Interactive Music Systems for the Web

Chi Kim

N17253820

Submitted in partial fulfillment of the requirements for the

Master of Music in Music Technology

in the Department of Music and Performing Arts Professions

in The Steinhardt School

New York University

Advisor: Robert Rowe

[DATE:2009/05/05]
Abstract

This research is about how to deploy interactive systems on the Web in order to take advantage of the fast growing internet. It explores the history, background, and technical aspects of interactive music and different Web technologies. It discusses how to bring them together to create a better user environment. Lastly, it includes the development process of a simple Web application for interactive music called Random, which can be found at [http://chikim.com/random/](http://chikim.com/random/).
Table of Contents

CHAPTER 1: INTRODUCTION  
CHAPTER 2: INTERACTIVE MUSIC SYSTEMS  
CHAPTER 3: INTERNET AND WORLD WIDE WEB  
CHAPTER 4: COMBINING INTERACTIVE MUSIC AND WEB 2.0  
CHAPTER 5: CURRENT TREND  
CHAPTER 6: OVERVIEW  
CHAPTER 7: TECHNICAL ELEMENTS  
CHAPTER 8: REVIEW  
CHAPTER 9: CONCLUSION  
BIBLIOGRAPHY
Chapter One: Introduction

This thesis will explore Web technology that would make interactive music more accessible to users. The internet has dramatically evolved in the past decade, and it plays an important role in many people’s daily lives. Many people no longer get entertainment media like video and music from physical stores. They can now enjoy such things from the comfort of their homes with a click of mouse. A lot of media types are moving to the internet, and it is also time for Interactive Music to do so. Interactive music is a fairly new medium, and it is constantly being improved. Thanks to modern computing power and broadband connections, now it is possible to bring interactive music to dynamic Web also known as a loosely defined term called “Web 2.0.”

Goals:

This research has two components: discussion and development. It is important to look at the progress of how interactive music has evolved over time in order to understand the future. This thesis will briefly look at technology related to interactive music as well as technology related to the web. It will also look at what is available on the Web as of 2009 in interactive music. Deploying interactive music to the Web is not always easy. In Web technology, there are restrictions such as limited bandwidth, limited physical devices to incorporate, browser compatibility, and so forth. This thesis will look at some statistical data to pinpoint the problems and explore ways to solve them.
The development part will focus on creating a simple Web application for interactive music. It will discuss the process of development from coding to launching it on the web. It will also discuss problems and solutions along the way.

**Motivation:**

The motivation behind this research is to help people develop interactive music systems on the Web by providing solid background information and exploring problems they might encounter while developing. Another motivation is to create an environment where non-musicians with little formal knowledge of music can interact with computers and compose relatively complex music in a non-traditional way. This allows non-musicians to participate in music creation activities without much knowledge of music.

**Benefits:**

Deploying interactive music on the Web environment has a number of benefits. For example, once a program is developed, it can be used in multiple platforms. It is possible for people from different places to interact with each other. Also, it does not require a special device or software, so people can enjoy it anywhere with internet access. It is possible to make it easy for users to share their music and create a dynamic community online. These will make interactive music more accessible and appealing to people. In addition, interactive music has educational and therapeutic values; therefore, greater accessibility would lead more people to benefit.
Thesis Organization:

This thesis is organized as the following outline. The entire thesis, excluding introduction and conclusion, can be divided into two parts. The first part (chapters two through five) will look at the overview and the current trend, and the latter chapters (six through eight) will focus on developing a small project. The second chapter will explore the history, background, and technical aspects of interactive music. The third chapter will explore the history, background, and technical aspects of Web technology. The fourth chapter is on how to bring them together. It will discuss things to consider when developing an interactive music system to deploy on the web. It will also discuss possible problems and solutions. The fifth chapter will focus on current trends. It will introduce and explore a number of projects that are currently available on the web. Chapter six will include an overview of the project (goals, features, and interface description), chapter seven will explain the coding aspects, and chapter eight will wrap up by reviewing the project (artistic analysis and future). Finally, chapter nine will conclude this thesis.
Chapter Two: Interactive Music Systems

Background:

Incorporating machines into the music-making process has a fairly short history; however, most music we listen nowadays is processed by machines to a certain extent whether the music is generated by computer or is simply amplified. People first started using machines in order to expand timbral content of their music (Rowe, 1993). Tape machines were very popular in late 1950’s and 1960’s to manipulate recorded sound to produce different timbres. In early 1960, people also started using computers to make music by generating sounds. Programmers needed to write codes for the computer and wait for the computer to generate those sounds. Since computers were not fast at that time, composers needed to wait a couple of weeks just to hear a piece that is few minutes long. Computers became more powerful to handle different tasks, so it can instantly generate sounds in real time as it gets input from users. In addition, only a few people in the past had access to computers to make music, but now we can see computers being used to make music everywhere from live concerts to a composers’ bedrooms.

Another way of using machine in music is to manipulate or algorithmically generate musical content to produce music (Rowe, 1993). For example, David Cope developed a system called Emmy that can emulate different composers through analysis of the composers’ works (Muscutt, 2007). It can produce music in many famous composers’ styles like Bach. This is possible because computers can handle certain tasks at a fast rate. They could go through thousands of data and analyze them, and they are best at number crunching. This allowed people to take advantage of computers for algorithmic composition. As a result, they could write programs to generate musical content as well as sounds. Once computers became fast enough,
interactive music systems could be developed in computers as well, and the systems were
capable of generating and manipulating both music and sound in real time.

Interactive music systems are machines that respond to performers’ actions to produce
music in a collaborative way. The key element is that both performers and systems contribute to
the outcome. As a result, generation and manipulation of musical content is the most crucial part
of interactive music systems even though sound generation plays an important role. Being able to
generate and manipulate is what makes a system interactive.

The level of interactivity can dramatically vary from system to system. There are
interactive music systems that listen to live performance and react to what they hear, and they
can provide real time interactions between the performers and the systems (Rowe, 1993). On the
other hand, as Winkler points out, we can even make a system interactive by just allowing the
program to change the tempo of music depending on user input (Winkler, 1998). Also, adaptive
music system used in games is interactive to a certain extent since games provide musical
feedbacks by generating or cueing different music depending on the situation. However, the
focus of this research is mainly on interactive music systems in which music is their primary
goal.

Classification:

There are many different types of interactive music systems, but we can categorize them
by different dimensions. In his book, Interactive Music System, Robert Rowe provides the
following classification system to categorize different interactive music systems (Rowe, 1993).
• Interpretation: Score-driven vs. performance driven
• Response: Transformative, generative, or sequenced
• Paradigms: Instrument vs. player

The first dimension categorizes by how the system analyzes music in time. Score driven systems utilize “predetermined event collections, or stored music fragments”, to respond to user input using meter and tempo system (Rowe, 1993). On the other hand, performance driven systems exploit “more general parameters, involving perceptual measures such as density and regularity” instead of metric system to analyze user input (Rowe, 1993).

The second dimension describes how the system responds. As the name suggests, Transformative methods apply transformations to existing music or user input “to produce variants … [which] may or may not be recognizably related to the original (Rowe, 1993). Generative method utilizes a set of rules to produce different materials using basic elements such as scales and duration (Rowe, 1993). Lastly, sequenced method uses pre-recorded materials to respond to user input, and the system may change: tempo of playback, dynamic shape, slight rhythmic variations, etc” (Rowe, 1993).

The last dimension classifies by how the system is designed to be perceived. Instrument paradigm systems are designed to be perceived as extended musical instruments, and its output can be far more complex than a traditional instrument (Rowe, 1993). When a performer plays an instrument paradigm system, it will be perceived as a solo performance; however, player paradigm systems are designed to emulate another player. Thus, when a performer plays with a player paradigm system, it will be perceived as a duet (Rowe, 1993).
Rowe (1993) gives an excellent example to illustrate how these dimensions can be used to classify an interactive music system.

Score followers are a group of programs able to accompany a human instrumental soloist by matching her realization of a particular score against a stored representation of that score, simultaneously performing a stored accompaniment part. Such applications are a perfect example of score-driven systems. The response technique is sequenced, since everything the machine plays has been stored in advance. Finally, score followers can be regarded as player paradigm systems, because they realize a recognizably separate musical voice, assuming the traditional role of accompanist in the performance of instrumental sonatas.

**System design:**

We can break down an interactive music system into a number of different components. As Winkler explains, we can perceptually look at these components as “the key elements in making and listening to music: [virtual] instruments, performers, composers, and listeners” (Winkler, 1998).

Virtual instruments behave … like real instruments, and can be played from the computer keyboard, a MIDI controller, or as an extension of a traditional instrument. Virtual performers may play with human performers, interacting "musical intelligence" in a duet, combo, or accompaniment role. Virtual composers create original music based on flexible and sometimes unpredictable processes specified by a real composer. … Finally, a virtual listener or virtual
critic may pass judgment by reacting to and altering the final outcome of a performance. (Winkler, 1998)

As Chadabe (1984) explains in his article called “Interactive Composing,” we can also break down an interactive music system into different components in a mechanical perspective.

- **Programmable device**: Listens and analyzes user input and constructs its feedback
- **Sound generator**: Output system through which its feedback is rendered
- **Controller**: Component that accepts user input

When developing an interactive music system, one needs to craft each virtual identity so that the system can provide a rich interactive experience for users. Programmable device is the brain of an interactive music system, and it plays a crucial role in defining its characteristic. It contains a number of different algorithms like performance interpretation algorithm, compositional algorithm, sound algorithm, and response algorithm. Each algorithm is responsible for each virtual identity described above, and they work together to produce a cohesive result, ultimately its response or output.

In order to make a system dynamically interactive, developers need to carefully define the relationship between the user input and the system’s response. There can be a number of variables to construct system’s response, and it is developers’ responsibility to decide which variable can be controlled by users. Also, it is equally important to decide how users can change those variables. At the end, the system’s response should ideally meet the following criteria explained by Chadabe (1984). The response should be “interesting and informative … [with] new, unexpected information,” and it should be also “recognizably related to the performer’s actions.” If the system’s response is not related to user input, the user input would not have any
meaning (Chadabe, 1984). Lastly, both the user and the system should contribute to the final outcome, and ideally both should be mutually influential. Developing a system with these in mind should yield a decent result. The main goal of an interactive music system should be to provide users with experience that is challenging, explorative, engaging, enjoyable, and intuitive (Robson, 2002).

Operation:

Operational stage is when a performer interacts with a system according to its design and produces music, and the process can be broken down into the following cycle.

- Initiation: User or system initiates.
- Process: System processes user input.
- Response: System responds to user input.
- Reaction: User reacts to the system’s response.

In operational stage, every time a performer plays an interactive system, he or she would not have the exact same conversation. It might be a similar conversation, but the result is going to be different. The performer would adapt to those unexpected changes and vary the performance as well. Winkler begins his book, *composing interactive music*, with the following statement:

Interaction is a two-way street. Nothing is more interactive than a good conversation: two people sharing words and thoughts, both parties engaged. …
On the other hand, when only one person does the talking it isn't interactive—it is a lecture, a soliloquy. (Winkler, 1998)

In this chapter, we briefly looked at the overview of interactive music systems and their basic designs. When developing an interactive music system, it is a good idea to define its goals before actually designing it and what we explored in this chapter will give a good guideline. In the next chapter, we will explore Web technologies that can help interactive music reach broader audiences.
Chapter Three: Internet and World Wide Web

Background:

Even though almost everyone in modern countries now has access to the Internet, the internet was not initially intended for mass usage. As Sterling (1996) explains, it began in the 1960's for the military purpose of having a decentralized network that can be sustained even when some computers on the network fail. ARPANET, the network from Advanced Research Projects Agency (ARPA) was the first US network established around academic institution and governmental agencies. Later in 1982, TCP/IP (Transmission Control Protocol/Internet Protocol) became the standard network protocol, and it is still the primary protocol we use today for the Internet.

The crucial part of the Internet was that anyone was able to join the network and exchange information using TCP/IP, and no one kept people from getting connected. This is what fueled the Internet to grow enormously. The Internet now became a personal tool, and it affects many people's daily lives in modern countries today. The chart below shows the current world internet usage statistic as of 2009, and we can see that it is growing at a fast rate. It plays an important social function and is big part of culture in most places.

*** Figure: USA Adult Internet Usage Statistic
World Wide Web (WWW):

In 1991, World Wide Web was born by Tim Berners-Lee at CERN where thousands of researchers and students were working. The amount of information gathered at the institution was too enormous, and they needed an effective way to organize the information and make it more accessible (Berners-Lee, 1989).

The aim would be to allow a place to be found for any information or reference which one felt was important, and a way of finding it afterwards. The result should be sufficiently attractive to use that the information contained would
grow past a critical threshold, so that the usefulness the scheme would in turn encourage its increased use. (Berners-Lee, 1989)

The main idea of WWW was to effectively manage large information by linking them together so that people can easily navigate between one document and another. Berners-Lee deployed the five components shown in the figure below to implement World Wide Web and achieved his main objectives which can be summarized into the following points:

• The provision of a simple protocol for requesting human readable information stored in remote systems accessible using networks

• To provide a protocol by which information could automatically be exchanged in a format common to the information supplier and the information consumer

• The provision of some method of reading text (and possibly graphics) using a large proportion of the display technology in use at CERN at that time

The provision and maintenance of collections of documents, into which users could place documents of their own

• To allow documents or collections of documents managed by individuals to be linked by hyperlinks to other documents or collections of documents

• The provision of a search option, to allow information to be automatically searched for by keywords, in addition to being navigated to by the following of hyperlinks
• To use public domain software wherever possible and to interface to existing proprietary systems

• To provide the necessary software free of charge. (From Hypertext to the World Wide Web, 2008)

*** Figure: Five Components for World Wide Web.

• Hyper Text Markup Language (HTML): contains actual contents and may point to other HTML documents.

• Server: stores HTML documents and disseminate them.

• Universal Resource Locator (URL): serves as address to locate documents

• Hypertext Transfer Protocol (HTTP): allows communication between servers and clients.

• Browser retrieves data from servers and display according to the HTML specification.

**HTML:**

HTML documents contain the actual content to be displayed and it consists of texts and graphics. In addition to actual content, HTML can also contain layout and format for how the document should be displayed. The contents in HTML documents can be organized with tables, lists, headings, and so on. HTML specification keeps getting updated by World Wide Web Consortium (W3C), and it is important to follow the specification for proper presentation.
**Server side scripting:**

HTML, however, is very static in the sense that people have to edit them manually in order to update. To change the static nature of HTML, People came up with server side scripting in which Web servers can dynamically generate HTML documents depending on user's request. This allowed people to interact with websites. The common language used for server side scripting includes (but not limited to) ASP, PHP, and Perl. Server side scripts can access database and files on the server side, and it can examine the user's request and display, update, or generate appropriate information. For example, if someone wants to post a song on a website without server side script, the following steps are necessary.

- Artist emails webmaster the song.
- Webmaster uploads the song to the server.
- Create HTML file that has information for the song.
- Upload the HTML file to the server.
- Update the HTML file that has the list of songs by adding the new entry.

If artist wants to post a new song or even change the song with new mix, the entire steps need to be repeated. With a server side script, artist can simply upload song directly to the server along with the information. Then the script would automatically generate a new HTML file and update the master list. The latter case is instant, whereas the former could take a long time.

**Database:**

A database is used along with server side scripting. A database simply stores data, and allows scripts to easily retrieve only needed data. The popular database technology used for Web
includes, but is not limited to, MYSQL and MSSQL. A database is optimized to efficiently perform any data operation such as comparing, merging, sorting, calculating, updating, and so on. Server side scripting works as a bridge between the database and user, and the steps below show how information flows between user and database:

- User specifies what he/she would like to see by clicking a link or providing some information.
- The information provided by user is transmitted to the script on server through http request.
- Script queries database to retrieve relevant data.
- Database retrieves specified data and passes it to the script.
- Script builds new HTML document to present the data in a way for user to understand.

*Client side scripting:*

To improve the user interface, people came up with client side scripting where browsers handle running the script from the users’ side. With server side script, users have to leave the current page and get a new HTML file from the server in order to update information shown on a page or the design. However with client side script, users can interact with the current page, e.g. showing or hiding additional information and validating a form. This allows better, faster interactions with website for users, and it reduces connections between clients and servers. In addition, it eases taxation on server resources since it assigns computation tasks to browsers. The major technologies for client side scripting include, but not limited to, JavaScript by Netscape and Visual Basic Scripting Edition (VBScript) by Microsoft.
AJAX:

Later, the term AJAX (Asynchronous JavaScript and XML) appeared in Web technology. It refers to a technique where scripts can asynchronously retrieve data from the server in the background. It is combination of server side scripting and client side scripting. AJAX reduces the traffic by retrieving only required information, and the user most of time would not notice that the browser is making HTTP requests in the background to communicate with the server. In addition, user interaction with webpages is seamlessly transparent without interruption. There are a number of technologies involved to implement AJAX, and Garrett explains:

AJAX isn’t a technology. It’s really several technologies, each flourishing in its own right, coming together in powerful new ways. AJAX incorporates:

- standards-based presentation using XHTML [(Extensible Hyper Text Markup Language)] and CSS [(Cascading Style Sheets)];
- dynamic display and interaction using the [DOM] (Document Object Model);
- data interchange and manipulation using XML [(Extensible Markup Language)] and XSLT [(Extensible Stylesheet Language Transformations)];
- asynchronous data retrieval using [XHR] (XMLHttpRequest);
- and JavaScript binding everything together. (Garrett, 2005)
AJAX is a new way for clients and servers to communicate in order to generate a dynamic HTML document, and the two figures below illustrate the difference between the AJAX model and the classic model.

- *** Figure: AJAX 1

- *** Figure: AJAX 2
Plug-in:

There are technologies that deliver programmable rich content for browsers, so people can write separate programs and embed them on WebPages. The programs can be anything from a simple mp3 player to a full blown music sequencer. The main technology includes, but is not
limited to, Flash by Adobe and Java by Sun Microsystems. The programming environments support both server and client programming, so implementing interaction over network is possible. The main benefits of the technologies are object-oriented programming and portability. Object oriented programming allows programs to be easily expanded because each object is written to be self-sufficient and reusable later. The plug-ins are runtime environment, which means that they are platform independent. Runtime environment allows programs to run on Mac, PC, or even mobile phones as long as the plug-in is installed on the browser.

**XML:**

XML is a format that can structure different data into text file format. It is very simple and easy to use. Web services such as YouTube and iTunes use XML technology to organize their libraries. World Wide Web Consortium explains XML as:

> Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. (Extensible Markup Language (XML), 2009)

XML is not limited to only the Web. It can be used in any environment that needs data structure. It is especially useful in Web technology because all the technology described above supports XML format. For example, JavaScript can communicate with PHP in XML, and Flash and Java Applets can communicate in XML as well. How XHTML and CSS are structured is
also very similar to XML. In addition, it is very easy to understand how they are structured. The following figure shows an example of an XML file. It is definitely a human readable document.

*** Example: XML Example

<?XML version="1.0" ?>

<!-- A Simple XML Example -->

<catalog>

<album title="Alone (Again)" artist="Bill Evans" genre="Jazz" year="1975">

<track number="1" length="7:03">

<title>The Touch of Your Lips</title>

<file>Bill Evans - 01 - The Touch of Your Lips.mp3</file>

</track>

<track number="2" length="8:58">

<title>In Your Own Sweet Way</title>

<file>Bill Evans - 02 - In Your Own Sweet Way.mp3</file>

</track>

</catalog>
XML defines a new way to structure data, and portability, efficiency, and wide support are its strengths. XML allow data to be exchanged between programs very easily and efficiently.
Building blocks for World Wide Web:

Using all the technology described above, we can create an interactive website for users. Like most software, we can also break down the different technologies into the following components by their function, and this is exactly how AJAX is designed to connect those elements all together.

- User interface layer
- Logic layer
- Database layer

User interface defines how users interact with a website with presentation layout, pictures, buttons, links, checkboxes, radio buttons, combo boxes, editable textboxes, animation, and so on. This is usually built with XHTML and Css. The logic block defines how to process the input from the user interface and behaves accordingly. This is usually built with scripts such as PHP, ASP, or PERL. The database simply stores informational data for logic layers to access. This is achieved with MySQL or MSSQL. The data flowing between database and user interface layer may be constructed in XML format.

Web 2.0:

The concept of Web 2.0 was conceived at a conference between O'Reilly and MediaLive International. The term is very vague, but it is loosely used to describe the next generation for World Wide Web, which focuses on user interaction. Web 2.0 combines all the technologies mentioned above to create a transparent environment for user to interact. Here are seven characteristics of Web 2.0 defined by O'Reilly Media (2005).
• Service: It is on-going service not a product.

• Leverage: Its target is the entire Web through customer self-service.

• Participation: Using the service is part of user’s participation.

• Data-driven: It grows and gets richer as more users contribute.

• Co-developers: Users are involved in development as co-developers.

• Platform-independent: It reaches out to different platforms including handheld devices.

• Lightweight user interfaces: It features easy content syndication, reuse of other data services, and "hackability" and "remixability.

*** Figure: Web 2.0 Meme Map
For better understanding of the characteristics listed above, here are some examples for Web 1.0 transformed into Web 2.0.

*** Table: Web 1.0 and Web 2.0 Comparison (O'Reilly, 2005)  

<table>
<thead>
<tr>
<th>Web 1.0</th>
<th>Web 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoubleClick</td>
<td>Google AdSense</td>
</tr>
<tr>
<td>Google AdSense</td>
<td>Flickr</td>
</tr>
</tbody>
</table>
### Conclusion:

What is amazing about Web technology nowadays is that it still uses original HTML. The new technology was developed to expand the original concept envisioned by Tim Berners-Lee. HTML did not become obsolete; we still use the core aspects of HTML. However, it is dramatically improved and transformed into something much better than original. Another trend we should pay attention to is Rich Internet Applications, the term coined by Macromedia. A lot of offline applications started going online. For example, word process and spreadsheet were once confined to offline. However, services such as Google Docs provide word processor and
spreadsheet online. People can now collaborate on projects more easily in Wiki style, and those projects became very portable. People started writing applications for Web and mobile environment more and more. Web 2.0 redefined the original concept of World Wide Web. World Wide Web does not mean just informational documents any more. Thus, it is a perfect time for interactive music to go online to take advantage of Web 2.0 as well.
Chapter Four: Combining Interactive Music and Web 2.0

We looked at general overview of interactive music systems and different Web technologies that are available for us to utilize. Now let us explore how we can combine them together to create richer experiences for users. In order to develop an interactive music system, developers need to choose a programming environment capable of accepting the input, analyzing the input, and creating the output on the fly. Also, it should be able to deal with musical data such as audio or notation. Languages mentioned in chapter three such as PHP, ASP, and PERL are good with dealing with HTML documents; however, they are not suitable for handling the types of data that an interactive music system would need. On the other hand, plug-ins mentioned in the chapter such as Flash and Java would be the perfect choice to tackle such projects. Both evolved into excellent programming languages with great potential, especially Flash with ActionScript 3, and they are widely supported in most Web browsers including mobile devices. Both support graphical interface, audio input/output, object oriented programming, and so on. One can develop an interactive music system in either platform. This research would not compare those two, but there are so many resources on the Internet to inform developers to make a perfect choice for what they need. One notable advantage of Java over Flash is that Java was not developed for just Web browsers. It is truly a virtual machine where it can be installed on robots and other devices that support Java technology. Whether a programmer write for Web browsers or robots, he or she would code the same way, and it can be migrated from one platform to another very easily.
**Limitation:**

As described in chapter two, the core element of interactive music systems is the programmable device. However, when deploying interactive music systems online, it is crucial to examine how we can develop controllers and sound generators. Since Web environment is a virtual environment, we cannot make or deploy unique physical devices to use as controllers and sound generators. As much as there are many good applications online for users, there are many bad applications online to abuse securities and hack into users’ computers. Therefore, browsers and runtime environment softwares have fairly tight security over types of access that internet application can have. When developing an internet application, we have to consider the following access limitations for security purpose:

- Access to files on client computers
- Access to non-generic external devices
- Outbound connection to unauthorized websites

Although users can modify those security features, it is usually designed to make it hard for an average internet user with limited knowledge in computers to do so. However, internet applications have the access to the following in general:

- Files packaged into the application
- Cookies generated by the same site that the application is from
- Generic external devices such as keyboard, mouse, monitor, audio and midi devices
- Outgoing connection to authorized websites
Even if we had access to unique external devices, we cannot assume that all the users who are going to use the application have the same devices. Therefore, it is a safe practice to stick to generic devices and make it more accessible to wider audience, which was one of the Web 2.0 concepts.

Another limitation we have to consider when deploying interactive music on the Web is the bandwidth. Speed Matters conducted a survey for internet speed, and Almost 230,000 people in USA participated in the survey in 2007 through 2008. According to the survey, the median download speed was 2.3 megabits per second, and the upload speed was 435 kilobits per second. It means that people can downstream up to 23 channels and upstream up to 4 channels of audio data encoded at 96 kbps per channel. However, the survey points out an important element about the survey result:

Very few people with dial-up took the test because it took too long. About 15 percent of Americans still connect to the Internet with a dial-up connection. So the median speeds in this report are actually higher than if dial-up Internet users had chosen to participate in the survey. In other words, even these dismal statistics paint a rosier picture than the reality. (A Report on Internet Speeds in All 50 States, 2008)

*** Figure: Broadband Speed
If we choose audio for sound output, another aspect we have to consider is latency. According to Ninjam, a Web service that provides an environment for musicians to jam online, the latency can be accumulated to more than 65 milliseconds: “sound hardware latency (>5ms), perceptual CODEC latency (>20ms), plus typical and theoretical network latency (>40ms) (Ninjam, 2008). In order for people to play together in synchronization, there should be no more than 25 ms latency which is known as ensemble performance threshold (EPT) (Renaud, Carot, & Rebelo, 2007). Because of bandwidth and latency issues, it would be very difficult to implement an interactive music system on the Web that utilizes online real time audio streaming.
**Work around for sound generators:**

There are a few options we can consider in order to work around the limitations above. First, if the system deploys an audio output, it can download audio samples beforehand to access them offline. Second, it can use synthesis method to generate sounds. Third, it can use MIDI for its sound source. Each solution has advantages and disadvantages.

The main advantage of using audio samples is the sound selection. It can include more authentic sounds compared to the alternatives below, especially if they are musical fragments. Also, it can include any sound that can be recorded in audio, so the selection is limitless. However, the disadvantages are that there is a limitation on how much we can include in the application due to the bandwidth issue. If the application downloads too many samples, it will take a long time to load, but it will have access to a wider selection and vice versa. Developers have to compromise between the number of samples and the time to load them. Also, it is hard to manipulate and transform audio samples in real time, so the developers have to select samples that can work together to make a musical sense. Otherwise, it would require too many samples with different variations for different situations. In addition, most virtual machines that run the application on browsers would throw a security error if the samples do not reside on the same domain that the application resides.

The main advantage of sound synthesis is flexibility. It is possible to transform elements such as timbre, pitch, volume, envelope, duration, and so forth in real-time. It would be possible to develop a system to treat each sound as notes on a traditional score. Also, synthesizing sounds can yield a very wide range of timbral contents. One of the disadvantages of sound synthesis is that it can be difficult to program especially if there is no API for sound synthesis for the particular environment. Also, it would require fair amount of knowledge in synthesis technique
in order to generate sounds with interesting timbres. Another disadvantage is that it would put much workload on the client computers to synthesize those sounds. If the application uses a complex synthesis technique that requires too many calculations, it would require a powerful computer to run. Again, developers have to compromise between speed and sounds.

The main advantage of using MIDI as a sound source is the efficiency. It is very light on system, and it uses a traditional notation system. It is possible to efficiently generate musical fragments on the fly without putting much pressure on the computer. Another advantage is that MIDI is very widely supported, and most computers have a sound card with MIDI capability. The main disadvantage of MIDI is limited sound selection. Most people would use the General MIDI sound designed for mass consumer market. General MIDI has a set number of sounds that are available as a standard, but the quality of sound is poor compared to the alternatives above. In addition, each sound may differ depending on the sound cards installed on the client computers. Therefore, each user would hear the result differently.

**Work around for controllers:**

In most cases, the application would have access to a mouse and a QWERTY keyboard as input sources. In order to overcome the limitation, developers can create virtual controllers on the screen to simulate physical controllers. Virtual controllers would require visual designing and graphical work, but they are much easier in general to design than actual physical controllers. Also, types of virtual controllers are limitless, and they are up to developers’ imagination.

It is easier to design graphical user interface on some programming environments, especially on Flash, so developers that would want to focus on creating virtual controllers may
choose Flash. Also, communication between Flash and Java is possible through JavaScript, so it is an option to put those environments together to take advantage of both strengths. However, JavaScript support is sometimes different from browser to browser, so developers might run into some problems in making it work on all browsers.

In addition to mouse and keyboard, microphone and Web camera are other options for input controllers. However, microphone and Web camera are tightly controlled for privacy intrusion. The application would need to go through process of obtaining a certificate and getting signed to pass the security. Another disadvantage of using microphones and Web cameras is that it is harder to program to analyze or process the input signals. Digital signal processing is required to analyze and manipulate those input signals, and it usually requires enormous system resource to achieve decent results.

*Multi-user environment:*

It is possible to develop an interactive music system to exploit the networking capability to implement a multi-user environment. In that case, developers would need to create two applications: one for the server to handle the interactions between clients, and another for clients to use. Both Java and Flash support the technology to server and client programming.

Similar to audio streaming described above, bandwidth and latency are issues when implementing a multi-user environment for collaboration. Particularly if each client exchanges audio data with a server to interact with other clients, the issues become even more prominent. It would require every user to be on a fast connection, and the server should be capable of handling an enormous bandwidth.
However, there is a workaround to implement a multi-use environment to avoid the issues. The following requirements need to be met for an application to achieve.

- Clients must not interact in musical timing.
- User input including audio data must get converted into a descriptive data before getting distributed.
- Each client must be able to reconstruct the required data such as audio from the descriptive data.

Each client would convert current state of application into a descriptive data, and the server would collect from each client, process, and distribute them back to the clients. This would be a perfect place to use XML as the data format. XML is very light and efficient, and all the technologies described in Chapter 3 support XML. It would be easy to make each technology to communicate in XML over the Internet, and tie them all together.

When developing an interactive music system for the web, it is important to consider the limitations of the Web such as security, limited controllers, bandwidth, latency, and so forth. As the Internet evolves, those limitations might not be issues in the future. However, developers need to tackle those problems either by considering the work-around discussed in this chapter as their options, or come up with creative ways to tackle the problems for now.
Chapter Five: Current Trend

There are a number of interactive music systems on the Internet, and each system has a unique approach in design. This chapter will introduce them by briefly reviewing their features.

Daisyphone by Nick Bryan-Kinns

URL: [http://gouda.dcs.qmul.ac.uk/](http://gouda.dcs.qmul.ac.uk/)

Daisyphone is an example of sequencing oriented programs. It is developed to create an interactive environment in a remote situation with a simple interface. Its goal is to create short loops through group collaboration. The notes are graphically represented, and people can contribute to the loop by adding or modifying the notes on the loop. Rather than assigning certain portion of music to each person, everyone has access to all the notes. However, people are aware of which part is contributed by whom. Also, timing is not critical when modifying, so the network latency does not interferes with user interactions. This program is user to user interaction rather than user to machine interaction. Daisyphone started as a Java applet, but now it works with iPhone as well. Similar projects include:

Music box

URL: [http://studentpages.scad.edu/~jgardn24/musicbox5.HTML](http://studentpages.scad.edu/~jgardn24/musicbox5.HTML)

Pianolina by David Krause, Volker Bertelmann, Fons Hickmann, and Simon Gallus

URL: [http://www.grotrian.de/spiel/e/info.HTML](http://www.grotrian.de/spiel/e/info.HTML)
Pianolina is a transformational interactive music system. Users can modify existing music such as ‘Elise’ by Beethoven and ‘Gymnopete’ by Satie. Users can vary the musical elements such as tempo and rhythm, and create their own patterns. Similar projects include:

Wind Chime Marimba by Rob Wright & Richard Polfreman
URL: http://www.soundtoys.net/toys/wcm-wind-chime-marimba

Neon Light by Macoto Yanagisawa
URL: http://www2u.biglobe.ne.jp/~macopism/noct/

Neon Light is an example based on visual and sound patterns, where each sound pattern is associated with a different visual effect. People can explore those patterns with mouse movements. Even though each visual and sound pattern is very attractive, there are a limited number of patterns for users to explore. Similar projects include:

Pixel by Pixel by Frédéric Durieu, Jean-Jacques Birgé, and Kristine Malden
URL: http://www.lecielestbleu.com/media/pixelbypixelframe.htm

Sound Garden by Babel & Binnorie
URL: http://www.cddc.vt.edu/host/404/interactive/soundgarden/

Sticky sound Elastic Structure by Santiago Ortiz (Colombia)
URL: http://moebio.com/santiago/pegajoso/

Sodaconductor by David Muth, Ed Burton, & Soda
URL: http://projects.soda.co.uk/sodaconductor/
Songsmith by Dan Morris, Sumit Basu, and Ian Simon

URL: http://research.microsoft.com/songsmith/

Songsmith is a unique Microsoft project to create a program that creates an accompaniment track for users. If a user simply chooses a style and sings or plays an instrument into a microphone, the program automatically creates an accompaniment track by analyzing the recorded audio. The approach is very interesting and creative, but the output of the program is not so promising. The program tries its best to match the audio with a typical chord progression, and the accompaniment patterns have a room for much improvement. Despite its weakness, this project is one of the most popular projects among the ones explored in this chapter due to its fun nature.

Pianographique by Jean-Luc Lamarque

URL: http://www.pianographique.net/

Pianographique is an example of a loop based system. Users can trigger different loops by using a mouse and pressing different letters on a computer keyboard. The loops are fragments of music or sound, and it is possible to improvise in real time by intertwining different loops. Each piece has a different selection of loops and visual elements, and they are grouped into three different categories. The visual elements changes depending on the loops being played. Similar projects include:

YourSpins by Digimpro

URL: http://www.yourspins.com/
Clock Din by Gord High & friends
URL: http://www.clockdin.com/

Sound Poem by Jörg Piringer
URL: http://joerg.piringer.net/

Sound poem is an example of interactive audio. Users can create different audio patterns through combining different vowels and consonants. It is audio oriented, but users can create different patterns and texture to incorporate in their music. Similar projects include:

The idea of order at Key West reordered by Jim Andrews
URL: http://vispo.com/stevens/index.htm

F8MW9 by Jim Andrews & Margareta Waterman
URL: http://vispo.com/mw/index.htm

Auditorium by Dain Saint & William Stallwood
URL: http://www.playauditorium.com/

Auditorium is an example of game oriented system. Users create music by achieving certain tasks. In game, music works as a background; however, music is the foreground and is main goal of the game. Similar projects include
La Pâte à Son by Frédéric Durieu and Jean-Jacques Birgé

URL: http://www.lecielestbleu.com/HTML/main_pateason.htm

Audiogame.net by Marc Em

URL: http://www.audiogame.net/

Folk song For the Five Points by David Gunn, Alastair Dant, Tom Davis, Victor Gama

URL: http://www.tenement.org/folksongs/

Folk song For the Five Points is an example of system that lets users to explore different sounds and music. Each point represents different location in New York City, and user can create a sonic piece through navigating those locations with a mouse. Similar projects include:

Manchester: Peripheral by the Folk Songs Project

URL: http://www.manchesterperipheral.com/

Cinco Cidades by the Folk Songs Project

URL: http://www.cincocidades.com/

Glass Engine by Phillip Glass and IBM

URL: http://www.philipglass.com/glassengine/
Glass Engine is an example of an interactive music player. Users can explore Phillip Glass’s works by sorting different musical elements of his works. Similar projects that allow users to interact with music library include:

Pandora by Tim Westergren, Tom Conrad, Nolan Gasser... (USA)

http://pandora.com/

Last.fm.

http://www.last.fm/

*Speculation:*

This list above would be a decent reference point for people who wish to develop interactive music systems on the Web in the future. After exploring the different systems, it is apparent that there is a problem that future developers need to solve. All the systems above provide interesting ways for users to interact with music. However, many systems did not gain popularity. They stay as abstract arts or just little toys. This would be another good research opportunity. In order to improve the popularity among people, it would be valuable to research what aspect makes people engaged with systems as more than just toys. Providing an environment that is fun and engaging for first time users and has room for exploration and developing skills is very important for having people continue to use them.
Chapter Six: Overview

Goals and Features:

The goal of this project was to create a Web environment where people can create music by interacting with a Web application on their browser. People can participate in the community by sharing the works that they have composed using the application. People can load other people’s works and modify those as well. Lastly, people can also interact with each other in the community through a commenting and rating system.

The intention of the development was to let users to create different patterns without specifying all the notes. Therefore, only basic knowledge of music such as tempo and division is necessary to use the application. Even if a user does not know about them, he or she should be able to figure it out fairly quickly after playing with various parameters. Overall, the process of generating patterns is pretty simple. The users just specify how overall pictures of the generated patterns should look like. Then, the computer fills in the rest of the details for the user.

The initial template is configured for a passage with a standard rhythm section (keyboard + drums + bass) + lead. Also, it has performance features. Users can save up to 12 patterns per each instrument into memory and instantly recall them later. The most crucial features of the program can be driven by keyboard shortcuts to speed up the process. For advanced users, it has a music shape editor that can transform the generated patterns even further. Also, the output can be routed to an external midi device or an internal midi sequencer using virtual Midi cable. Therefore, the output can be recorded and later edited in greater detail.
**User Interface**

*** Figure: User Interface

![User Interface Interface](image-url)
The figure above is the main user interface. The user interface is quite simple. The program has a menu bar for accessing features that affects in a global level such as play, stop, change midi delay, tempo, and so on. The chord is applied to the global level. The second menu is for the features that affects in a channel level, such as generating one shape, toggle continuous generation, solo, mute, and so on. The next menu is to switch current channel. Switching channels, muting, and soloing can be done in the main window; the actions are also on a separate menu for shortcut purpose. The last menu is for sequencing. It has regular transport function stop, play, previous next, and go. Also, you can add, remove, and insert the current pattern into the sequencer. The toolbar has the essential buttons for fast access.

On the main window, there is global information section on the left, and channel information on the right. The global information displays current tempo, chord, and the last command performed. The tempo button can be clicked to change the current tempo, and the chord button generates a new chord. The channel information displays current name, division, polyphony, density, minimum density, maximum density, leap, low note, high note, low velocity, and high velocity. Each parameter except density can be changed by clicking on the
corresponding button. The density displays the current density, and users can control the density through setting up the minimum and maximum density.

Division controls how one bar is divided into beats. In other words, it controls the shortest note values. Polyphony limits the number of voices. The two densities control the rest vs. note. The greater the value is, the busier it gets. Leap controls the maximum steps that music shape is allowed to skip. It is important to know that one step is one note in a scale, not an interval. For example, if there is a minor third in a current scale, minor third can be treated as one leap since there are no notes between them. The low note and high note controls the minimum and maximum note values that are allowed. The velocities work in a similar way. Finally, at the bottom, there are two tabs: one for music shape editor and one for mixer provided by JMSL. It is possible to control the program with shortcuts. Please look at the shortcut chart below.

*** Table: Shortcuts

<table>
<thead>
<tr>
<th>Channel</th>
<th>Shortcut</th>
<th>Global Shortcut</th>
<th>Command</th>
<th>Global Shortcut</th>
<th>Command</th>
<th>Sequencer Regex</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortcut</td>
<td>Command</td>
<td>Shortcut</td>
<td>Command</td>
<td>Shortcut</td>
<td>Command</td>
<td>Shortcut</td>
<td>Command</td>
</tr>
<tr>
<td>G</td>
<td>Generate pattern</td>
<td>P</td>
<td>Play/Stop</td>
<td>Q</td>
<td>Play/Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl+A</td>
<td>Toggle continuous pattern generation</td>
<td>C</td>
<td>Generate chord</td>
<td>A</td>
<td>Add</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Mute</td>
<td>Ctrl+M</td>
<td>Unmute all</td>
<td>I</td>
<td>Insert</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Solo</td>
<td>Ctrl+S</td>
<td>Unsol all</td>
<td>R</td>
<td>Remove</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>Switch channel 0-10</td>
<td>C</td>
<td>Generate chord</td>
<td>N</td>
<td>Next</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift+1-0</td>
<td>Switch channel 11-20</td>
<td>Ctrl+C</td>
<td>toggle continuous chord generation</td>
<td>V</td>
<td>Previous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------</td>
<td>--------</td>
<td>------------------------------------</td>
<td>---</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift+F1-12</td>
<td>Recall pattern</td>
<td>Ctrl+T</td>
<td>Tempo</td>
<td>Ctrl+G</td>
<td>Go to…</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift+F1-12</td>
<td>Store previous pattern</td>
<td>Ctrl+D</td>
<td>Midi latency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter Seven: Technical Elements

As described in chapter four, the two main runtime programming environments that are suitable for developing an interactive music system are Flash and Java. The main reason that this program was developed in Java is an API called Java Music Specification Language (JMSL: explained later) developed by Phil Burk and Nick Didkovsky. JMSL is the main engine of this project, and it made the process so much easier. It allowed creating a program that would be much harder to implement on other platforms from scratch. The program was first developed as an application instead of as an applet because of the capability to reroute JMSL output to a midi sequencer using MidiShare, a library that enables midi output in Java with an accurate time stamp. This allowed the result to be edited even further on a midi sequencer. However, MidiShare cannot run as an applet. This research led Didkovsky to add JavaSound support for external midi devices, and now JMSL output can be routed through a midi sequencer without MidiShare. Finally the program was successfully able to migrate to Web environment as an applet which can run on any Java enabled Web browser.

Java Music Specification Language (JMSL):

Here is a quick overview of JMSL to understand how this project works. JMSL is an API for Java that supports "hierarchical scheduling of composition objects, sequence generators, distribution functions, music notation, and other music related tools" (Burk & Didkovsky). JMSL also supports Softsynth's JSyn, a synthesizer; Grame’s MidiShare; a native MIDI library supporting MIDI with accurate time stamp, Robert Marsanyi's Java MIDI, another library supporting MIDI In Java, and Sun's JavaSound MIDI, a Java built-in MIDI package (Burk &
Didkovsky). JMSL also supports score related tasks in JScore package, but this project has not utilized the package yet.

JMSL is organized with composable objects which can be scheduled to launch in time. Composable objects are not limited to musical data. Also, those objects can be grouped as collections. Parallel collection launches its all children at once, sequential collection launches its children in a specified order which can be dynamically changed, and queue collection can manage its children in a queuing style (Burk & Didkovsky).

MusicShape represents data values to be played in JMSL, and again it is not just limited to music. Data can be anything that can be expressed as double precision floating points on a table. In musical data, each column might represent duration, pitch, and volume. It is Instrument’s and Interpreter’s job to interpret those data stored in MusicShape. JMSL has its own Instruments that play MusicShape through JSyn or MIDI. The power of JMSL is that people can build custom MusicShapes, Instruments, or Interpreters to fit their needs (Burk & Didkovsky).

Core Design:

The following section loosely follows the Java syntax. Methods are indicated with () after their names, class names start with an upper case letter, and variable names start with a lower case letter. To find out how the features discussed below are implemented in detail, please check out the source codes with comments.

This applet extends the JMSL class called MusicShape in order to expand its capability, and it has a new class called Chord that can generate chords and scales. These two classes form
core of this applet. The functions worth mentioning in the main class are upload and download, which utilizes XML, JavaScript, and PHP to make it possible to download and upload music generated with the applet.

Each channel is represented by a class called AutoMS (AutoMS.java), and each AutoMS also has a MidiInstrument attached that interprets the MusicShape data stored in AutoMS to play through a MIDI device. Then each AutoMS is added to a ParallelCollection called playingCol which groups all the channel and launches them at the same time. One ParalellCollection presents one bar of music, and it can be added to a SequentialCollection called sequenceCol in order to build music. The applet allows sequencing operation such as add, insert, and remove on sequenceCol.

The playingCol repeats forever until the user stops it. Every time each AutoMS gets launched, they go through the following process specified in RepeatPlayable (AutoMSPlayable.java). The implemented method play() calls generate() if generate value is true for each AutoMS. If the continuous (cont) is false, it sets the generate to false after generating a shape. If continuous is true, it leaves it true to allow new shapes to be generated continuously. It also checks if recall value is greater than 0. And Then calls recall() to rebuild the MusicShape from the storage. It also stores the last played music shape into history buffer in case the user wants to store that shape.

When store() is called, it stores the last played MusicShape in tempo 60. When recall() is called, it retrieves the MusicShape from the specified slot and rebuilds it at the current tempo. The modify() changes the notes in the current music shapes so that the notes match a new chord.
The AutoMS (AutoMS.java) has the following class variables which describe how individual MusicShape in AutoMS should look like: division, polyphony, lowNote, highNote, leap, minDensity, maxDensity, lowVel, highVel, and type. Also, AutoMS class has the following static variables which affect the all the MusicShape: tempo, chordRoot, and chordType.

Generate(), the composing algorithm, looks at AutoMS class variables stored in individual AutoMS and builds the MusicShape using the following rule:

- Calculate the smallest duration using tempo and division.
  
  If the tempo was 60, and the division was 8, the shortest duration would be 0.5 second.

- Remove all the data in the current shape to start new.

- If type is 3, build the metronome shape and return.

- If type is 1, get notes for a scale that matches the current chord. If type is 2, get notes for the current chord.

  If type is 1, and chord is C major, it would get all the possible notes for C major scale between lowest and highest notes. If type is 2, it would get all the possible notes for C major chord between lowest and highest notes.

- Get the closest note from the last note played which is stored in this.note.

- Generates a timeline for hit points which lasts one bar by calling getHits().

- Figure out the next hit and randomly chooses the note duration that lasts between current hit to next hit.

- Chooses the number of intervals between 1 and polyphony.
• For each interval, generates a leap from last note and assigns to music shape. Assign current note to last note.
• If the leap is 0, randomly select a note from all possible notes (Percussion).
• If the note duration did not last until the next hit point, fill the gap with rest until the next hit.
• Assign the current hit to the next hit and repeat until the current hit is last division.

The generate() calls getHits() to populate the timeline. The getHits() assigns weights to rest and note using JMSL’s WeightedObjectChooser according to min density and max density. Then it randomly pulls the rest or hit and assigns to the timeline.

For example, if both minDensity and maxDensity had an equal value (0.5), the number of notes and rests populated would be the same. If the division was 8, a result might look like “00110101.” There are 4 notes (1) and 4 rests (0). However, a note might occupy adjacent rests for longer duration.

The Chord class (Chord.java) is all static. It mostly has list of chord notes, chord names, scales, and chord Scale relationship. The Chord.getNote() gets the note list from the list, and returns the only notes belong to the specified chord within the range between min and max note values. The Chord.getName() returns the root name (C, D, Eb…) and the chord name (Mi7, 7b9, Sus4…).
Generation example:

Let us take an example and see how the program would generate the notes. Tempo is 60. Chord root is c. Chord type is major. Here is the setup for our AutoMS.

```java
AutoMS ms = new AutoMS();

ms.setName("Lead");

ms.setDivision(8);

ms.setMinDensity(0.4);

ms.setMaxDensity(0.6);

ms.setLeap(3);

ms.setLowNote(60);

ms.setHighNote(83);

ms.setPolyphony(2);

ms.setType(1);
```

Here are the steps that the generating function would take. It first calculates the smallest duration using the formula $240 / \text{tempo} / \text{division}$. This case would be 0.5 second. It empties the MusicShape. It gets the c major scale between 60 (middle c) and 83 (b above one octave), and it stores as an array. The result would be {60, 62, 64, 65, 67, 69, 71, 72, 74, 76, 77, 79, 81, 83}. If the last note played was 73, 72 would be chosen as the closest note. Since array index starts from
0, the index would be 7. If the last played note is out of the range, it would return a random note index within the range.

Then the function calls getHits() to obtain the array that contains note or rest. The getHits() function chooses a density between 0.4 (minDensity) and 0.6(maxDensity) for this example. Suppose it chose 0.5 as the density and returned the following array, \{1, 0, 0, 1, 1, 0, 1, 0\}. There are total 4 notes and 4 rests. The function generate()loops through the array and assign notes and rests. The slot 0 has a note, so it finds out when the next note would occurs. In this example, it would be slot 3 since slot 1 and slot 2 have a rest. When there are rests between current note and the next note, it randomly chooses a note duration. In this example it would be between eighth and dotted quarter since one slot means eighth. Suppose it chose quarter as the note duration.

If this polyphony is greater than 1, the function will choose how many notes would play simultaneously. For this example, it would choose between one and two. If it chooses 2, it would repeat the following process twice. If this leap is greater than 0, it will choose a leap between -leap and +leap. It would be between -3 and +3 in this example. Then it would adjust the index so that it does not go out of the range.

```java
int maxNoteIndex = notes.size()-1;

int leap = JMSLRandom.choosePlusMinus(this.leap);

noteIndex += leap;

if (noteIndex < 0) noteIndex=0;

if (noteIndex > maxNoteIndex) noteIndex=maxNoteIndex;
```
else noteIndex = JMSLRandom.choose(maxNoteIndex);

    note = notes.get(noteIndex);

    It adds the note to the MusicShape with duration, velocity, and hold time.

    int vel = JMSLRandom.choose(lowVel, highVel);

    if (i == intervals - 1) add(duration, note, vel, duration);

    else add(0, note, vel, duration);

    Finally, if there are still slots with a rest before the slot with a note, it fills the
    MusicShape with a rest. It would be an eighth rest in this example.

    double remain = (nextHit - (hitIndex + length)) * unit;

    if (remain > 0) add(remain, 0, 0, remain);

    // Loading and Saving:

    Loading and saving is achieved through four different technologies: Java Applet,
    JavaScript, PHP, and MySql. The applet uses XML to save and load data by communicating with
    other technologies. When user clicks Save (Random.java->buildGui()), using JSObject the
    button calls validate() (Random.js) method from the JavaScript embedded in HTML. The
    JavaScript retrieves the form data filled by the user on the HTML document, and it passes title,
    description, and keywords to the applet. If everything goes right, the applet calls upload()
(Random.java). The upload() method goes through all the ParallelCollections and AutoMSs added to the SequentialCollection for sequencer and records each variable and its MusicShape to build an XML file. The XML file gets passed to DataOutputStream which passes to HttpURLConnection. HttpURLConnection builds a multipart/form-data and passes it to upload.PHP on the server. Upload.PHP receives the data from the applet, and validates if the user has the right to upload their works. If the validation passes, it saves the XML file on the server directory and inserts the record on the MySQL database as a new post.

Here are the steps for loading music. When user clicks a link of the music he or she wants to listen, show.PHP gets loaded with the id number for the entry specified by the user using get method. Show.PHP retrieves the entry from MySql database and builds an HTML document for the user. If the entry was specified as private, show.PHP presents the user with the form that asks for password. If the user enters a password, show.PHP again validates the password with the database. If the entered password is valid, show.PHP builds a new document and embeds the applet to play the music. Show.PHP retrieves the URL of the XML file that upload.PHP created and passes the URL to the applet to load through <PARAM> tag on the HTML file. I.E. <PARAM name='url' value='http://chikim.com/random/upload/chi2.XML'/>. When the applet gets launched, init() checks if the PARAM tag exists. If it does, it calls download(). Download() retrieves the XML file using HttpURLConnection. Then it goes through the XML file and rebuilds the all the AutoMS and ParallelCollections and, add them to a SequentialCollection, and launch it to play.
**Community Function:**

Besides through sharing their music, there are features that allow users to interact through rating and commenting system. Sharing, rating, and commenting requires registration, and those systems are implemented using PHP, JavaScript, AJAX, and MySql. As explained above, the process is very similar. The user fills out a form, JavaScript validates, PHP process the form by executing MySql functions, and builds new documents. There are two places that used AJAX: username Availability for registration and rating system. Instead of PHP builds a new document, JavaScript passes the user input to PHP, gets response, inserts new DOM element on the current HTML, and updates the document. For example, the user can find out if the user name he or she entered is available or not on the registration page without actually submitting the form.
Chapter Eight: Review

Analysis and Evaluation:

Let us now apply the classification devised by Rowe from Chapter two to this project. First, this would be a score driven system since it uses a traditional tempo and metric system. Second, this would fall under the generative category since it uses algorithm to generate music patterns without any pre-composed materials. Lastly, this would be categorized as instrument paradigm; it would be perceived as a solo when someone plays the system.

Also let us apply the criteria explained by Chadabe from chapter two. Having interesting and informative response is very subjective; however, the output definitely should contain new, unexpected elements through the random generator. Also, its response should be recognizably related to the performer’s actions since the parameters for composing algorithm are directly specified user. Lastly, its outcome is mutually influential since both the user and the system contribute to the final result.

In order to evaluate how challenging, explorative, engaging, enjoyable, and intuitive the program is, it would require conducting a survey and looking at the statistical data. However, one thing that this program lacks is the intuitiveness. The interface has so many controllers so that first time users would not be able to know how the interface works. It would be much better to represent different values graphically so that users can have both audio and visual feedback. Also, it would be better to create different graphical controllers for the user to understand how different controllers are related to different values. This would create a more intuitive interface for users.
Lastly, let us look at which Web 2.0 concepts defined in Chapter Four this project meets. This is not software that users install on their computers. The program and the Web environment can be constantly updated as users use it. User participation is definitely one of the key concepts for this project, and it would grow as more users participate. This project is open to public as an open source project, and anyone who wishes to participate in developing further is welcome. One of the strengths of Java applets are platform independent, and any Web browser with Java support would be able to run this applet.

*Improvement and Future:*

There are still some technical glitches to be improved. Even though it pulls up scales for specified chord, sometimes notes sound wrong, due to the fact that the program does not count the chord scale relationship in terms of the key that it is in. In fact, the concept of key is not implemented yet. Also, the keyboard voicing does not have smooth transition, and it does not have the rules for low interval limit. Therefore, even though it plays perfectly valid chord notes, it could sound unpleasant when the notes are clustered especially in low range.

This project definitely has much room for improvements, so further collaboration for development would be crucial. Especially, it has room for improvement in the graphic user interface and the composing algorithm to generate more interesting patterns. Also, it could include JSyn synthesis and audio samples to expand the timbral contents. Other future implements might include midi support where the program can be controlled by midi, including chord generation and firing some of the actions. Although, too much control can eliminate the fun from average users, it could also implement other parameters that can affect the generation.
This project took a spin from the original plan, which was a standalone application to produce computer generated music, but it now is deployed on the Web as a more accessible applet. Overall, it is believed to be a success in terms of usability and features.
Chapter Nine: Conclusion

Summary

In Chapter Two, we looked at the overview of interactive music through exploring system design. When developing an interactive music system, it is important to first define its goal. Looking at the different types of systems and selecting which aspects to include into the system would help as a starting point. Once the goal is defined, developers would want to break down the system into different components (controllers, sound generators, and programmable device), and define their relationship to each other.

In Chapter Three, we explored different Web technologies that are available to us. Each technology offers a unique feature, and they can be incorporated together to make Web environment more transparent for users. Chapter Three also discussed how a website can be broken down into three different components in terms of their functions: user interface, logic, and database. Each layer interacts with each other, and it is crucial to establish a well-organized communication line between them. Web 2.0 is the new generation of Web environment, and it combines server side scripting, database, client side scripting, plug-ins, AJAX, and so on. As described in chapter three, Web 2.0 is not just a technology. It is a new business model for companies, and new experience for the Internet users.

In Chapter Four, we explored how to deploy interactive music systems on the Web. There are different limitations of the Internet, and it is important to develop a system that would work with those limitations. Developers need to design controllers and sound generators according to the limited selection of devices. It is important to comply with the different security restrictions for browsers. In addition, when connecting different technologies through JavaScript, it is
important to consider the compatibility for different browsers since each browser handles JavaScript differently. When the system depends on the bandwidth, it is important to balance bandwidth vs. features. Lastly, the latency over the network needs to be factored into the system design.

In Chapter Five, we explored different interactive music systems that are currently available on the Web. There are many different types of systems, and the range of complexity is very wide: from a very simple system to complicated system exploiting network interactions. The list of the programs would serve as a good reference for people who wish to develop an interactive music system on the Web in the future.

In Chapter Six through Eight, we looked at how an actual project can be implemented on the Web. The project used different technologies surveyed in chapter three such as PHP, MySql, Java, JavaScript, XML, Css, AJAX, and so on. It is still on-going project, and everyone is welcome to participate and contribute to the project in order to make it better.

**Future, Direction, Conclusion**

Interactive music is still at its infancy as an art, and it is only known to a relatively small group of people. People such as Joel Chadabe, Robert Rowe, and Tod Winkler immensely contributed to interactive music in an academic setting, and they provided a solid foundation for other successors to build upon their researches and experiments. It is now time for interactive music to go toward the mainstream in order to appeal to a wider audience.

There are a few routes that interactive music can take in order to move forward. First, developers may incorporate more elements from popular music to their systems. Since many
people are already familiar with pop music, they would have easier time to connect and engage with interactive music. Second, developers may collaborate more with visual artists who would like to experiment to expand their arts. This would help interactive music to branch into a wider genre as well as gain more publicity.

Third, developers may come up with ways to incorporate interactive music in different applications. The applications that could be further explored for greater potentials include interactive music in game (not adaptive music), and interactive player for music library. The latter case provides an opportunity to collaborate with researchers from music information retrieval field as well. Lastly, interactive music can explore more of educational aspects for children. This would allow interactive music to gain exposures not only to children but also educators and parents.

The internet is growing every second at a fast rate. It has a huge impact on modern society, and everyone is affected by it to a certain extent. The internet has a huge potential for multimedia for the mass; therefore, this is a perfect time to invest in interactive music and let it flourish on the Internet. When interactive music matures and becomes more accessible to people, it would have benefits for many people.
Bibliography


From Hypertext to the World Wide Web. (2008, April 15). Retrieved March 1, 2009, from Computer Science @ Wellesley College: http://cs.wellesley.edu/~cs215/Lectures/L00-HistoryHypermedia/FromHypertextToWWW.HTML


