The Organ Web App

Beici Liang

Supervisor: Gyorgy Fazekas

School of Electronic Engineering and Computer Science
Media and Arts Technology Programme
Queen Mary University of London

Project Report August 2015

Abstract

With the development of web technology, web applications, or web apps for short, have had a profound effect on the way people get informations and communicate with one another. As web apps can be developed to have similar functionality to the native apps and accessed on a range of different platforms, recent trends have shown much interest in the development of web apps, which are normally built using HTML, CSS and JavaScript. This report presents the development of The Organ Web App which aims to share the historically important heritage of the Henry Willis pipe organ with as wide an audience as possible.

In particular, this report presents the findings of a five months advanced project placement on the study of organ, undertaken at the Union Chapel. A web app that introduces different aspects of the organ is presented, which also features a small collection of audio samples from the organ in order to let the user press the keyboard and hear different timbres. The app can be used in a creative interactive way to make the Henry Willis organ more accessible to the local community and beyond, especially for people who have never been to the Union Chapel.

This report outlines the following: how digital technologies have engendered new music application; notices for web application design; introductions of the main web technologies applied in the app development; a detailed explanation of the implementation process. The app is evaluated by online questionnaire: positive and negative aspects of the design are identified. The Organ Web App as a whole serves as a tool to provide users with a comprehensive exploration of the Henry Willis pipe organ.

Acknowledgements

Firstly I would like to thank Claire Singer, Janet Gilbert, Les Mommsen and Andy Gardner at the Union Chapel for their enthusiasm and warm help for granting me get access to the Henry Willis pipe organ, recording the audio, filming the hydraulics and providing me with so many archives for the app. Also thank James Richardson-Jones at the Duplex Pipe Organ and Blower Company Ltd. for his professional explanation of the hydraulic engines.

From Queen Mary I would like to thank Gyorgy Fazekas for his invaluable support over the last five months, without him this project would not have been possible. Mark Sandler has been a constant support for his guidance and suggestions. I would also like to thank Jingwen Bai for teaching me so much in programming and photography.

Finally I would like to thank all those who are being so encouraging and supportive of this research.

This research was funded by the Chinese Scholarship Government (CSC) as part of the Centre for Doctoral Training in Media and Arts Technology at Queen Mary University of London.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

This project is a five months advanced placement as part of the Organ Project undertaken at the Union Chapel in the summer of 2015. The main task of the project is to reach out beyond the chapel's live audiences by creating an app on a chosen platform that features a small collection of audio samples from the Henry Willis pipe organ, which can be used in a creative interactive way to the user.

The organ at Union Chapel was designed and built specially for the size and acoustics of the new chapel building in 1877 by master organ builder Henry "Father" Willis. It is undoubtedly one of the finest in the world. It is one of just two organs left in the United Kingdom, and the only one in England, with a fully working original hydraulic (water powered) blowing system, which can be used as an alternative to the electric blowers. The organ is deliberately hidden away behind ornate screens under the rose window as shown in Figure 1.1 to make the congregation avoid being distracted by the sight of an organ or organist so as to let the music itself to be the focus during worship, which itself actually hints at the organ's importance, with

Figure 1.1: The organ at the Union Chapel.

its depiction of eight angels all playing different musical instruments.

As the Organ Project aims not only to honour the legacy of this very special instrument by keeping up a regular organ recital diary and having it performed in styles for the live audiences, but also to make it more accessible beyond the local community. Also since now HTML5 can embed audio and video content directly into the web page, which brings more possibilities for the web multimedia technology, typically leading to expand audiences having access to more diverse options. The investigation presented in this report outlines the development of a web application that is designed as an exploratory tool to provide the user with information about different parts of the Henry Willis pipe organ, as well as to play the organ with six different timbres using computer keyboard and mouse.

1.2 Research questions

When researching on how to make the Henry Willis pipe organ more accessible not only for the live audiences, but also for the people who don't have access to the Union Chapel, I was faced with the following research questions:

1. What are the platform we can develop the app on? Web app or native app?

2. How can we design the interface based on a chosen platform?

3. What are the implementation steps that should be taken to build the app in order to enhance the accessibility of the organ?

1.3 Summary of the structure of the report

Firstly the report begins with a discussion about how digital technologies take effect on music applications. I shall contextualise the work by looking at how the ways of making, sharing, teaching and learning music has been changed through the digital technologies. My intention is to highlight the importance of web application design and web technology. With a particular focus on the web application I shall then develop a framework for thinking about the development challenges of this project.

Following this is a section which provides the detailed design and implementation process, including any requirement the app need to meet, any specification compared with the native app, and implementation guidelines, highlighting the main API and libraries applied in the app.

The next section presents the evaluation of the app by online questionnaire. Strong and weak aspects of the usability are identified, as well as aspects that would benefit from user evaluation.

The final section presents the future work and conclusion. All in all, the report aims to demonstrate how versatile the organ can be through the web-based app - that it is not simply a church instrument, but in fact the world?s first synthesiser which can be incorporated into most genres of music.

Chapter 2

Related work and theoretical context

2.1 Digital technologies on music applications

With the rapid development of digital technologies, the way of making, sharing, teaching and learning music has been fundamentally changed in the last decade [9]. This section will present a brief overview of the impacts of digital innovations, introduce the functions of new music applications, and determine their possible uses for music learning context.

There are many ways of contributing to create new mode of music production, music promotion and music learning. Especially with the rise of the participatory Internet, known as Web 2.0, and the popularisation of portable digital devices such as laptops, smart phones and tablets, a big amount of digital applications related to music have emerged in order to build a broader music learning environment. Due to the participatory websites such as YouTube, Vimeo and Facebook, people's access to music has changed, which blurs the boundary between creator and learner as users of all ages are able to post their own music and learn from others. Therefore

music learning enabled with Web 2.0 is often recognised as learner initiated, learner created, learner directed, and learner distributed [10]. Along with the development of digital technologies, the music innovations will continue to redefine the concepts of music making, sharing, teaching and learning.

Music software programmes operated on digital devices, referred to as "apps" (applications), evoke new ways to music practice and develop all the time. Commonly the music apps can be categorised as the following four kinds [9]:

1. music education tools with scores and lessons provided;
2. music toys and games;
3. music tools that can be used to tune the instruments, as well as record and edit pieces;
4. virtual music instruments.

In every platform, there are applications as combined with new digital technologies that provide creative ways for turning music learning into interesting settings. In addition, these applications can be learned straightforwardly and easy to use. According to the above classification, the following music apps are introduced as examples. As a matter of fact, numerous applications can even cover all the functions mentioned in the four categories.

The Orchestra [2] is an iPad app that provide enable users to experience classical music with features including video, a synchronised core and comment by the conductor. Users can also understand how instruments work in the orchestra and gain insights into the inner workings of the Philharmonia Orchestra.

Taiko no Tatsujin [3] is a rhythm game where players match up beats to on screen icons. Different symbols indicate the user what to hit and when.

It provides users with a variety of platforms. For mobile version, fingers are used for drumming.

Audiotool [1] is a web app powered by Flash. It is an online music production studio that allows users produce their own music in the browser by connecting virtual devices, such as drum machines and synths. Different instruments and sound effects are also available in order to create a satisfactory track which can be promoted to different platforms.

Ocarina [15] is one of the earliest mobile-musical apps for iPhone that presents a flute-like physical interaction using microphone input, multi-touch, and accelerometers. It also provides a social interaction that allows users to listen in to others playing Ocarina around the world using GPS location and cloud-based networking.

Whether the applications are developed on web, iOS, Android, or some other platforms, digital technologies have changed the music practice. As the web application can relieve the developer of the responsibility of building a client for a specific type of computer or a specific operating system, I built a web application for the project. Design issues specific to the web application under development will be discussed in the later section.

2.2 Web application design

The web is no longer just a way of presenting information on a computer screen. With the development of the basic building blocks of web technology, HTML, CSS and JavaScript, a wide variety of client systems are able to use information from the web, which makes the web application run like native installed applications on any platform. Therefore, the engineers can save time as there is no need for rebuilding the app for different platforms, and launch products and features as soon as they are ready. Moreover, users are able to get the latest version of the app just by hitting refresh. The cost

savings are substantial [14].

Though the web application uses the same network protocols as the desktop web environment, it's inadequate to think of a web application as a website. There are some considerations listed as follows when designing the web application so as to allow the developer create only one file but provide different experiences on a variety of devices, including desktops, smart phones, tablets and so on [7].

1. Multiple views

   The lack of a browser's toolbar for navigation means the back feature should be implemented by the developer. Generally there is only one HTML file and then using JavaScript will change the view. Therefore, tab navigation at top or bottom, top toolbar for going back, or key and touch paginating for sequential views can be used as multi view mechanisms.

2. Layout

   The layout can be different even with a single platform. A fixed or a liquid design should be decided to support different screen sizes, orientation modes, and physical screen dimensions.

3. Input method

   Since there are different input methods, such as touch, keyboards and pads, the web application need to support all these input methods. One way of handling touch devices is to change the layout for larger components.

4. One-view application

   One-view application is commonly used for financial, weather, news and some other indicator applications, which have an information view and

maybe a second view for explaining the details of the main view. One-view applications can be easy to create but feature powerful functions.

5. Multiplatform application

   All the code can be delivered from the server in order to let user receive the latest version of the web application without intervention. Therefore, one way to reuse the code for all the platforms is to define a class using JavaScript, which is a dynamic language, for the `body` tag that will be used by the CSS file to define different styles for platform variants. Another approach is to have various CSS files for each platform.

Understanding the architectural and usability concepts is critical to design the web application for a variety of platforms. Main supported technologies will be discussed in the next section for better understanding the development process of this project.

2.3 Web technologies

As it has been mentioned above, the web apps are normally built using HTML, CSS and JavaScript. They are regarded as client-side languages used in web programming. How to create and process content in a web app by these languages will be introduced in the later three subsections respectively. Using them as a base, the detailed technical issues of The Organ Web App development will be discussed in the next chapter.

2.3.1 HTML

HTML (the Hypertext Markup Language) [12] is the language for describing the structure of pages, which allows the format of documents to be viewed on screens. The formatting codes are all written in plain text. One of the

most important features of web documents is that they contain hyperlinks which let users navigate to other documents.

The fundamental syntactic units of HTML are called *tags*. A tag is a format name surrounded by angle brackets. End tags which switch a format off also contain a forward slash. With different kinds of tags, HTML can publish documents with headings, text, tables, lists, images, etc [13]. For example, the text surrounded by a h1 beginning tag and a end tag will be set to the style h1.

```
<h1> Text in an h1 style. </h1>
```

All HTML documents follow the same basic structure. They have a head which contains control information used by the browser and server and a large body. The body contains the content that displays on the screen and tags which control how that content is formatted by the browser. The basic document is:

```
<html>
    <head>
        <title>A HTML document for example</title>
    </head>

    <body>
        <h1> This is a heading. </h1>
        <p> This is a paragraph. </p>
    </body>
</html>
```

The entire document is surrounded by `<html> ... </html>` so that the software can know it is now processing HTML. Documents stored with a `.html` extension to their file name are automatically treated as HTML files. The head of the document can be used to import stylesheets, files containing

scripting code, or information about the document itself which is called *meta-data*. The body contains the content that displays on the screen and tags which control how that content is formatted by the browser.

2.3.2 CSS

CSS (Cascading Style Sheets) [5] is the language for describing the layout of pages, which includes positioning on the page and the choice of fonts, colours, borders. backgrounds and so on. In particular, the page may be viewed on different platforms with images, text, tables, or other elements in a variety of styles. Straightforward HTML is not able to cope with this diversity, however, CSS comprised of a set of formatting instructions can apply styles to the HTML document. Generally the following three mechanisms are used [6]:

1. Inline - the style can be defined within the basic HTML tag. This inline styling changes the text colour of a single heading:

   ```
   <h1 style="color:blue">This is a Blue Heading</h1>
   ```

2. Internal - the style can be defined in the `<head>` section and applied to the whole document. This internal styling changes the text colour of all the paragraphs:

   ```
   <html>
       <head>
           <style>
               p {colour: green}
           </style>
       </head>
       <body>
   ```

```
        <p>This is a paragraph.</p>
        <p>This is another paragraph.</p>
    </body>
</html>
```

3. External - the style can be defined in external files called stylesheets which can then be used in any documents by including the stylesheet via a URI. This external styling changes the style which are declared in the external `styles.css` file:

```
<html>
    <head>
        <style>
            <link rel="stylesheet" href="styles.css">
        </style>
    </head>
    <body>
        <p>This is a paragraph.</p>
        <p>This is another paragraph.</p>
     </body>
</html>
```

2.3.3 JavaScript

JavaScript is capable to program at both the server and the client side. However, server-side JavaScript is used less frequently than the client-side. From the client side perspective, JavaScript is a scripting language whose primary uses are to validate from data and to create dynamic HTML documents [8].

As button clicks and mouse movements are easily detected, such form elements can be described in JavaScript to make interactions with users. JavaScript can provide feedback to the user according to the computations

triggered by the form elements. The input can also be fetched from the user in dialog window when there is no form. JavaScript makes interactions with users easy to program in order to generate new content display dynamically.

Another interesting capability of JavaScript was made possible by the development of the Document Object Model (DOM), which allows JavaScript scripts to access and modify the CSS properties and content of the elements of a displayed HTML document, making formally static documents highly dynamic [11].

The point of the DOM and JavaScript is to allow developers to write scripts that change elements on a page in response to user input, thereby allowing pages to be more responsive and more like applications. There are three basic principles that should be observed in order to use JavaScript sensibly [4].

Pages with scripts should degrade gracefully. That is they should still display acceptably, present all their information and provide all their services, even if scripting is not supported or is disabled in a user's browser.

Graceful degradation can more easily be achieved by observing the second principle: *behaviour should be separated from content and structure.* This is sometimes called *unobtrusive scripting.* It is emphasised that content and structure should be separated from appearance. This is achieved using stylesheets, which are connected to page elements using `class` and `id` attributes, without otherwise interfering with the markup. Scripts should be similarly attached to a page without interfering with the markup, and `class` and `id` attributes can be used here to connect scripted actions with elements of the page, in a way that it will described in details in the next chapter.

The third principle should hardly need stating, but it is frequently neglected: *scripts should work correctly.* This implies that they must be thoroughly tested, and analysed to demonstrate their correctness. Ideally, scripts should work correctly in all browsers, but because of the legacy of incom-

patible scripting language implementations and object models, it is difficult to guarantee that every script will work in all older browsers. It is better to insist that scripts work in all browsers that correctly implement the DOM and JavaScript standards, and aim for graceful degradation in others. Basically, this means treating non-standard browsers as if they didn't implement scripting at all.

Chapter 3

Web app development

3.1 System requirements

As the Organ Project aims to share the historically important heritage of the Henry Willis pipe organ with as wide an audience as possible, one way of meeting the requirement is to make a virtual tour featuring different aspects of the organ by a cross-platform application.

The goal of Organ Project is not only to honour the legacy of the pipe organ by keeping up a regular organ recital diary, but also to programme concerts right across the genre pool, breathing fresh life into the organ by having it performed in styles which have never previously been attempted on a mechanical organ. However, only the live audiences in these concerts can feel how versatile the organ can be. People who have never been to the Union Chapel may just simply know the organ is a church instrument. With the popularisation of digital devices such as laptop, smartphone and tablets, versatility of the organ can be reflected by an application in the devices, which provides an opportunity for people to learn more about the organ.

The main reason for making a virtual tour is because the organ at Union Chapel is deliberately hidden away behind ornate screens under the rose

window when under construction. The console is directly behind the pulpit so that the congregation will not be distracted by the sight of an organist. The music played by the organ will be the focus during worship. Since the application concentrates on the organ itself, a virtual tour can lead users to know where different parts of the organ are located. Especially for people don't have access to the organ, a virtual tour can provide users with a more intuitive experience.

Different aspects of the organ refer to its history, console, pipes and hydraulics. The organ was designed and built specially for the size and acoustics of the new chapel building in 1877 by master organ builder Henry "Father" Willis. It is one of just two organs left in the United Kingdom, and the only one in England, with a fully working original hydraulic (water powered) blowing system, which can be used as an alternative to the electric blowers. The console shown in Figure 3.1 has 3 manuals (the organ term for keyboards): swell, great and choir, plus full foot pedals. It has 37 speaking and 2 sound altering stops (these are the knobs on either side of the organ console). By drawing the stops the player engages sets of pipes, which fall into various categories of sound quality - diapasons, strings, flutes, reeds - and at various pitches and volumes. There is also an enclosed swell organ with a mechanism for opening the box, thus controlling the volume. In order to let users operate the console, the application should provide an interface which allows users to press the keyboard, draw the stops and hear different timbres.

When the organ was restored in 2012 the engines formed a core part of the restoration but a system was needed to revive the engines without resort to wasting water. The present hydraulic system makes use of an electrically driven centrifugal water pump, regulated via various pressure bypass valves and safety valves, to draw and pressurise water from a large holding tank and feed it, via the original control valves, to the engines. The exhaust

Figure 3.1: The console of the organ.

water is then simply fed back to the holding tank. In order to explain how the hydraulics works in a more intuitive way, a video and a text description should be offered in the application.

The next section will discuss how to meet the above requirements from a technical perspective.

3.2 System specifications

3.2.1 The platform chosen

The main reason for building the application as a web app is that the cross-platform HTML content can reduces the cost of the project by being shared among different platforms such as iOS and Android. Moreover, the content can be mobile optimised, which provides an appropriately sized and touchscreen-enabled experience for mobile visitors. Table 3.1 presents some of other differences between native apps and web apps. As the app in this report aims to be shared with more users, and all of the required functions and

effects can be achieved by the existing well-developed APIs and libraries, which will be discussed hereafter, the app is developed based on the web platform and named as “The Organ Web App”.

3.2.2 Web audio API

The fifth version of the HTML standard adds many new syntactic features, among which the `<audio>` element provides native support for audio playback in all modern browsers [**?**]. However, there are significant limitations of the `<audio>` element such as no way to pre-buffer a sound nor analyse sounds. In order to provide a versatile system for controlling audio on the web, the Web Audio API is developed, which is completely separate from the ¡audio¿ tag. As a high-level JavaScript API, it is widely used in games, interactive applications and music synthesis applications. Therefore, the Web Audio API is applied in The Organ Web App for allowing users to hear different sounds when playing the keyboard in the app, which features pre-recorded audio samples of six timbres from the organ.

The `AudioContex` is the main concept for building the Web Audio API. It shapes the audio graph by defining how the audio modules linked together from its source to its destination. Each module is represented by an `AudioNode`. The node's properties can be modified when the audio passes through. The following code presents how to initialise an `AudioContex`.

```
var contextClass = (window.AudioContext ||
  window.webkitAudioContext ||
  window.mozAudioContext ||
  window.oAudioContext ||
  window.msAudioContext);
if (contextClass) {
  // Web Audio API is available.
```

The issues	Web app	Native app
Hardware integration	Access through the browser is limited. Access to geolocation can be made by Javascript.	It has all the access including camera, microphone, accelerometer, etc.
Distribution	It can be accessed through URL. It can behave like a native app by being bookmarked to home screen on iOS.	It should be downloaded through app stores, and most app stores require approval.
Performance	The device has to interpret the code into native code at run time, which takes time especially for multiple animations.	The code is precompiled, therefore high performance can be fast without interpretation once the app runs.
Cross-platform	It can share the same code across multiple platforms. CSS can specify the size once the size of screen changes.	Code is written in a specific language for each platform. It has to be installed first before running.
Updates	As all the contents are saved on the server, users can update an app just like refreshing a website.	Users need to download the latest version through the app stores.

Table 3.1: Web app vs. native app

```
  var context = new contextClass();
} else {
  // Ask the user to use a supported browser if it is not available.
  alert('Web Audio API is not available in your browser.');
}
```

The `AudioBuffer` is used to cache the audio samples, which can be multiple formats such as WAV, MP3 and OGG. As `XMLHttpRequest` makes sending HTTP requests very easy, it is also used to load an audio sample into the Web Audio API. As the audio sample is a binary file, the `responseType` should be set as `arraybuffer`, which will be processed by `decodeAudioData`. This all happens asynchronously and doesn?t block the main UI thread. The following code presents how to load an organ audio sample.

```
var organSampleBuffer = null;
var context = new webkitAudioContext();

function loadOrganSample(url) {
  var request = new XMLHttpRequest();
  request.open('GET', url, true);
  request.responseType = 'arraybuffer';

  // Decode asynchronously
  request.onload = function() {
    context.decodeAudioData(request.response, function(buffer) {
      organSampleBuffer = buffer;
    }, onError);
  }
  request.send();
}
```

Once the `AudioBuffer` has been loaded, a source node can be created and connected into the audio graph. Then call `start(0)` to play the sound. Supposed the above organ audio sample has been loaded to the buffer, the following code shows how to play the audio.

```
function playSound(buffer) {
  var source = context.createBufferSource();
  source.buffer = buffer;
  source.connect(context.destination);
  source.start(0);
}
```

The `playSound` function can be called anytime, for instance, mouse clicking or keyboard pressing. The Web Audio API also allows developers to change basic properties of the audio, such as timing and loudness, visualise the audio, apply spatial effects and much more.

3.2.3 JavaScript Libraries

3.3 Prototype

3.3.1 Paper prototype

3.3.2 Design process

3.3.3 Implementation

3.3.4 Testing

Bibliography

[1] Audiotool. `http://www.audiotool.com`.

[2] The orchestra. `http://orchestra.touchpress.com`.

[3] Taiko no tatsujin. `http://taiko-ch.net`.

[4] Chris Bates. *Web Programming Building Internet Applications*. John Wiley & Sons, 2002.

[5] Bert Bos, Håkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading style sheets, level 2 css2 specification, 1998.

[6] Nigel Chapman and Jenny Chapman. *Web design: a complete introduction*. John Wiley & Sons, 2007.

[7] Maximiliano Firtman. *Programming the mobile web*. " O'Reilly Media, Inc.", 2010.

[8] Danny Goodman. *JavaScript bible*. John Wiley & Sons, 2007.

[9] Peter Gouzouasis and Danny Bakan. The future of music making and music education in a transformative digital world. *The University of Melbourne refereed e-journal*, 2, 2011.

[10] Manuel London and MJ Hall. Unlocking the value of web 2.0 technologies for training and development: The shift from instructor-controlled,

adaptive learning to learner-driven, generative learning. *Human Resource Management*, 50(6):757–775, 2011.

[11] Gavin Nicol, Lauren Wood, Mike Champion, and Steve Byrne. Document object model (dom) level 3 core specification. *W3C Working Draft*, 13:1–146, 2001.

[12] Dave Raggett, Arnaud Le Hors, Ian Jacobs, et al. Html 4.01 specification. *W3C recommendation*, 24, 1999.

[13] Robert W Sebesta. *Programming the world wide web*. Pearson Addison Wesley, 2008.

[14] Antero Taivalsaari and Tommi Mikkonen. The web as an application platform: The saga continues. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 170–174. IEEE, 2011.

[15] Ge Wang. Designing smule?s ocarina: The iphone?s magic flute. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 303–307, 2009.