Ada Lovelace Day

An Introduction to Sonic Pi

Ada was the first to express the potential for computers outside mathematics and her theory of how computer sequenced music could be achieved was incredibly accurate. 115 years later in 1951, the University of Manchester's Ferranti Mark 1 computer performed what is believed to be the very first computer score. The program, which was a composition of *Blah Blah Black Sheep* was written by Christopher Strachey, a maths master at Harrow and a friend of computing legend Alan Turing. So, to celebrate ALD15, let's create some music by writing computer code with Sonic Pi - a live coding environment based on Ruby, originally designed to support both computing and music lessons in schools, developed by Sam Aaron in the University of Cambridge Computer Lab. Sonic Pi was designed to run on Raspberry Pi computers but it's also available for free on both OS and Windows platforms. It's simple to use, intuitive and lots of fun!

Part 1: How to play notes, create loops and adjust bpm

Let's start with learning how to tell Sonic Pi that we want to play a note.

Sonic Pi is familiar with MIDI numbers and also musical notation. So, if you're familiar with sheet music or notes, you'll be able to start right away. Alternatively, MIDI is a useful way to compose and is a useful tool for quickly testing your notes and adjusting them by lowering their value (making your note lower) or increasing it, (making the pitch higher). SonicPi isn't case sensitive, so don't worry about lower & uppercase text.

Underneath the '*#Welcome to Sonic Pi*' text, try typing in:

```
Play 65
```

and press the **Run** button - Did you hear it?

Now, try lowering the value. This should make produce a lower note

```
Play 60
```

And, by increasing the number, we hear a higher note

```
Play 70
```

How cool is that? Super simple.

Let's try creating a sequence of notes. Type in the following MIDI values (or copy and paste):

```
play 65
play 60
play 68
play 65
play 60
play 56
```

Whoah, that didn't sound quite right. So far, Sonic Pi knows to play the notes, but rather than playing them in sequence, it's playing them all at once. It still sounds kinda cool though, right? This is how we write chords.

If we want to Sonic Pi to play each note in a sequence, we have to write a command that tells the software to take a break. Try typing in 'sleep' underneath each note, like this:

```
play 65
sleep 1
play 60
sleep 1
play 68
sleep 1
play 65
sleep 1
play 60
sleep 1
play 56
sleep 1
```

You'll notice that the pace of the sequence is quite slow. Sonic Pi's default BPM (beats per minute) is 60. We can alter the time between each notice by adjusting the sleep value (sleep 0.3 for example), but a much simpler way is to adjust the BPM, which means we can keep to a rule that's based on bars (sleep 1, being one bar).

To alter the BPM of your track, type this at the top of your code and press play:

```
use_bpm 140
```

You should now be hearing a faster sequence of notes being played.

Let's write a slightly longer sequence of notes. Try this (note the highlighted yellow text):

```
use_bpm 140
  play 65
  sleep 1
  play 60
  sleep 1
  play 68
  sleep 1
  play 65
  sleep 1
  play 60
  sleep 1
  play 56
  sleep 1
  play 53
  sleep 1
  play 48
  sleep 1
  play 49
  sleep 1
  play 53
  sleep 1
  play 60
  sleep 1
  play 65
  sleep 1
  play 63
  sleep 1
  play 65
  sleep 1
  play 60
  sleep 1
  play 68
  play 65
```

These two notes should play simultaneously, (G4 and F4), creating a chord.

Let's try looping this sequence. We could copy & paste this code over and over, but that would make our code gigantic and difficult to pinpoint any changes we want to make. A much easier way is to wrap our sequence of notes with a command that tells it to play a certain number of times, and a command that tells it to stop. Type this at the top of your code:

```
2.times do
```

This tells Sonic Pi to play the sequence below two times

To indicate where the loops ends, paste this at the bottom of your code

```
end
```

You should now have something like this:

```
2.times do
use_bpm 140
play 65
  sleep 1
  play 60
  sleep 1
  play 68
  sleep 1
  play 65
  sleep 1
  play 60
  sleep 1
  play 56
  sleep 1
  play 53
  sleep 1
  play 48
  sleep 1
  play 49
  sleep 1
  play 53
  sleep 1
  play 60
  sleep 1
  play 65
  sleep 1
  play 63
  sleep 1
  play 65
  sleep 1
  play 60
  sleep 1
  play 68
  play 65
  sleep 1
end
```

So far, we've learned how to create notes, how to control the phrasing (silent gaps between each note), how to loop your sequence a certain number of times and how to alter the BPM of your track. Sweet.

Part 2: Percussion, effects and samples

Sonic Pi offers many ways to write percussion into our code. We can choose drum sounds, apply effects, such as modulation, reverb or compression and we can also trigger drum samples. Let's look at how to create a simple drum kit, coniststing of a kick drum, snare and hi-hat.

Let's begin by creating a loop. Select a different window to work in by choosing a different Buffer. We're currently on Buffer 0. Let's move to Buffer 1. Replace '# Welcome to Sonic Pi v2.6' with '# My drums'.
Let's stick with 140bpm. Type the following block into Sonic Pi:

```
use_bpm 140
live_loop :hh do
  sample :drum_cymbal_closed
  sleep 0.5
end
```

live_loop is a useful way to create an endless loop. *:hihat* is the name we've given this block. It can be named anything, but its purpose is for us to quickly identify what instrument it is. At the moment our hi-hat (*sample :drum_cymbal_closed*) is playing at 140bpm and playing twice every bar. Let's try adding an effect by updating your code to look like this:

```
use_bpm 140
live_loop:hh do
  with_fx :echo, mix: 0.6 do
   1.times do
      sample :drum_cymbal_closed
      sleep 0.5
    end
  end
end
```

You'll now hear that the echo effect that we have applied is making the hi-hat repeat.

Let's now add a snare. Update your code to look like this (note the highlighted yellow text) and hit **Run**:

```
use_bpm 140
live_loop:hh do
  with_fx :echo, mix: 0.6 do
   1.times do
     sample :drum_cymbal_closed
     sleep 0.5
   end
  end
end
live_loop(:snare) do
  with_fx :distortion, mix: 1 do
    1.times do
      sleep 1
      sample :elec_mid_snare
      sleep 1
    end
  end
end
```

We've added some slight distortion to our snare sound with a value of '1'. You'll also notice that the sleep value has been added before the instrument is called, and then again afterward. This tells Sonic Pi to pause the snare hit on the first bar, and play on the second. Now we just need a kick drum to fill play once at the start of every bar. Update your code to look like this note the green highlighted text.

```
use_bpm 140
live_loop:hh do
  with_fx :echo, mix: 0.6 do
    1.times do
      sample :drum_cymbal_closed
      sleep 0.5
    end
  end
end
live_loop:snare do
  with_fx :distortion, mix: 1 do
    1.times do
      sleep 1
      sample :elec_mid_snare
      sleep 1
    end
  end
end
live_loop:bass do
  with_fx :level, amp: 1 do
    1.times do
      sample :bd_haus
      sleep 2
    end
  end
end
```

Hit **Run** and feel the funk.

Let's paste our drums into our other composition, which should still be sitting in Buffer 0. Here is how it should look:

```
use_bpm 140
live_loop:hh do
  with_fx :echo, mix: 0.6 do
    4.times do
      sample :drum_cymbal_closed
      sleep 0.5
    end
  end
end
live_loop:snare do
  with_fx :distortion, mix: 1 do
    1.times do
      sleep 1
      sample :elec_mid_snare
      sleep 1
    end
  end
end
live_loop:bass do
  with_fx :level, amp: 1 do
    1.times do
      sample :bd_haus
      sleep 2
    end
  end
end
live_loop:melody do
  2.times do
    use_synth :subpulse
    use_bpm 140
    play 65
    sleep 1
    play 60
    sleep 1
    play 68
    sleep 1
    play 65
    sleep 1
    play 60
    sleep 1
    play 56
    sleep 1
    play 53
    sleep 1
    play 48
    sleep 1
    play 49
    sleep 1
    play 53
    sleep 1
    play 60
    sleep 1
    play 65
    sleep 1
    play 63
    sleep 1
    play 65
    sleep 1
    play 60
    sleep 1
    play 68
    play 65
    sleep 1
  end
end
```

We can also use drum samples, which do all the hard work, but are stuck at a preset BPM.

Move to Buffer 2.

In the following example, I've used a drum & bass loop *'sample :loop_amen_full'*

If we want to tell Sonic Pi to play this sample twice and loop at correct point, we type this

```
2.times do
  sample :loop_amen_full
  use_bpm 140
end
```

Here's how to play the D&B loop with our melody. Our sample is highlighted in green and our original melody, which you can take from buffer 0 is in yellow. Type this and hit **Run**

```
2.times do
  sample :loop_amen_full
  use_bpm 140
  use_synth :subpulse
  play 65
  sleep 1
  play 60
  sleep 1
  play 68
  sleep 1
  play 65
  sleep 1
  play 60
  sleep 1
  play 56
  sleep 1
  play 53
  sleep 1
  play 48
  sleep 1
  play 49
  sleep 1
  play 53
  sleep 1
  play 60
  sleep 1
  play 65
  sleep 1
  play 63
  sleep 1
  play 65
  sleep 1
  play 60
  sleep 1
  play 68
  play 65
  sleep 1
end
```

Part 3: Share your music with us!

Hopefully, this simple introduction to Sonic Pi has given you a feel for how you can write music. As part of ALD15, we will be sharing our Sonic Pi tracks via our very own SoundCloud Account. If you'd like to share your own track with us, feel free to save your composition (with your **own name** as the filename) and email your text file to **interactive-content@ed.ac.uk**. Alternatively, you can share it with us by popping it on our USB stick, which we can pass around the room.

THE UNIVERSITY of EDINBURGH