Virtual Piano

Proposal By:
Lisa Liu
Sheldon Trotman

November 5, 2013

I. Abstract:

Who says you need a piano or keyboard to play piano? For our final project, we plan to play and display music based on hand gestures that imitate piano playing. The project can be divided into three parts: video capture and processing, sound processing, and display.

Gesture detection will be one challenge in our project. We will use frame-by-frame comparison to determine if there is a significant enough difference in finger position. Individual fingers or sets of fingers will be tagged with colors to aid tracking and detection. This detection can be done in real-time to imitate the instantaneous reaction of a key press on a piano. Due to time constraints, it may be difficult to track 10 individual fingers, so primarily we aim to track 2 (perhaps to play Chopsticks), but ideally we would like to track all fingers.

At the most basic level, the Virtual Piano will play a pure tone based on what notes it sees. However, in real life, instruments do not sound like pure tones. Piano notes will be stored in memory and can be recalled and played when needed. In response to information about the speed of the hand, the volume of the note will increase or decrease. If time allows, we may adjust the attack and decay of the note as well.

Lastly, we would like to display the last few notes played on a staff on the computer screen. The note name should also accompany the notes. Given enough time, we can also display the length of each note.

II. System Block Diagram:

The Virtual Piano will need to detect gestures and output sounds and pictures based on those gestures. Figure 1 shows the basic subsystems we would like to design: Visual Gesture Detection, Display Output, and Note Processor

Within the Visual Gesture Detection Module, we will detect a gesture, analyze a frame, store it in metadata storage, and compare it to the next frame in order to determine the change in position of a finger. A spatial analyzer will determine if a change in position represents a pressed key.

Once the Visual Gesture Detection Module determines that a key is pressed, the Note Processor will send information to the Audio Output Module (to output a sound) and the Display Output Module (display the notes).

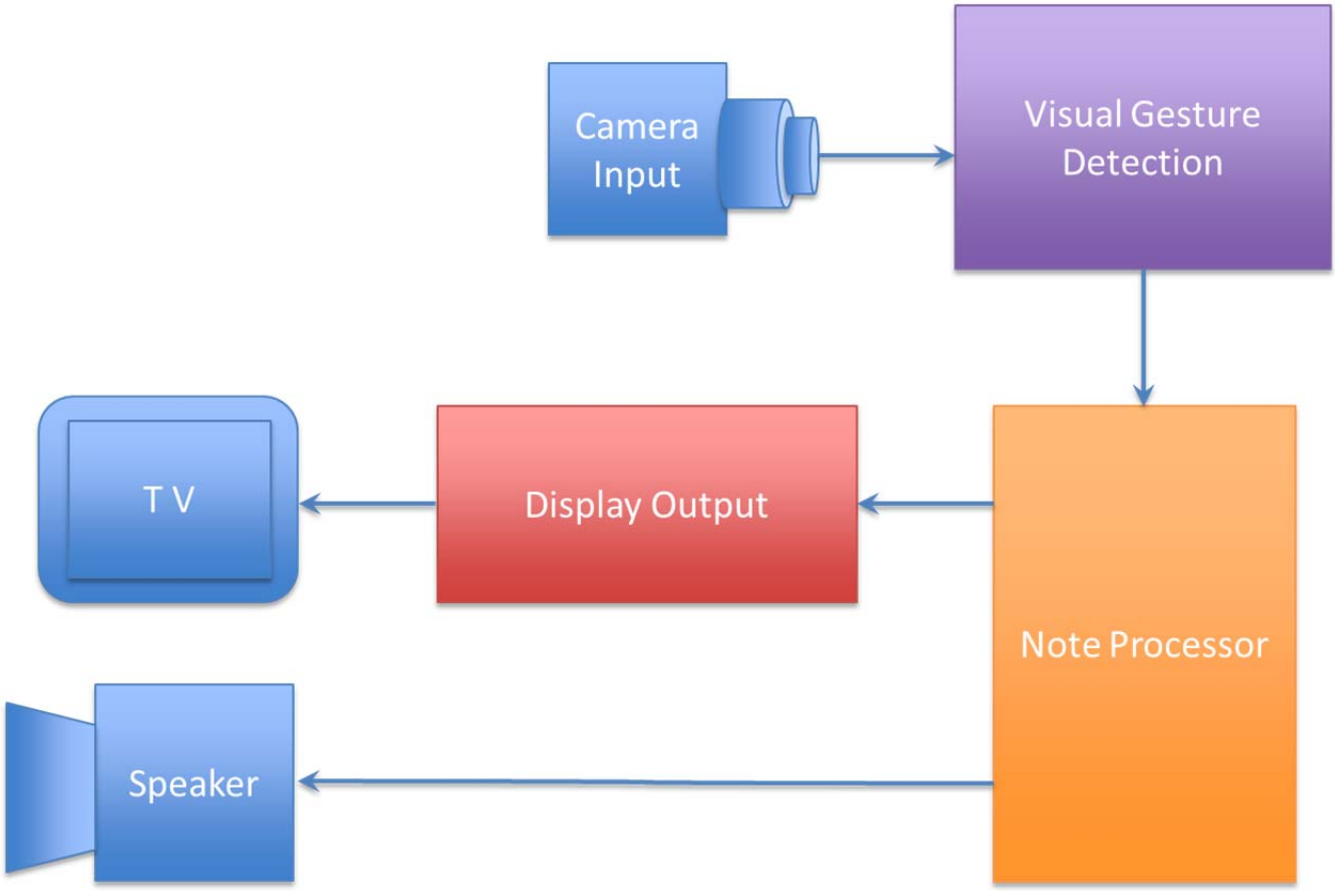

Figure 1. **General Block Diagram:** Graphical explanation of project design. We take in a camera input and after visual and logic processing we output display and audio data.

III. Sub-Modules:

We break each of the three major modules, gesture detection, note processing, and output display, into smaller modules, and go over the functionality of each.

The Visual Detection module allows us to determine whether a note has been played. It also tells us the attack of the note, or how hard or quickly the note was hit. Figure 2 shows the flow of information in the Visual Gesture Detection module.

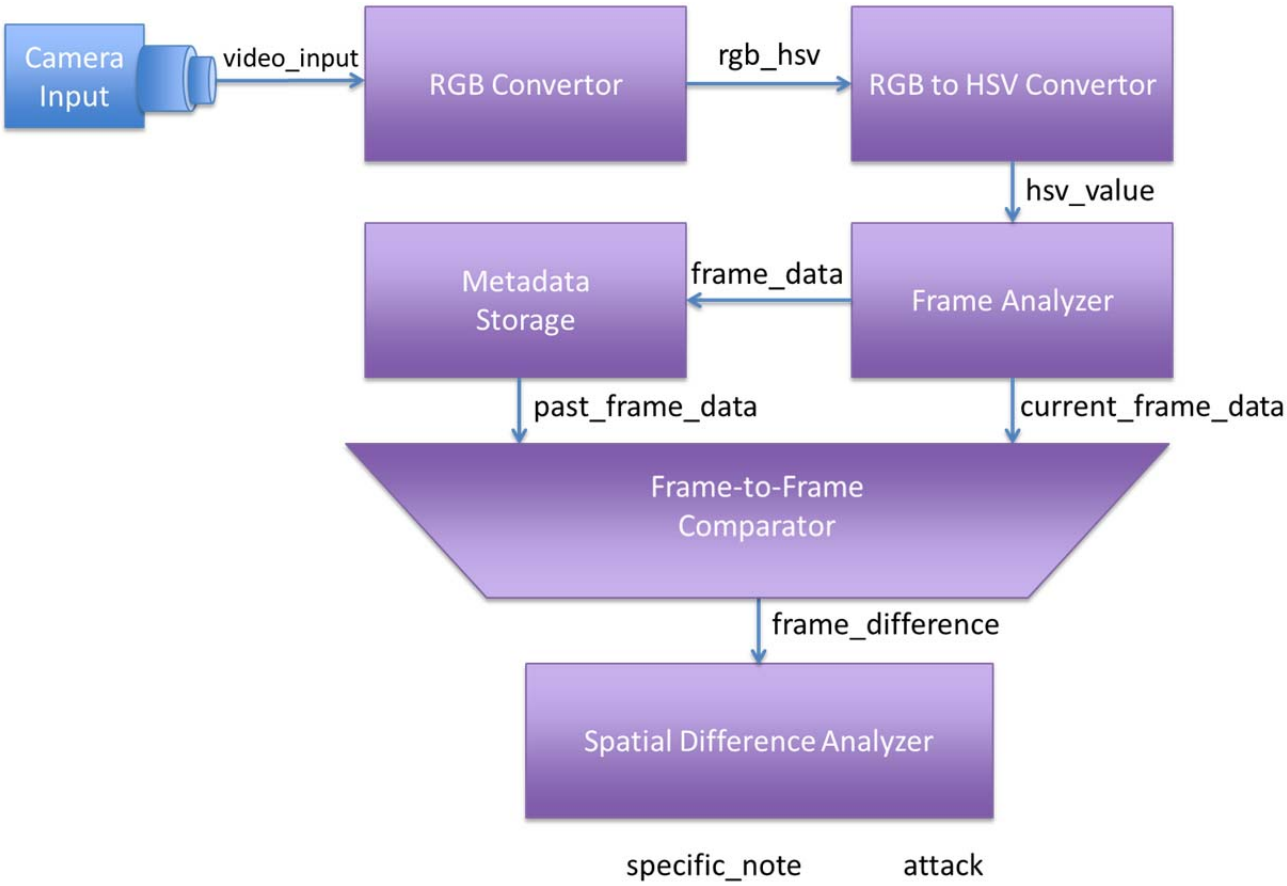

Figure 2. **Gesture Detection**: The RGB and HSV converter takes the camera input and changes it into a form that is easier to analyze. We compare frames with past frame stored in memory in order to determine the note played and the attack on the note.

Camera Input:

Raw data received from user interaction in the frame of the camera. This data is sent to the frame analyzer.

Frame Analyzer:

Here, detection of the specific colors used for the fingertips is done and passed on to be stored, as well as compared to previous data. The analyzer is one of the most complex portions of this project and will take the majority of the runtime.

Metadata Storage:

The data extracted from past frames are kept for future comparison. When a new frame arrives it is stored and the previous data is outputted to the Comparator.

Frame-to-Frame Comparator:

This module receives current data and previous data and does a direct comparison of finger placement difference from frame to frame. This is used in the next section information is used in the spatial difference analyzer. In future applications, velocity will come into play and will alter the sound.

Spatial Difference Analyzer:

Once the comparison of relative difference of the fingers is made, an overall spatial difference is necessary to see if the person has struck a key. The pressing of the key is handed off to the Note Processing module.

Once the Visual Gesture Detection module has determined a key and the attack it was hit with, the Note Processor must determine how the note should sound. Figure 3 shows how the modules interface with the Gesture Detection module and labkit inputs.

Metronome:

The metronome module plays a ticking sound at regular intervals so that the user can time his piano playing. A ticking sound is stored on ROM or flash memory and sent to an audio controller every time an enable signal is high

Metronome Timer:

This module determines how often a tick is played. The speed, which varies from 1 Hz to 2 Hz, is set by the switches. The metronome will send an enable signal four times per metronome tick, in order to have enough granularity to measure eighth notes correctly.

Note Counter:

Determines whether the note played is an eighth, quarter, half, or whole note. It receives a periodic tick from the metronome timer and the current note being played form the Visual Gesture Detection module and counts how long it lasts.

Sound Transformer:

Based on the detected note and attack of that note, it will call the correct note from memory and transform the sound be adjusting the attack and volume of the note. Over the length of the note, it will also allow the note to decay.

Audio Controller:

The audio controller combines sound from metronome and piano notes as needed.

Compact Flash Card:

Stores raw piano notes and outputs the sound stored at address.

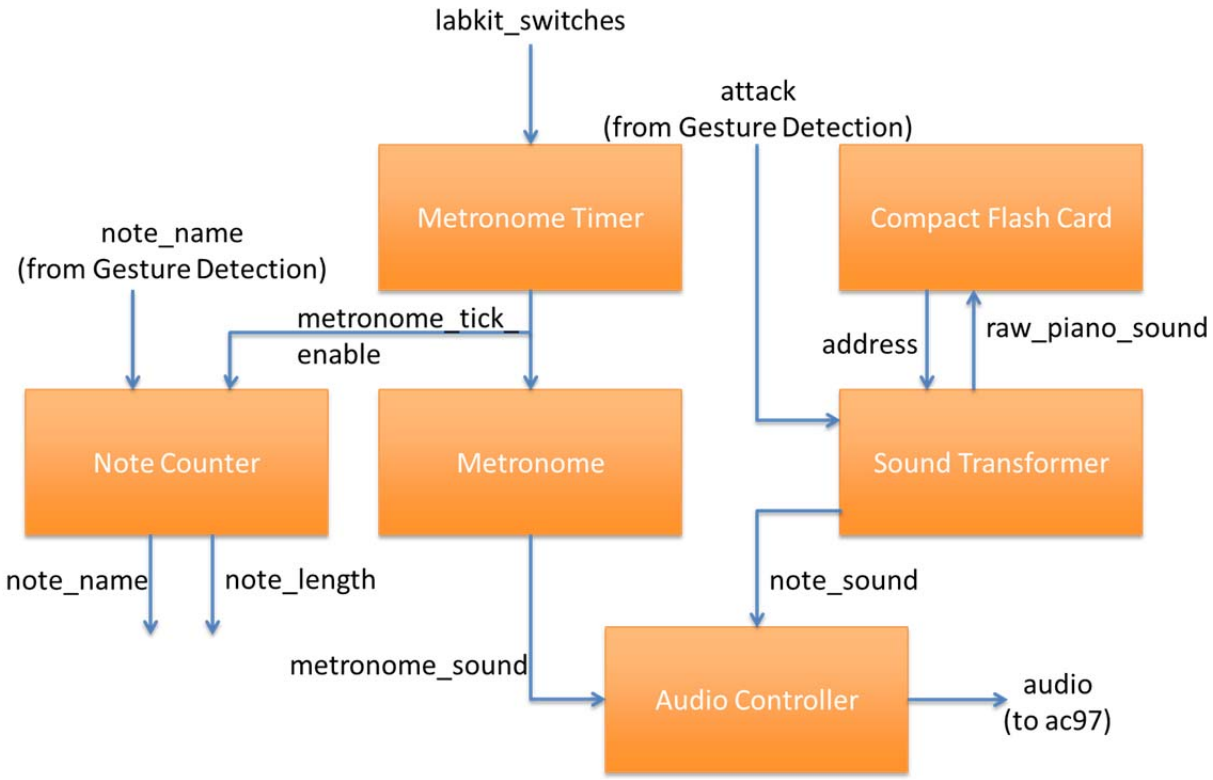

Figure 3. **Note Processing**: The note processor serves two main functions: duration measurement and sound transformation. The left side of the figure deals with duration measurement, and the right side deals with sound transformation.

The Note Processor model sends sound directly to the speaker for output. However, the note name and note length need to be processed by the Display Output module before being displayed on the screen.

Figure 4 shows the last few steps needed before the notes detected by the Visual Gesture Detection module make it to the screen.

Sprite library:

Stores sprites for eighth, quarter, half, whole notes, and keys. When a certain note is called for, it will output the sprite data.

Music logic:

Keeps track of the last 8 notes played, regardless of note length. If time allows, it may observe when two eighth notes are adjacent and replace the two eighth note sprites with a barred eighth note sprites.

Display:

In the final step before VHA output, we combine all display elements for each frame.

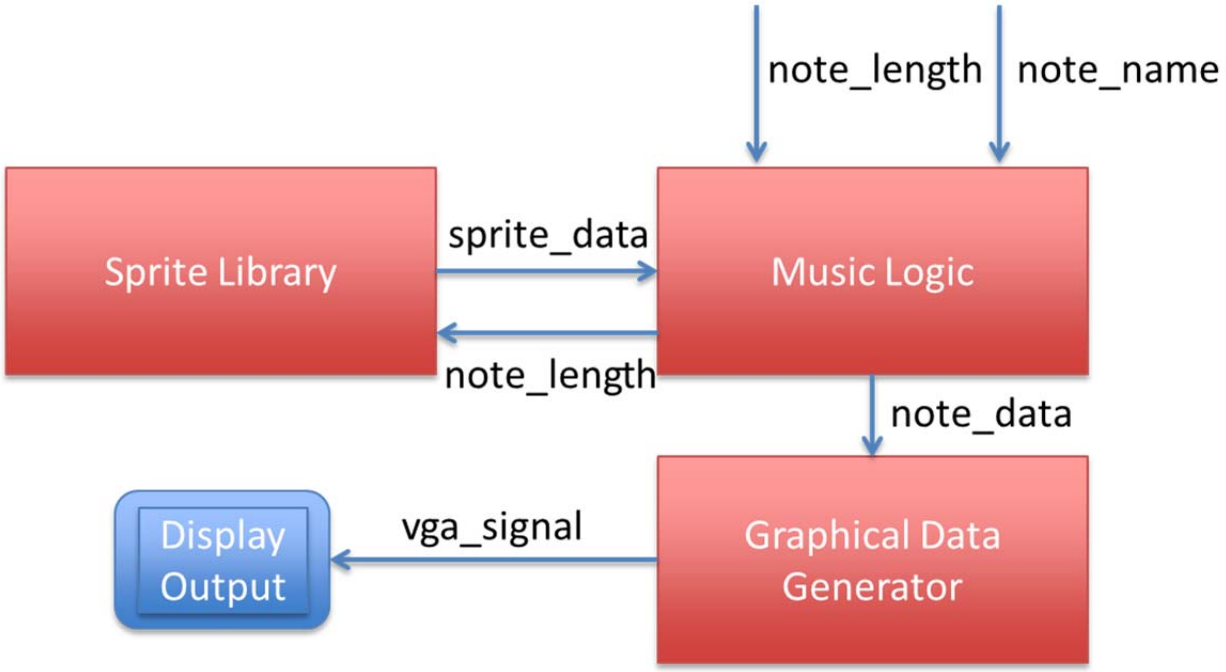

Figure 4. **Output Display**: From the Note Processor module, the Output Display module receives the note name and note length. It calls the sprite information and determines how to display the sprite on the bar staff.

IV. Project Plan

Week of 11/4

- Block diagram conference
- Start implement gesture detection
- Start figuring out how to store and recall audio

Week of 11/11

- Project design presentation
- Test and debug gesture detection. Be able to detect 2 fingers, hopefully more.
- Test and debug audio output. Minimum, be able to play unmodified tones.
- Start display module.
- Project checklist meeting

Week of 11/18

- Debugging of display module.
- Start implementing note-processing module.

Week of 11/25

- Thanksgiving
- Debug anything needed in note processing, display, gesture detection, or audio.

- Try to implement any stretch goals

Week of 12/2
- Debug and test whole design
- Prepare for project presentation

V. External Components
- Gloves w/ brightly colored fingertips
- NTSC Camera
- 6.111 Labkit

VI. Division of Labor
This project will be split as evenly as possible among the two of us. A specific breakdown of obligations is as follows:

Lisa:
- Graphical Data Generator
- Display Output
- Sound Transformer
- Note Timer
- Flash Memory
- Audio Output

Sheldon:
- Camera Input
- Frame Analyzer
- Metadata Storage
- Frame-to-Frame Comparator
- Spatial Difference Analyzer