1 – Hoş geldiniz ☺

Sonic Pi'a hoş geldiniz. Umarım siz de çılgın müzikler yapmaya başlamak için, benim size öğretmek için heyecanlı olduğum kadar heyecanlısınızdır. Müzik, sentez, programlama, kompozisyon, performans ve daha fazlasına doğru eğlenceli bir yolculuğa başlıyoruz.

Fakat durun, ne kadar da kabayım! Kendimi tanıtmama izin verin – Ben Sam Aaron – Sonic Pi'ı yaratan kişiyim. Beni Twitter'dan @samaaron adıyla bulabilirsiniz, size selam vermekten mutluluk duyarım. Ayrıca, seyirci önünde Sonic Pi ile kodlayarak canlı müzik yapma performansımı da görmek isteyebilirsiniz.

Eğer aklınızda Sonic Pi'ı geliştirmeye dair fikirler varsa lütfen aktarın, geri dönüt almak çok yardımcı oluyor. Fikrinizin sonraki yenilik olabileceğini asla bilemezsiniz!

Bu rehber kategorize edilmiş bölümlerden oluşmaktadır. Bu rehberi baştan sona kolay bir öğrenme aracı olması için yazdım, istediğiniz bölümlere özgürce girip çıkmaktan çekinmeyin. Eğer eksik bir kısım olduğunu düşünüyorsanız daha sonraki versiyonlarda eklemem için bana söyleyebilirsiniz.

Son olarak, diğer insanları kodlayarak canlı müzik yaparken izlemek de çok iyi bir öğrenme yoludur. Düzenli olarak http://youtube.com/samaaron kanalında canlı yayın yapıyorum, uğrayıp selam vermekten ve soru sormaktan çekinmeyin ☺

1.1Canlı Kodlama

Sonic Pi'ın en ilgi çekici özelliklerinden bir tanesi, sanki gerçek bir olanak tanımasıdır. Demek oluyor ki sadece biraz pratikle Sonic Pi'ı alıp sahneye çıkabilirsiniz.

-Zihninizi Boşaltın

Rehberimizin devamında Sonic Pi'ın detaylarına inmeden önce size, canlı kodlamanın nasıl bir deneyim olduğunu göstermek istiyorum. Çok fazla (ya da hiç) anlamasanız da endişelenmeyin, sadece koltuğunuza yaslanıp keyfini çıkartın.

-Canlı Döngü

Haydi başlayalım, aşağıdaki kodu yukarıdaki katmanlardan birine kopyalayın:

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

Şimdi, `Run` tuşuna tıklayın ve hoş bass bateri ritmini duyun. Ne zaman sesi susturmak isterseniz `Stop` tuşuna basabilirsiniz. Ancak hemen basmayın, henüz değil. Onun yerine aşağıdaki adımları takip edin:

1. Bass bateri sesinin hala devam ettiğinden emin olun
2. değerini `0.5`'den `1` gibi daha yüksek bir değerle değiştirin
3. `Run` tuşuna tekrar basın
4. Bateri hızının nasıl değiştiğini fark edin.
5. Son olarak, bu anı unutmayın, bu ilk canlı kodladığınız an ve son da olmayacak…

Tamam, bu yeterince basitti. Haydi mix'e başka şeyler ekleyelim. `sample :bd_haus` satırının üstüne `sample :ambi_choir, rate: 0.3` satırını ekleyin. Kodunuz bu şekilde gözükmeli:

```
live_loop :flibble do
  sample, rate: 0.3
```

```
  sample :bd_haus, rate: 1
  sleep 1
end
```

Şimdi, biraz kodla vakit geçirin. Değerleri değiştirin – daha yüksek değerler kullandığınızda ne oluyor, daha düşük değerler ya da negatif değerler? `:ambi_choir` için `rate:` değerini değiştirdiğinizde (mesela 0.29) ne olduğunu görün. Çok çok küçük bir değeri seçerseniz ne oluyor? Eğer çok küçük bir değer verirseniz bilgisayarınız ritme yetişemeyip bir hata vererek duracaktır (eğer bu olursa daha büyük bir `sleep` değeriyle programı tekrardan çalıştırın).

`Sample` satırlarından birine `#` ekleyerek yoruma dönüştürün, böylece kodunuzdan çıkmış olacak:

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

Bilgisayarınızın o satırı nasıl görmezden geldiğini fark ettiniz mi, böylece o satırı duymuyoruz. Buna yorum denir. Sonic Pi'da yorumları kodumuzdan bir şeyler çıkarmak ya da kodumuza bir şeyler eklemek için kullanabiliriz.

Son olarak, size oynaması eğlenceli olacak bir şey bırakmama izin verin. Aşağıdaki kodu alıp farklı bir katmana kopyalayın. Şimdi, çok fazla anlamaya çalışmadan sadece iki tane farklı döngü olduğunu görün – yani iki şey aynı anda devam ediyor. Şimdi, en iyi yaptığınız şeyi yapın – deneyin ve oynayın. İşte birkaç tavsiye:

- Mavi `rate:` değerlerini değiştirip sesin nasıl değiştiğine bakın.
- `Sleep` değerlerini değiştirerek iki döngünün de nasıl birbiri etrafında farklı
- Yoruma dönüştürdüğünüz satırı tekrar eski haline getirip arkada çalan gitarın keyfini çıkartın.

- Mavi `mix:` değerlerinden birini 0 ile 1 aralığında değiştirin.

`Run` tuşuna basıp değiştirdiğiniz şeyin sonraki döngü başlangıcında sesteki etkisini görün. Eğer bir şekilde zor durumda kalırsanız endişelenmeyin, sadece `Stop` tuşuna basın, kodu silip değiştirilmemiş halini tekrardan yapıştırın. Böylece tekrar doğaçlamaya başlayabilirsiniz. Hata yapmak öğrenmenin en kısa yoludur…

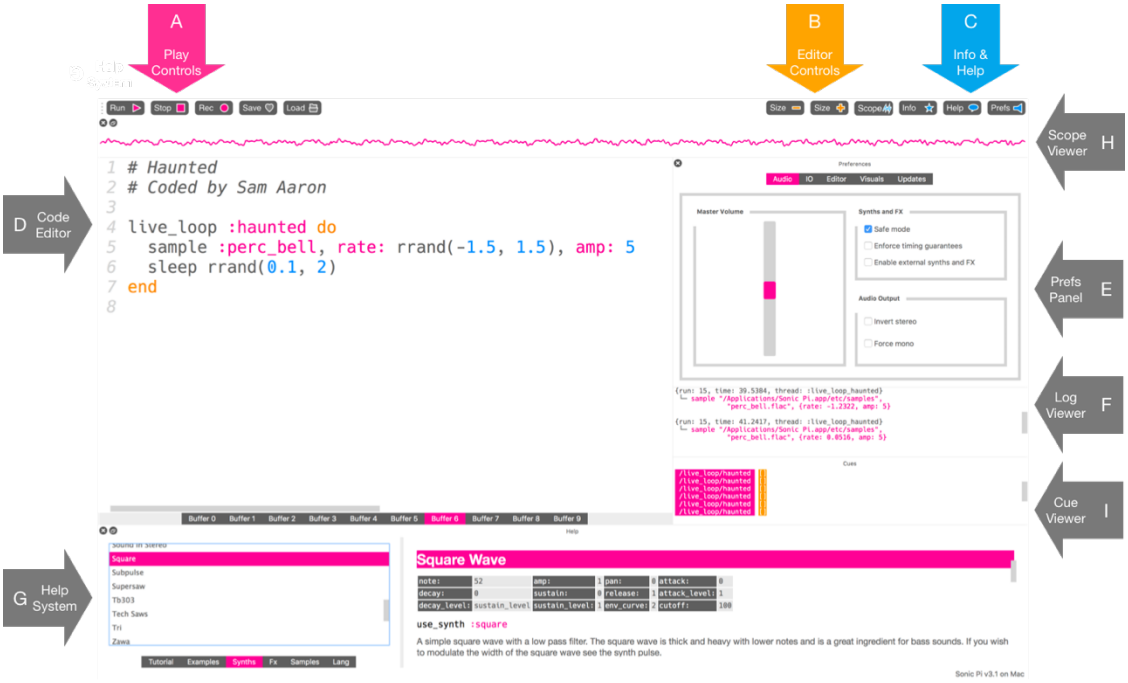
```
live_loop :guit do
  with_fx :echo, mix: 0.3, phase: 0.25 do
    sample :guit_em9, rate: 0.5
  end
#  sample :guit_em9, rate: -0.5
  sleep 8
end

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

Şimdi, merakınız tetiklenene ve bununla daha neler yapabileceğinizi düşünmeye başlayana kadar denemeye ve oynamaya devam edin. Şimdi rehberin geri kalanını okumaya hazırsınız.

Ee, daha ne duruyorsunuz…

1.2– Sonic Pi Ara yüzü

Sonic Pi müzik kodlamak için çok basit bir ara yüze sahip. Haydi biraz zaman geçirerek keşfedelim.



- A – Oynatma Kontrolleri
- B – Düzenleme Kontrolleri
- C – Bilgilendirme ve Yardım
- D – Kod Düzenleyicisi
- E – Tercih Paneli
- F – Geçmiş Paneli
- G – Yardım Sistemi
- H – Kapsam Görüntüleyici

A. Oynatma Kontrolleri

Bu pembe tuşlar sesleri başlatma ve bitirmenin ana yoludur. Run tuşu düzenleyicideki kodu çalıştırmaya, Stop tuşu da çalışan bütün kodları durdurmaya, Save tuşu kodu dış bir dosyaya kaydetmeye ve Record *tuşu ise çalan sesi kaydetmeye (WAV dosyası şeklinde) yarar.*

B. Düzenleyici Kontrolleri

Bu turuncu tuşlar kod düzenleyicisini kontrol etmenizi sağlar. *Size +* ve *Size –* tuşları yazıyı büyütmeye ve küçültmeye yarar.

C. Bilgilendirme ve Yardım

Bu mavi tuşlar bilgilendirmelere, yardıma ve tercihlere erişiminizi sağlar. *Info* tuşu Sonic Pi'ın kendisi hakkında – çekirdek ekip, tarihçe, yardımcı olanlar ve topluluk- hakkında bilgi veren bir pencere açar. *Help tuşu yardım sistemini (G) aktifleştirir ve Prefs tuşu basit sistem parametrelerini kontrol edebileceğiniz bir tercihler penceresi açar.*

D. Kod Düzenleyicisi

Bu alan kodunuzu yazdığınız ve müzik icra ettiğiniz alandır. Kod yazabileceğiniz, silebileceğiniz, kesip yapıştırabileceğiniz basit bir yazı düzenleyicisidir. Word veya Google Docs'un basit bir versiyonu gibi düşünebilirsiniz. Bu düzenleyici kelimeleri anlamlarına göre otomatik olarak renklendirir. Bu özellik başlangıçta garip gelebilir, ancak zaman geçtikçe ne kadar faydalı olabileceğini göreceksiniz. Örneğin, bir şeyin mavi olmasından onun bir sayı olduğunu anlayabileceksiniz.

E. Tercih Paneli

Sonic Pi Bilgilendirme ve Yardım bölümündeki *prefs* butonuna tıklanarak ulaşılabilen bir tercihler penceresine sahiptir. Bu içinde bir miktar ayar bulunan tercihler penceresinin açılmasını sağlar. Örneğin, mono ses moduna geçme, stereoya geçme, Raspberry Pi'de ses seçici vb.

F. Geçmiş Paneli

Kodunuzu çalıştırdığınızda programın ne yaptığı bu panelde gösterilir. Varsayılan olarak yarattığınız sesle eş zamanlı olarak sesle ilgili bir mesaj yazar. Bu özellik kodunuzdaki hataları görmek ve kodunuzun ne yaptığını anlamak için önemlidir.

G. Yardım Sistemi

Sonic Pi'ın en önemli kısımlarından birisi de pencerenin altında yer alan yardım sistemidir. Bu sistem mavi *Help* tuşuna tıklanarak aktifleştirilebilir. Yardım sistemi Sonic Pi'ın içinde bulundurduğu her şeyle, bu rehber, synthler, örnekler, FX ve Sonic Pi'ın sağladığı bütün fonksiyonların listesi, ilgili yardım ve bilgiler içerir.

H. Kapsam Görüntüleyici

Kapsam görüntüleyici duyduğunuz sesi aynı zamanda görmenizi sağlar. Saw sesinin gerçekten de bir testereye benzediğini ve basit bip sesinin ise bir sinüs dalgası olduğunu kolayca görebilirsiniz. Ayrıca dalgaların boyutundan çaldığınız sesin kısık ya da yüksek olduğunu da anlayabilirsiniz. Oynayabileceğiniz 3 tane kapsam bulunmakta – varsayılan kapsam sağ ve sol kanalı birleştiren bulunmakta, ayrıca her bir kanal için ayrı bir kapsam oluşturan stereo kapsamı bulunmakta ve son olarak da Lissajous eğrisi sağ ve sol kanalın arasındaki ilişkiyi kullanarak güzel resimler çizebilirsiniz (https://en.wikipedia.org/wiki/Lissajous_curve).

1.3– Çalarak Öğrenmek

Sonic Pi sizi bilgisayarı ve müziği aynı anda çalarak ve deneyerek öğrenmeye teşvik eder. En önemli şey eğleniyor olmanız ve ayrıca nasıl kodlanacağını ve doğaçlama yapılacağını da öğrenmiş oluyorsunuz.

-Hata Yoktur

Hazır bu konudayken size biraz yıllarca süren canlı kodlayarak müzik yapma hakkında edindiğim bilgileri tavsiye olarak vermeme izin verin – hata yoktur, sadece fırsatlar vardır. Bu benim sıklıkla Caz hakkında duyduğum bir şey fakat aynı şey canlı kodlama için de geçerli. Ne kadar tecrübeli olduğunuz önemsiz – yeni başlayan birinden yıllarca kodlayarak müzik yapmış birine kadar herkes bazen beklenmedik sonuçlar alabilir. Bazen bu beklenmedik sonuçlar kulağa güzel gelebilir ve bazen müziğinize onu da dahil edersiniz. Fakat, bazen tamamen kötü de olabilir. Ne olduğu önemsiz, tek önemli olan onunla ne yaptığınız. Sesi alın, değiştirin ve müziğinizle uyumlu mükemmel bir şeye dönüştürün.

-Basit Başlayın

Öğrenirken muhteşem şeyler yapmanın cazibesini göreceksiniz. Fakat, bu isteği birazcık sonraya saklayın. Şimdilik sadece basit şeyleri düşünüp bundan zevk almaya çalışın ve basit adımlar atın. Aklınıza bu basit adımlar hakkında fikirler geldiğinde onu gerçekleştirmeye çalışın ve verdiği fikirlerle çalın. Gerçekten yol kat etmekle meşgul olmaya başlamadan önce, yaptıklarınızı başkalarıyla paylaşmaktan çekinmeyin

2-Synths

Bu kadar giriş yeter, artık seslere geçiş yapabiliriz.

Bu bölümde synthleri tetikleme ve manipüle etme temellerini göreceğiz. Synth synthesiser kelimesinin kısaltması. Sytnhleri kullanması biraz karışık. Fakat Sonic Pi kullanımı kolay bir hale getiriyorç

2.1- İlk Bipleriniz

Aşağıdaki örnek koda bakalım:

```
play 70
```

Bu kod her şeyin başlangıcı. Bu kodu kopyalayıp kod alanına (Run tuşunun altındaki beyaz alan) yapıştır ve sonra Run tuşuna bas!

-Biipp!

Etkileyici! Tekrar bas. Şimdi yine bas. Yine…

Vay be çılgın, Eminim bütün gün boyunca bu tuşa basıp durabilirsin. Fakat kendini sonsuz Beep'lerin içinde kaybetmeden önce, numarayı değiştirmeyi deneyebilirsin:

```
play 75
```

Bir fark duyabiliyor musun? Daha düşük bir numarayı dene:

```
play 60
```

Düşük rakamlar düşük perdede seslere, yüksek rakamlar yüksek perdede seslere yol açıyor. Aynı piyano gibi, piyanonun alt tarafındaki (sol tarafı) tuşlar daha düşük notalar çalarken üst tarafındaki (sağ taraf) tuşlar daha yüksek notaları çalıyor. Aslında bakarsak bu numaralar piyanonun tuşlarına denk geliyor, `play 47` demek piyanonun 47'inci notasını çal demek. Bu da demek oluyor ki `play 48` aslında bir nota üstte (sağdan bir nota üstte). 4.oktav C, numara 60 oluyor.

-Akorlar

Tek bir nota çalmak eğlenceli fakat birden fazla notayı aynı anda çalmak çok daha iyi: Alttaki örneği kopyalayıp yapıştır ve dene:

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

Senin de fark ettiğin üzere birden fazla `play` yazdığında hepsi aynı anda çalıyor. Kendin yeni notaları dene, hangileri birlikte güzel oluyor? Hangileri kötü oluyor? Denemekten korkma, keşfet!

-Melodi

Notaları ve akorları oynatmak eğlenceli, fakat bir melodiye ne dersin? Ya bir notayı başka bir notadan sonra oynatmak istersen o zaman napıcaksın? Tek yapman gereken arlarına `sleep` koymak!

```
play 72
sleep 1
play 75
sleep 1
play 79
```

Ne kadar güzel, küçük bir arpej. Peki `sleep 1`'deki `1` ne demek oluyor? `sleep`'ten sonra gelen numara *sleep'in süresini* belirtiyor. Şimdi arpejimizi daha hızlı yapmak istersek eğer `sleep`'lerin değerlerini daha küçültmemiz gerkiyor.

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

2.2 Synth Seçenekleri: Amp ve Pan

Ne kadar sizin hangi notayı çalmak istediğinizi ve hangi modeli başlatmak istediğinizi kontrol etse de Sonic Pi size çok geniş bir ses kontrol mekanizması sunuyor. Bu eğitimlerde çoğunun üstünden geçeceğiz ve bu tür mekanizmalar için yardım sisteminde büyük bir arşiv var. Fakat, şimdilik bunlardan en önemli ikisi olan amplitude ve pan üstünde daha fazla duracağız.

-Seçenekler

Sonic Pi kendi synth'leri için seçenekler kavramını destekler. Seçenekler, duyduğunuz sesin özelliklerini değiştiren ve kontrol eden, kontrollerdir. Her synth, sesini hassas bir şekilde ayarlamak için kendi seçeneklerine sahiptir. Ancak `amp:` ve zarf seçenekleri (başka bir bölümde ele alınmıştır) gibi birçok ses tarafından paylaşılan ortak seçenek grupları vardır.

Seçeneklerin iki ana bölümü vardır: isimleri (kontrolün adı) ve değerleri (kontrolü ayarlamak istediğiniz değer). Örneğin `cheese:` adında bir seçeneğe sahip olabilirsiniz ve bunu 1 değerine ayarlamak isteyebilirsiniz.

Seçenekler, virgül kullanarak çalmak için aramalara iletilir ve ardından `amp:` gibi bir seçenek adı bulunur: (iki nokta üst üste'yi unutma:) ve sonra bir boşluk ve seçeneğin değeri. Örneğin:

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

(`cheese:` aslında geçerli bir seçenek değil, biz sadece onu örnek göstermek için kullandık.)

Birden çok seçeneği virgülle ayırarak geçirebilirsiniz:

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

Seçeneklerin sırası önem arz etmemektedir.

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

Synth tarafından tanınmayan seçenekler sadece göz ardı edilir. (Açıkça saçma tercih isimleri olan `cheese` ve `beans` gibi!)

Yanlışlıkla aynı tercihi farklı değerlerle iki kez kullanırsanız, sonuncusu kazanır. Örneğin, `beans:` burada 0,5 yerine 2 değerine sahip olacak:

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

Sonic Pi'deki pek çok şey seçenekleri kabul ediyor, bu yüzden onları nasıl kullanacağınızı öğrenmek için biraz zaman harcayın. İlk seçeneğimizle oynayalım: `amp:`.

Genlik

Genlik, bir sesin yüksekliğinin bir bilgisayar temsilidir. Yüksek genlik yüksek bir ses çıkarırken düşük genlik sessiz bir ses çıkarır. Sonic Pi, zaman ve notları temsil etmek için sayılar kullandığı gibi, genliği temsil etmek için de sayılar kullanır. 0 genliği sessizdir (hiçbir şey duyamazsınız), 1 genliği ise normal ses seviyesindedir. 2, 10, 100'e kadar daha yüksek bir genlik bile artırabilirsiniz. Ancak, tüm seslerin genliği çok yükseldiğinde, Sonic Pi'nin o sesleri sıkıştırmak için kompresör denen şeyi kullandığını unutmayın çünkü bu ses kulaklarınız için çok yüksek. Bu genellikle sesi çamurlu ve garip yapabilir. Bu nedenle, kompresyondan kaçınmak için 0 ila 0.5 aralığında yani düşük genlikleri kullanmayı deneyin.

Genliğe Bağlı

Bir sesin genliğini değiştirmek için `amp:` seçeneğini kullanabilirsin. Örneğin, yarı genlikte oynamak için 0.5'e geçin:

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

Çift genlikte çalmak için:

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

`amp:` opt, yalnızca aradığı kişiyle çalınacak aramayı değiştirir. Bu nedenle, bu örnekte, çalınacak ilk çağrı yarım ses düzeyindedir ve ikincisi varsayılana döner (1):

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

Tabii, farklı `amp:` değerleri de kullanabilirsiniz.

```
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
```

Sesin Nerden Geleceğine Karar Vermek

Kullanmak için bir başka eğlenceli seçenek ise `pan:` sesin stereo olarak kaydırılmasını kontrol ediyor. Sesi sola kaydırmak, sol hoparlörden duyduğunuz ve sağa kaydırmak, sağ hoparlörünüzden duyduğunuz anlamına gelir. Değerlerimizde, tamamen solu temsil etmek için -1, merkezi temsil etmek için 0'I ve tamamen sağı temsil etmek için 1 kullanırız. Elbette, sesimizin tam konumunu kontrol etmek için -1 ile arasındaki herhangi bir değeri kullanmakta özgürüz

Sol hoparlörden bip sesi çıkaralım:

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

Şimdi, sağ hoparlörden çalalım:

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

Son olarak, her ikisinden de merkezden oynayalım (varsayılan konum):

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

Şimdi, git ve seslerinin genliğini ve kaymasını değiştirerek eğlen!

2.3-Sentezleri değiştirmek

Şu ana kadar bip sesi çıkarırken az çok eğlendik. Ama büyük ihtimalle basit bip sesleri çıkarmaktan sıkılmaya başladınız. Sonic Pi sadece bunları mı sunuyor? Tabii ki sadece bip sesi ile kalmıyor. Bu bölümde Sonic Pi ın bize sunduğu daha ilgi çekici ses aralığını keşfedeceğiz.

-Synths

Sonic Pi'ın synths adı altında birden fazla farklı enstrümanları vardır. Ayriyeten modeller önceden kaydedilmiş seslerken, synths sizin kontrol etme şeklinize göre yeni sesler oluşturma kapasitesine sahiptir (ki bunları daha sonra bu öğreticide keşfedeceğiz.). Sonic Pi'ın synths'leri çok güçlü ve süratlidir ve onları keşfederken ve oynarken çok eğleneceksiniz. İlk olarak geçerli sentezi kullanmak için nasıl seçmemiz gerektiğini öğrenelim.

-Vızıltılı "saw"lar ve "prophet"ler

Saw wave adında eğlenceli bir ses var- hadi bir deneyelim:

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

```
sleep 0.25
play 62
```

Haydi başka bir ses deneyelim- the *prophet*:

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

Bu iki sesi kombine etmeye ne dersiniz? İlk birini sonra da ötekini çalalım:

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

Şimdi aynı sesleri birlikte çalalım(bunu aralarına duraklama komutu(sleep) koymadan yapabiliriz.):

```
use_synth :tb303
play 38
use_synth :dsaw
play 50
use_synth :prophet
play 57
```

Unutmayın ki `use_synth` komutu sadece onun ardından yazılan `play` komutlarına etki eder. Bunu *büyük bir değiştirme*, yani yeni sesleri hangi sentezde olursa olsun hangisini gösteriyorsa kullanma, olarak düşünebilirsiniz.

-Synths keşfetmek

Sonic Pi'ın size hangi synths oynatabileceğini görmek için alttaki menüde yardım ekranındaki(Örnekler ve Fx arasında) synth seçeneğine bak. 20 den fazla seçenek var. İşte benim birkaç favori synths:

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

2.4-Zarflarla Süreç

Daha önceki bir bölümde seslerimizi ne zaman tetikleyeceğimizi kontrol etmek için uyku komutunu nasıl kullanabileceğimize baktık. Ancak, seslerimizin süresini henüz kontrol edemedik.

Seslerimizin süresini kontrol etmemiz için bize basit ama güçlü bir yöntem sağlamak için Sonic Pi, bir ADSR genlik zarfı nosyonu (bu bölümde daha sonra ADSR'nin ne anlama geldiğini anlatacağız). Bir genlik zarfı, kontrolün iki yararlı yönünü sunar:

- control over the duration of a sound
- control over the amplitude of a sound

-Süreç

Süre, sesin dayandığı uzunluktur. Daha uzun süre, sesi daha uzun süre duyduğunuz anlamına gelir. Sonic Pi'nin seslerinin hepsinin kontrol edilebilir bir genlik zarfı vardır ve bu zarfın toplam süresi sesin süresidir. Bu nedenle, zarfı kontrol ederek süreyi kontrol edersiniz.

-Genlik

ADSR zarfı sadece süreyi kontrol etmekle kalmaz, aynı zamanda sesin genliği üzerinde hassas kontrol sağlar. Tüm duyulabilir sesler sessizleşmeye başlar ve sonunda sessizleşir ve arada sessiz olmayan bir kısım içerir. Zarflar, sesin sessiz olmayan bölümlerinin genliğini kaydırmanıza ve tutmanıza izin verir. Bir gitar amplifikatörünün sesini nasıl açıp kapatacağı konusunda talimatlar vermek gibi. Örneğin, birinden "sessizliğe başlaması, yavaşça tam ses seviyesine yükselmesi, bir süre tutması, ardından hızlı bir şekilde sessizliğe geri dönmesini" isteyebilirsiniz. Sonic Pi, bu davranışı zarfların yardımıyla programlamanıza izin verir.

Özetlemek gerekirse, daha önce gördüğümüz gibi, 0 genliği sessizlik ve 1 genliği normal sestir.
Şimdi sırayla zarfların her bir kısmına bakalım.

-Serbest Bırakma Evresi

Zarfın varsayılan olarak kullanılan tek kısmı yayınlanma süresidir. Bu, synth'in sesinin azalması için geçen zamandır. Tüm sentezlerin 1 serbest bırakılma süresi vardır, bu, varsayılan olarak 1 vuruş süresine sahip oldukları anlamına gelir (varsayılan 60'ta BPM 1 saniyedir):

```
play 70
```

Not 1 saniye boyunca sesli olarak duyulur. Devam et ve zamanı geç :-) Bu daha uzun süren bir süreç için kısa bir bölüm:

```
play 70, release: 1
```

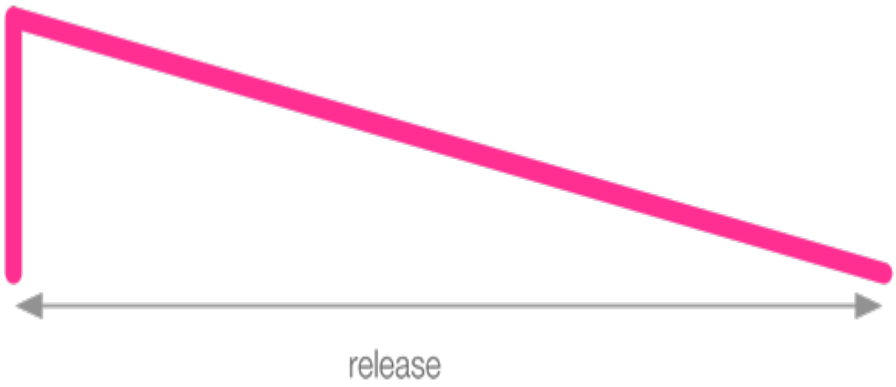
Bunun nasıl aynı şekilde ses çıkardığına dikkat edin (ses bir saniye sürer). Ancak, süreçin değerini değiştirerek süreyi değiştirmek artık çok kolay

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

Çok küçük bir süreç değeri kullanarak synth sesini çok kısa yapabiliriz:

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

Sürecin süresi, serbest bırakma evresi olarak adlandırılır ve varsayılan olarak doğrusal bir geçiştir (yani düz bir çizgi). Aşağıdaki diyagram bu geçişi göstermektedir:



Diyagramın en solundaki dikey çizgi, sesin 0 genlikle başladığını ancak derhal tam genliğe kadar çıktığını gösterir (bu, daha sonra ele alacağımız saldırı aşamasıdır). Tam genliğe geldikten sonra lineer bir şekilde sıfıra doğru hareket eder ve `release:` tarafından belirtilen süreyi alır. Daha uzun salma süreleri daha uzun sentetik az ses çıkışları üretir.

Bu nedenle, bırakma süresini değiştirerek sesinizin süresini değiştirebilirsiniz. Müziğinize serbest bırakma süresi ekleyerek bir oyun oynayın.

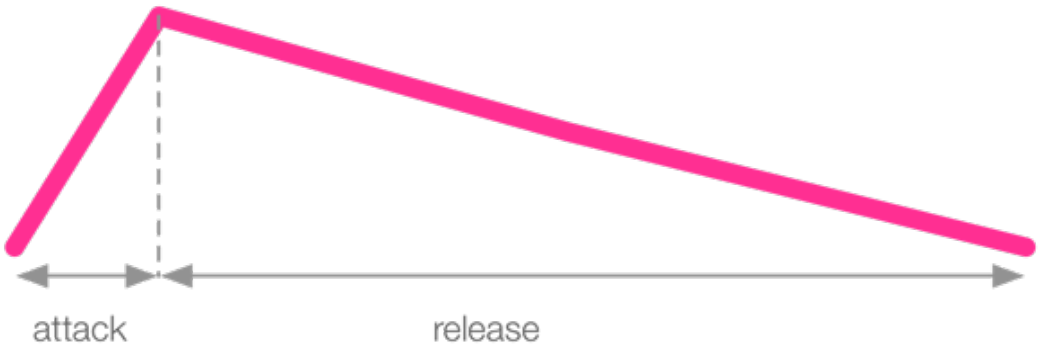
-Atak Evresi

Varsayılan olarak, atak evresi tüm synth'lerde 0'dır, bu da hemen 0 genlikten 1'e geçmeleri anlamına gelir. Bu, synth'a ilk vurmalı bir ses verir. Ancak, sesini kısmak isteyebilirsiniz. Bu durum `attack:` seçeneği ile başarılabilir. Bazı seslerde kısmayı deneyin:

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

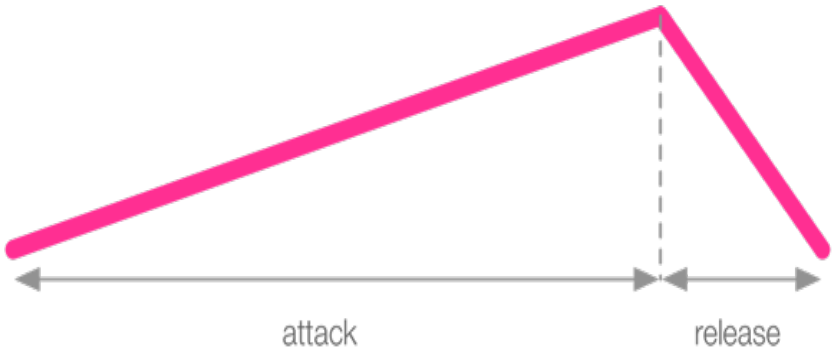
Birçok seçeneği aynı anda kullanabilirsiniz. Örneğin kısa bir atak ve uzun bir serbest bırakmayı deneyin:

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



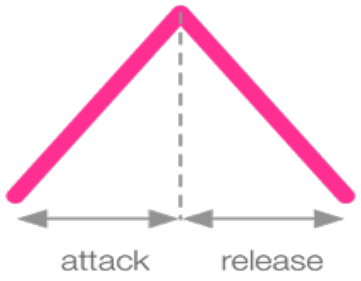
Tabii ki, bunları farklı sırada yapabilirsiniz.

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



Son olarak kısa sesler için kısa ataklara ve serbest bırakma sürelerine sahip olabilirsiniz.

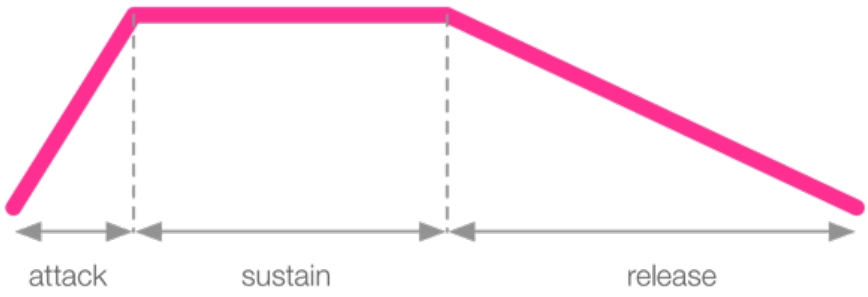
```
play 60, attack: 0.5, release: 0.5
```



-Sürdürme Evresi

Atak ve serbest bırakma zamanlarını belirtmeye ek olarak, sürdürme aşamasını kontrol etmek için bir sürdürme süresi de belirleyebilirsiniz. Bu, sesin atak ve serbest bırakma evreleri arasında maksimum genlikte tutulduğu zamandır.

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

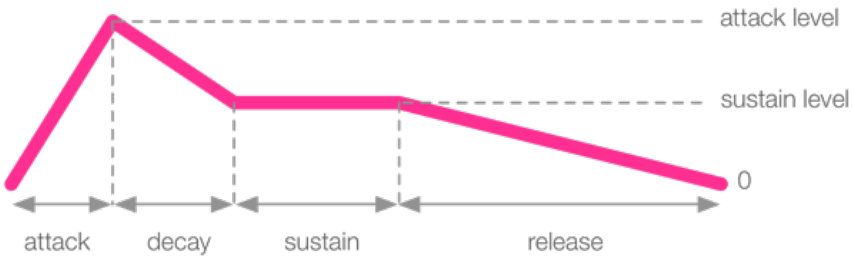


Sürdürme süresi, isteğe bağlı bir serbest bırakma evresine girmeden önce karışımda var olmasını istediğiniz önemli sesler için kullanışlıdır. Tabii ki, hem `attack:` hem de `release:` seçeneklerini 0'a ayarlamak tamamen geçerli ve sadece sürekli olarak solmaya veya sese solmayacak şekilde kullanmak için sürdürme seçeneğini kullanın. Ancak, uyarı olarak, serbest bırakmada 0 değeri seste tıklamalar üretebilir ve 0.2 gibi çok küçük bir değer kullanmak genellikle daha iyidir.

-Zayıflama Evresi

Ekstra kontrol seviyesi için ayrıca bir zayıflama süresi belirleyebilirsiniz. Bu, zarfın atak ve sürdürme evreleri arasına uyan ve genliklerin `attack_level:` konumundan `decay_level:` (açıkça ayarlanmadıkça, `sustain_level:` ayarlanacağı) konumuna düşeceği bir zaman aralığı belirten bir aşamasıdır. Varsayılan olarak, `decay:` seçeneği 0'dır. Fakat hem atak hem de sürdürmenin varsayılanı 1'dir, bu nedenle herhangi bir etkiye sahip olmak için zayıflama zamanları için bunları belirtmeniz gerekir:

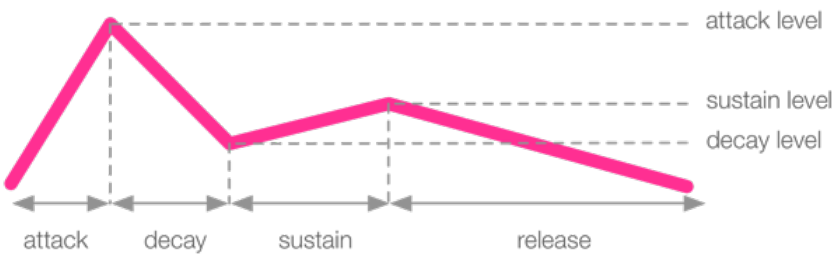
```
play 60, attack: 0.1, attack_level: 1, decay: 0.2,
sustain_level: 0.4, sustain: 1, release: 0.5
```



-Zayıflama Seviyesi

Son bir not olarak, ne kadar `decay_level:` seçeneğinin varsayılanı `sustain_level:` seçeneği ile aynı olsa da, onları zarfların üstündeki tam kontrolü sağlamak için farklı değerlere ayarlayabilirsiniz. Bu size örnekteki gibi zarflar yaratmanızda yardımcı olur.

```
play 60, attack: 0.1, attack_level: 1, decay: 0.2,decay_level: 0.3,
       sustain: 1, sustain_level: 0.4, release: 0.5
```



`decay_level:` seçeneğini `sustain_level:` seçeneğinden daha yükseğe kurmak da mümkün.

-ADSR Zarfları

Özetlemek gerekirse:

Atak: (attack:) 0 genlikten `attack_level` genliğine kadar geçen süre.

Zayıflama: (decay:) `attack_level` genlğinden `decay_level` genliğine kadark geçen süre.

Sürdürme: (sustain:) `decay_level` genliğinden `sustain_level` genliğine kadar geçen süre.

Serbest Bırakma: (release:) `sustain_level` genliğinden 0'a kadarki geçen süre.

Bir sesin süresinin, bu aşamaların her birinin zamanlarının toplamı olduğunu not etmek önemlidir. Bu nedenle, aşağıdaki ses 0.5 + 1 + 2 + 0.5 = 4 vuruş süresine sahip olacaktır:

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

Şimdi zarfları seslerinizde kullanarak keyfini çıkarabilirsiniz.

3- Örnekler

Kendi müziğinizi geliştirmenin bir başka yolu da önceden kaydedilmiş örnek sesleri kullanmaktır. Büyük hip-hop geleneğinde bu sesleri örnekler olarak adlandırıyoruz.
Eğer dışarıya çıkıp mikrofonunuzla yağmurun kanvasa çarpış sesini kaydederseniz bir örnek yaratmış olursunuz.

Sonic Pi kendi bünyesinde barındırdığı 130 farklı örnek ses ile çok eğlenceli şeyler yapmanızı sağlar. Aynı zamanda kendi seslerinizi oluşturmanızı ve değiştirmenizi de sağlar. Hadi biraz bakalım…

3.1- Örnekleri çağırmak

Bip sesini çalmak sadece bir başlangıçtı. Önceden kaydedilmiş örnekleri çağırmak çok eğlencelidir. Deneyin:

```
sample :ambi_lunar_land
```

Sonic pi kendi içerisinde çalabileceğiniz çok fazla örnek içerir. Onları tıpkı notalardaki `play` komutunu kullandığınız gibi kullanabilirsiniz. Eğer birden fazla örnek ses çalmak istiyorsanız onları birbiri ardına yazın:

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

Eğer aralarında zaman boşluğu olmasını istiyorsanız `sleep` komutunu kullanabilirsiniz:

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

Sonic Pi bir sonraki sese başlamadan önceki sesi beklemez, buna dikkat edin. `sleep` komutu yalnızca seslerin *tetiklenmesinin* ayrılmasını sağlar. Bu fırsat, sesleri üst üste bindirerek onları katmanlandırmanıza olanak tanır. Derslerimizin ilerleyen kısımlarında seslerin *sürelerini* kontrol etmeyi öğreneceğiz.

Örnekleri Keşfetmek

Sonic Pi'ın içinde verilen örnekleri keşfetmenin 2 farklı yolu vardır. İlk olarak, buradaki yardım sistemini kullanabilirsiniz. Bu yardım menüsü altındaki örnekler seçeneğine tıkladıktan sonra kategorinizi seçin daha sonra uygun seslerin bir listesi karşınıza çıkacaktır.

Buna alternatif olan başka bir yol da Sonic Pi'ın *otomatik tamamlama mekanizmasıdır*. `sample :ambi_` gibi basit bir başlangıç yazın ve aşağısında beliren örneklerin isimlerinden seçmeye başlayın.

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

Şimdi örnekleri kendi kompozisyonunuzda birbirine karıştırın!

3.2- Örnek Parametreleri: Amp ve Pan

Synth'lerde gördüğümüz gibi seslerimiz parametrelerle kolayca kontrol edilebilir. Örnekler de bu parametre sistemini destekler. `amp:` ve `pan:` arkadaşlarımızı yerlerinde ziyaret edelim.

Örneklerin Ses Katını Yükseltme

Synth'lerde olduğu gibi örnek seslerimizin de genliğini aynı yaklaşımla değiştirebiliriz:

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

Örnekleri Kaydırma

Sonic Pi içerisinde örnek seslerimizde `pan:` parametresini kullanabiliriz. Örnek vermek gerekirse seslerimizin hangi kulağımıza ne kadar yoğunlukta ve ne kadar süre geleceğini ayarlayabiliriz:

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

Son olarak bazı synth varsayımlarını `use_synth_defaults` ile ayarlarsanız bunların örnek ses tarafından yok sayılacağını da unutmamanız gerekir.

3.3- Örnek Sesleri Germe İşlemleri

Artık bir müzik oluşturmak için çeşitli synth'ler ve örnek sesler çalabildiğimize göre, bunları yaratıcı, ilginç ve benzersiz bir biçimde nasıl kullanacağımızı öğrenme zamanı da geldi. Öncelikle *stretch* ve *squash* örneklerimizi keşfedelim.

Örnek Seslerin Temsili

Örnek sesler onları yeniden üretmek için hoparlör konisinin hareketlerini temsil eden numaralarla önceden kaydedilmiş seslerdir. Hoparlör konisi içeri ve dışarı belirli sürelerde hareket edebilir ve sayılar koninin ne kadar içeri ya da dışarıda bulunacağını denetler. Kayıtlı bir sesi bozulmaksızın çoğaltmak için örnek sesimizin saniyede binlerce numara depolaması gereklidir! İnanılmaz! Sonic Pi bu numara listesini alarak doğru sıralamada ve yoğunlukta hoparlörünüzü ses çıkartması için besler. Bununla birlikte sesi değiştirmek amacıyla hoparlörün beslenme hızını değiştirmek de eğlenceli olabilir.

Oranların Değiştirilmesi

Sonic Pi'ın ambiyans seslerinden biri olan `:ambi_choir` sesini ele alalım. Varsayılan değerleri değiştirmek için `rate:` özelliğini kullanabiliriz:

```
sample :ambi_choir, rate: 1

sample :ambi_choir, rate: 0.5
```

Oh, neler oldu öyle? Hem örnek sesimizin oynaması 2 kat daha uzun sürdü hem de sesimizin oktavı düştü. Haydi bunu biraz daha detaylı bir biçimde inceleyelim.

Hadi Uzatalım!

Amen Break üzerinde değişiklik yapılması oldukça zevkli bir fonksiyondur.

```
sample :loop_amen
```

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

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

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

Ah, sonuncu örnekte tersten mi çaldı o? Şimdi Farklı şeyler denemeye devam edin ve kendinizi müziğe kaptırın!

3.4- Zarflanmış Örnek Sesler

Bir ADSR zarfı kullanarak bir örnek sesin genliğini ve uzunluğunu değiştirmek mümkündür. Ancak, synth'lerde bulunan ADSR zarfları normalden biraz daha farklı çalışır.

Amen Zarfı

```
sample :loop_amen
```

Haydi şimdi `attack:` parametresini kullanalım:

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

Daha kısa bri geçiş için, daha kısa bir attack değeri seçin

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

Otomatik Sürdürme

Sustain parametresi sayesinde kullandığımız seslerin uzunluğunu yani çalma sürelerini ayarlayabiliriz.

Soluk Çıkışlar

Burada hem `attack:` hem de `release:` parametrelerini beraber kullanabiliriz.

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

Buradaki değerler ile oynayarak müziğinizi keşfetmeye devam edin!

3.5-Kısmi Örnekler

Bu kısımda Sonic Pi'In modellerini daha da çok keşfedeceğiz. Nasıl kullanıldığını göstermiştik. Örneğin:

```
sample :loop_amen
```

Daha sonra hızlarını nasıl değiştirdiğimiz gördük:

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

Daha hafifletmek için:

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

Aynı zamanda başlangıcının `sustain:` komutu ile nasıl başlayacağının yanında attack ve release komutlarını ayarlamak için de kullanılabilir.

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

Fakat bunun yanında modellerin nerden başlayacağını söyleyebilsek güzel olmaz mıydı?

Başlangıç Noktası Seçme

0 'ın başlangıç 1'in son ve 0.5'in yarı yol olduğu, 0 ve 1 arasında sıradan bir başlangıç noktası seçmek mümkün. Alttaki kodda olduğu gibi:

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

,
Son çeyreğine nerdesiniz:

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

Son Noktayı Seçmek

Benzer bir şekilde 0 ile 1 arasında sıradan bir bitiş noktası seçilebilir:

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

Başlangıç ve Bitiş Belirleme

Modelin sadece bir kısmı için:

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

Peki başlangıç noktasının bitişinden daha büyük seçersek ne olur? Tersten çalmaya başlar:

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

Hız ile Düzenleme

Modellerin sadece bir kısmını daha yavaş ya da daha hızlı yapmak ister misin? Şunu bir dene:

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

Zarflar ile Düzenleme

Sonunda bunların hepsini ADSR zarfları ile daha ilgi çekici bir hale getirebiliriz.

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

3.6-Harici Modeller

Eğer Sonic Pi içerisinde bulunan modeller yetersiz geliyorsa kendi kayıtlarınızı da dahil etmek mümkün. Bunun için parçanızın taşınılabilirliğini kontrol etmek durumundayız.

-Taşınabilirlik

Eğer eserinizi sadece Sonic Pi bünyesindeki modeller ile yaparsanız bunu arkadaşlarınızla dahi paylaşıp keyfini çıkarabilirsiniz.

Fakat, kendi parça modelleriniz kullanırsanız, arkadaşlarınızla bu parça modeli de paylaşıp eğlenceye devam edebilirsiniz. Göz önünde bulundurulması gereken bir özellik.

-Yerel Modeller

Şu şekilde WAV, AIFF veya FLAC tipi dosyaları parça modeller olarak kullanabilirsiniz:

```
# Raspberry Pi, Mac, Linux
sample "/Users/sam/Desktop/my-sound.wav"
# Windows
sample "C:/Users/sam/Desktop/my-sound.wav"
```

Sonic Pi otomatik olarak modeli yükleyi çalacaktır. Aynı zamanda standart komutları kullanarak modeli düzenleyebilirsiniz.

```
# Raspberry Pi, Mac, Linux
sample "/Users/sam/Desktop/my-sound.wav", rate: 0.5, amp: 0.3
# Windows
sample "C:/Users/sam/Desktop/my-sound.wav", rate: 0.5, amp: 0.3
```

3.7-Model Paketleri

NOT: BU kısım Sonic Pi ile uyumlu parça modelleri yaratanlar ve satın alanlar için düzenlenmiştir. Sonic Pi bünyesindeki modeller yeterliyse bu kısmı geçebilirsiniz.

Büyük model dosyaları ile çalışmak biraz sıkıntılı bir durum yaratabilir. Örneğin:

```
/path/to/my/samples/
```

Dosyasında şu modeller olsun:

- 100_A#_melody1.wav
- 100_A#_melody2.wav
- 100_A#_melody3.wav
- 120_A#_melody4.wav
- 120_Bb_guit1.wav
- 120_Bb_piano1.wav

Piyano modelini kullanmak için şu yolu deneyebilirsin:

```
sample "/path/to/my/samples/120_Bb_piano1.wav"
```

Sonra gitar modeli için:

```
sample "/path/to/my/samples/120_Bb_guit.wav"
```

Burada dosya ve model isminiz bildiğimiz bir durum var, ya bunları bilmeseydik.

-Model Paketlerini İndexleme

Eğer ilk modeli duymak istersen:

```
sample "/path/to/my/samples/", 0
```

Değişken kullanarak daha basit bir yol izlenebilir.

```
samps = "/path/to/my/samples/"
sample samps, 0
```

Şimdi ikinci modeli duymak için:

```
samps = "/path/to/my/samples/"
sample samps, 1
```

Artık modellerin isimlerine gerek duymadığımızı fark ettin mi? Hatta dosyadaki model sayısından daha fazla bir sayı girersen de içlerinden birini çalmaya devam edecektir.

-Model Paketleri Filtreleme

Genellikle indeksleme yeterli olduğu halde modelleri düzenlemek ve sıralamak için daha çok güce ihtiyacımız var. Tesadüfi, birçok model paketleri yararlı bilgiler içeriyor. Örneğin:

- `100_A#_melody1.wav`
- `100_A#_melody2.wav`
- `100_A#_melody3.wav`
- `120_A#_melody4.wav`
- `120_Bb_guit1.wav`
- `120_Bb_piano1.wav`
-

Bu isimlerde birden fazla bilgi olduğunu fark ettin mi? 100 ve 120 sayıları modellerin bpm değerini göstermekte. Aynı zamanda A# ve Bb notalarında çalındığını göstermekte.

Bu bilgileri sadece istediklerimizi çalmak amacıyla kullanabiliriz. Örneğin:

```
samps = "/path/to/my/samples/"
sample samps, "120"
```

Bu kod bizi ilk eşleşen modele götürecek yani gitar1 modeline.

```
samps = "/path/to/my/samples/"
sample samps, "120", 1
```

Bu kod ise bize 120 bpm'deki 2. Eşleşmeyi sağlayacak.

```
samps = "/path/to/my/samples/"
sample samps, "120", "A#"
```

Hatta az sonra görüleceği üzere birden fazla filtre kullanılabilir.

```
samps = "/path/to/my/samples/"
sample samps, "120", "Bb", 1, lpf: 70, amp: 2
```

-Kaynaklar

Model filtresi pre-arg sistemi 2 tür bilgiyi algılıyor: kaynaklar ve filtreler. Kaynaklar potansiyel adaylar için kullanılan bilgi olarak gösterilebilir. Bir kaynak 2 formda olabilir.

1. "/path/to/samples"– bizi geçerli dosyaya ulaştıran bir yol
2. "/path/to/samples/foo.wav" – bizi geçerli modeler ulaştıran bir yol

`sample` komutu önce tüm kaynakları toplayarak bir muhtemel aday listesi oluşturuyor. Bu liste önce geçerli yolları ekliyor daha sonra geçerli `.flac`, `.aif`, `.aiff`, `.wav`, `.wave` dosyalarını ekliyor. Örneğin:

```
samps = "/path/to/my/samples/"
samps2 = "/path/to/my/samples2/"
path = "/path/to/my/samples3/foo.wav"

sample samps, samps2, path, 0
```

Varsayılan olarak, yalnızca bir dizindeki örnek dosyalar aday listesine toplanır. Bazen aramak ve filtrelemek istediğiniz birçok iç içe geçmiş model klasörünüz olabilir. Bu nedenle, yolun sonuna ** ekleyerek, belirli bir klasörün tüm alt klasörlerindeki tüm örnekler için özyinelemeli bir arama yapabilirsiniz.

```
samps = "/path/to/nested/samples/**"
sample samps, 0
```

Son olarak, ilk önce kaynakların gitmesi gerektiğini unutmayın. Herhangi bir kaynak belirtilmezse, yerleşik örneklem grubu birlikte çalışılacak adayların varsayılan listesi olarak seçilecektir.

-Filtreler

Bir aday listeniz varsa şu filtreleme türlerini kullanabilirsiniz:

`"foo"` Dizeler, dosya adındaki alt dize oluşumuna (eksi dizin yolu ve uzantı) filtre uygular.

`/fo[oO]/`Normal İfadeler, dosya adının (eksi dizin yolu ve uzantısı) desen eşleşmesine filtre uygular.

`:foo` - Anahtar kelimeler adayları, anahtar kelimenin dosya adıyla doğrudan eşleşip eşleşmediği konusunda filtreleyecektir (eksi dizin yolu ve uzantısı).

`lambda{|a| ... }`- Bir argüman olan işlemler aday filtre veya jeneratör işlevi olarak değerlendirilir. Mevcut adayların listesinden geçirilecek ve yeni bir aday listesi (örnek dosyalara uygulanan geçerli yolların bir listesi) geri göndermesi gerekir.

`1` - Rakamlar bu indekse sahip adayı seçecektir (gerekirse bir halka gibi sarma).

-Karışımlar

Son olarak, kaynak veya filtre yerleştirebileceğiniz her yerde listeleri kullanabilirsiniz. Liste otomatik olarak düzleştirilecek ve içerik düzenli kaynaklar ve filtreler olarak değerlendirilecektir. Bu nedenle aşağıdaki `sample` çağrıları anlamsal olarak eşdeğerdir:

```
sample "/path/to/dir", "100", "C#"
sample ["/path/to/dir", "100", "C#"]
sample "/path/to/dir", ["100", "C#"]
sample ["/path/to/dir", ["100", ["C#"]]]
```

-Sarma

Bu, model paketleri işlemek ve kullanmak için gerçek güce ihtiyaç duyan insanlar için gelişmiş bir bölümdü. Bu bölümün çoğu çok fazla anlam ifade etmiyorsa, endişelenmeyin. Henüz bu işlevselliğin hiçbirine ihtiyacınız yok. Ancak, ne zaman ihtiyacınız olduğunu bilirsiniz ve büyük örneklem dizinleriyle çalışmaya başladığınızda geri dönüp tekrar okuyabilirsiniz.