
Sonic Pi
Sam Aaron
&
The Core Team
Developed at the
University of Cambridge Computer Laboratory
with kind support from the Raspberry Pi Foundation
Version 2.6

日本語版

Part 1

1 章　Sonic Pi へようこそ(^o^)

Sonic Pi とは、ライブコーディングで新しいサウンドを演奏するために設計されたオープンソースのプログラミング環境です。

安価で小型のコンピュータ Raspberry Pi（ラズベリーパイ）にプリインストールされています。 Raspberry Pi 用の他、Mac OS 版と Windows 版も公開されており、どちらも無料でダウンロードできます。

- Sonic Pi 公式サイト
- Raspberry Pi 公式サイト

音楽は、作曲、記譜、演奏という方法で制作され、楽しまれてきました。では、新しい道具としてのコンピュータは、 この方法の中で音楽とどのように関わる事ができるのでしょうか。自動作曲やアクティヴ・スコア・ミュージックといった 新しい楽譜としての機能、また演奏者や楽器としてのコンピュータなど、コンピュータを使った試行錯誤によって刺激的な音楽が生まれてきました。

楽器としてのコンピュータを使う際の選択肢は大きく分けて 2 つあります。テクニックを駆使し テクノロジーを操るか、テクノロジーに操られるかのどちらかです。そして問題点として、 ギターを弾くように独創的な奏法が生み出されないことが指摘されていました。コンピュータには、プログラミングの過程で創意工夫の多くが生み出され、この Sonic Pi は、ソフトや機材に組み込まれた音に捕らわれずに創意工夫によって演奏するための考え方が詰まっています。

開発者と共にオープンソースやクリエーティヴコーディングといった考え方に賛同し、 このソフトを入り口に、プログラミングの面白さを知り、創造的な活動に役立ててもらえる事を願っています。

以下は Sonic Pi の使い方（チュートリアル）を日本語化したものです。

Welcome friend:-)

Sonic Pi（ソニックパイ）へようこそ。これから説明するクレイジーな音作りに、あなたも夢中になることを願っています。音楽、シンセ、プログラミング、作曲、演奏など、これから学ぶことは、本当に刺激的なものになるでしょう。

でも、ちょっと待ってください。まずは自己紹介が必要でしたね。失礼いたしました！私は、Sonic Pi を作成した Sam Aaron といいます。Twitter 上の@samaaron で見つけることができますので、連絡くれるとうれしいです。もしかすると、 僕が観客の前で演奏しているライブコーディングのバンド Meta-eX にも興味を持ってもらえるかもしれません。Meta-eX の演奏の中で使っているトラックの一つを、Sonic Pi の例の中で見つけることができます。

Sonic Pi を改善するために、気づいたことや、アイデアを持っていたら、是非、知らせてください。 フィードバックはとても役立ちます。あなたのアイデアが次の重要な機能になるかもしれません！

最後に、このチュートリアル（使い方）は、カテゴリ別にグループ化されたセクションに分かれています。 始まりから終わりまで、簡単に学べるように手引きを書いたので、自由に、色々なセクションを覗いてみてください。何か、不足している点に気がついたら、 将来のバージョンのために検討したいので、知らせてください。

では、早速はじめましょう！

1.1 ライブコーディング

Sonic Pi の最もエキサイティングな側面のひとつは、まるでギターをライブで演奏するかのように、 ライブでコードを書いて音楽を作ることができることです。つまり、ステージやコンサートで Sonic Pi が 使えるということです！

心を解き放て

これからのチュートリアルで、実際の Sonic Pi の詳しい使い方に入る前に、まず、ライブコーディングがどんなものか体験してみましょう。あまり（もしくは全然）わからなくても、心配ご無用！そのまま席についたまま、楽しんでいきましょう。

ライブループ

さぁ、はじめましょう！下のコードを上の空欄の Workspace(ワークスペース)にコピーしてみましょう。

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

左上の Run ボタンを押すと、いい感じの速さでバスドラムの音が聞こえてきます。Stop ボタンを押せば、いつでも音を止めることができます。ですが、ここではまだ Stop を押さずに、次のステップを実行しましょう。

1. バスドラムがまだ鳴っていることを確認します。
2. sleep の値を、0.5 から、1 より大きい値に書き換えてみましょう。
3. Run ボタンをもう一度押します。
4. ドラムの早さがどのように変わったかを確認してみましょう。
5. この瞬間を忘れないでください。これが、あなたが初めて Sonic Pi でライブコーディングをした瞬間です。この瞬間を忘れないでください。そしてこれが最後にはならないはず・・・

これは簡単でしたね。では他の要素を加えてみましょう。
sample :bd_haus の上に、 sample :ambi_choir, rate: 0.3 を追加してみます。コードはこのようになるはずです。

```
live_loop :flibble do
sample :ambi_choir, rate: 0.3
sample :bd_haus, rate: 1
sleep 1
end
```

では、ちょっと遊んでみましょう。値を変えてみてください。大きな値、小さな値、もしくはマイナスの値にしたとき、何が起こるでしょうか？では、:ambi_choir の rate: の値をほんの少し（0.29 とか）変えた時、どうなるでしょう？ sleep の値をすごく小さくすると、どうでしょう？操作が速すぎると、エラーが出て コンピュータが止まってしまいます。これはコンピュータがついていけないからです。（そんな時は、より大きい値を sleep に設定して Run ボタンをもう一度押しましょう。）では、sample の行に # をつけることで、「コメント」してみてください。

```
live_loop :flibble do
sample :ambi_choir, rate: 0.3
#   sample :bd_haus, rate: 1
sleep 1
end
```

コンピュータに#をつけた行を無視するよう命令したので、聞こえませんね。これは「コメント」と呼ばれます。Sonic Pi では、要素を削除したり追加したりするのにコメントを使います。最後に、楽しむための技をお伝えしましょう。以下のコードを、上の空欄の Workspace(ワークスペース)にコピーします。 そして、2 つは同時にループ（繰り返し）します。まずは精一杯、体験して楽しみましょう！

いくつかの試してみましょう。

・rate:の青い値を変更し、sample の音が変わることを聞いてみましょう。

・sleep の時間を変更し、それぞれのループを異なる速度で繰り返すのを聞いてみましょう。

・sample の行のコメントを解除して（#を削除）、ギターの逆再生を楽しみましょう。

・いくつかの mix: の値を 0 (最小値) から 1 (最大値)の間で変えてみましょう。

Run ボタンを押し、次に繰り返しがどう変化したかに耳を傾けることを忘れないでください。 うまくいかなくても気にしないでください。Stop ボタンを押して、Workspace 内のコードを削除して、 新しいコードをコピーアンドペーストして、再び演奏の準備をすれば良いです。 失敗することがどんなことよりも学習の近道になるのですから。

```
live_loop :guit do
with_fx :echo, mix: 0.3, phase: 0.25 do
 sample :guit_em9, rate: 0.5
end
#   sample :guit_em9, rate: -0.5
sleep 8
end

live_loop :boom do
with_fx :reverb, room: 1 do
 sample :bd_boom, amp: 10, rate: 1
end
sleep 8
end
```

さあ、これらが実際にどう動くか好奇心が動き出すまで、 そしてあなたが次に何をしたらいいか知りたくなるまで、演奏と実験を続けましょう。さあ、残りのチュートリアルを読む準備ができました。

次にあなたを待っているものは・・・

1.2 Sonic Pi のインターフェース

Sonic Pi のインターフェース(操作画面)は、音楽をコーディング(記述)するためとてもシンプルです。 ちょっと見てみましょう。

- A - 演奏の操作
- B - 編集の操作
- C - 情報とヘルプ
- D - コードエディタ
- E - ログ画面
- F - ヘルプシステム

A. 演奏の操作

ここにあるピンクのボタンで、音を再生したり停止するための主要な操作ができます。 Run ボタンは編集画面内のコードを走らせます。Stop

ボタンですべてのコードの動作を停止します。 Save ボタンは外部ファイルにコードを保存し、Record ボタンは演奏中の音を (WAV ファイルに)録音します。

B. 編集の操作

これらのオレンジ色のボタンで編集画面を操作します。 Size +と Size -ボタンは文字サイズを大きくしたり、小さくします。 Align ボタンはより専門的に見えるようにコードを整えてくれます。

C. 情報とヘルプ

これらの青いボタンで、情報、ヘルプや設定にアクセスします。Info ボタンは、 概要、コア、協力者、コミュニティ、ライセンス、ヒストリーといった Sonic Pi 自体についての情報を画面に表示します。 Help ボタンは F のヘルプシステム表示を切り替えます。 また Prefs ボタンは基本的なシステムを操作するための設定画面の表示を切り替えます。

D. コードエディタ

ここでは音楽を作曲/演奏したりするためのコードを書く領域です。コードを書いたり、消したり、 切り取り、貼り付けなどを行うためのシンプルな編集画面です。Google Docs やワードだと思ってください。 編集画面ではコードの意味合いに応じて自動で色つけします。はじめは変わっていると思うかもしれませんが、 とても使いやすくなるはずです。例えば、数値は青色で示されます。

E. ログ画面

コードを走らせた時、ログ画面にはプログラムがどうなっているかが表示されます。既定では、正確な再生時間を毎音ごとにメッセージとして見ることができます。コードのデバッグ（欠陥を取り除く作業）に利用でき、コードが何をしているかを理解することにも役立ちます。

F. ヘルプシステム

最後は、Sonic Pi インターフェースのもっとも重要な部分の一つである、画面の下にあるヘルプシステムです。青い Help ボタンによって表示/非表示を切り替えることができます。ヘルプシステムは、ヘルプと情報を含んでいます。このチュートリアル、入手できるシンセのリスト、 サンプルや実例、エフェクト（FX）のリスト、そして音楽をコーディングするために備えている Sonic Pi の 全機能のリストといった、Sonic Pi の全てについての、ヘルプと情報です。

1.3 遊びを通した学び

Sonic Pi は、遊びと実験を通して、コンピューティングと音楽の両方を学ぶ手助けをしてくれます。 もっとも大切なことは、楽しむこと。そうすれば、コードや作曲、演奏を、学ぶ前に自然と身につけられるでしょう。

間違いはない

この章の間に、何年も音楽とライブコーディングを学んできた僕から、ちょっとアドバイスを。

間違いはない、あるのはただ可能性だけ。

これは、しばしばジャズについて言われることですが、ライブコーディングにも同様に言えることです。経験に関係なく、あなたが全くの初心者でも、熟練のコーディング使いでも、作り出すコードが まったく予期せぬ結果になることがあります。めちゃくちゃクールに聞こえる場合は、そのコードを実行すればいいのですが、しかし、すごく耳障りで場違いに聞こえる場合もあります。 ですが、そうなっても問題ではありません。重要なのは、あなたが次にすべきことです。 音を取り除き、それを操作し、素晴らしいものに変化させるのです。観客は熱狂するでしょう。

シンプルにはじめよう

学習していると、今すぐすごいことをしたい気持ちに駆られます。しかし、今はその気持ちをこらえ、 それは後に到達する遠い目標として持ってください。今のところは、あなたの頭の中にあるすごいことに向かう価値ある小さな一歩として、楽しくやりがいをもって書くという「 最も単純な事」を考えていきましょう。一度、その誠実に学ぶイメージを持って、試し、 それを構築し、そして再生していけば、斬新なアイデアを与えてくれるでしょう。 すぐに、あなたは、楽しみながら、確実に上達をすることに大忙しになっているでしょう。

でも、みんなと作品をシェア（共有）することは忘れないでください！

2 章 シンセ

Sonic Pi の紹介はこれくらいにして、さっそく音を試してみましょう！

この章では、基本的なシンセの出し方と操作方法を紹介します。シンセは、 音を作り出すシンセサイザーという響きのよい言葉を短縮したものです。典型的なシンセは、使いこなすのが非常に複雑です。特にアナログのシンセには、沢山のパッチワイヤーとモジュールがついています。ですが、Sonic Pi では、とても簡単で親しみやすい方法で、このシンセの力を手に入れることが出来ます。

Sonic Pi の分かりやすくシンプルなインターフェイス(操作画面)に騙されないでください。 もし使いこなせれば、洗練されたサウンド操作を可能にします。きっと驚くはずです。

2.1 初めての音

下記のコードをみてください:

```
play 70
```

ここからすべてが始まります。アプリの上部のコードウィンドウ（実行ボタンの下に大きな空白）にコピーして貼り付けます。そして、左上の Run ボタンを押してみましょう。

ビープ音がなった!

ビックリした？もう一回押してみましょう。そしてもう一回..。

わお！すごい！一日中楽しんでいられそうだけど、でも待って。ビープ音の連続に夢中になる前に、数値を変えてみましょう。

```
play 75
```

違いが分かりますか？より低い値も試してみましょう。

```
play 60
```

つまり、低い値は音程（ピッチ）の低い音を、より高い値は音程（ピッチ）の高い音を鳴らします。 ちょうどピアノのように、低い部分の鍵盤（左手側）が低い音色を演奏し、高い部分の鍵盤（右手側）が高い音色を奏でます。実際に、数値はピアノの鍵盤と関係しています。play 47 は、 は実はピアノの 47 番目の鍵盤を演奏することを意味しています。play 48 は一音上がる（右隣りの鍵盤）ということです。第４オクターブの C は 60 ということになります。続いて、play 60 を鳴らしてみましょう。

もし、これがみなさんにとって何の意味をなさなくても心配しないでください。わたしも始めた時は同じでした。いま重要なのは、低い数値は低い音を、高い数値は高い音を生み出す、ということを知っておくことです。

和音

音符を奏でることはとても楽しいですが、同時にたくさんの音符を鳴らすとさらに楽しくなります。 これを試してみましょう。

```
play 72
play 75
play 79
```

いいですね！複数の play を書くと、全て同時に演奏されます。自分で試してみましょう. どの数値がいい組み合わせですか？逆にひどい音の組み合わせはどうでしょうか？経験、探求しながら、自分自身で見つけてみましょう。

メロディー

音符と和音を演奏するのは楽しいですね。でも、メロディーの演奏はどうすればいいでしょうか？同時ではなく、一音ずつ演奏したい場合は？それは簡単です。音符の間に sleep を入力すれば可能です。

```
play 72
sleep 1
play 75
```

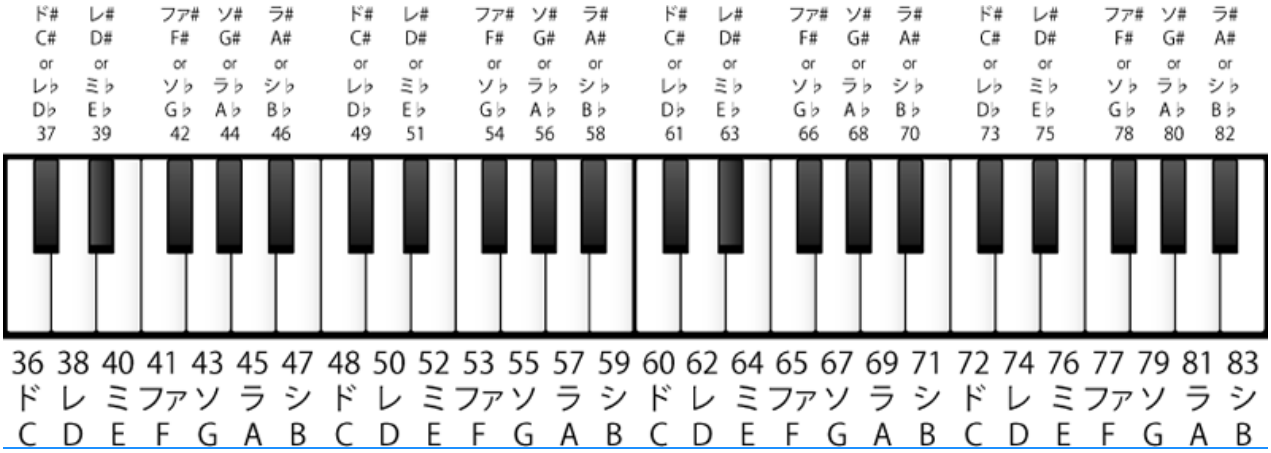
```
sleep 1
play 79
```

なんて素敵なアルペジオ（和音を続けて弾くこと）！では、sleep 1 の 1 は何を 意味するのでしょうか？これは一拍休む、という意味ですが、とりあえず今は、一秒休む、と考えましょう。では、アルペジオをもう少し早くしたいと思ったら？それは、短い値を使えばよいのです。 例えば半分の値 0.5 ではどうでしょう？

```
play 72
sleep 0.5
play 75
sleep 0.5
play 79
```

早くなりましたね。では、自分で時間を変えてみましょう。違う時間と音符を使ってみましょう。 たとえば play 52.3 や play 52.63 のような中間的な音符でも演奏してみてください。 通常の全音符で演奏し続ける必要は、まったくありません。遊んで、楽しみましょう。

伝統的な音階の名称



みなさんの中で、すでにいくつかの音楽記号を知っている人は、たとえば C とか F# などを使って メロディーを書きたいかも知れません（もしそうでなければ心配無用です、Sonic Pi を楽しむためには必要ありません）。Sonic Pi はそれをカバーしてくれます。以下のことをやってみましょう。

```
play :C
sleep 0.5
play :D
sleep 0.5
play :E
```

演奏する音階の前にコロン':'を入れることを忘れないでください。コロンを置くと、C が:C のように色が変わります。 また、音名のあとに数値を追加してオクターヴを指定することもできます。

```
play :C3
sleep 0.5
play :D3
sleep 0.5
play :E4
```

半音♭(シャープ)にしたい場合は、'play :Fs3'のように音名の後に's'を追加します。 半音♯(フラット)にしたい場合は、'play :Eb3'のように'b'を追加します。

夢中になって、自分の曲を作って楽しみましょう。

2.2 シンセのパラメータ：Amp と Pan

Sonic Pi はあらゆる音を作り出し、コントロールするパラメータの全てを提供しますので、どんな音を演奏するか、そしてどんなサンプルをトリガー（きっかけ）にするのかはあなた次第です。このチュートリアルでは、これらのパラメータに関してと、ヘルプシステムの詳細を紹介します。ひとまず、最も便利な 2 つ、Amplitude（振幅）と Pan（パン）を紹介します。ですが、まずはパラメータを、見てみましょう。

パラメータ

Sonic Pi はシンセのためのパラメータ（変数）という概念を備えています。パラメータは、あなたが耳にするサウンドの特徴をコントロールしたり、変更したりするための手段で、演奏に反映されます。シンセはそ

れぞれ、 細かく音をチューニングするためのパラメータを持っています。この Sonic Pi では、一般的なパラメータを多様な音色として amp:とエンベロープ・パラメータ（別のセクションで説明）で使用できます。

パラメータには 2 つの主要な役割があり、ひとつはその名前（制御の名前）、もうひとつは 数値（あなたが制御したい値）です。例えば、cheese:いうパラメータがあったとして、 1 の値にセットしたいとします。

パラメータは、play の後に ,（コンマ）を入れて、その後、amp:（コロン : を忘れずに）のような パラメータの名前、スペース、そしてパラメータの値、というように渡していきます。例えば、

```
play 50, cheese: 1
```

（ cheese: は無効なパラメーターです。例として使っています。）

カンマを使って区切り、複数のパラメーターを使用することができます。

```
play 50, cheese: 1, beans: 0.5
```

パラメーターの順番は重要ではないので、以下は同じものです。

```
play 50, beans: 0.5, cheese: 1
```

シンセで認識されないパラメータは無視されます（ cheese チーズと beans 豆などは明らかに馬鹿げた名前でしょう！）

もし偶然同じパラメータを 2 回、違う値で使った場合は、最後のものが有効です。 例えば、ここでの beans:は、0.5 ではなく 2 の値が採用されます。

```
play 50, beans: 0.5, cheese: 3, eggs: 0.1, beans: 2
```

Sonic Pi の中の命令には多くのパラメータをもっているので、その使い方に ちょっとだけ時間を使って、習得してください！

それでは、最初のパラメータ amp:で演奏してみましょう。

アンプ（振幅）

アンプは音の大きさをコンピュータで表したものです。高アンプは大きな音を生成し、 低アンプは静かな音を生み出します。Sonic Pi は時間と

音符を、数字を使って表現するので、アンプにも数字を使用します。1 の値が通常の音量であるのに対して、0 はサイレントです（何もきこえませんよ）。 2、10、100 というふうに、アンプを上げることができます。ただし、すべての音の全体の振幅が 高くなりすぎると、大きな音になりすぎないように、Sonic Pi では、コンプレッサー（圧縮）と呼ばれる 効果を使用して確実に音を抑えることを覚えておいてください。これは多くの場合音がこもって 奇妙に聞こえます。ですから、圧縮を防ぐために 0 から 0.5 のような値を使用してみてください。

音量を上げる

音の大きさを変えるために、 amp: パラメータを使います。 例として、半分のアンプで演奏するために、0.5 にしてみます。

```
play 60, amp: 0.5
```

倍の音量で演奏するために、2 にしてみます。

```
play 60, amp: 2
```

amp:パラメータは、関連付けられている play への命令だけを変更します。そのため、下の例では、最初の命令は半分の音量になり、次にはデフォルト（1 の値）に戻ります。

```
play 60, amp: 0.5
sleep 0.5
play 65
```

もちろん、それぞれの命令で異なった amp:の値を設定して演奏することもできます。

```
play 50, amp: 0.1
sleep 0.25
play 55, amp: 0.2
sleep 0.25
play 57, amp: 0.4
sleep 0.25
play 62, amp: 1
```

パンニング

もうひとつの面白いパラメータは pan:です。ステレオで音の位置を制御します。 左に音をパンすることは左のスピーカーから音が聞こえることを意味し、 右にパンすれば右のスピーカーから音が聞こえます。値としては、-1 は完全に左、 0 は中心、1 は完全に右、というようにステレオの領域で表現することができます。もちろん、音の正確な位置をコントロールするために、-1 から 1 の間のどの値でも使用することができます。

左のスピーカーから音を鳴らしてみましょう。

```
play 60, pan: -1
```

では、右のスピーカーから鳴らします。

```
play 60, pan: 1
```

最後に元の通り、中心から鳴らしてみます（通常の位置です）。

```
play 60, pan: 0
```

では、続けてあなたの音のアンプやパンを変えて楽しんでみましょう！

2.3 シンセの切り替え

これまで、ビープ音を鳴らして楽しんできました。けれども、もしかするとあなたはこのベーシックなビープ音に退屈し始めているかも知れません。これが Sonic Pi が提供してくれるサウンドの全てなのでしょうか？ビープ音だけではなくて、もちろん もっと他にもたくさんのライブコーディングがありますよね？はい、あります。このセクションでは Sonic Pi がもたらすエキサイティングな音の範囲を探求してみます。

シンセ

Sonic Pi は、シンセサイザーの略であるシンセと呼ばれる楽器の領域を持っています。 サンプルがすでに録音された音であるのに対して、シンセはあなたがそれをコントロールすることに応じて 新しいサウンドを

生み出すことができます（このチュートリアルの後半でみていきます）。Sonic Pi のシンセは、 パワフルで表現力に富んでいて、それで演奏したり探求したりすることを楽しめるはずです。 最初に、ひとまずここでシンセを使うために、選び方を学んでみましょう。

酔っ払いの saw と prophet

面白い音は、ノコギリ（saw）波です。試してみましょう。

```
use_synth :saw
play 38
sleep 0.25
play 50
sleep 0.25
play 62
sleep 0.25
```

他の音 prophet を試してみましょう。

```
use_synth :prophet
play 38
sleep 0.25
play 50
sleep 0.25
play 62
sleep 0.25
```

2 つのサウンドを繋げてみたらどうでしょう。一方のあとに、もう一方を。

```
use_synth :prophet
play 38
sleep 0.25
play 50
sleep 0.25
play 62
sleep 0.25
```

では同時に。

```
use_synth :saw
play 38
sleep 0.25
play 50
sleep 0.25
use_synth :prophet
play 57
sleep 0.25
```

use_synth コマンドは、 下に続く play にだけ影響していることに注意してください。 大きなスイッチのようなものだと考えてください。入力されたどんなシンセでも、新しい play への命令として演奏されます。use_synth で新しいシンセにスイッチを移すことができます。

シンセを見つける

演奏するために Sonic Pi がどんなシンセを備えているかを見るには、左端のメニュー（効果 の上）にあるシンセオプションを見てください。20以上が用意されています。これらは、わたしのお気に入りの数種類です。

- :prophet
- :dsaw
- :fm
- :tb303
- :pulse

音楽の中でシンセを切り替えて遊んでみてください。音楽の違うセクションで違うシンセを使うように、シンセを組み合わせて新しい音を作って楽しんでください。

2.4 エンベロープでのデュレーション

前半のセクションでは、音を走らせたい時にどのように sleep コマンドを 使用するかを見てきました。ですが、わたしたちはまだ音のデュレーション（再生時間）について 制御できていませんね。

音のデュレーションを制御するための、パワフルでシンプルな手段として、 Sonic Pi には ADSR amplitude envelope（ADSR アンプエンベロープ）という 概念があります（ADSR が何を意味するかはこのセクションの後半で紹介します）。アンプエンベロープは制御のための 2 つの便利な側面を有しています。

- 音のデュレーション（再生時間）を制御する
- 音のアンプ（音量）を制御する

デュレーション（再生時間）

デュレーションとは、音が発生する長さのことです。長いデュレーションは、より長く音を発生させることを意味します。Sonic Pi の全てのサウンドは、アンプ・エンベロープで制御でき、ひとつのエンベロープのデュレーション（再生時間）の長さは、サウンドの長さでもあります。

アンプ（音量）

ADSR エンベロープは、デュレーションの制御だけでなく、サウンドのアンプ（音量）の緻密な制御も可能にします。全てのサウンドは、サイレント（無音）で始まり、間に音があり、 またサイレントで終了します。エンベロープを使用すると、サウンドの音がある部分の音量を スライドさせたり、保持したりすることができます。これは、音楽の音量を上げたり下げたり するのを、誰かに指示するような感じです。例えば、あなたは誰かに「無音で始めて、 ゆっくりとフル・ボリュームにして、少しそのままで、そして一気に無音に戻る。」と頼むとします。 Sonic Pi はエンベロープでこれをプログラムできるようにしてくれます。

前のセクションで見てきたように、アンプの 0 は無音、1 は通常の音量です。

リリース・タイム（終わるまでの時間）

通常、全てのシンセのリリース・タイム（終わるまでの時間）は 1 です。これは、終了するまでに 1 秒間のデュレーションを持っているというこ

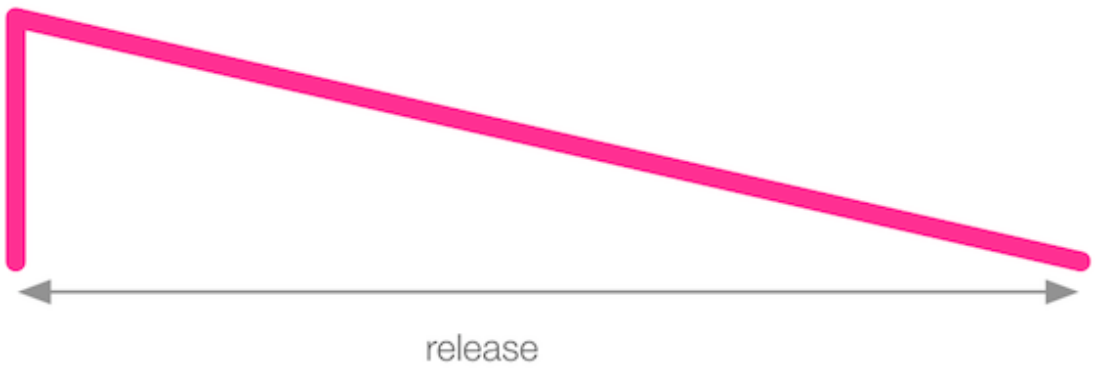
とです。 play に対して release:という命令を渡すことで、デュレーションを 変更することができます。例えば、2 秒間シンセを再生するには、release:を 2 に設定します。

```
play 60, release: 2
```

非常に小さなリリース・タイムの値を使って、シンセのサウンドをとても短くすることができます。

```
play 60, release: 0.2
```

だから、リリース・タイムって一体なんでしょうか？サウンドが、フル・アンプ（だいたい 1 の値）から ゼロ・アンプ（無音）になるまでにかかる時間のことです。これはリリース・フェーズと呼ばれていて、 リニア・トランジション（直線的な移行、つまり真っ直ぐ）です。以下の図は、この移行を表したものです。



図の左端の縦の線は、サウンドが 0 の音量(ゼロ・アンプ)からスタートしていることを示しています。 しかし、すぐにフル・アンプに上がります（これは私たちが後で学ぶ、アタック・フェーズというものです）。 いったんフル・アンプになり、それから release:で指定した値を取りながら、 直線的にゼロに下がります。長いリリース・タイムは 長いシンセ・フェード・アウト（徐々に消えること）を生成します。

これにより、リリース・タイムを変更して、あなたのサウンドのデュレーションを 変えることができます。自分の音楽にリリース・タイムを追加して演奏してみましょう。

アタック・タイム（音が発生する時間）

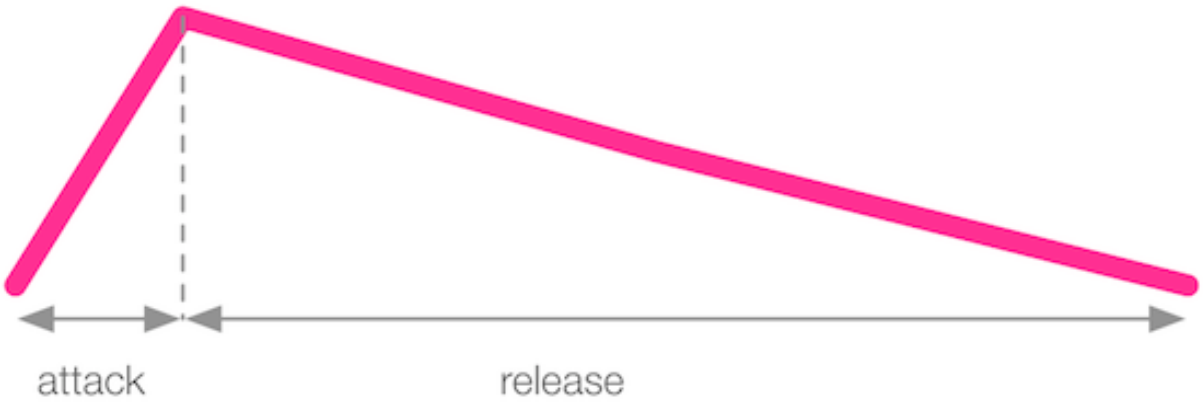
通常、アタック・フェーズは全てのシンセにおいて 0 です。つまり、0 アンプから 1 にただちに移動することを意味します。シンセは最初から音が打たれます。 けれども、あなたは音をフェード・イン（徐々に大きく）したいかも知れません。 これは、attack:の命令で実現することができます。いくつかの音をフェード・インしてみましょう。

```
play 60, attack: 2
sleep 3
play 65, attack: 0.5
```

複数の命令を使うこともできます。例えば、短いアタックに、長いリリース、やってみましょう。

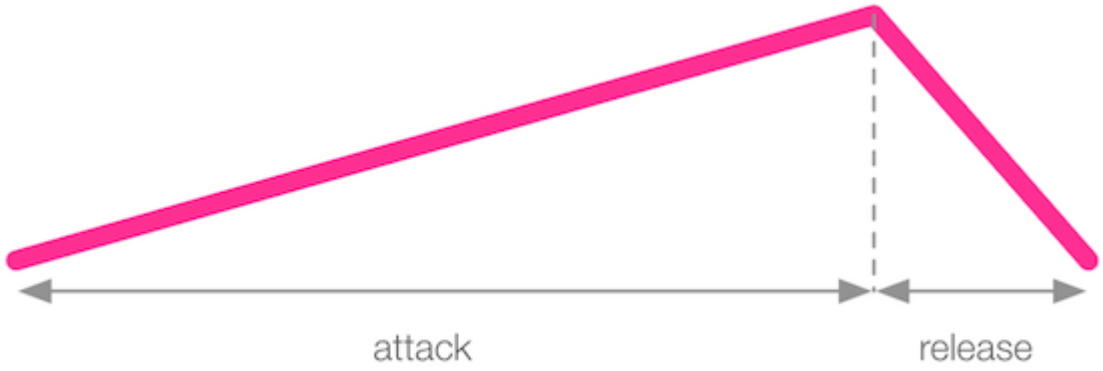
```
play 60, attack: 0.7, release: 4
```

この、短いアタックに長いリリースのエンベロープは、以下の図のように表します。



もちろん、ほかに変更することもできます。長いアタックに短いリリースを試してみましょう。

```
play 60, attack: 4, release: 0.7
```



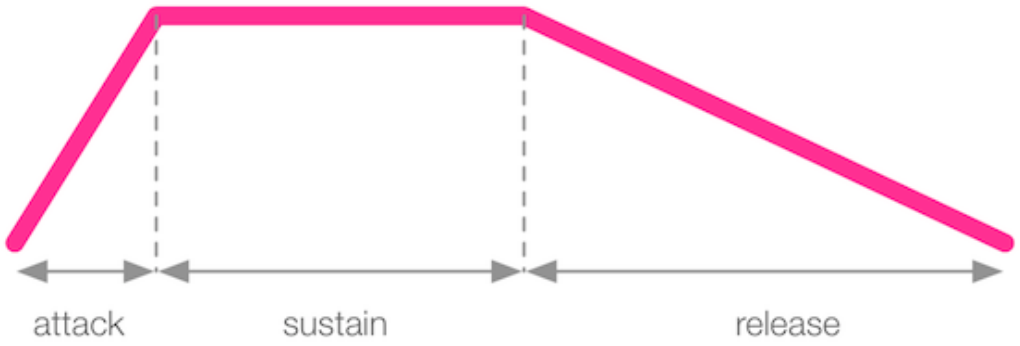
そして、アタックとリリース両方短くして、短いサウンドにもできます。

```
play 60, attack: 0.5, release: 0.5
```

サステイン・タイム（持続時間）

アタック・タイムとリリース・タイムの設定に付け加えて、サステイン・タイム（持続時間）を 指定することができます。サステイン・タイムとは、アタックとリリースの間でフル・アンプで 音が鳴り続ける時間のことです。

```
play 60, attack: 0.3, sustain: 1, release: 1
```

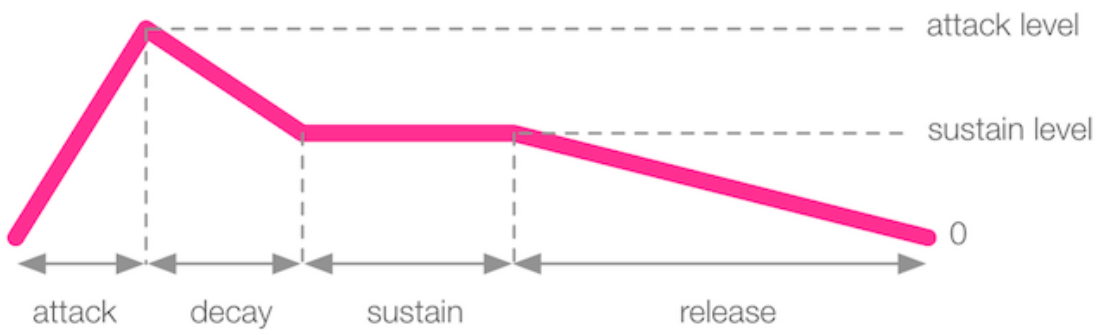


サステイン・タイムは、オプションのリリースフェーズに入る前に、ミックスの中で主となる 存在感を与えたい重要なサウンドにとって有効です。もちろん、アタックとリリース両方の命令を 0 にセットすることも全く有効ですし、サウンドに対して、完全にフェードインなし、 フェードアウトなしにするためにサステインを使えば良いのです。けれど、注意してください、0 のリリースはオーディオの中にクリック音を生成します。だいたいの場合、0.2 のようなごく小さい値を使う方が良いです。

ディケイ・タイム（減衰時間）

そして、今あなたがさらなる制御のレベルを必要としているならば、ディケイ・タイムというものを 設定することもできます。これは、アタックとサステインの間にくるエンベロープのフェーズで、 音量が attack_level から sustain_level に落ちる時間を指定します。通常では、ディケイの指令は 0 で、アタックとサステインのレベルは両方とも 1 です。ですから、ディケイ・タイムで効果を得るには、アタック、サステインのレベルも指定しなくてはなりません。

```
play 60, attack: 0.1, attack_level: 1, decay: 0.2, sustain_level: 0.4, sustain: 1, release: 0.5
```



ADSR エンベロープ

つまり要約すると、Sonic Pi の ADSR エンベロープには、以下のフェーズがあります。

1. attack（アタック）- 0 アンプから attack_level までの時間
2. decay（ディケイ）- 音量を attack_level から sustain_level まで移行させる時間
3. sustain（サステイン）- 音量を sustain_level で保持する時間
4. release（リリース） - 音量を sustain_level から 0 に移行させる時間

サウンドのデュレーションは、これらのフェーズの合計であることに注意することが大切です。 したがって、以下のサウンドは 0.5 + 1 + 2 + 0.5 = 4 で、4 秒のデュレーションになります。

```
play 60, attack: 0.5, decay: 1, sustain_level: 0.4, sustain: 2, release: 0.5
```

ではあなたのサウンドにエンベロープを追加して演奏してみましょう。

3 章 サンプル

あなたの音楽を発展させる方法が他にもあります。すでに録音された音を使うことです。偉大なヒップホップの伝統では、これらのあらかじめ録音された音のことを、サンプルと呼びます。つまり、マイクを持って外に出て、雨が優しくキャンバスを打つ音を録音しに行けば、それだけでサンプルを作ることができます。

Sonic Pi は、サンプルで楽しいことがたくさんできるようになっています。90 以上のパブリック・ドメイン（著作権がない）サンプルが、ジャム（即興演奏）するために準備されているだけでなく、自分で操作して演奏することも可能にしてくれるのです。さっそくみてみましょう。

3.1 サンプルを使う

ビープ音を演奏するのは最初だけです。もっと面白いのは、録音済みのサンプルを使うことです。やってみましょう。

```
sample :ambi_lunar_land
```

Sonic Pi は演奏のためのたくさんのサンプルを収録しています。play コマンドを使うようにサンプルを使うことができます。複数のサンプルを演奏するには、ひとつひとつ、順番に書いていきます。

```
play 36
play 48
sample :ambi_lunar_land
sample :ambi_drone
```

音と音の間隔を空けたいなら、 sleep を使います。

```
sample :ambi_lunar_land
sleep 1
play 48
sleep 0.5
play 36
sample :ambi_drone
```

```
sleep 1
play 36
```

最初のサウンドが終わる前に、次のサウンドが始まることに注意してください。sleep コマンドは、サウンドの開始の間隔だけを記述しています。これによって、簡単にサウンドを重ね合わせて、おもしろいオーバー・ラップ（重複）のエフェクト（効果）を生み出すことができるのです。このチュートリアルの後半では、エンベロープでサウンドの duration（再生時間）を制御する方法についてみてみます。

サンプルを探す

Sonic Pi が収録しているサンプルの種類を知るには、2 つの方法があります。一つ目は、ヘルプシステムを使うことです。下のメニューの中の、samples をクリックし、カテゴリーを選ぶと、使用可能なサウンドが表示されます。 あるいは、オート・コンプリーション（自動補完）システムを使うこともできます。 sample :ambi_のような複数のサンプルを束ねるグループ名を入力し始めると、同じグループ内から選べるサンプルの名前がリストとして表れます。下のサンプル・グループを入力して試してみましょう。

- :ambi_
- :bass_
- :elec_
- :perc_
- :guit_
- :drum_
- :misc_
- :bd_

さあ、あなたの曲でサンプルのミックスを始めてみましょう！

3.2 サンプルのパラメータ

シンセの項目で見てきたように、パラメータで簡単にサウンドを制御することができます。サンプルは全く同じパラメータの仕組みを備えています。これから何度も登場する amp: と pan: をもう一度見てみましょう。

サンプルをアンピングする

シンセで使ったのと全く同じ方法で、サンプルのアンプ（音量）を変えることができます。

```
sample :ambi_lunar_land, amp: 0.5
```

サンプルをパンする

サンプルで pan: のパラメータを使うこともできます。例えば、定番のドラムフレーズのアーメン・ブレイクを左耳で聞いた後に、半分は通過して再度右耳で聞く方法です。

```
sample :loop_amen, pan: -1
sleep 0.877
sample :loop_amen, pan: 1
```

0.877 は、:loop_amen サンプルの半分の再生時間であることに注意してください。 そして、複数のシンセを use_synth_defaults（あとで説明します）と使用している場合、サンプルはこれを無視して再生します。

3.3 サンプルを引き延ばす

すでに私たちは、音楽を作るために様々なシンセやサンプルを演奏することができます。そろそろ、音楽をもっとユニークで面白くするために、シンセとサンプルを編集する方法を学ぶ時間です。まずは、サンプルをストレッチ（引き延ばす）したり、スクワッシュ（圧縮）する方法をみてみましょう。

サンプルで表現する

サンプルはあらかじめ録音されたサウンドで、サウンドとして再生されるためにスピーカーの振動板（スピーカーのコーン）をどのように動かすかを表すための数値として収録されています。スピーカーのコーンは行ったり来たり（振動）するため、数値は、コーンがどれくらいの距離を行ったり来たりすべきかをその時々で表す必要があります。録音されたサウンドを忠実に再生するために、サンプルは概して 1 秒間に何千もの数値を収録しておく必要があるのです！Sonic Pi はこの数値のリストを使って、正しい方法でサウンドを再生するため、コンピュータのスピーカーを振動させる適切なスピードを提供します。一方で、サウンドを変えるために、スピーカーに与えられた数値でスピードを変えるのも楽しいですよ。

レートを変える

アンビエント・サウンドのひとつ、:ambi_choir で演奏してみましょう。デフォルト（既定値）で演奏するには、sample に rate: の命令を渡します。

```
sample :ambi_choir, rate: 1
```

これはデフォルト（既定値）のレート（数値）1 で演奏するので、まだ何も変わったところはありません。ですが、数値を他の値に変えてもよいのです。0.5 はどうでしょう。

```
sample :ambi_choir, rate: 0.5
```

ワオ！何が起きたのでしょう？そう、2 つのことが起きました。一つ目は、サンプルは再生に 2 倍の時間をかけていました。二つ目は、サウンドが 1 オクターブ低くなっていました。もう少し詳しく、これらのことを探ってみましょう。

引き延ばしてみる

引き延ばしたり圧縮したりして楽しいサンプルは、定番ドラムフレーズのアーメン・ブレイクです。通常のレートでは、ドラムン・ベース（音楽のジャンルのひとつ）のトラックを想定します。

```
sample :loop_amen
```

:loop_amen ですが、レート（数値）を変えると、音楽のジャンルが変わったように聞こえます。半分のスピードにすると、懐かしいのヒップホップに。

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

スピードを上げると、ジャングルのジャンルになります。

```
sample :loop_amen, rate: 1.5
```

では、最後のお楽しみとして、マイナスのレートを使うと何が起きるでしょうか。

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

ワオ！逆再生！いろいろなサンプルを、異なったレートで演奏してみましょう。ものすごく速い数値や、おかしなくらいゆっくりになる数値を使いながら、どんなおもしろいサウンドを作れるか、試してみましょう。

サンプル・レートの解説

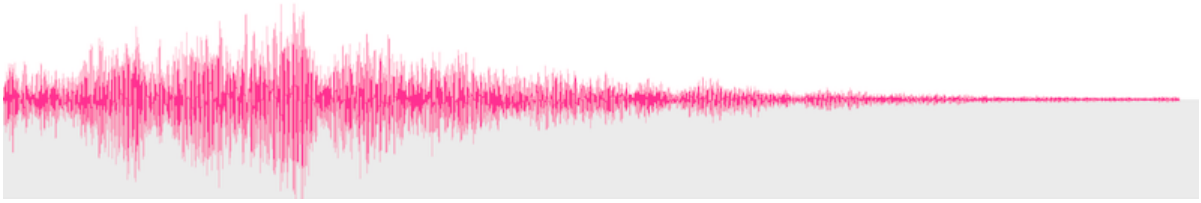
サンプルは、バネのように考えると便利です。再生速度（プレイバック）は、バネを縮めたり伸ばしたりするようなものです。もしサンプルをレート（数値） 2 で再生した場合、半分の長さにバネを縮めることになります。ですので、サンプルは半分の時間で演奏されるので、より短くなります。もしサンプルを半分のレートで再生した場合、2 倍の長さにバネを伸ばすことになります。そのため、サンプルは再生に 2 倍の時間をかけるため、より長くなるのです。もっと縮める（レートを上げる）と、短く再生され、さらに伸ばす（レートを下げる）と、短く再生されます。

バネを縮めることは、その密度（1cm あたりのコイルの数）を増やすことです。これは高いピッチ（音程）のサンプルに近づくことになります。同様に、バネを伸ばすことは、密度を減らすことで、低いピッチ（音程）のサウンド近づくことになります。

サンプル・レートの背後にある計算

（このセクションはさらに細かい部分に興味がある人向けです。飛ばしても構いませんよ・・・。）

上記で見てきたように、サンプルは、スピーカーが一定の時間どう振動すべきかを表現した、長大な数値のリストによって表されています。この数値のリストで、これと同じようなグラフを描いてみます。



あなたはこれと同じような図を、以前に見たことがあるかも知れません。これはサンプルの波形と呼ばれるものです。これは数値のグラフです。このような波形はだいたい、1 秒間に 44100 ポイントのデータを有しています（これはナイキスト・シャノンのサンプリング定理によるものです）。サンプルが 2 秒間続くなら、スピーカーに 1 秒間に 44100 ポイントのレートを与えることで 88200 の数値で表現されます。もちろん、1 秒間に 88200 ポイントで、2 倍のレートとすることもできます。これは、ですので、1 秒間で再生されます。1 秒間に 22050 ポイントで、半分のレートで再生することも可能です。この場合、4 秒で再生されます。

サンプルのデュレーション（再生時間）は、再生（プレイバック）レートに影響されます。

- 2 倍の再生レートは、再生時間を半分にする
- 半分の再生レートは、再生時間を 2 倍にする
- 4 分の 1 の再生レートは、再生時間を 4 倍にする
- 10 分の 1 の再生レートは、10 倍長く再生させる

これを以下の数式で表します。

new_sample_duration = (1 / rate) * sample_duration

再生レートを変えることは、サンプルのピッチ（音程）にも影響します。周波数や波形のピッチはどれくらい速く上下に動くかで決定されます。我々の脳は、どういうわけか、スピーカーの速い動きを高い音符に、遅い動きを低い音符と捉えます。だからあなたは時々、大きなベース・スピーカーが超低音を吐き出すのを見ることができるのです。実際に、高い音を出すスピーカーより、もっとゆっくり振動しているのです。

もし波形を取り出して圧縮したら、1 秒間でさらに頻繁に上下します。これがサウンドをより高い音にするのです。上下の運動を 2 倍にすると、周波数が 2 倍になるということになります。ですので、サンプルを 2 倍のレートで再生すると、聞こえる周波数が 2 倍になるということです。また、レートを半分にすると、周波数が半分になるということでもあります。他のレートはそれに応じて周波数に影響します。

3.4 エンベロープ・サンプル

ADSR エンベロープを用いて、サンプルのデュレーション（再生時間）とアンプ（音量）を編集することもできます。しかしながら、シンセの ADSR エンベロープとは少しだけ異なったように作用します。サンプルに使用するエンベロープは、サンプルのデュレーション（再生時間）とアンプ（音量）を減らすことしかできません。決して増やすことはできないのです。サンプルは、再生され終わったときか、エンベロープが完了したときのいずれかで停止します。どちらが先でも停止します。もし、非常に長い release: を使ったとしても、サンプルのデュレーション（再生時間）は延長されません。

アーメン・エンベロープ

わたしたちのお気に入りのフレーズ、アーメン・ブレイクに戻ります。

sample :loop_amen

パラメータが指定されていない場合、全サンプルが最大アンプ（音量）で聞こえます。もし 1 秒間のフェード・インをしたい場合、attack: パラメータを使います。フェード・インとは、徐徐に音が大きくなることです。

```
sample :loop_amen, attack: 1
```

もっと短いフェード・インの場合は、アタックの値を小さくします。

```
sample :loop_amen, attack: 0.3
```

オート・サステイン

ADSR エンベロープの特徴で、標準的なシンセに使用するエンベロープと異なる点は、サステイン（持続）の値です。 標準的なシンセのエンベロープでは、もし手動で変更しなければ、 0 に設定されています。 ですが、サンプルでは、サステインの値は通常、魔法のように自動的にセットされています。サステインの時間は、サンプル全てを演奏出来るように調整されます。デフォルトの値を渡さないときに、サンプルがフルで流れるのはこのためです。 アタック、ディケイ、サステイン、リリース（この４つの頭文字が ADSR）の値が全て 0 のときは、何の音も聞こえません。 Sonic Pi はサンプルの長さがどれくらいなのか計算し、アタック、ディケイ、リリースの時間を差し引いて、 あなたのサステインの時間を割り出します。もし、アタック、ディケイ、リリースの合計がサンプルのデュレーション（再生時間）より長い場合、サステインは 0 にセットされます。

フェード・アウト

これまでのことを探求するため、我らがアーメン・ブレイクをさらに詳細にみてみます。Sonic Pi にアーメン・ブレイクのサンプルがどれくらいの長さなのか尋ねてみます。

```
print sample_duration :loop_amen
```

1 秒間のサンプルの長さは 1.753310657596372 と答えるでしょう。ここでは、便宜的に 1.75 としますリリース時間を 0.75 にすると、驚くべきことが起こります。

```
sample :loop_amen, release: 0.75
```

サンプルの最初の 1 秒をフル・アンプで再生し、最後の 0.75 秒はフェード・アウトします。これが オート・サステインの機能です。標準では、リリースはいつもサンプルの最後から動作します。 もしサンプルの長さが 10.75 秒なら、最初の 10 秒はフル・アンプで再生し、最後の 0.75 秒でフェード・アウトするということです。

通常、release: はサンプルの最後でフェード・アウトすることを覚えておいてください。

フェード・インとフェード・アウト

サンプルのデュレーションの間に、フェード・アウトとインを行うには、attack: と release: を使用します。オート・サステインの仕組みと一緒に使用可能です。

```
sample :loop_amen, attack: 0.75, release: 0.75
```

アーメン・ブレイクのサンプルのデュレーションは 1.75 秒なので、アタックとリリースの時間が 1.5 秒まで追加されると、 サステインは自動的に 0.25 秒にセットされます。これで簡単にサンプルをフェード・イン、アウトすることができます。

明白なサステイン

手動で sustain: を 0 などにセットして、通常の ADSR シンセの挙動に簡単に戻すこともできます。

```
sample :loop_amen, sustain: 0, release: 0.75
```

いま、サンプルはトータルで 0.75 秒間だけ再生されました。attack: と decay: の通常値は 0 ですので、 サンプルはフル・アンプ（音量）で再

生され、0 秒間サステイン（維持）し、リリースの 0.75 秒間で 0 アンプに戻ります。

打楽器のシンバル

この仕様を、長いサンプルを短く、より打楽器のように活用するため、効果的に使うことができます。 :drum_cymbal_open:

```
sample :drum_cymbal_open
```

上のサンプルでは一定時間シンバルの音が鳴っているのが聞こえます。ですが、もっとパーカッシブ（打楽器的）にしてみましょう。

```
sample :drum_cymbal_open, attack: 0.01, sustain: 0, release: 0.1
```

シンバルを叩いた後のサステイン（維持）の時間を増やすことで、響かせるような効果も出すことができます。

```
sample :drum_cymbal_open, attack: 0.01, sustain: 0.3, release: 0.1
```

今すぐ、サンプルを楽しくさせるエンベロープをいじってみましょう。面白い結果を得るために数値を思いきり変更してみましょう。

3.5 部分的なサンプル

このセクションでは、Sonic Pi のサンプル再生について、私たちが探求してきたことをまとめます。 簡単にまとめてみましょう。

サンプルを再生する方法をみてきました。

```
sample :loop_amen
```

まず、サンプルのレート（数値）を変更する方法を見つけて、このように半分の速度で再生してみました。

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

次に、サンプルの徐徐に大きな音で再生するフェード・インを知りました。（半分の速度でやってみましょう）

```
sample :loop_amen, rate: 0.5, attack: 1
```

また、sustain: に明確な値とアタックとリリースの両方に短い値を設定することで、サンプルを打楽器のように使用できる方法を見つけました。

```
sample :loop_amen, rate: 2, attack: 0.01, sustain: 0, release: 0.35
```

では、いつもサンプルは、先頭から再生する必要はあるのでしょうか？そして、いつもサンプルを最後まで演奏して終了する必要はあるのでしょうか？

開始点の選択

サンプルは、どこからでも、好きな場所を選んで再生することが可能です。 サンプルの開始は 0 で、1 が終了です。0.5 は、サンプルの途中ということになります。アーメン・ブレイクの後半を再生してみましょう。

```
sample :loop_amen, start: 0.5
```

では、どのようにすればサンプルの最後の 1/4 を再生できますか？

```
sample :loop_amen, start: 0.75
```

終了点を選択する

同様に、サンプルの 0 から 1 までの値を、終了点として選ぶことができます。 アーメン・ブレイクを前半までで終わらせてみましょう。

```
sample :loop_amen, finish: 0.5
```

開始と終了の指定

もちろん、わたし達は、好きな部分を再生するために、開始と終了の２つを組み合わせることができます。どのようにすれば真ん中あたりの短い場所だけを選べるのでしょうか？

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

もし、終了位置を、開始位置として選ぶとどうなるでしょう？

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

かっこいい！！！逆再生になりますね！

レートとの組み合わせ

そして新しい機能として、すでに学んだ rate: と 開始と終了を使った部分再生を組み合わせることができます。

```
sample :loop_amen, start: 0.5, finish: 0.7, rate: 0.2
```

エンベロープとの組み合わせ

そして、わたし達は面白い結果を創りだすために、ADSR エンベロープと、今までのことを組み合わせることができます。

```
sample :loop_amen, start: 0.5, finish: 0.8, rate: -0.2, attack: 0.3, release: 1
```

今すぐ、サンプルとこれまでで紹介した楽しい内容を組み合わせて演奏してみましょう！

3.6 外部サンプル

内蔵のサンプルは、すぐに使用でき、再生することができる一方で、 あなたは別に録音された音を試してみたいと願うかもしれません。 Sonic Pi は完全にこの願いを叶えます。 まずは、あなたの音源の携帯性について、確認しましょう。

Portability

あなたが、内臓のシンセやサンプルだけを使って作曲した場合、コードだけが、あなたの音楽を忠実に再生ために必要なものになります。でもちょっと考えてみてください。それは、とても凄いことですよね！あなたの想い通りの音楽が再生出来る作品は、テキストによるシンプルなものなので、電子メールで周りの人に送ったり、 Gist で公開することが出来ます。コードを持ちさえすれば、あなたの友人と本当に簡単に共有することができます。

しかし、もしあなたが、自身で録音したサンプルを使ってしまうと、この携帯性が失われてしまいます。なぜならば、あなたの音楽を他の人達が再現しようとしても、あなたのコードだけでは再生出来ず、記録したサンプルが必要になってしまうからです。このことは、あなたの音楽を他の人が操作したり編集したり、 試してみたりする可能性を制限してしまうことがあるということです。もちろん、この事は、 あなた自身が録音したサンプルを使うことを止めるという事ではなく、 方法（データを送って、正しく共有するなど）に配慮していけばよいという事です。

自分で録音した（ローカルな）サンプル

では、どうやって、あなたが録音した WAV ファイルや AIFF ファイルをコンピューターで再生するのでしょうか？ sample のファイルを置く場所を指定（パスを通す）するだけで、再生できるようになります。:

```
sample "/Users/sam/Desktop/my-sound.wav"
```

Sonic Pi は自動的にサンプルを読み込んで再生します。あなたは、sample に今まで使ってきた全ての設定値を、あなたが録音した音に使うことができます。

```
sample "/Users/sam/Desktop/my-sound.wav", rate: 0.5, amp: 0.3
```

4 章 ランダム化

音楽にすこし面白さを加えるために、ランダムという素晴らしい方法があります。Sonic Pi は音楽にランダム性を追加するためにいくつかの素晴らしい機能を持っていますが、 Sonic Pi のランダムは、真のランダムではありません。これは一体何を意味しているのでしょう？勉強を開始する前に、この衝撃的な真実を見ていきましょう。

再現性

たいへん便利なランダム関数に、二つの数字の間 （最小値と最大値）でランダムな値が得られる rrand があります。（rrand はレンジド・ランダムの短縮です。）ランダムな音階を演奏してみましょう：

```
play rrand(50, 100)
```

おぉー、ランダムな音階を演奏しましたね。これは、音階 77.4407 を演奏しました。

50 と 100 の間の素敵なランダム音階でしたね。でも、ちょっと待ってください、上記で、私はあなたが再生したランダムな音階を正確に予測していまえんか？何か、怪しくないですか？ 再度コードを実行してみてください。ランダムのはずが、再び 77.4407 が選ばれましたよね？ 実は、ランダムにすることができないのです！ これが答えとなる擬似ランダムであり、真のランダムではないということです。Sonic Pi は、 反復可能な数としてのランダムを与えます。これは、あなたがマシンで作成した音楽が、誰か他の人のマシンで 再生されても同じように聞こえることを確認するのに大変便利な機能です。あなたの曲の中でいくつものランダ ム性を使用している場合でも。

もちろん、音楽に与えられる側面として、もし「ランダム」が 77.4407 を毎回選択した場合、 それは非常に面白くありません。しかし、それはしていません。以下のことを試してみてください。

```
loop do
play rrand(50, 100)
sleep 0.5
```

```
end
```

そう！最終的には、ランダムに聞こえますね。ランダム関数へ続いて呼び出されるその後の実行結果は ランダムな値を返します。ただし、また再生する場合正確に乱数値の同じシーケンスを生成し、まったく同じ 音が鳴ります。ランボタンが押されるたびに、まるですべての Sonic Pi コードが毎回同じ時間に戻るかように蘇ります。

それはまさにシンセのデジャヴのように繰り返されます！

ホーンテッド・ベル

ゾクッとするようなベルの音を使い、ランダム動作を取り入れた楽しい作例です。で繰り返しサンプルのベル音:perc_bell をループさせ、ベル音の間にランダムな数値と sleep を用いています。

```
loop do
sample :perc_bell, rate: (rrand 0.125, 1.5)
sleep rrand(0.2, 2)
end
```

ランダムなカットオフ

ランダム化のもう一つの楽しみ方の例は、ランダムにシンセのカットオフを加えることです。これを試してみるには絶好のシンセは、:tb303 エミュレータです。

```
use_synth :tb303

loop do
play 50, release: 0.1, cutoff: rrand(60, 120)
sleep 0.125
end
```

ランダムの種(シード)

もし、Sonic Pi が提供するランダム数値の特定の配列が気に入らない場合、use_random_seed を介すことで 別の開始点を選択することが可能です。シードの規定値は 0 ですので、異なるランダム体験のために別のシード番号を入力してみましょう！

まず下記を考えてみてください：

```
5.times do
play rrand(50, 100)
sleep 0.5
end
```

このコードを実行するたびに、 5 音階の同じフレーズが聞けるでしょう。異なるフレーズを聞くには、シードの数値を変更します。

```
use_random_seed 40
5.times do
play rrand(50, 100)
sleep 0.5
end
```

こうして異なる 5 音階のフレーズを生成します。シードの数値を変更することによって、あなたの好きなフレーズを見つけることができます。他の人と共有すれば、彼らも、あなたが聞いたものとまったく同様のフレーズを聞くことができるでしょう。

有用なランダム関数をもう少し見ていきましょう。

選択

一般的な方法は、あらかじめ用意したリストの中からランダムに選択することです。例えば、60、65 または 72 の中から 1 音を演奏することができます。choose を用いれば、リストから一つの項目を選択させることできます。この場合、まず、カンマで区切った番号のリストを角括

弧でラップし、 配置する必要があります：[60, 65, 72]。次にそれらを choose に続ける必要があります。

```
choose([60, 65, 72])
```

どんな音になるか聞いてみましょう。

```
loop do
play choose([60, 65, 72])
sleep 1
end
```

rrand

すでに rrand について触れてきましたが、再び実行してみましょう。これは、2 つの値の間にある乱数のみを実行します。 この意味するところは指定値の最大と最小のいずれの値も含まれません。- 常に両者の間にある値です。 そして、その番号は常に浮動小数点になります - それは整数ではなく、分数です。

rrand(20, 110)で返される浮動小数点数の例：

- 20.343235
- 42.324324
- 100.93423

rrand_i

時々、あなたは小数点ではなく、整数の乱数を望むこともあるでしょう。これは rrand_i を用いることで 解決できます。それは小数点を除いて rrand と同様に最小値および最大値の範囲この場合最大値、最小値は含まれます）に潜在するランダム値を返す動作をします。下記は rrand_i(20, 110)により返される数値の例です：

- 20
- 46
- 99

rand

rand は、0 を含む最小値と最大値未満の間のランダムな浮動小数点数を返します。規定値では 0 と 1 の間の値を返します。したがって、amp: 値をランダム化する際に便利です。

```
loop do
play 60, amp: rand
sleep 0.25
end
```

rand_i

rrand_i と rrand の関係と同様に rand_i は 0 と特定の最大値の間の整数値を返します。

dice:サイコロ

ランダムな数字を出す際に、サイコロ投げをまねてみたくなることもあるでしょう。これは、常に最小値が 1 である rrand_i の特殊なケースです。dice を呼び出す時はサイコロの面の数を指定する必要があります。標準的なサイコロは 6 面で、dice(6)では、1 、2 、3 、4 、5 、または 6 を返すサイコロと同様の作用をします。しかし、空想のボードゲームのように、4 面、12 面、または 20 面サイコロで値を見つけたいこともあるでしょう。ことによっては 120 面のサイコロだって！

one_in

最後に、サイコロ投げをまねて、例えば 6 という一番大きな数値を出してみたくなることもあるでしょう。one_in はサイコロの面の数分の 1 の確率で true を返します。したがって one_in(6)では 6 分 1 の確率で true、それ以外の場合は false を返します。true と false の値は、このチュートリアルの次のセクションで説明する if 文で非常に有用です。

さあ、ランダム性を使ってコードをまぜこぜにしていきましょう！