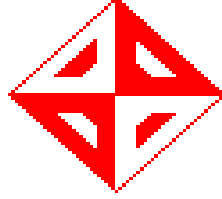


Middle East Technical University
Department of Computer Engineering



CENG491
Computer Engineering Design I
Design Report

ÖZGÜR YAZILIM



Özgür Özgür
Fırat Erdoğan
Onur Demircan
Abdulkerim Mızrak

1. Project Definition	3
2. Design Goals	3
3. Overall Architecture	4
4. Scenario	5
5. MODULES	7
5.1 Graphical User Interface	7
5.2 Game Engine	11
5.2.1. Server Game Engine	11
5.2.2. Game Engine (Client side)	11
5.3 Input Module	11
5.4 Menu Module	12
5.5 Artificial Intelligence Engine Module	17
5.6 Graphic Engine Module	17
5.7 Network Module	18
5.8 Audio Module	23
5.9 Chat Module	23
6. CLASS DEFINITIONS	24
6.1 BaseObject Class	24
6.2 Position Class	25
6.3 Treasure Object Class	25
6.4 Food Object Class	26
6.5 Furniture Object Class	28
6.6 Character Class	29
6.7 Virtual Player Class	30
6.8 Map Classes	30
6.8.1 POINT	30
6.8.2 WINDOW	31
6.8.3 DOOR	31
6.8.4 ROOM	32
6.9 Network Class	32
6.10 Audio Class	34
6.11 GameEngine Class	34
6.12 Input Class	35
6.13 AIEngine Class	35
6.14 GraphicsEngine Class	36
6.15 Chat Class	37
7. TOOLS	38
8. CLASS DIAGRAM	39
9. USE CASE DIAGRAM	40
10. DFD	43
10.1 Data Flow Diagram Level 0	43
10.2 Data Flow Diagram Level 1	43
10.3 Data Flow Diagram Level 2	45
11. SEQUENTIAL DIAGRAM	49
12. PROJECT SCHEDULE	50
12.1 Project Task Set and Workpackages.....	50
12.2 Gantt Chart.....	51

1. PROJECT DEFINITION

Treasure Hunt is a massively multiplayer online game (MMOG) with 3D graphics. It is designed to support hundreds of online players to interact and play together in a virtual environment.

The game has a nice scenario. Players are restricted in connected multi-storey buildings with lots of rooms connected by several paths. Players are given the chance of choosing one of the pre-defined characters. In the game, characters have calorie values increase with the food they eat. On the other hand as time passes, movement and metabolism result in the decrease of calorie values. There are some predefined calorie limits which determine the characters movement capability. According to calorie value, character can walk or run or crawl. There are some food objects that characters can compensate the energy loss by eating. Characters have special characteristics, for example, some characters are vegetarian. Vegetarian characters can not eat meat. Moreover, some characters are allergic to some kind of food. Players dedicate themselves to find the treasure. Players approaches the TREASURE, wandering inside the building and passing steps by finding step related treasure objects. Finally the one most skillful, fast and wise hunter will reach the TREASURE.

2. DESIGN GOALS

Treasure Hunt is going to be a massively multiplayer online game which supports 3D Graphics Rendering, Multimedia (sound), and Game AI.

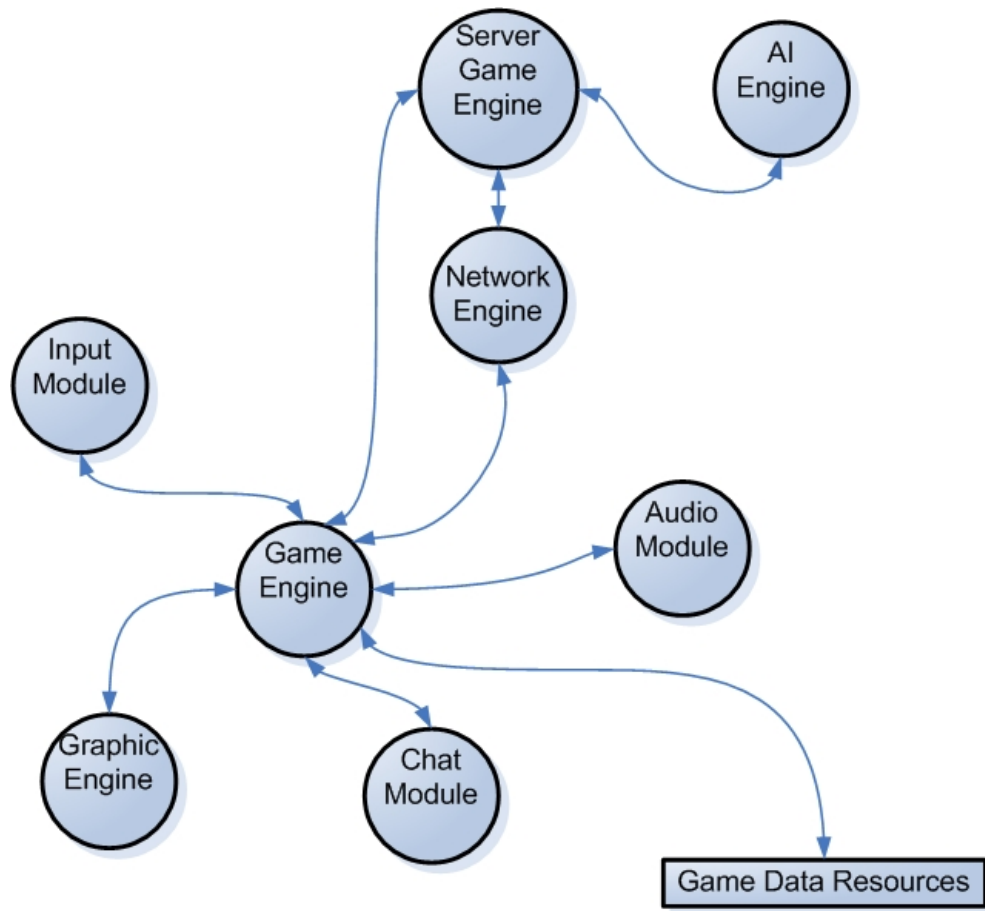
In this report, our main goal is to show the design made by our group in order to document our project properly. This report is going to describe a detailed functionality of the system, main components of the system and their interaction with each other, graphical user interfaces, and a gantt chart.

In general, we will not only try to implement a game with successfully played in any way, but also try to satisfy the following software design goals as well.

- **Object-Oriented** – Object oriented design results in self-contained modules that are easy to manage and maintain. The modular design of Treasure Hunt makes it easy to implement, test, maintain, and extend.
- **Extensibility** – The system evolution for games such as Treasure Hunt is endless, so the design should allow for the implementation of future requirements with minimal changes to the current design. We are planning to build a system with ‘plug-in’ property. So that, we will add new objects easily to the system.
- **Ease of use** – It is important that the application has a clean and easy to use interface. The interface should be intuitive and look very familiar to both new and experienced gamers.

- **Performance** Treasure Hunt is designed to minimize the resources occupied and to maximize the output in terms of frames per second. The game is designed to be responsive, robust, and error-free in any system that meets the minimum system requirements specified in the non-functional requirements.

3. OVERALL ARCHITECTURE



Game Engine is the core component of overall architecture. This component controls the game mechanism. The synchronization between graphic, audio, chat and network engine is provided by game engine. Moreover Game Engine is responsible for providing step related puzzles. In the game we have game data sources such as models, characters etc. The puzzles are also in the game data resources. The puzzles are kept in a database according to the difficulty levels.

Game server engine is responsible for initialization part of the game. When the game starts position of characters, food objects, treasure objects are designated by game server engine. The positions are sent to game engine through network engine. The network engine provides interaction in the game with other players by providing connection between game engine and game server engine.

Chat module coordinates with game engine and broadcast the data using network engine. Inputs are dispatched by input module and processed in the game engine. Game engine processes the inputs and output related data from game data sources. The processed data in the game loop is sent to graphics and audio module. Graphic engine renders the output data from game engine. The sound is broadcasted from audio module.

4. SCENARIO

Being an MMOG Treasure Hunt has runs on two sides which are server and client. A player should install the game on his PC, and besides he should have a stable Internet connection to take part in the game.

Game sessions are initiated at every 5 minutes on server after the end of last session. A player is not able to join a running game session.

At joining the game phase players should involve in an existing team or should initiate a brand new team. Teams can involve at most 4 people. The player who initiated the team is assigned as the team leader by server. If the team leader disconnects or leave the team, server automatically assigns another member of the team as the new team leader.

At the end of this 5 minutes joining period, the server checks for the balance of teams. If there is an apparent difference between teams, such as single player teams against four player ones, server reassigns some of teams (actually the weak ones) to ensure that teams are balanced. At the time of game start, the server informs the clients by sending team number they belong to and also the player_id of team leader. The server also assigns a position to every client randomly, and sends this position data to related client machines. Data packets involving the positions of treasure and food objects, which are also placed in a random manner, are broadcasted to every client.

Game takes place in multi-storey buildings close each other. Each storey consists of many rooms. The rooms have different sizes and involves furnitures and treasure or food objects if not taken yet. The outside world contains some static objects such as trees, cars, etc. There can also be some food objects, but there's no way to find treasure objects outside the buildings.

After the initialization part described above, the game starts by assigning first objective to every group. The first four objectives involves finding four treasure objects. The last objective is the fifth one, which is denoted as TREASURE and makes the team who finds it first the winner. Each objective defines a step in game. At each step, number of treasure objects will be equal to the number of teams. Since the objective is assigned to the teams, a team has the chance to advance the next step when just one of the members found the treasure object. At this point the team will face a puzzle which is a multiple choice question. The team leader has the role of spokesman, i.e. he is the only one capable of answering the question. This encourages members to use the in-team chat to decide the right choice. In the case of wrong answers, team loses time and energy which decreases their chance to win the game. Every wrong answer triggers a new puzzle until a right answer is reached. When answered correctly, team advances to next step and new objective is assigned to the team. Another important thing about the puzzles is that the difficulty level increases throughout advancing the steps.

As mentioned above, players start the game with an initial energy value, which is 800. This energy decreases through time. This energy level limits players movements according to some threshold values. A player is not able to run when his energy is under 700, and he is not able to walk at normal speed when it is under 500. He is not able to make even slow walk when under 300, and so crawls only. The energy level can decrease until a minimum value, which is 200. This energy can be increased by eating food objects found in the environment. But like the minimum threshold value, there is also a maximum limit of 1000 which cannot be passed no matter how much food is eaten. Another important point here is the collectivism about food among the teams. Team members can stock a limited amount of food in an imaginary food basket, and each member is able to consume food from this basket. Since characters in game have some special characteristics like being vegetarian or allergic as mentioned above, the teams whose members cooperate has more chance to win the game. For

example, a member will not suffer from shortage of food -even if he faces with allergic foods all the time- by the help of food founded and stored by the other team members.

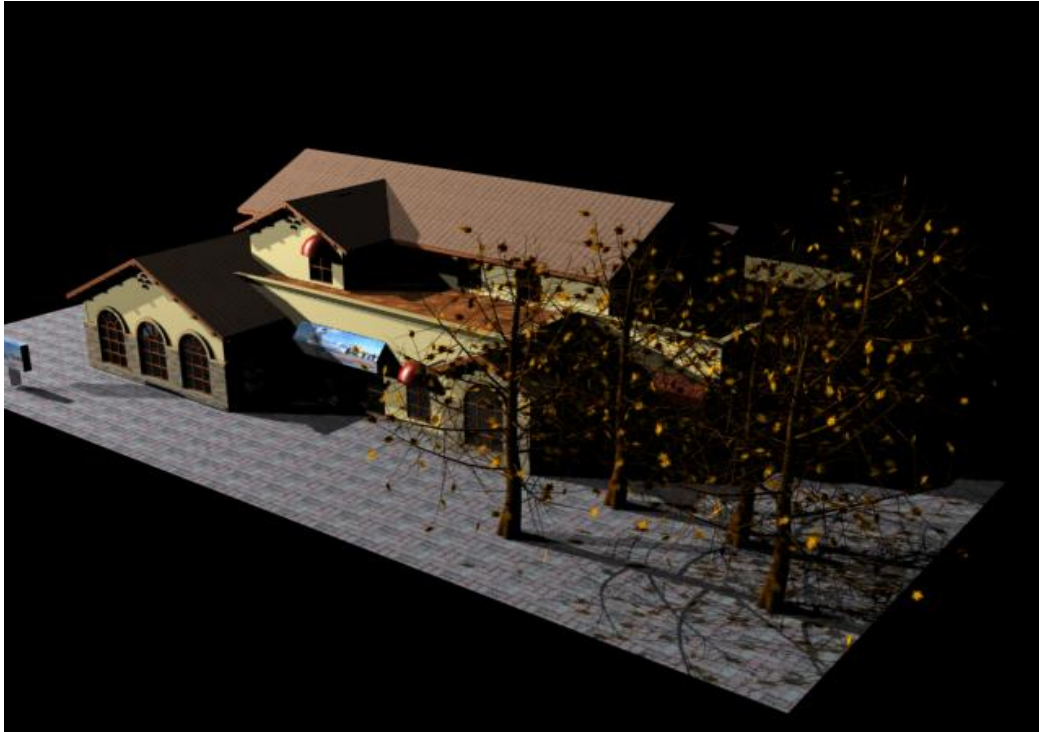
Another important concept about game flow is the amount of food in the environment. If the number of food objects in the environment is very low, which may cause the game to last after a really long time, the server intervenes this situation by embedding some food objects randomly into environment and of course broadcasting this information to every client machine.

5. MODULES

5.1 Graphical User Interface

Interface 1.

The player will see this nice home when he first start to the game. But inside of the home will not be as small as it seems from outside. Game takes place in multi-storey buildings close each other that each storey consists of many rooms.



Interface 2.

This is another interface from outside of the buildings.



Interface 3.

We will try to create such an interface below in the home. As seen, this place is a kitchen and there are some furniture used in the kitchen. The objects and foods will be on, in, or under such furnitures. The player will look for objects and foods during the game.

On the left of the screen, the objects that have to be collected are listed. The objects that have a sign '?' on them have to be found by the group as soon as possible. If the object is found the '?' sign will be disappear.

On the right of the screen, foods that are collected are listed. Each player have to collect food. Each food will be appear on this part, if a food is eaten by a player in the group this food will be disappear.

On the top of the screen a button, "OPTIONS", is available. It will be mentioned in the game menu part of this report.



Interface 4.

This interface is another camera position of the Interface 3.

Let's mention about the bottom part of the screen.

The most-left part shows the nick name and the calorie value of the player. If the calorie value is decrease the player has to have foods, otherwise the player will slow down very much.

The center part shows a "Chat Screen" which enables the player to communicate with his/her friends. By this tool, players in a group will be organized in a better way. They will help each other, they will give room numbers where objects or foods are available.

The right-most part shows a 2D map which shows the player position in the home.



5.2 Game Engine

The game engine will be implemented as a module. It will involve data about game progress and some member functions to update these data accordingly.

5.2.1. Server Game Engine

The game engine on the server side will provide synchronization among the clients. At the beginning of the game it will initialize the initial places of the characters, foods and treasure objects. Moreover if the amount of food decreases dramatically and if the players have problem of finding food, game engine will provide food to the environment. The balance of food and calorie is managed by game server.

5.2.2. Game Engine (Client side)

This engine will work synchronously with game engine that works on the server side. The game engine on the client side controls the game mechanism. This is responsible for the management of subcomponents such as graphics, network and audio engines. The puzzles will be provided by game engine on the client side according to difficulty levels.

5.3 Input Module

We will implement an input module that will handle with all the input data through the use of mouse, keyboard, and console. The mouse, keyboard, and console events will be captured by frame listener binded with OGRE window and necessary callback functions will be called.

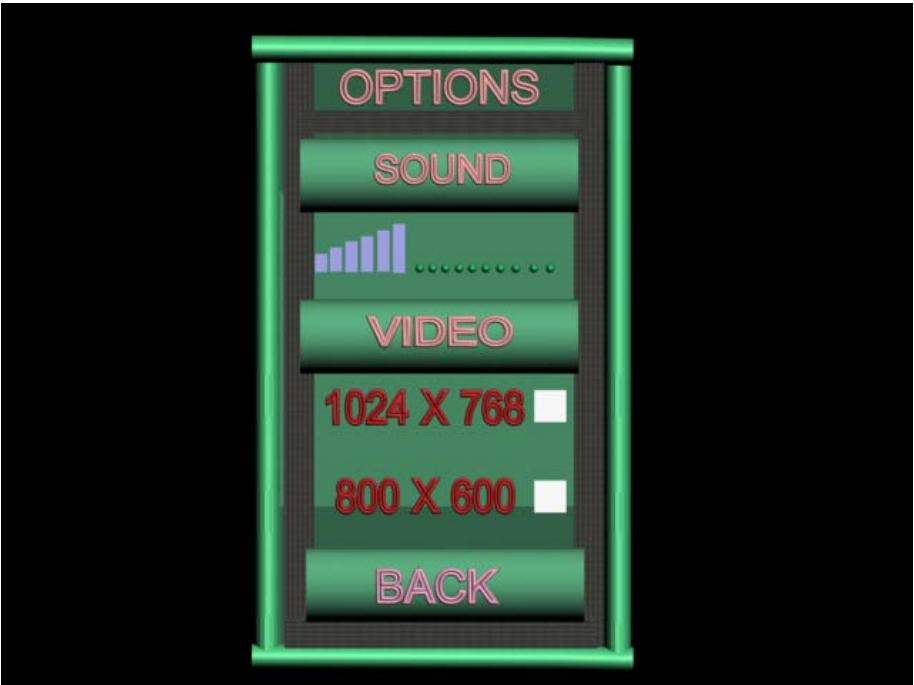
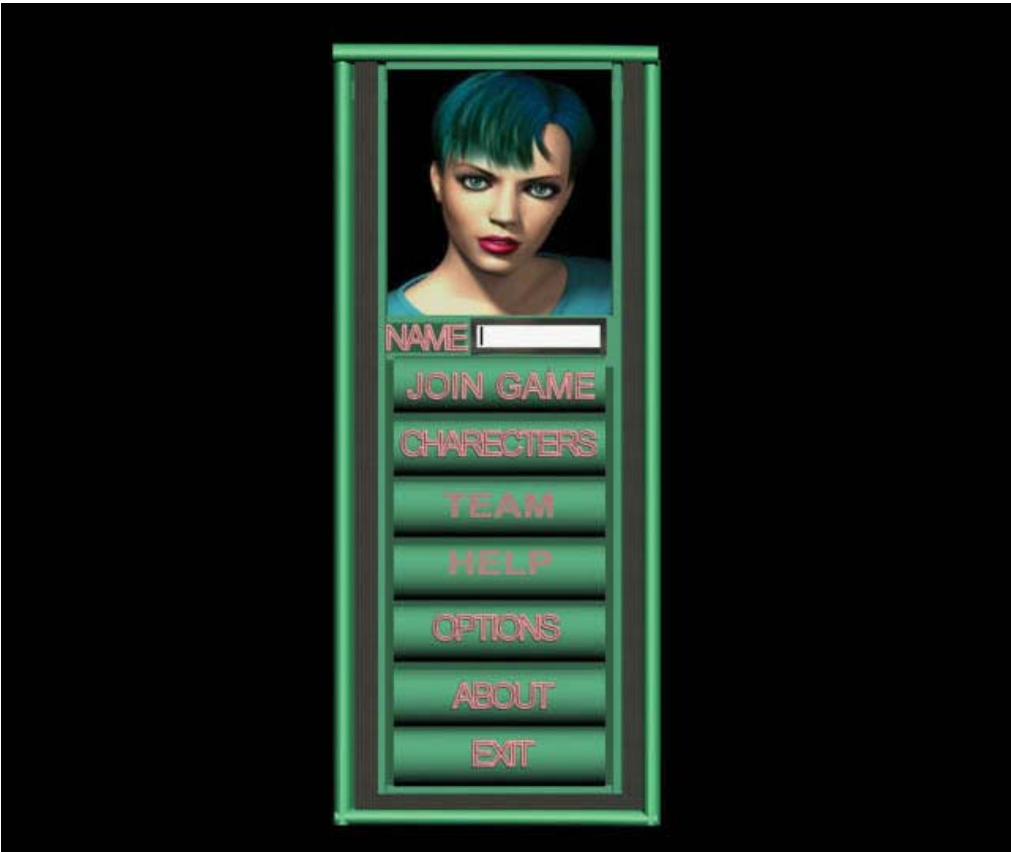
5.4 Menu Module

The game will include two menus that provide the user to interact with the game. The content of these menus is determined, of course, according to the general properties of game.




a) Game Main Menu

This menu involves items to start a game. The items in this menu are:

- **Set Name**
This item leads user to set name.
- **Join Game**
This item is used to register user for the game of next session. Server starts each session of game in a predefined time period. The player that wants to join the game has to wait current session to terminate.
- **Choose Character**
This item is to make the user select one of pre-defined characters in the game.
- **Team**
This item is to choose one of existing teams or set up a new team to join.
- **Options**
This item provides change of graphics, audio and keyboard controls.
- **Help**
This item provides a help manual for the game.
- **Exit**
This item is used to exit from the game.
- **About**
About us and game.



Choose Character

		
Character Name	Character Name	Character Name

Team

Nick Name

Join Team

Create New

<div>Team Name</div> <div>*Team Leader</div> <div>*Player Name</div>	<div>Team Name</div> <div>*Team Leader</div> <div>*Player Name</div> <div>*Player Name</div>	<div>Team Name</div> <div>*Team Leader</div>
<div>Team Name (Full)</div> <div>*Team Leader</div> <div>*Player Name</div> <div>*Player Name</div> <div>*Player Name</div>	<div>Team Name</div> <div>*Team Leader</div> <div>*Player Name</div> <div>*Player Name</div>	<div>Team Name</div> <div>*Team Leader</div>
<div>Team Name</div> <div>*Team Leader</div> <div>*Player Name</div>	<div>Team Name</div> <div>*Team Leader</div>	<div>Team Name</div> <div>*Team Leader</div>

b) In Game Menu

- **Objectives**

The last task assigned to that user is seen by this item.

- **Food Basket**

Food that stocked by team members.

- **Inventory**

Inventory that player had get so far.

- **Map**

A 2D map of the current storey is observed from top view by this menu item.

- **Chat**

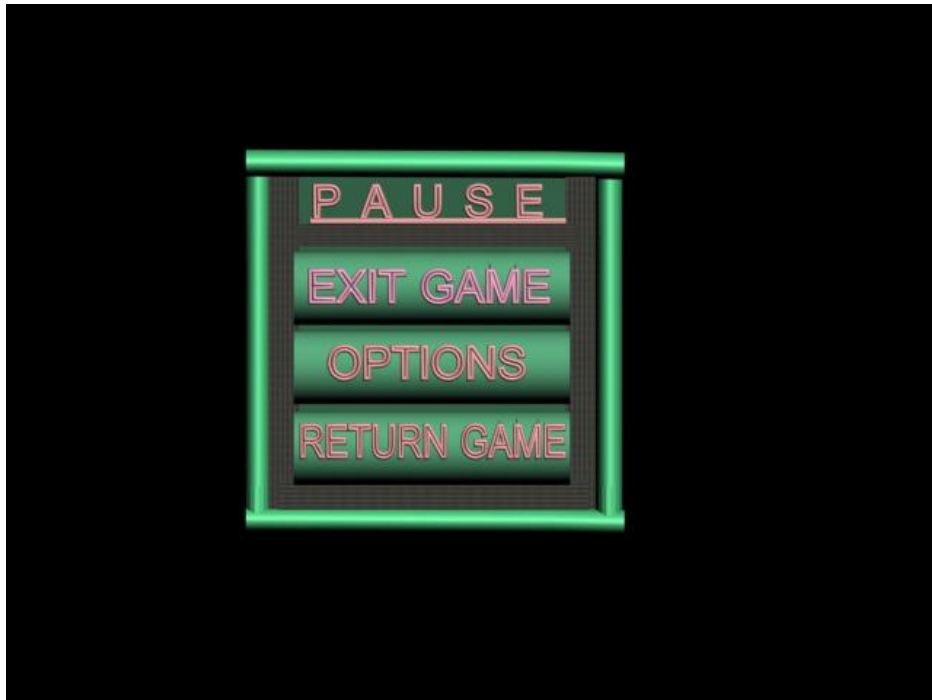
Player can chat with each other.



c) Game Pause Menu

By this menu only the player pauses.

- **Leave Game**
The user will quit from the current game session.
- **Options**
This item provides change of graphics, audio.
- **Return To Game**
This item provides to resume game.



5.5 Artificial Intelligence Engine Module

Being a MMOG, one should be able to play the game although there's no one online besides him/her. This requires implementation of some AI. For this purpose, our game involves Virtual Players (a.k.a. VP). These players, controlled by an AI engine, has the same attributes and methods as the real players. However, they are not controlled by human players. So that, if an action needs to be taken, action is not triggered (i.e. call to the related [VP prefixed] member function) by a user input, but from the AI engine according to the current situation & progress of the character, and state of the environment. Namely, if the energy level of the character is under some predefined value, then finding food has more priority. If not, then the VP does not spend time and energy to take food, but tries to find the treasure as early as possible (i.e. finding treasure has more priority). And throughout this process, the VP does not go into a loop or does not visit the same room multiple times for the same purpose.

5.6 Graphic Engine Module

Graphic Engine is the one of the most important modules of our project. Graphic Engine will handle all of the rendering operations during the game according to user input from player input and feedbacks from game engine.

We will use OGRE 3D (Object-Oriented Graphics Rendering Engine) as our graphic engine. OGRE (Object-Oriented Graphics Rendering Engine) is a scene-oriented, flexible 3D engine written in C++ designed to make it easier and more intuitive for developers to produce applications utilizing hardware-accelerated 3D graphics. The class library abstracts all the details of using the underlying system libraries like Direct3D and OpenGL and provides an interface based on world objects and other intuitive classes. . Because our game project will be a highly object oriented project and integration of OGRE 3D is easier relatively we decided to use this engine for the rendering issues of the scene.

5.7 Network Module

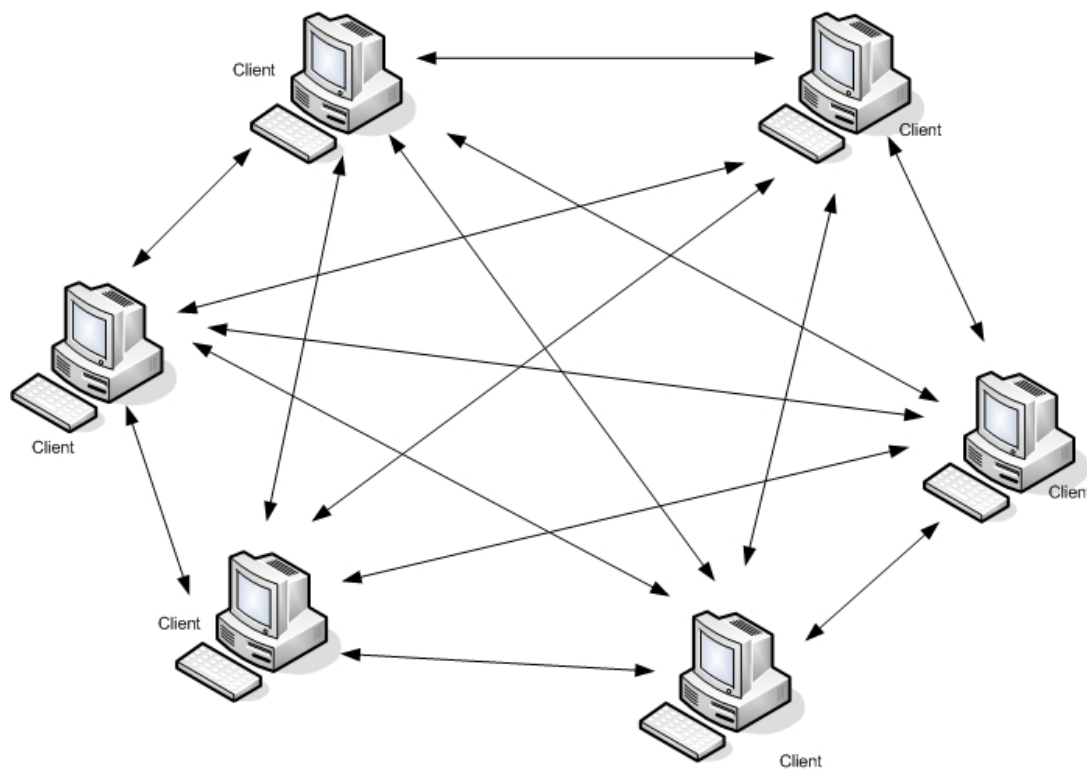
Treasure hunt is a massively multiplayer online game connecting players through the internet .It is a real time application with the interaction of server and clients.Since it is a real time process, we have to handle some problems because of some external constraints such as connection rate and speed.

Designing the network module and handling problems is one of the most important work for Treasure Hunt.

We will run game engine on server part on the other hand graphic engine, audio engine, physics engine on the client parts.Also we think that we will provide a second computer as a server to run the ai engine.This will be done for efficiency constraints.Server will have a database schema keeping the building_id and room_id of each player and the step the team is in.

Server will send the player_id of other players to a player if they are in the same room. If one enters a room the player_id's of the other players who were in that room is send by main server to this player and the player_id of the new player is send to the existing players.The inverse logic applies to the case when someone leaves the room.

In the same room situation the data transfer is independent of main server.Only the position data transfer is in client-client manner but the other type of data transfer will involve main server.This is illustrated in the figure below.



Same Room Condition

A time period of 50 ms will be enough for the position data transfer in the same room. We have estimated that if we send the position in every 50 ms accurately there will be no lag in the game. And if a player takes a treasure or food object the object_id of this object will send to main server. And main server will send the object_id immediately to the remaining clients as the object is now passive.

We will have one main server and many clients. Server will be the fastest computer with the best connection and other computers will be clients. Although there are many ways of encoding packets, they are all transmitted as either UDP or TCP packets. TCP packets are very good for transferring data, but TCP packets are not so good for games. Because of the TCP protocol, TCP packages are often delayed (resulting in games with a lot of lag) and arrive as streams rather than packets. As a result we have to implement our scheme to separate the data. On the other hand UDP packets are very good because they are sent right away and sent in packets. Therefore data can easily be distinguished. Since efficiency is another constraint it is also good to use UDP protocol. Since TCP has a handshake protocol.

However UDP have some disadvantages and we have to deal with problems below:

UDP packets are not guaranteed to arrive. All the packets that we send, some fraction or possibly none of the packets could be get. For example a player could pick the treasure but this information would be lost. As a result, player could not see the treasure object of the next step, although he has picked the object of previous step.

UDP packets are not guaranteed to arrive with the same order it has been sent. It would be a huge problem for our game. Because it is an important concept in our game who does before.

UDP packets have no protection from hackers but it is not a important task to handle in this step.

UDP transport does not provide flow control or aggregation so it is possible to overrun the recipient and to send data inefficiently.

Another problem according to the connection rate and speed is explained in this scenario: A player gets a food object and the object_id will be send to server. After that the server will send all the clients that the food has taken. So food object will dissappear. But what can be done if another player gets the same food with same object_id during this period (client to server and server to cients).

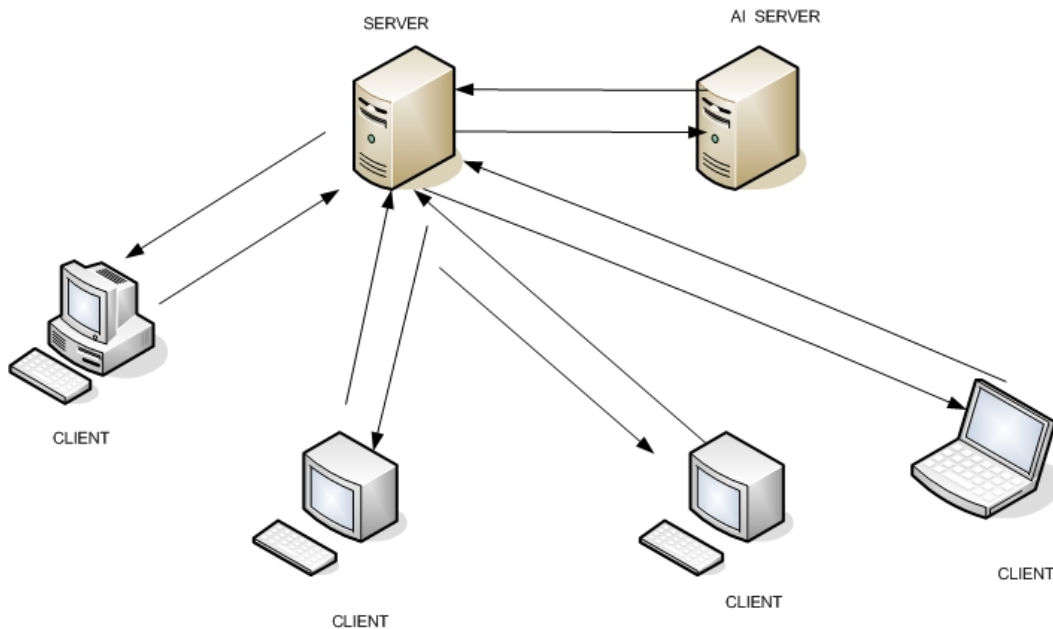
Moreover another problem is that if the second players' information comes before the first.

We have to define an error management system. Assuming the internet is not reliable, we have to handle connection problems rather than block, lock-up, or crash.

We have to define a resend package method for the packages that does not arrive.

We have to define order and sequence method that the packages arrive out of order.

We have to define a flow control and aggregation method.

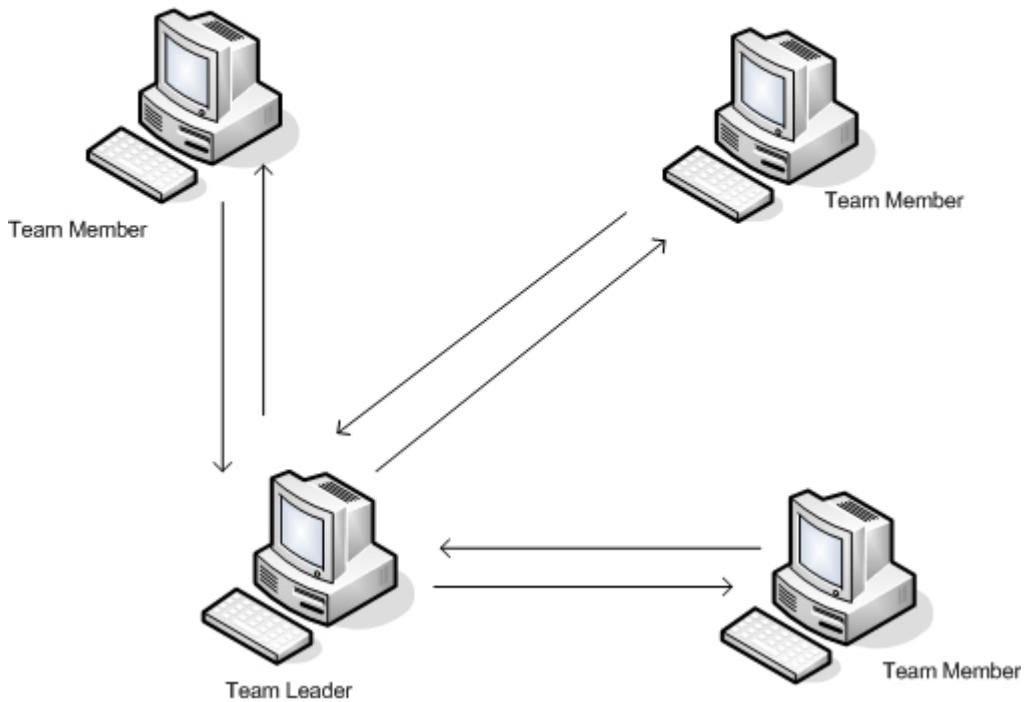


TEAMING-UP in NETWORK MODULE

There will be teams of at most 4 players as mentioned in the scenario and one of them will be the leader. In order to achieve efficiency in data transfer the team leader will act as a server in chat component. Through chat the communication data of team does not send to main server, but to the team leader. Team leader broadcasts this data to all team members.

Since the treasure object is assigned to the team, the puzzle is sent to all members of that team when one of them found the object. But the only one who is capable of sending the answer to the main server is the team leader. The answer is sent by team server (team leader) according to the discussion in team chat.

TEAM COMMUNICATION SCHEMA



Data Architecture

The data dictionary consists of the definitions of game data flowing between server-client, and client-client.

Main Server-Client

Initialization flow

- If.1.** Client sends the team number he prefers.
- If.2.** Server will assign position of each player and send this to related client game engine.
- If.3.** Server will broadcast the object_id's and positions of each treasure and food object to every client.
- If.4.** Server will send the team leader's player_id to other members of the same team.
- If.5.** Server will send the team_id of each player to himself and player_ids of other members of the team.
- If.6.** Client will send the building_id and room_id of itself to main server.(which are calculated from position values.)These values are both zero when player is outside of the buildings.
- If.7.** Server will inform the players who are in the same room by sending the player_ids of other players in same room to each one.

In-Game Flow

Changing a room:

lg.1. Client will inform the server by sending the building_id and room_id of the new room.

lg.2. Server will inform the players in previous room by sending the player_id of the player who exited the room with exit information.

lg.3. Server will inform the players in the new room by sending the player_id of the player with enter information.

Picking an object:

lg.4. Client will send the object_id of the object to server.

lg.5. Server will broadcast the object_id to all clients.

Game State Change:

lg.6. If the answer to the puzzle is right, new step number is sent to server due to state change.

In-Team Communication (Chat)

lg.7. Client will send the message to client which is team leader.

lg.8. Team leader client will broadcast the message to all team members.

Initialization Flow			
Package Code	Package Content	Content Type	Raw Data Size
If.1.	team number	1*int	2 Bytes
If.2.	position	3*float+1*int	14 Bytes
If.3.	Object_id, position	1*int+3*float+1*int	16 Bytes
If.4	Player_id	1*int	2 Bytes
If.5.	team_id, player_id(s)	1*int+# of_team_members*int	2-10 Bytes
If.6.	room_id,building_id	1*int+1*int	4 Bytes
If.7.	player_id(s)	# of_players_in_room*int	n*2 Bytes

In-Game Flow			
Package Code	Package Content	Content Type	Raw Data Size
Ig.1.	room_id,building_id	1*int+1*int	4 Bytes
Ig.2.	player_id	1*int	2 Bytes
Ig.3.	player_id	1*int	2 Bytes
Ig.4.	object_id	1*int	2 Bytes
Ig.5.	object_id	1*int	2 Bytes
Ig.6.	step_id	1*int	2 Bytes
Ig.7.	message	n*char	n bytes
Ig.8.	message	n*char	n bytes

The data size of the packages are mentioned as “Raw Data Size”, because they will be sent after an encoding process to reduce the package size. Huffman Encoding will be used to encode the messages in in-team chat.Reducing packet size will be applied for the sake of efficiency.

5.8 Audio Module

Game will have different soundtracks during game and waiting for the next session. The soundtrack will change between the steps. Sound effects will be used during games. Sound options will be controlled and volume level is adjusted by user.

We will use FMOD as our sound library. Fmod provide us all our needs. The most important feature of FMOD sound library for us is that using minimal resources and being scalable. We will use sound library in two place in our game. One for environment sounds like footsteps. And the other one for playing game musics. FMOD also provide us to play multiple sounds simultaneously, so that we can play some game music and environment sounds at the same time.

The Audio Module will hold the path of the selected sound file and load the file to memory when desired. The Audio Module will call the required functions from the fmod library. Player can input the properites of audio module like play/stop music, change volume of music from the user interface. After initialization in the beginning of the program, it starts playing when program call ‘play’ and stops when we call ‘stop’ function.

5.9 Chat Module

Chat module will enable players conversing with other players,in real time.Players will be able to post his/her message in a chat window and watch others post their messages on the game screen. During game interaction between players make some lively dialogue,and a much more enjoyable game. Moreover user can send some sound effects.When they find a treasure object he can send a sound effect. A sound effect can be a laugh,haw-haw,"you have no chance", "go home", "I am the hunter", "Yeah I have found".

6. CLASS DEFINITIONS

6.1 BaseObject Class

- **Attributes:**

It is the base class that treasure object, food object and furniture object that inherit from.

object_name: It is a unique string value to each object which identify the object. We will have treasure object, food object and furniture object and all of them will have a unique name such as key, book, laptop or cake, apple, bread or table, chair and computer.

object_id: Objects which have the same name will uniquely identify which object it is.

object_position: It is a pointer to position object.

angular_direction: It is a float number that provides the direction of object according to the angular constraints.

model: the name of model object that will be used while importing objects. (such as model.3ds)

texture: the name of texture file that will be used while importing object.

- **Methods:**

void init_model(): Initializes all Object3DS instances with its respective *.3DS file from an external resource using void initialize() method. It also initializes position and scaling factor of object using void set_scalar() and void set_position() method.

void initialize(): Initializes Object3DS instances from a source of *.3DS file using model and texture attribute.

void set_position(): Sets the position of objects according to the angular direction using rotate and translate methods.

void render(): The top-level drawing function. This function makes all calls necessary to create the world.

void disappear(): This function makes picked food and treasure objects invisible. Also if the player is at the first step of treasure hunt he/she cannot see the treasure objects related to remaining steps.

void appear(): This function makes treasure objects visible. If the player passes the first step of treasure hunt he/she can see the treasure objects related to next step.

void set_scalar(): This function is used to scale the model from its natural size.

void translate(): This function will be used to translate 3D objects.

void rotate(): This function will be used to rotate 3D objects.

6.2 Position Class

- **Attributes:**

x_coordinate : It is a float value indicating the x coordinate of the object in xy plane.

y_coordinate : It is a float value indicating the y coordinate of the object in xy plane.

z_coordinate : It is a float value indicating the relative z coordinate of contact point of the object.

floor_info : It is an integer value indicating at which floor the object is.

- **Methods**

Get_Position() : gets position attributes.

Change_Position() : changes position attributes.

6.3 Treasure Object Class

- **Attributes:**

Object_name, object_id, object_position, angular_direction, model and texture attribute will be inherited from base object class.

is_active: If the player takes the treasure object the object will lose brightness (inactivated) so other players cannot take this object.

Books, Keys, Phone, Notebook, Cup, Money, Pass Card, T-shirt, Pants, Shirt, Socks, Tea-pot, Life Preserver, Tape Recorder.

- **Methods:**

void init_model():

void initialize():

void set_position():

void render():

void disappear():

void appear():

void set_scalar():

void translate():

void rotate():

Methods above will be used from the base class by inheriting.

void activate_deactivate(): If the player takes a step related item it will be deactivated for all other but the next steps' item will be all activated.

6.4 Food Object Class

- **Attributes:**

is_active: If the player picks the food object the object will lose brightness (inactivated) so other players cannot take this object.

calorie: It is the integer value that shows the calorie value of the food object.

is_animal: It is the boolean value that shows the food is pertaining to an animal. Vegetarian players could not eat this type of food.

Cake: Calorie =200 Is_animal=0	Coke: Calorie=80 Is_animal=0
Bread: Calorie=100 Is_animal=0	Ice-Cream: Calorie=100 Is_animal=0
Cheese: Calorie=50 Is_animal=0	Hamburger: Calorie=100 Is_animal=1
Banana: Calorie=50 Is_animal=0	Fish: Calorie=200 Is_animal=1
Chocolate: Calorie=150 Is_animal=0	Buttock: Calorie=300 Is_animal=1
Watermelon: Calorie=30 Is_animal=0	Chicken: Calorie=200 Is_animal=1
Apple: Calorie=20 Is_animal=0	

- **Methods:**

void init_model():

void initialize()

void set_position():

void render():

void disappear():

void appear():

void set_scalar():

void translate():

void rotate():

Above methods will be used from the base class by inheriting.

void deactivate(): If the player takes a food item it will be deactivated for all others.

void eat(): If the character does not have any allergic problem to the founded food object the calorie of the food will be added to energy of character. If it has it would not add.

6.5 Furniture Object Class

Table, Chair, Wardrobe, Personal Computer, Refrigerator, Sofa, Blackboard, Flower, Garbage.

- **Methods:**

void init_model():

void initialize():

void set_position():

void render():

void disappear():

void appear():

void set_scalar():

void translate():

void rotate():

These methods will be used from the base class by inheriting.

6.6 Character Class

- **Attributes**

name : It is a string unique to each character which is given by player or if it is virtual player this name is given by game engine.

type : It is an integer to indicate type of character, which are predefined by us.

status : It is an integer indicating the status of the character as 'Stand', 'Slow Walk', 'Walk', 'Run'

energy : It is an integer to determine the speed of character. Throughout the game, this value will decrease from its initial value, and can be increased by eating food found in rooms. If this value is under a threshold value, character can only do 'Slow Walk'.

position : It is a pointer to Position object.

crouch: It is an integer indicating whether the character is crouching or not. So that in raycasting the camera position is lowered.

inventory : it is an array of object_names that have been taken.

direction: Its is float to determine the angle between character's current direction and x axis.

allergic_foods : it is array of object_names that cannot be eaten by character.

is_vegetarian : it is an integer indicating whether the character is vegetarian or not. If so animal foods can not be eaten by character.

is_virtual : It is an integer indicating whether the character is a virtual player or not. If so, VP_Base_Function is called throughout the game to control the virtual player.

- **Methods:**

Move(): This function gets an input from player and according to that input changes the position and status of character, if possible (i.e. There is not any object in front of the

character). To change the position it uses some information of the character. To determine the direction of move it uses the **direction** of the character. The current **energy** of character and input from player is used to determine the speed.

Rotate(): This function gets an input from player and according to that input changes the direction of character.

Pick_Treasure() : This function provides taking a treasure object. If called, this function removes the object from the map and add that object to the inventory of the player.

Pick_Food() : This function provides taking a food object. If called and the food is not in allergic_foods, this function removes the food object from the map, and increases the energy of the player according to energy value of that food.

Decrease_Energy() : This function is called in main loop according to time value of game. This function is called every 10 seconds, and it decreases the **energy** by an amount of 30 calories.

6.7 Virtual Player Class

This is a class inherited from Character Class.

- **Attributes:-**

The ones inherited from character class.

- **Methods**

VP_Base_Function() : This is the function to control the virtual player(VP) throughout game, and according to the situation & energy of the VP, it decides to call helper functions (either VP_Find_Food(), or VP_Find_Treasure()).

VP_Find_Food() : This function is called by VP_Base_Function() when the energy level is under a predefined threshold level. When called, required Move(), and Pick_Food() functions are called by this function till finding some food to gain energy.

VP_Find_Treasure() : This function is called by VP_Base_Function() when the energy level is over a predefined threshold level. When called, required Move(), and Pick_Food() functions are called by this function till VP finds the trasure object or the energy level of VP falls under the treshhold.

6.8 Map Classes

6.8.1 POINT

- **Attributes**

position : It is a pointer to **POSITION** object.

- **Methods**

Get_Position() : get position of point.

6.8.2 WINDOW

- **Attributes**

window_id : It is an interger unique to each window.

first_point : It is a pointer to **POINT** object.

second_point : It is a pointer to **POINT** object.

width : it is a float indicating width of window.

heigth : it is a float indicating heigth of window.

heigth_from_ground : it is a float indicating heigth of window from ground.

- **Methods**

Draw_Window() : draw the window

6.8.3 DOOR

- **Attributes**

door_id : It is an interger unique to each door.

first_point : It is a pointer to **POINT** object.

second_point : It is a pointer to **POINT** object.

width : it is a float indicating width of window.

height : it is a float indicating height of window.

is_open : it is a bool indicating whether open or not.

- **Methods**

Draw_Door() : draw the window.

Open_Door() : open the door.

Is_It_Open() : is this door open or not

6.8.4 ROOM

- **Attributes**

room_id : It is an integer unique to each room.

corners : It is an array of room floor polygon's corner **POINTS**.

doors : It is an array of **DOORS** of room.

windows : It is an array of **WINDOW**s of room.

objects : It is an array of **OBJECT**s of room.

- **Methods:**

Draw_Room() : Draw this door.

6.9 Network Class

Server will have a database schema keeping the building_id and room_id of each character and the step_id of each team.

- **Attributes**

player_id: it is an integer which is unique to all user.

team_id: it is an integer which denotes the team number that user belongs

step_id: it is an integer denoting the step number of the user's team

ip_no: it is an integer number which keeps the related ip number of all other objects.

Position *object_position;

character_position : It is a pointer to position object.It takes the position of a character.

taken_tre_object_id[]: It is an integer array of object_ids of the taken treasure objects.

taken_food_object_id[]: It is an integer array of object_ids of the taken food objects.

- **Methods (server)**

send_data(): Sends the related data to clients.It has a resend method package for the data doe not arrive to clients.

listen_port():Listens the same port for all clients.

take_data(): Takes and handles the data sent by client.

Server will send the position of other players to a player if they are in the same room.And if a player takes a treasure or food object the object_id of this object will send to server.And server will send the object_id to the remaining clients to tell the objects are now passive.

- **Attributes (client)**

struct clientdataschema

```
{  
    player_id  
    ip_no  
    character_position  
}
```

- **Methods (client)**

send_server_data(): Sends the related data to server. It has a resend method package for the data does not arrive to server.

send_client_data(): Sends the related data to clients which are in the same room as this client is in.It has a resend method package for the data does not arrive to recepients.

take_server_data(): Takes and handles the data sent by server.

take_client_data(): Takes and handles the data sent by other clients in the same room.

order_packages(): Orders that the packages arrive out of order

Character_Encoding(): Encodes characters By Huffman Coding

Character_Decoding(): Decodes Huffman Encoded characters.

6.10 Audio Class

- **Attributes**

audio_id : it is unique integer indicating the id of audio data.

volume : it is float indicating volume of sound.

- **Methods:**

Play_Audio() : This function play the audio which's id is **audio_id**.

Stop_Audio() : This function stop the audio which's id is **audio_id**.

Set_Volume() : This function set the **volume** of audio.

Get_Volume() : This function get the **volume** of audio.

Get_Audio_File() : This function get the audio which's id is **audio_id**.

6.11 GameEngine Class

This is a class handling the game progress.

- **Attributes:**

position *current_position: this is a pointer to a position type object, holding the current position of the character.

int objective: holds the object_id of the object, that is assigned to character to find.

- **Methods:**

Update_Environment_Info(): When called, this function updates the environment information.

Update_Objective(): This function is called when an objective is finished (i.e. assigned object is found). If the objective is not the last one(i.e. the found object is not the treasure), the **objective** attribute is updated according to the new objective.

6.12 Input Class

- **Attributes:**

chat_message : it is a string value that the player writes on the chat console.

mouse_input : it is a position object that specify the place that mouse clicked

keyboard_input : it is a key from keyboard that have a functionality in our game. Most probably the keys that have functionality in our game will be Enter, Up, Down, Left, Right, Space and etc.

- **Methods**

void Get_Message_From_Mouse() : Get mouse input

void Get_Message_From_Keyboard(): Get keyboard input

void Get_Message_From_Console(): Get the message written to the chat console

void Send_Message_From_Mouse() : Send mouse input to the appropriate classes

void Send_Message_From_Keyboard(): Send keyboard input to the appropriate classes

void Send_Message_From_Console(): Send the message written to the chat console

6.13 AIEngine Class

VIRTUAL PLAYER

This is a class inherited from Character Class.

Attributes: -

Methods:

VP_Base_Function() : This is the function to control the virtual player(VP) throughout game, and according to the situation & **energy** of the VP, it decides to call helper functions (either **VP_Find_Food()**, or **VP_Find_Treasure()**).

VP_Find_Food() : This function is called by **VP_Base_Function()** when the energy level is under a predefined threshold level. When called, required **Move()**, and **Pick_Food()** functions are called by this function till finding some food to gain energy.

VP_Find_Treasure() : This function is called by **VP_Base_Function()** when the energy level is over a predefined threshold level. When called, required **Move()**, and **Pick_Food()** functions are called by this function till VP finds the trasure object or the energy level of VP falls under the treshold.

4.14 GraphicsEngine Class

- **Attributes: -**

- **Methods:**

void Draw_map(): It will render the parts of the building that are visible to the player.

void Draw_furnitures(): It will render the the furnitures visible to the player.

void Draw_objects(): It will render the objects looked for by the player.

void Draw_treasure(): It will render the Treasure, if it is visible by the player.

void Draw_characters(): It will render the characters visible to the player.

void Draw_food(): It will render the foods visible to the player.

void Draw_All(): It will call the methods above in the correct order.

4.15 Chat Class

Since chat is implemented to be used to communicate inside the teams, and the team leader acts as a server in this scheme (additionally anyone has the chance of being leader); the methods are implemented in two ways as server/teamleader and client/team member.

- **Attributes:**

static Audio *audios; This audio will contain the audio files played when the user send a sound effect.

static int sound_id; This integer will be unique to audio files

- **Methods (client / team member)**

send_team_data() : This function will send the message to the team leader to be broadcasted to team members .

take_team_data() : This function will receive messages from team leader.

send_sound_effect(): This function will send the sound_id of the audio object to team leader for broadcasting.

play_sound_effect(): This function will play the audio of the related sound_id.

- **Methods (server / team leader)**

send_data() : This function will broadcast the data received from team members or input from himself (team leader).

take_data(): This function will be inherited from network class.

take_sound_effect():This function will send the sound_id of the object to all clients.

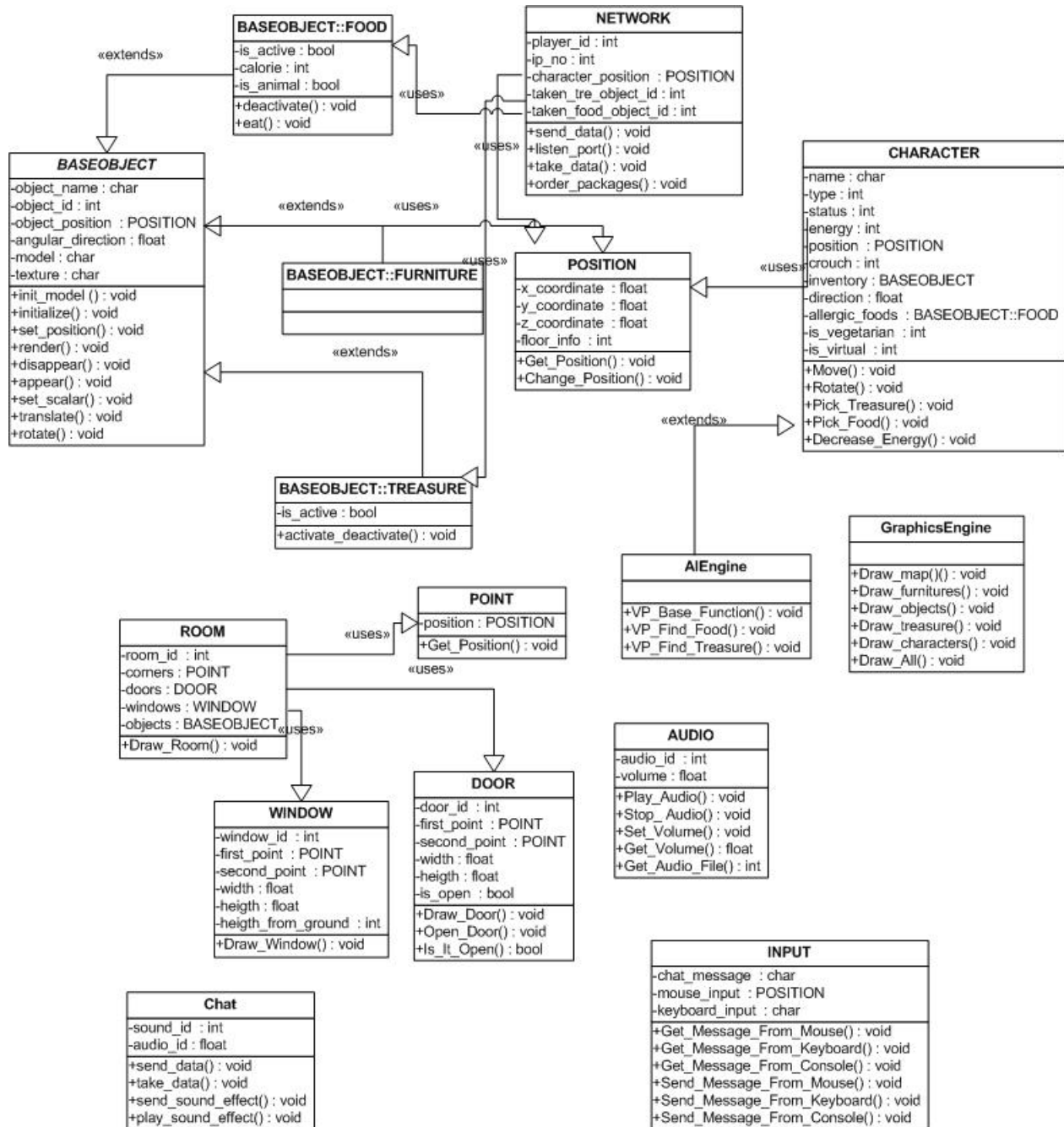
send_sound_effect():This function will send the sound_id of the object to all team members.

7. TOOLS

Throughout the implementation of our game, needs will arise by means of different aspects. After some research and analyze we decided on following issues.

- Our game will run on **Microsoft Windows XP** operating system. The reason is its wide usage and compatibility with other tools which will be used throughout the project.
- We will develop our game on **Microsoft .NET** platform.
- In terms of graphics, we will use **C++** for implementation and **OpenGL** as graphics library. The reason is being experienced about C++ programming language and OpenGL as a team.
- We will also use **OGRE** (Object-Oriented Graphics Rendering Engine). Being a flexible open-source engine and implemented using C++, makes it a considerably helpful tool for our project.
- Handling images and textures raises a need to use a tool. For this purpose, we will be using **Adobe PhotoShop**, being one of the most powerful & easy-to-use tools, and experience of team members.
- Another need arises by means of handling 3D modelling issue. **3D Studio Max** will be used for this purpose

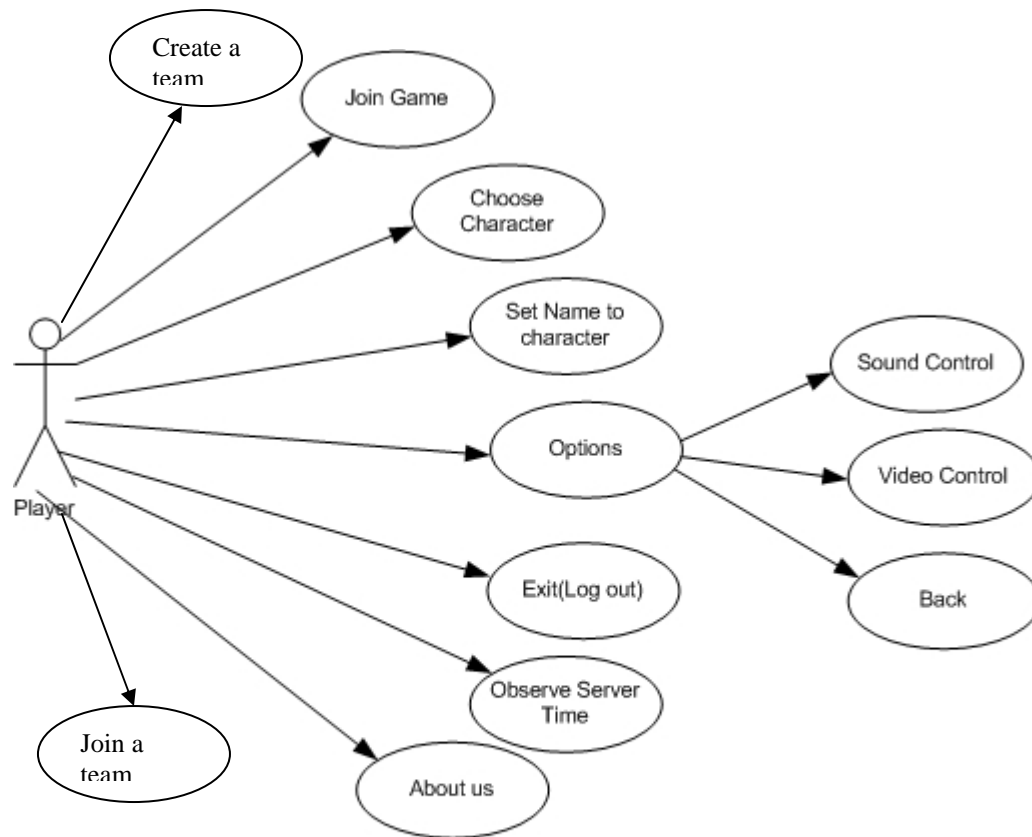
8. CLASS DIAGRAM



9. USE CASE DIAGRAM

a)Game Main Menu Use Case:

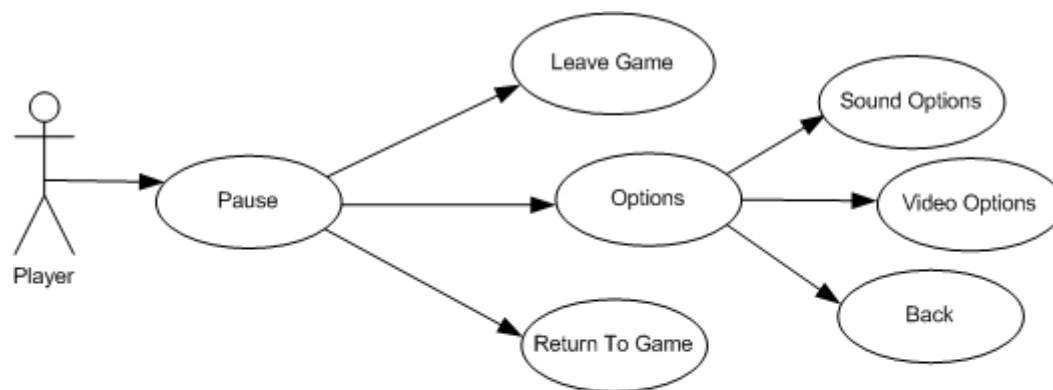
These are the use cases that user will face while joining the game.



GAME MENU

b) Game Pause Menu

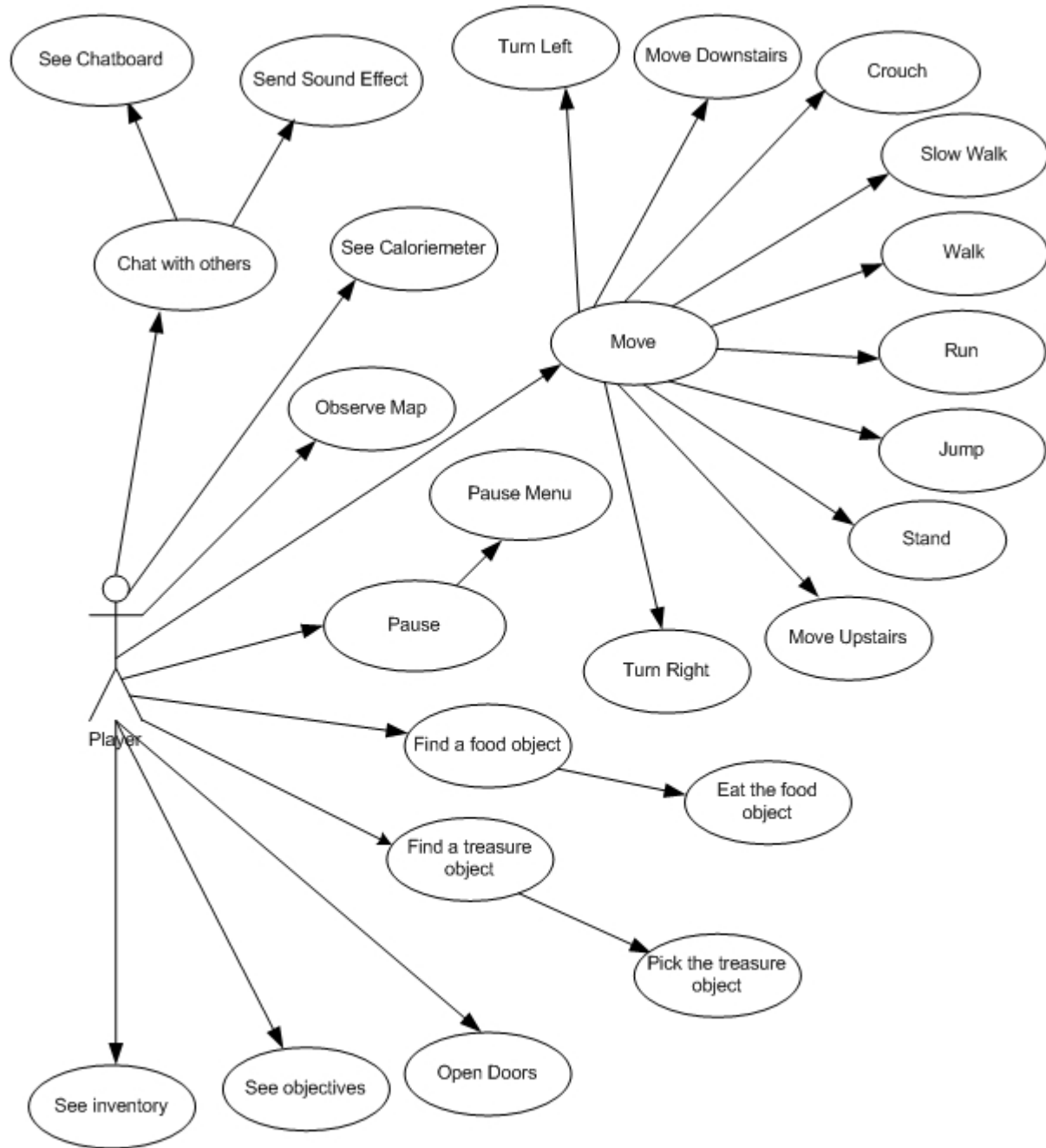
These are the cases that user can do while pausing the game.



PAUSE MENU

c) In Game Menu Use Cases

These are the use cases that user will face while playing the game.

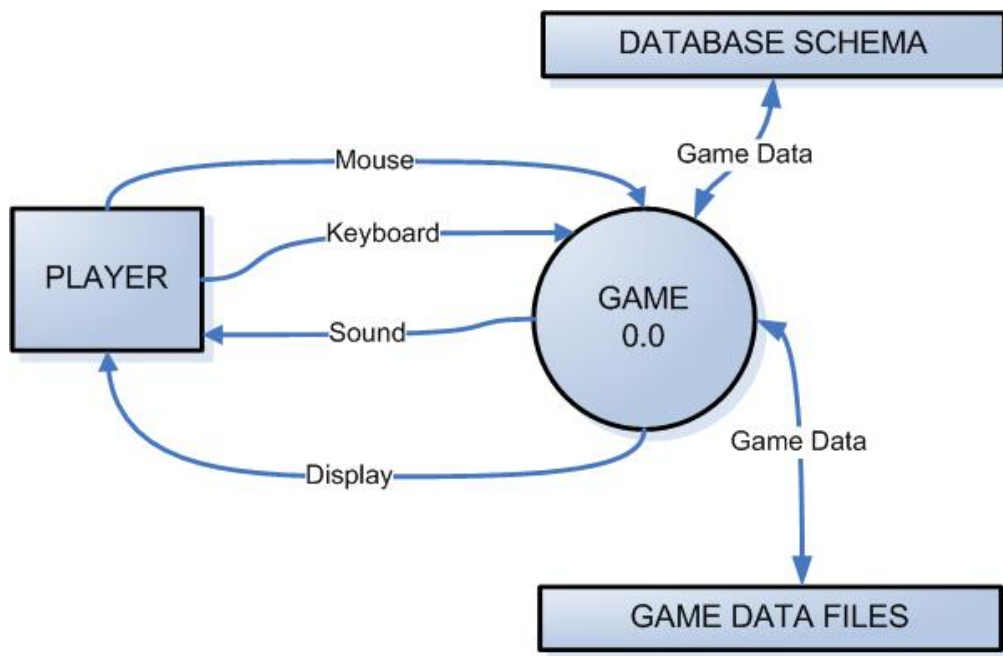


IN GAME MENU

10. DFD

10.1 Data Flow Diagram Level 0

Level 0 DFD in Figure 1 of 9.1 is the overall system. It shows the interaction between player and game, it also shows the interaction between game and resources.



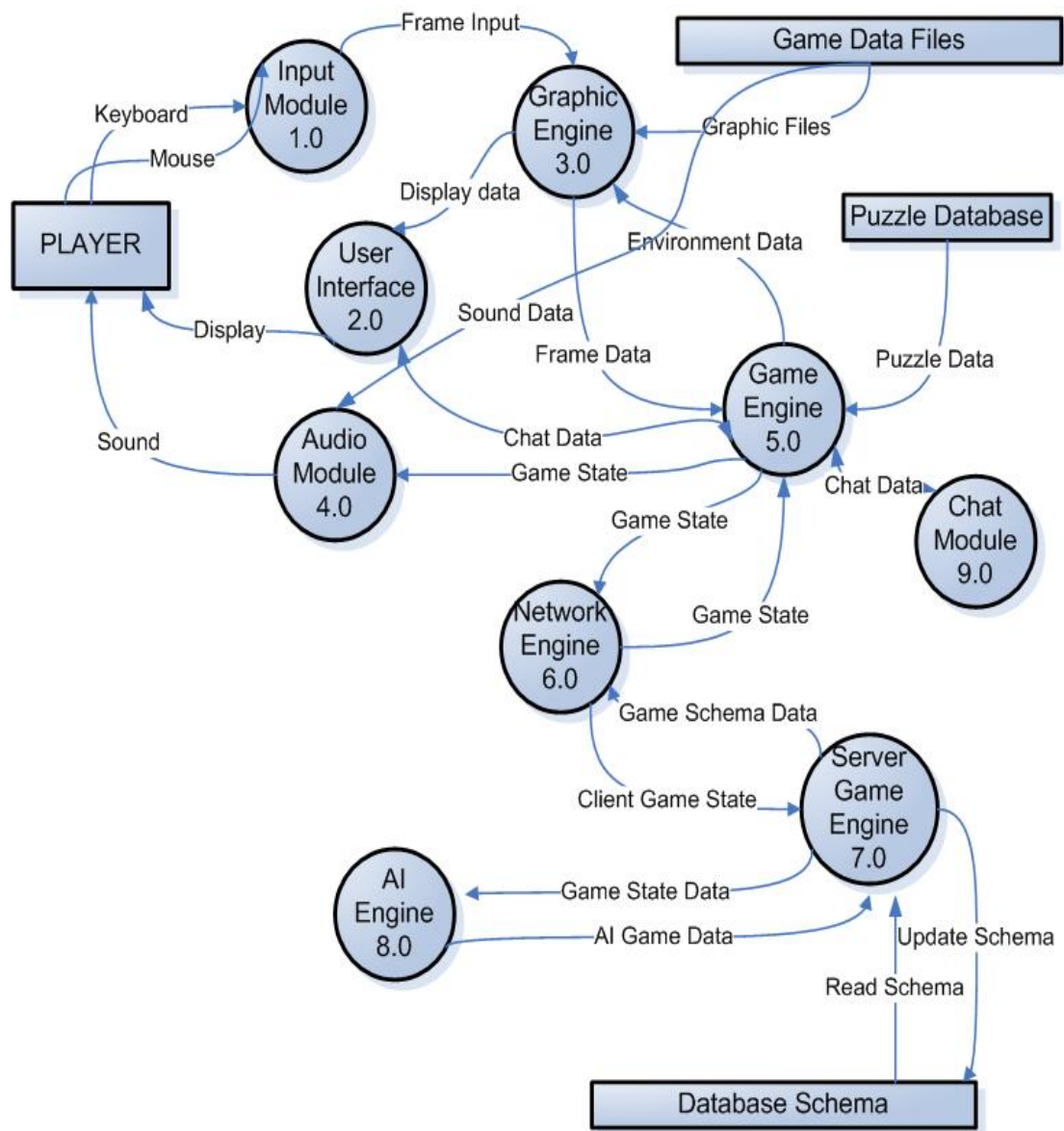
DFD LEVEL 0

Figure 1 of 8.1

10.2 Data Flow Diagram Level 1

Figure 1 of 8.2 is more detailed look on overall system. Innerstructures and sub-modules are described in a detailed way. Player will provide inputs using keyboard and mouse and take graphics and audio outputs. Game client sends scene data to graphics engine and graphic engines sends graphics outputs to the game client. Game client sends game state to audio and player take audio input.

Network component provides data transfer between game client and game server in Figure 1 of 8.2. Game server sends game data to AI client and takes AI Game data.



DFD LEVEL 1

Figure 1 of 8.2

10.3 Data Flow Diagram Level 2

Game Engine

Detail look on Game Engine in Figure 1 of 8.3

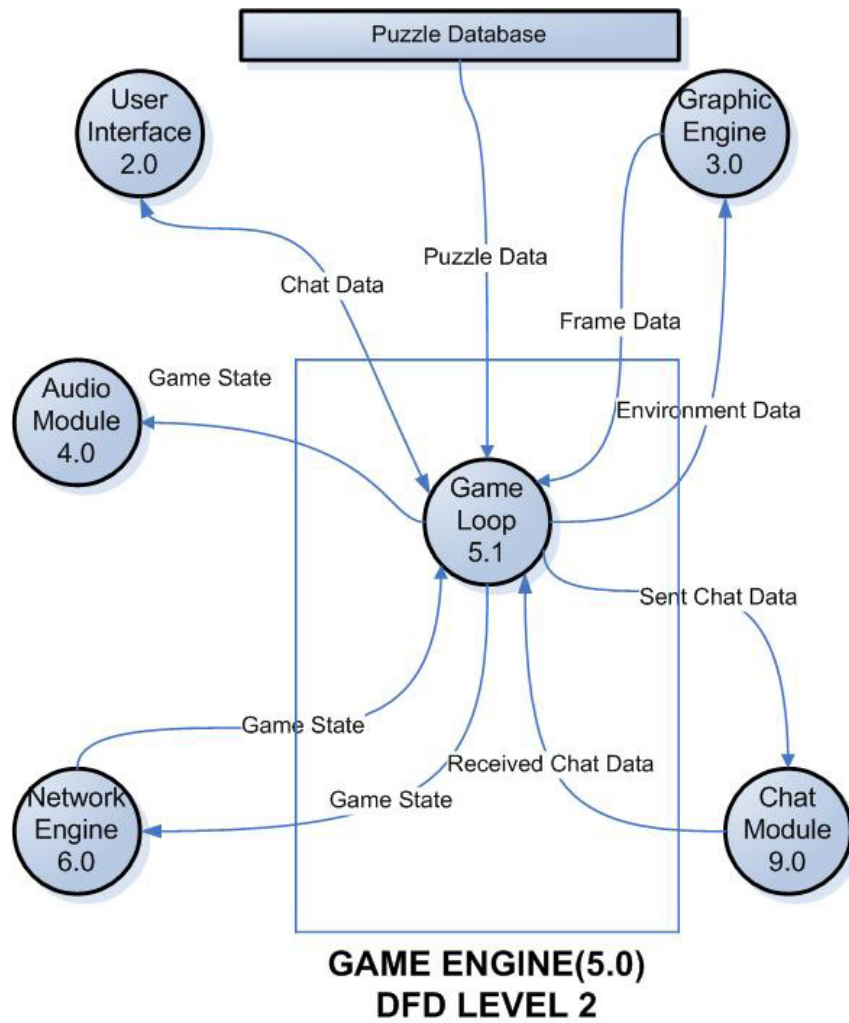


Figure 1 of 8.3

Network Module

Game Data is transferred from clients to game server, game server to clients by network engine and with use of a database schema in Figure 2 of 8.3

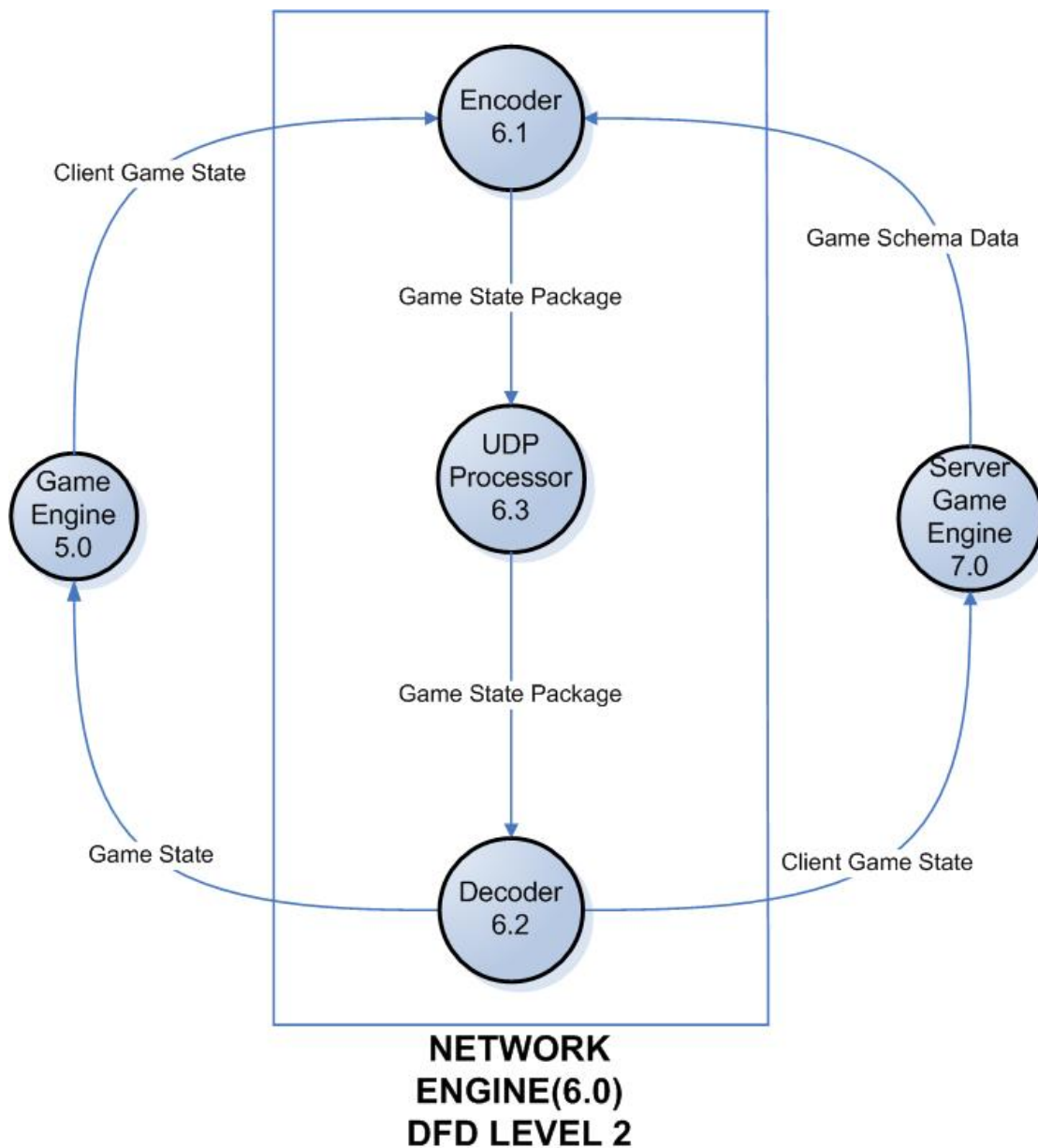
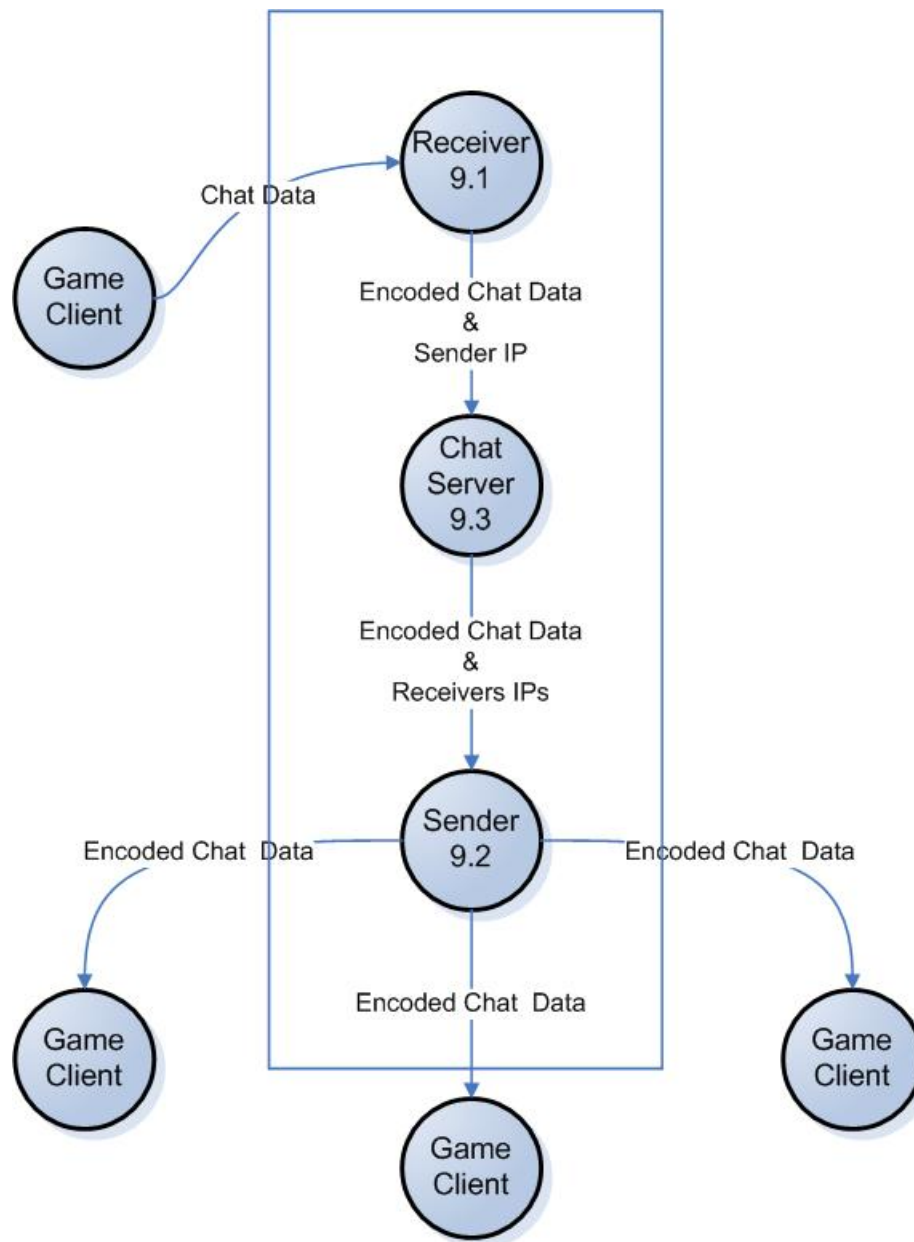


Figure 2 of 8.3

Chat Module

Chat Data is transferred from clients to clients by chat module in real time. Figure 3 of 8.3



CHAT MODULE (9.0)
DFD LEVEL 2

Figure 3 of 8.3

Audio Module

The Audio Module will hold the path of the selected sound file and load the file to memory when desired in Figure 4 of 8.3

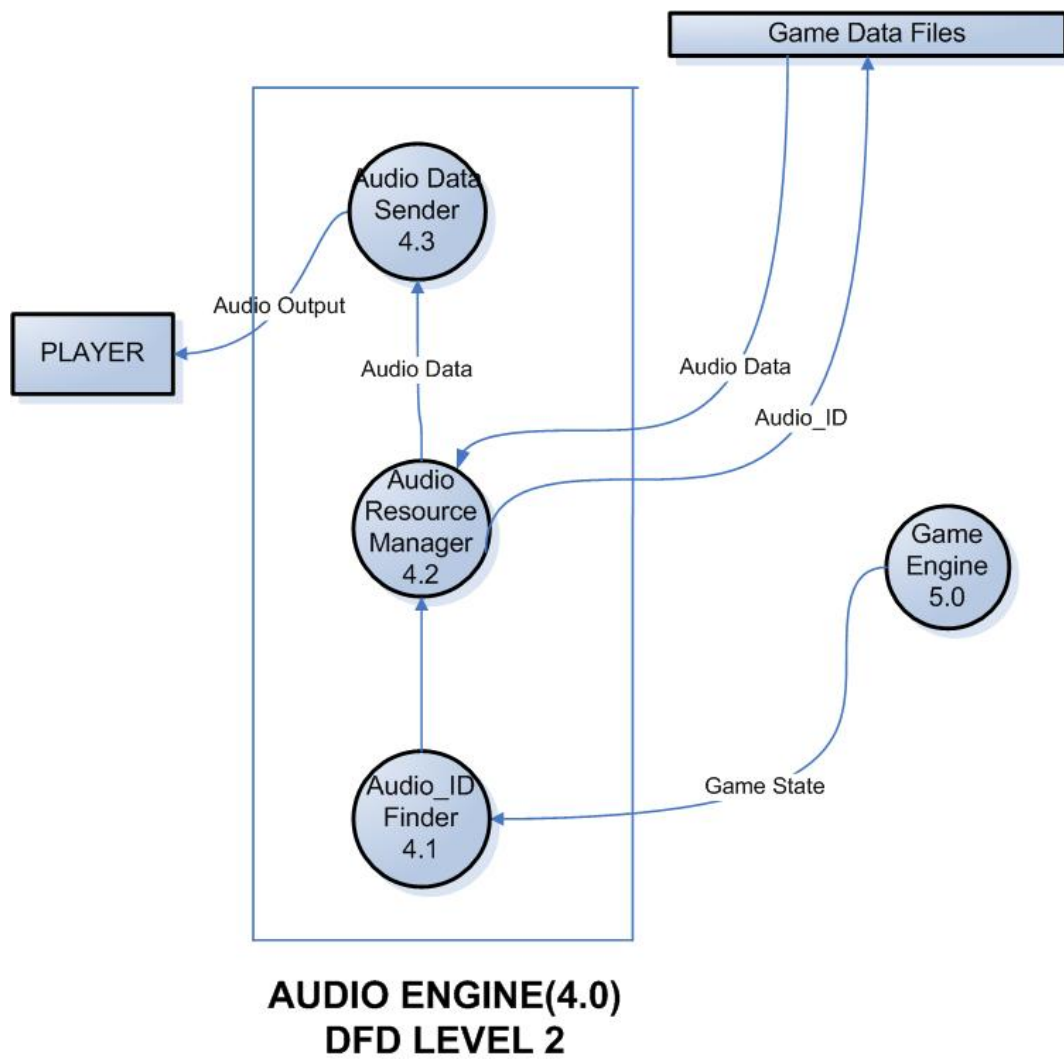
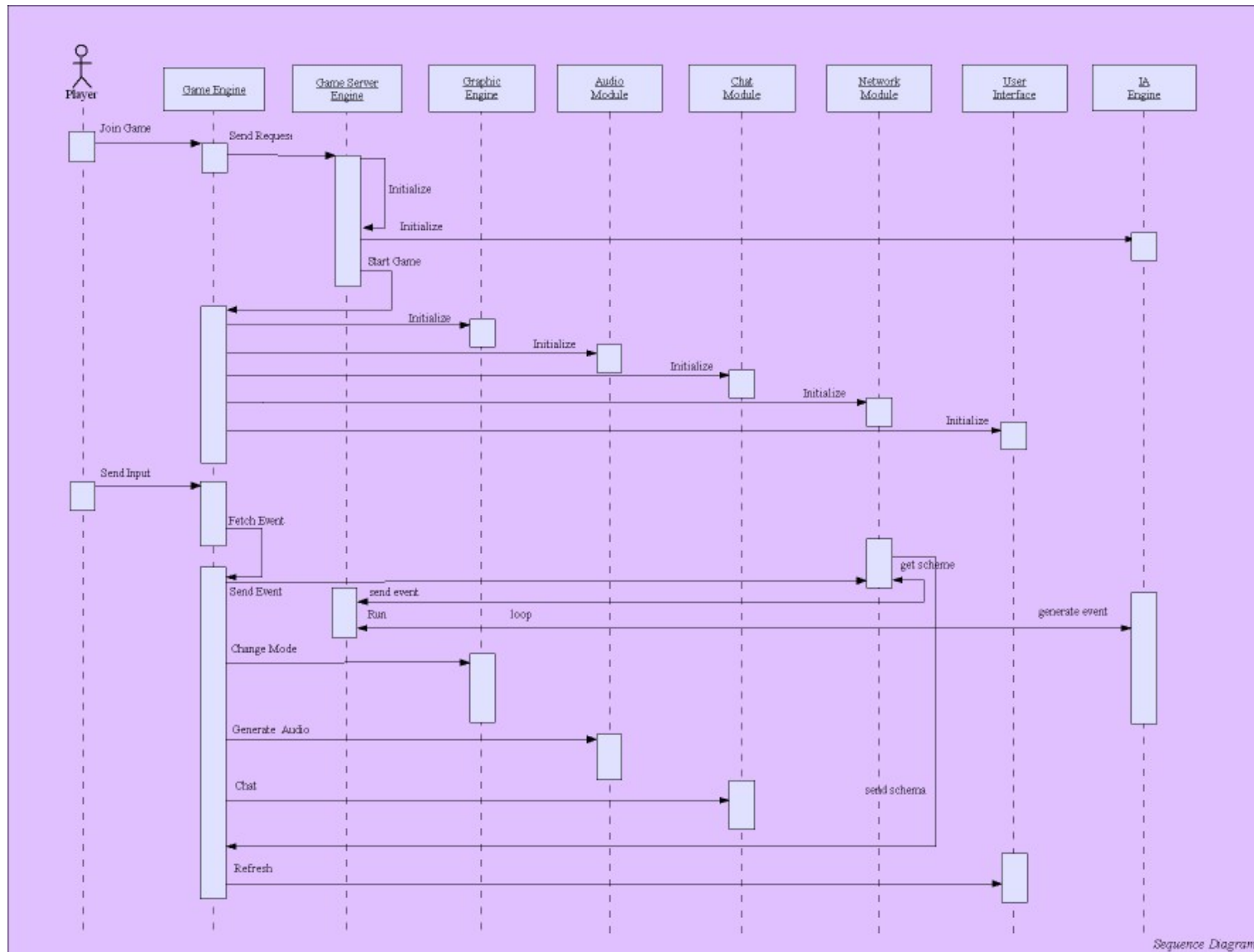


Figure 4 of 8.3

11. SEQUENTIAL DIAGRAM



12. PROJECT SCHEDULE

12.1. Project Task Set and Workpackages

In this part we will define tasks and categorize each task in an appropriate workpackage. We will not include the tasks which were accomplished so far, instead we will include the tasks that have to be accomplished.

Workpackage 1 : Management of the project

This workpackage includes tasks relevant to project management. It includes the following tasks:

- DELIVERABLE: Detailed design report
- Web Page Preparation
- DELIVERABLE: Configuration management and development plan
- DELIVERABLE: Test plan specification

Workpackage 2: Client-Side Game engine development

This workpackage includes tasks relevant to client-side game engine development and testing of the game. It includes the following tasks:

- Integrating animation of characters to the engine
- Client Side Game Engine Prototype Development
- DELIVERABLE: Prototype implementation
- Engine subsystems integration
- GUI development (CEGUI)
- Character engine development
- Render engine development
- Network engine development(client-side)
- Artificial Intelligence engine development
- Scripting
- Alpha Testing
- Beta testing & Debugging
- Help Menu design

Workpackage 3 : Server-Side Game engine development

This workpackage includes tasks relevant to server-side game engine development and testing of the game. It includes the following tasks:

- Server-Side Game Engine development
- Network engine development(server-side)
- Artificial Intelligence engine development
- Alpha Testing
- Beta testing & Debugging
- Help Menu design

Workpackage 4 : Game resource production

This package includes the tasks required for the development, gathering and integration of the game resource data that will be used by the engine. It includes the following tasks:

- Game resource gathering
- Modeling 3D objects
- Map Modelling
- Modeling characters
- Modeling Textures
- Animating characters
- Audio Production (sound effects, soundtrack)

Workpackage 5 : Finalization Package

This package includes tasks that gather and finalize the products of the game. It includes the following tasks:

- First development snapshots
- Game production initial version
- Game production final version

12.2. Gantt Chart

Task Name				Duration	Start	Finish
1	WP1:Management Of The Project			149 days	Mon 01.01.07	Tue 29.05.07
2	Project Management			149 days	Mon 01.01.07	Tue 29.05.07
3	DEL: Detailed design report			18 days	Mon 01.01.07	Thu 18.01.07
4	Web Page Preparation			8 days	Sat 10.02.07	Sat 17.02.07
5	?DEL: Configuration management and development plan			15 days	Wed 21.02.07	Wed 07.03.07
6	?DEL: Test plan specification			15 days	Thu 08.03.07	Thu 22.03.07
7	WP2: Client-Side Game Engine Development			141 days	Fri 05.01.07	Fri 25.05.07
8	Integrating Animation of Characters to The Engine			10 days	Fri 05.01.07	Sun 14.01.07
9	Client Side Game Engine Prototype Development			10 days	Wed 10.01.07	Fri 19.01.07
10	DEL:Prototype Implementation			10 days	Sun 14.01.07	Tue 23.01.07
11	Engine Subsystems Integration			80 days	Thu 25.01.07	Sat 14.04.07
12	GUI Development (CEGUI)			80 days	Thu 25.01.07	Sat 14.04.07
13	Character Engine Development			80 days	Thu 25.01.07	Sat 14.04.07
14	Render Engine Development			80 days	Thu 25.01.07	Sat 14.04.07
15	Network Engine Development(client-side)			80 days	Thu 25.01.07	Sat 14.04.07
16	Artificial Intelligence Engine Development			80 days	Thu 25.01.07	Sat 14.04.07
17	Scripting			80 days	Thu 25.01.07	Sat 14.04.07
18	Alpha Testing			10 days	Mon 16.04.07	Wed 25.04.07
19	Beta testing & Debugging			30 days	Thu 26.04.07	Fri 25.05.07
20	Help Menu Design			5 days	Tue 17.04.07	Sat 21.04.07
21	WP3 : Server-Side Game Engine Development			121 days	Thu 25.01.07	Fri 25.05.07
22	Server-Side Game Engine Development			30 days	Thu 25.01.07	Fri 23.02.07
23	Artificial Intelligence Engine Development			30 days	Thu 25.01.07	Fri 23.02.07
24	Network Engine Development(server-side)			80 days	Thu 25.01.07	Sat 14.04.07
25	Alpha Testing			10 days	Mon 16.04.07	Wed 25.04.07
26	Beta Testing & Debugging			30 days	Thu 26.04.07	Fri 25.05.07
27	Help Menu design			5 days	Tue 17.04.07	Sat 21.04.07
28	WP4 : Game Resource Production			106 days	Tue 09.01.07	Tue 24.04.07
29	Game resource Gathering			106 days	Tue 09.01.07	Tue 24.04.07
30	Modeling 3D Objects			70 days	Wed 14.02.07	Tue 24.04.07
31	Map Modelling			70 days	Wed 14.02.07	Tue 24.04.07
32	Modeling Characters			70 days	Wed 14.02.07	Tue 24.04.07
33	Modeling Textures			70 days	Wed 14.02.07	Tue 24.04.07
34	Animating Characters			70 days	Wed 14.02.07	Tue 24.04.07
35	Audio Production (sound effects, soundtrack)			70 days	Wed 14.02.07	Tue 24.04.07
36	WP5 : Finalization Package			86 days ?	Thu 01.03.07	Fri 25.05.07
37	First Development Snapshots			31 days?	Thu 01.03.07	Sat 31.03.07
38	Game Production Initial Version			55 days?	Thu 01.03.07	Tue 24.04.07
39	Game Production Final Version			31 days	Wed 25.04.07	Fri 25.05.07