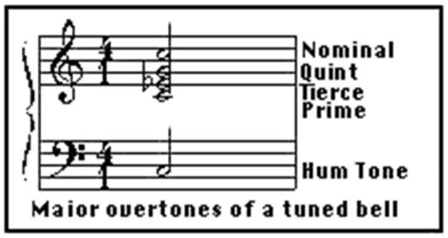
CPSC 432
Pong Trairatvorakul

Final Project Writeup

My final project focuses on recreating the bells of Harkness tower by additive synthesis with the main focus on the decay portion of the sound.

I started off the project by doing research on the characteristics of carillon bells and found that there are five major partials we have to consider: Hum tone (octave below the strike tone), strike tone, tierce (minor third above strike tone), quint (perfect fifth above the strike tone), and nominal (octave above strike tone. This diagram from the website of the Guild of Carillonneurs of North America shows these five partials for a strike tone of middle C.



From http://www.gcna.org/bells.html

After knowing which partials to look at, I analyzed recordings of bells from Harkness Tower using a free software called SPEAR which showed the audio spectrum of the sound. Here, I isolated each of the five main partials and analyzed its amplitude envelope as well as frequency fluctuation. The frequency remained approximately constant throughout the duration of the sound, however the amplitude envelope does vary between partial. One of the major characteristics is the nominal is much louder than the other harmonics but decays much faster. On the other hand, the tierce is initially quieter, but has a longer decay time. I experimented with the constants for each partial until the sound most closely resembles the sound of the decay of the real bells. As a final test, I also showed it to other carillonneurs who are familiar with the bells in Harkness tower, many of whom agreed that the synthesized decay is close to the real decay.

Another aspect of the sound is the overall amplitude envelope. In addition to the obvious exponential decay for all the bells, one notable thing about the real bells is that lower bells are bigger, and thus resonate for longer. I analyzed the decay time for bells around different ranges and found that this was somewhat inversely proportional the frequency with a proportionally constant of 5000.

After reaching a good effect on the decay, I then tried to synthesize the attack sound. This proved to be very challenging since the sound is much more complex. I tried various methods including injecting white noise at the beginning with a very sharp decay, as well as detuning multiple harmonics on the attack followed by a sharp decay of those sounds. After doing more research on the topic, I found that there does not seem to be a well-defined consensus about what constitutes a good attack sound. I tried to analyze the field recordings. However, unlike the decay portion, there were no distinct characteristics. Furthermore, the ambient noise in the recording also did not allow the attack sounds to be easily identified. One hypothesis was that the attack sound consisted of a lot of noise. However, I was not able to test this out with the current program since it would take too much computation time to do so. Thus the submitted version of the sound has an attack that is clearly an attack with a sharp decay. It sounds authentic around the absolute pitch 90 – 100 range, but not so much for lower ranges.

Sound Files:
There are four sound folders submitted. One is the recording of the real carillon, one is the sounds produced by the program, one is the reversed version of that, and one is just the decay.

Composing Music

My initial plan was to demonstrate this sound by making a composition in Haskell so I wrote a function that turns music values into one that would allow the decay of the bells. Essentially, the function recursively changes the durations of each note to be 10 while maintaining the relative durations between notes. Essentially the function is like holding the pedal down endlessly.

However, after finishing the sound synthesis portion, I discovered that Haskell takes a very long time to render each sound and runs out of memory even for short 4-note melodies and compiling using GHC with optimizations. Thus, it was not possible to adequately render a whole composition.

As a result, I decided to export individual notes from Haskell then compile in Logic Pro.

I wrote an auxiliary function to export all the notes in the range of the carillon.

How to use the program

The program itself should be very self-explanatory.

The major functions are:

`writeNote ap`

Takes an absolute pitch value then renders a wav file of that note for 20 seconds

`writeAll lo hi`

Calls witeNote for pitches from lo to hi (inclusive)

Prints the progress.

`toCari a`

Takes Music a as input then outputs a Music a as output where each note is lengthened to have length 10 (eg. 20 seconds) but maintaining the length between notes.

This is analogous to holding down the pedal.