Seth Goldenberg
Final Project Report

M.A.R.T.H.A.
MIDI Activated Real Time Hemispheric Apparatus

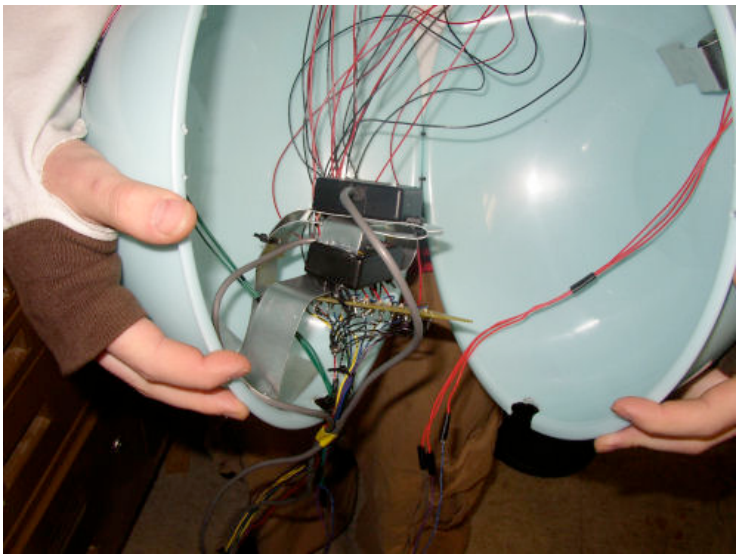
Summary

In creating the M.A.R.T.H.A., we were looking to somehow implement the idea of using a sphere, a simple yet elegant form, to produce musical sounds. Pulling from our days of math long ago, we decided to measure the rotation on the 3 axes of the sphere and use them as controllers. This instrument would have more of an experimental inclination than my previous project, the Flying Fists of MIDI—an instrument that played notes in a scale.

Though the MARTHA can be fairly versatile, we found it was most advantageous to use it as a drum machine or sampler. Rolling it on the forward-back axis controls the speed of playback. This has an interesting effect on sounds of beats and various sound clips from musical recordings or speech samples. Rolling M.A.R.T.H.A. on the left-right axis controlled the filter frequency. M.A.R.T.H.A. may have many of the same capabilities of a cut-and-dry drum machine, but its response to tilt makes it much more dynamic and certainly more entertaining to watch in use.

Materials and Construction

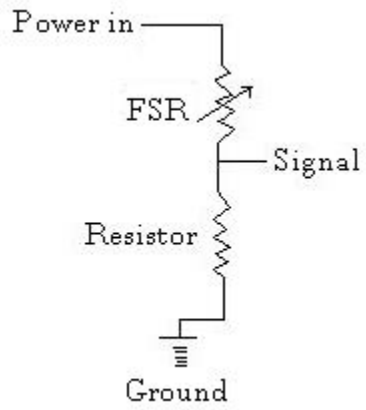
2 robin's egg blue Martha Stewart Living salad bowls
1 circuit board = 10 force-sensing resistors (FSRs), 10 10k ohm resistors
2 G-Force Accelerometers
Lots of wire and solder
Metal, zip ties, and hot glue to hold it all together



Jordan was responsible for M.A.R.T.H.A.'s body. The two salad bowls were placed together to form a sphere-like shape. Jordan cut two holes on opposite poles of the sphere for wires to exit from inside. The breadboard and two accelerometers were mounted on a piece of metal that was shaped to fit inside the bowls. He tried to center the accelerometers as much as possible so that less rotation could produce a greater range. The assembly was then hot glued to the inside of the bowls. The M.A.R.T.H.A kept its sphere-like figure with several zip-ties that fastened it shut.

Sensors

10 force-sensing resistors (FSRs) were placed on the outside of the bowls. Five were placed on each bowl. We used a larger sensor for the thumb. To receive input, I had to construct a bias-reduction circuit. The FSR was used in conjunction with a 10K ohm resistor. I replicated this circuit for each FSR on a breadboard. The FSRs had a range of 0 to about 127. They were not as dynamic as I had hope as I found it difficult to get values below 90.



Our group spent much of the early portion of the project trying to find an appropriate accelerometer. We experimented with a 2-axis +/- 50g and a 1-axis +/- 10g accelerometer. Luckily we stumbled upon two expensive yet unused +/- 2g sensors. When plugged into Max, they gave a range of 38 to 90 on the force of gravity along by rotating the sensor along its axis with virtually no noise. This was a large enough range to translate these values for any Max/MSP function. In Max, we limited the values to the range of the tilt so that if the ball was moved any other way than tilting it on its axis, the values would not exceed the tilt range.

Max/MSP

My main contribution to the project was creating the Max patch. It was fairly simple to get readings from the sensors. Only the accelerometer readings had to be scaled. The biggest decision we made on the patch was deciding to write in MSP. Matt had the idea of using one of the axes to control the speed of the patch, which I had no idea how to do in Max. George, our TA and resident Max/MSP expert, suggested an MSP function called Groove that did exactly what we needed.

Though MSP was fairly intuitive, it was somewhat confusing to understand the paradigm and to find the right higher-level functions for it. For each of the 10 sounds, I loaded audio files into buffers and created a bang that would switch the audio files loaded for each hand separately.

Groove was simple but there were many things I had to pay attention to in dealing with audio. I did not have to worry about retriggering a note. When the FSR is first pressed, it simply plays the note when it goes above a threshold and stops when it goes below it. The FSR also controls the volume of the sound. The audio clips loop as long as the FSR was pressed. I set the audio clips to stop playing when released to make it less taxing on the CPU. It was tricky to understand how everything worked. I was able to get it working from reading the help examples and some help from George.
We used the left-right axis accelerometer to control filter frequency. The function (svf~) was very easy to pick up from the help patch. However, I had trouble getting a filter setting we all liked. The hardest part was getting the center position to sound like there was no effect at all. George helped us out by writing a simple crossfade patch to use with both low-pass and high-pass filters simultaneously. Titling the sphere to the left cut out the low frequencies, and tilting it left cut the high ones. It was exactly the sound we were looking for.

Sounds & Samples

MSP eliminated the need for Reason since no synthesis was taking place. We did have to find audio files we wanted and cut them down to the same length. All the music clips had to be at the same tempo as well. We all searched for samples and beats that we liked with Matt leading the charge. I found a couple of samples and Matt showed me how to edit them in Peak. Matt arranged a good mix of speaking and musical sounds. The beats were primarily hip-hop. The other sounds were pulled from the internet by Matt and me or old records by Jordan and ranged from mariachi songs to a haunted house soundtrack. Since all the sounds were cut to the same size, they fit perfectly with one another when triggered at the same time. It requires a little coordination on the user's part but is pretty simple to just pick up and play.

Problems and Surprises

Working with MSP was certainly a challenge, if only because it was new to me and had to learn it with a deadline in mind. I ran into a couple bizarre problems though. When I

had a counter set up wrong, the FSR was also controlling the speed of the playback. Little idiosyncrasies like this made writing the patch take a lot longer than it should have. Another problem that was definitely not expected was finding a casing for the actual instrument. Believe it not, M.A.R.T.H.A. was not the name we came up with when we started with the idea. We had unsuccessful searches at Target, Michael's, Newbury Comics, and Staples. We didn't find Martha's bowls until I made a trip to K-Mart thanks to Professor Lehrman. Although we were going for a sphere shape, we could not use a ball because we would have to cut it in half to insert the circuits and sensors.

Doing it again

Though we didn't realize it when we started, Groove is simply a kind of pitch bend—sounds get higher pitched as they are sped up. We could have had the same effect in Reason and scrapped MSP. That being said, I enjoyed the challenge of MSP and could write a similar patch in MSP easily now that I have an understanding of the language. It would be interesting to plug in samples of an instrument playing a note and play the M.A.R.T.H.A. as more of an instrument, similar to the NN-19 module in Reason. I would have liked to try adding a third axis of rotation to add another even more subtle effect, such as delay or reverb, given more time. Overall, I was very pleased with the final product and how much we were able to do by the end of the semester.