Granuvolver

Edderic Ugaddan

December 11, 2013

Contents

1 Gratitude

Special thanks goes to Dr. Benjamin Broening and Dr. Barry Lawson. First of all, thank you for being wonderful top-notch professors. I've really enjoyed all my classes with you, learning from you, and having one-on-ones with you,

no matter how stressful it was at times. Thank you for helping me create my interdisciplinary major, Computer Music, and taking me under your wings as an advisee, giving me advice and showing me a different way to look at things whenever I was getting too caught up with an idea. In the end, I feel that I've become a much better programmer, musician, and most of all, person.

2 Introduction

The objective of this Computer Music interdisciplinary senior thesis is to explore the combination of Granular Synthesis and Convolution. This entails programming a custom-made Max for Live (M4L) MIDI instrument that combines the two technologies, analyzing the sounds that come out of this combination, and finally creating music using the resulting sounds as source material. Granular synthesis is the method of producing interesting, texturally rich sounds by overlaying grains, sets of samples that are about 100 ms. or less in length. Convolution describes the mathematical process of sliding multiplication of two signals. To the knowledge of the author, no MIDI instrument has been made that is specifically made only for exploring the combining of these two technologies.

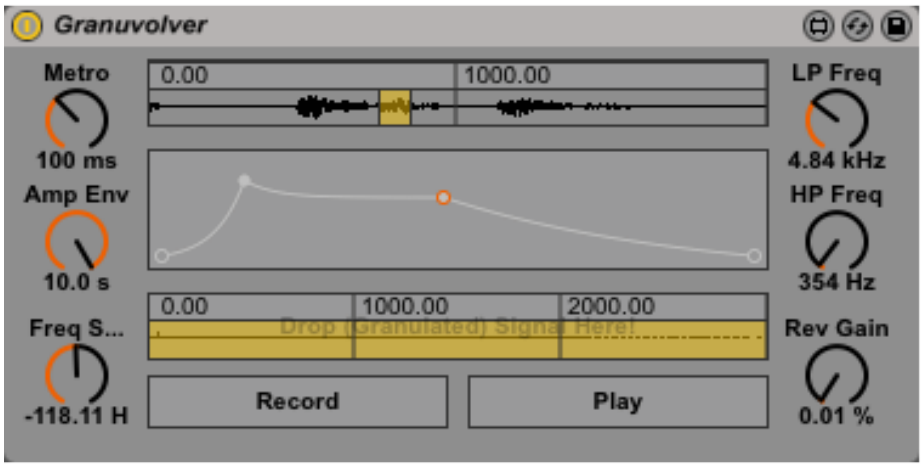


Figure 1: Granuvolver

3 Granular Synthesis

3.1 What is it?

Granular Synthesis is the creation of sound via means of aggregating various sets of samples called grains. A grain is defined as:

> . . . a brief microacoustic event, with a duration near the threshold of human auditory perception, typically . . . [being less than] one tenth of a second, serving as a building block for sound objects.[1, Loc. 1024]

It was largely inaccessible in the past due to technological limitations of analog circuitry and tape. However, thanks to the advances in computing and hardware, composers are now able to digitalize audio and work with the latter in a sample-by-sample basis, thereby letting composers explore the "extreme limits of perception and performance."[2, Loc. 334] Combining these grains helps us create animated, texturally rich sonic atmospheres.[1, Loc. 1024]

4 Convolution

4.1 What is it?

Convolution, in digital signal processing, is the mathematical process of sliding and overlapping two signals together, point-wise multiplying the coefficients, and summing the products. In more mathematically-precise terms,

$$(f * g) = \sum_{m=-\infty}^{\infty} f[m]g[n-m] \tag{1}$$

where f and g are the two signals being convolved.[3, pp. 121]

4.2 Reverberation

Convolution, amongst music producers, is most famous for simulating the reverberation of physical spaces by a sound object. For example, consider recording a drum that is hit in a small room, and then recording the reverberation of a large concert hall (stimulated by a quick transient such as a loud clap). Convolving the two signals results in a "smearing"; the transient of the drum hit is smeared by the long reverb tail of the concert hall. The acoustical result would sound as if the drum hit was played in the large concert hall.

4.3 Algorithmic Efficiency

The naive implementation, which is called "direct" convolution, has a time efficiency of $O(n^2)$, where n is the number of samples. This is terribly inefficient, considering that n is quite large even for a small amount of time–the standard CD-quality audio signal has a sampling rate of 44.1 kHz (three seconds of recording, for example, would amount to greater than 10^5 samples, and the resulting floating-point operations for convolution would be in the order of 10^{10}).

Performing the Fast Fourier Transform (FFT), the point-wise multiplication, and the accompanying inverse FFT of the two signals shaves the time complexity to $O(n\log n)$. A *drastic* improvement indeed. However, since n is quite large, the

convolution will still incur lots of latency and thus cannot be considered "real-time." More complicated FFT algorithms take advantage of the audio signals' *linearity* to speed up the convolution process. Partitioning the signals and doing many smaller FFTs in parallel helps decrease latency to the point of making it usable for real-time convolution.[4] Dr. Alex Harker's real-time implementation, called *multiconvolve~*, is a Max/MSP object that is used in this thesis to help speed up the music making process.[5]

5 Granuvolver

5.1 Why Max?

Granuvolver was originally made using the cross-platform Virtual Studio Effect (VST) SDK using C++. VST, developed by Steinberg, Inc. is a very popular audio plugin format that is supported by major Digital Audio Workstations (DAWs), such as Logic, Ableton Live, Cubase, ProTools, and Reaper. Unfortunately, implementing VSTs is not trivial. VST SDKs do not come with built-in audio effects or GUI elements. Also one can easily get stuck working more on IDE configuration settings than on actual programming. See section on build settings for Visual Studio, for example.

Max, a proprietary software developed by Cycling 74, is a graphical, programmable, patching environment written in C. Objects have inlets and outlets that one can connect to other objects. One can create more powerful objects by making one that is composed of many built-in objects. One can also create external objects in Javascript and Java very easily. However, if one wants more efficient performance and does not mind doing manual garbage collection, one can even develop externals in pure C, C++, and even in Objective-C. Unfortunately, a regular undiscounted license is a hefty $399.

Yet, Max ultimately became the final implementation environment solely for several reasons. First, it allows fast prototyping. Max (vers. 6.1) has over 1,000 native objects that can be quickly used, such as objects for dropping (and recognizing paths) of audio files, displaying waveforms and selecting sections of the waveforms, and filtering an incoming audio signal. By providing easily editable and usable GUI elements and providing DSP objects, one can really put more time into actual music production and less into development. Second, users can develop Max for Live (M4L) objects, which can be easily and seamlessly patched, edited, and used within powerful Ableton Live 9 software for further editing and processing.

5.2 Features

1. Stereo-based granulation of audio signals.

2. Granulator uses Gaussian Normal Distribution with editable mean and pseudo-width to stochastically pick the next area to iterate.

3. Polyphonic (up to 3 simultaneous voices).
4. Subtractive synthesis (using a combo of low-pass and high-pass filters)
5. Slope of curves of amplitude envelope are editable.
6. Granulation of a signal can then be recorded into buffer to be used for real-time convolution of the granulated signals.

5.3 Installation

5.3.1 OSX 64-bit

Granuvolver has been tested on OSX 10.7.5 Lion using Ableton Suite 9.0.6 and Max 6.1.1 64-bit. It should work on Ableton Live 9.0+ as well, as long as the user has Max for Live plugin installed.

1. Open Max 6.1.1. Place Granuvolver.amxd along with its subpatches (ones ending in *.maxpat*) in the search path of Max. While at the menu bar, select *Options* and *File Preferences*. Press the + button (in the bottom left corner) to add the Granuvolver folder. See Figure 2 on page 5.

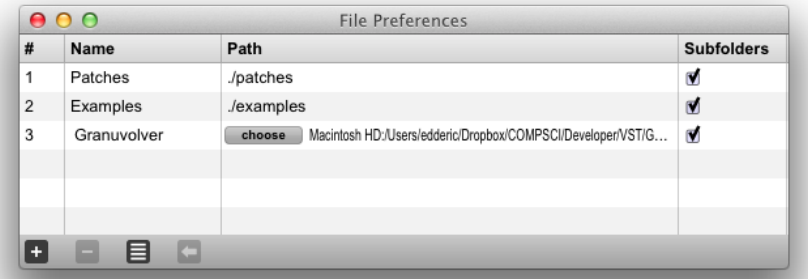


Figure 2: Added the Granuvolver folder to the search path

2. Open a new (or existing) project by Ableton Live (or Suite) 9 and place *Granuvolver.amxd* onto a MIDI track. See Figure 3 on page 6.

5.4 How to use

1. Drag a *stereo* (2-channel) recording onto the waveform object on the top-center third half. Stereo (as opposed to mono) for version 1 is a must–mono causes the granulator to only play the left channel. After doing so, the waveform will only show the left channel in order to save screen space. See Figure 4 on page 6. If a stereo audio file is not available, user can record

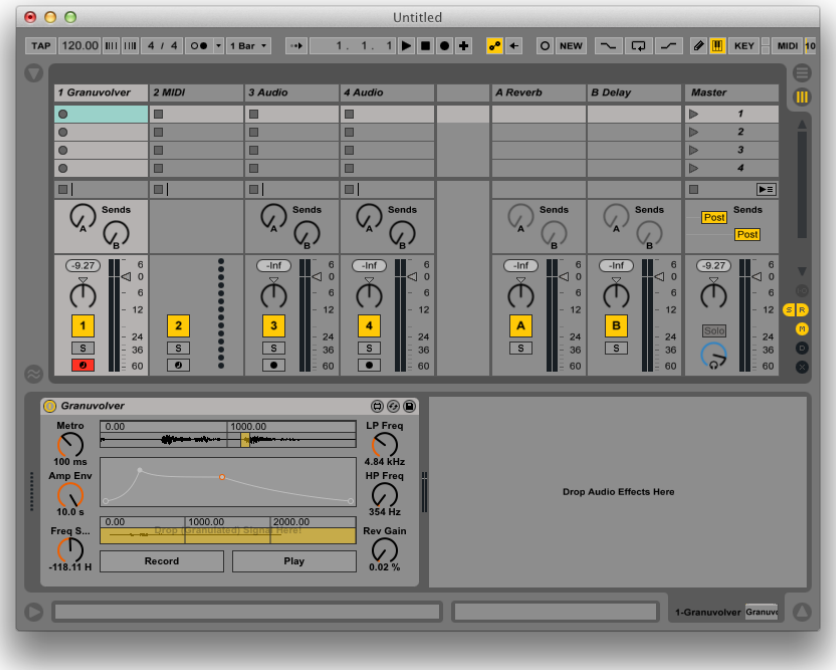


Figure 3: Added Granuvolver to a MIDI track

.

one. Make sure *I.O* at the right-hand side is selected (yellow). Use an Audio track, and make sure to select *Ext. In 1/2* as *Audio From*. Start recording by pressing a row's circle button right underneath the track name. See Figures 5a and 5b on page 7.

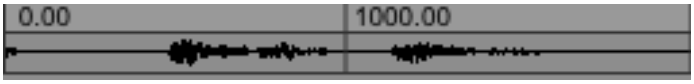
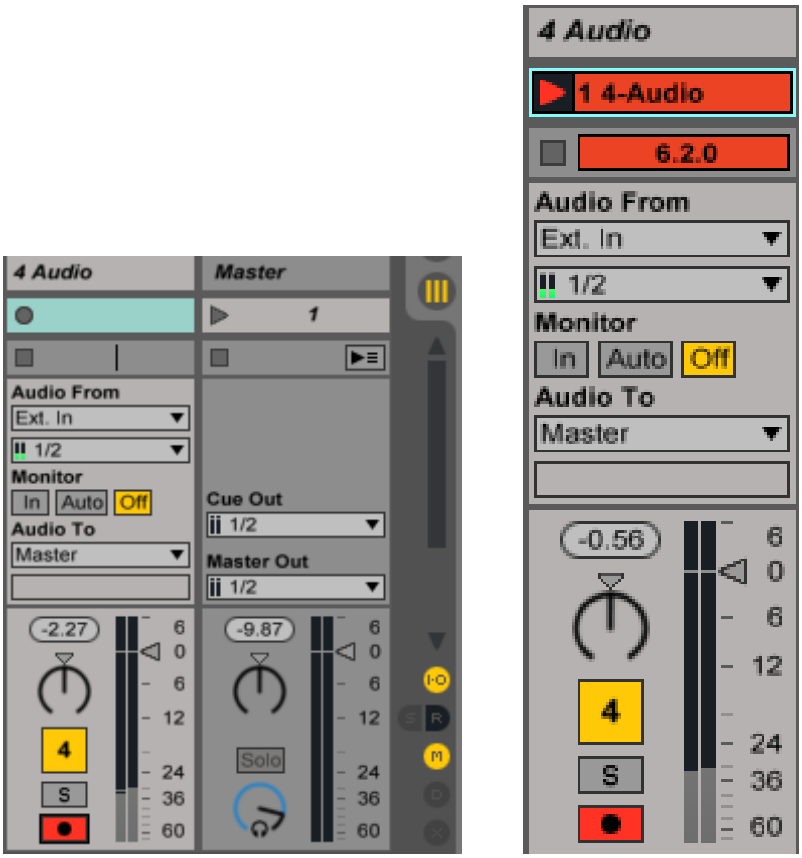


Figure 4: Added a 2-channel audio signal onto the top-center waveform area to be granulated

2. Play MIDI notes. You should hear the waveform. The waveform should be transposed depending on the note pressed.

3. Click and drag on the waveform with an up movement. This section by default is the *stochastic iterator range selector* (you can actually change it as to edit the loop length instead by pressing *CTRL* as described in the next enumerated instruction.) Doing so activates the Gaussian Normal



(a) Setting up (b) Recording into a clip

Figure 5: Recording a stereo signal

pseudo-random generator. The middle of the left and right bounds of the yellow area is the mean of the Gaussian. Thus, the larger the area covered by yellow, the more "random" it should sound. You should hear stuttering in stereo. Select a range for the granulator to stochastically iterate by dragging the selection left or right. See Figure 6 on page 8.

4. Click and hold *CTRL* to be able to switch the *stochastic iterator range selector* into one that controls *loop length*. While holding *CTRL*, click and drag up to extend the loop length, or down to decrease the loop length. The shorter the loop length, the more the sound accumulates high frequencies and clicks, and vice versa.

5. Play around with the *metro* dial. This parameter affects how quickly the Gaussian Pseudo-Random generator outputs a number that influences the section of sound to iterate. The shorter it is, the faster the change, and

vice versa.

6. Play with the *amp env* dial. This dial influences the length of the amplitude envelope. Once you like the length, you could further customize the changes in amplitude by editing the amplitude envelope.

7. Amplitude envelope, by default, is given an Attack-Sustain-Decay-Release (ADSR) envelope. X-coordinates represent time, starting when a note is pressed, while Y-coordinates represent amplitude over that period of time. The orange dot is the Sustain dot. Make sure to only have one sustain. You may add white dots (representing other non-sustain ones), as many as you wish if you'd like a more complicated amplitude modulation. You could remove the extra white dots by focusing the cursor to one, holding down *SHIFT*, and then pressing on the dot. You could also change the steepness of slopes by holding down *ALT* and clicking and dragging up or down on the curve of interest, depending on how steep you want the curve to be. See Figures 7a and 7b on page 8.

8. Play around with other dials like the Freq. Shift (Frequency Shifter) and the Low Pass and High Pass Frequency knobs to further change the sound to your liking.

9. Press down *Record* to record a signal that is being granulated. The recording can be up to about 3 seconds.

10. Adjust the *Rev. Gain* (Reverb Gain) accordingly and adore the lush sounds that come out of *Granuvolver*. Careful with the gain as gain spikes from the convolution might occur, depending on frequencies present in the convolution–some frequencies might get too resonant and "blow up," causing some undesirable clipping.

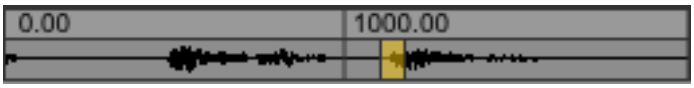


Figure 6: Selected stochastic loop area

(a) Adding new points

(b) Changing slope of curves

Figure 7: Editing amplitude envelope

5.5 Future Improvements

5.5.1 Known Bugs

5.5.2 Suggested Improvements

6 Music Piece

7 Conclusion

8 References

References

[1] C. Roads, The History of Microsound, in *Microsound*, 1st ed. Massachusetts, MIT Press, 2004, ch. 2, loc. 1024/4377 (Kindle), ISBN 0262182157

[2] C. Roads, Granular Synthesis, in *Microsound*, 1st ed. Massachusetts, MIT Press, 2004, ch. 2, loc. 272/4377 (Kindle), ISBN 0262182157

[3] Smith, J.O., Chapter 7: Fourier Theorems for the DFT. The Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications. W3K Publishing, pp. 121. ISBN 9780974560748

[4] Battenberg, Eric. et. al. Advances in the Parallelization of Music and Audio Applications. The Center for New Music and Audio Technologies and The Parallel Computing Laboratory, University of California Berkeley (2010)

[5] Harker, Alexander and Tremblay, Pierre Alexandre (2012) The HISSTools Impulse Response Toolbox: Convolution for the Masses. In: ICMC 2012: Non-cochlear Sound. The International Computer Music Association, pp. 148-155. ISBN 9780984527410