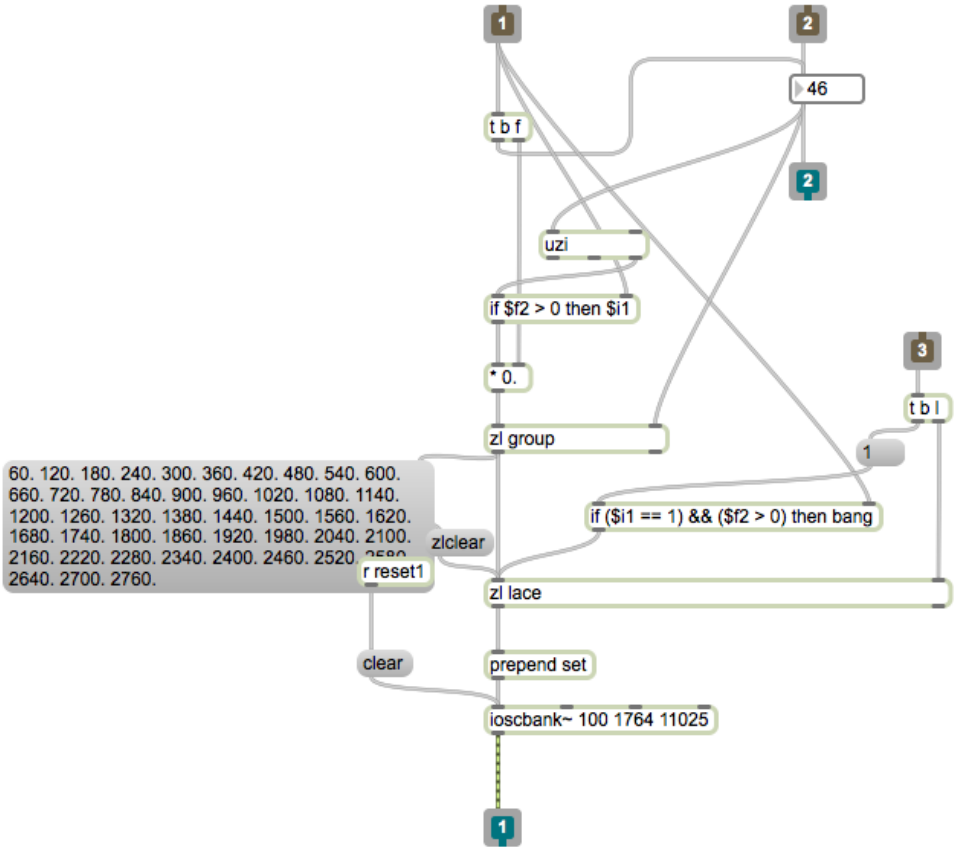
MarjiSynth

By: Marjan Kalanaki

Jan 2014

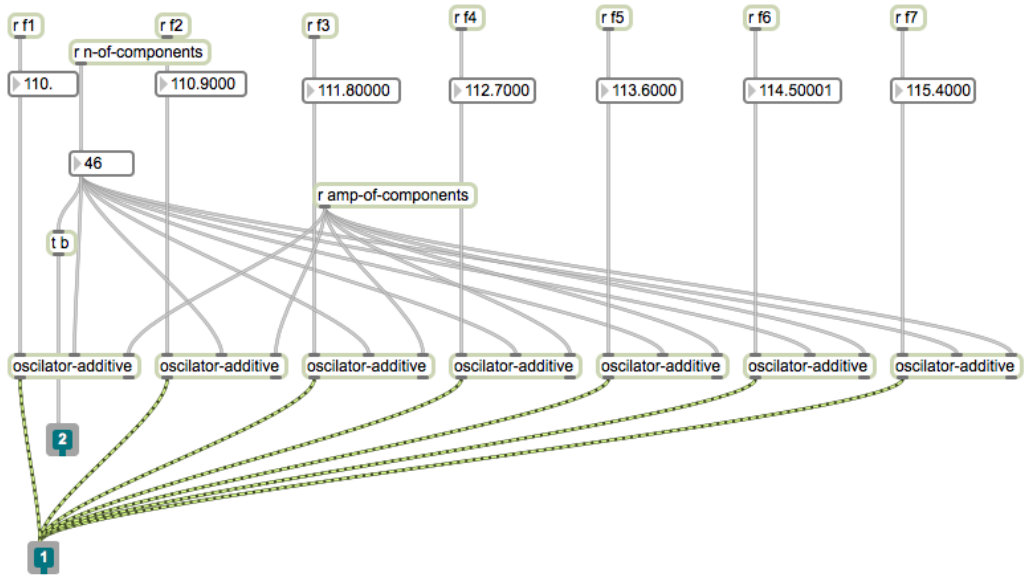
This is a brief report about my work of building two devices using different synthesizing techniques in MAX/MSP application and how I could approach the ideas for this piece of work.

Studying the "Electronic Music and Sound Design" book by Alessandro Cipriani and Maurizio Giri, I was inspired to create a synthesizer that could produce a waveform with selectable number of harmonics and selectable volume and phase for each harmonic. In the book there was a kind of similar approach making use of wavetables readable by "wave~" object but I was thinking of applying the "ioscbank~" object which is an oscillator bank. It seemed to be very complicated at first but I could gradually figure out how to proceed. Using the uzi object and giving it the number of components (harmonics) for example 32, it would emit 32 index values from its right outlet. Multiplying these values (1 to 32) by the fundamental frequency would produce the frequencies of all harmonics and then grouping them by zl group object they could get ready to go into the "ioscbank~". On the other hand, simultaneously the amplitude values of all the harmonics are outputted as a message from the multislider. Both groups of frequencies and the amplitudes could be sorted passing through the zl lace and then with giving the resulting message the word "set" at the beginning they could be inputted to the "ioscbank~" object. This way a waveform with controllable harmonics could be produced. The algorithm of this patch is shown in figure 1.



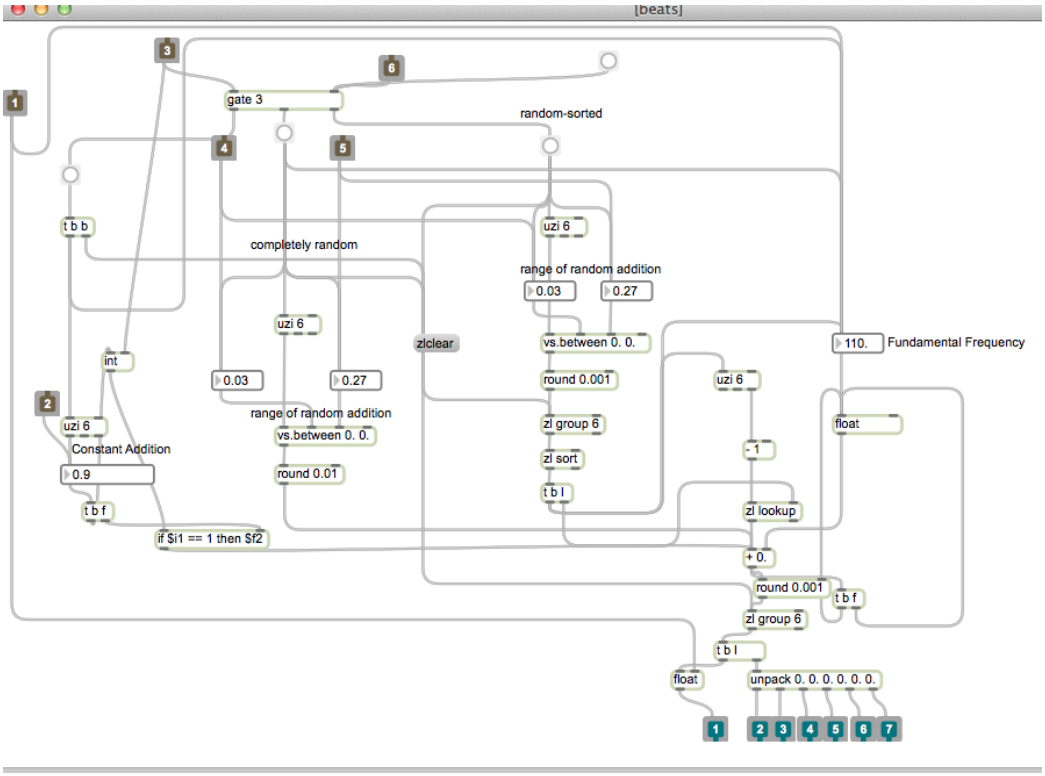
(Figure 1)

The book also drew my attention to the phenomenon known as beats, which is a cyclical variation in the amplitude of a sound risen from two pure tones with very close frequency but not necessarily same amplitude. Considering two or more complex waveforms, the relation between all of their harmonics can create a rhythm of beats, which is an extremely interesting concept. I duplicated the additive synthesizer patch 7 times as I wanted to explore the effect of mixing the created sounds with very close frequencies considering that the frequency difference must be less than about 12.5Hz for two pure tones to be heard as beats. This mix could create beautiful rhythms by just exploiting the beats. Figure 2 shows the patch for mixing the 7 complex sounds.



(Figure 2)

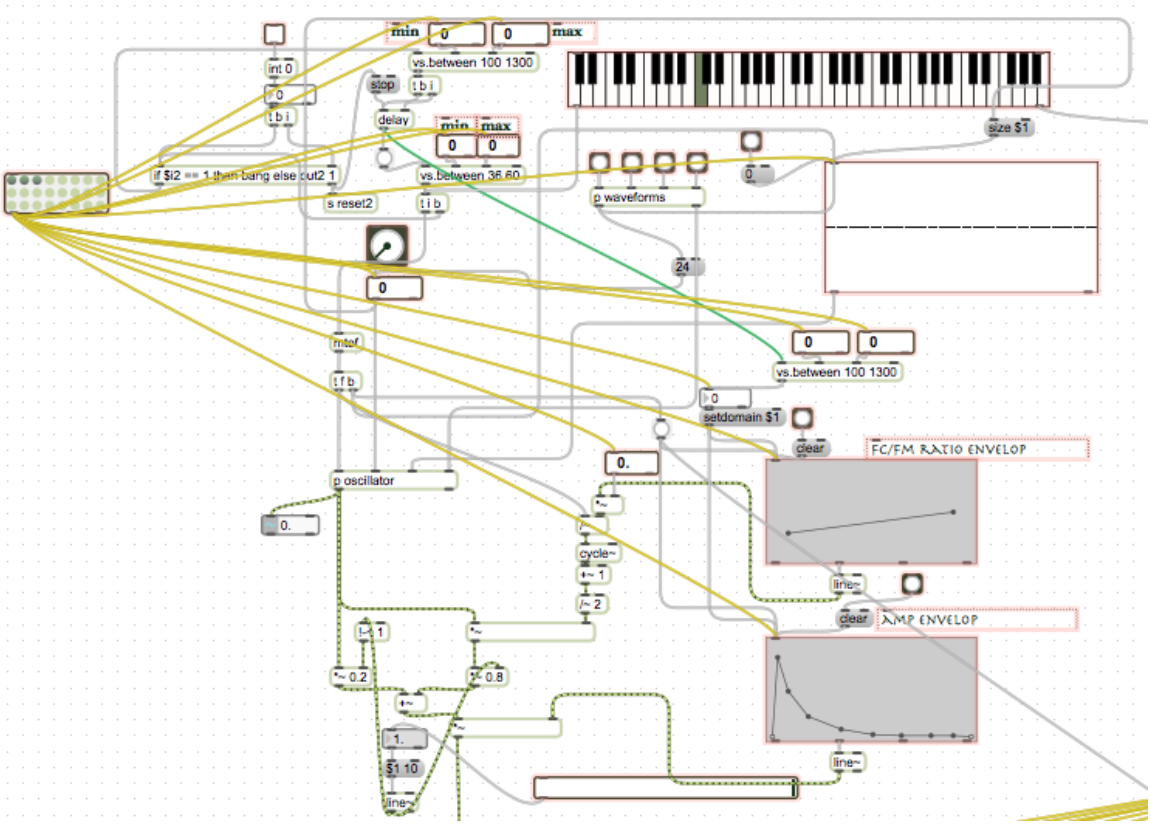
After that I started to create an algorithm that could automatically produce the fundamental frequencies since different relations could produce different rhythms and it would have been much fun to experiment with the difference in frequencies. My first approach was to set the same selectable intervals between the frequencies. I also worked out two different systems capable of creating 7 random numbers with selectable minimum and maximum values, then the first number would be added to the first frequency, the second would be added to the result of the previous addition and so on. The first system was to create completely random numbers but in the other one, the created numbers would be sorted in ascending order. This way the intervals between the frequencies could be more related to each other (As previously mentioned, I wanted to experiment with the interval relations). Figure 3 displays the 3 algorithms for achieving the fundamental frequencies for 7 additive sound generators.



(Figure 3)

The second device that I built was an amplitude modulation for which I wanted to have a complex waveform as the carrier. Therefor with nearly the same technique as the first device, I created the waveform using ioscbank~ object with controllable number of harmonics and controllable amplitude and phase for each harmonic. To modulate the amplitude of the carrier I made use of the cycle object whose frequency could be controlled in a number of ways: Using a function object and setting its y-axis values between 0-10, then multiplying its outputs by a constant selectable number and dividing them by the fundamental frequency of the carrier. As a result of the envelope drawn in function object, the frequency of the modulator could change, producing a stronger modulation. I also applied another function object to create an amplitude envelope and multiplied its outputs to the resulting signal.

In order to increase or decrease the modulation depth I outputted a separate signal from the carrier giving it to a multiplier object and also sent the modulated signal to another multiplier object. But there had to be a relation between these multiplications because the signal must not exceed 1, which is the limit over which the clipping occurs. To increase the flexibility of changing the modulation depth I added a slider and set the minimum and maximum level between 0-1, then sent it's value once to the modulated signal and once to the non-modulated signal after being subtracted by 1. The explained process of building the amplitude modulation can be seen in figure 4.



(Figure 4)

For the amp modulation device, to be able to better control the signal in different frequencies I decided to add 4 filters including a low pass, a high pass and two peak notch filters using 4 “filtergraph” object connected to a “pack” object so their outputted coefficients could all be sent to the “cascade~” object.

In addition, I added a section to this device where different kinds of plug-ins could be loaded and used if they were located in the same folder as the max/msp patch file.

In my work I made use an abstraction named as “vs.between” that I downloaded from the website of the Electronic Music and Sound Design book and also I used one of the patches from the book in my amplitude modulation (to have some default waveforms as the carrier) which can calculate the amplitudes of the first 24 harmonics of the saw tooth wave, square wave, impulse wave and triangle wave.