連載

SUPERCOLLIDER チュートリアル (1)
SUPERCOLLIDER TUTORIALS (1)

美山 千香士
Chikashi Miyama
ケルン音楽舞踏大学
Hochschule für Musik und Tanz Köln

概要

本連載では、リアルタイム音響合成環境の SuperCollider(SC) の使い方を、同ソフトを作品創作や研究のために利用しようと考えている音楽家、メディア・アーティストを対象にチュートリアル形式で紹介する。

SuperCollider(SC) is a realtime programming environment for audio synthesis. This article introduces SC to musicians and media artists who are planning to utilize the software for their artistic creations and researches.

1. SUPERCOLLIDER とは

1.1. 概要

SuperCollider(SC) は James McCartney によって開発が始められた、リアルタイム音響合成とアルゴリズム作曲のためのプログラミング環境である。当初は有償ソフトウェアとして販売されていたが、2002 年より GNU General Public License(GPL) に基づくフリーソフトとなり、現在は有志により開発が進められている。

1.2. 特徴

1.2.1. テキストベースのオブジェクト指向言語

SC では、Max や Pd 等の GUI を用いた音響合成プログラミング環境と異なり、**SC Language** という専用の言語を用いてテキストベースでプログラミングを行う。SC Language は Smalltalk を基調とするインタプリタ型オブジェクト指向プログラミング (OOP) 言語であるため、OOP 言語に精通したユーザーはカプセル化、継承、多態性などの OOP のプログラミング技法を利用して SC での音響合成プログラムを行う事ができる。

1.2.2. マルチ・プラットフォーム

SC は Mac、Windows、Linux、FreeBSD で動作するマルチ・プラットフォームなソフトウェアであるため、様々な環境で柔軟に運用する事ができる。特に、最近注目が高まっている、Raspberry Pi[1](図 1]) など安価なシングルボード・コンピュータ上でも SC は稼働するため、音を自動生成するプログラムを基調としたサウンド・インスタレーションや組み込み分野での活用も期待されている。

図 1. Raspberry Pi

1.2.3. クライアント・サーバーモデル

SC は実際に音を生成する **scsynth** と、音生成のための命令の送信やイベント処理等を行う **sclang** という自律的な 2 つのソフトウェアから構成されており、この 2 つのソフトウェアはクライアント・サーバーモデルを形成している (図 2)。

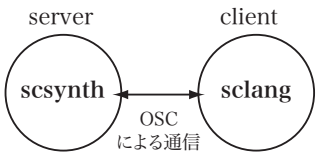

図 2. クライアント・サーバーモデル

scsynth と sclang は CNMAT の開発した **Open Sound Control(OSC)**[2] というプロトコル用いて通信を行うため、複数の sclang から scsynth をコントロールするこ

と、また、sclang の代わりに OSC を送信できるソフトウェアをクライアントとして用いる事も可能で、例えば Max や Pd から scsynth に OSC メッセージを送り、音の生成を促す事も可能である (図 3)。サーバーとクライアントは両方同じマシンでも、ネットワークで繋がった別々のマシンでも稼働させる事ができる。

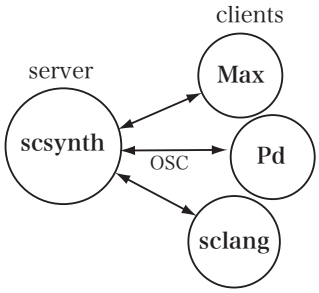

図 3. scsynth は様々なクライアントから制御可能

1.3. 動作確認環境

当記事掲載のソースコードは Mac 版の SC3.6.5 を使って動作確認を行っている。また、操作は基本的に Mac 版の SC についてのみ解説する。Win 版、Linux 版の SC を使っている場合は、若干操作が異なる可能性があるので留意されたい。

1.4. 情報元

本連載以外に自主的に SC を学ぼうというユーザーのために、以下、SC に関する情報源を紹介する。SC の日本語の情報は現在のところまだ潤沢とは言い難いが、英語のものは充実してきている。

1.4.1. ヘルプ・ファイル

SC のヘルプ・ファイルはかなり綿密に情報が記載されており、これをあたる事で大体の疑問が解決される。各ヘルプに実行可能なコードが複数紹介されているので、音を出しながら仕組みを理解していく事ができる。また、ヘルプ・ブラウザ（後述）のサブメニュー、「Browse」タグをクリックすると、テーマごとにドキュメントがリストされるので、ここから「Tutorials」を選ぶと「Getting-Started」や「Mark_Polishook_tutorial」といったチュートリアルを見つける事ができる。これらをひと通り勉強することで SC の基本的なしくみを理解することが可能である。

1.4.2. The SuperCollider Book

MIT Press から出版されている The SuperCollider Book[3] は SC のバイブル的書籍であるが、この本は多くの著者が SC に関する様々なテーマについて執筆したものをまとめたもので、初心者がステップ・バイ・ステップで SC を習得するのに適しているとは言い難い。むしろ中級者以上がさらなる SC の活用法を学ぶのに適した本といえる。

1.4.3. SuperCollider Japan

日本の SuperCollider のポータルサイト [4]。tn8 氏と umbrella_process 氏によって運営されている。SC に関する最新のニュースが日本語で読める他、日本語の wiki やフォーラムもあり、上級者に SC に関する質問をする事もできる。また、SC のワークショップの情報なども掲載されている。

2. ダウンロードとインストール

2.1. ダウンロード

SC は現在 sourceforge.net にホストされている。
`http://supercollider.sourceforge.net`
にアクセスし、「Download SuperCollider for Mac, Linux, or Windows」のリンクをクリックし、自分の環境にあったものをダウンロードする。

2.2. インストール

Mac の場合は、ダウンロードが完了すると「ダウンロード」フォルダの中に「SuperCollider-3.6.5-OSX-universal.dmg」というディスクイメージができるので、これをダブルクリックし展開する。すると、図 4 のようなウインドウが表示されるので、単純に SuperCollider のフォルダをアプリケーションフォルダにドラッグする。これでインストールは完了である。

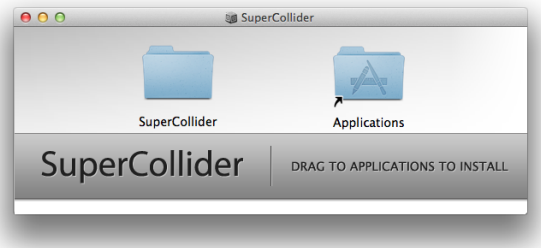

図 4. アプリケーションフォルダにドラッグ

3. 音を出すには

3.1. アプリケーションの起動

アプリケーションフォルダ内の SuperCollider フォルダの中に「SuperCollider」というアプリケーションがあるので、そのアイコンをダブルクリックするとアプリケーションが立ち上がり、図 5 のような画面が表示される。

図 5. SC のウインドウ

SC のウインドウはデフォルトで 4 つの部分から構成されており、左部分が実際にコードを書いていく、**コード・エディタ**、右上がヘルプを表示する**ヘルプ・ブラウザ**、右下がエラーやメッセージを表示する**ポスト・ウインドウ**、そして最下部が現在の SC の状態を表示する**ステータス・バー**と呼ばれる。

3.2. 正弦波を鳴らす

それでは、手始めに 440 Hz の正弦波を鳴らすコードをプログラムし、実行してみる。

3.2.1. コーディング

まず、以下のコードをウインドウ左部のコード・エディタにそのまま書き写す。

ソースコード 1. 正弦波

```
1 {
2     SinOsc.ar;
3 }.play
```

3.2.2. サーバーの立ち上げ

コードを書き終わったら、サーバー (scsynth) を立ち上げる。サーバーを起動するには、メニューの「Language」から「Boot Server」を選ぶ (ショートカットは [Cmd+B])。サーバーが立ち上がる前は図 6 上のように、ステータス・バーの「Server:」の項が白だが、立ち上がると図 6 下のように緑色になる。

図 6. SC サーバーのステータス

3.2.3. コードの実行

サーバーが起動したらいよいよコードを実行する。[Cmd+A] でコードの全体を選択し、[Cmd+Return] を押すと、440 Hz の正弦波がコンピュータのスピーカーから聞こえるはずである。もし、聞こえてこない場合は以下の点をチェックする。

- コンピュータのスピーカーのボリュームは上がっているか、ミュートされてないか
- オーディオ・インターフェースなど音に関連した外部装置が繋がっていないか
- サーバーが正常に稼働しており、ステータスが緑色に表示されているか

3.2.4. 音を止める

上記のコードで生成される正弦波は自動的には止まらない。これを強制的に停止させるには「Language」メニュから「Stop」を選ぶか、そのショートカットの [Cmd+.] を入力する。**SinOsc** はその名の通り、正弦波のオシレータであり、デフォルトで 440 Hz の正弦波を生成する。このように SC には音を生成する SinOsc のような「部品」が沢山用意されており、基本的にこれらの「部品」を組み合わせて音作りを行う。SC ではこの「部品」の事を **Unit Generator**、略して **UGen** という。SC において、SinOsc のように音を生成するだけでなく、分析したり、加工したりする役割を持つ「部品」は全て UGen である。UGen を実際に使うには、UGen の名前の後に**.ar** か**.kr** を付加する必要がある。.ar を付加した場合、**オーディオ・レートで**、.kr で作った場合**コントロール・レート**でその UGen は動作する。コントロール・レートで動作する UGen からの出力は音として聞くことは出来ないが、その出力を利用してオーディオ・レートで動作する他の UGen のパラメータをコントロールする事が可能である。また、コントロール・レートの UGen は信号の出力頻度がオーディオ・レートの UGen に比べ低いため、CPU 負荷が軽い。

3.3. 周波数、振幅の操作

SinOsc.ar はデフォルトで 440 Hz の正弦波を生成するが、この周波数を変更するにはソースコード 2 のように SinOsc.ar の後にカッコを付け SinOsc.ar(880) のように書く。

ソースコード 2. 周波数の変更

```
1 {
2     SinOsc.ar(880);
3 }.play
```

これで先ほどの音より 1 オクターブ高い 880 Hz の音が生成されるようになる。この「880」という数値のように括弧の中に書いて出力を変化させるものを**引数 (arguments)** という。引数の値がその UGen にどのように作用するのかは、ヘルプ・ファイルを見て確認する必要がある。

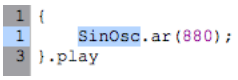

図 7. ヘルプ・ファイルを見る

ヘルプ・ファイルを開く方法は図 7 のように、まずヘルプ・ファイルを見たい UGen を選択し、「Help」メニューから「Look Up Documentation for Cursor」を選択するか、そのショートカットの [Cmd+D] を入力する。するとヘルプ・ブラウザにその UGen に関するヘルプ・ファイルが表示される。

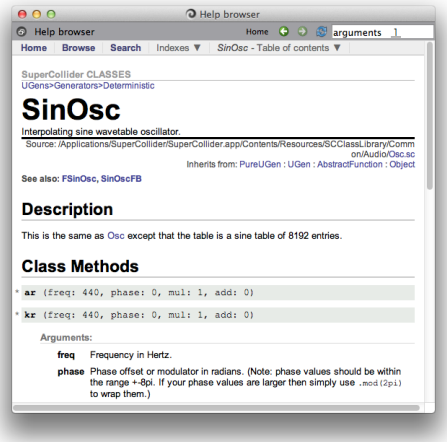

図 8. ヘルプ・ファイルで引数をチェックする

ヘルプ・ファイルの Class Methods の項目に図 8 のように.ar や.kr の後にどのような引数を書くことができるのかが説明されている。このヘルプ・ファイルによると、SinOsc は最大 4 つまでの引数を取ることができ、それぞれオシレータのパラメータ、*freq*、*phase*、*mul*、*add* を設定しているのが分かる。各引数には初期値が設定されており、引数を書かなかった場合はその初期値が用いられる。ソースコード 1 では、全ての引数が書かれていないので、*freq* の初期値である 440 が用いられた。そのため、440 Hz、つまり「ピアノの真ん中のラ」(A4) の音が生成された。また、この正弦波の音を小さくするには、1 より少ない数を SinOsc.ar の第 3 引数、*mul* に指定し、乗算を行えば良い。例えば 0.1 を指定すると正弦版の出力に 0.1 が乗算され、正弦波の振幅が 1/10 になり、音の聞こえも静かなものになる。

ソースコード 3. 振幅を小さくする

```
1 {
2     SinOsc.ar(880, 0, 0.1);
3 }.play
```

SinOsc

正弦波を生成するオシレータ。

.ar(*freq*, *phase*, *mul*, *add*)
.kr(*freq*, *phase*, *mul*, *add*)

freq · · · 周波数を Hz で指定
phase · · · 位相をラジアンで指定
mul · · · 出力値にこの値を掛ける
add · · · 出力値にこの値を足す

3.4. 波形の可視化

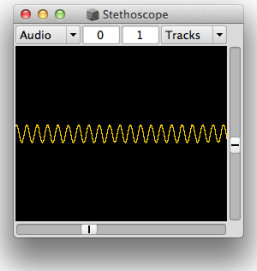

図 9. 波形を可視化する

今までのコードではコードの最後に.play が書かれていたが、この代わりに.scope を使うと、図 9 のように音の生成と同時に波形を表示させる事が可能である。

ソースコード 4. 波形の可視化

```
1 {
2     SinOsc.ar(880, 0, 0.1);
3 }.scope
```

3.5. グリッサンドを作る

これまでのプログラムは周波数や振幅が定数によって固定されていたが、**Line** を用いると、周波数などのパラメータを一定時間をかけて徐々に変化させることが可能となる。例えばソースコード 5 を実行すると、440 Hz から 880 Hz まで 10 秒かけて周波数が上昇するので、グリッサンドを作る事ができる。

ソースコード 5. グリッサンド

```
{
    SinOsc.ar(Line.kr(440, 880, 10), 0, 0.1);
}.scope
```

ここで用いられている Line は.ar ではなく.kr である、このようにパラメータを操作するための UGen にはコントロール・レートの.kr を使う事が可能である (.ar の使用も可)。

Line

ライン・ジェネレータ。初期値から目的値に所要時間をかけて徐々に値を変化させる。

.ar(*start*, *end*, *dur*, *mul*, *add*, *doneAction*)
.kr(*start*, *end*, *dur*, *mul*, *add*, *doneAction*)

start ··· 初期値
end ··· 目的値
dur ··· 所要時間
mul ··· 出力値にこの値を掛ける
add ··· 出力値にこの値を足す
doneAction ··· 所要時間が経過した時に行う処理を指定

3.6. クレッシェンドを作る

Line の出力を周波数に適用すれば、グリッサンドが作れるが、これを振幅に適応すればソースコード 6 のようにクレッシェンドを作る事も可能である。

ソースコード 6. クレッシェンド

```
{
    SinOsc.ar(880, 0, Line.kr(0, 1, 10));
}.scope
```

3.7. マウスでパラメータをコントロールしてみる

さらに SinOsc の引数はマウスでリアルタイムにコントロールする事も可能である。

ソースコード 7. マウスによるコントロール

```
{
    SinOsc.ar(MouseX.kr(440, 880), 0, 0.1);
}.scope
```

ソースコード 7 で使われている **MouseX** は、マウスの現在の X 座標を引数で指定した範囲にスケーリングして出力するので、マウスカーソルを画面左端に配置すると 440 Hz の正弦波、右端に配置すると 880 Hz の正弦波が生成され、左右に動かすとそれにともなって正弦波の周波数が変化する。

MouseX, MouseY

マウスカーソルの X,Y 軸上の位置をスケーリングして出力。

.kr(*minval*, *maxval*, *warp*, *lag*)

minval ··· 最小値
maxval ··· 最大値
warp ··· 最小・最大値間補間方法。0 で線形、1 で指数補間
lag ··· ラグ要素、多いほど値の出力が滑らかになる

3.8. 他のオシレータを使ってみる

SC には予め、SinOsc の他にも様々な種類のオシレータが含まれている。ソースコード 8、9、10 はそれぞれ鋸歯状（ノコギリ）波、矩形波、三角波のオシレータによる音の生成を行なっている。

ソースコード 8. 鋸歯状波

```
{
    Saw.ar(880);
}.scope
```

ソースコード 9. 矩形波

```
{
    Pulse.ar(880);
}.scope
```

ソースコード 10. 三角波

```
{
    LFTri.ar(880);
}.scope
```

SC にはこの他にも様々なオシレータがある。ひと通りオシレータの音を聞いてみたい場合は、ヘルプ・ブラウザの上方にあるタブ、「search」を選び「Tour of UGens」というキーワードでファイルを検索すると「Tour of UGens」というファイルが見つかる。このファイルには、SC で使える主要なオシレータが羅列されているので各オシレータの音を順番に試す事ができる。

4. 複数の音を同時に出す

4.1. UGen の出力を足す

複数の音を同時に出すには単純に 2 つ以上の UGen の出力を足す。ソースコード 11 では 400 Hz と 500 Hz

の正弦波を同時に聞くことができる。

ソースコード 11. 複数の音を同時に鳴らす

```
{
    SinOsc.ar(400, 0, 0.1) + SinOsc.ar(500, 0, 0.1);
}.scope
```

4.2. 変数を使う

UGen の出力を直接足す他にも、出力を一度 a や b などの**変数**に代入し、後に足す事もできる。変数はアルファベット 1 文字の小文字の場合は事前に宣言する必要はない。ソースコード 12 の例では、400 Hz の正弦波を変数 a に、500 Hz にの正弦波を変数 b に代入し、4 行目で a と b の加算を行っているため、ソースコード 11 のプログラムと同じ結果が得られる。

ソースコード 12. 変数の利用

```
{
    a = SinOsc.ar(400, 0, 0.1);
    b = SinOsc.ar(500, 0, 0.1);
    a + b;
}.scope
```

SC は {} 内の最後の行に書かれている音を生成する。例えば、ソースコード 13 のようにもし最後の行に a+b の代わりに、b のみを書いた場合、その前に a と b を足していたとしても b、つまり 500Hz の正弦波しか聞こえない。

ソースコード 13. 最後の行に b を置く

```
{
    a = SinOsc.ar(400, 0, 0.1);
    b = SinOsc.ar(500, 0, 0.1);
    a + b;
    b;
}.scope
```

4.3. 長三和音を作る

現在まで音の高さは全て周波数によって指定してきたが、ある数値に対して.midicps を付加すると、SC はその数値を MIDI ノート・ナンバーとして解釈し、周波数に変換する。故に、MIDI ノート・ナンバーを用いて 12 音平均律でピッチを指定する事も可能である。例えば、ソースコード 14 を実行すると 220 Hz の正弦波が聞こえてくるが、これは、57 が MIDI ノートナンバーで A3(ピアノの真ん中のラより 1 オクターブ低いラ) であり、それが.midicps で周波数値=220 に変換されているためである。

ソースコード 14. .midicps

```
{
    SinOsc.ar(57.midicps, 0, 0.1);
}.scope
```

これを利用し、C4 の 60、E4 の 64、G4 の 67 にそれぞれ.midicps を送り、周波数値に変換し、3 つのオシレータに引数として渡し、出力される信号を全て加算すると、長三和音を作る事ができる。

ソースコード 15. 長三和音

```
{
    a = SinOsc.ar(60.midicps, 0, 0.1);
    b = SinOsc.ar(64.midicps, 0, 0.1);
    c = SinOsc.ar(67.midicps, 0, 0.1);
    a + b + c;
}.scope
```

5. 参考文献

[1] *Raspberry Pi* , http://www.raspberrypi.org(アクセス日 2013 年 6 月 3 日)

[2] Wright, M. and Freed, A. “Open Sound Control: A New Protocol for Communicating with Sound Synthesizer” *ICMC 1997* プロシーディングス pp.101-104, 1997

[3] Wilson, S., Cottle, D. and Collins, N. (eds), *The SuperCollider Book* The MIT Press, 2011

[4] *SuperCollider Japan* , http://supercollider.jp(アクセス日 2013 年 6 月 3 日)

6. 著者プロフィール

美山 千香士 (Chikashi MIYAMA)

作曲家、電子楽器創作家、映像作家、パフォーマー。国立音楽大学音楽デザイン学科より学士・修士を、スイス・バーゼル音楽アカデミーよりナッハ・ディプロムを、アメリカ・ニューヨーク州立バッファロー大学から博士号を取得。Prix Destellos 特別賞、ASCAP/SEAMUS 委嘱コンクール 2 位、ニューヨーク州立大学学府総長賞、国際コンピュータ音楽協会賞を受賞。2004 年より作品と論文が国際コンピュータ音楽会議に 12 回入選、現在までに世界 18 カ国で作品発表を行っている。2011 年、DAAD(ドイツ学術交流会) から研究奨学金を授与され、ドイツ・カールスルーエの ZKM で客員芸術家として創作活動に従事。近著に「Pure Data-チュートリアル＆リファレンス」(Works Corporation 社) がある。現在ドイツ・ケルン音楽舞踏大学電子音楽スタジオ非常勤講師。http://chikashi.net