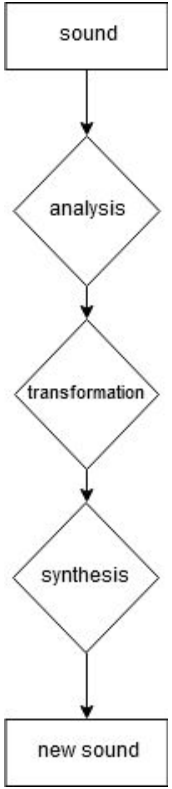
Resynthesis

Resynthesis is the process which involves analyzing the structure of a sampled sound into a data description of an audio signal, which can be seen as a blueprint, and trying to recreate/rebuild that structure from this blueprint, generally with modifications such as pitch shifting, timbral modifications, time warping etc. The basic idea is you start with a real sound, extract data detailing its spectras changes over time and model the frequency and waveform of each harmonic to create a resynthesized version. The first synthesizers to use resynthesis were vocoders.



Sonic Pi

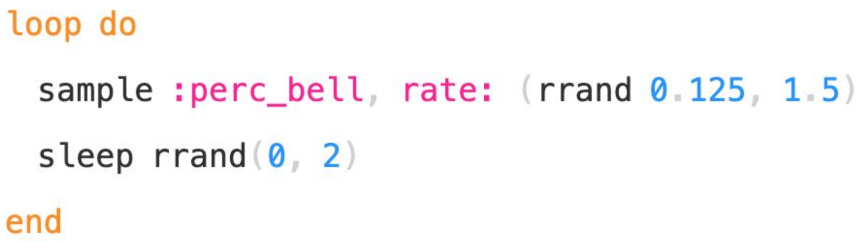
Welcome to the future of music.

What is Sonic Pi

- an open source programming environment developed by Dr Sam Aaron
- designed to explore and teach programming concepts through the process of creating new sounds
- a scheme of work which emphasises the importance of creativity in the learning process
- specifically targeted towards introductory KS3 Computer Science
- It has been successfully trialled at a number of schools
- it was written in Ruby, Erlang, Clojure, C ++, and Qt

How does it work?

- It works on the Raspberry Pi
  - It comes pre-installed with the newest Raspbian Stretch
- Sonic Pi is based on Ruby
  - **Ruby** is an interpreted, high-level, general-purpose programming **language**. It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan. **Ruby** is dynamically typed and uses garbage collection.



```
loop do
  sample :perc_bell, rate: (rrand 0.125, 1.5)
  sleep rrand(0, 2)
end
```

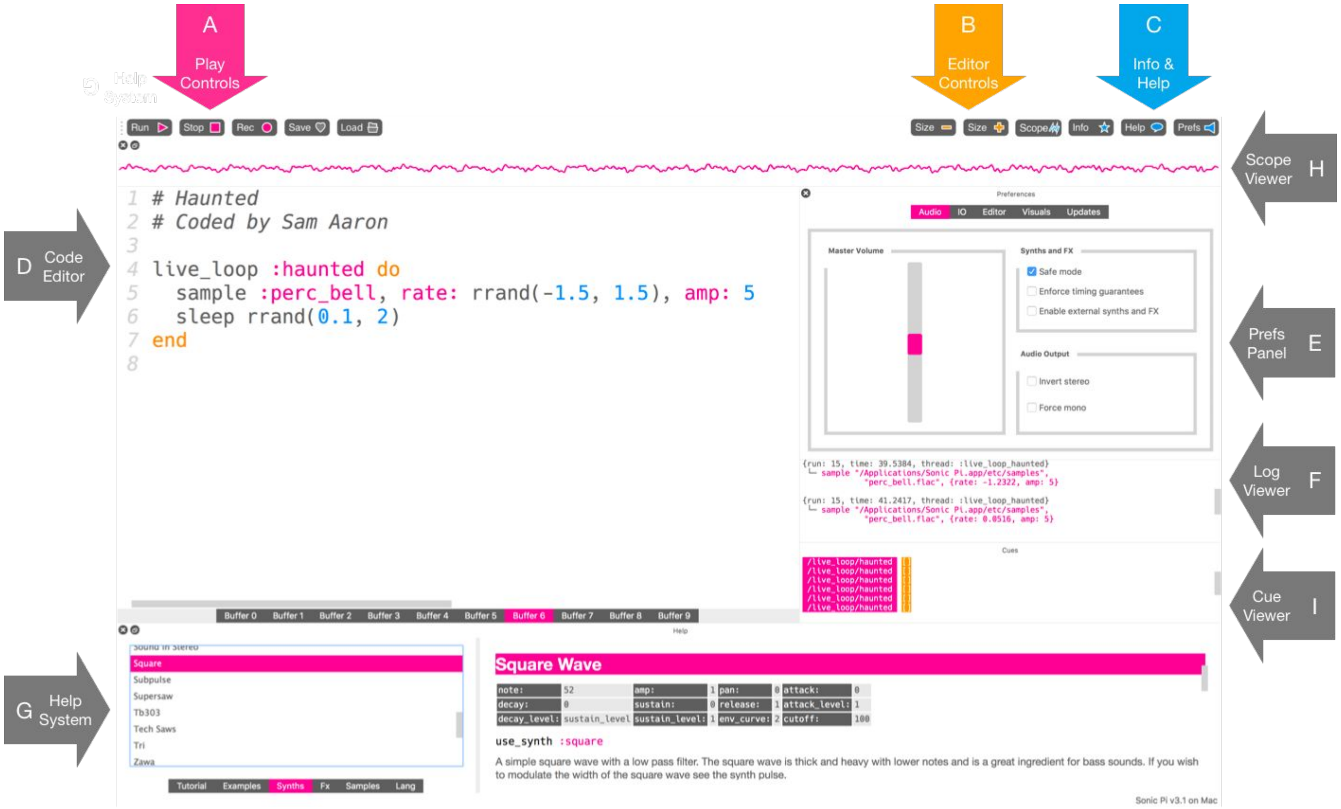
Pre-installed on Raspbian Stretch

Classroom ready

Sonic Pi designed, implemented and developed with extensive classroom trials in close collaboration with teachers, such as:

- What a Raspberry Pi is, what its main features are, and how to set up and use one
- How to write text-based code to produce a music track
- Computer Science concepts:
  - Sequencing
  - Debugging
  - Iteration
  - Conditionals
  - Data Structures
  - Algorithms
  - Functions
  - Concurrency

Interface



```
# Haunted
# Coded by Sam Aaron

live_loop :haunted do
  sample :perc_bell, rate: rrand(-1.5, 1.5), amp: 5
  sleep rrand(0.1, 2)
end
```

☑ Safe mode

□ Enforce timing guarantees

□ Enable external synths and FX

□ Invert stereo

□ Force mono

```
{run: 15, time: 39.5384, thread: :live_loop_haunted}
 └─ sample "/Applications/Sonic Pi.app/etc/samples",
            "perc_bell.flac", {rate: -1.2322, amp: 5}

{run: 15, time: 41.2417, thread: :live_loop_haunted}
 └─ sample "/Applications/Sonic Pi.app/etc/samples",
            "perc_bell.flac", {rate: 0.0516, amp: 5}
```

```
/live_loop/haunted
/live_loop/haunted
/live_loop/haunted
/live_loop/haunted
/live_loop/haunted
/live_loop/haunted
```

Square Wave

note:	52	amp:	1	pan:	0	attack:	0
decay:	0	sustain:	0	release:	1	attack_level:	1
decay_level:	sustain_level	sustain_level:	1	env_curve:	2	cutoff:	100

```
use_synth :square
```

A simple square wave with a low pass filter. The square wave is thick and heavy with lower notes and is a great ingredient for bass sounds. If you wish to modulate the width of the square wave see the synth pulse.

Live Coding

Thanks to its use of the SuperCollider synthesis engine and accurate timing model, Sonic Pi is used for live coding of music and other forms of algorithmic music performance and production, so that the code can be manipulated, changed and adapted in real time; this means coders can perform their code rather than playing pre-written programs.

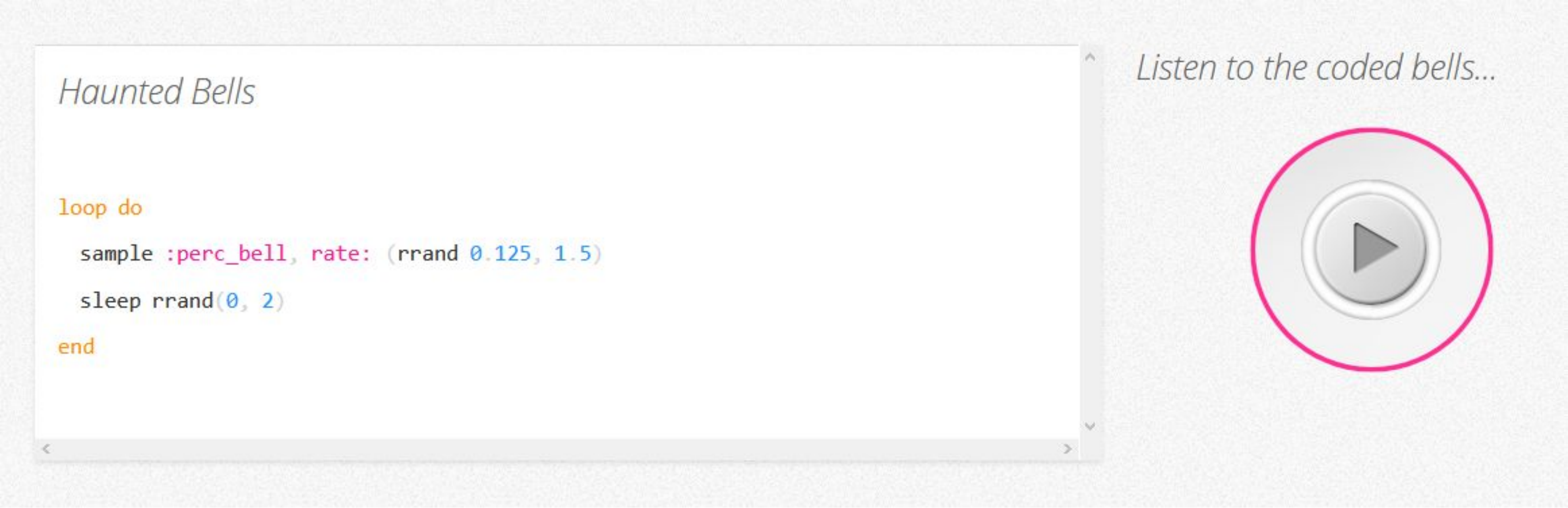
Sam Aaron also uses the software to perform live with his band.

https://www.youtube.com/watch?v=oDHumac84aw

https://www.youtube.com/watch?v=zdc94RAX9UY

Code Examples

https://sonic-pi.net/



Haunted Bells

```
loop do
  sample :perc_bell, rate: (rrand 0.125, 1.5)
  sleep rrand(0, 2)
end
```