CMPT 365
Final Project
Tristan Bayfield
Mehmet Atakan Gunal
December 3, 2010

In our final project, we wanted to explore the use of tangible user interfaces (TUI) in relation to audio signal processing and graphic visualizations. Tangible user interfaces are an aspect of multimedia that has been relatively recently becoming available for exploration in the mass consumer / computing enthusiast domain as opposed to limited research domain. However, we believe that it is a very important aspect in multimedia as it provides a very engaging dimension to the user interaction and it will gain more importance in the future as the current technologies in the field mature and new ones get developed. As we are actively involved with different aspects of electronic music, we were particularly excited about the possibilities of this new way of human-computer interaction in terms of musical expression.

The main idea for this project is taken from the concept of the Reactable[1] developed by the Music Technology Group[2] (MTG) at Universitat Pompeu Fabra in Barcelona, where a camera and software are used to track marked objects called fiducials, on a surface. As the fiducials are moved around in the field of view of the camera, their coordinates and degree of rotation can be used as a control data for any parameter. Different combinations of fiducial movements, then, can be used to create musical signals, effects and gestures, and the whole system in turn becomes an instrument itself. reacTIVision[3] is the open-source counterpart of the Reactable project. It provides the tracking software for the fiducials and uses the open-source TUIO[4] protocol, which "allows the transmission of an abstract description of interactive surfaces, including touch events and tangible object states. This protocol encodes control data from a tracker application (e.g. based on computer vision) and sends it

to any client application that is capable of decoding the protocol."[4] It is an extension of the Open Source Control[5] (OSC) protocol and is also used in the popular Microsoft Kinect controller. We used reacTIVision to implement our own surface based tangible controller using a webcam, a table and fiducial markers printed on small pieces of paper.

As the programming environment for the implementation of the controller and the audio processing part of the project, we used the Max/MSP/Jitter[6] interactive graphical programming environment (where Max is a collection of objects for MIDI manipulation, MSP for audio and Jitter for video, collectively they are commonly referred to as just Max). We have written an external object for Max in C programming language using the Max/MSP/Jitter Software Development Kit[7] (SDK), which receives the tracking information, containing the fiducial marker id, the ordering information, x and y coordinates and the rotation angle of the fiducials, from the TUIO client object through the Max patcher (a conventional name for programs written in Max) and computes the audio processing parameters from these control data. The computed parameters are, in turn, returned to the patcher where the audio processing is implemented.

We have built a Max patcher to implement a multiple oscillator Frequency Modulation (FM) instrument. We have six oscillators creating three types of waveforms, each associated with a unique fiducial marker. There are two fiducials each for controlling independent sine waves, sawtooth waves and square waves. When the corresponding fiducial is placed on the surface, the oscillator gets activated, and if it is the only one in the field of view (of the camera), the audio signal of the oscillator is output from the soundcard. The frequency of the oscillator can be changed by rotating the fiducial clockwise to increase its frequency and counter-clockwise to decrease it. If another fiducial is placed on the surface, distanced away from the first one, the audio signal from this second oscillator is also

sent to the soundcard independently, with its frequency being able to change in the same fashion. If however, the second one gets close enough to the first fiducial, then this second one becomes the modulator of the first one (the first one thus becomes the carrier signal in the FM synthesis). When the frequency is changed in this situation, the modulation frequency of the carrier is changed. As the two fiducials get closer, the modulation index of the FM synthesis gets higher. This FM synthesis can be set up in a serial way, with a third fiducial placed within the proximity of the second one to modulate it, or in a parallel fashion where the third one is placed within the proximity of the carrier as well, thus the sum of the two modulators modulating the carrier signal. These carrier-modulator chains can be extended to contain as many fiducials available (limited to six in the current implementation), including the possibility of multiple independent carriers. Depending on the order in which the fiducials are placed and removed from the surface, as well as their proximity to each other, the roles of their corresponding oscillators change between being independent oscillators, carrier signals or modulator signals. At the basis of our implementation of the external Max object that enables this structure lies a weighted, unidirectional adjacency matrix that keeps track of the distances between two fiducials and the order of their entrance to the field. There is a path from a fiducial A to a fiducial B, if A was placed after B. The weights are the distances between fiducials and if the distance between A and B is below a set threshold, then A starts to modulate B. The oscillators and the FM synthesis structure is built with preexisting Max object primitives into a patcher that also serves as binding the program together and being a graphical programming environment, provides a nice graphical user interface. Additional visual feedback is provided by calculating and displaying the resulting waveform (with an oscillator) and the spectral power distribution (with a spectrogram). Moreover, the feed from the camera can be seen at all times as either the grayscale images or the images as transformed by the reacTIVision software for tracking. The tracked positions of the fiducials are visible in both visions.

We set out to accomplish numerous goals at the start of this project. Our objectives included, programming our own Max external object using the MaxMSP SDK; applying audio processing within our external using a sound processing libraries called STK[8] and SndObj[9] respectively, as well as using another library called flext, to enable us do our programming in C++ instead of C, which is what the Max SDK supports; and we wanted to incorporated the third library of objects included with the actual MaxMSP software called, Jitter, which allows for video manipulation. These goals were rather lofty. We also found that incorporating some of the libraries, including STK and flext were going to be more trouble than it was worth getting all the components to cooperate. Part of our problem was also that we initially did not have a very clear idea of what we wanted to accomplish. Thus numerous goals were set. Fortunately however, we narrowed down our scope early on, deciding to focus on FM where the signal processing would be done in a Max patcher and the logic for handling fiducials would be done in the external object, using C. While we are satisfied with the result so far, further improvement would include, the addition of more roles for fiducial objects (beyond the roles of carrier or modulator). Other roles might include amplitude modulation or wave file triggering etc. We would also like to build an interface similar to the reacTable design, though, it would have been too expensive for us to do at this time. We are keeping this in mind for the future, and actually consider this project to be the start of a much longer term one.

More Information

1) http://www.reactable.com/
2) http://mtg.upf.edu/
3) http://reactivision.sourceforge.net/
4) http://www.tuio.org/
5) http://opensoundcontrol.org/
6) http://cycling74.com
7) http://cycling74.com/products/sdk/
8) https://ccrma.stanford.edu/software/stk/
9) http://sndobj.sourceforge.net/