Multiplayer browser games. Checkers.

Maksym Kurylovych
Institute of Computer Science
University of Tartu
max.kurylovych@gmail.com

1. Abstract

In this article we will consider some different ways of checkers implementation and will see on some functions of SignalR. Additionally, we will see some code on JavaScript and ASP.Net

2. Introduction

We live in a great time, a time where technologies are closely linked to everyday routine, where games are no longer just games but an artwork. People of all ages love videogames and if single-player games were popular in 1990's - early 2000's, after 2005 multiplayer games became the most desired. However, browser games become more and more popular as well. That is why the current project is related to games through the browser.

3. Ways of implementation

There were several approaches to implement and to realize the application. Although, there are a lot of ways of doing this author considered three different approaches that contains different programming languages and different technologies. The game should be based on Standard U.S. Rules [1].

3.1 PHP

The first, the most obvious idea was to use PHP and its Ratchet [2]. Right now it is very common to use this programming language to realize small projects. Why? Well, it is a script language so code there is very short, not like on C# or Java and it means that even person without deep knowledge in programming would be able to create simple application there in a very short time. That is why PHP very popular among web developers.

3.2 Java

The second considered way was to implement everything in Java with the help of jWebSocket [3]. jWebSocket is a flexible, compatible, and scalable technology used for particularly online collaboration, online gaming, streaming, remote control and monitoring applications profit from the high speed data exchange.

3.3 .Net

More and more people are using PHP for web development, so Microsoft has decided to make .NET alive again, or to make it more popular. That is why the company has created a library called SignalR [4] that allows programmers to create real-time applications in a very simple way. The good thing of this library is that SignalR supports Web Sockets and includes different APIs as well as group connections and authorization.

4. SignalR

Of course, SignalR is a very usefull but you should not use this technology unless you need what it offers, which is, to say, a basic, duplex web service layer focusing on JavaScript clients. Basically, if you are new to .NET that is not the right thing with what you have to start. However, if you are going to work with ,NET clients, it would be better is you will stick with WCF as it is more flexible. SignalR is a good choice when you need a .NET based web service backend for a JavaScript application because it is certainly simpler to configure that WCF. The only main

real disadvantage to it is that it is pretty new and miss some important features that programmers would like to have and also it still has some bugs (these will eventually get fixed, of course).

SignalR [5][6] based on work with web-sockets. On client side after connection we receive proxy object that gives us an opportunity to use hub's methods that lie on the server. During the app creation the main problem was in understanding how SignalR works; why when we want to call method we should call it with a small letter? Right now it is clear that it is proxy object specific but it was not mentioned anywhere. Another problem was that client cannot identify from what method the message has been received. It would be really good if it would be possible to completely encapsulate methods but not only when we call them. Client does not understand from where message came, he only knows that it comes. Modern library that was created to simplify client – server communication seems very low-level.

5. The Code

Below, you can see an example of basic functions that serves to create a chat for the application. The chat was created in the application to allow players to communicate with each other.

```
// Add new user
public void Connect(string userName)
{
    var id = Context.ConnectionId;

    if (!Users.Any(x => x.ConnectionId == id))
    {
        Users.Add(new User { ConnectionId = id, Name = userName });

        // Send message to current user
        Clients.Caller.onConnected(id, userName, Users);

        // Send message to all users except the current one.
        Clients.AllExcept(id).onNewUserConnected(id, userName);
    }
}
```

pic.1. Add new user

```
// Disconnect user
public override System.Threading.Tasks.Task OnDisconnected(bool stopCalled)
{
    var item = Users.FirstOrDefault(x => x.ConnectionId == Context.Connec-
tionId);
    if (item != null)
    {
        Users.Remove(item);
        var id = Context.ConnectionId;
        Clients.All.onUserDisconnected(id, item.Name);
    }

    return base.OnDisconnected(stopCalled);
}
```

pic.2. Disconnect user

```
// Send message
public void Send(string name, string message, int flag)
{
    if (Users[0].ConnectionId == Context.ConnectionId)
    {
        Clients.All.addMessage(name, message, flag, 1);
    }
    if (Users[1].ConnectionId == Context.ConnectionId)
    {
        Clients.All.addMessage(name, message, flag, 2);
    }
}
```

pic.3. Send message

The following code declares a reference to a hub proxy.

```
var chat = $.connection.chatHub;
```

pic.4. Declares a reference

The following code is how you create a callback function in the script.

```
chat.client.broadcastMessage = function (name, message) {
        // Html encode display name and message.
        var encodedName = $('<div />').text(name).html();
        var encodedMsg = $('<div />').text(message).html();
        // Add the message to the page.
        $('#discussion').append('<li><strong>' + encodedName
            + '</strong>:  ' + encodedMsg + '</li>');
    };
```

pic.5. Callback a function

The following code shows how to open a connection with the hub.

```
$.connection.hub.start().done(function () {
        $('#sendmessage').click(function () {
            // Call the Send method on the hub.
            chat.server.send($('#displayname').val(), $('#message').val());
            // Clear text box and reset focus for next comment.
            $('#message').val('').focus();
        });
    });
```

pic.6. Open connection

Below you can see the code that represents how login works in the program. Basically, users have to login first and only after they would be able to see chat and a board with checkers.

```
var chat = $.connection.chatHub;
var num_player = 0;


$('#chatBody').hide();
$('#loginBlock').show();
// Link to a afromatic genereted hub proxi

// function that calls by hub what message received
chat.client.addMessage = function (name, message, flag, userid) {
    console.log(flag);
    if (flag == 0) {
        // Add message on a webpage
        $('#chatroom').append('<p><b>' + htmlEncode(name)
            + '</b>: ' + htmlEncode(message) + '</p>');
    }
    else if (flag == 1) {
        //moveChecker(name, message, r0, c0);
        //var piece = pieces[message];
        //var tile = $.evalJSON(name);
        console.log(userid + 'user id');
        onlineMove(name, message);
    }
    else if (flag == 3) {
        num_player = userid;
    }
};
```

pic.7. Function that runs by hub

The following code represents the move piece function. You can see how checker change it position and what will happen if it reaches the opposite side of the board.

```
//moves the piece
this.move = function (tile) {

    this.element.removeClass('selected');
    if (!Board.isValidPlacetoMove(tile.position[0], tile.position[1])) return
false;
    //make sure piece doesn't go backwards if it's not a king
    if (this.player == 1 && this.king == false) {
        if (tile.position[0] < this.position[0]) return false;
    } else if (this.player == 2 && this.king == false) {
        if (tile.position[0] > this.position[0]) return false;
    }
    //remove the mark from Board.board and put it in the new spot
    Board.board[this.position[0]][this.position[1]] = 0;
    Board.board[tile.position[0]][tile.position[1]] = this.player;
    this.position = [tile.position[0], tile.position[1]];
    //change the css using board's dictionary
    this.element.css('top', Board.dictionary[this.position[0]]);
    this.element.css('left', Board.dictionary[this.position[1]]);
    //if piece reaches the end of the row on opposite side
    //crown it a king (can move all directions)
    if (!this.king && (this.position[0] == 0 || this.position[0] == 7))
        this.makeKing();
    Board.changePlayerTurn();
    return true;
};
```

pic.8. Move the piece function

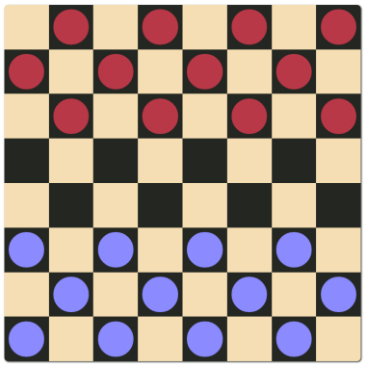
Below, you can see the representation of board and figures realization. 0 means that the field is empty, 1 – first player figures, 2 – second player figures.

```
//The initial setup
var gameBoard = [
    [0, 1, 0, 1, 0, 1, 0, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 1, 0, 1],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [2, 0, 2, 0, 2, 0, 2, 0],
    [0, 2, 0, 2, 0, 2, 0, 2],
    [2, 0, 2, 0, 2, 0, 2, 0]
];
```

pic.9. The board and its figures

6. Results

Below, you can see some results: deck with checkers and chat windows.



pic.10. The boardx

pic.11. User chat 1

pic.12. User chat 2

7. Conclusion

During the seminar the checkers game was created. Additionally, we took a look at some JavaScript and .Net code and looked at SignalR technology by Microsoft.

8. References

1 Erik Arneson 'How to Play Checkers - Standard U.S. Rules' 04 April, 2017, https://www.thespruce.com/play-checkers-using-standard-rules-409287

2 Rarchet. WebSockets for PHP, http://socketo.me/

3 jWebSocket. The open source solution for realtime web developers, http://jwebsocket.org/

4 Learn About ASP.NET SignalR, https://www.asp.net/signalr

5 Zhuyun Dai 'ASP.NET SignalR Basis Step by Step (Part 1)' 07 August, 2013, https://www.codeproject.com/Articles/633378/ASP-NET-SignalR-Basis-Step-by-Step-Part

6 Tutorial: Getting Started with SignalR 2 https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/tutorial-getting-started-with-signalr

8 Jamsa, Kris. C/C++/C# Programmer's Bible. The Ultimate Guide to C/C++/C# Programming. Second Edition. Delmar, 2002