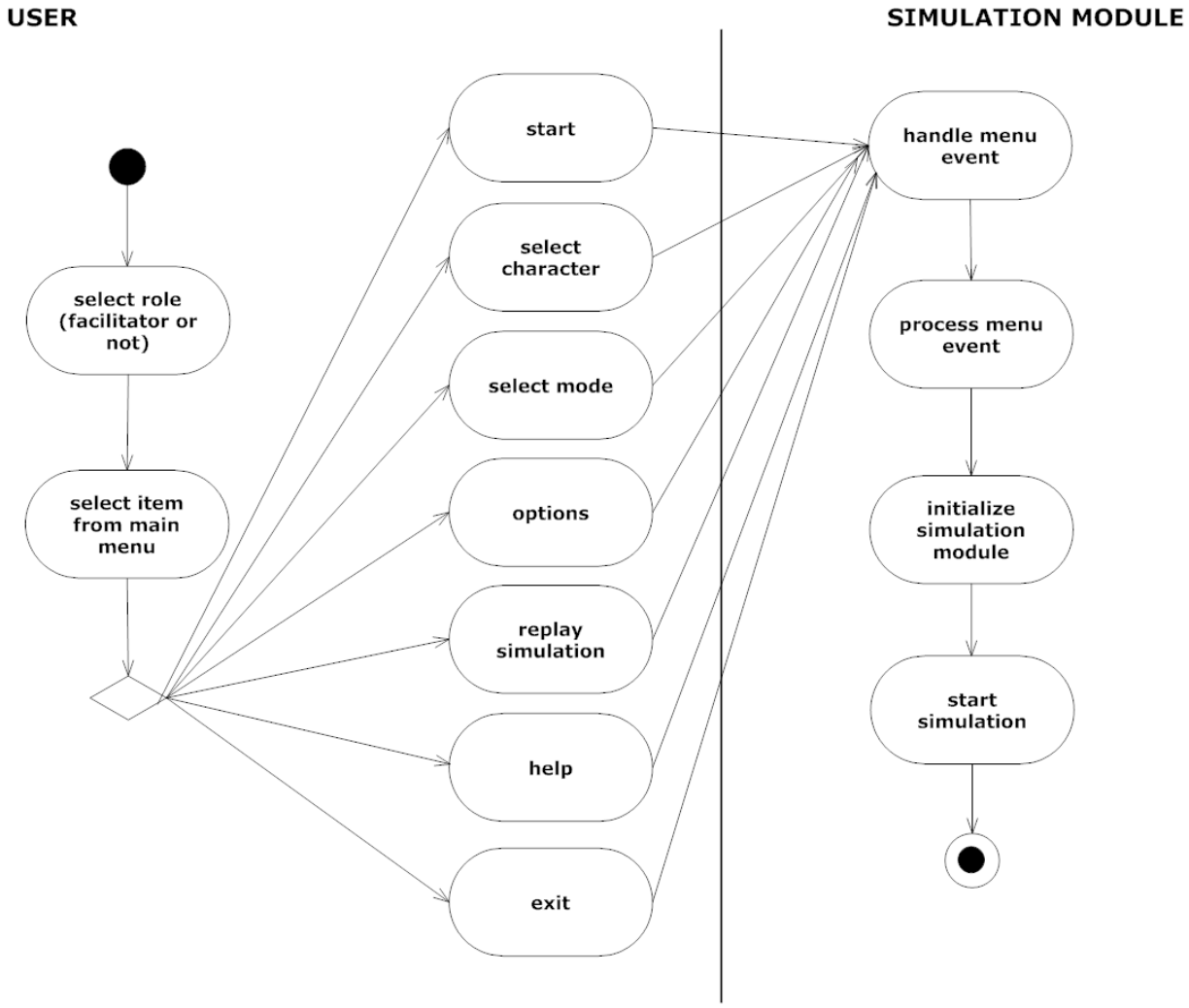

Figure – 38: Menu Activity Diagram

Manage Inventory Activity Diagram:

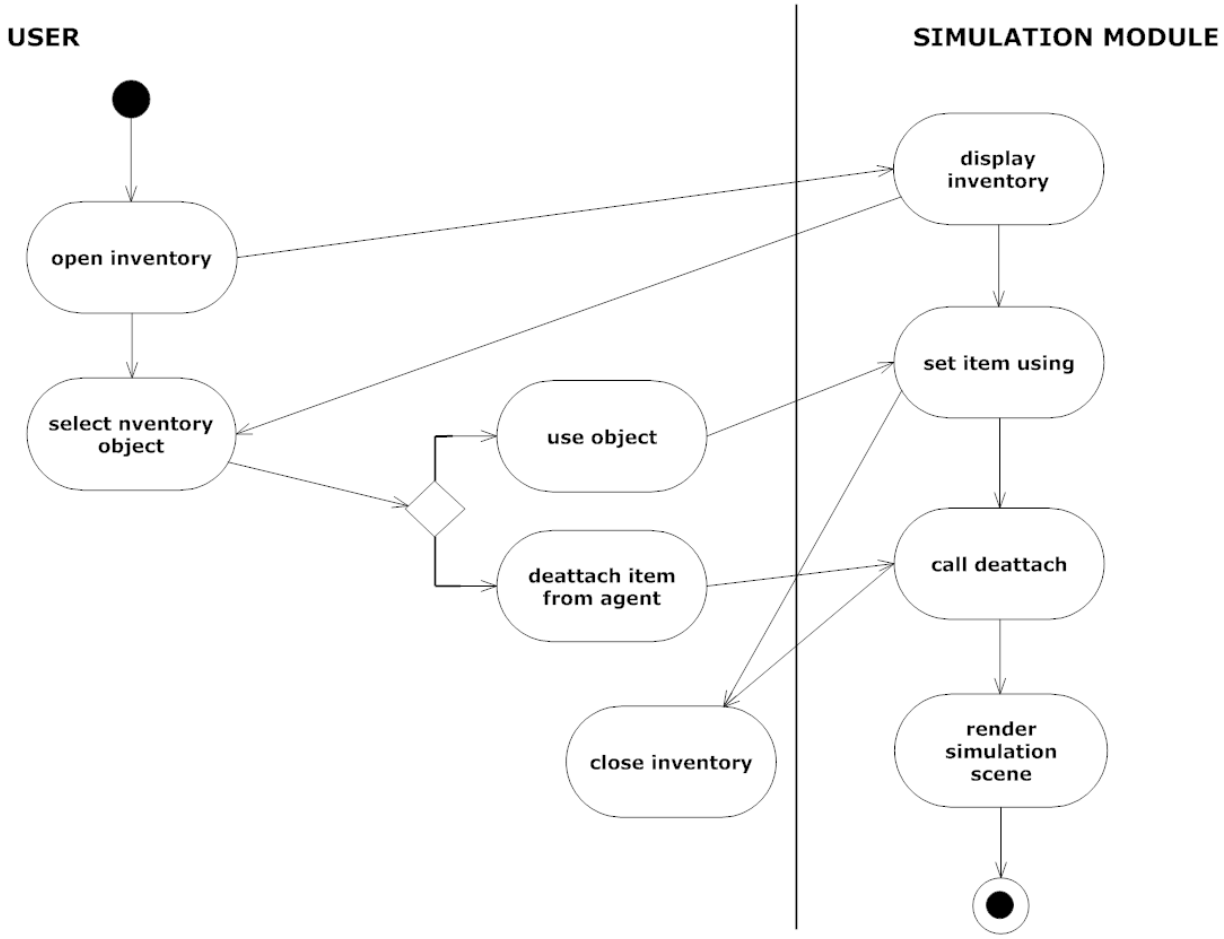

Figure – 39: Manage Inventory Activity Diagram

5.6 Sequence Diagrams

Simulation Sequence Diagram:

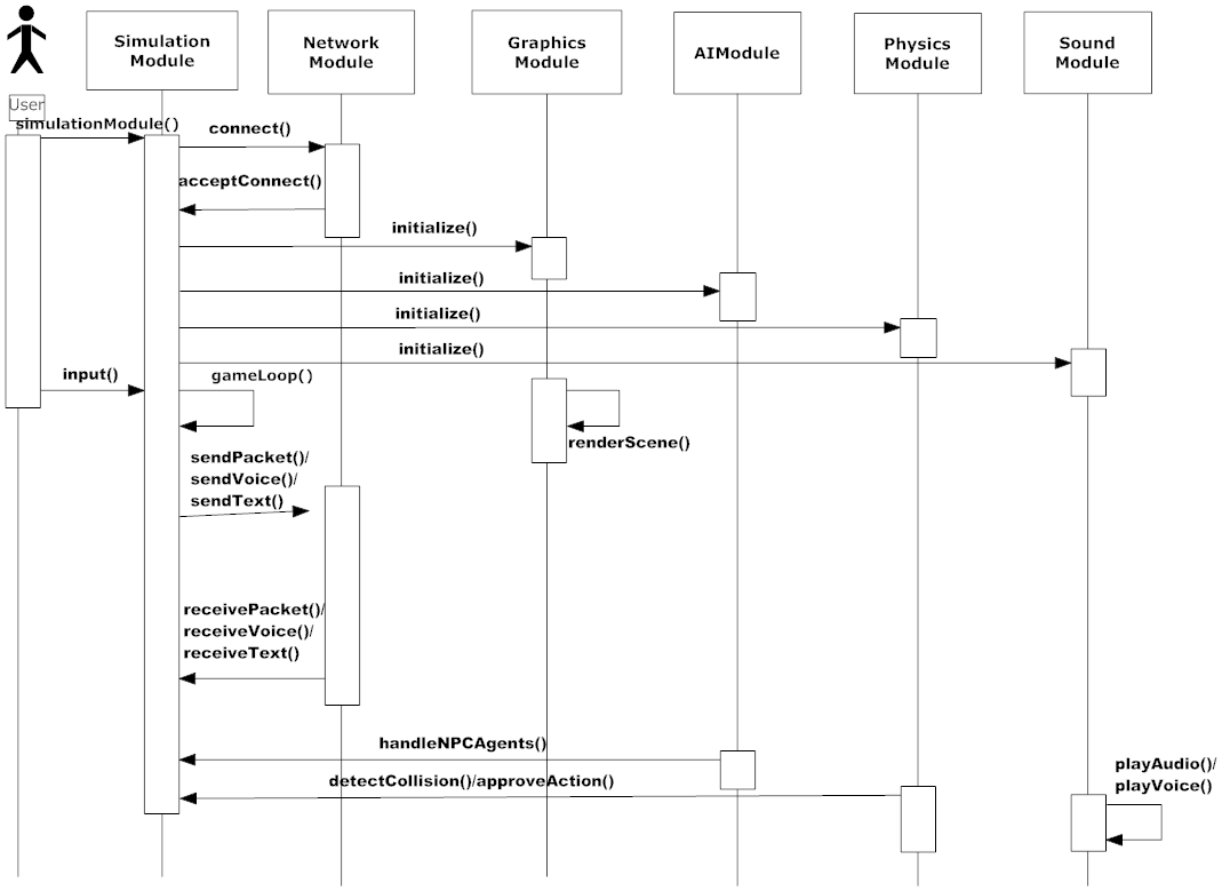

Figure – 40: Simulation Sequence Diagram

Menu Sequence Diagram:

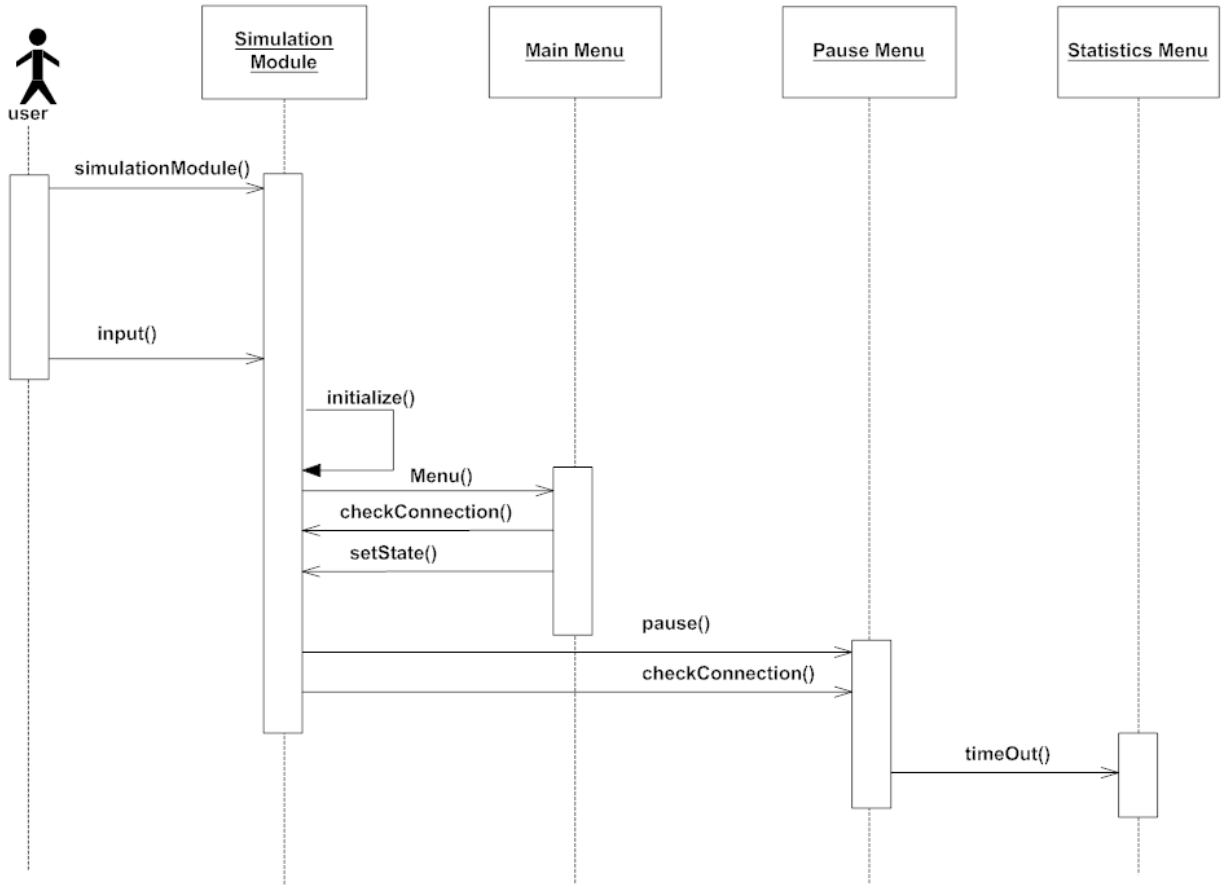

Figure – 41: Menu Sequence Diagram

5.7 ER Diagram

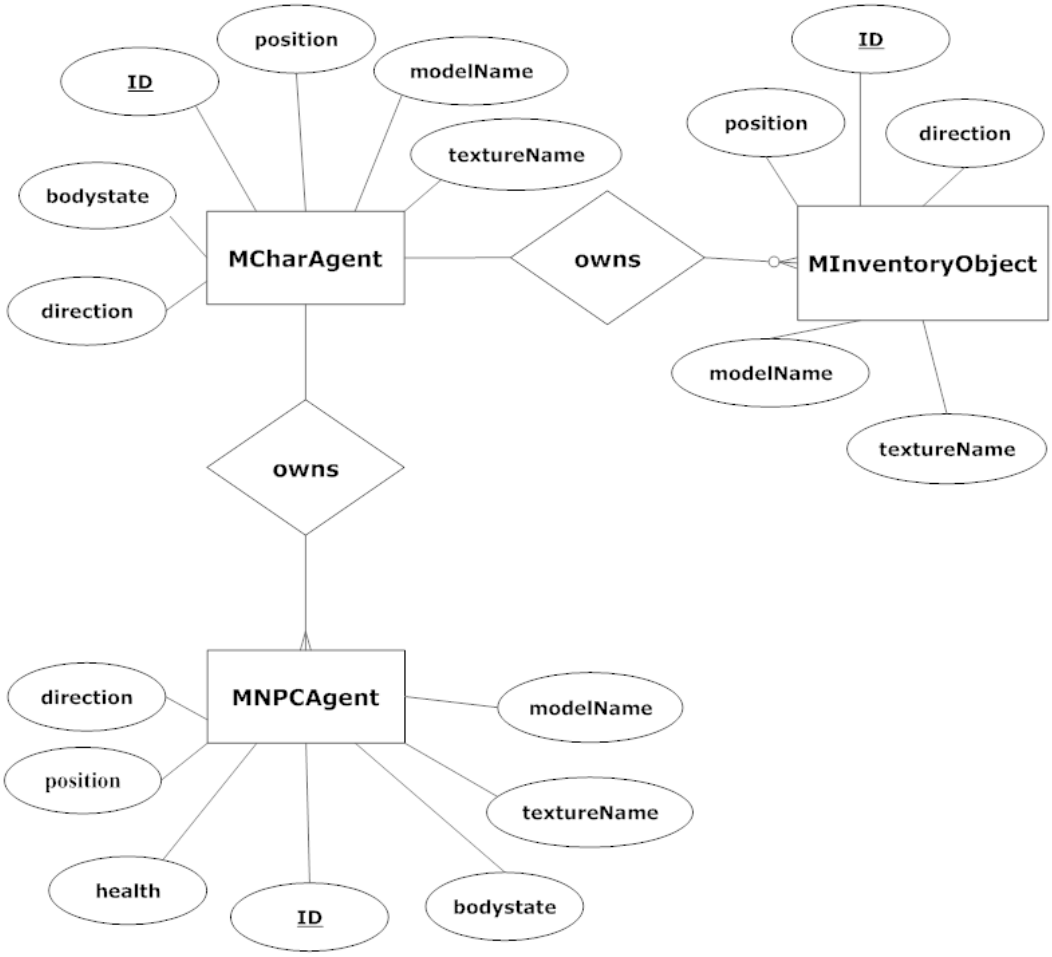

Figure – 42: ER Diagram

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.

6. 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.

6.1 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.

6.2 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.

6.3 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.

6.4 Variable Naming Conventions

The variable naming will not be very critical expect 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.

6.5 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.

7. PROJECT SCHEDULE

This section covers detailed information about the overall development and the remaining future work plans. The development section mainly contains the development progress of the prototype.

7.1 Current Stage of the Project

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

The development of prototype first started on different branches where the team members separately worked on network module, voice communication and graphics module. As the team members achieved a stage on the separated parts, these parts have been integrated in the simulation engine.

As the architectural design of the project states, the simulation engine firstly initializes the network module, sound module and graphics module according to the inputs given by the user. The prototype expects the user to enter information about the server/client information, the IP address of the server and connection port. The prototype also expects the user to enter a nickname if the user will be a client. The nicknames will be used on the server side to determine which client is in current connection with server.

The network module is developed using openTNL library. The general communications are done by using data packets. However, the network module contains some event communication methods for special cases. These events prevent the latency of some actions that are needed to be notified instantly. The network module contains broadcast methods for the server in order to broadcast the actions to the all clients.

The network module also handles the voice communication messages. As the user presses the key ‘R’, the voiceRecorder records the voice and using the LPC10 codec it encodes the captured voice into a byte buffer. The encoded voice is transmitted through the network module using the c2svoice method. As the server receives a voice message, the message is broadcast to the all clients except the sender in order to prevent the echo effect in the client. The voice message is decoded using the sound module and the decoded buffer is queued to the sound modules play buffer. The queued buffer is played as the simulation loop ticks.

In order to integrate the graphics module with network module, the OGRE library's rendering loop is modified such that OGRE's rendering function can be called in each simulation loop tick. Consequently the simulation module can handle both network module and graphics module in the simulation loop.

The simulation module initializes the graphics module so that the mesh files, GUI layouts and textures are loaded from the configuration files. The graphics module attaches the entities notified by the clients to the scene nodes. The graphics module creates different scene nodes for each client attached to the main node. However, no entity is created for the server side since it is considered as the facilitator. Each client can select a mesh file for its object and can translate it. The translation of the client is notified to the server using network module and this movement is broadcasted to the other clients. Using the translation information each client updates the rendered scene with the new object coordinates.

New graphical user interfaces are implemented with using CEGUI. These menus do not take part in the prototype since they are irrelevant to the context of the prototype. These menus are created in XML structure and the implementation occurs in the .layout files. Various objects that are provided by CEGUI are used, like buttons, radio buttons, textboxes, sliders etc. At the design step of the interfaces, usability was the main concern.

7.2 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 implementation will be discussed.

After the prototype release, project members will utilize the term holiday for finding 3D models for the simulation. If the searches terminated unsuccessfully, the necessary models for the project like the cruise ship and the human models for the characters will be modeled using 3D Studio Max. Since it is a time consuming process which also requires creativity, the most suitable time period is the term holiday.

Physics module of the simulation will also be added to the implementation after the models are integrated. As stated previously ODE is used as physics engine, which is widely used with OGRE.

After these steps are handled, previously designed interfaces will bind to the implementation and character specific instances of the clients will begin developing.

7.3 Gantt Chart

#	Name	Start	Finish	Duration
1	**Managing Project**	**9/28/2007**	**5/30/2008**	**176.0 d**
2	Project Management	9/28/2007	5/30/2008	176.0 d
3	DEL:Proposal Report	10/8/2007	10/8/2007	0.0 d
4	DEL:Analysis Report	11/4/2007	11/4/2007	0.0 d
5	DEL:Initial Design Report	12/3/2007	12/3/2007	0.0 d
6	DEL:Detailed Design Report	1/18/2008	1/18/2008	0.0 d
7	DEL:Prototype Demo	1/18/2008	1/18/2008	0.0 d
8	**Developing Concept**	**10/1/2007**	**12/4/2007**	**47.0 d**
9	Scenario Decision	10/1/2007	10/31/2007	23.0 d
10	Developing Scenario and Characters	11/1/2007	12/4/2007	24.0 d
11	Character and Object Design	11/8/2007	11/20/2007	9.0 d
12	Simulation Flow Design	11/8/2007	11/20/2007	9.0 d
13	Interface Design	11/15/2007	11/30/2007	12.0 d
14	**Producing Simulation Resources**	**9/28/2007**	**5/5/2008**	**157.0 d**
15	Resource Gathering	9/28/2007	5/5/2008	157.0 d
16	Character Modeling	11/20/2007	3/5/2008	77.0 d
17	3D Objects Modeling	11/20/2007	3/5/2008	77.0 d
18	Texturing Models	3/5/2008	5/5/2008	44.0 d
19	Character Animations	3/5/2008	5/5/2008	44.0 d
20	3D Scene Modeling	11/20/2007	3/20/2008	88.0 d
21	Sound Selection	1/25/2008	5/5/2008	72.0 d
22	Scene Lightning	2/5/2008	5/5/2008	65.0 d
23	**Developing Simulation Engine**	**9/28/2007**	**5/5/2008**	**157.0 d**
24	Open Source Engine Analysis	9/28/2007	1/1/2008	68.0 d
25	Identifying Engine Requirements	9/28/2007	11/22/2007	40.0 d
26	Designing Engine	11/15/2007	12/12/2007	20.0 d
27	Designing Engine Architecture	11/15/2007	12/12/2007	20.0 d
28	Prototype Implementation	11/20/2007	12/7/2007	14.0 d
29	3D Graphics Module	11/20/2007	5/5/2008	120.0 d
30	Basic Physics Module	1/25/2008	5/5/2008	72.0 d
31	I/O Module	11/20/2007	5/5/2008	120.0 d
32	Sound Module	12/10/2007	5/5/2008	106.0 d
33	Basic AI Module	1/25/2008	5/5/2008	72.0 d
34	Network Module	11/20/2007	5/5/2008	120.0 d
35	GUI Module	11/20/2007	5/5/2008	120.0 d
36	**Final Release**	**5/6/2008**	**5/30/2008**	**19.0 d**
37	Alpha Testing	5/6/2008	5/9/2008	4.0 d
38	Beta Testing & Debugging	5/12/2008	5/15/2008	4.0 d
39	Installation Manual	5/15/2008	5/26/2008	8.0 d
40	User Manual	5/15/2008	5/26/2008	8.0 d
41	DEL:Final Demo	5/30/2008	5/30/2008	1.0 d