On the development of a Multi-Platform Football Game Utilizing WebGL and Three.js

Mohammad Mahdi Mohebbian
Department of Computer Engineering
University of Science and Culture
Tehran, Iran
m.mohebbian@usc.ac.ir

Reza Shahinpour
Department of Computer Engineering
University of Science and Culture
Tehran, Iran
reza.sh9170@gmail.com

Shahabeddin Asgari
Department of Computer Engineering
University of Science and Culture
Tehran, Iran
askari.shahab@gmail.com

Seyyed Amir Hadi Minoofam
Department of Electrical and Computer Engineering
Nazar Abad Center
Islamic Azad University
Alborz, Iran
minoofam@naiau.ac.ir

Abstract—Browser-based games are played directly in web browsers, and naturally, they do not need any additional software or plug-in installations. This feature is one of the primary advantages of online and browser-based games to the offline games which one could install on his/her target platform. There are countless challenges that cause lots of game programmers to avoid online games such as, the variety of web browsers and their diversity in the Internet speed and bandwidth, and yet the game must be usable for every user with any browser using different bandwidth. There are vast game engines that can design online games; but the only drawback is due to their complicated structures, yields the results not always be that light to capable on loading simply on various computers. Thus, in this paper, we designed a football game using the basic technologies of game engines such as WebGL, HTML5, and Three.js to avoid complex structures and achieve advanced speed and convenience for different users with different equipment. Also, we demonstrate the structures and methods of collision detection in the way of our development. In the end, the resulting game is so light yet absorbing that can be run on any web browsers with the most basic features.

Keywords—WebGL, HTML5, Browser-based applications, Online games, Collision detection

I. INTRODUCTION

Video games and consoles can be effective on people behaviors, intelligent level, and their creativity. Nowadays there are multiple platforms for gaming; we have consoles such as XBOX 360 and Play Station 3. Furthermore, we got online games and stand-alone games. We’ll now discuss the advantages of online games to stand alone games, and we would demonstrate the different modules and sections are forming an online game in the following.

The browser-based online game, compared to the traditional stand-alone game client, is getting more and more popular among people, as it requires no explicit installation, no plug-in implementations and can achieve cross-platform so easy, cause the game is not tied down to a certain platform or operating system. Browser-based app systems used to be just 2D graphics or flash based. However, HTML5 will change this deal. HTML5 is the latest revision of the HTML standard, a language for structuring and presenting content for the World Wide Web [1, 2, 3]. To study the performance and feasibility of browser-based online games with HTML5, we designed and implemented a framework for such game system using WebGL technology.

The remainder sections of this paper are organized as follows. We will introduce the technical background of our framework and introduce the tools in Section 2, including the technologies of HTML5 and WebGL. In the next section, we would discuss some related projects and papers linked to browser-based projects and WebGL technology. The architecture of our framework will be illustrated in details in Section 4 which includes game scheme and UML (Unified Modeling Language) diagrams. The implementation is following in Section 5. Finally, the paper concludes in Section 6 with future directions of this work.

II. BACKGROUND

Our framework for browser-based online games is designed with HTLM5, using the technologies of WebGL.

A. HTML5

HTML5 is a new engine for HTML language, which is for structuring and presenting content in the World Wide Web, the core technology of the Internet. HTML5’s core aims have been to improve the language to support multimedia and graphics better, reduce web developers’ workloads and keep it easily readable by humans. In particular, the new features added into HTML5, including the <video>, <audio>, <canvas> elements, and so on [3, 4].

In this study, we are going to use the <canvas> only, which prepares the specific region for the implementation of WebGL codes.

B. WEBGL

WebGL (Web-based Graphics Library) is a graphics related software library that is designed to integrate the extended JavaScript programming language with OpenGL ES (OpenGL for Embedded Systems) to generate accelerated 3D graphics within any compatible web browsers [1, 4]. WebGL provides 3D computer graphics APIs for the HTML5 canvas without any browser plug-ins. This action is useful to develop 3D web
pages with complex structures, especially 3D browser-based online games [3, 5].

C. THREE JS

Three.js was first released by Ricardo Cabello to GitHub in April 2010 [6, 7]. Three.js allows the creation of GPU-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins. This achievement is possible thanks to the advent of WebGL. The code was first developed in ActionScript, then in 2009 ported to JavaScript. With the advent of WebGL, Paul Brunt was able to add the renderer for this quite easily as Three.js was designed with the rendering code as a module rather than in the core itself [8, 9].

III. RELATED WORKS

Bijin and Xu have developed an online platform for multiplayer online games and evaluated the feasibility and efficiency of the designed platform. This platform consists of three primary parts, the Client level, the server level, and game engine. In the mentioned online game all of the players will connect to the server and become a live player in the game compete and try to eliminate other players to win the game [10].

Yikang, Xinging and Lei have used WebGL for simulating the outer space environment. In this simulated model, the user would be able to change the texture of the available objects in space and customize the amount of transparency and refugence [11].

So far, various web elements and web environments have contributed to Web GIS as alternative technologies of existing GIS technologies. Maps and services such as google maps and google earth play role as the main component of web GIS. Woo Kim and Lee developed a platform for visualizing graphical information and services such as Google maps using WebGL technology to confront the limitation of the existing methods. As most of the current GEO apps still handle raster data as an image and this causes limitation of expressiveness because delicate expression of the raster on Google Earth is impossible. But if we use WebGL as a client, the huge size of raster data causes overload during execution procedure. So the developers came up with the solution of using multi-thread programming to execute and generate the graphical information on separate cores [12].

Physics was one of the most important aspects of the game, and it still is. An enormous number of researchers are studying the development of physics in games. Yogya and Kosala integrated three different open source platforms based on WebGL for physics implementation on games. They have evaluated platform’s accuracy, efficiency, integrity and the compatibility of these integrated platforms toward each other; the result is referenced as the best platform available for physics implementations [13].

Moloo and his co-researchers developed a 3D tour model for the Mauritius University using WebGL technology so that people and students would be able to reconnoiter around the university and its environment in a simulated, yet accurate model. There are two major problems with this developed online application, the first one is the dullness of loading object’s textures, and the other one is the absence of a logical strategy for physical interaction between camera and all other objects [14].

Norm Badler and his student, the University of Pennsylvania, were also working on a Massively Multiplayer Online Game (MMOG), using WebGL game Engine. They believe that MMOG is a perfect way to escape from the routine life and enjoy a wonderful experience of online competition with other peoples all around the world. They demonstrate their tools as WebGL and Web Socket. They have designed the 3D world and characters and used Web Socket to connect players from different locations and muster them in the game. The main goal of this game for each online player is to try to eliminate others and promote its rank among other players. Badler quoted that he is not going to design the next Warcraft, but their code base would be a perfect start for people who would love to develop MMOG [15].

Raghu Raman, Unnikrishnan KV, and Smrithi Rekha V among lots of other researchers are developing a simulated online laboratory. Online Labs (OLab) are revolutionizing education by offering access to content anytime and from any place. OLabs have a real deep impact on everyone in the science field including professors and also students. OLabs often contain contents such as animations, Audio, Videos, simulations and textual information. Current content in OLabs is the 2D simulated environment. 2D productions come with its limitations of low accuracy and low realism. But in this case, due to the presence of WebGL, the 2D objects have the rendering potential of a 3D object without the need of any plug-ins or installs on the user end. The simulated OLAP in this project is a 3D model, a convex lens experiment in OLabs physics using WebGL and dynamic cube mapping [16].

IV. ARCHITECTURE

In this section, we would discuss the different parts and divisions of our development. In Fig.1 we demonstrate altered parts of the project and the association among them using UML Class diagram. As one can see in the described UML, the game consists of many core components such as players and ball, etc. The game is running in several different states which would change as any particular event happens (e.g. if any of the players scores the game would enter the “scoring” state) in fact any significant change in the game routine would affect component states.

This game is a two person game, and each player has a different interface, but they have some components and basic concepts in common, so we would have to inherit each player from a more abstract class. The second player has an avatar in the game so it is obvious that the avatar model would be in the composition of the 2nd player class. In the next section, we would be representing the implementation phase of the game.
Fig. 1. The UML Class diagram of the football game

Fig. 2. Entrance procedural diagram

Fig. 3. Football game routine procedural diagram
For a demonstration of the game implementation, we should exhibit two different matters, the logical implementation, and the graphical implementation.

- Logical Implementation

The logical implementation would show the game routine and methods and the way it works in general. In Fig. 2, we would show the behavior and the main route of the game from the very beginning. Fig. 2 would be regarding the entrance and preface of the game which would include signing up or logging into the website to prepare for the game. As one can see in Fig. 2 after each of the players logged into the system, they should choose their side. Each player has its interface; the “shooter” should use the mouse, and the “goalkeeper” would be using keyboard interface.

In the next step, the primary procedure of the game would begin, and players start to compete and exchange the ball among each other and try for gaining more scores.

The crucial purpose of this game is to have two players compete with the only help of an ordinary Internet connection. Each player must have a factor for score comparison so as to determine the winner. We have shown in Fig. 3 that each player tries to gain more score to achieve victory. The winner would be chosen by calculating the combination of each player scores, which contains both numbers of goals s/he achieved and a number of balls s/he caught. According to Fig. 3, each player tries to score a goal by shooting the ball toward the opponent, and when the opponent takes the shot, s/he tries to catch the ball to increase her/his total score. All important phases in the game is illustrated in Fig.3 such as, the restarting points in which the game restarts from the beginning. The official starting point is when the 1st player clicks on the left button and start the ball rolling toward the goalkeeper, each time that any player scores, the game would restart with new scores. It is obvious that not every score receive would restart the game, the game would not be a restart for some saves score, and this number would be calculated during the game.

There should be some algorithms and procedures to implement that we would enucleate in the following, to dig a little deeper in the explained routine.

One of the main and critical processes is collision algorithms for a goalkeeper and shooter.

We cannot use the same collision algorithm for both of players because of the difference in interface levels (e.g. goalkeeper has an avatar while shooter only uses the mouse for playing).

The collision algorithm for a goalkeeper is explained as Fig. 4.

When it is time for the goalkeeper to dig a little deeper in the explained routine, catching the ball, the collision monitoring algorithm will start looking for ball collision, if the ball reaches the goalkeeper depth, which is the first condition in the code in Fig. 4.

This algorithm does not only check the collision of ball and goalkeeper, but it also observes that the incoming ball fits in the goal frame.

```
Input: goalkeeper, ball, shooter
Output: ball speed
begin
if (ball depth reaches the goalkeeper depth)
    if (ball position fits in goal frame)
        if (distance( ball.x, keeper.x ) > e)
            shooter.goal++
            reset keeper position
            reset ball position
        end if
    end if
if (distance( ball.x, keeper.x ) < e)
    keeper.saves++
    change ball direction response
    increase ball direction range.x
    increase ball rotation speed
    increase ball transition speed
end if
return ball speed
end if
```

When the ball fits in the goal frame and reaches to goalkeeper depth and goalkeeper fails to catch it, it would be a score for the shooter. Otherwise, it would be a save score for goalkeeper. Hence, as to determine that a ball would cross the goalkeeper or would be saved, the game would calculate the absolute difference between ball and goalkeeper distance. If the measured distance exceeds a certain number (e), that means the goalkeeper received a goal, otherwise, means the goalkeeper caught the ball. The collision algorithm and procedure for the 1st player (shooter) is quite different because of the alteration in interfaces.

First player’s interface was a mouse so the collision technique would have to interact with mouse click event. In the following figure, we demonstrate the mouse collision process in the form of pseudocode. As appearing in Fig. 5, the process for the mouse is completely different from goalkeeper avatar. In the goal keeper’s case, collision area would be compared and looked up in the border of keeper avatar; the 1st player is on the other side. According to the specified solution for mouse collision, the clicked coordinate would be scaled in the inner windows size which depends on the browser status. This action would make the game adaptable in any Internet browsers. In the next step, the newly defined coordinate would be compared with ball position. There are some similarities between these collision algorithms, for instance, both of these algorithms executes and start their monitoring techniques if the ball exceeds a certain depth toward them. That is why the first condition in both pseudocodes is the depth checking condition. The 1st player must reject the ball and send it back to the goalkeeper before it gets too close. Otherwise, it would be a score for goalkeeper, so the next thing that should be checked is the distance of the ball from base depth point of the scene. However, if the shooter clicks on the mouse, it means that he is trying to catch the ball, so now the clicked coordinates should be compared to the ball position to determine if the ball is saved or not.
One of the advantages of this game is its lightness and simplicity in graphical implementation which allows the product to run on any device and any web browsers.

The objects in this game were built by a combination of basic and common objects, except for the goalkeeper who is built using only a .json file. All surfaces in this game have textured which makes the game more attractive and buzzing. The only surface that looks difficult to be textured is the ball because of its spherical shape. Fortunately, Three.js has better methods and routines for applying textures on objects which are much simpler than OpenGL. We tried to make the user interface both striking and simple at the same time, so the scoreboard in the Fig. 6 is not an HTML element, but it is a built component which is rendered in the game. There is a volume bar at the top right of the screen to change game sound volume, such as viewer’s sound, shooting sound, etc. Fig. 6 demonstrates a screenshot of the football game. Also, the full version of the game has been uploaded to following URL: http://geekylife.ir/hittheball.

Fig. 5.  pseudocode for mouse click collision (the 1st player)

- Graphical Implementation

As we all know the most favorite category of games is online games, that’s because of the ease of access to these games with no plug-ins and installations. The advantage of our development is that the most basic technologies of great game engines are used so that we can control the game’s complexity and keep its structure simple and easy to load. Moreover, this game would have two players, and two users can compete using an ordinary Internet connection in one the most beloved sports of all times with simple yet eye-catching graphical backgrounds.

We will further add WebSocket to our architecture to connect players from different locations for competition and place them in a global scoreboard with different records.

REFERENCES