MIDI Projection Visualization

Hong Wing PANG
COMP4971C - Independent Studies Project
Department of Computer Science and Engineering, HKUST

Supervised by Prof. D. Rossiter

Fall Semester
2017

Abstract

This project focus on creating a visualization application written in Processing, which takes in MIDI signals from the controllers of a DJ set, turns them into certain visual patterns and is shown to the audience via a projector. A DJ can, through this application, be able to mix audio and produce visual effects simultaneously during a live performance. The project aims to produce visualizations that dynamically resembles the audio that is being played, while at the same time is aesthetically pleasing to watch. This report covers the devices and software used in visual generation, the ideas and actual implementation of the visual application, as well as a demonstration of the result of this project via a video.

Contents

1. Overview

Visual performances are a popular addition to DJ performances, found in concerts and clubs. Traditionally, the DJ is responsible for mixing the audio together from existing songs, music works, or small clips of music; then certain visual effects are manipulated and put together in synchronization with the music. The latter job is separately handled by a VJ (video jockey).

This project aims to combine the two into one, by designing a visual application that would react to MIDI signals produced from a DJ set. The visuals are produced in real-time using Processing 3, a programming language-IDE combination based on Java that are primarily designed for artists to create visual works. The visualizing application should be able to react to incoming signals, provide an appropriate visualization, which is projected onto a surface. In other words, the DJ is capable of mixing music and producing visual effects at the same time, of which both are controlled by the same set of controls.

Ideally speaking, the visualization tool should be able to match the sounds that are being played. This project explores one such visualization scheme as well as the related calculations required to produce it.

2. Methodology

2.1 Structure

The whole setup includes two computers: the DJ-side computer and the visual-side computer. The visual-side computer is responsible for running the main visualization application and projecting it towards a screen.

The connections between the two computers and various devices / software packages are shown in Figure 1.

2.2 Workflow

The workflow of the project starts from the music input, which consists of multiple MIDI devices incoming from the DJ set. These signals are then transmitted to the visual-side computer via an Ethernet connection.

The visualization program actively intercepts such signals and generates the visualization graphics accordingly. In this project, the concept of "waves" are explored such that there are 7 separate waves drawn in a 3D wireframe style. These waves are mapped to 7 set of controls on the DJ side, such that the waves's motion (height, color etc.) can be controlled independently. Since the DJ uses the same set of controls for his own music and audio mixing, the goal is to achieve a mapping between the "waves" shown and the audio played. To further add to the visual effect, we further experimented on changing the tone and tempo of the mixed music by reflecting them in the background objects of the "waves", as well as changes in camera angles.

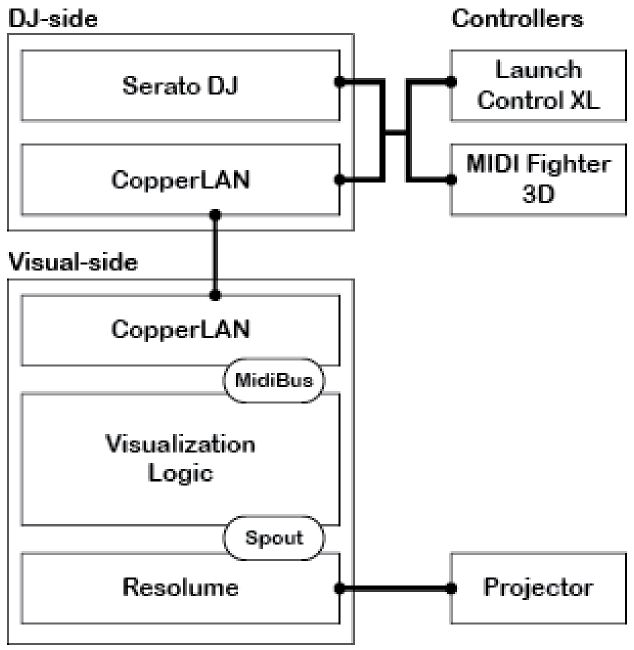

Figure 1: The structure of different devices and softwares

Lastly, the visualizer exports the output graphics towards the projection software. This is performed on a regular basis during run-time, in order to be used for live performances. The projection software sends the graphics received onto a projector via HDMI, producing the end result.

2.3 DJ-side setup

The DJ-side setup is mostly independent from this project; a regular laptop with DJ software connected to multiple MIDI controllers will suffice. The only additional requirement is an Ethernet connection, and software to send the MIDI signals to the other side.

In this project, we used the *CopperLan* networking framework, which allow creating "virtual MIDI cables" via Ethernet connections. We connected two MIDI controllers into the `VMIDI 3` virutal input port of the visual-side computer.

The controllers used are:

- Novation Launch Control XL (abreviated as LCXL hereafter)
- MIDI Fighter 3D (MF3D)

The DJ-side computer is a *Macbook Pro* connected to the above controllers. *Serato DJ* is used for producing and mixing audio.

Figure 2 shows the two controllers used in this project. In the LCXL, there are 8 columns of sliders and buttons; each of the first to seventh column is used to control a specific "wave" whereas the eighth column is for controlling movement of the overall visual. 6 out of the

8 knobs in the second-topmost row are also mapped. These are covered in more depth in sections 2.5 and 2.6.

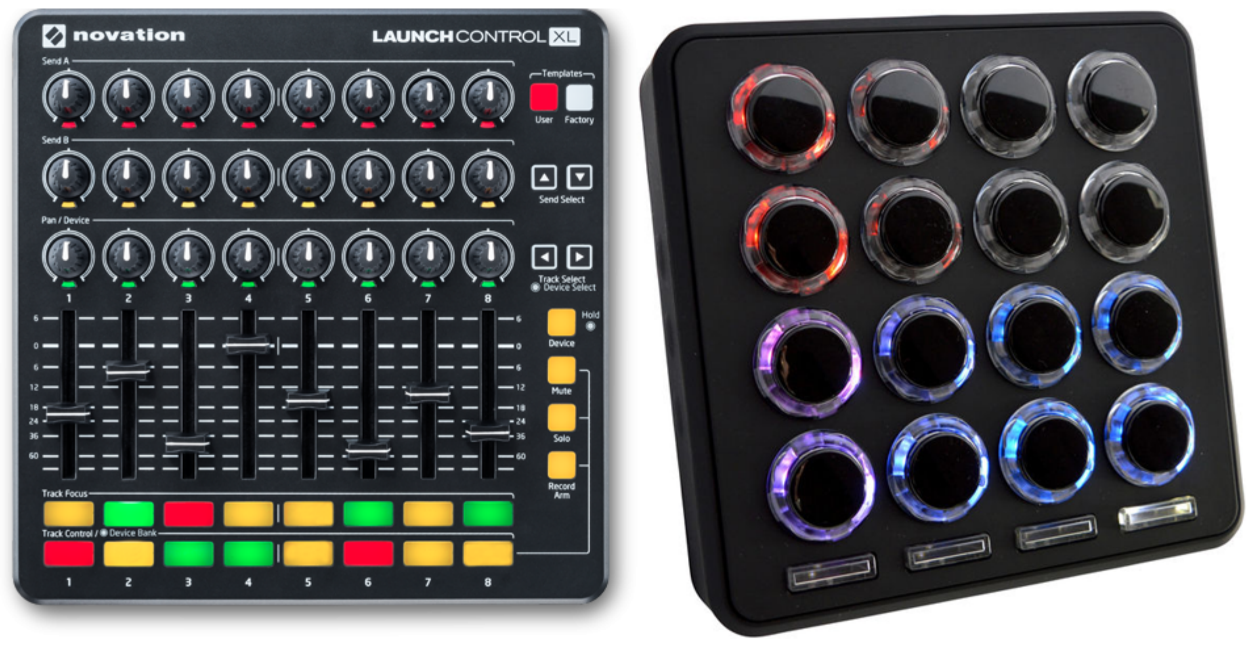

Figure 2: Controllers used in this project

2.4 Receiving input

The visual-side computer also requires *CopperLan* to be installed to receive the MIDI signals sent over. The visualizer application is programmed in Processing, where I used the *MidiBus* library so that an input signal from `VMIDI 3` will trigger handler methods in the program, as shown below:

```
void noteOn(int channel, int pitch, int velocity);
void controllerChange(int channel, int number, int value);
```

The MIDI protocol is able to identify two types of input controls: **buttons** and **faders**. Pressing down a button will trigger the `noteOn` handler, where `channel` and `pitch` gives a unique combination for each button.

The fader, on the other hand, represents an analog input and triggers the `controllerChange` handler. This is either a knob or a slider on the LCXL. `channel` and `number` again identifies the fader that is changed, whereas `value` represents the value of the fader from 0 to 127.

2.5 Waves visualization

The visualization is based on a large triangular "mesh", which is 125 triangles wide and 60 triangles long. Each vertex on this mesh has various properties associated with it, most importantly the height. By changing the height of individual vertices, the “waves” can be visualized.

There are 7 waves controlled by the left seven buttons at the 2nd bottom row of the LCXL. The waves will appear at defined positions on the mesh, forming a hexagon pattern. Pressing the button once will cause the visualizer to create a wave in the corresponding position every 4 seconds; at the same time, the DJ software will also produce a note at the same interval. Thus, the DJ can initiate a pattern of audio notes and visual waves by pressing the buttons at the correct time. This pattern can be changed as the performance progresses. Pressing the button for a second time will turn off the corresponding wave.

The height of a given wave is calculated by a sine curve with exponential decay:

$$h(x,t) = Ae^{-\lambda x}\sin(2\pi(\frac{x}{f} - \frac{t}{v}))$$

This is a function of x, the Cartesian distance of the vertex with the center of the wave; and t, the time elapsed after a wave is triggered, so that the wave will spread out as time passes.

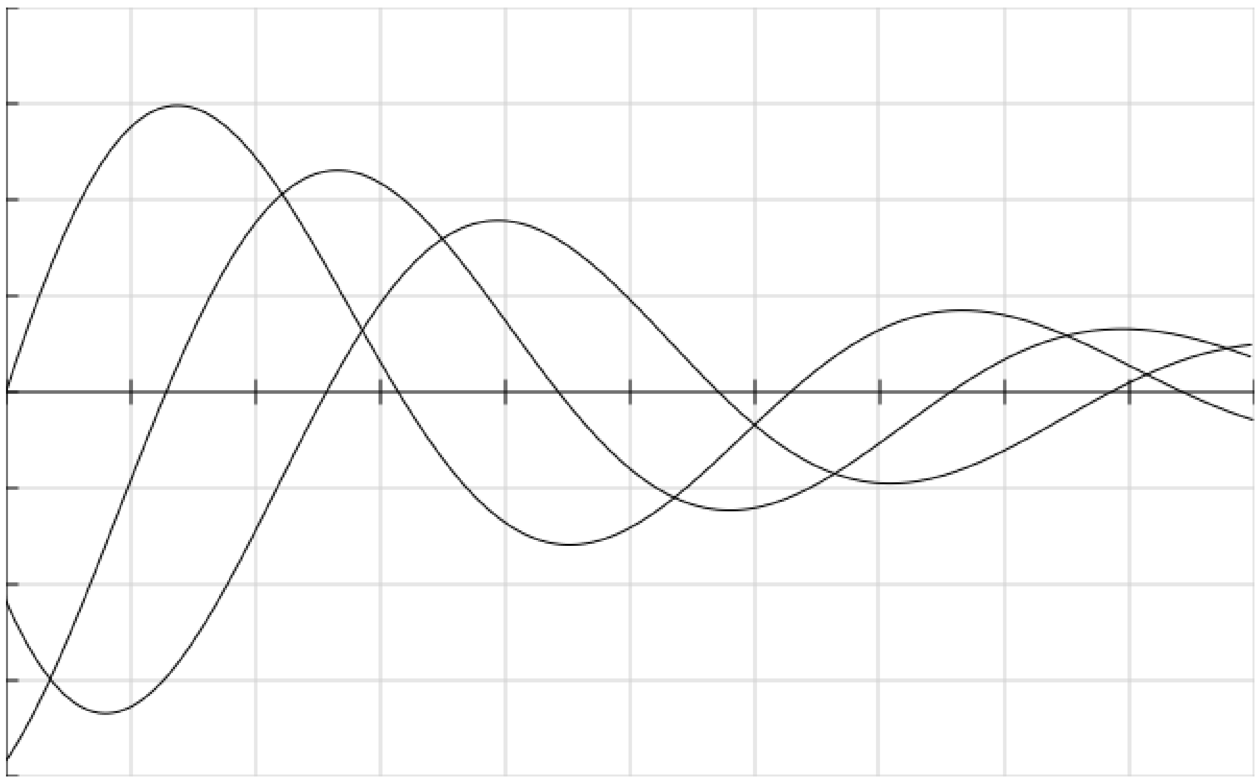

Figure 3: $h(x,t)$ at three values for t

A is the amplitude of the wave, denoting its highest point. This is controlled by the sliders above the buttons; each slider corresponds to the height of a wave. The same slider also changes the color of the tip of the wave.

The other parameters are fixed and do not change according to MIDI input. They are λ, the extent of decay in height as the wave spreads out; f, the frequency of the wave; and v, how fast the wave moves out. Lastly, we only visualize one cycle of a wave or else the visualizer would look like a ripple tank. This is done by substituting $h(x,t)$ with 0 if the distance x is outside its correct range.

Figure 3 shows the general shape of the $h(x,t)$ function for three points in time. The y-axis ($x = 0$) is the center of the wave, and since x is the absolute distance, values for $x < 0$ are not shown. As time progresses the curve is translated to the right, and the wave spreads out. Note that in the visualizer, all parts of this curve except for one wavelength-long of wave is trimmed, so there will be at most one peak and trough at each given time.

In the DJ software, there is also a fader that adds an echo effect to the sounds produced, as well as changing its tone to a deeper sound. In the visualizer, I decided that this will make the waves drop to a deeper level, so that it now resembles rocks dropped into water, attempting to match the sounds played. This is done by multiplying $h(x,t)$ by a multiplier p when $h(x,t) < 0$, i.e. the vertex is pulled down. The value of p can range from 1 to 2 based on the fader value.

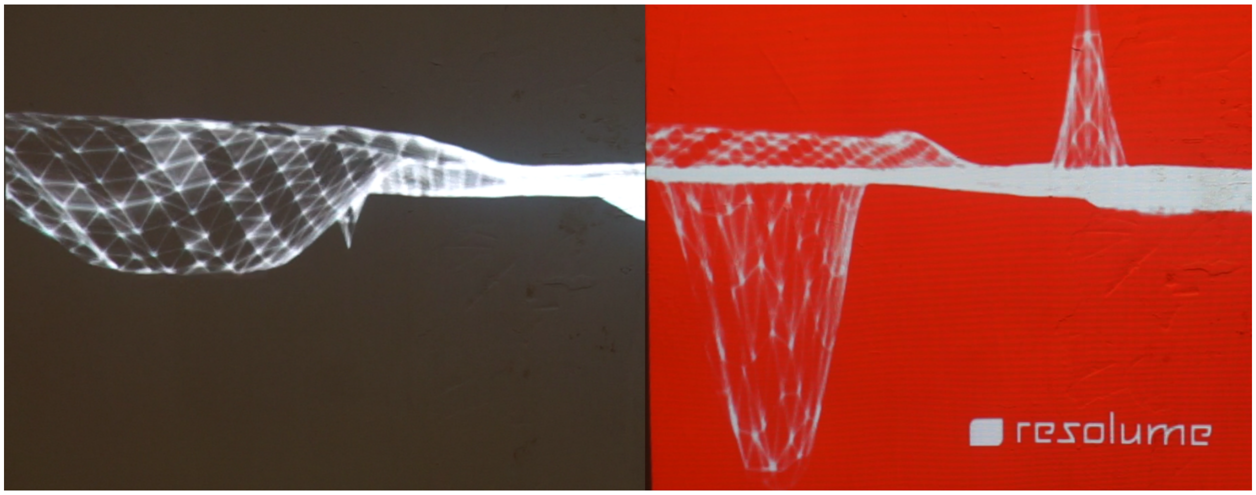

Figure 4: Waves without (left) and with (right) being drawn deeper

2.6 Other visualizations

We also attempted other visualizations that affect the overall presentation.

One can notice that there are 8 sets of sliders and buttons in LCXL. While the 7 sets to the left are used for controlling the timing of waves and their height, the rightmost set is used to control the movement of the whole triangular mesh. To give the mesh a randomized, organic feeling, the heights of the individual points are randomized. However, pure randomization with make the mesh look spiked and eccentric. Hence, I used the `noise(x, y)` native method found in Processing, which generates *Perlin noise* such that adjacent points will not differ in height significantly. This gives the feeling of natural landscape / terrain.

Furthermore, `noise()` always return the same value when `x` and `y` is fixed. With this in mind, I have added an offset value to `y` (the front-back axis). This offset can be increased as time progresses, giving the illusion of the terrain moving forwards.

The mesh is stationary when the visualizer is initialized. When the 8th button is pressed, the mesh moves forward, increasing the feeling of a flying terrain with waves superimposed onto it. The 8th slider then controls the speed that the terrain moving (i.e. how much the offset in `y` is increased per animation frame). When the 8th button is pressed for a second time, the terrain stops moving again.

Other visualizations include:

- The main camera angle, from looking at the waves from the side to a birds-eye view of the waves.
- Tilts in camera angle, using buttons in MF3D
- The lighting of the whole mesh.
- An overlay pattern onto the mesh, which gives the illusion of the mesh forming into a 3D cube pattern.
- The background color. It changes to bright red to match with the change of tone in audio.

2.7 Projection

The projection is done using *Resolume Arena 5.* To transfer the Processing visualization into Arena, *Spout*, a real-time video sharing framework, is used. In the visualizer, the Spout library for Processing is imported and each frame is sent out. The Spout plugin in Arena is setup to receive these frames, and project them using the projector connected to the visual-side computer's HDMI port.

3. Demonstration

The following demo video shows the various functionalities implemented in the visualization application. The screen to the right shows Allan playing the music in real-time using the LCXL and MF3D controllers; the screen to the left is the visualization projected onto a wall in the Off Limits Studio.

https://www.youtube.com/watch?v=thf3bprqSQY

This demonstration can be separated into 3 stages:

- The first part starts from a side view, with a basic pattern of waves and notes played repeatedly. The height of the "terrain" is gradually increased, which after some time starts to flow and advance towards the viewer's direction. This culminates in violent shaking of the camera angle (by quickly alternating buttons on the MF3D side), which is matched with a rattling sound. The agitated motion continues for a while and dies down to the original simpler pattern.

- The second part begins with a change in wave looping pattern, followed by the deepening of the tone of notes. The waves can be seen as dropping significantly deeper below the surface compared with before. The background is turned to bright red, contrasting with the monotonic white waves. The camera raises to view the waves at an angle above.
- The last stage is indicated by the edges turning dark and a cube pattern takes over the mesh. This is followed by a new pattern, pressing the 7 wave buttons at the same time such that the waves appear together forming a hexagonal pattern. The background returns to black and the camera-rattling effect is shown again in the ending.

4. Conclusion

This project focuses on combining visual effects with a DJ live performance, by taking in MIDI signals from the controllers and drawing with Processing in real time. The result is a visualization tool based on the concept of superimposed waves and organic flow meshes, which changes in shape and color through various calculations over time, which can be projected onto a flat surface via a projector.

The biggest limitation comes from the large amount of vertices in this visualization model. Originally it was planned to introduce multiple layers of the mesh stacked on top of each layer, but doing so gives a reduction in fps and overall response. This is possibly caused by the continuous exporting of video data to the projection software.

A future possible area of improvement is to map the projection over a 3D object. It is known that *Resolume Arena* allows slicing the video output into separate parts, which can be distorted and mapped towards the sides of real 3D objects. There are numerous existing examples of ready-made visual performances projected on building services, so it would be a truly interesting idea to do the same to DJ real-time performances as well.

5. Acknowledgement

Thanks must be given to Allan from Off Limits HK, who has not only provide space and the required tools for this project, but also spent many hours in helping testing the visualization system. He has given useful advice and provide ideas on the artistic aspect of the visualizations, as well as how the visualizations can be tweaked to resemble the changes from the incoming audio.