Towards an Artistic Synthesis of Audio/Video Composition

Steve Wilson

Center for Research in Arts, Technology, Education and Learning, Wichita State University, Wichita, Kansas 67260, U.S.A.

1. Introduction

The quest for innovation and experimentation in art is ongoing. Early electronic music explored by Babbitt, Stockhausen, and others provided a new creative outlet for composers. Computers are now capable of complex real-time video manipulation, which, when combined with electronic sound synthesis, can create an exciting new mode for artistic expression.

Audio and video manipulation techniques were researched to develop a programmable application to control the presentation of live video through the manipulation of information derived from either prerecorded or live music. A vocoder, a set of fixed-frequency bandpass filters distributed across the audio spectrum that generates a value proportional to the amount of energy at the frequency track by the filter,[1] was used to divide the spectral output of the audio stream and send specific frequency bands to different accumulators. Once the audio frequency spectrum is divided, the numbers sent to each accumulator are used to control different aspects of the video process such as camera selection, filter type, and video effects.

This project consists of developing a programmable application in which to implement this audio-video synthesis. To demonstrate the application, the researcher composed a musical work demonstrating a coherent synthesis of audio and video in the form of a prerecorded musical composition with a real-time, fully automated video presentation.

2. Discussion

The first step involved determining the best software tools with which to code the audio-video interactivity, which would, in turn, drive the selection of operating system and programming environment needed for the project. Pure Data[2] (Pd), a graphical programming environment, allows the composer detailed control and composition of aural and visual material in a real-time environment. Also, it is particularly useful for the proposed research because it provides the programmer a rich toolset to express the creative concept being developed. Pd is primarily an audio application, but there are a number of externals (libraries that run inside of Pd) that may be used to control video. After testing many of the available video object libraries, PDP, PiDiP, GEM, and Framestein; PDP and PiDiP seemed the richest libraries in which to code the video manipulation.

Selection of this toolset indicated that a Linux platform would be the most flexible for my intended task. I choose to use the Ubuntu/Debian distribution of Linux running on a 1.7GHz PC with 1GB RAM. Using Ubuntu rather than MS Windows required the location, compilation, linkage, integration, and testing of each of the externals from source code.

To implement and control the audio video presentation and interaction, some additional equipment was necessary: four Logitech QuickCam 4000 Pro web cameras, an M-Audio UC-33e MIDI controller, and a USB 2.0 hub. The Logitech cameras provided 640x480 video resolution at 30 frames per second, and the MIDI controller provided tactile control over the many video effects parameters with 47 assignable MIDI controllers.

Binary distributions were not available for the selected externals, which made it necessary to compile both PDP and PiDiP from source code. This was, in fact, recommended by the developers to ensure optimized system performance and also guaranteed use of the most current available version. In order to build these tools it was evident that three more objects would be necessary, GEM2PDP, PDP2GEM, and Pix_2PDP. These support tools were also built from source code. This began the lengthy process of trouble-shooting each external because of the many dependencies needed.

After successfully executing Pd and the external video libraries without errors, programming began. The first steps involved building four inputs to read from the video devices, four outputs for the finished video, an analogue to digital converter for live input from a microphone, and an audio file player to stream the composition from the disk. Upon attaching the four cameras, it was immediately evident that the computer architecture supporting PDP was not nearly powerful enough to handle so much incoming video information. After exploring the option of streaming video from a number of computers dedicated to one camera each, it became necessary, because of time constraints, to scale the project back to one camera. This reduced the CPU load considerably, allowing more processor power for manipulating the video.

The frequency spectrum was divided exponentially to both save CPU power and to correspond to the musical octave (the interval between any two frequencies with a 2:1 ratio). Each bandpass filter's center frequency was tuned to an F-sharp in nine octaves ranging from C 16.351Hz to C 8372.018Hz[3] with a Q of 10 percent. So that every change in video parameters would correspond to clearly audible audio frequency changes, and thus making a greater impact on the listener/viewer, the audible spectrum (up to 20kHz) was truncated to 8372Hz. These filters send their numbers to nine bins (one for each octave) and are then routed to various parts of the program to control a variety of parameters.

Each bin's output is routed to a number box, which decides where the signal will go next. Each number box is connected to a pair of objects designed to only let numbers greater than a determined amount through – one of lower value and one of higher value. If the numbers from the bin are high enough, they pass through and trigger a toggle, which is then sent on to various video parameters. For example, if bin number 2 (32-65Hz) receives a number greater than 60, it routes the video feed through the 'pdp_mosaic' patch until the number falls below 60.

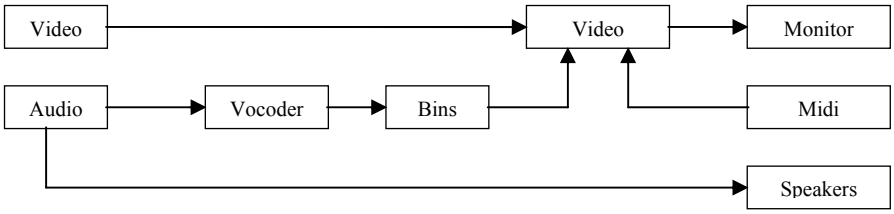

Fig. 1. Signal Flow

3. Conclusions

The application was able to successfully stream audio, process the incoming information, and derive live video control. It creates a variety of video effects based entirely on the audio composition itself and would be different for every work. It also presents interesting possibilities when using a microphone instead of streaming prerecorded audio. The camera could watch the performers who could, in turn, manipulate their appearance on the video screen by playing specific pitches. The most successful aspect of the project is the correlation between musical sounds and changes in video display. After observing the application for a few minutes, one can begin to tell which sounds are triggering the different effects. Consequently, the musical form is imposed on the video creating a visual representation of harmonic form.

The application produced very interesting results and was successful as a proof of concept. Time limitations prevented the full implementation all the planned video effects. After working with PDP for some time, it was found that the creation of original video effects is only possible using the c++ programming language, which was beyond the scope of this project. The next step in this project would be to learn c++ and to begin to create new video manipulation tools for PDP and PiDiP. These new tools will be designed with the intent of audio interaction and will move one step closer towards an artistic synthesis of audio/video composition.

4. Acknowledgements

Thanks are due to John Harrison for expert assistance and inspiration throughout the entire project. His patient tutorials in the use of Linux and how to use open source software tools were greatly appreciated. Thanks also to the Center for Research in Arts, Technology, Education, and Learning (CRATEL) for financial support.

[1] Curtis Roads, in *The Computer Music Tutorial*, The MIT Press, Chapter 5, p 197, 1999.
[2] Miller Pucket, University of California San Diego, http://www-crca.ucsd.edu/~msp/software.html.
[3] Frequency measurements from E.J. Quinby's Musical Pitch Relation Chart.