連載

SUPER COLLIDER チュートリアル (4)
SUPER COLLIDER TUTORIALS (4)

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. 今回の目標：SC でエフェクタを作成する

前回までは、SC を用いて音を生成する方法を学習してきたが、今回はマイクからの入力音をリアルタイムに加工する方法に焦点をあて、リング・モジュレーション、ディストーション、ピッチ・シフター、ディレイ、リバーブなどのエフェクトを SC で作成する。ハウリングを避けるために、本稿のプログラムを実行する際には、ヘッドフォンなどを着用すること。また、OS 側の設定で必ずマイクがオーディオの入力機器に選択されていることを事前に確認すること。Mac の場合は、マイクを「環境設定」パネルで OS のオーディオ入力デバイスとして設定すると、SC サーバーの起動時にリスト 1 のように、マイク入力がリストの最初に列挙される。

リスト 1. 入出力デバイスのリスト

```
Number of Devices: 3
   0 : "Built-in Microph"
   1 : "Built-in Input"
   2 : "Built-in Output"
```

2. マイクからの入力音を得るには

マイクからの入力を SC で得るには、リスト 2 のように **SoundIn** という Ugen を用いる. このプログラムで単純にはマイクからの音声をオーディオ出力デバイスに送っている。マイクからの音がそのままヘッドフォンで聞くことができれば成功である。

リスト 2. マイク入力

```
{
   SoundIn.ar([0,1]);
}.play
```

SoundIn

コンピュータのマイクやサウンドカードからの音声入力を得る。*bus* に Array を与える事で複数のチャンネルの入力を取得することもできる。

.ar(*bus*, *mul*, *add*)

bus･･･ 入力チャンネル番号。Array を与えることで、複数のチャンネルを同時に取得可能。

3. リング・モジュレーション

最も単純なエフェクトとして、K・シュトックハウゼンの「マントラ」などで著名なリングモジュレーションを実装する。このエフェクトはリスト 3 のように、SoundIn からの入力音にサイン波を掛けあわせて実現する事ができる。

リスト 3. リング・モジュレーション

```
{
   SoundIn.ar([0,1]) * SinOsc.ar(MouseX.kr
       (20,880));
}.play
```

4. ディストーション

音信号を増幅し歪ませる事で、豊かな倍音を含んだ太い音を得る、ロック・ギターでお馴染みのディストーションは SC3 ではリスト 4 のように.distort メソッドを用いてプログラムする事ができる。

リスト 4. ディストーション

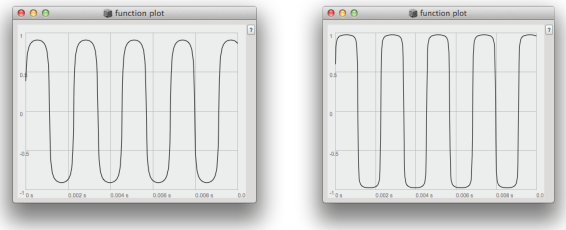


図 1. distortion(左) と softclip(右) により歪ませた正弦波

```
{
   SoundIn.ar([0,1], 100).distort /100;
}.play
```

ゲインを調整する事で歪みの深さを調整する事も可能である。リストではゲインを SoundIn の *mul* 引数を 100 に設定して 100 倍に増幅し、.distort により歪ませた後、「/ 100」で音量の調整をしている。この他にも.softclip を使うと.distort と違った歪み方をさせることが可能である。図 1 は 10 倍に増幅したサイン波をそれぞれ.softclip と.distort で歪ませた時の違いである。

5. ピッチ・シフター

入力音のピッチを自在に変化させるピッチ・シフターは **PitchShift** という Ugen を用いれば簡単にプログラム可能である。過剰に音高を変化させると原音の音色と全く異るものとなるので、原音の音色をなるべく保ちたい場合はあまり極端なパラメータを設定しない方がよい。リスト 5 では *pitchRatio* のパラメータを 0.5 に設定しているため、原音が半分の速度での再生される、このため原音より 1 オクターブ低くなる。0.5 のような固定値の代わりに **MouseX** などでリアルタイムに pitchRatio のパラメータを変化させる事なども可能である。

リスト 5. ピッチシフター

```
{
   PitchShift.ar(SoundIn.ar([0,1]),
       pitchRatio:0.5);
}.play
```

例えば、原音より短三度高い音と長三度低い音を付加してピッチシフターを利用して和音を作る事も可能である。このように音階上にピッチシフトを行う場合は、以前に学習した.miditocps を利用する。

リスト 6. ピッチシフターによる和音

```
{
   a = PitchShift.ar(SoundIn.ar([0,1]),
       pitchRatio: 3.midicps / 0.midicps);
   b = PitchShift.ar(SoundIn.ar([0,1]),
       pitchRatio: -4.midicps / 0.midicps);
   a+b
}.play
```

.midicps は、MIDI ノートナンバーから周波数を出力させるメソッドであり。0.midicps を実行すると、MIDI ノートナンバー「0」の周波数を得られる。MIDI ノートナンバー「0」より長三度高い「4」と短三度低い「-3」の周波数をそれぞれもとめ、それを「0」の周波数と比較することで、周波数比を得られる、これを *pitchRatio* の値として PitchShift.ar に与える事でピッチシフターを用いた長三和音を作る事ができる。

PitchShift

ピッチシフトを行う Ugen。音程は pitchRatio で指定する。dispersion のパラメータを用いることで、音程のランダマイズなども可能。.ar(*in*, *windowSize*, *pitchRatio*, *pitchDispersion*, *timeDispersion*, *mul*, *add*,)

in· · · 入力信号

windowSize· · · ウインドウ・サイズ

pitchRatio · · · ピッチ・レシオ

pitchDispersion · · · ピッチ分散度

timeDispersion · · · 時間分散度

6. ディレイ

ここでは入力音を一定時間遅延させ、やまびこのような効果を得るディレイとその応用を数種作成する。

6.1. シンプルなディレイ

遅延効果を SC で用いるには **DelayN** という Ugen を仕様する。この Ugen の第 2、第 3 引数はそれぞれ、最大ディレイ・タイムとディレイタイムで、最大ディレイ・タイムの値を元に、SC はディレイ用のバッファを用意し、それを利用してディレイ・タイムで指定された時間だけ入力音を遅延させて出力する。最大ディレイ・タイムを超えたディレイ・タイムを設定する事はできない 7。

リスト 7. ディレイ

```
{
   DelayN.ar(SoundIn.ar([0,1]), 0.5, 0.5)
}.play
```

DelayN

.ar(*in*, *maxdelaytime*, *delaytime*, *mul*, *add*)

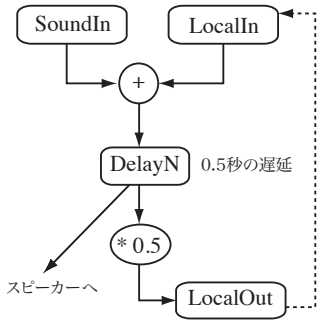
入力音を指定の時間遅延させる。*in*· · · 入力信号。

maxdelaytime· · · 最大ディレイ・タイム。

delaytime · · · ディレイ・タイム。*maxdelaytime* を超える時間を指定できない。

6.2. フィードバック・ディレイ

遅延させた信号を振幅を弱め、もう一度 Delay.ar に入力することによって、入力音が減衰しながらも何度も繰り返されるフィードバック・ディレイが実現できる。このようにプログラムの中でフィードバックを実現するには、リスト 8 のように **LocalIn** と **LocalOut** という Ugen を用いる。リストでは、DelayN による 0.5 秒の遅延処理と * 0.5 による振幅の減衰が施された信号をリストの 4 行目 LocalOut に送り、それが 2 行目で LocalIn によって取り出され、原音と加算されている。LocalOut にオーディオ信号の Array を渡す事で、複数のチャンネルを LocalOut に送る事も可能。複数のチャンネルを LocalOut に送った場合は、そのチャンネル数を LocalIn の第 1 引数として指定する。



リスト 8. フィードバック・ディレイ

```
1 {
2   a = SoundIn.ar([0,1]) + LocalIn.ar(2);
3   d = DelayN.ar(a, 0.5, 0.5);
4   LocalOut.ar(d*0.5);
5   d
6 }.play
```

6.3. ピンポン・ディレイ

フィードバック・ディレイにアレンジを加え、リスト **??**のように各ディレイが左右のスピーカーから交互に聞こえるようにすることもできる。

リスト 9. ピンポン・ディレイ

```
1  {
2    i = SoundIn.ar(0) + SoundIn.ar(1);
3    d = DelayN.ar(i, 0.25, 0.25);
4    l = LocalIn.ar(2);
5    i = l[0] + i;
6    d = l[1] + d;
7    l = DelayN.ar(i, 0.5, 0.5) * 0.5;
8    r = DelayN.ar(d, 0.5, 0.5) * 0.5;
9    LocalOut.ar([l,r_]);
10   [l,r+d]
11 }.play
```

リストでは 2 チャンネルの LocalIn/Out を使っている。4 行目で、LocalIn の引数の 2 を与え、オーディオ信号の Array を取り出し、それぞれマイクからの入力音とマイク入力音を 0.25 秒遅延させたものに足している (5,6 行目)。そして、それぞれに DelayN と*0.5 で個別に遅延と減衰処理を施し LocalOut とスピーカーに送っている。

6.4. マルチタップ・ディレイ

リスト 10 のように DelayN と Array と組み合わせて、リズミカルなディレイを作る事も可能である。尚、このエフェクトはその性質上入力ソースがモノラルである必要があるため、2 行目でステレオ入力信号を加算しモノラル化している。

リスト 10. マルチタップ・ディレイ

```
1 {
2   a = [1, 2, 4, 6, 7] * 0.1;
3   Mix.ar(DelayN.ar(SoundIn.ar([0,1]), a, a
        ));
4 }.play
```

このコードで変数 a はマルチタップ・ディレイのリズムパターンが定義された Array が格納されている。Array は [1, 2, 4, 6, 7] というリズムパターンに 0.1 が掛けられているので、その内容は [0.1, 0.2, 0.4, 0.6, 0.7] となる、この Array を DelayN の最大ディレイ・タイムとディレイ・タイムにそれぞれ適応している。このように引数に Array が与えられた場合、SC は自動的に DelayN を複製する。ここでは 5 つの要素からなる Array が与えられたため DelayN が 5 つ作られ、それぞれに異なった最大ディレイ・タイム及びディレイ・タイムが指定される。この全ての DelayN からの出力を **Mix** という Ugen を使って全て加算している。2 行目の定数 0.1 を変更する事で、リズムを保持したまま、テンポだけを変える事が可能である。

Mix

array 内の音声信号を全て加算する.ar(*array*)

array··· 音声信号の Array、全てのチャンネルが加算される

7. リバーブ

残響効果を入力音に加えるには、以下のように FreeVerb という Ugen を利用する。

リスト 11. リバーブ

```
1 {
2   FreeVerb.ar(SoundIn.ar(0), 1.0);
3 }.play
```

Reverb

.ar(*in*, *mix*, *room*, *damp*, *mul*, *add*)

in··· 入力信号
mix··· 原音と加工音のバランス、1,0 で加工音のみ、0.0 で原音のみ
room ··· ルームサイズ、1.0 が最大、0.0 が最小。
damp ··· 高周波数のダンプ。0 から 1 の範囲で指定

8. フィルタ

SC には様々なフィルタが予め用意されている。リスト 12 では、入力音にハイパス・フィルタを施し、3000Hz 以下の低周波成分を減衰させている。

リスト 12. ハイパス・フィルタ

```
{
  HPF.ar(SoundIn.ar(0), 3000);
}.play
```

HPF(High Pass Filter) の他にも LPF(Low Pass Filter)、BPF(Band Pass Filter)、Moog、TwoPole など様々なフィルタが用意されている。「Filter」をヘルプで調べると、全てのフィルタのリストを見ることができる。

HPF

2 次バターワース・ハイパスフィルタ.ar(*in*, *freq*, *mul*, *add*)

in··· 入力信号
freq··· カットオフ周波数

9. ワウワウ

リスト 13 では、バンド・パスフィルタ中央周波数のパラメータを LFO でコントロールすることにより、「ワウワウ」のエフェクトを作成している。

リスト 13. ワウワウ

```
{
  l = SinOsc.ar(5, 0, 1000, 1500);
  BPF.ar(SoundIn.ar(0), l, 0.5);
}.play。
```

BPF の三番目の引数は rq(reciprocal of Q) 値であり、この値が少なければ少ないほどフィルタを通過する帯域幅が狭くなり、強くワウワウがかかる。

BPF

2 次バターワース・バンドパスフィルタ.ar(*in*, *freq*, *rq*, *mul*, *add*)

in··· 入力信号
freq··· 中央周波数
rq··· reciprocal of Q 値

10. エフェクタを組み合わせる

SC3 ではこれまでに制作してきたエフェクターを組み合わせる事も可能である。リスト 14 ではディストーション・ディレイ・リバーブを連結して、より複雑なエフェクトを実装し、それを「myEffect」という SynthDef として定義している。

リスト 14. エフェクトの組み合わせ

```
SynthDef("myEffect",{
  i = SoundIn.ar(0);
  d = (i * 100).distort / 50;
  d = DelayN.ar(d, 0.5, 0.5);
  d = d + FreeVerb.ar(d, 1.0) ;
  Out.ar(0, d);
}).load;

Synth("myEffect")
```

11. まとめ

今回は SC による様々なエフェクトのプログラミングを学習した。エフェクタは、さらに SC のバス (Bus) と言われる機能を利用して、エフェクタの Synth を複数繋ぐ、音を生成する Synth とエフェクタを定義した Synth を繋いで、自作シンセ音に各種エフェクトを適応するという事も可能である。このバス機能については、また回を改めて紹介する。

12. 参考文献

[1] *SuperCollider*, http://supercollider.sourceforge.net(アクセス日 2014 年 6 月 10 日)

13. 著者プロフィール

e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 19 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・ケルン音楽大学講師。スイス・チューリッヒ芸術大学コンピュータ音楽・音響研究所 (ICST) 研究員。バーゼル造形大学のプロジェクト「Experimental Data Aesthetics」プログラマ。http://chikashi.net