Oren Farber - 2017

ID: 024910044

Matix Game

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

Table of Contents

[Introduction 3](#_Toc491765212)

[Matix Game – Concept and Rules 3](#_Toc491765213)

[Matix System Requirements Analysis 3](#_Toc491765214)

[Game Client Requirements 3](#_Toc491765215)

[Game Server Requirements 4](#_Toc491765216)

[Database 4](#_Toc491765217)

[Player Information 4](#_Toc491765218)

[Game Information 4](#_Toc491765219)

[User Stories 4](#_Toc491765220)

[Login and Registration to the Game Server 5](#_Toc491765221)

[Update Player Details 6](#_Toc491765222)

[Change Player's Password 6](#_Toc491765223)

[Multi Players Game 7](#_Toc491765224)

[Single Player Game 8](#_Toc491765225)

[Start a new game 8](#_Toc491765226)

[Playing the Game 9](#_Toc491765227)

[Get player statistics. 9](#_Toc491765228)

[User activities 10](#_Toc491765229)

[Login and registration to game server 10](#_Toc491765230)

[Update player details 11](#_Toc491765231)

[Multi Player Game 11](#_Toc491765232)

[Single Player Game 12](#_Toc491765233)

[Start a New Game 12](#_Toc491765234)

[Playing the game 13](#_Toc491765235)

[Get Player Statistics 13](#_Toc491765236)

[Matix System Architecture Design 14](#_Toc491765237)

[Game Server Design 14](#_Toc491765238)

[Game Management 14](#_Toc491765239)

[Game Action 15](#_Toc491765240)

[Communication 17](#_Toc491765241)

[Database & Data Access Layer 19](#_Toc491765242)

[Data Layer Interface 22](#_Toc491765243)

[Game Client Design 24](#_Toc491765244)

[Client UI 24](#_Toc491765245)

[Client components 26](#_Toc491765246)

[Matix System Installation and Operation 27](#_Toc491765247)

[Logging Tool 28](#_Toc491765248)

[Matix Game Server Installation 28](#_Toc491765249)

[System Database Installation 28](#_Toc491765250)

[Game Client Installation 28](#_Toc491765251)

# 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. It should have connection only 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.

### Player Information

For players, we should use email address as an identification key and should be presented to other players with his nickname.

* First name
* Last name
* Nick name
* Email
* Password
* Type

### Game Information

For each game, we must save the it's initiate time the players of the game and its generated board. During the game, we should keep the activity of the games so we can reconstruct the process of the game.

## User Stories

The section describes the following user stories

* Log in and registration to the game server
* Update player's details
* Update player's password
* Multi player game
* Single player game
* Start a new game
* Playing the game
* Get player statistics.

### Login 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 address.



### Change Player's Password

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



### Multi Players Game

A player can choose to play with other players by selecting multi player game. The client should add the player to the waiting list and show the player a list of other waiting players



### Single Player Game

Select to play the game with the server without a human player. While selecting the server a game should be started



### Start a new game

When a second player is selected, a new game can be started. The players receive the board generated by the server 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 application should enforce the game rules and instruct the player with current turn and playing direction



### Get player statistics.

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



## 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
* Multi Player Game
* Single Player Game
* Start a new game
* Playing the game
* Get player statistics.

### 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 be done only once.



### Update player details

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



### Multi Player Game

While a player selects to play with a human player the server adds him to the waiting players list and send back a list of waiting players.



### Single Player Game

While a player selects to play with the server. The server generates a new game and sends it details to the client



### Start a New Game

While a game is started the server generates a random board, select for each player its direction for multi players game the server chooses who will start the game. For single player game, the human player always starts.



### Playing the game

On its turn, the player can move the token to a new cell by double clicking the new one. The game page should inform the player who's should play. The server should verify that the action is valid, update the database with the action and check that the game is not ended yet.



### Get Player Statistics

User can show its playing results



# 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 and select to play with another player, he should receive a list of waiting players if exist. The server should add the player to a waiting players list and send notification to all other waiting clients with the updated list.

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

While a player selects a second player from the waiting list. The server should start a new game and sends the generated board 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.

### Game Action

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

* Login and Registration to the Game Server
* Update Player Details
* Multi Player Game
* Single Player Game
* Start a New Game
* Playing the Game
* Get Player Statistics

#### Login and Registration to the Game Server

The sequence diagram describes the flow for registration and login scenarios



#### Update Player Details

#### Multi Player Game

#### Single Player Game

#### Start a New Game

#### Playing the Game

#### Get Player Statistics

### Communication

This section describes the communication layer. This layer is implemented as a separated library using WCF technology. The communication protocol supports Duplex type and allows sending messages from the client to the server and to use callback method for notifying the client on changes.

#### Operation Status Enumeration

Enumerate the available result status of operation

* Success,
* Failure,
* Invalid Email,
* Invalid Password,
* Operation time Out,
* Rejected,
* Invalid Action

#### Login Message

The message sent from the client with login parameters

##### Parameters

* LoginData (structure)

##### Reply

* LoginResult (structure)

#### LoginData

The class contains the player information for login message.

* Email (string)
* Password (string)

#### LoginResult

The class contains the result information for login message

* Operation Status (enumeration)
* Player's Nickname (string)

#### User Registration

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

##### Parameters

* UserInformationData (structure)

##### Reply

* RegistrationResult

#### UserInformationData

* Email Address (string)
* First Name (string)
* Last Name (string)
* Nickname (string)
* Password (string)

#### RegistrationResult

* Operation Status (enumeration)
* Message (string)

#### Update User Details

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

##### Parameters

* UserInformationData (structure)

##### Reply

* Operation Status (enumeration)

#### 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 a list of current waiting players with some statistics information

##### Parameters

* The current players email address (string)

##### Reply

* List of WaitingPlayerResult structure

#### Waiting Player Result

The class contains the information of the current waiting players and the operation status of the message.

* List of WaitingPlayer (structure)
* Operation Status (enumerator)

#### WaitingPlayer

The class contains information of waiting players that can be shown to the player

* Nickname (string)
* Total Games (integer)
* Number of winnings (integer)
* Total Score (integer) – Can be negative value

#### Select Player to Play

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.

##### Parameters

* Email Address (string) – The email address of the sender
* Nickname (string) – The nickname of the second player

##### Reply

* Operation Status (enumerator)

#### Select Robot to Play

The player selects to start a new game with the server.

##### Parameters

* Email Address (string) – The email address of the sender

##### Reply

* Operation Status (enumerator)

#### Quit the Game

Notify the server that a player quite the current game he is playing

##### Parameters

* Email Address (string)

##### Reply

* None

### Database & Data Access Layer

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

#### Database Tables

This section describes the database tables. 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 | Type |
|  |  |  |  |  |  |  |  |

* 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 - must have a unique value
* Password Hash - A generated hash string from the password.
* Email – Players unique email address – must have a unique value
* Type – Whether the player is a Human or a Robot

##### 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 – Foreign key from Players table
* Login Time – Login event time
* IP Address – Players client IP address
* Logout Time – Log out event time
* Reason – A string for short description

##### Games

This table stores game unique information.

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

* Game Id - Unique id for this record – Primary key
* Create Time – Create time event
* Horizontal Player Id – The Id of the horizontal player – Foreign key from Players table
* Vertical Player id – The Id of the vertical player - – Foreign key from Players table
* Cell Matrix – An XML matrix that contains the generated game board.

##### Game Activities

This table describe the process of the game.

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

* Activity Id - Unique id for this record – Primary key
* Game Id – The id of the game in Games table - Foreign key
* Player Id – The Id of the player that do the game activity - Foreign key from Players table
* Cell Row – The row number of the new selected cell
* Cell Column – The column number of the new selected cell
* Cell Value – The value of the selected cell

##### Players History

This table is a connection table to record the game result for each player and game.

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

* History Id - Unique id for this record – Primary key
* History Time – The time of the event, should be the time the game ended
* Player Id – The player id this information refers to - Foreign key from Players table
* Game Id – The game id this information refers to - Foreign key from Games table
* Win – True whether this player wins this game
* Score – The score of the player in the game- This value should be the summary of all the cell values selected by that user during the game.

### 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. The application should allow the player to connect and login to the game server and to allow him to play the game.

### Client UI

The section describes the UI components we should use.

#### Welcome Page

The welcome page allows the user to select the requested operation login the system and selecting a game type.

#### Error Page

The Error page shows an error in case the client could not connect to the server.

#### Login Page

The login page allows the user login into the system using email address and password or selecting to register in case of a new user.

#### Registration Page

The registration page allows the user to register its basic information to the server. On registration, the user must update the following parameters

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

#### Update player details

The update page allows the user to update the following fields

* First Name
* Last Name
* Nick Name

#### Change Password

The page allows the user to change its current password

#### Game Page

The game page allows the player to play the game. It shows the board and players name and score. The page allows the player on its turn to move the token to a valid position.

….

##### Board User Control

The board should be implemented as a user control. The board shows 64 cells with integer values. The player can select a cell and to receive its value by double clicking on the cell.

##### Circular Progress Bar User Control

Use a waiting control on top of the game after the page is shown and still waiting to the generated board on the server

#### Players List Page

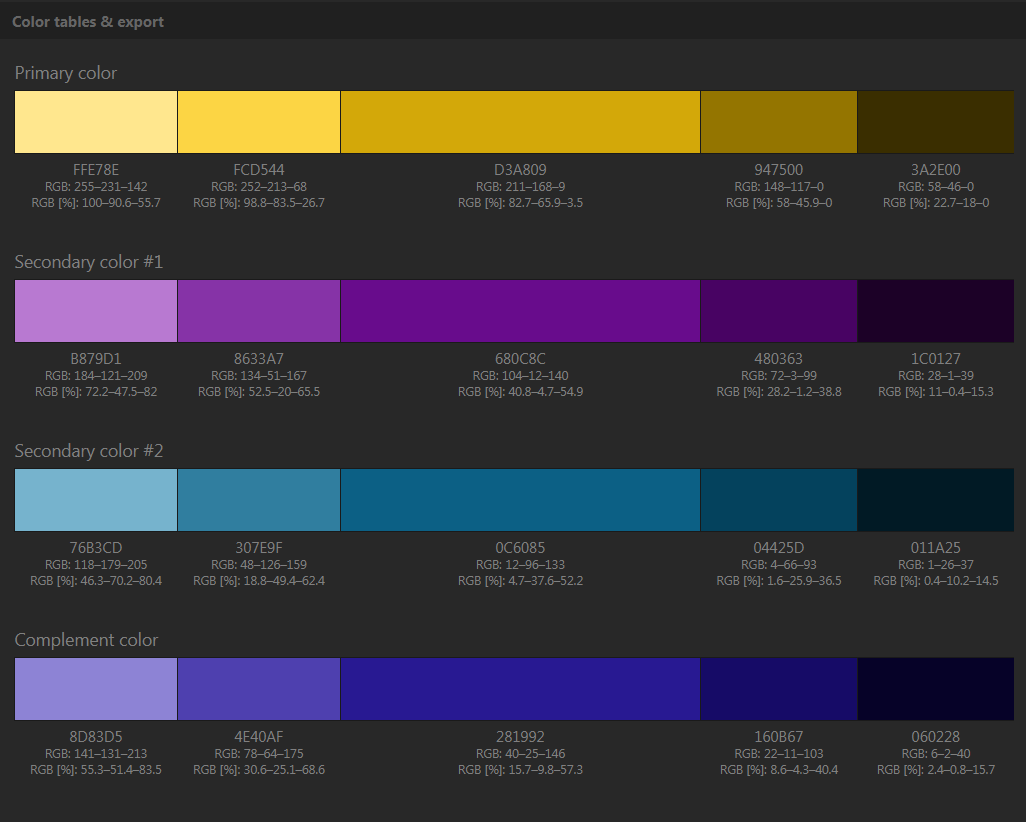
While a user select to play with another player he should receive the list of currently waiting players. The page show the players details and allows the user the choose a player to play with.

#### Player Statistics Page

The page shows the user statistics. The application should show it as a user request or at the end of a game.

#### Colors

This section describes the colors selected for the application. Basically, I choose to use colors from the following palette.



#### Font

The font selected for the client is 'Rockwell'.

### Client components

This section describes all the components, that are not pages, which are part of the game client

#### 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