Using Mobile Devices to Create A Shared Music Experience

Danielle Penny, *Course 6-3, MIT Class of 2017*

***Abstract*—Tutti is a web app, used primarily on mobile, that aims to allow a large crowd to perform a pre-composed musical piece together. Taking inspiration from the MIT Engineers song and MIT style of collaboration, this app assigns each person a varying role throughout the piece which dictates which parts of the music they perform through their phones audio speakers. Users will use a simple interface and interaction methods such as pushing to hold to create unique motifs that they can hear change throughout the piece. A graphics client displays a visualization of all the clients interactions with the piece, while a conductor client controls when the piece starts and stops. The development of this application explores the ideas of interactive music, developing an interface to encourage creativity and exploration of the music, and how to develop a system architecture that best serves the musical architecture of the piece.**

Keywords*—*Interactive Music, Mobile Apps.

I. Introduction

What if we could use our phones to create a shared musical experience? The goal of this project is to create a web application where people can connect and create a real time shared musical performance by each person's phone becoming an 'instrument' that plays a role in a piece of music. Evan Ziporyn from the Music Department composed a piece 'Engineered Engineers' that was the foundation for creating parts and an interface to best represent players' roles in this piece. The app, named Tutti, is planned to be used at the MIT Campaign for a Better World Events, starting in LA on February 7, where crows of 50-200 people will each use their phones to play some of the notes from the piece, similar to how instruments of an orchestra have different parts that play together to create the piece. The piece is based on the MIT Engineers song, and has a few rhythmic motifs that served as the backbone from which the client structure was designed.

A. Previous Work

As technology and smartphones become increasingly prevalent in day to day life, we see a rise in use of these in interactive music. Interactive music has been used in a variety of contexts, from therapy to educational technology to performances, and in particular, as a means for enabling audience interaction.

Some research, such as Choi et all, focuses on the smartphones abilities to detect different movements vs the actual way it produces music. Their paper explores the phones built-in tri-axis accelerometer and its ability to detect shaking and movement-based gestures. Fabiani, Bresin, and Dubus created MoodifierLive, a smartphone application that is used in music performance, but uses rule based automatic musical sounds. It attempts to study how a system can input gestures, and use those to interpret emotions, and match the outputted music to reflect those emotions.

The Stanford Mobile Phone Orchestra studied social music interaction and in particular, how to best enable audience participation. They note the importance of having the audience feel like they have a purpose, and how the social aspect of a performance can encourage participation, especially in large audiences. Performed in 2010, their interface primarily used HTML and AJAX, and lacked the sophisticated music libraries that exist today (ie WebAudio, MIDI.js, etc).

In a chamber music and audience piece Glimmer, Jason Freeman highlights the challenges of empowering audience members, noting, But could the work ever make all 600 audience members feel truly indispensable to its performance? Large-audience participatory works cannot promise instant gratification: giving each person a critical role; requiring no degree of experience, skill, or talent; and creating a unified result which satisfies everyone. This project seeks to create a web application that gives audience members a sense of ownership over the music, as well as a closeness to MIT.

II. Critical Design Features

A. Fault-tolerance

From a technical standpoint, one of the most important features is that the server is fail-proof. The server must be fault tolerant, and able to handle varying loads up to around 200 people. To minimize lag, we minimize the work the server has to do by putting most of the work on the clients. One of the main technical challenges was designing a system that is complex enough to allow for multiple client roles and role switching, while also minimizing the work done by the server.

B. Usability and Interface Compatibility with the Music

From a design and experience perspective, the audience must enjoy the musical experience, and feel ownership over which parts of music they create. The interface was designed to be simple enough so that people can easily learn the controls, and easy for audience members to manipulate the sounds they produce without requiring a music background. The degrees of freedom given to each person was tested thoroughly to allow for the optimization of ownership and quality of the performed piece. We want to give people a sense of ownership where they feel they can control the sound they create, while also making sure that the piece sounds good as a whole.

In one of the initial prototypes, we gave audience members total control over what notes they play, by supplying them with a range of MIDI notes and instruments. While this was a fun experience if it were just one person hearing their music, it became clear that a roomful of people making unrelated melodies would not sound like they were part of the same performance, instead sounding disjointed and cacophonous.

On the other end, another prototype let each person play an exact part from the composed piece, and no matter what they pressed, the output sounded exactly like their part. This guaranteed the summed output sound exactly like the composed piece, but did not give people the feeling that their interaction with the piece mattered. The final evolution provided players with three buttons, each corresponding to a pre-composed part (played as a WAV file). Audience members could toggle these parts on and off.

III. Technical Stack and Architecture

The main components are an all-knowing server, several clients (who are only aware of their own music, and

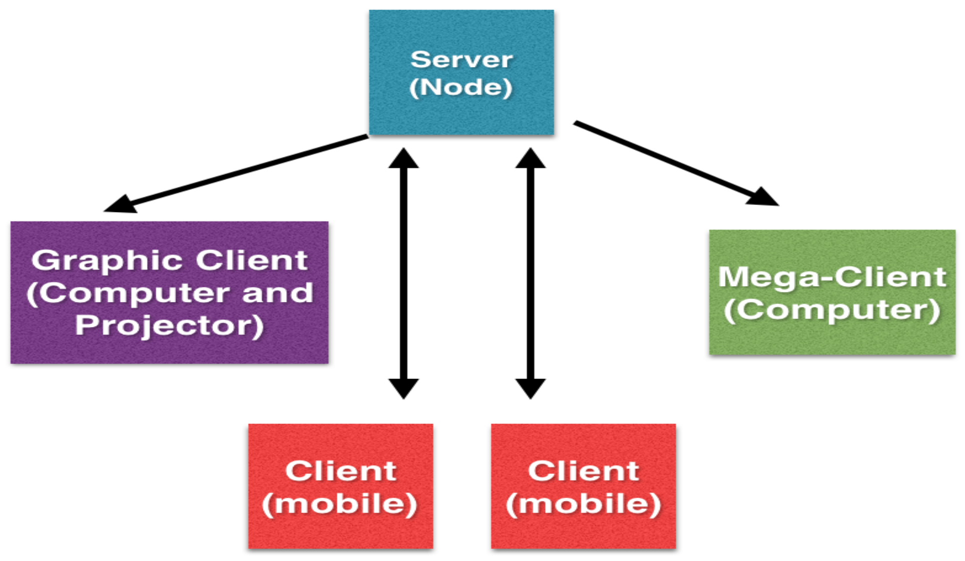


Fig. 1. Block Diagram of Major Components.

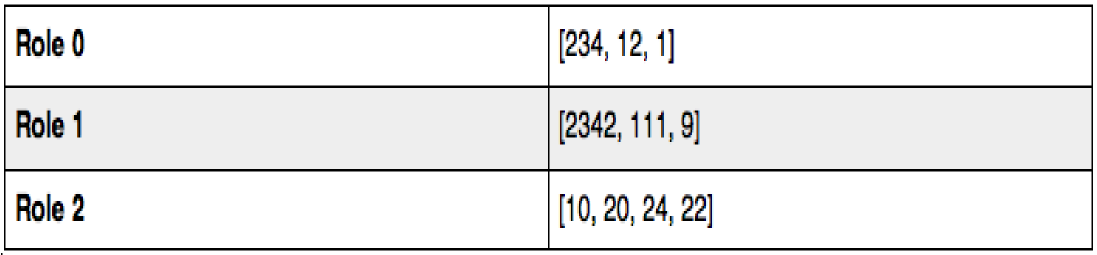
connect via a website on their phone), a conductor, and a ’mega client’. While each client only has a small part of the music to play, there is a mega client that plays the parts of the piece that do not sound good on phones (notes below a certain frequency) on a high quality set of speakers, provided at the venue. A ’Graphic Client’ (a computer and projector) provides a visualization of the piece as a whole (still under development). The phone interface responds to user inputs with simple graphic changes (ie the button changes color), but the graphics client responds to all user inputs with simple animations. Completing the graphics client is beyond the score of the current project, but will be finished in time for the first performance.

The audio is generated using ToneJS, which uses the Web Auido API, and is primarily playing different WAV files that each client receives upon connection. Each role is represented by a different WAV file. The web interface is created using Semantic UI.

IV. Interaction Between Components

A. Server

The server keeps track of the piece in terms of current time, distribution of

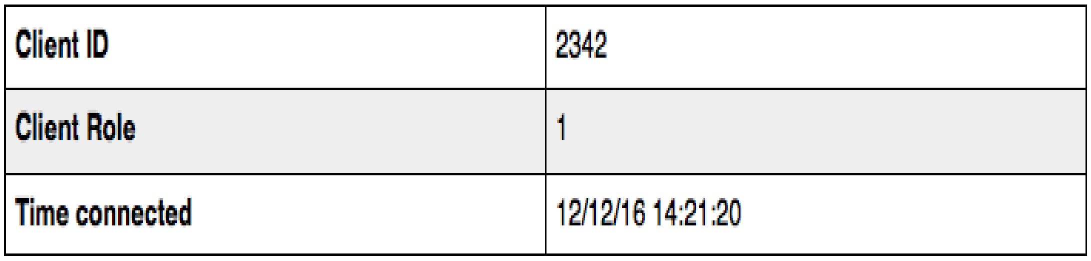


Role 0	[234, 12, 1]
Role 1	[2342, 111, 9]
Role 2	[10, 20, 24, 22]

Fig. 2. The server keeps track of how many clients are assigned to each role. Depending on when in the piece the system is, there will be different ratios of people assigned to each role. For example, at the beginning there are only chords, so most people will be role 0. When the eighth note (faster) rhythmic patterns are introduced, some people will be reassigned to role 1

parts among clients, when a client interacts with the website, and clients assigned to each part. The server runs on the Node.js platform, and multi client logic is done in socket.io. The app runs on an instance of an Express server, which is continually listening for client connections using Socket.io. Every time a client connects, they are assigned an ID, and a role (which tells them which parts of the piece they can play). The conductor can issue a play command, which emits a ready to start signal. Once the server receives this message, it relays to all the clients that their audio can begin playing. Before this point, clients can be connected to the webpage and see (and interact with) the interface, but do not produce sound.

The megaclient, or the conductor, dictates when the piece starts. The conductor enters through his own webpage, and when he presses start, all connected clients receive a message that the piece has begun, and their phone can start creating sounds. Once the conductor presses start the global variable measure starts incrementing. Measures are the units of measurements/time used in music, to dictate the current position in a piece of music, relative to the number of beats that have passed. For example, if the pieces tempo is 60 beats per minute, has 4 beats per measure, and starts at 11:11:20, at 11:11:24, it will be at measure 1, at 11:12:20 it will be measure 15, etc.



Client ID	2342
Client Role	1
Time connected	12/12/16 14:21:20

Fig. 3. Information stored by the server when a client connects

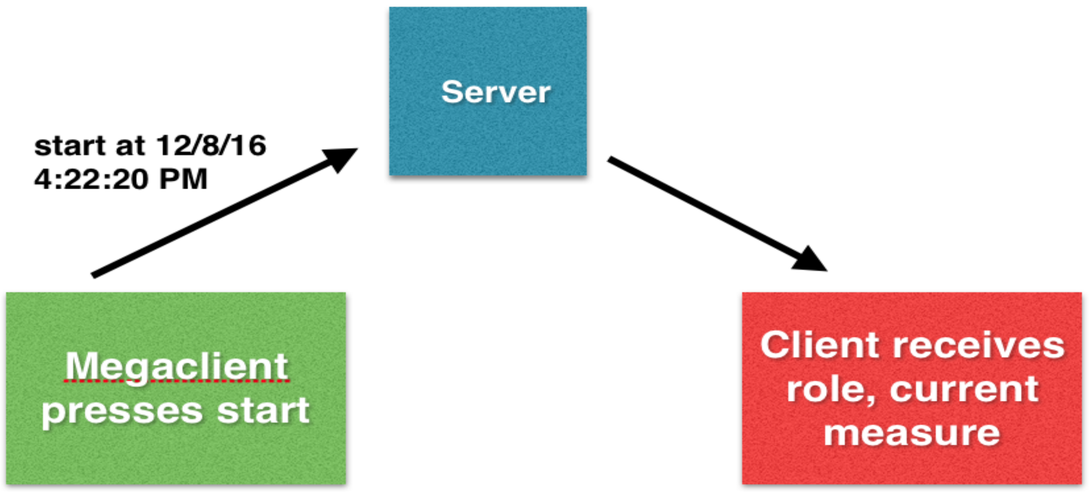

Server
start at 12/8/16 4:22:20 PM
Megaclient presses start
Client receives role, current measure

Fig. 4. The megaclient conductor relays a ready to start message to the server, which relays a start message to the clients with the current measure

B. Client Connection

Clients can connect before the piece begins, or during the piece. A client connecting before the conductor presses start is not allowed to make a sound until the piece starts. However, if a client connects in the middle of the piece (after the conductor starts it), it receives information with the current measure from the server, and is therefore able to play exactly in time. Regardless of when a client connects, it is assigned a random role in the piece that switches every 16 measures. Updates are sent from the server to the client every 16th note (roughly every 104 ms). These updates inform all clients which measure theyre currently on, and these are synced across all clients.

C. *Client-side logic*

Client-Side Logic When the client opens the webpage, all WAV files will be loaded at once, even if they do not correspond to the role the client was assigned. Originally, clients were going to receive batches of WAV files in updates from the server each time the clients role changed. While this would decrease the initial load time, as clients would have to load less WAV files on the initial page load, ultimately this was not the optimal structure for the system. This is because this would greatly increase the time needed for the clients to process the updates. Instead, almost all WAV file-related logic will be done on the client side, reducing load on the server. All the server gives in its updates is the current measure, if the piece has started, the time the piece was started, and the clients roles (as integers), which correspond to array indices of which WAV file players should be played on the client side.

D. *Client Interface*

All clients will see a simple interface of three different colored buttons that fill their screens. Each will be push to hold, and correspond to a different WAV file (their assigned role will determine which WAV files these are). The position of the button also holds significance. For example, in some roles it might be that the top button corresponds to a part with the highest pitches, and the bottom part has the WAV file with the lowest pitches. Or, it might be that the top button has the part with the fastest rhythms (ie sixteenth notes), and the bottom has the slowest rhythms (ie eighth notes).

V. Next Steps

This is an ongoing project that I plan on continuing to pursue even after UAP ends. Within the next few weeks there will be a load test to see see how the system acts with a large number of users (100). The main factors to look at here are:

- Is it an acceptable load time?
- Is the 'tutorial' screen clear enough? (to be implemented)
- What does the sound experience sound like hearing so many other

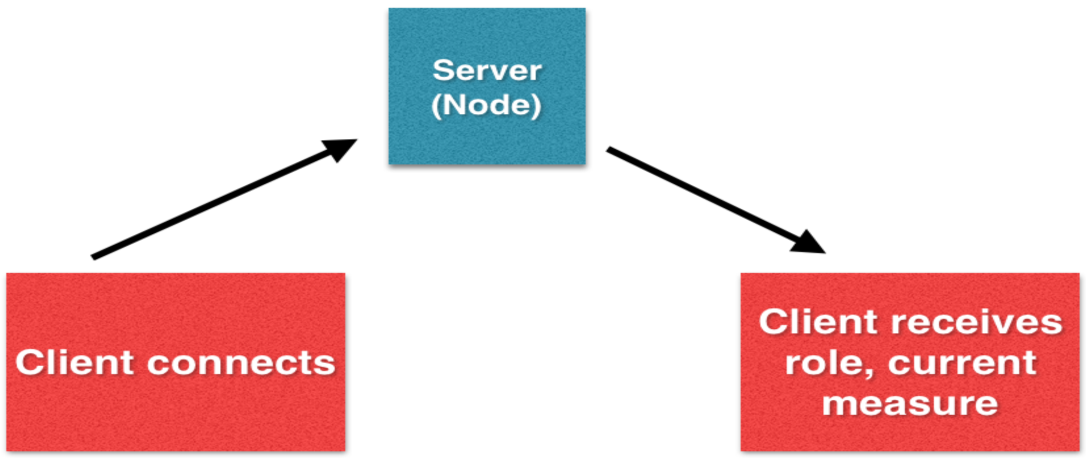


Fig. 5. A client connects.

devices along the one you are controlling?

- Can it load on different devices?
- Can you tell what is your part? Is everything in sync?

The next major component that needs work is the graphics client. This is the interface that will visualize the clients interactions with the system. For example, every time a client presses a button, an explosion graphic appears on the screen. This visualization will probably be displayed on a big projector at the front of the room during the event, and will be a fun way for audience members to feel another sense of participation in a group, as they can see other clients music even if they cannot hear it. The actual responsiveness and accuracy is probably not important however, because there will be so many interactions that even if the graphics are displayed randomly, there will be no way for people to tell it is inaccurate. The most important feature here is probably that it looks fun and engaging, and that it is obvious to the audience that this visualization represents the audience themselves and how they interact with the app.

VI. Conclusion

While there were challenges present that are common in most technical projects, such as accuracy, fault tolerance, user interface, many of the more interesting problems came from developing an app that comes at the intersection of music and

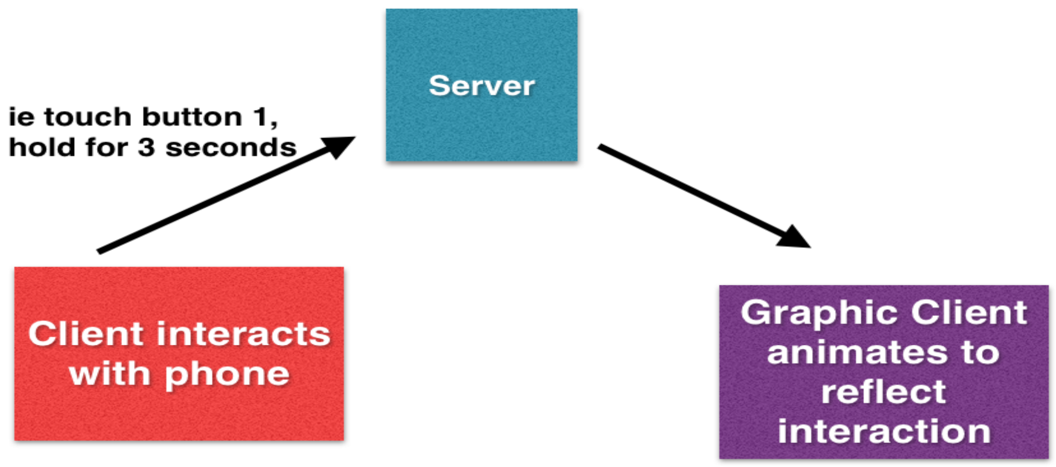

Server
ie touch button 1, hold for 3 seconds
Client interacts with phone
Graphic Client animates to reflect interaction

Fig. 6. A client interacting with the phone will relay a simple message to the graphics client, which will produce an animation.

technology, specifically when the users are not necessarily familiar with music. Because of the repetitive, motif-filled nature of the piece, it was important to develop a way of playing audio that used these repeats to its advantage, to save buffer space and load time. In addition, because of these motifs, roles were developed in such a way that would highlight these motifs. Similarly, a simple user interface was developed with push to hold functionality that would allow the user to focus on the music, instead of the interactions with the website, since these interactions are ideally easy to learn and easy to do. This project allowed me to explore two passions, music and computer science. It was exciting and challenging to get to build something from scratch. Applying computer science to different fields is always a learning experience, and poses new sets of challenges that one doesnt encounter normally. This project highlighted the importance of designing a system to best match its use case, and how to design an interface that will be the most helpful to achieving the goal of the product. I hope Tutti will bring the joy of music to those who use it, and will serve as another example of interactive music systems and the positive roles they can play in todays society.

VII. Score Fragment

Figure 7 depicts a sample of the score that was the basis for this piece. Sibelius was used to convert the parts into WAV files. A, B, and C show different rhythmic motifs, that translate into different parts. Part A is the solo motif, which is the recognizable 'we are the engineers' tune. Part B is the chord bassline, which is divided into top, middle, and bottom elements (which correspond to different buttons). Part C is a faster rhythmic motif, which is part of another role.

VIII. Code

This project's code is available at github.com/dpenny/Tutti

Acknowledgment

Thank you to my advisor Eran Egozy for amazing mentorship, advice, and for introducing me to this opportunity. Thank you to Evan Ziporyn for composing such a great piece, and providing advice and teaching me about the composition process.

References

[1] Choi, Eun-Seok. "2005 IEEE International Conference on Industrial Technology." IEEE Xplore - Conference Table of Contents. 2005 IEEE International Conference on Industrial Technology, 14 Dec. 2005. Web. 13 Dec. 2016.

[2] Chupka, Zachary. "INTERACTIVE MUSIC SYSTEMS." WPI (2014): n. pag. Print.

[3] Fabiani, Marco, and Roberto Bresin. "Moodifier-Live: Interactive and Collaborative Expressive Music Performance on Mobile Devices." Conference: Proceedings of the International Conference on New Interfaces for Musical Expression (n.d.): n. pag. Web.

[4] Lee, Sang Won. "Echobo : A Mobile Music Instrument Designed for Audience To Play." Music and Acoustics (2013): 65-113. Nime.org. Web.

[5] Oh, Jieun, and Ge Wang. AUDIENCE-PARTICIPATION TECHNIQUES BASED ON SOCIAL MOBILE COMPUTING (n.d.): n. pag. Stanford. Web.

[6] Weitzner, Nathan, Jason Freeman, and Stephen Garrett. "MassMobile an Audience Participation Framework." (n.d.): n. pag. Nime. UMich EECS. Web.

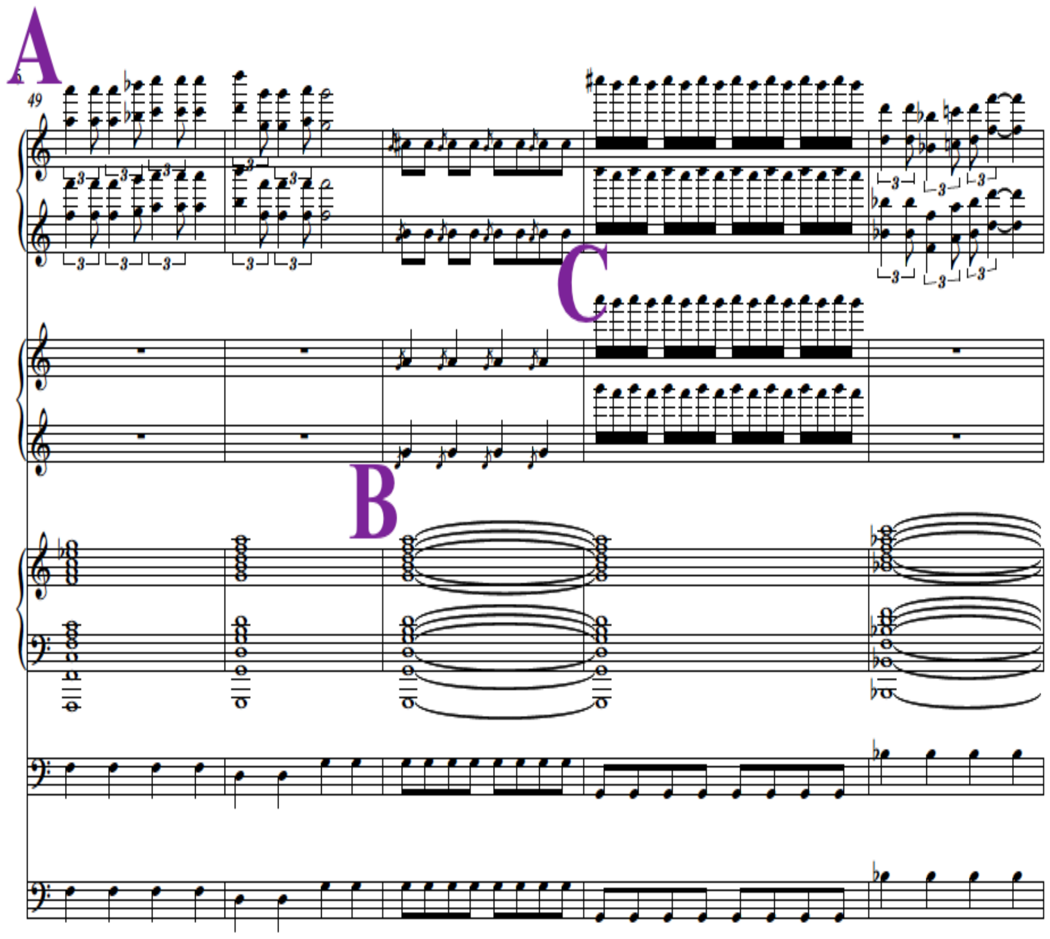


Fig. 7. A sample of the score. See section VII for more details