Granuvolver

Edderic Ugaddan

December 5, 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.

3 Granular Synthesis

4 Convolution

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Granuvolver was originally made using the cross-platform Virtual Studio Effect (VST) SDK using C++.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 is feeling more hardcore–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.

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.

The advantage of this was to be able to create MIDI instruments

6 Music Piece

Well, and here begins my lovely article.

7 Conclusion

8 References

. . . and here it ends.