3D Audio Expression Box

Capstone Proposal

Team Members:

Victoria Suha

Mark Lang

Jack Davis

Chris Babroski

Nick Knaian

Advisor:

Professor Shafai

Table of Contents

1. Abstract

The purpose behind our project is to create a device which enables the user to intuitively create audio effects in 3D space. At the project's most basic level, we are attempting to map three separate parameters of audio processing to x-y-z axes, the coordinates of which the user is able to set and change fluidly using his or her own hands. Typically, effects are applied using sliders or knobs on a mixing board, which is uninteresting and sometimes tedious. We believe that moving one's hand and hearing the levels of applied musical effects mirror that movement will not only make for a very natural, emotive experience, but will also revolutionize the way that modern musical artists and performers utilize post-processing effects.

We intend to use motion-tracking technology to map the user's hand movements in real-time within a "cube of interactivity." We will then send the coordinates of these movements to a DSP module, wherein the values on any given axis will be assigned to an effect parameter. An example of this is a bandpass filter, in which case the three variable parameters might be center frequency, selectivity, and gain. The effects will be applied to the digitized audio input and then outputted as an analog signal, ready to be played through a speaker.

2. Introduction

Modern music creation and performance falls into two primary categories. Musicians can create and perform music using real instruments, either acoustic or amplified, or musicians can utilize electronic equipment, which may include a laptop running specialized software, called a digital audio workstation (DAW), as well as one or multiple Musical Instrument Digital Interface (MIDI) controllers. Traditional methods of music performance with real instruments usually requires bands with multiple people in order to play the variety of instruments involved in pieces of music. In the case of a one-man band performer, instrumentation is limited due to the impracticality of playing multiple instruments at once. The modern use of a laptop running a DAW remedies this problem, by allowing a single user to perform a piece of music with the aid of digital loops that temporarily record a section of audio and play it back repeatedly. MIDI controllers can simulate a plethora of real instruments, and they allow the user to tweak settings to find the sound they are looking for. The use of a DAW in conjunction with MIDI controllers allows for the performance of musical pieces with less hardware and with fewer personnel.

A typical band often includes a variety of instruments such as guitar, keyboards, drums, bass, and vocals. Therefore, a band with this type of instrumentation may require up to five members to perform. However, the same type of instrumentation can be performed by a single person through the use of a DAW and MIDI controllers. A MIDI controller can be used to simulate many instruments, such as pianos, synthesizers, electric keyboards, guitars, drums, horns, string sections, etc. This instrument simulation can be done either in the MIDI controller itself, or in the DAW. The use of electronic systems to simulate instruments means the user can make real-time audio adjustments, such as affecting the timbre of the instrument. This gives the user a lot of freedom in all elements of the performance.

In modern music performance, the use of DAWs and MIDI controllers is increasing rapidly. With the increasing popularity and emergence of electronic music, now anyone can purchase the necessary equipment to compose full arrangements and perform them in the comfort of their own homes, and without spending large amounts of money on instruments and hardware. Many prominent musicians exclusively use this type of setup, and large bands still frequently incorporate elements of the DAW and MIDI controller setup into their usual performance setup.

There are many popular software packages on the market that musicians utilize. One of the most popular DAW's worldwide is Ableton Live. Ableton is a DAW that allows for full scale audio recording, production, and processing. It is catered towards aiding in the creative process, and focuses on real-time manipulation of sounds and looped instruments to create an improvisational feel. The most recent version comes preloaded with 9 virtual instruments and 41 effects, all of which are fully customizable by the user. The main draw of this software is that it has a user interface that is very intuitive for real-time looping and manipulation, which is what makes it the top choice amongst electronic musicians and DJs all over the world.

For our project, we are trying to design a system that merges traditional music performance with modern digital technology. Our design will incorporate live performance of real instruments, and will allow the user to modulate the sound in real time using a hardware device without the necessity of a computer. The hardware that will be involved in modulating the sound will be a physical three-dimensional space which the user moves his or her hand through, with a motion tracking system recording the location of the user's hand in this space and providing an output corresponding to the coordinates of the location on each axis. These values will then control digital effects with parameters based on the received hand coordinates. This will preserve the traditional showmanship involved in playing real instruments, while providing the performer with an intuitive way to modulate the sound and tonal quality of their performance.

3. Problem Formulation

Our main design goal involves creating a three-dimensional space in which hand movements are captured by a motion tracking system, and using those coordinates to affect and interface with an audio input. This will allow hand gestures to control the tonal quality of the audio coming through the system. This project will consist of two main subsystems: motion tracking, and audio manipulation/DSP. For the motion tracking, a camera system will track the hand movements and gestures the user makes within the defined "cube of interactivity", and calculate normalized values based on the location and gestures of the hand.. These values will be used to modulate the sound of incoming audio signals. The audio signals will pass through an analog-to-digital converter (ADC), and then to a DSP module for digital audio effects processing. The data from the motion tracking system will provide commands to a microcontroller which will modify the parameters of the audio effects in the DSP module. The digitally processed audio will then pass through a digital-to-analog converter (DAC) and be output to several possible audio connectors.

An example of an audio effect that can be implemented is bandpass filtering, with the axes corresponding to center frequency, bandwidth, and gain. Another audio effect example is distortion, with axes corresponding to clipping threshold, distortion bandwidth, and gain. There are many audio effects that can be implemented using DSP, providing the user with a wide range of creative freedom over the sound that comes out, and we hope to eventually add the ability to customize what each axis is modulating and which effects are being utilized at a given time.

In order to carry out this design, the system will be split into two sections that will be developed at the same time. The first part is the motion tracking, and will involve learning how to use the hardware we select, programming it to be able to recognize hand movements and gestures within our cube of interactivity, and generating a usable output corresponding to those movements. The second part of the system will be the audio signal path. This consists of A/D conversion, DSP programming, and D/A conversion for the output. The DSP programming will open up a lot of possibilities for building our design, as there are many types of effects that can be implemented for audio processing. The final step will be to combine these two subsystems so that the motion tracking output interfaces with a microcontroller that can change the DSP effects in real time.

4. Research & Analysis

The following sections outline and investigate the different components of each subsystem. Through our research we have narrowed down our design which will be discussed in future sections of the proposal.

4.1 Motion Sensing

One of our goals for incorporating the motion controller is to place it as a stationary unit, which tracks the user's hand gestures. Therefore, our research on motion controllers does not include any type of 'wearable' device. We researched different virtual reality motion controllers (the Xbox Kinect and Leap Motion), an image sensing board (Pixy), and capacitive sensing.

4.1.1 Xbox Kinect

The Xbox Kinect sensor is commercially advertised as a convenient tool to control your Xbox and TV with video capabilities, voice recognition, and virtual reality gaming. Aside from purely entertainment purposes, Microsoft has an extensive array of developer toolkits to utilize the power of the sensor. The Kinect has the capability to track up to six people at a time while tracking up to 25 skeletal joints per person. It can also track the angle the person is leaning, where the person's face is looking, basic facial expressions, coordinate mapping of the camera, depth, and color space, all up to a distance of 3.5 meters. Development of the sensor is done in C++, C#, Visual Basic or any other .NET language. The Kinect sensor costs $99.99 and the adapter for development costs $49.99. Therefore, the total cost would be about $150. There is a large community of developers that use the Kinect, therefore there are plenty of tutorials, starter code, and overall support for this device.

Figure 1: Xbox Kinect.

4.1.2 Leap Motion Controller

The Leap Motion Controller is a much smaller USB device that scans an area of about eight cubic feet. The device itself is the center frame of reference and the motions are tracked directly above it. The main purpose of the device is to track both hands as well as each individual finger. It can distinguish between which user's hand is used, the position and velocity of the palm, the strength of clenching a fist as well as the strength of pinching two fingers together, and the direction of the movement of fingers. There are also a series of gestures that the Leap Motion recognizes such as a single finger tracing a circle, swiping, key tapping, and screen tapping. Coordinates are mapped in millimeters and the field of view is already designed as a rectilinear box, which ideally matches the design of our project. Similarly to the Kinect, the Leap Motion has a large development community and extensive documentation. However, there are software development kits for many more languages than the Kinect which include: C++, C#, Objective C, Unity, Java, Python, Javascript, and the Unreal Engine. The controller is physically less cumbersome, and its cost is $79.99, which is cheaper than the Xbox Kinect.

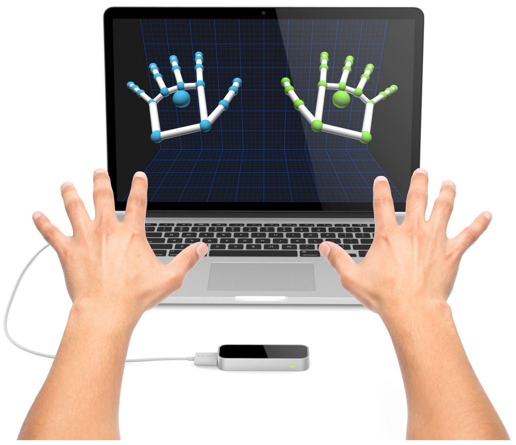

Figure 2: Leap Motion.

4.1.3 Pixy Camera

The Pixy is an image-sensing board equipped with a camera and processor that is designed to detect and track objects and easily interface with microcontrollers such as an Arduino or Raspberry Pi. The Pixy can identify different objects (mainly based on color) and with the aid of servo motors, the device can tilt, pan, and move around to keep the learned object in the center of the field of view. The Pixy sends updates of the tracked object's position and size in the field of view at a rate of 50 times per second.

Each object taught to the Pixy can be represented in code as a "Pixy object." This object has a print method along with the following 5 attributes: x and y coordinates, width and height of the object in pixels, and signature. The signature represents the memorized location in an order of objects from 1 to 7. With the coordinates, adjusting the parameters of the audio effects will be as easy as setting variables equal in the Arduino sketch if we employ DSP shields.

The Pixy camera's field of view is given to be 75 degrees horizontal and 47 degrees vertical. With some brief calculations we deducted that to mimic the eight square foot 3D space offered by the Leap, the Pixy would need to be located 2.3 feet away from the user. This calculation was determined without the panning and tilting functionality, which proves that we would most likely need this functionality in our final design. Ultimately, increasing the effective field of vision is an area we can explore in more detail during our proof of concept testing and design.

The Pixy costs $70 and if we want to increase the effective field of vision, the pan/tilt servo kit is about $40. Another interesting feature is that the Pixy is able to detect the point of a laser as an object. If we decide to use a laser system as a visual aid in one of our future add ons, the Pixy could interface with this.

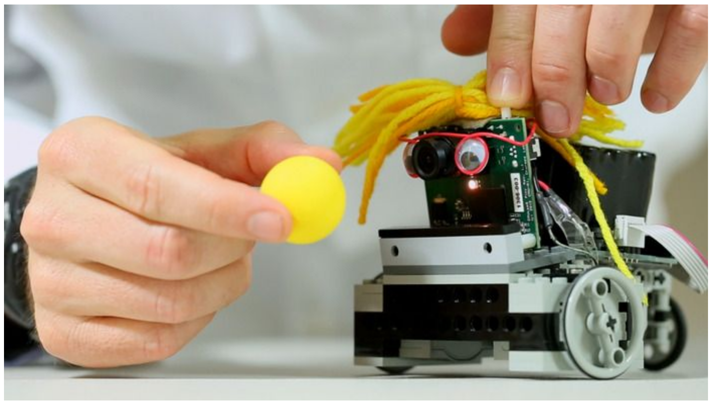
Figure 3: The Pixy.

Table 1: Comparison of System Requirements.

	Xbox Kinect	**LeapMotion**	**Pixy**
Processing Power	64-bit dual-core 3.10+ GHz processor	AMD Phenom II, Intel Core i3/i5/i7 processor	No external processor needed
Memory	4GB RAM	2GB RAM	No external memory needed
Power Consumption	12 Watts	2.5 Watts	0.7 Watts

After compiling the data in **Table 1**, we quickly realized that both the Xbox Kinect and the Leap require a lot of processing power, which means that we would not be able to solely run these devices with a microcontroller.

4.1.4 Capacitive Sensing

Capacitive Sensing is another method we could use for spatial tracking. It utilizes the difference in dielectric constant between air and other materials (in our case, the human body). When a conductive object approaches a capacitive surface such as a metal sheet, the electric field generated causes the capacitive element to deposit or accumulate charge. The distance between the two conductive elements determines the strength of the electric field. This in turn affects the rate of change of total charge within the capacitive element. In other words, the derivative of the charge on our capacitive element is proportional to its proximity to our hand. So, we can utilize several of these capacitive elements in orthogonal directions and measure their rates of charge/discharge to triangulate the position of our hand in 3D space. One of the benefits of this method is that it's extremely cheap and easy to set up, and involves very little software. Potential downsides of this method could include its spatial accuracy/sensitivity, and the difficulty we might have in determining the proportionality between rate of charge and the distance from the capacitive surface.

4.2 Effect Creation with DSP

The goal of our system is to be able to create a variety of musical effects on any audio input and be able to change the effect parameters by allowing the user to move their hand in three-dimensional space. To create these effects we decided that using digital signal processing techniques would provide more versatility than implementing specific effects in hardware, as well as allow for future updates and additions to the available effects. Our implementation of creating effects via digital signal processing is hardware-specific, so we will outline the different choices in hardware available and some tools that we can use to create our audio effects.

4.2.1 Digital Signal Processing Hardware

Digital signal processing is the act of measuring, filtering or manipulating an analog signal by numerically processing the signal in the digital domain. Therefore, many digital signal processing algorithms require heavy use of mathematical operations that usually need to be carried out quickly in order to correctly process a signal. Our system, and many other systems requiring real-time digital signal processing, must have a low enough latency for audio to be manipulated without noticeable lag or destruction of the input signal. Because of this issue, there are DSP microprocessors specifically created with architectures designed for the computation of DSP algorithms with higher efficiency, less power consumption and lower latency than general microprocessors. One other method of implementing digital signal processing algorithms is to use an FPGA to perform the DSP in hardware. However, due to our levels of experience with FPGAs and the amount of information available, we have decided not to do further research into creating our audio effects on an FPGA. After researching different digital signal processing chips and microcomputers we came up with three audio/DSP hardware systems that we could use to create our desired effects.

If we decide to use a specific digital signal processor chip, there is an open source digital signal processing board available called freeDSP. The board comes with audio inputs and outputs and is based around an ADAU1601 SigmaDSP audio processor from Analog Devices. This board contains all hardware needed for creating our audio effects and is easy to communicate with using an external microcontroller. Because of the specific digital signal processing hardware, using this board offers great real-time audio effect value with very low latency. The downside of using this system is that we would still require another computing system, like a microcontroller or microcomputer, to communicate effect parameters directly to the digital signal processor on the freeDSP board or would have to cycle through different effect programs stored in an EEPROM on board to communicate with the audio processor. This may cause lag between a user moving their hand inside the cube and the audio being affected.

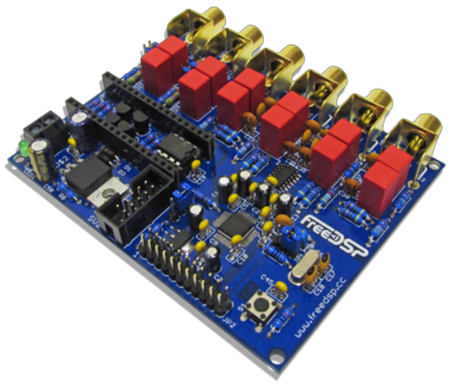

Figure 4: freeDSP board.

Another option is to use a general microprocessor to run our digital signal processing algorithms. There are a number of microcomputer and microcontroller boards available that allow us to interface with hardware. Because we have to sample audio, perform digital signal processing on the audio and then convert the audio back to an analog signal, we have chosen microcomputers with support for interfacing with audio. We researched multiple microcomputer options and came up with the following two systems: the Arduino with Open Music Labs' DSP shield, and the BeagleBone Black with the Bela cape.

The Arduino Uno is a board containing an ATmega328P microcontroller and all necessary peripherals. The Uno itself is not very powerful; however, there is an external DSP shield made by Open Music Labs that contains a Wolfson WM8731 Codec that acts like a digital signal processor and would be able to create different audio effects. This communicates with the Arduino over SPI and also contains an audio input and audio output. One downside to using this system is that there is no amplifier on board which would allow us to plug in directly to a speaker system. If we wanted to use speakers, we would have to add our own amplifier circuit. With a bit of hacking, this DSP shield could also be used with other microcontrollers and microcomputers with SPI communication capability. This allows us to use more powerful microcontroller boards if we find the Arduino Uno not powerful enough.

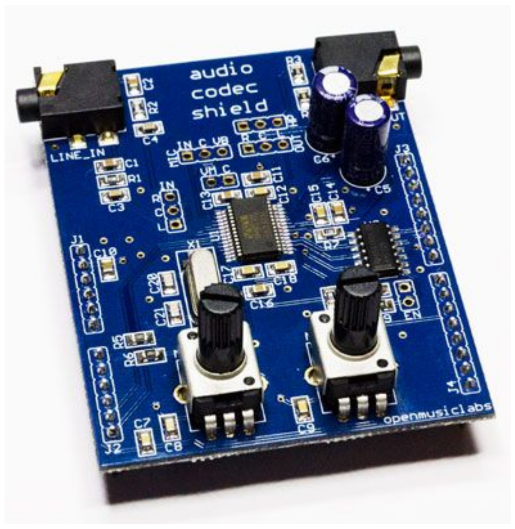

Figure 5: Audio Codec Shield.

The BeagleBone Black is a microcomputer powerful enough to perform digital signal processing algorithms with low latency. What is so special about the BeagleBone Black, however, is a cape (shield for beaglebone) that was designed out of the Augmented Instruments Laboratory, part of Queen Mary University of London, called Bela. Bela is an embedded platform for 'ultra-low' latency audio and sensor processing. The Bela has a powerful Linux computer embedded into it that can provide hard real-time performance with 1ms latency. It has its own dedicated hardware and software environment that give Bela task priority over the BeagleBone. Along with power, Bela also contains a number of audio inputs and outputs and includes an amplifier giving it the ability to connect directly to speakers. This system is designed specifically for applications like our project, however, the Bela is not currently on sale and this must be considered when choosing our components. Table 2 below shows a comparison of the systems mentioned above.

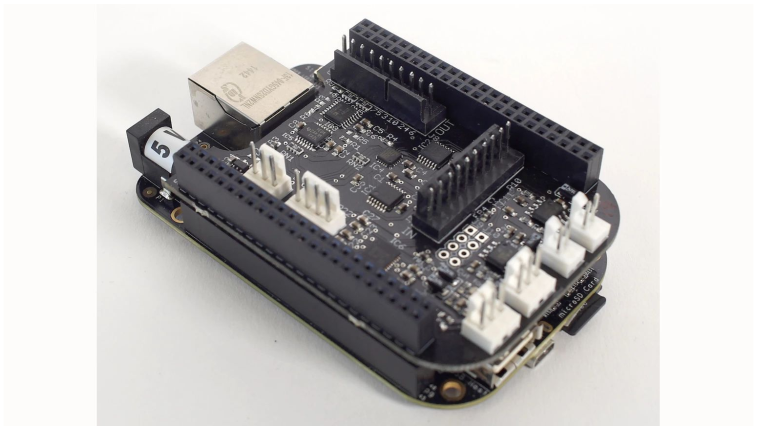

Figure 6: Beaglebone Black with Bela Cape.

Table 2: Comparison of DSP Tools.

	freeDSP Audio Board	**Arduino w/ Codec Shield**	**Beaglebone Black w/ Bela**
Pros	-DSP on board is designed to run signal processing algorithms quickly -Able to interface with any microcontroller/microcomputer -Graphical programming using SigmaStudio	-Libraries and examples already available for working with Codec shield and real-time audio -Can be used with a number of different microcontrollers	-Designed for extremely low latency with real-time Audio & DSP -Comes with browser-based oscilloscope -Libraries for a number of programming languages -Graphical programming using PureData
Cons	-Complicated documentation -Limited program space in on board EEPROM -Possible latency issues when communicating instructions from external microcontroller/computer	-No amplifier built in for direct connection to speakers -Computing power limited by microcontroller can cause high audio latency	-Limited availability

4.2.2 DSP Software/Tools

Depending on which embedded system we choose to implement our DSP algorithms, there are a number of different digital signal processing tools available. If we decide to use the freeDSP board, there is software available called SigmaStudio that would allow us to program the digital signal processor on board with a graphical user interface. This would make it easier to work with the already complex system and allow us to prototype effects more efficiently. If we choose to use the Arduino Uno and DSP shield system, there are a number of DSP libraries and effect examples available that we could use to create our own. Lastly, if choosing the Beaglebone Black and Bela, the Bela has a number of DSP libraries available in C++ and Python and also comes with software that can compile Pure Data programs into optimised C code. Pure Data is a powerful graphical programming language that makes working with audio and digital signal processing quicker and more intuitive. Being able to program our effects and digital signal processing algorithms with a graphical tool would allow us to prototype and work with complicated hardware faster than learning how to use separate libraries for different systems. Another feature is a browser-based oscilloscope that comes with the Bela and would allow us to see our effects on the input audio more clearly.

5. Design & Testing Strategies

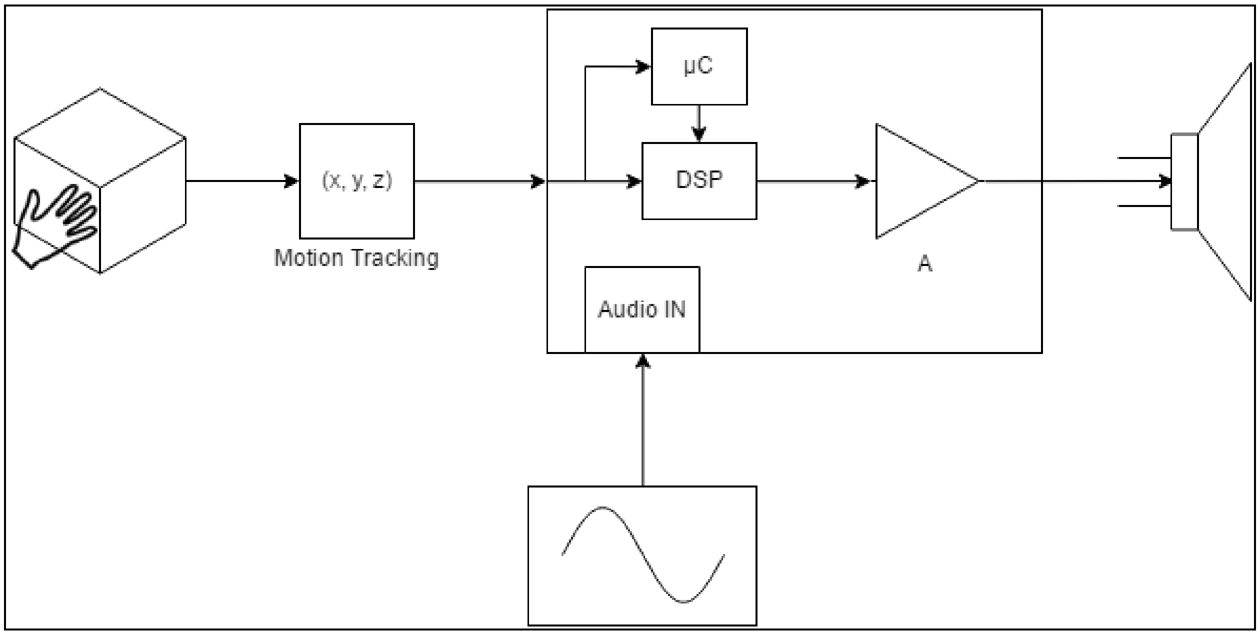

Figure 7. System Concept Design.

5.1 High Level Design

Our ideal system design will consist of the Pixy Camera with pan/tilt servo kit for motion tracking, and the Beaglebone Black with Bela cape for DSP. We chose the Pixy because of its ability to gather and transfer data quickly, updating fifty times a second, without requiring an external laptop. Bela with a Beaglebone Black was chosen because of the Bela cape's low audio latency with real-time digital signal processing ability and because of the many software tools available for Bela. A high level system design utilizing these components is as follows:

The system is designed so that a single user can provide an audio signal input as well as modulation parameters for the digital audio effect. There will be a variety of audio connectors, allowing the user to work with any type of audio inputs. For example the user could play an instrument such as a guitar, and use a ¼" cable to connect directly to the system, or use a microphone to pick up the sound and provide an electrical signal to the system via an XLR cable. The user could also play any line level instrument such as a synthesizer or a drum machine, or even get audio from a laptop or a phone. This audio will then be routed to the Bela for analog to digital conversion followed by digital signal processing.

The DSP unit will apply audio effects to the digital signal, and the parameters of these audio effects will be variable based upon the input of the user's hand location in 3D space. The cube of interactivity will be predetermined, and as the user moves his/her hand through the space, the Pixy Camera with pan/tilt servo kit will track the motion of the hand and follow it while calculating the x,y,z coordinates of the hand within the space. The Pixy will send these coordinates to the Beaglebone Black at a rate of 50 times per second, at which point an algorithm in the microcontroller will turn these coordinates into commands that will modulate predetermined parameters of the audio effect in the DSP unit of the Bela cape.

After the audio is digitized and modulated with the effect, the Bela will perform digital to analog conversion on the digital audio signal, and then amplify it to a power level that is suitable for driving external speakers connected by the user. At the output of the system a variety of audio connectors will be available, similarly to the input. There will be connectors for ¼", ⅛", XLR, etc. so that the user can pair this system with many speaker or headphone setups.

5.2 Proof of Concept Tasks

During the Summer II and Fall semesters, we are planning a proof of concept design and testing period. Our main goals during this period are to get familiar with the motion tracking and DSP systems we have researched, and testing the two systems separately to determine whether they will be able to work together in the full implementation of the project.

First, we will purchase a Pixy and the accompanying pan/tilt kit and test out its functionality by interfacing it with the Arduino Uno. One of our group members owns a spare Uno so that will allow us to immediately start testing, which is critical since the Bela might take time to acquire. Our first task will be to implement the sample Pixy code to read the coordinates of a tracked object. With this sample code, we will determine if the Pixy can detect a user's hand as an object. If not, we will experiment with more vividly colored objects, such as stickers, that we could attach to the hand that would not obstruct the user from playing their instrument.

After we determine a workable object to be tracked, we will dive into more of the specifics of obtaining the x,y,z coordinates of the object in code. The x and y axis coordinates should be straightforward to obtain, but z will require some mapping of the object's size. Thus, we will need to try out different mapping algorithms to see how accurately we can obtain the z coordinate, and also test how much the accuracy suffers if the user is changing the angle of their hand. If we experience difficulty using the Pixy to accomplish our design goals, we will introduce a subtask to start investigating the possibility of using capacitive or ultrasonic sensing for our motion tracking.

Once we have the Pixy performance and capabilities scoped out, we will move on to testing out our DSP options. If by this time we can obtain a Bela, we will purchase one alongside a BeagleBone Black. If the Bela is not available to us, we will go forward with the Arduino and purchase the Open Music Labs DSP shield.

With either device, the Arduino or the BeagleBone Black, our next task will be to experiment with its workflow and capabilities by trying to implement an effect using the included DSP libraries. If we are using BeagleBone Black we may experiment with using PureData during this first task. During this task we want to isolate the DSP system from all other interfacing variables that will be present in the full implementation. So the system will most likely consist of some audio signal input to the microcontroller via ⅛" port, modulated from our effect created in software and output to a pair of speakers.

After we verify that we can create an effect and apply it to an audio signal, our next task will be to test out a system where we can change the parameters of the DSP effect in real time. One simple way of testing this out initially could be hooking up potentiometers to analog inputs of the microcontroller and routing these analog inputs to parameters of our effect.

At this point, we would have the proof that each system, motion tracking and DSP, can work separately and produce the inputs/outputs necessary to work with each other. If the steps needed to connect the two systems are rather simple, we could connect them at this point, and if not we could make a plan to connect them for the Spring semester. Also, after experimenting with each system, we would have a decent idea about the quality of performance and the main areas we would need to focus on during the design and execution of the full implementation.

5.3 Full Implementation Tasks

Once we have separately determined that our motion tracking and DSP subsystems are working as expected, we will begin merging the two and testing the performance of our system.

First, we expect that we will need to do some sort of spatial hand motion calibration testing so that we can set our desired “cube of interactivity” and ensure that we are able to create a functioning program/script to normalize our hand coordinates to a custom scale for implementation of our DSP effects.

We will then focus our efforts upon optimizing our DSP code so that we experience as low latency as possible in the application of our effects. In the event that there is noticeable audio latency, we will test the robustness of our DSP scripts, chip, and if necessary use a dynamic spectrum analyzer to test the propagation of our signal throughout the system to determine where the bottleneck occurs.

If we are experiencing sluggish responsiveness in our audio effects at the output, then we will attempt to optimize our datastream between the Pixy and our DSP system, and time our scripts to see if a particular calculation is causing the lag.

Finally, we will build the physical chassis to hold our project, and outfit it with the necessary I/O ports of ⅛ in, ¼ in, and XLR (microphone) so that our system can be used in a variety of situations and with a variety of inputs.

6. Additional Features

Following the full design implementation, if time permits, the following additional features may be developed which would enhance the user experience and be more visually stimulating. In no specific order, these features include:

- Creating a 4th dimension with the user pinching their fingers and/or clenching their fist
- Looping the hand gestures to enhance creativity
- Adding recording capabilities and creating looping for the recorded tracks to allow the user to put down their instrument and spend more time on sound modulation
- Creating laser tracking of hand location and coordinate confirmation
- Adding the ability to customize DSP axis parameters
- Incorporating a visual aid of the recorded spectrum

7. Cost Analysis

Product	**Quantity**	**Cost**
Pixy & Pan/Tilt Kit	1	$115.00
Bela	1	$70.00
Beaglebone Black	1	$55.00
Arduino	1	$30.00
Codec Shield	1	$30.00
Raspberry Pi 3	1	$40.00
Misc. Parts	1	$300.00
Speakers	1	$100.00
	Total Cost	$740.00

The main purpose of this cost analysis is that it includes the overall cost for multiple designs as discussed earlier. We generously allotted an amount for miscellaneous parts, which include the casing for our final product as well as materials for our future add-ons. Ultimately, each of our materials are fairly inexpensive and we are confident that our final design will be under the maximum allotted budget.

8. Timeline

Project Brainstorming	■										
Research	■										
Proposal	■										
Final Report										■	■
Final Presentation											■
Purchasing Materials		■									
Proof of Concept Design			■	■	■	■					
POC Testing			■	■	■	■					
Final Design							■				
Motion Control Design								■	■		
Motion Control Testing									■	■	
DSP Design								■	■		
DSP Testing									■	■	
System Integration										■	
Mechanical									■	■	
Add-ons (if time permits)								■	■	■	
	Jun 16	Jul 16	Aug 16	Sep 16	Oct 16	Nov 16	Dec 16	Jan 17	Feb 17	Mar 17	Apr 17

9. Conclusion

We are proposing to design a device that allows a music performer to affect the tonality of audio with no more than a hand gesture. Our system aims to antiquate the use of knobs and sliders for applying effects, instead utilizing a much more natural mechanism: three-dimensional space. We will use motion tracking technology to collect x-y-z coordinate data, which we will then send to the DSP system. The DSP component will map those values to effect parameters before applying those effects to an audio input. The final product will be an easy-to-use and intuitive way for users to change the way played music will sound. The timeline and planning detailed in this proposal ensures that our project will be completed on time, and should we have extra time, we will expand upon our design with a number of additional features.

10. Bibliography

"Analog Devices : Analog Dialogue : Capacitive Sensors." *Analog Devices : Analog Dialogue : Capacitive Sensors*, http://www.analog.com/library/analogdialogue/archives/40-10/cap_sensors.html.

"Arduino Playground - CapacitiveSensor." *Arduino Playground - CapacitiveSensor*, http://playground.arduino.cc/main/capacitivesensor?from=main.capsense.

"Audio Codec Shield." *Open Music Labs*, http://www.openmusiclabs.com/projects/audio-codec-shield/arduino-audio-codec-shield/index.html.

"Avid." *ProTools*, http://www.avid.com/pro-tools.

"Bela.io." *Bela.io*, http://bela.io/.

"CMUcam5 Pixy." *Overview*, http://www.cmucam.org/projects/cmucam5.

"CMUcam5 Pixy." *Will Pixy tracksense laser light*, http://cmucam.org/projects/cmucam5/wiki/will_pixy_tracksense_laser_light.

"Coordinate Systems." *Coordinate Systems — Leap Motion C# SDK v3.1 documentation*, https://developer.leapmotion.com/documentation/csharp/devguide/leap_coordinate_mapping.html.

"A DIY Audio Dsp Project." *freeDSP*, http://www.freedsp.cc/.

"Developing with Kinect for Windows." *Developing with Kinect*, https://developer.microsoft.com/en-us/windows/kinect/develop.

"Gestures." *Gestures — Leap Motion C# SDK v2.3 documentation*, https://developer.leapmotion.com/documentation/csharp/devguide/leap_gestures.html.

"Hands." *Hands — Leap Motion C# SDK v3.1 documentation*,

https://developer.leapmotion.com/documentation/csharp/devguide/leap_hand.html.

"Kinect for Xbox One." *Xbox.com*,

http://www.xbox.com/en-us/xbox-one/accessories/kinect-for-xbox-one.

"Kinect Hardware Requirements and Sensor Setup." *Kinect hardware setup*,

https://developer.microsoft.com/en-us/windows/kinect/hardware-setup.

"LiveCreate, Finish, Perform." *Learn more about our music making software Live*,

https://www.ableton.com/en/live/.

"Pixy Camera: Detect the Colour of the Objects and Track Their Position." *Open Electronics*,

http://www.open-electronics.org/pixy-camera-detect-the-colour-of-the-objects-and-track-the
ir-position/.

"What Are the System Requirements?" *Leap Motion Support*,

https://support.leapmotion.com/entries/39315178-what-are-the-system-requirements-.

"What Is the Leap Motion Controller?" *Leap Motion Support*,

https://support.leapmotion.com/entries/91355303-what-is-the-leap-motion-controller-.

http://www.youtube.com/channel/UCZWJf3quTMb-lxg2rn-g6DA. "Pixy Kickstarter Video."

YouTube, YouTube, Feb. 2014, https://www.youtube.com/watch?v=j8sl3nmlyxm.