Ceng 491
Initial Design Report

Massively Multiplayer Online Role Playing Game Project
Virtual Turkey

MECAC
Cinar Kilcioglu
Mert Degirmenci
Umit Cavus Buyuksahin

December 26, 2010
<table>
<thead>
<tr>
<th>6.2.6</th>
<th>Turkey Map Screen</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.7</td>
<td>Quest Details Screen</td>
<td>27</td>
</tr>
<tr>
<td>6.2.8</td>
<td>Trade Screen</td>
<td>28</td>
</tr>
<tr>
<td>6.2.9</td>
<td>Character Information Screen</td>
<td>28</td>
</tr>
<tr>
<td>6.2.10</td>
<td>Options Screen</td>
<td>29</td>
</tr>
<tr>
<td>6.2.11</td>
<td>Video Options Screen</td>
<td>30</td>
</tr>
<tr>
<td>6.2.12</td>
<td>Audio Options Screen</td>
<td>30</td>
</tr>
<tr>
<td>6.2.13</td>
<td>Controller Settings Screen</td>
<td>31</td>
</tr>
<tr>
<td>6.2.14</td>
<td>ESC Menu Screen</td>
<td>31</td>
</tr>
<tr>
<td>6.3</td>
<td>Screen Objects and Actions</td>
<td>32</td>
</tr>
</tbody>
</table>

7 Libraries and Tools 33

8 Time Planning 33

  8.1 Term 1 Gantt Chart | 34
  8.2 Term 2 Gantt Chart | 34

9 Conclusion 35
List of Figures

1 Virtual Turkey .................................................. 3
2 Trade System ................................................. 4
3 The Complete Data Model ..................................... 6
4 Quest Data Object ............................................. 7
5 NPC Data Object ............................................... 7
6 Character Data Object ........................................ 8
7 Vehicle Data Object .......................................... 9
8 Account Data Object ........................................... 9
9 Treasure Data Object .......................................... 10
10 Entity Relationship Diagram ................................. 11
11 Game Architecture ............................................ 13
12 Component Diagram .......................................... 14
13 Register message .............................................. 15
14 Login message .................................................. 15
15 Logout message ............................................... 16
16 Chat message ................................................. 16
17 Update message ............................................... 16
18 Treasure hide message ....................................... 17
19 Treasure dig message ...................................... 17
20 Quest message ............................................... 17
21 Trade message ............................................... 18
22 Sequence diagram for client network component .......... 18
23 Sequence diagram for server network component ......... 20
24 Sequence diagram for physics component .................. 22
25 Sequence diagram for management component ............ 23
26 Connection Screen ............................................ 25
27 Account Screen ............................................... 25
28 Create Account Screen ..................................... 26
29 Login Screen .................................................. 26
30 Display Screen ............................................... 27
31 Turkey Map .................................................... 27
32 Quest Details Screen ....................................... 28
33 Trade Screen .................................................. 28
34 Character Information Screen ............................... 29
35 Options Screen ................................................................. 29
36 Video Options Screen .......................................................... 30
37 Audio Options Screen ............................................................ 30
38 Controller Settings Screen ....................................................... 31
39 ESC Menu Screen ................................................................. 31
40 Game Screen Flowchart .......................................................... 32
41 Term 1 Gantt Chart ................................................................. 34
42 Term 2 Gantt Chart ................................................................. 34
1 Introduction

This document contains the initial design descriptions of “Virtual Turkey” which is a mas-
sively multiplayer online role playing game (MMORPG). The approach used in this specifi-
cation is adapted from IEEE recommended practices [1]. MECAC, the project group, assumes
full responsibility of the requirements presented in this document.

1.1 Problem Definition

Turkey has variety of places with cultural inheritance to visit. In order to bolster touristic
interest in Turkey, we will develop an online multiplayer game which will provide culture
enthusiasts around the world the opportunity to preview sightseeing places in Turkey. To
meet the requirements of an MMORPG, the game being developed is expected to support
thousands of players concurrently. The project will later be published under GNU license
when it reaches the maturity of other well-known open source MMORPGs.

1.2 Purpose

This initial design report intends to provide complete design descriptions of “Virtual Turkey”.
The descriptions suggested in this document will serve as a guideline throughout the devel-
opment process of this project. The end-product will be tested against the requirements to
ensure the quality of the software produced.

1.3 Scope

This document contains a complete description of the initial design of “Virtual Turkey”.
This framework covers essentials of the MMORPG project. Its main intention is to provide
software design description of the system according to Software Requirements Specifications.

1.4 Overview

The report contains seven sections. First section introduces the project “Virtual Turkey”.
In the second section, system overview is explained. The details of the project is started
to be given in section 3. This section explains the design considerations. Data design and
system architecture are described in section 4 and 5, respectively. User interface of the game
is presented in section 6. Finally, libraries and tools, planning of the project are presented
and the report is concluded.
1.5 Definitions and Abbreviations

DoS  Denial of Service  
GNU  GNU is not Unix  
GUI  Graphical User Interface  
IP  Internet Protocol  
MMORPG  Massively Multiplayer Online Role-Playing Game  
MPI  Message Passing Interface  
NPC  Non-Player Character  
RSA  Rivest, Shamir and Adleman  
SDD  Software Design Descriptions  
SRS  Software Requirements Specifications

1.6 References


2 System Overview

The MMORPG project, Virtual Turkey, mainly serves the purpose of introducing the worth seeing monuments all around Turkey. The map of Virtual Turkey with missions is given in Figure 1. The player will be traveling the country in order to collect coins and the gold scattered over some secret places. While visiting and learning about the invaluable historical places of Turkey to carry out the quests, the player will have more chance to gain treasure; however, the treasure has to be hidden due to non-transportable nature of it, which makes the game more mysterious and riveting. Each player is planned to have several attributes to make the game-play more realistic. Player needs these attributes in order to use quest system, trade system, and chat system.

Quest System description includes the format of the quests, quest preliminaries and quest completion conditions. As a preliminary condition, the player will be asked to have a specific item in his backpack or an amount of gold owned. These two options are the only planned conditions for the time being. In addition, players will have a limit on how many quests they can take; thus, if this limit is reached, the player is automatically rejected. To finish a quest, the player will have to go to a certain place, pick up a certain treasure or item. To add a
new dynamic to the game, the place of this goal will not be given to the player, instead, a riddle with an answer which tells the place will be told to the player.

Trade system will consist of item description, trade conditions, and actual trade system. An item is any object that a player can own. According to this description, items will be tradable and be of some value. There will be usable items and non-usable items. Usable items are items such that there will be in the backpack waiting to be used by the player. On the other hand, non-usable items are items that will be traded or have a value but will not be usable by the player such as diamond. Moreover, there will be no restrictions on trade between players. Sample trade system is given in Figure 2.

The chat system is simple and user-friendly. A review of the current state is done based on game-concept and it is decided that the chat module will stay the same.

3 Design Considerations

3.1 Design Assumptions, Dependencies and Constraints

Security: The communication between the server and the client component of MMORPG should be encrypted with RSA algorithm. Server should also check the cheating case with offloading major physics calculations to server-side. In order to prevent abuse of the server component, software on the server of MMORPG should block the recurring requests from the same IP address. Server should log common proxy addresses as well to check if the client
is performing DoS attack.

**Software System Attributes:**

1. **Maintainability:** The modification of the source code should be disabled. The extensions should be applicable directly without modification to the back end. The patching service should be regarded as a different component in case extensions would be offloaded to a different server.

2. **Portability:** Both the client and the server component of the software should be portable. The target hardware platform for client component is unknown. Only requirement for the client software to be deployed is the operating system that belongs to Windows NT family. The server side of MMORPG should be independent of specific hardware or software configuration.

3. **Scalability:** Most important software system attribute of “Virtual Turkey” is its scalability. Deploying additional servers and dividing the persistent world should not require additional extension on the client software.

### 3.2 Design Goals and Guidelines

In order for “Virtual Turkey” to be massively multiplayer, the server should support 1000 users concurrently. The delay of communication between server and the user should not
exceed 0.1 second. The database transaction should not consume more than 10% of the time needed to process single request from client. The client software should enable the users to log-on the MMORPG within 5 seconds. The clients should be able to sign-in in less than 10 seconds.

In order for the MMORPG to be maintainable, the GUI of the server component should be robust for the administrator to update information or monitor the network traffic. The administrator should be able to log-on to the server within 5 seconds even when the server supports a thousand players online. When the administrator clicks “Manage Accounts” or “Manage NPCs” buttons, the GUI should show the requested information within 10 seconds and commit the changes within a minute. Monitoring network traffic should not increase the overhead on server-side as such information can directly be computed on-the-fly. When the administrator clicks monitor network traffic, the GUI on the server component should show the current traffic summary within 10 seconds.

4 Data Design

4.1 Data Description

The focus of the data model presented in this section is on Character and NPC data objects. Account, Vehicle, and Treasure data objects are associated with Character object. NPC object, on the other hand stores the information about the trade items of it. Non-playing characters has no associated vehicles or accounts as it can be observed from Figure 3. Following sections describes all the data objects and their major attributes in more detail.

**Quest** This data object represents quest given by non-player characters. Database needs to store the identifier of the quest which is only major attribute of it.

**NPC** The NPC data object - as its name suggests - represents a single non-player character in “Virtual Turkey” MMORPG. Besides its unique identifier, it has three major attributes namely position, owner, and type.

**Character** This data object represents the character of the user associated with the account. The user character may have associated Vehicle or Treasure data objects.

**Vehicle** This data object represents the vehicle in “Virtual Turkey”. Only characters can use the vehicles in the MMORPG to transport their character between different locations of the persistent world. Each vehicle can be associated with only one character.
Figure 3: The Complete Data Model
**Account**  This data object stores the account information of players. The account data object does not store the character information. However, it references a character data object. This approach enables players to have multiple characters in MMORPG.

**Treasure**  This data object represents the treasure items that the characters poses. The treasure data objects have associated value and position in the virtual world. This data objects are intended to be traded for items by characters.

### 4.2 Data Dictionary

This section describes the major system entities along with their types and descriptions. Attributes of each system entity have been listed.

#### Quest

![Quest Data Object](image)

**id:** The unique identifier of the quest data object. It is stored as integer.

#### NPC

![NPC Data Object](image)

**id:** The unique identifier of the NPC data object. It is stored as integer.
**owner:**  The owner character of the NPC data object. It is stored as integer.

![Character Data Object](image)

**Figure 6: Character Data Object**

**Character**  This data object represents the character of the user associated with the account. The user character may have associated Vehicle or Treasure data objects.

- **name:**  The unique identifier of the Character data object. It is stored as string.
- **account:**  References corresponding Account data object. It is stored as integer.
- **last played:**  The time that the character last played “Virtual Turkey”. It is stored as date object.
- **created on:**  The time that the character has been created by the user. It is stored as date object.
- **position:**  The location of the character in the persistent world of MMORPG. It is stored as three integers.
- **quests completed:**  The quests that the character has completed successfully. It is stored as list of integers pointing to quest objects completed.
- **quests in progress:**  The quests that the character has attempted to complete. It is stored as list of integers pointing to quest objects that are in progress.
- **is online:**  The boolean attribute showing whether the character is connected to persistent world or not. It is stored as a boolean variable.
- **vehicle:**  The current vehicle of the character that references a Vehicle data object. It is stored as integer pointing to corresponding vehicle system entity.
Vehicle Data Object

Vehicle This data object represents the vehicle in “Virtual Turkey”. Only characters can use the vehicles in the MMORPG to transport their character between different locations of the persistent world. Each vehicle can be associated with only one character.

id: The unique identifier of the Vehicle data object. It is stored as integer.

character: References the corresponding character object that owns the vehicle. It is stored as integer.

position: The current location of the Vehicle in the persistent world of MMORPG. It is stored as three integers.

Account Data Object

Account This data object stores the account information of players. The account data object does not store the character information. However, it references a character data object. This approach enables players to have multiple characters in MMORPG.
e-mail:  The email address of the account owner. It is stored as string.
password:  The password of the account owner. It is stored as encrypted string.
name:  The nick name chosen by the account owner. It is stored as string.
address:  The address of the account owner. This attribute is optional. It is stored as string.
last signed on:  The last date when the account holder has logged-in. It is stored as date object.
created on:  The date when the account was created. It is stored as date object.
online:  Boolean attribute showing if the account owner is connected. It is stored as boolean variable.
confirmation code:  The confirmation code for account owner. It is stored as encrypted string.
confirmed:  Boolean attribute showing whether the account was confirmed or not. It is stored as a boolean variable.

![Table of Treasure Data Object](image)

**Figure 9: Treasure Data Object**

*Treasure*  This data object represents the treasure items that the characters poses. The treasure data objects have associated value and position in the virtual world. This data objects are intended to be traded for items by characters.

- **id:**  The unique identifier of the treasure data object. It is stored as integer.
- **value:**  Value of the treasure data object. It is stored as integer.
- **position:**  The location of the treasure data object in the persistent world of MMORPG. It is stored as three integers.

### 4.2.1 Relationships

This section describes the relationship between the data objects described in the previous section.
NPC - Quest: Each NPC may be associated with one or more quests. Those quests are to be completed by characters. However, a quest may not be associated with multiple NPCs.

NPC - Item: Each non-playing character may own one or more items. However, an item can not be owned by multiple non-playing characters. NPCs are able to trade the items with playing characters.

Character - Item: Each character may own one or more items. However, an item can not be owned by multiple characters. Characters can trade the items and treasures with other characters or NPCs.

Character - NPC: Each non-playing character is owned by only one character. An NPC without an owner or a character without an NPC can exist.

Character - Vehicle: Each character can only own one vehicle in the MMORPG. A vehicle can not be associated with multiple characters.

Character - Account: Each character can be associated with at most one account. However an account owner may own multiple characters.

Character - Treasure: A character may collect zero or more treasure, however a treasure can only be owned by a single character.

![Entity Relationship Diagram](image)

Figure 10: Entity Relationship Diagram
5 System Architecture

5.1 Architectural Design

The game architecture has been shown in Figure 11. As seen in Figure ‘Virtual Turkey’ has two main parts, clients and server. The relationship between these two parts is accomplished by package transferring between each part’s own communication layer.

Game loop, which belongs to clients component, interacts with NPC part of the server component and during the interaction, information is passed to NPC through communication layers. Information can be gain gold, next position of character, vehicles etc. It also initiates the game to start.

Database inside the server component stores the information such as account information of the users, last position of the players, and their attributes. It also has a connection with server physics engine, which enables the physical system of the game. Following sections of this document decomposes overall architecture of the system to its components and their descriptions.
5.2 Description of Components

This section describes the major components of “Virtual Turkey”. All of the components are organized under two packages. As seen from Figure 12, the communication between the two packages is carried out through a TCP channel.

Figure 12: Component Diagram

5.2.1 Client Network Component

This component of the MMORPG describes how the clients communicates with the server through the network. The game play heavily depend on the quality of network connections. Elegant design is a must for “Virtual Turkey” to be massively multiplayer game. Following sections elaborates more on design of network component.

Processing narrative for client network component  Major responsibility of the network component is to handle message transmission between from the client to the server. For initial design, some of the possible messages are enumerated below. In the processing detail of this component, the meanings of the messages will be described.

```
enum ClientToServerMessageTypes
{
    Register, Login, Logout, Chat, Update, TreasureHide, TreasureDig, QuestRequest, QuitQuest,
    CompleteQuest, QuestApproval, QuestQuitOK, QuestDisApproval, QuestCompleteOK,
    QuestCompleteNotOK, TradeApproval, TradeDisApproval, TradeOK, TradeNotOK,
    TradeRequest, Trade
}
```
The input interface of the client network component interacts with the graphical user interface of the clients. The messages encoded according to network protocol is transmitted through network. The output interface of the client network component is a TCP channel ultimately reaching to server’s network interface.

This section describes the details of how client network component interacts with the client user interface and server network interface to achieve communication in between. The major functions of the component along with its attributes are described in the following sections.

**Register** This network interaction creates a new user account. A confirmation code will be send from server to users email address as a result of this transaction. The attributes of this interaction is as follows.

<table>
<thead>
<tr>
<th>Client to Server</th>
<th>Server to Client</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>Register</td>
<td>Username, email</td>
</tr>
<tr>
<td></td>
<td>password ,player</td>
</tr>
<tr>
<td></td>
<td>model</td>
</tr>
<tr>
<td></td>
<td>RegisterOK</td>
</tr>
<tr>
<td></td>
<td>Username, password</td>
</tr>
<tr>
<td></td>
<td>RegisterDenied</td>
</tr>
<tr>
<td></td>
<td>Username, password</td>
</tr>
</tbody>
</table>

Figure 13: Register message

**Login** This function will let a user to connect to the game server. The username and password will be given to the server and the server will check the name and password from the database. if the check returns true, the server will return to the client the client ID, else, username and fault will be returned.

<table>
<thead>
<tr>
<th>Client to Server</th>
<th>Server to Client</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>Login</td>
<td>Username, password</td>
</tr>
<tr>
<td></td>
<td>LoginOK</td>
</tr>
<tr>
<td></td>
<td>Client ID</td>
</tr>
<tr>
<td></td>
<td>LoginNotOK</td>
</tr>
<tr>
<td></td>
<td>Username</td>
</tr>
</tbody>
</table>

Figure 14: Login message
**Logout**  The client will send her/his player id and success or fault will be returned.

<table>
<thead>
<tr>
<th>Client to Server</th>
<th>Server to Client</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>Logout</td>
<td>Client ID</td>
</tr>
<tr>
<td></td>
<td>LogoutOK</td>
</tr>
<tr>
<td></td>
<td>LogoutNotOK</td>
</tr>
</tbody>
</table>

Figure 15: Logout message

**Chat**  Player sends her/his chat data, and it is forwarded to other clients.  

<table>
<thead>
<tr>
<th>Client to Server</th>
<th>Server to Client(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>Chat</td>
<td>ChatData</td>
</tr>
<tr>
<td></td>
<td>String ChatData</td>
</tr>
</tbody>
</table>

Figure 16: Chat message

**Update**  This network interaction is the main routine of the game. Once the GameTick() occurs in the clients, they will send their data to the server, and the server will forward their data to other clients according to their area of interest.

<table>
<thead>
<tr>
<th>Client to Server</th>
<th>Server to Clients (the clients which are in the area of interest of that client)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>Update</td>
<td>PlayerData</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 17: Update message

**Treasure Hide**  Once a player wants to hide her/his treasures(golds, money), the information will be send to the server and the server will process the action.

**Treasure Dig**  Once a player wants to take back her/his or another ones treasures(golds, money), the information will be send to the server and the server will process the action.

*Initial Design Report for Virtual Turkey*
The quests will be done only with an NPC. Once a player wants to be in a quest with an NPC, the related data will be forwarded. The NPC will check if the player meets the requirements or not, and the quest request is approved or disapproved. Client1 is the player and client2 is the NPC. Client to Server (and forwarded to the NPC) Server to client (The data is coming from NPC) Header Arguments Header Arguments

<table>
<thead>
<tr>
<th>Client to Server (and forwarded to the NPC)</th>
<th>Server to client (The data is coming from NPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td><strong>Header</strong></td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td><strong>Arguments</strong></td>
</tr>
<tr>
<td>QuestRequest</td>
<td>Quest ID, client1 ID, client2 ID</td>
</tr>
<tr>
<td>QuitQuest</td>
<td>Quest ID, client1 ID, client2 ID</td>
</tr>
<tr>
<td>CompleteQuest</td>
<td>Quest ID, client1 ID, client2 ID</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Trade  A player can make a trade with a NPC or another player. If the trade request is approved, the trade should start. Trade Client1 Client2 Itemlist means, client1 wants to buy Itemlist from Client2. If client1 has enough money, the trade returns success, else returns fault.
Dynamic behavior of client network component  The behavior of client network component has been exemplified in Figure 22.

5.2.2 Server Network Component

This component is responsible for updating the information of clients while balancing the network load on the server.

Processing narrative for server network component  There are a number of messages that are sent to client according to their requests. The types of the messages are given below. For actual interpretation of these messages, one can refer to client network component. More importantly, this section describes the prioritization in sending these messages. The band-width of the server computer has to be utilized as it is a limited budget. The details of this utilization is given in following sections of this document.
enum ServerToClientMessageTypes
{
  RegisterOK, RegisterDenied, LoginOK, LoginNotOK, LogoutOK,
  LogoutNotOk, Chat, Update, UpdateTreasures, TreasureHideOK, TreasureHideNotOK,
  TreasureDigOK, TreasureDigNotOK, QuestRequest, QuitQuest, CompleteQuest
  QuestApproval, QuestQuitOK, QuestDisApproval, QuestCompleteOK, QuestCompleteNo-
  tOK,
  TradeApproval, TradeDisApproval, TradeOK, TradeNotOK, TradeRequest, Trade
}

server network component interface description
The server network component interfaces with all client network components in a full-duplex manner. The transmission in between is a TCP channel over internet.

server network component processing detail
The most important attribute of server network component that affects the processing is its nxn state matrix where n is the number of players. This matrix holds the update information that has to be sent to each other. A formula that computes the interest area of each player determines the prioritization of each update message.

  For example, an update from a player far away reaches slowly compared to an update from a near-by player. This optimization is perhaps the most important factor enabling Virtual Turkey to be massively multiplayer.

  The terms and the weights of the priority formula is tentative as an optimal formula requires massive amount of experimentation. The processing of server network component is given below as a pseudo-algorithm.

  1. Gather update information from clients.
  2. Put the information into NxN matrix where N is the number of players and matrix[i][j] denotes the update that has to be sent from player i to player j.
  3. Compute the most urgent set of messages that fits the bandwidth of the server.
  4. Send the urgent messages and go-back to 1.

Alg.1 Pseudo-algorithm of server network component.
Dynamic behavior of server network component The behavior of server network component has been exemplified in Figure 23.

![Sequence diagram for server network component](image)

**Figure 23: Sequence diagram for server network component**

5.2.3 Physics Component

This component is responsible for physics calculation for the persistent world of MMORPG. The computation includes collision detection and all other sorts of physical interactions of the players. For the purpose of preventing cheating, major computation and checking of the physics engine will be performed on the server-side.

**Processing narrative for physics component** The physics component will be present in both clients and the server. Processing of both sides is similar except the fact that the end-results are different. The output of the physics component will be used for cheating checking on server-side while the same results - if not modified - will be used in graphics engine on client-side.

**physics component interface description** The physics component on client side interfaces with client graphical user interface. Each movement update of a player is sent to physics component and resulting computation is sent to the graphics engine. The result can be hit-back from a wall, or position change of running action. Every result is crosschecked with the server.

The interface of the physics component is little bit different although the processing is - and should be- exactly the same. The server-side physics engine receives the update information through the server network component and transmits the output to the client.
network component. The result of this calculation is mostly confirmation of the client physics engine results. In case of cheating, the physics component of the server sends log-out message to client network component.

**physics component processing detail**  There are two major functions of physics component. These are collision detection and position calculation. Description of the two functions are given below. The computation is handled by a JigLibx physics engine that is specifically developed for Microsoft XNA framework.

**Collision detection**: This function will calculate the distance between objects in the persistent world. A collision will be reported in case two objects closer than a threshold.

**Position calculation**: For each moving object the physics engine will compute the next position according to friction, velocity, and other environmental parameters in the game.

**Dynamic behavior of physics component**  The behavior of physics component has been exemplified in Figure 24.

### 5.2.4 Management Component

The management component will provide the utilities that system administrators need. It encapsulates several crucial processing units of the game such as data management, non-playing characters, and game loop. Object-oriented design of each subcomponent enables single and consistent interface to other components in the system.

**Processing narrative for management component**  This component resides on server-side. The major sub-components are quest system, virtual environment management, account management, and NPC handling. These functions are to be used by the administrator of MMORPG. The software requirement specification for the Virtual Turkey necessitates the management tools described in this section. Description of the processing details for each sub-component are described in the following sections of this document.
management component interface description The component interfaces with administrator via a graphical user interface. The input from Administrator will be handled directly within management component. The results of administrator actions will be transmitted to other components if necessary.

management component processing detail The processing functions for each subcomponent is given in this section. Following paragraphs describe the detailed description of the subcomponents.

**Quest Management** The quest system will be able to handle the main questing aims, such as a reward system and task completion systems. The quest management tool enables the administrator to add and remove quests from the GUI. This management involves the management component interact with the database to perform transactions.

**Virtual Environment Management** The administrator will use the management component to edit the virtual environment. Each change in the persistent world will be reflected to the clients using the network component of the server.
**Account management** Accounts of the players will be managed through user interface of the management component. Common tasks are deletion and insertion of new accounts.

**NPC handling** Each NPC will be handled by a thread running on server, but separate from the server program. Thus, communication with server and NPC threads should be implemented in order to provide the game NPC feature.

**Dynamic behavior of management component** The behavior of management component has been exemplified in Figure 25.

![Sequence diagram for management component](image)

Figure 25: Sequence diagram for management component

### 5.3 Design Rationale

The decomposition of the MMORPG to the components described above is agreed upon all group members. The most discussion was on distribution of physics component to the server and the client. The suggested component design prevents cheating. The trade-off was between cheating detection and server load. Opponents of this design emphasized the need for low-cost server. However, we have agreed on the fact that the persistent world MMORPGs have a lot of cheating players and there is a compelling need for cheating detection.

Another point where MECAC group members were separated is on server network architecture. Current design suggests a centralized server-client network architecture. This requires huge server cost for achieving massive number of players. However, peer to peer architectures are also considered while proposing this initial design document. Although such an MMORPG could be novel research, the we have decided to pursue the standard centralized design due to time limitations.
6 User Interface Design

6.1 Overview of User Interface

The interaction between the user and the game is done via user interface. First, the user enters server IP and his login information. If it is authenticated, the game starts. The most extensive interaction is occurred in this stage. There are many user interfaces connected to the main gameplay screen. The user reaches these interfaces and plays the game via keyboard controllers. The connected interfaces are explained in section 6.2. Main keyboard controllers are W, A, S, D keys. These keys enable the character - the user - to walk through the game. According to view angle of the camera, W key moves the character forward, S key moves it backward, and A and D keys moves left and right, respectively. Moreover, the character can move diagonally by hitting two keys at the same time. These keys also active when the character rides in a vehicle or an animal, which can be accomplished by hitting Enter key. Q and ESC keys enable the user to reach other interfaces. The character carries the gold with C key, and hides it with H key. When the user quits the game, interaction between the user and the game is over.

6.2 Screen Images

6.2.1 Connection Screen

The first step when the user runs the game is entering the IP of the server. If IP is valid, the user connects to the server, else warning screen denoting that the connection is unsuccessful appears.
6.2.2 Account Screen

Figure 27: Account Screen

If the user has already have an account, he enters login button. To create a new account, he selects create account button.

6.2.3 Create Account Screen

Username, password and e-mail fields are filled in by the user. If the username and e-mail is unique, the account is created when he hits OK button.
6.2.4 Login Screen

The user enters his username and password to this screen to login his account.

6.2.5 Display Screen

This is the main display screen. There is a user’s avatar, gold bar and character’s name, and level on the top left corner of the screen. The chat box is on the bottom right corner. There is also a mini map, which shows the position of the user on the map, on the top right corner.
6.2.6 Turkey Map Screen

This is the 2D map screen where player can choose where to go and see which monument is where. From the main display screen, player can come to map screen by hitting ESC and choosing “Map Screen”.

6.2.7 Quest Details Screen

This is the screen where player can see the details of the quest, which are related to the continuation of the player in the game. By hitting Q key, player can see this screen.
6.2.8 Trade Screen

This is the screen of trade operation. Player can trade with an NPC or another player in this screen. Player can reach this screen when he is in the interaction screen with an NPC or a player.

6.2.9 Character Information Screen

When the player hits left CTRL key, this character information screen comes. All information about the character is displayed in this screen, such as visited places, gained gold, and time.
played. Moreover, by clicking “Friends” button in the character information screen, a player can see his friend list, information about his friends, and can send messages to them.

6.2.10 Options Screen

By hitting ESC and choosing “Options”, the player comes to this screen. From here, he can reach video options, audio options, controller settings option, and gameplay options.
6.2.11 Video Options Screen

This is the video options screen where player can change detail level in the game or screen resolution. This screen is reached from options screen.

6.2.12 Audio Options Screen

This is the audio options screen where player can change sound volume or music volume. This screen is reached from the options screen.
6.2.13 Controller Settings Screen

![Controller Settings Screen](image1)

Figure 38: Controller Settings Screen

This is the controller settings screen from where player can change the controls of the game. This screen is reached from options screen.

6.2.14 ESC Menu Screen

![ESC Menu Screen](image2)

Figure 39: ESC Menu Screen

This is the ESC menu screen, which can be reached by user’s hitting “ESC” button. From here, player can access options or map screens. He can also choose logout or quit game options.
6.3 Screen Objects and Actions

When the user executes the game, the first screen that he comes across is the login window. Once the user enters server ip and his login information, the game starts. From this point, the interfaces that the user will see are denoted in Figure gameflow.

![Game Screen Flowchart]

Figure 40: Game Screen Flowchart

The interface denoted with 1 in Figure gameflow is the main display screen. Once the user hits ESC, second interface, ESC menu screen, will be available. When the user selects map screen in this page, third interface, Map screen, will be seen. Options screen, denoted with 5 in Figure gameflow is connected to audio options screen, video options screen, and controller settings screen, number 4, 9, and 10, respectively. Main display screen is also connected to quest screen, number 6, and character info screen, number 7. From character info screen, the user can reach trade screen.
7 Libraries and Tools

Both the client and the server component of MMORPG will be an application for Windows NT family of operating systems. As both components will be developed with C# programming language, Microsoft’s .NET 4.0 software framework will be needed. XNA 4.0 runtime libraries will be used for client-side graphics computation. The physics engine of client will be leveraged by JigLibx which is specifically designed for XNA. For network communication, the client and the server software will depend on LidGren library.

The interaction between server and client will be maintained on TCP channel. All other communications will be carried out on shared memory. The components within the server will use MPI to communicate over shared memory.

8 Time Planning

The plan is to prepare the detailed design report and prototype demo till the end of the semester. Second semester mainly will be based on configuration management, system design development, and system testing.
8.1 Term 1 Gantt Chart

<table>
<thead>
<tr>
<th>Number</th>
<th>Task</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project Management</td>
<td>14/10/2010</td>
<td>27/5/2011</td>
<td>226</td>
</tr>
<tr>
<td>5</td>
<td>Detailed Design Report</td>
<td>17/12/2010</td>
<td>4/1/2011</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Web Page</td>
<td>12/12/2010</td>
<td>27/5/2011</td>
<td>167</td>
</tr>
<tr>
<td>7</td>
<td>Bug Management</td>
<td>12/12/2010</td>
<td>27/5/2011</td>
<td>167</td>
</tr>
<tr>
<td>8</td>
<td>System Component Analysis</td>
<td>20/11/2010</td>
<td>10/1/2011</td>
<td>52</td>
</tr>
<tr>
<td>9</td>
<td>Server - Client Analysis</td>
<td>20/11/2010</td>
<td>12/12/2010</td>
<td>23</td>
</tr>
<tr>
<td>10</td>
<td>Database Analysis</td>
<td>20/11/2010</td>
<td>20/12/2010</td>
<td>31</td>
</tr>
<tr>
<td>11</td>
<td>Network Analysis</td>
<td>20/11/2010</td>
<td>1/1/2011</td>
<td>43</td>
</tr>
<tr>
<td>12</td>
<td>NPC Analysis</td>
<td>20/11/2010</td>
<td>7/1/2011</td>
<td>49</td>
</tr>
</tbody>
</table>

Figure 41: Term 1 Gantt Chart

8.2 Term 2 Gantt Chart

<table>
<thead>
<tr>
<th>Number</th>
<th>Task</th>
<th>Start</th>
<th>End</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>System Configuration</td>
<td>15/2/2011</td>
<td>1/4/2011</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>Server Configuration</td>
<td>15/2/2011</td>
<td>22/2/2011</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Client Configuration</td>
<td>23/2/2011</td>
<td>1/1/2011</td>
<td>7</td>
</tr>
<tr>
<td>21</td>
<td>Database Test</td>
<td>7/5/2011</td>
<td>10/5/2011</td>
<td>4</td>
</tr>
<tr>
<td>22</td>
<td>NPC Test</td>
<td>11/5/2011</td>
<td>13/5/2011</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>Alpha Test</td>
<td>14/5/2011</td>
<td>16/5/2011</td>
<td>3</td>
</tr>
<tr>
<td>26</td>
<td>Finalization</td>
<td>21/5/2011</td>
<td>27/5/2011</td>
<td>7</td>
</tr>
<tr>
<td>27</td>
<td>Post Production</td>
<td>21/5/2011</td>
<td>27/5/2011</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 42: Term 2 Gantt Chart
9 Conclusion

In this document, initial design for ‘Virtual Turkey” has been presented. The design for data model, system architecture, and user interfaces have been outlined. Decomposition of the system to its components has been presented in system architecture section. This initial design document will guide the implementation and serve as a basis for the detailed design document. MECAC assumes full responsibility for the design suggested in this document.