MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION
Federal State State-Financed Educational Institution of High Professional Education
South Ural State University (National Research University)
Faculty of Computational Mathematicsand Informatics
Department of SystemProgramming

THESIS IS CHECKED

Reviewer,
CEO of AppBit Software, Ltd
_____ D.A. Dikov
"01" 06 2016

ACCEPTED FOR THE DEFENSE

Head of the department, Dr. Sci., Prof.
_____ L.B. Sokolinsky
"13" 06 2016

DEVELOPMENT OF A WEB-SOCKET BASED ONLINE BROWSER GAME

GRADUATE QUALIFICATION WORK
SUSU–02.04.02.2016.115-133.GQW

Supervisors:

PhD, Assoc. prof.
_____ P.S. Kostenetskiy

Teacher
_____ A.I. Semenov

Author,
the student of the group VMI-216
_____ S.J.M. Janabat

Normative control
_____ O.N. Ivanova
"01" июня 2016

Chelyabinsk–2016

MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION

Federal State State-Financed Educational Institution of High Professional Education
South Ural State University (National Research University)
Faculty of Computational Mathematicsand Informatics
Department of SystemProgramming

APPROVED

Head of the department,
Dr. Sci., Prof.
_____ L.B. Sokolinsky
"09" 02 2016

TASK
of the master graduate qualification work
for the student of the group VMI-216
Saad Jasim Mohammed Janabat
in master direction 02.04.02
"Fundamental Informatics and Information Technologies"
(master program "Database Technologies")

1. **The topic** (approved by the order of the rector from 15.04.2016 No. 661) Development of a Web-Socket based online browser game.
2. **The deadline for the completion of the work:**05.06.2016.
3. **The source data for the work**

3.1. Pterneas V. Getting Started with HTML5 Web-Socket Programming. – USA: Packt Publishing, 2013. – 110 p.

3.2. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. // IEEE Internet Computing. – USA: IEEE, 2010. – Vol.14, No 6. – P. 80–83.

3.3. Cederholm D. Handcrafted CSS: More Bulletproof Web Design. – USA: NewRiders, 2009. – 240 p.

3.4. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – Boston: Addison-Wesley Professional, 2013. – 208 p.

4. **The list of the development issues**

4.1. Develop Web-Socket based online browser game.

4.2. Study JavaScript, HTML5 and CSS.

4.3. Learn Node.js and Web-Socket technologies.

4.4. Implement the client part.

4.5. Implement the sever part.

4.6. Test developed system.

5. Issuance date of the task: 09.02.2016.

Supervisors

PhD, Assoc. prof. P.S. Kostenetskiy

Teacher A.I. Semenov

The task is taken to perform S.J. Janabat

Student: Tanabat Saad Jasim Mohammed

Supervisor: A.I. Semenov

Topic: Development of a Web-Socket based online browser game

The calendar plan of the execution of master graduate qualifying work (GQW)

No	Phase	Duration	Deadline	Report	Actual date of execution	Supervisor's signature
1.	Introduction and literature review	1 month	February, 25th	1. Task of the master graduate qualification work 2. Text of Introduction 3. References	25.02	
2.	Development of the model, design of the system	1 month	March, 15th	1. Text of chapter 1 (theoretical part).	15.03	
3.	Implementation of a system	1 month	April, 15th	1. Software system 2. Text of chapter 2 (implementation part).	10.04	
4.	Testing and debugging of the system, experiments	2 weeks	May, 1st	1. Set of tests 2. Text of chapter 3 (experimental part).	1.05	
5.	Full text	2 weeks	May, 15th	1. Full text of GQW	13.05	
6.	Validation of the text by supervisor	1 week	May, 22nd	1. Electronic version of the GQW text checked by the supervisor	22.05	
7.	Normative control	3 days	May, 25th	1. Twisted text of GQW signed by student, supervisor and normative controller	25.05 10.06.16	
8.	Proposal defense	1 week	May, 25th – June, 1st	1. Twisted text of GQW signed by student, supervisor and normative controller for the signature of the Head of the Department Head about accepting for the defense 2. A signed review of the supervisor 3. A review of the reviewer, signed and notarized at his place of work 4. Implementation act (if exists) 5. Presentation of the report in PowerPoint	25.05 03.06.16	

TABLE OF CONTENTS

ACKNOWLEDGEMENT

First I would like to express thanks to our God to help me for do this project and foremost I offer my sincerest gratitude to my supervisor, teacher A.I. Semenov. who has supported me I attribute the level of my Master's degree to his encouragement and effort and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

Special thanks are given to Faculty of Computational Mathematics and Informatics,Department of System Programming of SUSU.I would also like thank all my friends who help me and supported me.

Last but not the least important I would like also thanks to my family members which includes elder brother and my sisters for their supports and encouragement through my live.Without their support it is impossible for me to finish my college and graduate education seamlessly.

I present the fruit of my success, to the spirit of the martyr's my father and my mother and to all the souls of the martyr's heroes of the popular crowd and the Iraqi Army who are defending Iraq and his land against terrorists.

INTRODUCTION

Problem

The computer game industry has experienced a huge increase in active players and income during the last decade. The emergence of online-multiplayer games has huge impacts on the value chain in the industry.

An online game [2] is a video game played over some form of computer network. This network is usually the Internet or equivalent technology, but games have always used whatever technology was current:modems before the Internet, and hard-wired terminals before modems.

The expansion of online gaming has reflected the overall expansion of computer networks from small local networks to the Internet and the growth of Internet access itself. Online games can range from simple text based environments to games incorporating complex graphics and virtual worlds populated by many players simultaneously. Many online games have associated online communities, making online games a form of social activity beyond single player games.

Browser games are computer games played directly in the web browser and do not need software installation. This accessibility has made those extremely popular and browser games have lured many players who would never buy a regular computer game to start playing. In the recent years, the evolution of browser games has been fast and a lot of new ones have been launched.

There is also been evolution on technologies used in browser games. Traditional implementation technologies, PHP [17], ASP [1] and Java [13] jsp-pages have gained new competitors in the form of browser plugin players, which allow use of sophisticated 3D graphics.

In addition to implementation technologies, different kinds of accessories for browser games have appeared. The games can be located inside an application platform, for example Facebook social utility. Different browser plugins are used to enchant play experience either to process game data for the player, or to modify the game graphics. Game engines have a strong position in regular computer game

development and recently those have appeared into browser game development also.

So, the main task of the project is to develop the Web-Socket based online browser game.

To reach this aim the following tasks are necessary to be solved:

- to study JavaScript, HTML5 and CSS;
- to learn Node.JS and Web-Socket technologies;
- to develop client part;
- to develop server part and to set connection;
- to test the developed system.

Structure

The work consists of introduction, 4 chapters, conclusion and reference list. The volume of work is 27 pages, the amount of the bibliography is 21 items.

1. TECHNOLOGIES REVIEW

HTML and CSS

HTML [6] is the standard markup language used to create web pages. With CSS (style sheet language used for describing the presentation of a document written in a markup language) [7] it is cornerstone technology used to create web pages.

HTML describes the structure of a website semantically and, before the advent of Cascading Style Sheets (CSS), included cues for the presentation or appearance of the document (web page), making it a markup language, rather than a programming language.

HTML elements form the building blocks of HTML pages. HTML allows images and other objects to be embedded and it can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets.HTML can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. HTML markup can also refer the browser to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

JavaScript

JavaScript [8] is a high-level, dynamic, untyped, and interpreted programming language. Alongside HTML and CSS, it is one of the three core technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions,

but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Scala

Scala [19] is a general purpose programming language. Scala has full support for functional programming and a very strong static type system. Designed to be concise, many of Scala's design decisions were inspired by criticism of the shortcomings of Java.

Scala source code is intended to be compiled to Java bytecode, so that the resulting executable code runs on a Java virtual machine. Java libraries may be used directly in Scala code and vice versa (language interoperability). Like Java, Scala is object-oriented, and uses a curly-brace syntax reminiscent of the C programming language. Unlike Java, Scala has many features of functional programming languages like Scheme, Standard ML and Haskell, including currying, type inference, immutability, lazy evaluation, and pattern matching. It also has an advanced type system supporting algebraic data types, covariance and contravariance, higher-order types (but not higher-rank types), and anonymous types. Other features of Scala not present in Java include operator overloading, optional parameters, named parameters, raw strings, and no checked exceptions.

Java Platform Enterprise Edition

Java Platform, Enterprise Edition or Java EE is a widely used enterprise computing platform developed under the Java Community Process. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition (Java SE), providing an API for object-relational mapping, distributed and multi-tier architectures, and web services. The platform incorporates a design based largely on modular components running on an application server. Software for Java EE is primarily developed in the Java programming language. The platform emphasizes convention ove configuration

and annotations for configuration. Optionally XML can be used to override annotations or to deviate from the platform defaults.

Node.js

Node.js [16] is an open-source, cross-platform runtime environment for developing server-side Web applications. Although Node.js is not a JavaScript framework, many of its basic modules are written in JavaScript, and developers can write new modules in JavaScript.

Node.js has an event-driven architecture capable of asynchronous I/O. Asynchronous I/O is a form of input/output processing that permits other processing to continue before the transmission has finished.

Such design choices aim to optimize throughput and scalability in Web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games).

Socket.IO

Socket.IO [20] is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for node.js. Both components have a nearly identical API. Like Node.js, it is event-driven.

Socket.IO primarily uses the Web-Socket protocol with polling as a fallback option, while providing the same interface. Although it can be used as simply a wrapper for Web-Socket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O.

JSON

JSON [14] is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is the most common data format used for asynchronous browser/server communication.

JSON grew out of a need for stateful, real-time server-to-browser communication without using browser plug ins such as Flash or Java applets, which were the dominant methods in the early 2000s.

JSON is a language-independent data format. It derives from JavaScript, but as of 2016, code to generate and parse JSON-format data is available in many programming languages.

Conclusion

The Node.js technology was chosen because of its fast work (comparing to JEE) and good popularity (comparing to Scala).

The other technologies were chosen to create this project because of the following reasons:

– they have good compatibility with each other;

– they all have cross-browser support [4, 3,5];

– the project created using these technologies would be platform independent.

2. SOFTWARE DESIGN

2.1. Game description

Fig. 1 demonstrates the approximate gameplay of the project.

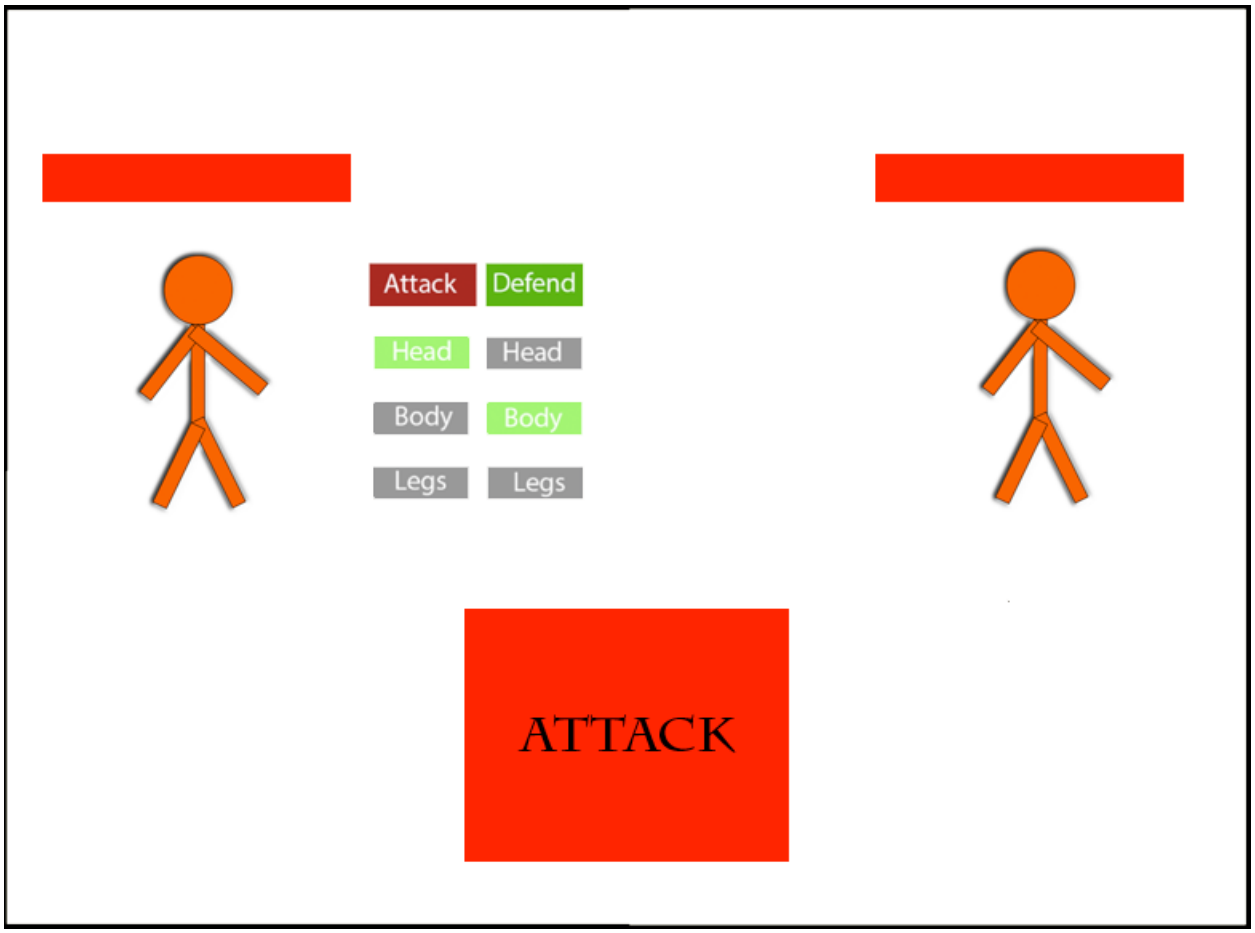

Fig. 1. Gameplay

On the left part of the game window there are:

– schematic image of your character;

– buttons to choose which, part of body is attacked;

– buttons to choose which, part of body is defended.

The right part of the window contains schematic image of your opponent.

On the bottom of the game window there is “Make attack” button. When player presses it, the move is done.

The approximate gameplay is:

– Player 1 chooses attack type;

– Player 1 chooses defense type;

– Player 1 presses «Make attack» button.
– Player 2 chooses attack type;
– Player 2 chooses defense type;
– Player 2 presses «Make attack» button;
– Server checks if there are any changes in players' stats;
– Server sends the differences to players;
– players make new move.

The game finishes when health points of any player become 0.

2.2. Game behavior

The game contains two connected parts or sub-systems: client and server.

Fig. 2 demonstrates the use case diagram [11] of the game.

When player wants to start the game he opens the web browser and goes to the game website (at the current moment it's hosted on localhost). The client part sends to the server the request to create the Web-Socket connection.

After getting the request the server creates the connection using Web-Sockets technology. Then server sends information about created Web-Socket to the client, so the connection is established.

2.2.1. Client

Client is a component responsible for user commands handling and sending information about them to server using Web Sockets. Also client draws the game scene and shows it to user.

Game scene consists of the parts described in paragraph 2.1.

The client has 5 main functions: *log in*, *log out*, *choose attack*, *choose defence* and *start the fight.*

Log in case means that client connects to the server and sends to it information such as login and password.

Log out case means that client disconnects from the server.

Choose attack case means that player chooses which part of opponents body to attack: head, body or legs.

Choose defense case means that player chooses which part of his body to protect: head, body or legs.

Start the fight case means that player finished his move and starts the fight after finishing previous two cases.

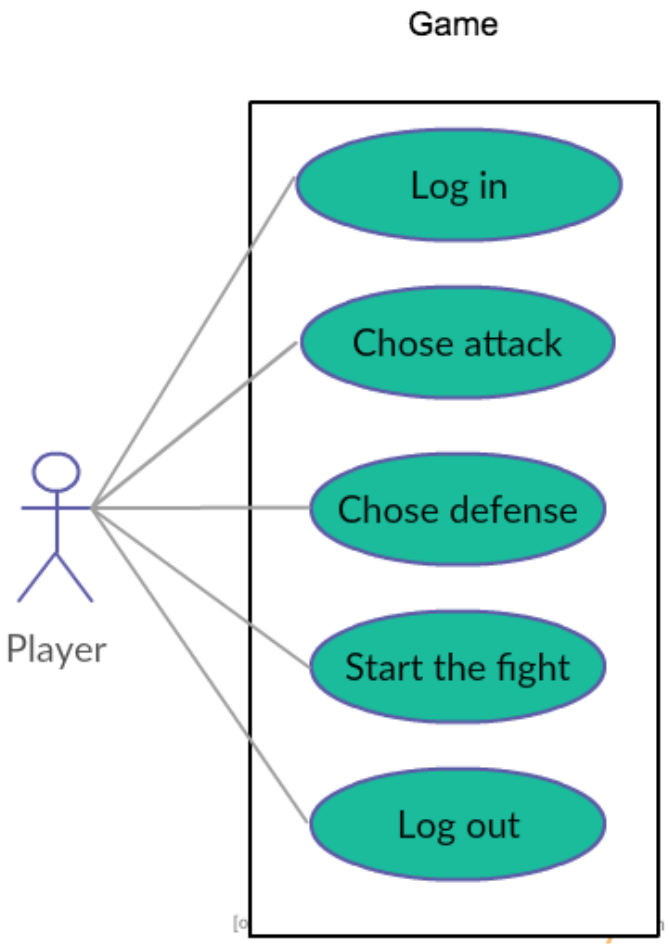

Fig . 2. Use case diagram

2.2.2. Server

The server is needed to create and maintain connection between players. Connection is created using Web-Sockets technology and Socket.IO library.

Also, it has 3 important functions: find opponent, send game info to players and show leaderboard.

Find opponent means that server matches the opponent for player. Search is based on player's stats: health points, attack damage and defense power.

Send gameinfo means that server calculates differences in players' stats before and after the move and sends new stats to players.

Show leaderboard means that server sends to client the list of best players.

2.3. Game process

Fig. 3 shows the sequence diagram of the project.

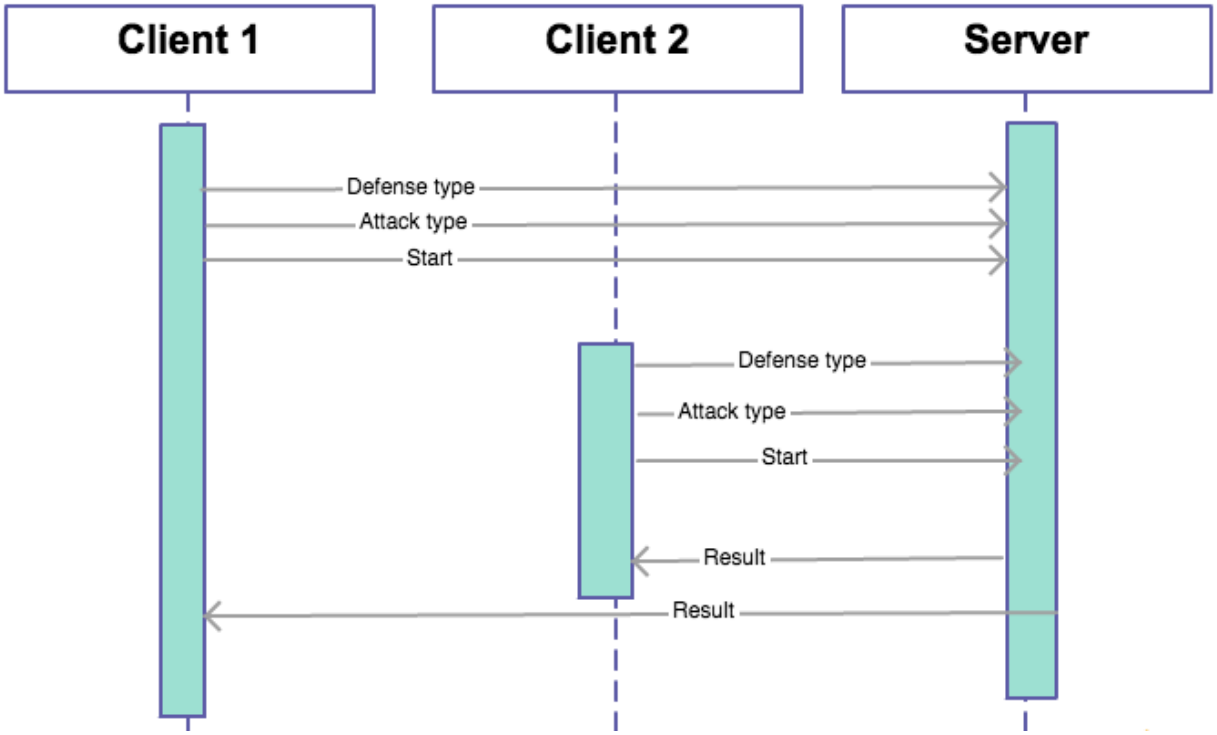

Fig. 3. Sequence diagram

At first Client 1 sends to the server information about attack type and defense type and the command to start fight. Then Client 2 does the same. After both clients made the move, Server calculates the changes in players' stats and sends new information to the Clients. Then the sequence repeats.

3. REALIZATION

3.1. Connection

Fig . 4 shows the example of request from browser to server for command `new WebSocket("ws://server.example.com/chat")`:

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: weblsocket
Connection: Upgrade
Origin: http://javascript.com
Sec-WebSocket-Key: Iv8io/9s+lYFgZWcXczP8Q==
Sec-WebSocket-Version: 13
```

Fig . 4. Browser request example

Header explanation:

– GET, Host – default HTTP headers from URL request;

– Upgrade, Connection shows that browser will work with sockets;

– Origin – protocol, domain and port of request sender;

– Sec-WebSocket-Key – random key generated by browser: 16 bytes in Base64 encoding;

– Sec-WebSocket-Version – protocol version. Current version: 13.

The sequence diagram of Web-Socket protocol is shown in (fig. 5.)

The first steps are::

– the client initializes the handshake;

– the server responses to the handshake.

Then client and server communicate sending each other messages.

The connection may be closed by any side when it is needed.

The sequence diagram which shown in fig. 5 its explain how the connection between the client and server. in this game when the client start play the game and start fight all the massages transfer by the web socket from the client to the server WebSockets is a good technology, depend on the ws protocol, that makes possible to create a continuous full-duplex connection stream between a client and a server.

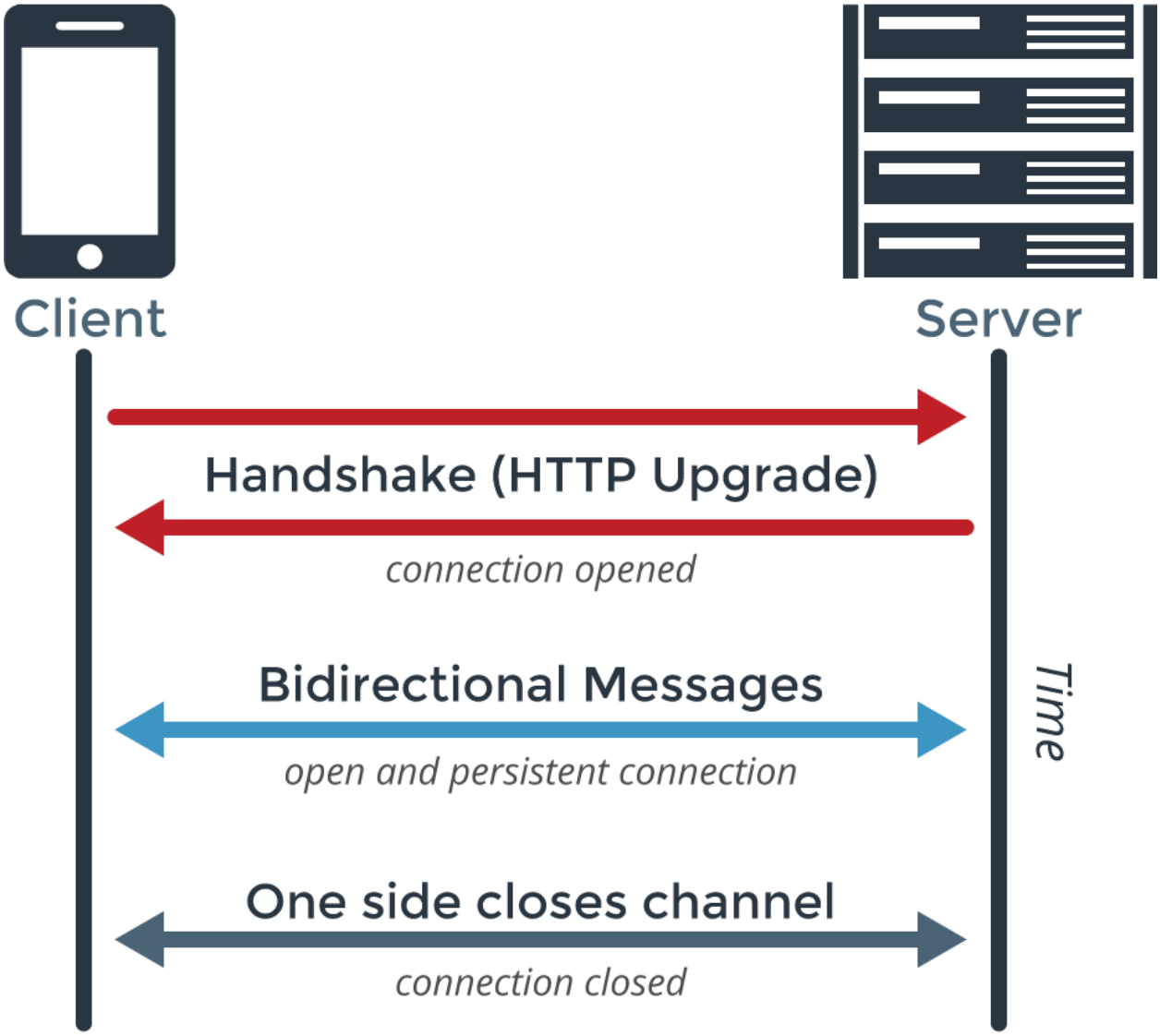

Fig. 5. Sequence diagram of Web-Socket protocol behavior

3.2. Client

Game client is a web page running in web-browser that supports HTML5 and JavaScript. Client consists of the following parts:

- welcome screen;
- game scene;
- user commands handler.

To start the game client must open the browser and write the URL of the game (at the current moment it's hosted on localhost) after that appear to the client welcome screen Welcome screen is shown in (fig. 6) It consists of game name and

“Start” button. its designed by simple and beautiful form to make game interesting and beautiful. after pressing “Start” button goes to game scence.

Fig. 6. Welcome screen

3.2.1. Game scene.

Game scene is created using HTML5 tag “canvas” that is used for drawing raster image using JavaScript. The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It is a low level, procedural model that updates a bitmap and does not have a built-in scene graph [18]. Game scene is shown in (fig. 7).

Game sence is created by HTML5 and javaScribt because these technology Best technologies to create 2D shapes and images and allow dynamic and interactive window so in this game we used these technologies for design welcome screen and other windows for the game.

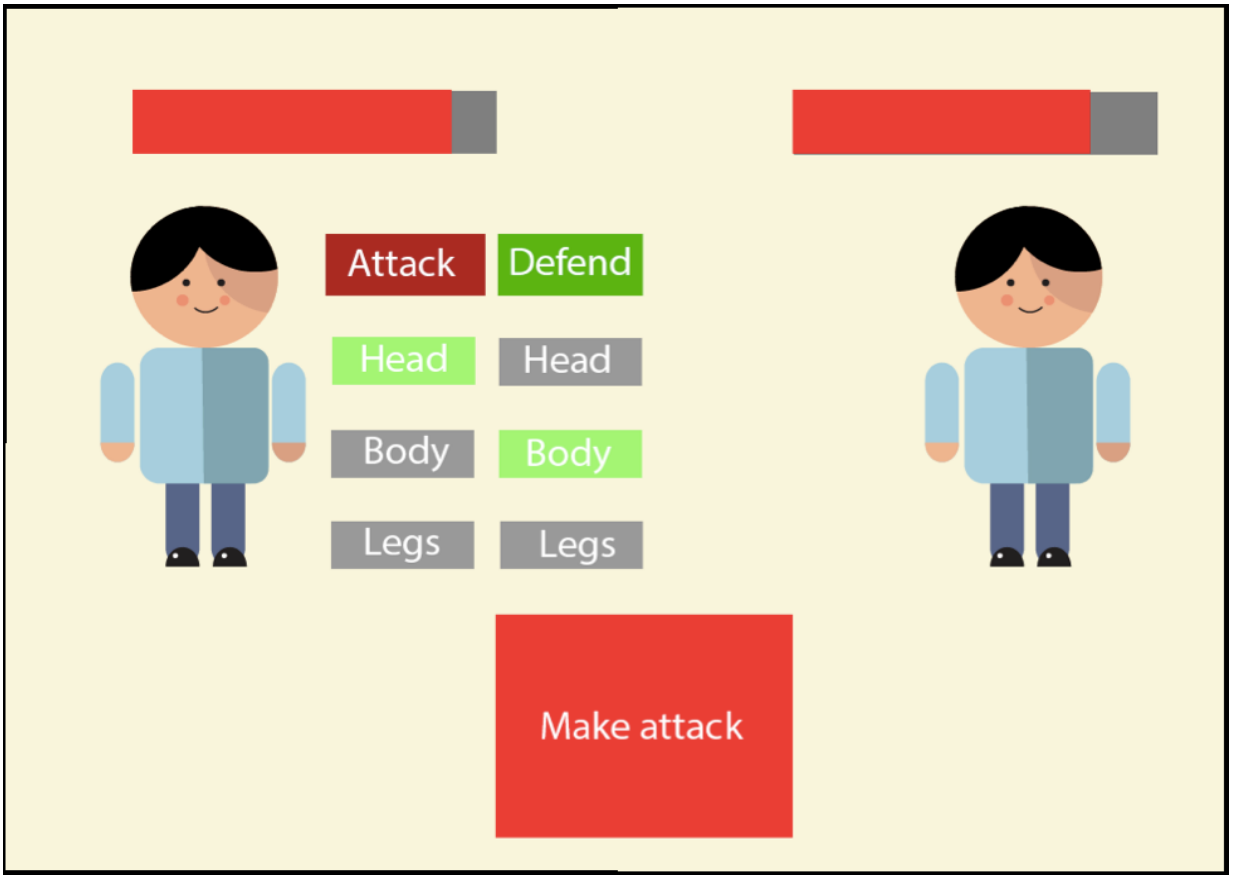

Fig. 7. Game scene

Canvas creates all game scene objects such as:

- players images;
- health bars;
- gameplay buttons.

Game scene objects are created by JavaScript function “drawCanvas()”

```
function drawCanvas() {
    if (canvas && canvas.getContext) {
        context = canvas.getContext('2d');
        drawBackground();
        drawPlayers();
        drawButtons();
    } else
        console.log("Canvas not found");
};
```

Fig 8. drawCanvas() function

Fig 9 demonstrates the source code of the client webpage.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<link            rel="stylesheet"            media="all"
type="text/css"href="style.css">
<script src="frontend.js"></script>
</head>
<body>
<div class='header'>
<h1 class="header-logo" draggable="true">Fight!</h1>
</div>
<div class='content-wrapper'>
<canvas id="mainCanvas">
</canvas>
</div>
</body>
</html>
```

Fig 9. Client source code

3.2.2. User commands handler

User commands handler is written in JavaScript language. It is used to handle user commands and send them to server.

Player inputs the commands using the game scene buttons:

- attack type;
- defense type;
- “Attack” button.

When user chooses attack and defense type buttons, user commands handler writes information to JSON file (fig. 10), which is sent to server when user presses “Attack” button.

We used JSON to sent messages between client and server because JSON supported by Java Script. JSON is a format for storing and transporting data

```
{ message:
            {
              class: “step”: {
    “attack”: “head”,
“defense”: “legs”
}
}
 timestamp: currentTimestamp
}
```

Fig. 10. JSON message from client example

After the “Attack button is pressed” by both players, the animation of fight begins (fig. 11).

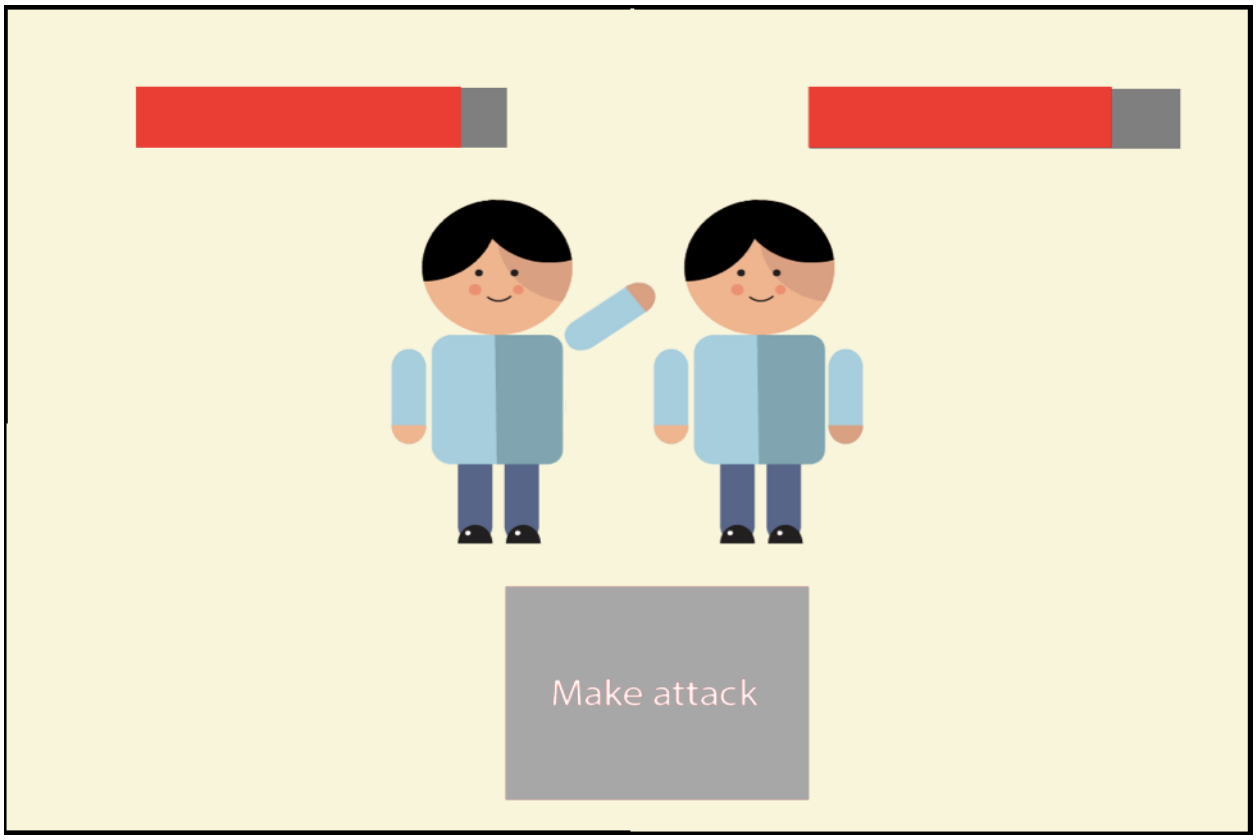

Fig. 11. Fight animation

3.3. Server

Server is a written using Node.JS. Server consists of two conceptual parts: first part (“Connector”) is responsible for establishing connection and sending information to users, and second part (“Statistics”) calculates changes of game

statistics. Information between server and client is sent by Web-Socket technology.

3.3.1. Connector

Main functions of “Connector” are:

- establish connection between server and client;
- find opponent for player;
- receive commands from player and redirect them to “Statistics” part;
- send updated game information to clients.

Web-Socket and Socket.IO are used to create fast asynchronous data exchange:

- to send commands from user to server;
- to send new statistics from server to user.

3.3.2. Statistics

“Statistics” is a part of server that calculates new information about players after each move. When “Connector” redirects user commands to “Statistics” (example of message is shown in (fig. 12)), “Statistics” starts calculation.

```
{ message:
            {
              firstID: player1ID: {
                    “attack”: “head”,
                    “defense”: “legs”
              }
              secondID: player2ID: {
                    “attack”: “legs”,
                    “defense”: “body”
              }
            }
 timestamp: currentTimestamp
}
```

Fig. 12. JSON message from “Connector” to “Statistics” example

If attack target of Player1 is coincides with defense target of Player2, Player2 gets no damage. If not, Player2 gets damage depending on which part was attacked.

After calculations “Statistics” generate JSON message with new information about user statistics and sends it to “Connector” to further send it to both players. Example of such message is shown in (fig. 13).

```
{ message:
          {
            firstID: player1ID: {
                  “hp” : “80”
            }
            secondID: player2ID: {
                  “hp” : “65”
            }
          }
}
```

Fig. 13. JSON message from “Statistics” to “Connector” example

After receiving of such message Client redraws players’ health bars to match values received from server.

3.4. Anti-Cheat System

There is a simple anti-cheat system implemented in the server.

Cheating in online games is defined as the action of pretending to obey the rules of the game, while secretly subverting them to gain advantage over an opponent.

In the developed project the player may try to gain the advantage by sending false attack commands to server from the web browser console or by changing his statistics (for example, health or attack damage).

To prevent it server checks if it’s now the current players turn (it’s impossible to make attack twice in a row) and calculates his statistics after every move.

4. EXPERIMENTS

To test the system that was developed we had to carry out the following experiments:

- test connection of client and server;
- test user interface;
- test that user commands are sent by client and received by server;
- test that server handles user commands and calculate new statistics correctly;
- test that new players' data is sent by server and received by client.

After performing all the experiments we determined that all developed functions and subsystems work correctly and accurate.

CONCLUSION

During this work we developed the Web-Socket based online browser game.

To get this goal, the following tasks have been solved:

- the review and analysis of existing solutions was conducted;
- the design of the software platform was created;
- the game components were developed;
- the tests and experiments on the game were made.

REFRENCE LIST

1. ASP.NET | The ASP.NET Site. [Electronic resource] URL: http://asp.net (the date of access:15.03.2015).
2. Batron M. Dungeons & Desktops. The History of Computer Role Playing Game . – USA: Wellesley, 2008. – 451 p.
3. Can I use json. [Electronic resource] URL: http://caniuse.com/#search=json (the date of access: 01.03.2016).
4. Can I use css. [Electronic resource] URL: http://caniuse.com/#cats=CSS(the date of access: 01.05.2015).
5. Can I use Web Sockets. [Electronic resource] URL:http://caniuse.com/#search=sockets (the date of access: 01.03.2016).
6. Castro E. HTML, XHTML, and CSS. – USA: Peachpit Press, 2006. – 456 p.
7. Cederholm D. Handcrafted CSS: More Bulletproof Web Design. – USA: NewRiders, 2009. – 240 p.
8. Crockford D. Java Script:The Good Part. – USA: O'Reilly, 2008. – 176 p.
9. Developer.mozilla.org. [Electronic resource] URL: https://developer.mozilla.org/en-US/Learn/JavaScript (the date of access : 20.03.2016).
10. Essential facts about the computer and video game industry. [Electronic resource] URL: http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf (the date of access: 15.04.2015).
11. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – Boston: Addison-Wesley Professional, 2013. – 208 p.
12. HTML tutorials point. [Electronic resource] URL: http://www.tutorialspoint.com/html/ (the date of access: 01.02.2016).
13. Java Server Pages. [Electronic resource] URL: http://java.sun.com/products/jsp (the date of access: 25.03.2016).
14. JSON.ORG. [Electronic resource] URL: http://www.json.org (the date of access: 25.03.2016).

15. Node.js. [Electronic resource] URL: https://nodejs.org/en/ (the date of access: 20.04.2015).

16. Node.jstutorial. [Electronic resource] URL: http://www.tutorialspoint.com/nodejs/ (the date of access: 01.04.2016).

17. PHP: Hypertext Preprocessor. [Electronic resource] URL: http://php.net (the date of access: 10.04.2016).

18. Pilgrim M. HTML5: Up and Running. – USA: O'Reilly Media, 2010. – 222 p.

19. Scala website. [Electronic resource] URL: http://www.scalalang.org/ (the date of access: 16.03.2016).

20. Socket.IO. [Electronic resource] URL: http://socket.io/ (the date of access:25.04.2015).

21. w3schools.com. [Electronic resource] URL: http://www.w3schools.com/ (the date of access: 020.03.2016).