SHORT NOTE

Introduction to sound programming in SuperCollider

Naofumi Aoki^{1,*} and Yota Morimoto^{2,†}

¹Graduate School of Information Science and Technology,
Hokkaido University,

N14 W9, Kita-ku, Sapporo, 060-0814 Japan

²Music Department, University of Birmingham,
Edgbaston, Birmingham, B15 2TT, UK

(Received 7 January 2013, Accepted for publication 3 April 2013)

Keywords: SuperCollider, Sound programming, Analog synthesizer, Acoustic education

PACS number: 43.10.Sv, 43.75.Wx
[doi:10.1250/ast.34.342]

1. Introduction

SuperCollider is a programming environment for realtime sound synthesis, initially developed by J. McCartney. It has been widely adopted for the creation of computer music [1]. In this paper, we will briefly demonstrate the possibility of sound programming in SuperCollider.

2. Architecture of SuperCollider

SuperCollider makes use of a client/server model. The client slang is an interpreter-style object-oriented programming language. The server scsynth executes sound synthesis. The communication between them is performed via OSC (Open Sound Control) protocol, which is developed as an alternative to MIDI (Musical Instrument Digital Interface) for faster exchange of musical information over TCP/IP network.

To start the sound synthesis server, one executes the following code;

```
s.boot;
```

Note that 's' is one of the reserved global variables. It stands for the sound synthesis server as default. To stop the server, one executes the following code;

```
s.quit;
```

3. Sine wave generator and additive synthesis

In SuperCollider, a sine wave can be generated in the following manner;

```
{ SinOsc.ar(440, 0, 0.2) }.play;
```

'SinOsc' is an UGen (Unit Generator) object. Its 'ar' method invokes an audio-rate generation of waveform. Within the parenthesis, frequency (in Hz), phase offset (in radians), and amplitude are provided as arguments. By using the curly bracket, one can define arbitrary waveform as shown in this example. Its 'play' method instantiates sound synthesis in the server.

The following code demonstrates an approximation of a sawtooth wave generator by adding multiple sine waves;

```
{
  var sine = 0;
  for(1, 10, {| i |
    sine = sine + SinOsc.ar(i * 440, 0, 0.2 / i);
  });
  Out.ar(0, sine);
}.play;
```

Within the for-loop, sine waves up until the 10th overtone are added. 'Out' UGen outputs the result of the additive synthesis.

4. Modeling a simple analog synthesizer

In general, an analog synthesizer consists of 3 modules called VCO, VCF, and VCA (Voltage Controlled Oscillator, Filter, and Amplifier, respectively). The sound synthesis of such an analog synthesizer is the result of signal flow among these modules as shown in Fig. 1.

The timbre of an analog synthesizer depends on the envelope of these modules. As shown in Fig. 2, an analog synthesizer generally has 4 parameters called ADSR (attack time, decay time, sustain level, and release time) for controlling the envelope.

SuperCollider can emulate such an analog synthesizer with short code. An example would be;

```
(
  SynthDef("analog synthesizer",
    {| gate=1, freq, amp, sustain |
      var sig, env;
      sig = LFSaw.ar(freq);
      sig = LPF.ar(sig, 4000);
      env = EnvGen.kr(Env.adsr(0.01, 0.3, 0.5, 1.0),
        gate, 1, sustain, 2);
      sig = sig * env * amp;
      Out.ar(0, sig);
    }).add;
)
```

The 'SynthDef' object takes 2 arguments. The former is the name of the synthesizer. The latter is its definition. In this example, a sawtooth wave generated by 'LFSaw' is low-pass filtered by 'LPF,' and enveloped by 'EnvGen.' The cut-off frequency of the low-pass filter is 4,000 Hz. 'Env' defines the envelope by using 'adsr' method that employs the ADSR parameters as arguments.

Such a synthesizer can be sequenced by using 'Pbind' object;

```
(
  Pbind(
    \instrument, "analog synthesizer",
    \degree, Pseq([0, 1, 2, 3, 4, 5, 6, 7], 1),
    \dur, 0.5,
  ).play;
)
```

This example uses the '\instrument' named "analog synthesizer" defined in 'SynthDef' object in advance. The '\degree' sets the sequence to be 8 ascending notes of the C

*e-mail: aoki@ime.ist.hokudai.ac.jp

†e-mail: yotamorimoto@gmail.com

Fig. 1 Signal flow of analog synthesizer.

Fig. 2 ADSR parameters for controlling envelope.

major scale. The ‘\dur’ sets the duration of each note to be 0.5 second.

Unlike other general programming environments, SuperCollider can create musical sound with short codes as shown in this example. This feature is preferable for some practical research topics such as sonification, which requires a programming environment for musical sequencing as well as sound synthesis.

5. Conclusions

As described in this paper, sound synthesis and musical sequencing are programmed seamlessly in SuperCollider. This functionality of SuperCollider makes it favored by computer music composers who wish to design their own compositional tools.

In Europe, conferences on SuperCollider are periodically held, where many artistic works and new sound synthesis techniques are presented [2,3]. These conferences cultivate the activities of sound programming by artists as well as researchers. Such popularity is also backed-up in code-sharing websites [4], where sound synthesis techniques are presented as SuperCollider codes. This could potentially be a useful resource for acoustic education.

However, SuperCollider in Japan is not as popular as in Europe yet. Although the official reference of SuperCollider has been published in English [5], it seems that the key to the popularization of SuperCollider in Japan would be to increase the availability of resources in Japanese. Such materials should further contribute to cultivate the activities of sound programming in Japan.

References

- [1] N. Aoki and Y. Morimoto, “Sound programming in SuperCollider,” *IEICE Tech. Rep.*, EA2011-36 (2011).
- [2] Y. Morimoto, “Hacking cellular automata: An approach to sound synthesis,” *Proc. SuperCollider Symp.*, Berlin (2010).
- [3] Y. Morimoto, “nl4sc: UGen implementations of nonlinear dynamical systems,” *Proc. SuperCollider Symp.*, London (2012).
- [4] <http://sccode.org>
- [5] S. Wilson, D. Cottle and N. Collins, Eds., *The SuperCollider Book* (The MIT Press, Cambridge, 2011).