Digital Theremin Synthesizer and Visualizer
6.111 Project Proposal

Alexander Spicer, Daniel Rodgers, and Jeffrey Chang

November 1, 2007

1 Overview

The end product that our team will design is intended to be a musical instrument played in a manner similar to the Theremin, an audio effects synthesizer, and a real-time music visualizer, all in one. Users will be able to move their hands in front of their body to create musical sounds, apply various filters and audio effects using a computer keyboard, combine the result with an audio stream of their choice, hear the music being created in real-time, and watch a colorful, moving visualization of their creation be continually generated on a computer monitor in front of them.

Our final project will essentially have three main input mechanisms and two outputs. Audio can be inputted by plugging in an mp3 player into the labkit. Music can also be generated by moving ones hands in front of a video camera connected to the system. The music will be combined with audio effects that the user chooses (which are specified via a computer keyboard, the third "input") and the end result will be played, in real-time, via speakers. The audio streams being supplied and generated will also be visualized and displayed, in real-time, on a monitor display.

We believe the following organization and modularization of our project will allow us to set up a simple system fairly efficiently, while providing us flexibility and plenty of room for sophistication and added complexity.

- Video processing to generate frequencies, and reading of the external analog audio input, will be handled by the Video Processing module, which Alexander Spicer will be in charge of.

- Audio effects and filters (e.g. echo, vibrato, sampling, equalizing) of the two incoming audio streams will be handled by the Audio Synthesis and Processing module, which Daniel Rodgers will be in charge of.

- Finally, the visualization of both audio streams on the external monitor will be handled by the Visualizer module, which Jeffrey Chang will be in charge of.

We will now describe each of these modules in turn from a technical perspective. We will also elaborate on the potential features and functionality of each module.

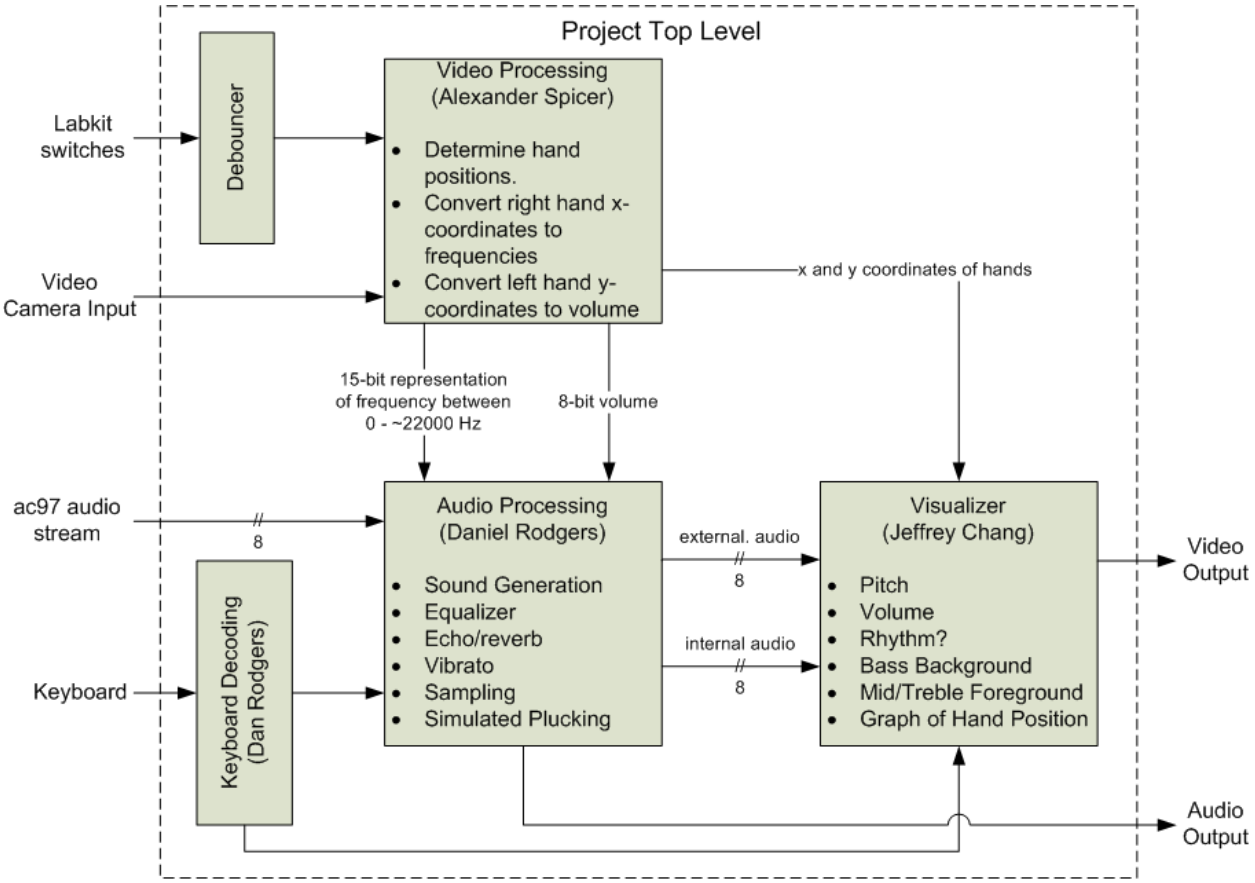

Figure 1: Top level block diagram of sub-components

2 Video Processing

2.1 Inputs and Outputs

The onboard ADV7185 decoder chip will provide a digital steam of video data from an external camcorder to the Video Processing module. The system will parse the incoming video data to determine the locations of the two hands on the screen. The top half of the screen will be dedicated to capturing the right hand's horizontal movement, to replicate the distance to the Theremin's pitch antenna, while the bottom half of the screen will be used to determine the left hand's vertical position, to simulate the Theremin's amplitude antenna.

The Video Processing module will also take as input a number of debounced onboard labkit switches. These will be used to determine what note the current on screen display is centered around and what range of frequencies the user can play at any given time. This feature will allow a large range of playable notes, with increasing accuracy as the screen is centered around specific notes.

The Video Processing module will output a 15-bit integer, which will correspond to the frequency of the note that is currently being played, along with an 8-bit volume integer. These will be processed by the Audio module to generate sound. Also, to aid in the visualization, the Video Module will also produce two integers, which correspond to the X coordinate of the right hand and the Y coordinate of the left hand, so that the Visualization module can output these as feedback on the screen.

2.2 Memories, Operations, and Throughput

The Video Processing will require some memory to store at least a portion of the incoming video steam. The goal isn't to process the entire video, but only the left most position of the right hand, and the bottom most portion of the left hand (in order to simulate the hand approaching the Theremin antennae).

The process will involve determining the location of the hands based on clusters of skintones and refreshing the X and Y location with every frame. Mathematical scaling will be performed to convert linear increases in hand position into logarithmic increases in frequency.

2.3 Plans for Testing

The sub-component can be testing independantly of the other modules. The video processing only produces quantized integer values for frequency, volume, and x/y coordinates, so testing can be performed easily by wiring the integer values up to displays and viewing the output as a hand moves.

2.4 Desired Functionality

The goal of the Video Processing module is to determine, in as real-time as possible, the x and y positions of the hands on the screen and to convert these integers into a corresponding pitch and volume based on user settings. The switches will allow the user to recenter the scales to produce as narrow or broad of a frequency range as is needed.

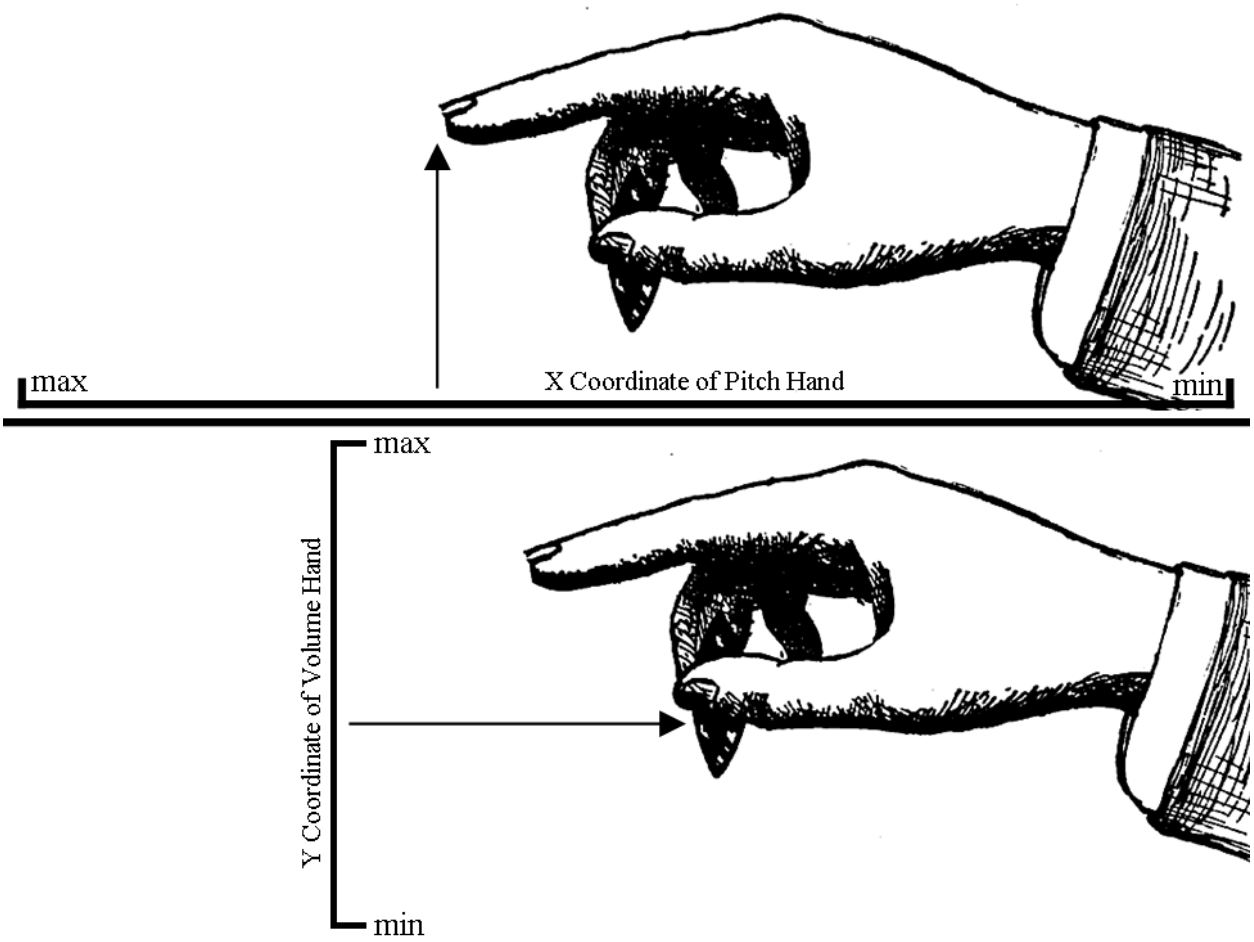

Figure 2: Example Image Capture

3 Audio Synthesis and Processing

3.1 Inputs and Outputs

The Audio Processing module will have three inputs - an 8-bit wide data stream directly from the AC97 controller built into the labkit, a 15-bit integer from the Video Processing module which represents the frequency of the note currently being played by the user, and the keyboard input from the user.

The outputs of the module are an 8-bit audio stream which is sent back to the AC97 to be played over the external speaker and represents the combination of the user generated sound and the external audio after they have been processed. The module also outputs the user generated audio and the external audio as two separate 8-bit audio steams for use by the Visualizer module.

3.2 Memories, Operations, and Throughput

The module uses a Fast Fourier Transform to manipulate the frequency components of the external audio signal. The ZBT RAM is used by this module, to allow for the possibility of sampling audio input for later playback.

3.3 Plans for Testing

Testing can be done independent of the other modules in the system. The user generated input can be simulated easily by using the labkit's switches to generate different input frequencies. The effects to be added to external audio can be tested and listened to with no assistance from the input module or the visualize module, allowing for simple stand-alone development and testing of this part of the system.

3.4 Desired Functionality

The audio control module is responsible for applying the desired audio effects and filters to the external audio input, as well generating the audio output of the system, which is a combination of the modified external audio and the modified user generated audio. Which audio effects to apply to the incoming audio streams are determined by the users input to the keyboard. The module also outputs the modified external audio and modified user generated audio directly to the visualization module. The effects available include echo/reverb, vibrato, and an equalizer. The module also allows for external audio to be saved and mapped to a keyboard key for later playback.

The module is also responsible for manipulating and fleshing out the frequency generated by the image processing unit. This is done by turning that frequency into the 8-bit signed audio stream expected by the AC97. Other effects, such as simulated strumming, can also be selected. The simulated strumming works by only changing the audio stream when the frequency of the input changes by an acceptable margin. If the frequency of the input does not change, this is akin to no new notes being played, so the amplitude of the audio wave dies out gradually, as it would if a string had been plucked.

4 Visualization

4.1 Inputs and Outputs

The Visualization module will have two inputs - an 8-bit-wide data stream (ext_audio_mod[7:0]) which consists of the external audio input (mp3 player) after the audio effects and filters have been applied, and another 8-bit-wide data stream (user_audio_mod[7:0]) consisting of the user-generated audio (Theremin-style) after the audio effects have been applied. Both will be signals sampled at a 44.1 kHz rate.

The Visualization module will use internal logic, plus most likely some stored sprites, to generate a dynamic video feed which will be linked to a VGA monitor. This visualization will update itself at a rate of 60 Hz. Furthermore, in the left and right-hand vertical margins of the screen, we plan to display some sort of visual indicator of the users current hand positions. This will allow the user to have some feedback on the positions of their hands as they are generating the music.

4.2 Memories, Operations, and Throughput

For any sprites that are used in the visualization, we will make use of the labkit's 4MB ZBT memory, or if needed, the 16MB Flash ROM. The logic and arithmetic operations used to calculate the desired visual output will need to be performed quickly enough so that the visualization does not lag behind the audio output by any noticeable amount. For frequency domain based visualizations, we may need to utilize an FFT, which is going to be included in the Audio Processing module.

4.3 Plans for Testing

Testing any visualization code on a given external audio input from an mp3 player should be fairly straightforward to set up. As for the user-generated signal, we can manually generate a series of frequencies and apply filters accordingly to have a simulated input to work with. The output should be hooked up to the VGA monitor from the start so we can test out various visualization logic and effects.

4.4 Desired Functionality

Besides the visual hand-position feedback we wish to display on the sides of the screen, the visualization itself will hopefully have a clear correlation with the audio content itself, yet be dynamic enough to be continually interesting and engaging. We hope to be able to parse the incoming audio stream into its core components to analyze changes in pitch, volume, and rhythm. Changes in these variables will correspond to different visual effects, such as color, shape, thickness, or movement. There are many possibilities for the types of visualizations and rules that we could follow. For example, we think it will be interesting if the external audio input (mp3 player) corresponded with the background of the visual display, while visualization obtained from the user-generated audio would be displayed in the foreground. To the user, this would reinforce the notion of "playing on top of" the existing music. Other ideas for the visualization characteristics include having bass frequencies dictate the appearance of the display across the entire screen, while treble frequencies would specifically affect the center of the screen and would change more rapidly. Incorporating time delays or fading effects might result in interesting visualizations too. Finally, since there are a plethora of visualization possibilities, we plan to include the ability to manually select (either via the labkit hardware or the computer keyboard) amongst a number of different visualization schemes. That way, the user can select one that is best, given their particular mood, or the genre of music being played.