Creating Coding School Sapporo 2017

札幌国際芸術祭編

Sonic Piでライブコーディング

金井 謙一

ARTSAT x SIAF LAB.

Sonic Pi

Sonic Pi

- http://sonic-pi.net/
- "The Live Coding Music Synth for Everyone"
- Sam Aaronが開発
- Rubyベース

- コードで音楽を表現する
- ライブコーディング
- プログラミング教育

コードで音楽を表現する

- スティーヴ・ライヒ - ピアノ・フェイズ（Piano Phase）
  - Sam Aaronのコードから
  - https://gist.github.com/samaaron/997ba2902af1cf81a26f
  - wikipedia参照

```
notes = (ring :E4, :Fs4, :B4, :Cs5, :D5, :Fs4, :E4, :Cs5, :B4, :Fs4,

use_synth :piano
live_loop :slow do
  play notes.tick, pan: -1
  sleep 0.151
end

live_loop :faster do
  play notes.tick, pan: 1
  sleep 0.15
end
```

コードで音楽を表現する

- Sonic PiコミッターのXavier Riley氏(heroku)の作ったコード
  - https://gist.github.com/xavriley/87ef7548039d1ee301bb
  - Sonic Piで最も有名なコード(個人的見解)
  - 2016年のRubyKaigiでも、Julian Cheal氏が紹介
  - http://rubykaigi.org/2016/presentations/juliancheal.html

ライブコーディング

- 「コンピュータの言語であるプログラムコードを直接操作することで、さまざまな音や映像をリアルタイムに生成する即興演奏の方法」
- TopLap https://toplap.org/
  - ライブコーディング・マニフェスト
  - https://toplap.org/wiki/ManifestoDraft
- Algorave https://algorave.com/
  - ライブコーダーによるレイヴイベント

- Sonic Piもlive_loop他の仕組みでライブコーディングが可能に
- Sam Aaronが自身のライブコーディングを配信している
  - https://www.youtube.com/user/samaaronuk/videos
  - 個人的に好みなのは、https://www.youtube.com/watch?v=G1m0aX9Lpts

プログラミング教育

- ラズベリーパイ(raspbian-jessie)に標準搭載
- Scratch, Processingに続く存在という認識(個人的見解)
- 教育用のリソースも公式ページにあったりする
- SIAFラボの子ども向けワークショップ(2015年)
  - http://siaf.jp/siaflab/2015/09/11/kodomo_ws/

Sonic Piを学ぶ

- 公式(付属)のチュートリアル
- 田所さんの講義資料
  - http://yoppa.org/teu_media17
  - http://yoppa.org/tau_sound16
- 公式(付属)の例

実際に使ってみよう

画面の簡単な説明


A
Play Controls
B
Editor Controls
C
Info & Help
D
Code Editor
E
Prefs Panel
F
Log Viewer
G
Help System
H
Scope Viewer
Coded by Sam Aaron
live_loop :haunted do
sample :perc_bell, rate: rrand(-1.5, 1.5)
sleep rrand(0.1, 2)
end
Trigger sample

ライブループ

- 以下のコードをBufferにコピー＆ペーストして、Runボタンを押してみてください。

```
live_loop :flibble do
  sample :bd_haus, rate: 1
  sleep 0.5
end
```

- 続いて、Bufferの内容を以下のように変更して、Runボタンを押してみてください。

```
live_loop :flibble do
 sample :ambi_choir, rate: 0.3  # この行を追加
 sample :bd_haus, rate: 1
 sleep 0.5
end
```

- ビートを止めずに演奏内容を変更できたと思います。
- 音を止めるには、Stopボタンを押してください。

- コードの解説

```
live_loop :flibble do      # live_loopブロックの宣言。:flibbleという名前を付与
 sample :bd_haus, rate: 1 # :bd_hausというサンプルを再生。
 sleep 0.5                 # スリープ。次の繰り返し開始までの拍数を指定。
end
```

- ライブループの仕組み自体について知りたい方は、以下を参照ください。
  - https://kn1kn1.github.io/2016/02/06/In-N-Out-Sonic-Pi-Pt.II.html

シンセ

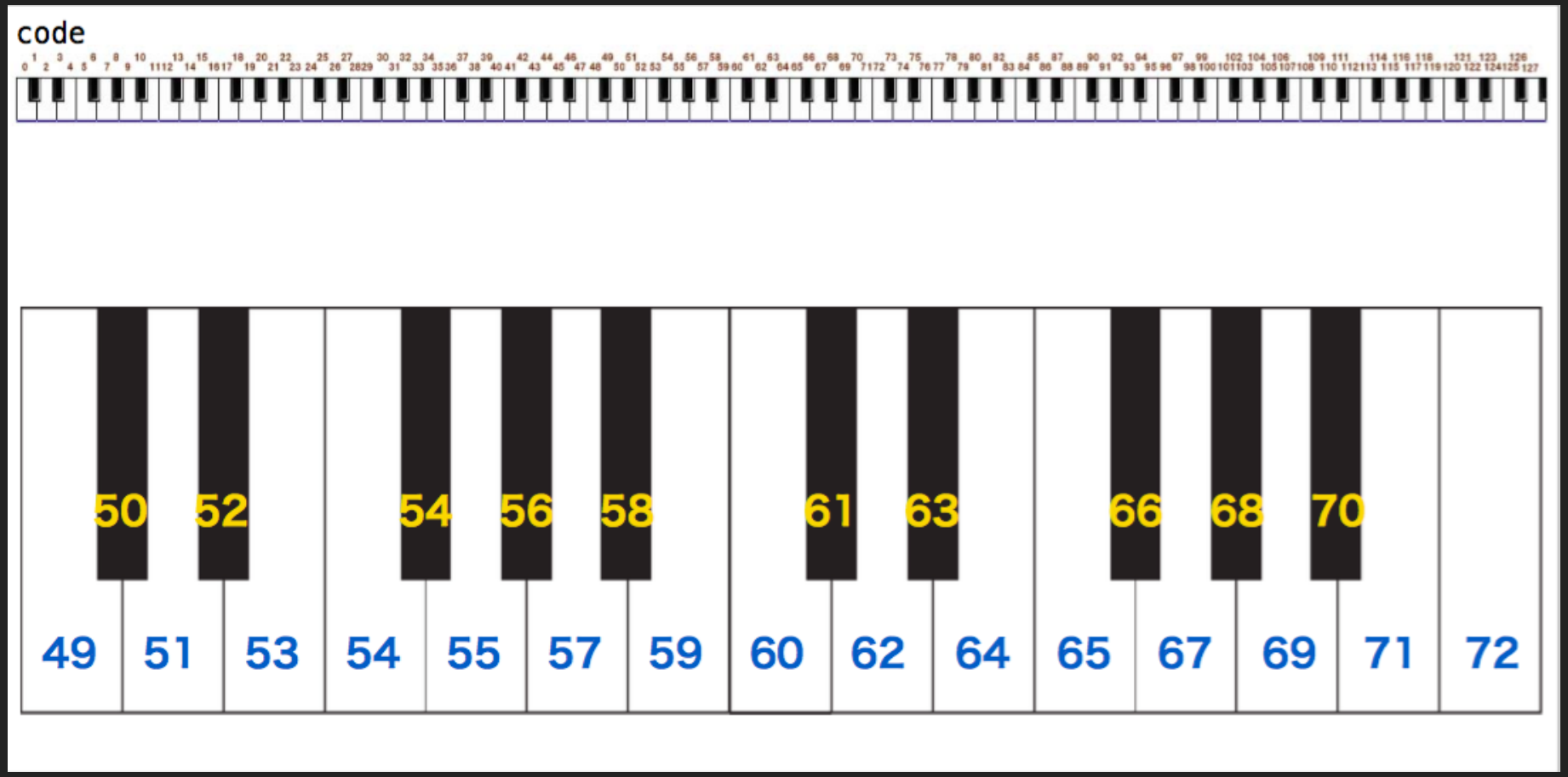
- 単純な音を出してみましょう。

```
live_loop :a do
 play 60
 sleep 1
end
```

- これは、サイン波で第4オクターブのCを鳴らしています。
- 以下のコードと同じです。

```
live_loop :a do
 use_synth :beep
 play :c4
 sleep 1
end
```

- playの後の60は、MIDIノート番号と呼ばれているものです。
  - 鍵盤との対応は以下を参照
  - (出典: http://yoppa.org/teu_media17/8322.html)



オプション（音量）

- playの後ろに、amp: 0.5を付けると音量を変えることができます。

```
live_loop :a do
  play 72, amp: 0.5
  sleep 1
end
```

- このkey: valueの組み合わせをオプションと呼んでいます。

オプション（パン）

- pan: -1で音の位置を変えることができます。

```
live_loop :a do
  play 72, amp: 0.5, pan: -1
  sleep 1
end
```

- 左から音が出たと思います。panで指定可能な値は、-1(左)〜0(中央)〜1(右)です。

和音

- 複数のplayを書くと、全て同時に演奏されます。

```
live_loop :a do
  play 72
  play 75
  play 79
  sleep 1
end
```

- chordで和音名で指定することも可能です。

```
live_loop :a do
  play chord(:E5, :minor)
  sleep 1
end
```

- 左下のヘルプの**命令**にchordがありますので、そちらを参照してください。

メロディー

- playの間にsleepを入れると、1音ずつ演奏されます。

```
live_loop :a do
  play 72
  sleep 1
  play 75
  sleep 1
  play 79
  sleep 1
end
```

- 同じことをplay_patternで書くことができます。

```
live_loop :a do
  play_pattern [72, 75, 79], sustain: 0
end
```

- play_patternにchordを渡すことも可能です。

```
live_loop :a do
  play_pattern chord(:E5, :m7)
end
```

スケール

- scaleで様々なスケール(音階)を取得できます。
- 以下は、C5メジャースケールです。

```
live_loop :a do
  play_pattern scale(:c5, :major)
end
```

- これを同じC5から始まる:hirajoshi(平調子)に変更してみます。

```
live_loop :a do
  play_pattern scale(:c5, :hirajoshi)
end
```

- 左下のヘルプの**命令**にscaleがありますので、そちらを参照してください。

シンセを切り替える

- ノコギリ波を使ってみます。

```
live_loop :a do
  use_synth :saw
  play 38
  sleep 0.25
  play 50
  sleep 0.25
  play 62
  sleep 0.25
end
```

- :prophetに変更してみます。

```
live_loop :a do
  use_synth :prophet
  play 38
  sleep 0.25
  play 50
  sleep 0.25
  play 62
  sleep 0.25
end
```

- 一音ごとに変更してみます。

```
live_loop :a do
  use_synth :tb303
  play 38
  sleep 0.25
  use_synth :dsaw
  play 50
  sleep 0.25
  use_synth :prophet
  play 57
  sleep 0.25
end
```

シンセを見つける

- 左下のヘルプに**シンセ**をクリックすると一覧が表示されますので、そちらを参照してください。
- 以下のようなものがあります。

```
:prophet
:dsaw
:fm
:tb303
:pulse
```

音の長さを変える

- 音の長さを変える簡単な方法は、releaseオプションを追加することです。

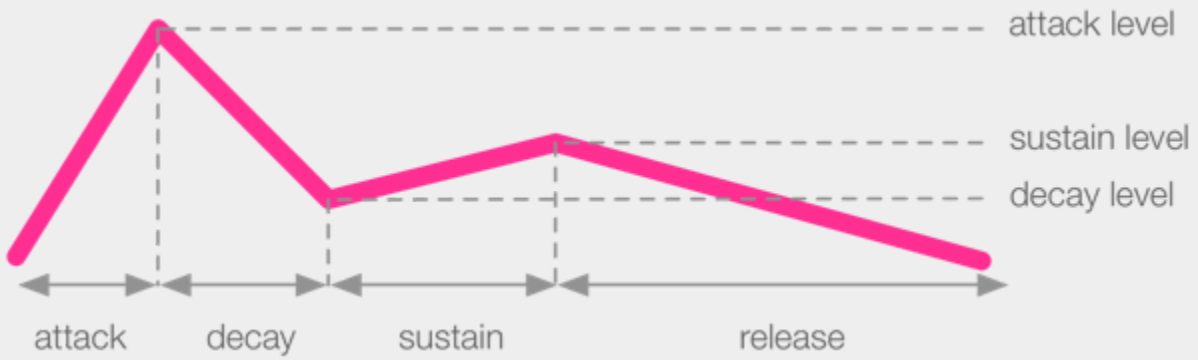
```
play 72
```

- 上のコードのplayの後ろに、release: 4を付けてみます。

```
play 72, release: 4
```

ADSRエンベロープ

- releaseはADSRエンベロープの一つです。



- ADSR
  - *attack*（アタック）- 音量を0からattack_levelまで移行させる時間
  - *decay*（ディケイ）- 音量をattack_levelからdecay_levelまで移行させる時間
  - *sustain*（サステイン）- 音量をdecay_levelからsustain_levelまで移行させる時間
  - *release*（リリース）- 音量をsustain_levelから0に移行させる時間

- Sonic Piでは、ADSRそれぞれの時間と、attack_level, decay_level, sustain_levelの音量をそれぞれ指定できます。

```
play 60, attack: 0.5, attack_level: 1, decay: 1, sustain_level: 0.4,
```

- attackを長くすることでアンビエント風になったりします。
- 例にあるDarin WilsonのAmbient Experiment

```
use_synth :hollow
with_fx :reverb, mix: 0.7 do

  live_loop :note1 do
    play choose([:D4,:E4]), attack: 6, release: 6
    sleep 8
  end

  live_loop :note2 do
    play choose([:Fs4,:G4]), attack: 4, release: 5
    sleep 10
  end

  live_loop :note3 do
    play choose([:A4, :Cs5]), attack: 5, release: 5
    sleep 11
```

サンプル

- サンプル音源を再生するには、sampleを使用します。

```
sample :loop_amen
```

- シンセと同じように、amp, panを指定可能です。

```
sample :loop_amen, amp: 0.5, pan, -1
```

- rateを指定することで、速く再生したり、遅く再生したりできます。

```
sample :loop_amen, rate: 2
```

```
sample :loop_amen, rate: 0.5
```

- マイナスの値を指定することで、逆再生も可能です。

```
sample :loop_amen, rate: -1
```

サンプルを見つける

- 左下のヘルプの**サンプル**をクリックし、カテゴリを選ぶと、使用できるサウンドのリストが表示されます。
- また、sampleまで入力すると、オートコンプリートによりサンプルの候補が出てくるのでこれを辿る方法もあります。

サンプルのオプション色々

- start, endで部分再生

```
sample :loop_amen, start: 0.4, finish: 0.6
```

- 逆再生で部分再生

```
sample :loop_amen, start: 0.6, finish: 0.4
```

- beat_stretchで指定した拍数に合わせる

```
live_loop :a do
  sample :loop_amen, beat_stretch:2
  sleep 2
end
```

- 逆再生も可能です。

```
live_loop :a do
  sample :loop_amen, beat_stretch:2, rate: -1
  sleep 2
end
```

ランダム

- 作品をより面白いものにするのに、演奏にランダム的な要素を追加してみましょう。

- rrandは引数で指定した最小値〜最大値の間の数値を返します。

```
live_loop :a do
  play rrand(50, 95)
  sleep 1
end
```

- rrandは少数を含む数値が返ります。
- rrand_iを使うと、整数のみを返すようになります。

```
live_loop :a do
  play rrand_i(50, 95)
  sleep 1
end
```

- chooseを使うと、リストから一つランダムに要素を返します。

```
live_loop :a do
  play [72, 75, 79].choose
  sleep 1
end
```

- chooseに渡すリストの生成に、chordやscaleも使用可能です。

```
live_loop :a do
  play chord(:E4, :m11).choose
  sleep 1
end
```

- shuffleを使うと、リストをシャッフルします。
- 次の例は、shuffleとplay_pattern_timedを使って生成した「リフ」です。

```
notes = chord(:E4, :m7).shuffle
times = [0.125, 0.25, 0.5].shuffle
live_loop :a do
  play_pattern_timed notes, times
  sleep 1
end
```

- ところで、先程の例は何度実行しても同じリフが生成されたと思います。
- Sonic Piのランダムは擬似乱数から生成されたもので、必ず同じものになります。
- これを変えるにはuse_random_seedを使います。

```
use_random_seed 1 # この行を追加
notes = chord(:E4, :m7).shuffle
times = [0.125, 0.25, 0.5].shuffle
live_loop :a do
  play_pattern_timed notes, times
  sleep 1
end
```

- リフが変わったと思います。
- use_random_seedに適当な値を入れていくことで、好きなだけリフを探求することができるようになりました。

- use_random_seedを使用しない場合は、use_random_seedに0を指定したのと同じです。

```
use_random_seed 0 # 0の場合、あってもなくても同じ
notes = chord(:E4, :m7).shuffle
times = [0.125, 0.25, 0.5].shuffle
live_loop :a do
  play_pattern_timed notes, times
  sleep 1
end
```