Oren Farber - 2017

ID: 024910044

Matix Game

Object-oriented programming workshop - Object-oriented Analysis & Design Document

Table of Contents

[Introduction 3](#_Toc485925707)

[Matix Game – Concept and Rules 3](#_Toc485925708)

[Matix System Requirements Analysis 3](#_Toc485925709)

[Game Client Requirements 3](#_Toc485925710)

[Game Server Requirements 4](#_Toc485925711)

[Database 4](#_Toc485925712)

[User Stories 4](#_Toc485925713)

[Log in and registration to the game server 4](#_Toc485925714)

[Update player details 5](#_Toc485925715)

[Change player's password 6](#_Toc485925716)

[Waite for a second player 6](#_Toc485925717)

[Start a new game 6](#_Toc485925718)

[Playing the game 6](#_Toc485925719)

[Get player statistics. 7](#_Toc485925720)

[Computer Player (single player) 7](#_Toc485925721)

[User activities 8](#_Toc485925722)

[Login and registration to game server 8](#_Toc485925723)

[Update player details 9](#_Toc485925724)

[Waite for a second player 10](#_Toc485925725)

[Start a new game 10](#_Toc485925726)

[Playing the game 10](#_Toc485925727)

[Get player statistics. 10](#_Toc485925728)

[Computer Player 10](#_Toc485925729)

[Matix System Architecture Design 11](#_Toc485925730)

[Game Server Design 11](#_Toc485925731)

[Game Management 11](#_Toc485925732)

[Communication 11](#_Toc485925733)

[Game Action 14](#_Toc485925734)

[Database 14](#_Toc485925735)

[Game Client Design 18](#_Toc485925736)

[Client UI 18](#_Toc485925737)

[Client components 18](#_Toc485925738)

[Matix System Installation and Operation 19](#_Toc485925739)

[Logging Tool 19](#_Toc485925740)

[Matix Game Server Installation 19](#_Toc485925741)

[System Database Installation 19](#_Toc485925742)

[Game Client Installation 20](#_Toc485925743)

# Introduction

This document is an object-oriented analysis and design document for a board game named Matix. The document describes the game concepts and rules, analysis and identify the software requirements and specifications and describe the implementation of the conceptual model produced during object-oriented analysis.

# Matix Game – Concept and Rules

The Matix game is a board game for two players. It has an 8\*8 board with random integer numbers between -15 to 15 and one token. A player plays by moving the token horizontally or vertically, one player can move the token horizontally and the second can move the token vertically, to a non-used cell on the board. The purpose of the game is to collect the highest score which is the summary of all the cell values the payer collected. The game ends when there is no free cell for the token to move.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| 15 | -9 |  | 7 |  | 8 |  |  |
|  |  |  |  |  |  |  |  |
|  | 2 |  |  |  |  |  |  |
|  |  | -5 |  |  | 1 |  |  |
| -7 |  |  |  |  |  | 15 |  |
|  | -2 | 4 |  |  | 2 |  |  |
|  |  |  |  |  |  |  |  |

# Matix System Requirements Analysis

These sections describe the requirements for this game system. It describes the required parts and the use cases a player can have.

The system should be a client server system and should allow several clients to connect the server at the same time. The system should keep information about the users, the progress of the games and the results of the games.

## Game Client Requirements

The client should be a windows application with a graphical user interface that allow the user to connect the server and play the game. The application should be installed on the user client machine. The client application shows the user its game board as it receives from the server and should allow the player to play the game according to the games rules. The client should send the game changes to the server and to reflect the second player actions as it received from the server.

## Game Server Requirements

The game server should be a windows application without user interface. The server is the responsible for managing the games of the clients connected to it. The server should support several games and several clients connected to it at the same time. The server should manage players and games information and save it in a database. The server is the game manager and responsible for generating a new board for a new game the enforce the game rules on player's actions that sends to it.

## Database

The database should save all the relevant information for players and games history. It should have only connection to the server. The database should allow us to truck the process of a game by saving all the actions players do during the game.

## User Stories

* Log in and registration to the game server
* Update player details
* Waite for a second player
* Single player
* Start a new game
* Playing the game
* Get player statistics.

### Log in and registration to the game server

While a player starts the client application he needs to connect to the server. To identify the player, the system needs a log in and registration process. The client should allow the player to save its log in properties locally. A user should register to the server so it can save its games data. Registration & login should base on player email address. The password the user uses must not be kept in the database. The system can save a hash value generated from that password. For presentation, a player must use a unique 'nick name' that will identity him among other players.



### Update player details

A registered player can update its details on the game server. The player must be logged in at that time. The process should allow the player to update all its properties except email.



### Change player's password

A player can change it password to a new one as a separate process.

…….

### Waite for a second player

While a player logged in to the game server the player receives a list of available players. The player can select to wait event if there are available players or to play with the server as single player.

………….

### Start a new game

When a second player is selected, a new game can be started. The players receive the board and show it with the player's details.



### Playing the game

On its turn a player can move the token on board to a free cell. The client user interface should enforce the game rules. First user can move horizontally and the second can move vertically.



### Get player statistics.

A player can have statistics about his previous games and score.



### Computer Player (single player)

Select to play the game with the server without a human player. The server should can play as a client.

#### Playing the game



## User activities

The section list the main activities of the system and show an activity diagram for each activity.

* Log in and registration to the game server
* Update player details
* Waite for a second player
* Start a new game
* Playing the game
* Get player statistics.
* Computer Player

### Login and registration to game server

The diagram describes the login and registration process. First a user must register and then he can be logged in to the server. The registration process should only be done once.



### Update player details

A registered player can update its details on the game server.



### Waite for a second player

### Start a new game

### Playing the game

### Get player statistics.

### Computer Player

# Matix System Architecture Design

These following sections describe the system architecture, its components and how the system should be implemented.

## Game Server Design

The game server should be created as a windows service application and implement WCF service for communication between the clients and the server. The server will be connected to a dedicated database that saves all the relevant players and games information.

All three parts should be implemented as separated class libraries so we can handle the software as separated layers.

### Game Management

This section describes the business layer that responsible for managing the games, create and update boards, and managing player's data.

#### Wait for second player

While a player connects to the server he should receive a list of waiting players if exist. If there are no waiting players, the server will add the player to a waiting players list. The server should send notification to all connected clients while a new player is added to the waiting list.

#### Select a player and start a new game.

While a player selects a second player from the waiting list. The server should notify the second player and the second player should apply or reject. On reject the first player receive a notification and remain at the waiting list. On apply the server generate a new game update the database and send the game details, board and players information to the clients.

#### Playing the game

The server notifies each player when its turn and wait to receive a game action. When that message receives the server check that it is legal, update the board of this game and add the information to the database. When updating ended the server notify the first client with acknowledge and update the second client with the change and that his turn to play.

#### Ending the game

When the server receives a game action it should check whether the second player can move the token, on its turn, to a free cell. If there is no free cell the game is ended and the server should calculate the score for each player, update the database and notify the clients who is the winner.

### Communication

This section describes the communication layer is implemented as WCF service as a separated library. The layer implements the protocol between the server and the client. Following the messages definition with the relevant parameter for each message.

#### Login Message

The message sent from the client with log in parameters

##### Parameters

* Email address
* Password

##### Reply

* Status
* Nick name

#### Registration

The message send registration data from a player to the server.

##### Parameters

* Email Address
* First Name
* Last Name
* Nick Name
* Password

##### Reply

* Status

#### Update Player Details

The message allows the user to update its first name last name and nick name.

##### Parameters

* Email Address – for identification only
* First Name
* Last Name
* Nick Name

##### Reply

* Status

#### Change player password

The message allows the user to change its password in the database.

##### Parameters

* Email Address
* Old Password
* New Password

##### Reply

* Status

#### Get waiting player list

The message returns the list of waiting players nick names

##### Parameters

* Email Address

##### Reply

* List of payer's nick names and players' ids

#### Select Second Player

The message sent from the first player to the server after the waiting players received and the user select one of the players to play with.

#### Confirm First Player

The message is sent from the server to the second player

##### Parameters

* Email Address
* Nick Name

##### Reply

* Status

#### Reject Player

The message sent from the second player to the server to reject the first player request to start a game.

##### Parameters

* Email Address
* Nick Name

##### Reply

* Status

#### Start a New Game

The message is a response from the second player to the server and allow to start a new game.

##### Parameters

* Email Address
* Nick Name

##### Reply

* Status

### Game Action

The section describes the main game actions and show a sequence diagram for each main action.

* Log in and registration to the game server
* Update player details
* Waite for a second player
* Start a new game
* Playing the game
* Get player statistics.
* Computer Player

### Database

This section describes the data access layer that contains database interface and the database tables.

#### Database Tables

This section describes the tables in the database. The purpose of the database is to store players and game information.

##### Players

This table contains player's unique information. The table contains one record for each player register to the system.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Player Id | Create Time | First Name | Last Name | Nick Name | Password Hash | Email |
|  |  |  |  |  |  |  |

* Player Id - Unique player id – Primary key
* Create Time – The first time a player register to the game
* First Name – Player first name
* Last Name – Player last name
* Nick Name – Players nick name
* Password Hash - A generated hash string from the password.
* Email – Players unique email address

##### Players Login

The table contains information of player's logins.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Login id | Player Id | Login Time | IP Address | Logout Time | Reason |
|  |  |  |  |  |  |

* Login id – Unique id for this record – Primary key
* Player Id – Unique player id
* Login Time – Login event time
* IP Address – Players client IP address
* Logout Time – Log out event time

##### Games

This table stores game unique information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Game ID | Create Time | Horizontal Player ID | Vertical Player ID | Cells Matrix |
|  |  |  |  |  |

##### Game Activities

This table describe the process of the game.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Activity ID | Game ID | Player ID | Activity Time | Cell Row | Cell Column | Cell Value |
|  |  |  |  |  |  |  |

##### Players History

This table is a connection table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| History ID | History Time | Player ID | Game Id | Win | Score |  |
|  |  |  |  |  |  |  |

#### Data Layer Interface

This section describes the interface the DAL expose to update or retrieve information to and from the database.

##### Add Player

Add a new player to the database and return its id.

##### Parameters

* Email address
* First name
* Last name
* Nick Name
* Password Hash

##### Return

* Player Id
* Status

##### Payer Login

Update player login

##### Parameters

* Email address
* Password Hash

##### Return

* Player Id
* Status

##### Update Player Information

Update player information

##### Parameters

* Player Id
* First name
* Last name
* Nick Name

##### Return

* Status

##### Update Players password

Update player password in the database

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

##### Get Player Details

Get player information based on its player id

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

##### Get Player Statistics

Get player games and score information

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

##### Create New Game

Creates a new game record in the database. The method return the new created game id.

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

##### Update Game Activity

Update database with game activities

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

##### End Game

Update the database that a game was ended and that we have a winner.

##### Parameters

* Player Id
* New Password Hash

##### Return

* Status

## Game Client Design

The game client application should be a WPF windows application.

### Client UI

The section describes the UI components we should use.

#### Colors

#### Font

The font selected for the client is 'Rockwell'.

### Client components

#### Cell

A cell represents a square with its attributes.

* Value – The random generated value of the cell. This value is added to the player score after the sell is selected.
* AlreadyUsed - A Boolean member indicates whether the cell used by one of the players or it is still free.
* IsToken – A Boolean member that indicates whether the current cell is a token. Should be only one token at a time on the board.

#### 

#### Board

The board is a matrix of cells with 8 rows and 8 columns.

* List of 64 cells

#### Player

A player represents one of the players in the game.

* PlayerID – The id of the player in the system database.
* PlayerName – The nick name of the player in the database.
* CurrentScore – The current score in the game.

#### User

A user is the player that use this client application. The user logged in the server.

* UserName – the name use to logged into the system.

# Matix System Installation and Operation

This section describes the installation and other operation needed to successfully install the game server as a windows service create the database and install a client.

## Logging Tool

All parts of the system should have log file that describes the flow of the software behavior. We should create one log file for the client and a second log file for the server both can use log4net for implementing the logger.

## Matix Game Server Installation

The section describes the operations needed to install the game service

* Install the service
* Add access rights to Network Service to the service location folder.

In order to install the game server as a service run the following command. Browse to the bin directory where MatixGameService.exe is located and run the following command.

***Installutil MatixGameService.exe***

If you have modified the service that is already installed, you can uninstall it by using following command:  
***Installutil /u MatixGameService.exe***

## System Database Installation

This section describes the scripts crating the Matix database. The database is based on SQL server and an instance of database service version 2008 or later should be installed. It can be installed on the Game Server machine or on a separated one.

First we should create the database using the following script.

* [Database\CreateMatixDatabase.sql](Database/CreateMatixDatabase.sql)

After the database created we can add the tables using the following scripts

* [Database\CreatePlayersTable.sql](Database/CreatePlayersTable.sql)
* [Database\CreatePlayersLoginTable.sql](Database/CreatePlayersLoginTable.sql)
* [Database\CreatePlayersHistoryTable.sql](Database/CreatePlayersHistoryTable.sql)
* [Database\CreateGamesTable.sql](Database/CreateGamesTable.sql)
* [Database\CreateGameActivitiesTable.sql](Database/CreateGameActivitiesTable.sql)

## Game Client Installation