2014

Report on the Development of a Web Application

MODULE: 5CC503

JOSH HADLEY: ST100240235

Contents

Introduction

The following report will outline the development of the web application I have created over the past few weeks. I will explain what my application is, the technologies I have used to create it, how these technologies have been used correctly and what I have done outside of developing my application to create it

Description of Application

My initial design for my application was to be a WebGL Card Game. However throughout the development, I decided to alter to the project slightly by turning it into a reaction game instead of a card game. The reason for this change is because I saw little value in creating a sole card game, but with a game based on the user's reactions I believed it would provide a more productive output, as well as give more data interaction.

My application to be clear is now a WebGL-based viewer that works alongside user interaction. Through the user's interaction, information is collected and stored in a database which is then shown through a chart on an appropriate page.

A step-by-step guide on how to use the application is as follows:

1. Access the web application through the url: http://cardgame.azurewebsites.net
2. Click the 'New User' button to register a new account.
3. Follow the instructions on the screen to perform the game. Note: Pressing A / D Keys will change the rotation speed of the game.
4. To Access the Leader boards or Statistics Page, click the relevant pages throughout the application.

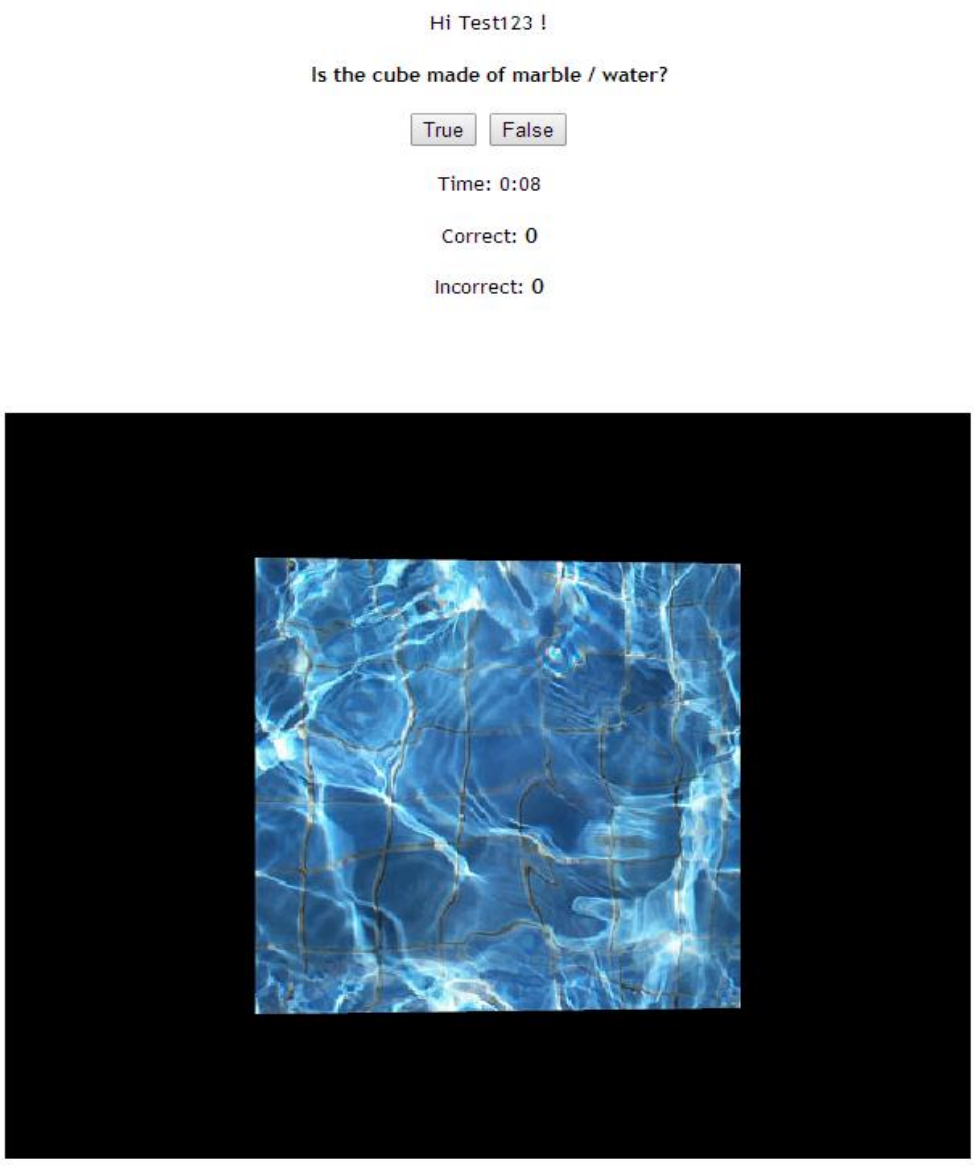

Figure 1.0 Screenshot of 'Reaction Game' Web Application

The Application has been tested and built through Google Chrome. Compatibility for other browsers should be available, however some browsers may not support the use of WebGL. If compatibility issues arise through WebGL, please make sure to update your browser.

Technologies Used

In the development of my application, I have used various technologies in order to create and fulfil the purpose of my application. A list of the technologies I have used are as follows:

1. ASP.NET
2. WebGL
3. JavaScript

ASP.NET

The core of my application has been developed through the use of the ASP.NET Web Pages. The reason I choose to use ASP.NET was through my own experience of using C# which I felt comfortable with, but also the fact I preferred the more easily suited object-orientated programming which would come in useful later on. Because of this the majority of my web pages have been built using the .cshtml format

The Drawback with using ASP.NET however is that despite being excellently tailored for server-side programming, there was little to no client side interaction being offered through the use of C#. However the use of Razor C# enabled my web application to be easily used in combination with scripting languages.

JavaScript

Apart from the use of .cshtml in the ASP.NET Framework, I have used JavaScript not only inside of my web pages, but also outside as pure JavaScript. The reason I choose to use JavaScript was mainly due to it being needed with WebGL in order to function correctly, but also because I have some experience with Java. Despite the similarities between Java and JavaScript today being quite different. I felt it was a good choice to make due to the flexibility of the functions I could perform with the client side aspect of my application.

However due to my JavaScript being used for client-side functionality, this created a huge security drawback by the fact that some of my JavaScript would be called when the page would be loaded. In order to counteract this issue, I ensured that the page that was responsible for the JavaScript that was needed on load was stored directly in a specific Layout page that has been hidden from the users and is unable to be browsed to.

WebGL

My goal from the start of the development of my application was to use WebGL. From my experience with the use of OpenGL, the use of WebGL was very similar in terms of implementing and allowed me to set up the game-side of my application. Through the use of JavaScript in combination with tracking the user's interaction, I was able to effectively use WebGL to the effect I aimed for it to have in my application.

In order to implement WebGL efficiently, I used two external libraries to help run my WebGL calculations. I have used glMatrix.js (glMatrix, 2014) to help with matrix and vector calculations, and webgl-utils.js (Giles, 2014) to provide additional functionality to some of my WebGL code.

The only drawback with the use of WebGL was that it had to be entirely written in JavaScript. This again caused issues mainly due to not being able to directly interact with Razor C# but also that the WebGL would be running on the client-side.

How Best Practise has been applied

Due to the application utilizing multiple HTML 5 technologies and interacting with a database containing data that would be needed constantly, it is important that I adhered to applying the best practises I could throughout my web application development. In order to make sure I used best practise, I used ASP.NET's overview of web development (Microsoft, 2014b) and JavaScript best practises (Kruse, 2014).

Consistency

In order to keep a professional looking web application not only on screen but also with my code, I made sure that my code followed the relative conventions that applied.

Firstly, I ensured that all of my files were stored correctly in relevant folders to maintain an organised data structure. Secondly, any web pages that were used for creating a HTML layout were correctly named with a '_' in front to prevent the page from being browsed to in a browser.

Inside my code, I made sure to keep all of my naming conventions accurate. As well as this, I ensured that all of my html tags have been used correctly and that an organised and simple layout has been achieved across the website.

Performance

To ensure that my web application maintains good performance, I have strictly followed the JavaScript conventions that were given to me.

I have ensured that the use of 'var' throughout my JavaScript has been accurate in order to ensure my scripts run without issues. As well as this, I have made sure that I also control elements by finding their IDs rather than directly calling them.

My .cshtml pages contain very little session variables which restricts the data usage throughout the application. Through the use of forms, I have ensured that any vital pieces of data have not been made accessible by the public and that all of my values have been encoded using the Razor '@' syntax to prevent any code being input

Security

In my forms, I have ensured that there is a sufficient level of validation that ensures accurate data is sent to and from my database. Any data that is passed from page to page is stored as Session variables and are not passed through the URL.

Any scripts that have been used 'onload' have been made sure they cannot be accessed mainly through the use of Layout pages that cannot be browsed to. No database connections are left open unless needed.

Particularly with scripts, I have made sure that if any require to look at a global variable stored in a page for instance that they are using 'Feature-detect' and not 'Browser-detect'. This ensures that the script is only run if the correct item is present in the web-page and not the browser itself.

UserObject Web Service

Throughout my web application, I have used the functionality of Razor C# in conjunction with forms to help send data from one page to another. However a specific web service I have used is one I have created which is called “UserObject”.

The web service is built on SOAP principles of reproducing a ‘User’ Object in my web application which will store various pieces of information inside of the .cshtml page for the game. A snippet of my web service can be seen below:

```
@functions {
    public class User {
        public int PlayerID {get;    set;}
        public string Username {get;    set;}
        public string Password {get;    set;}
        public int  Correct{get; set;}
        public int  Incorrect{get; set;}
    }

    public static User GetUser(int ID) {
        var db = Database.Open("Card Game");
        var selectQueryString = "SELECT Username, Password, ID FROM Accounts WHERE
ID =" + @ID;
        var data = db.Query(selectQueryString);
        User user = new User();
        foreach(var row in data)
        {
            user.Username = @row.Username;
            user.Password = @row.Password;
            user.PlayerID = @row.ID;
        }

        return user;
    }
}
```

Figure 2.0 Razor C# Code of ‘UserObject’ Web Service

With the web service, I am easily able to collect a user data from both of my tables. Despite not having an instance where my web service does not have both tables feeding information, the web service in conjunction with the various other functions throughout the web site i.e. Forms, Razor# in-line functions, the web service allows the managing of the user in the game easily.

As well as this service, in order to generate my statistics charts, I use a .specific cshtml page that uses only Razor C#. It specifically renders the chart constantly from the database table ‘Leader boards’ and presents it on the web page ‘Statistics’. The chart function uses ‘ASP.NET Web Page Helper’ that allows components to be accessed with single lines of Razor code. (Outercurve Foundation, 2014)

Independent Learning and Research

In order to achieve the best possible application for what I set out to do, I had to perform a numerous amount of independent research and research in order to create it.

WebGL

In order to actually use WebGL, despite my knowledge of OpenGL I still needed to understand and learn what actually WebGL is and how to use it.

Using a series of web tutorials by Giles (2014), I quickly taught myself how to use and display my WebGL code on a web page. Following the tutorials and taking specific elements, I began to use my own knowledge and began to implement other features such as a countdown timer, functions that accessed data stored on the server and other input handling events.

If I had not researched or learnt from the tutorials before implementing them into the web site, I believe I would have been unable to produce a working result that I was both happy and pleased with.

CSS and Layout Pages

As my application grew in size, it began to dawn on me that my application's presentation was very unprofessional as the pages were very inconsistent and sporadic. I began to look into alternatives and soon realised that I could use a combination of CSS and Layout pages to make a 'house style' for my application.

Using a series of web tutorials on how to create a basic ASP.NET web application by Microsoft (2014a), I quickly understood how and what exactly CSS files are and how they are implemented. I create two CSS pages that would be used for different purposes. The first one is responsible for the global house style of the website, whereas the second CSS file was used to make my WebGL canvas look more presentable in the web page.

As well as this, I also used a series of Layout pages that were used to maintain a basic style of the website and to create sections using tags. With these sections I was able to customise the look and feel of my web application much more easily and made sure it was at a professional standard.

Chart Data

As my application moved into the final stages, I wanted to explore other avenues that would increase the complexity and usefulness of my website. Already possessing a 'Leaderboard' table, I began to come up with the idea of also including a statistics page where the total results would be placed in charts which could be analysed.

Having no idea on how or what to do in order to achieve this goal, I decided to explore the ASP.NET tutorial website until I found a list of tutorials that told me the options I had when it came to displaying data. (Microsoft, 2014c)

Spending some time reading through and trying the tutorials, I implemented two charts into my web application. The charts themselves would be drawn based on the data they received which would be constantly updated from the database.

Because of this addition, I felt that my application now offers not just an opportunity to relax and mess around with a simple game based on reactions, but now it can also provide insight into the bigger picture of what is going on statistics wise which could be useful to other projects.

Conclusion

In conclusion, I have enjoyed my time developing my web application. I am impressed by the application I have been able to produce and I am happy I was able to explore the areas I wanted to. I feel my application meets the requirements given to myself in the specification for the module, as well as looking further into more interesting and complex features.

If I had to pick my one weakness with my application, it would be the scale of the WebGL component. I am happy with what I have used however I feel it doesn't offer enough variety to be considered as a complex piece. If I could make a few changes, I would add a bit more interaction to my WebGL component be either adding more options to the game itself, or by possibly creating more instances where I could perform further tests that differ from the original.

References

Giles, T. 2014*. The Lessons | Learning WebGL*. [ONLINE] Available at: http://learningwebgl.com/blog/?page_id=1217. [Accessed 1st May 2014]

glMatrix. 2014. *glMatrix.* [ONLINE] Available at: http://glmatrix.net/. [Accessed 01 May 2014].

Kruse, M. 2014. *Javascript Best Practices*. [ONLINE] Available at: http://www.javascripttoolbox.com/bestpractices/. [Accessed 1st May 2014].

Microsoft. 2014a*. Introducing ASP.NET Web Pages 2 : The Official Microsoft ASP.NET Site*. [ONLINE] Available at: http://www.asp.net/web-pages/tutorials/introducing-aspnet-web-pages-2. [Accessed 1st May 2014].

Microsoft. 2014b. *Web Development Best Practices : The Official Microsoft ASP.NET Site*. [ONLINE] Available at: http://www.asp.net/aspnet/overview/web-development-best-practices. [Accessed 1st May 2014].

Microsoft. 2014c. *Displaying Data in a Chart : The Official Microsoft ASP.NET Site.* [ONLINE] Available at: http://www.asp.net/web-pages/tutorials/data/7-displaying-data-in-a-chart. [Accessed 1st May 2014].

Outercurve Foundation. 2014. *NuGet Gallery | ASP.NET Web Helpers Library 2.1.20710.2*. [ONLINE] Available at:https://www.nuget.org/packages/microsoft-web-helpers/2.1.20710.2. [Accessed 1st May 2014].

Trask, J. 2014. *webgl-utils.js - webglsamples - Various WebGL Samples - Google Project Hosting .* [ONLINE] Available at: https://code.google.com/p/webglsamples/source/browse/book/webgl-utils.js?r=41401f8a69b1f8d32c6863ac8c1953c8e1e8eba0 . [Accessed 1st May 2014].