Bandit! Final Report

Prepared by: Connie Kwan, Harvey Abaya, John Jang, Jonathan Nguy, Rhys Yu

March 20, 2013

Table of Contents

Abstract

This paper is an overview about the parts and the development process in constructing a mobile application that utilizes bluetooth communication for a class: Physical Layer CS 117, taught by professor Mario Gerla.

BandIt is an Android application that allows users to play musical instruments with their friends via Bluetooth. We want to give players the ability to choose and play from a variety of instruments. Players would also be given the option to play by themselves or play together with other nearby players who have the same app.

Introduction

With the prevalent use of mobile devices, it is a brilliant idea to utilize the portability of these devices to supplant non-conveyable objects. An example would be musical instruments.

Music has always been enjoyed by everyone. Making music, is even much more exciting. This thought has sparked an idea in the minds of our team members and have provided motivation to create an application that uses mobile devices to create music. Getting together and creating a mini band is a favorite hobby that musicians celebrate with their friends. Since musical instruments could be quite cumbersome to carry around all the time, why not magically convert these bulky instruments into something that can be hidden in our pockets. That is where the mobile application, Band it, comes into play.

Band-It app should be able to provide the users with choices for a musical instrument to play. Users should be able to play their instrument by tapping on the screen of their mobile devices. After running the application, users will first be presented with the main screen. In here they could select either multiplayer or solo play mode. After selecting a mode they should be presented with a screen with which they could select their instrument.

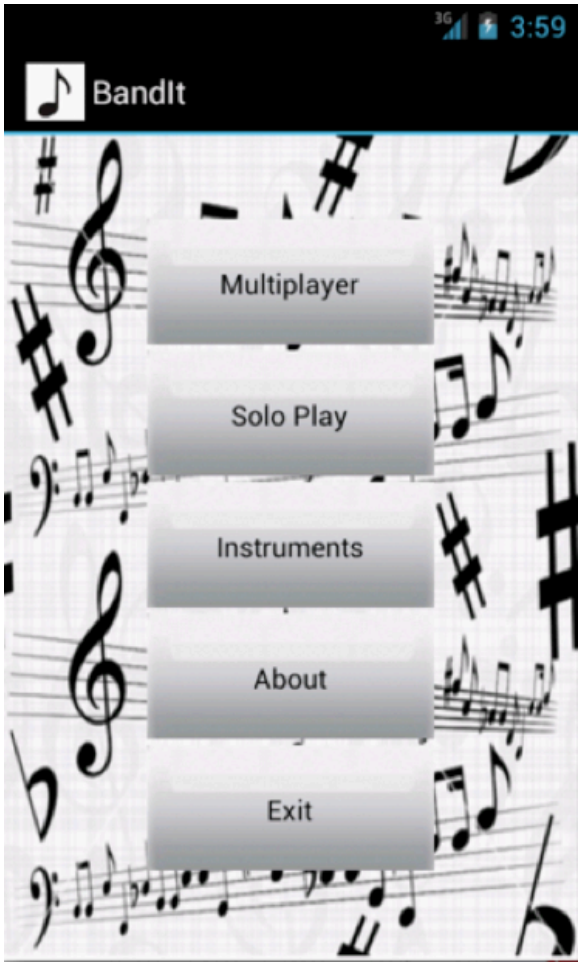
One of the main feature of this app is to deliver interactivity among players. One of the main objectives of this app is to have at least 2 people involved. Each player is able to

access the application on their own devices. These shall provide their own interface and be able to enjoy the music making collectively as a group. This is achieved by utilizing bluetooth as the communication layer. Through this, players could connect with each other and be able to hear

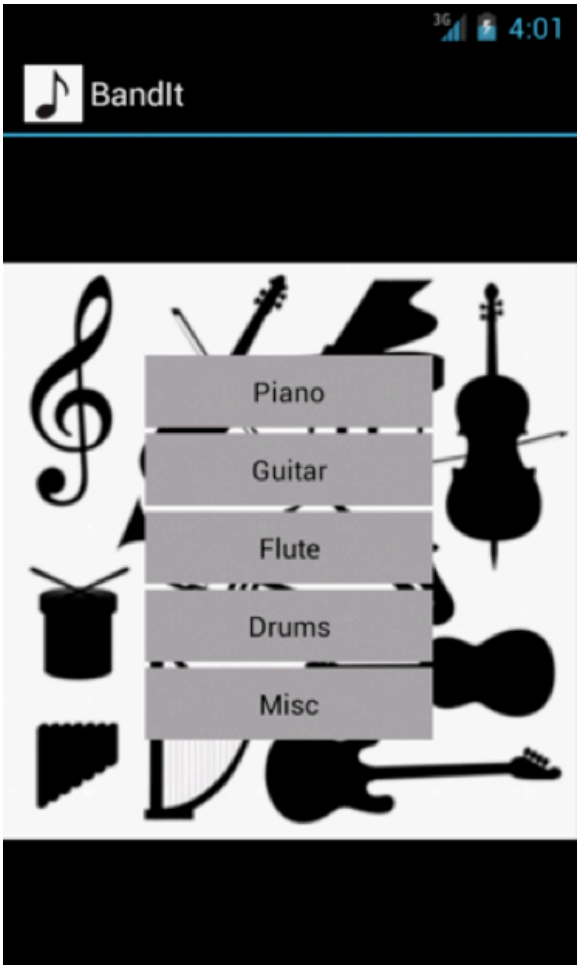
what they play and what their friends are playing all at the same time. This makes it more enjoyable.

Although, Band-it could obviously not be even close to replacing these instruments. This app is essentially just for entertainment that could provide fun-filled enjoyment between friends utilizing their mobile devices.

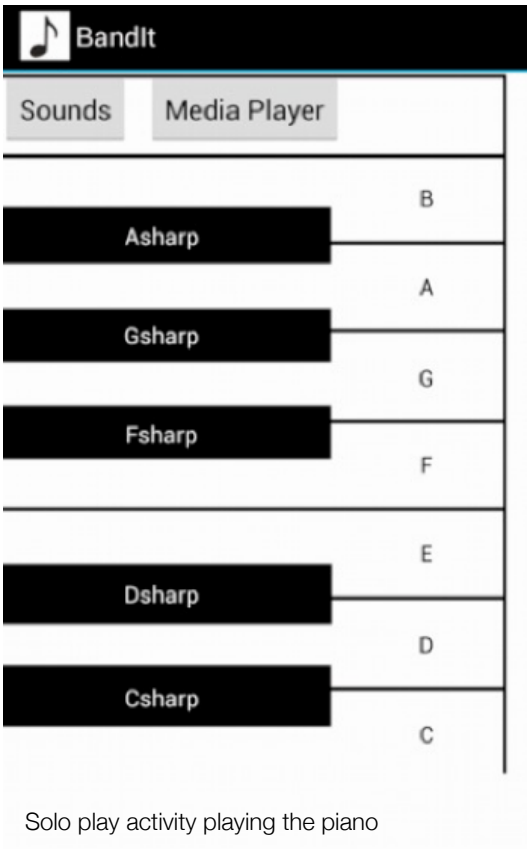
Screenshots



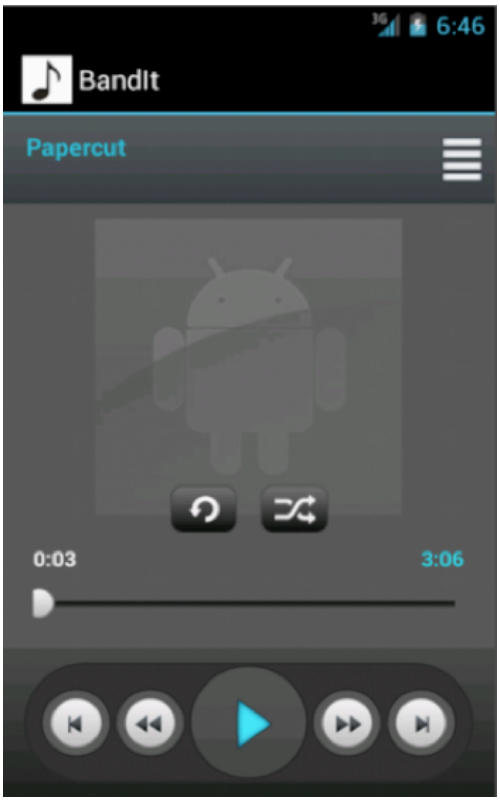
Main activity



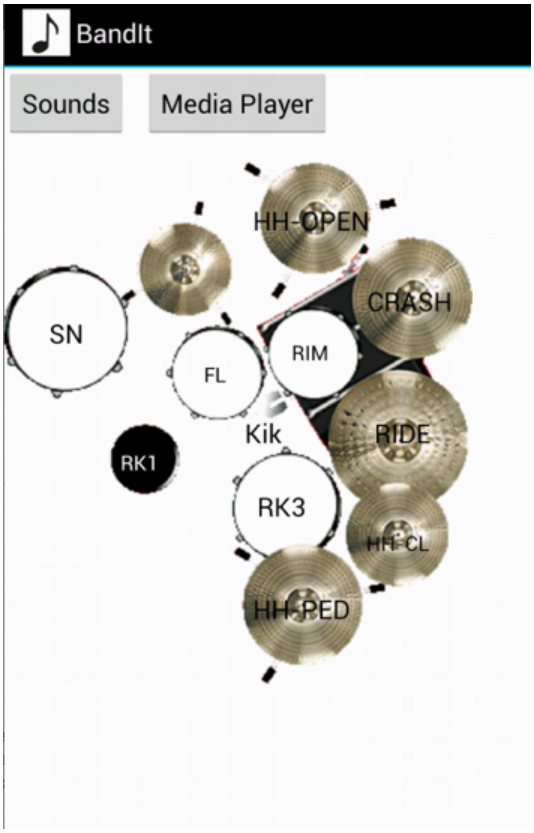
Instrument list activity



Solo play activity playing the piano



Media player activity



Drums activity inside solo play

Connie Kwan, Harvey Abaya, John Jang, Jonathan Nguy, Rhys Yu

Contributions

Rhys Yu

Main menu – designed the main menu layout and implemented the code, both for the graphical component (.xml) and functionality component (.java).

Instruments menu – designed the instruments menu layout and implemented the code, both for the graphical component (.xml) and functionality component (.java).

Solo Play activity – designed the Solo Play layout and implemented the code, both for the graphical component (.xml) and functionality component (.java).

Multiplayer activity – designed the Multiplayer layout and implemented the code, both for the graphical component (.xml) and functionality component (.java). The difference between Multiplayer activity and Solo Play activity lies solely in the Multiplayer activity's ability to use Bluetooth.

Bluetooth component – researched how Bluetooth works by reading up in Android forums, documentations from Google and making sense of the sample BluetoothChat code provided by Google. Also responsible for integrating, coding and testing the Multiplayer activity to use Bluetooth.

Piano instrument activity – designed the piano instrument layout and implemented the code, both for the graphical component (.xml) and functionality component (.java). Also used SoundPool to generate Piano sounds.

Piano code inside other activities – responsible for designing and integrating piano instrument sounds and functionality to the code from Solo Play and Multiplayer activities.

Main Structure and Intents – responsible for designing how the app's activity states work and as it changes from one activity to another, how the intents and intent flags work.

Task Management, Delegation, Lead Design – responsible for task management and delegation, lead design decisions and keeping up with project milestones.

Harvey Abaya

Drums Instrument Activity -- One of the instruments that could be selected by users is the Drums. When a user selects the drum instrument, he/she shall be presented with this screen below: I am in charge of displaying the drums correctly and providing the proper functionality for each button. Each drum piece is clickable, by tapping the button. After tapping each drum piece the appropriate sound should play. First, we needed a collection of sound files. The sound files must be of .ogg extension, because we use the soundpool API provided by the Android Development Kit, and it only works with .ogg extensions apparently. The soundpool will load all these sound clips and provide an ID value for each sound file.

Drums Layout -- The layout should then display the drum image on the background. A collection of buttons will be placed on the image. They must be correctly positioned over the drum pieces. The buttons will then have an action to play a sound file by using SoundPool to play the sound by using the id values of the loaded sound clips. Depending on which drum piece to play, once the button is tapped, the appropriate sound will play. Thus giving the drum-like instrument in the mobile device.

Game Design and Architecture -- I contributed in coming up with the overall idea of building the music app. Part of it was designing how the app will run and what are needed to build the mobile application. Aside from that, I also the high level architecture of how the communication for the multiplayer will be handled.

Bluetooth Socket Programming Development -- I also helped Rhys in getting the Bluetooth communication set-up for the application. One of the open-source programs that is part of the Android Development Kit samples is the Bluetooth Chat programming. Since the bluetooth chat somewhat resembles the desired bluetooth connection that is needed for the app. The bluetooth portion of the app was built on top of the bluetooth chat application, since it is easier to use high level functions instead.

Jonathan Nguy

Integration Insight -- My task was to combine all the instruments we implemented into Solo Play and Multi Play. Once my partners got their instruments working in the Instruments List, my goal was to get all those instruments working in Solo and Multi Play. We used Instruments list to mostly test each instrument to make sure they worked before we included them into the more important

parts of the game. First, to debug each instrument, each had its own activity so we could narrow down the problems if we had any. Once they were tested enough, I got the code and put it into Solo Play and Multi Play.

Solo Play -- The goal of Solo Play was to get the instruments working in one activity, so that we could save time swapping activities when the user wanted to play with a different instrument. To achieve this, I incorporated all the instruments into one activity and made the interaction work when swapping instruments. This included incorporating all the sounds and buttons into one activity and making the sounds play correctly according to which instrument was selected.

Multi Play -- Multi Play was a bit more difficult. Since there were multiple devices working together, we had to implement in a way that it wouldn't need to send sounds over Bluetooth, because that would take too much bandwidth and may cause significant delays. We decided to implement this so that we instead send numerical values, and then each client will decipher that number and then play the song according to what sound was transmitted. Also, the reason why we had to integrate them all together into one is that we had to face the problem that the Bluetooth connection would have to be reconnected every time there was an instrument switch. Since we wanted to avoid this, we had to integrate all the instruments into one activity. The implementation for Multi Play was definitely more difficult because of the number encoding.

Reports and Presentations -- I organized the reports and created the presentations that we used for the class.

Connie Kwan

Create layout and integrate sounds controls for flute -- My task was to create another instrument for our Band It application -- flute. Creating another music instrument was not easy in the beginning. After continuous research on various forums and websites and exchanging information between teammates, I had finally came up with a layout using .xml, .java file for an additional activity, multiple .xml files for additional styles and strings.

Media Player activity inside Solo Play and Multiplayer – worked together with John Jang in integrating the Media Player activity code to the app. This activity is taken from AndroidHive.info and menu was added to give users an option to go back to previous activity as opposed to using the physical defective 'back' button provided and force closing the whole app.

Troubleshoot with issues -- When we are working on the eclipse platform, there are a few issues that we have encountered:

a.) How to get to set the IDs correctly so that we can get R.java correctly

- IDs are not set correctly in .xml files for layout so that R.java fails to build.

b.) Unable to see sound files after adding into the drawable directories

- refresh is needed whenever we modify the file externally.

c.) The flute activity is unexpectedly forced to shut down

- the error does not really indicate what happened to the application until reading LogCat that tells you possible errors and find that we haven't manifested the flute activity yet.

d.) Trouble with running on some of the platforms on AVD - the application needs to run on Google API in order to be shown on the emulator.

There are some more things which I want to implement, for example multiple touches on the screen for a realistic playing experience, but due to time constraints, we will leave it as the future development of the application. Overall, I tried not to get help from others so that I can learn more throughout the process.

John Jang

Media Player activity inside Solo Play and Multiplayer – worked together with Connie Kwan in integrating the Media Player activity code to the app. This activity is taken from AndroidHive.info and menu was added to give users an option to go back to previous activity as opposed to using the physical defective 'back' button provided and force closing the whole app.

Audio Sample Collection -- The first step to building the musical instruments used in BANDit was to collect the audio samples that would be triggered with the press of a button, e.g. a key on the piano keyboard, a drum on the drum kit, etc. To accomplish this, the software instruments contained within Apple's Garage Band were used as instrument plug-ins within Logic Studio 9. After loading the different instrument timbres, each note of the different instruments was first recorded as MIDI. Because MIDI itself is not audio, the recorded midi had to be re-recorded as .wav audio files. These were then imported into the website http://audio.online-convert.com/convert-to-ogg for the final conversion process to .ogg files, which was the only audio file type supported in our implementation of the instruments.

Guitar Instrument Activity -- Upon collecting the audio samples, John implemented the guitar instrument interface, and its selection button in Solo Play mode. It was initially planned to control the guitar in a similar manner that a real guitar would be played - with multi-touch enabled frets for chord tabulators and strumming for the strings. However, time did not allow us to delve deeper into figuring out multi-touch control on the smartphones, and we simply resorted to piano-like layout of buttons for controlling the guitar instrument. The audio files for the guitar were recorded as chords however, to give it a more realistic strum of a real guitar.

The buttons on the guitar were placed using relative positioning, which allowed the programmer to position the buttons relative to the top and left edges of the screen. As it was decided to implement a keyboard-like buttons that would be spaced out evenly, this was the more logical choice. In creating the guitar instrument interface, however, an unexpected error persisted when attempting to run the instrument in the emulator which kept on crashing the application. It turned out that this was due to the file size of the background .png photograph file being too large, which was causing memory to run out.

Future Works

Instruments

Our goal is to eventually add all the instruments we can. This can include instruments like the banjo and the cello. It wouldn't take too much time to implement new instruments, but we would have to grab more sounds and create new layouts for them.

Layouts

Many of our layouts were pretty boring and very blocky. If we had a photoshop expert, we could have made our instruments look more professional and more lifelike. If we were to spend more time with this, it is possible to make the instruments just overall better looking.

Scaling Layouts

One problem we faced was scaling layouts due to the many sizes of Android devices. Since our button placements were somewhat specific, it was difficult to accommodate for all the different sizes. In the future, we could make sure all the layouts look good on any Android device, regardless of its size.

Multicast

We had trouble getting multicast to work, so we were limited to 1 on 1 connections. If we spent more time with multicast, we could expand this to have more than 2 people playing the game at a given time.

Generate File

Everyone loves recording things. If we spent more time with this, we would possibly have the ability to save the music a user plays, so he or she can replay those sounds after. This is particular useful when playing over a song and with multiple users. If people want to see how well they combined their sounds, this would be a perfect addition.

Multitouch

Since some of our instruments would have different sounds if played with multiple fingers, such as the flute, we could implement this to track multiple finger presses. This was a bit difficult to implement first, so we decided to keep this as a future goal and instead get our instruments working as is.

Enable Wi-Fi

If we were able to enable Wi-Fi, we could remove the requirements and limitations of Bluetooth, and the users would be able to play with anyone who has access to the game. This would be a long term milestone, but it would be an interesting addition to the game.

Conclusion

In the end, our group was successful in implementing a majority of our proposed features, including Bluetooth. Our end product is able to play musical instruments using specially designed layouts while also providing an option to play with nearby players via Bluetooth. It is important to note, however, that we are limited to 1-to-1 Bluetooth connection and are unable to do multicast as we encountered time constraint and understanding problems with it.

As mentioned in the previous section, there are still several potential features we could implement if we had more time.

References

http://developer.android.com/ - Android documentation page

http://developer.android.com/guide/topics/connectivity/bluetooth.html - Android Bluetooth documentation page

www.androidhive.info - MediaPlayer code