KS3/4/5

Sonic Pi for music teachers

by David Ashworth

David Ashworth is a freelance education consultant, specialising in music technology. He is project leader for www.teachingmusic.org.uk and he has been involved at a national level in most of the major music initiatives in recent years.

INTRODUCTION

Sonic Pi is an easy-to-use, freely downloadable software application. It is aimed at those new to computer programming, but powerful enough to generate really sophisticated music.

Part of the stimulus for its development was to stake a claim for music in the new computing curriculum for schools, with its emphasis on computer programming as an important digital skill. Students are already finding it a very engaging way to make music, and also one that makes strong connections with their musical cultures and helps them develop broader and deeper levels of musical understanding.

The program comes with tutorials and worked examples, which fully demonstrate the application's possibilities. Users can generate melodic lines, chords and sequences, assign them to different synthesisers, edit the timbral profile of synthesised sounds, add various looping and chance structures, import sampled sounds and process these sounds using a range of effects. The coding makes it clear what is happening musically at each stage, so that students get good insights into how sounds can be organised musically.

In this resource, I provide some pre-programmed templates for musical activities. These can be set up to encourage exploration of a range of musical concepts such as chord progressions, theme and variations, building melodies and so on. Using the 'live looping' function allows students to make adjustments to various musical parameters as the music is playing. Using 'what if'-type questions engages learners to try out ideas, make musical decisions and develop programming skills.

Although this is a relatively new approach to making music, the fundamentals of good music making still apply. We still need to think about how good melodies develop organically, and how rhythm patterns are structured and fit together in ways that listeners can appreciate and recognise. The priority still has to be making music that sounds good – this is not just about clever coding. So always ask the question – is it musical? Will a listener find in this music some musical reference points they can connect with?

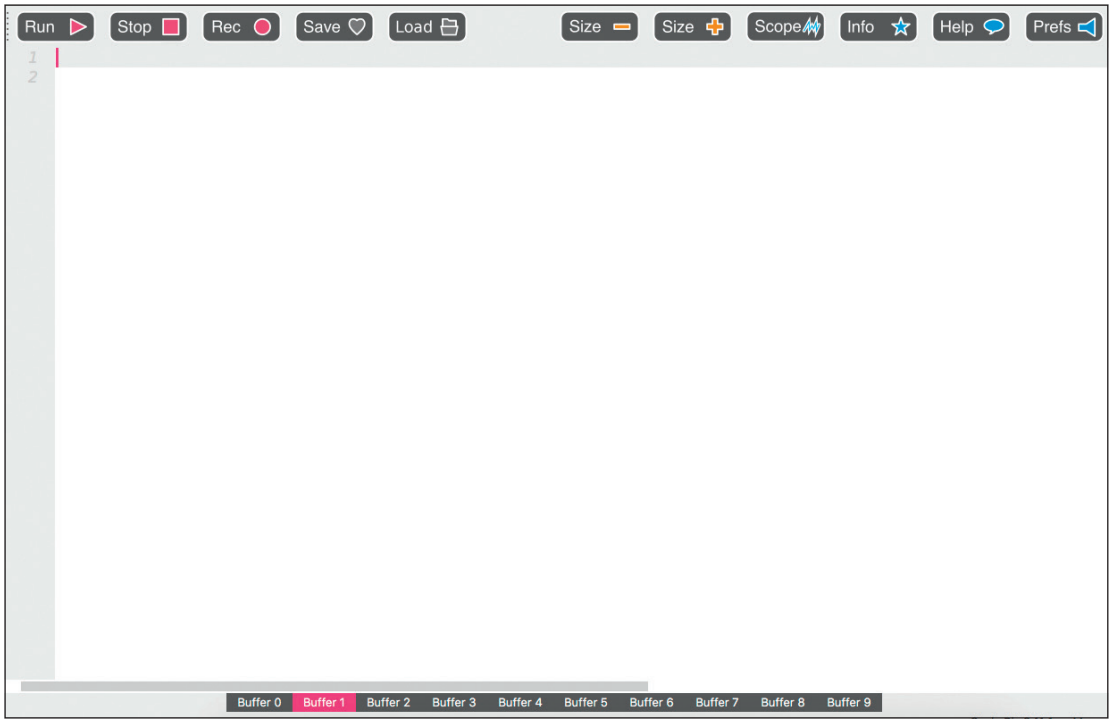
This resource has been written specifically to support the classroom music teacher in exploring new, innovative ways of delivering the music curriculum. The activities are aimed at older students, but teachers will find that they can easily be adapted to suit a wide range of ages, abilities, levels and teaching contexts.

SONIC PI: THE INTERFACE

Download the Sonic Pi program using this link.

The software has been shown to run satisfactorily on school network systems, but you may want to consult your ICT support team to advise on the best ways of doing this.

This is a screenshot of the interface:



A quick explanation of some of the functions:

- Run = play the file.
- Stop = stop the playback.
- Rec = record the session as a wav file.
- Save = save the coding as a text file.
- Load = load previously saved coding.
- Size to zoom in/out on coding script.
- Help to access online tutorials, synths, samples and example files.
- Buffers at the bottom of the screen to access different coding sequences.

FIRST STEPS

It's time to start coding! First move the cursor down to line five. (We'll leave the top four lines blank, so that we can insert additional coding at a later stage.)

Now type *play 60* – then hit **Run**.

You should hear a single note, which anyone with perfect pitch will recognise as a middle C.

On the next line, type *play 64* – then hit **Run**.

You should now hear two notes being played together – middle C and a note four semitones higher – E.

On the next line, type *play 67* – then hit **Run**.

You should now hear a C major triad.

We could continue to stack notes in this way to build up even richer chords, but in order to separate the notes to begin forming melodies, we need to use another command – **sleep**.

The sleep command will generate a space between triggering notes in a sequence. For example, this coding will provide a duration of one second between the start of each note:

```
play 60
sleep 1
play 62
sleep 1
play 64
sleep 1
```

ACTIVITY 1

Create a melodic line of eight to 12 notes by using ***play*** and ***sleep*** commands. Choose from the following values for play:

60 62 64 65 67 69 71 72

These notes, played in order, equate to a scale of C major. So **play** values of 60 62 64 60 60 62 64 60 would give the start of 'Frère Jacques'.

ACTIVITY 2

The numbers given for pitch values above represent the standard MIDI numbers associated with given notes. This chart shows how they 'translate':

60	62	64	65	67	69	71	72
c4	d4	e4	f4	g4	a4	b4	c5

It's also possible to use the pitch letter (plus the number that specifies the octave) in coding melodies.

So ***play :d4*** would give the note D. It's important to prefix the semicolon when using letter names of pitches in coding.

Now create a short melody using letter names, rather than MIDI values.

Programming rhythm patterns

So far, we've only created melodies using notes of equal duration. By modifying sleep values, we can create rhythmic variety.

So if we take ***sleep 1*** to represent a crotchet, then *sleep 0.5* would give us a quaver. Similarly *sleep 2* would be a minim – and so on.

So to generate a crotchet-quaver-quaver pattern on a repeated note of C:

```
play 60
sleep 1
play 60
sleep 0.5
play 60
sleep 0.5
play 60
sleep 1
play 60
sleep 0.5
play 60
sleep 0.5
play 60
sleep 1
play 60
sleep 0.5
play 60
sleep 0.5
play 60
sleep 2
```

Notice we end this phrase on a *sleep 2*, which is, of course, a minim.

ACTIVITY 3

Copy/paste the coding above into a buffer in Sonic Pi. Now edit some of the play pitch values to create a musically interesting phrase in the key of C major. Use the chart given in Activity 2 above for a range of possible *play* values.

ACTIVITY 4

Sooner or later, a decision has to be made as to whether to use note names or MIDI values when coding pitches in a sequence. Both have advantages and disadvantages. Numbers are quicker to input (no colon needed) and show directly the intervals in semitones between notes. Letter names are easier to relate to positions in a given key.

It's possible to mix pitch names and MIDI numbers freely in the same sequence, but that would probably only serve to confuse things further.

Extend the C major chart in both directions to give a greater range of notes.

					59	60	62	64	65	67	69	71	72	74				
					b3	c4	d4	e4	f4	g4	a4	b4	c5	d5				

Now work out the MIDI notes for a scale of D major:

d4	e4	fs4	g4	a4	b4	cs5	d5

(Note the use of 's' to indicate a sharp note. A flat is noted with 'b'.)

Now programme a short melodic phrase in D major.

Developing a melody

Having explored creating musical phrases, we can now move on to writing a piece of music. In the activity that follows, we will produce a piece comprising four phrases of equal length. The piece will have a strong sense of key and will include elements of repetition.

ACTIVITY 5

Begin with the phrase you created in Activity 3. Alternatively write a new phrase, which has the same duration. Notice that the sleep values for this phrase add up to eight – which means the phrase lasts for eight seconds or eight beats. You might want to begin the phrase on the note C to help establish the sense of being in the key of C major.

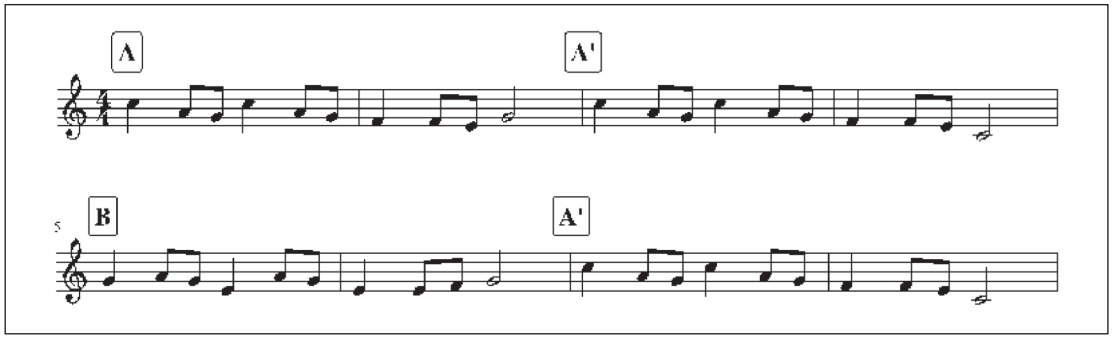
We are going to make a piece that has an AA'BA' structure.

The phrase we're starting with is the A section. To create the A' section copy/paste the first phrase coding below the original phrase. (Leave a one-line gap between the phrases so that it's easy to see where the first phrase ends and the second begins.) Tweak this second phase slightly so that it comes to rest on the key note C.

Leave another line space and create phrase B. This should be something new, but it should still have an overall length of eight beats.

Finally copy/paste the A' section to complete the piece.

Translated to standard notation, the final piece might look something like this:



Changing the tempo

Remember the blank lines we left at the beginning of this piece? We can use those to add additional musical commands.

On line 1, type *use_bpm 120*

Then press **Run**. You will notice the tune now plays at twice the speed (the default setting is 60 bpm).

Changing the sound

To do this, we need to access the various synths that come with Sonic Pi. Press the **Help** button, then the **synths** button at the bottom left of the screen. The synths are given in a list. Scroll through them and try some out by typing this example command on line 2:

use_synth :pretty_bell

Adding effects

With the Help menu still open, now select **fx**. Scroll through the options. To add some reverb type this instruction on line 3:

with_fx :reverb, mix: 0.7 do

At the very end of your piece type **end** on a new line.

The mix value indicates the amount of reverb added to the music. Mix values can be between 0 and 1. So a value of 0.7 can be thought of as adding 70% reverb. Consult the **Help** section for more on adding and tweaking effects.

Working with samples

In addition to synthesised sounds, it's also possible to work with audio samples in Sonic Pi. In this section, we look at some simple commands that will allow students to edit and manipulate recorded sounds.

The program comes with a library of pre-recorded samples, so it makes sense to get to know what's already available before going much further. The samples are stored in folders – ambient sounds, bass sounds, etc. To view them, press the **Help** button, then select the **Samples** tab at the bottom left of the screen. Open the 'Ambient sounds' folder, and scroll down a little to see the samples in this collection.

```
sample :ambi_soft_buzz
sample :ambi_swoosh
sample :ambi_drone
sample :ambi_glass_hum
sample :ambi_glass_rub
sample :ambi_haunted_hum
sample :ambi_piano
sample :ambi_lunar_land
```

To hear these samples, select an empty buffer and type in one of the sample commands shown above. For example:

sample :ambi_drone

Now press **Run** to hear the sample.

ACTIVITY 6

Build up a textured soundscape by programming a few samples. For example, a sequence might begin as follows:

```
sample :ambi_drone
sleep 1
sample :ambi_haunted_hum
sleep 2
sample :ambi_lunar_land
```

Varying the sleep values will control the degree of overlap. Note the auto complete feature, which will make programming sequences like this much easier.

When you have several samples playing at once, you can avoid aural clutter by 'spreading' the sounds across the stereo mix. Simply add a **pan** command:

```
sample :ambi_haunted_hum, pan: 1
```

A pan value of 1 will pan the sound hard right. A value of -1 pans hard left. Use values in between these extremes for more subtle panning.

Stretching samples

A sample can be stretched to increase its duration and alter its pitch. Stretching is also likely to have a radical effect on the timbre of the sample. We do this by adding a **rate** command:

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

A rate value of 0.5 will stretch the sample to twice its length and lower the pitch by an octave.

A rate value of 2 will compress the sample to half its length and raise the pitch by an octave.

A negative rate value will make the sample play backwards.

ACTIVITY 7

Add some rate values to the soundscape created in the previous activity to add further variation.

Looping samples

In the samples collection, there's a folder called 'Samples for Looping'. We can use these to create a continuous drum beat if we put them in a looping command:

```
loop do
 sample :loop_amen
 sleep 2
end
```

Run this loop and you will hear a slight hiccup between repeats. This is because the actual sample length is slightly shorter that our sleep value setting of two seconds. The best way around this is to stretch the sample length to exactly two seconds using a **beat_stretch** command:

```
loop do
 sample :loop_amen, beat_stretch:2
 sleep 2
end
```

ACTIVITY 8

Use the following coding as a template for exploring the potential for transforming samples:

```
loop do
 sample :loop_amen, beat_stretch:2, rate:1
 sleep 2
end
```

Now experiment with different **rate**, **beat_stretch** and **sleep** values. For example:

```
loop do
 sample :loop_amen, beat_stretch:0.6, rate:0.1
 sleep 0.6
end
```

Share the most successful ones with the rest of the class. Consider using one of these Sonic Pi loops as a percussion part in a small group composition or performance, where others are playing conventional instruments.

Importing samples

Despite the fact that there's a good (and ever growing) sample collection in Sonic Pi itself, there will come a time when many students will inevitably want to bring some of their own samples into the Sonic Pi framework. This is relatively straightforward, and is clearly explained in the **Help** section of the programme. From the **Tutorials** tab, go to **Samples/External Samples** for guidance on how to do this.

Exploring chord progressions

It's possible to create chords in Sonic Pi by simply stacking up **play** commands. However, there is a quicker way:

```
loop do
 play chord(:E3, :minor)
 sleep 2
 play chord(:A3, :minor)
 sleep 2
 play chord(:E3, :minor)
 sleep 2
 play chord(:B2, :'7')
 sleep 2
end
```

The **play chord** command will generate a chord on the given starting note. All standard chord types (major, minor, diminished, etc) are included, and a reference list of possibilities can be found under the **Lang** reference tab in the **Help** section.

Adding a **release** command will help the chords to sustain smoothly:

```
play chord(:E3, :minor), release:2
```

ACTIVITY 9

Using the model above, get students to create their own progression of eight to 16 chords. Add **use_bpm** and **use_synth** commands at the beginning to change the sound and the tempo, if required.

Play along to the progression using a guitar or keyboard – or improvise a tune over the top on any instrument.

The approach can also be used to create basic backings for folk or pop song standards.

Working with arpeggios

The **play_pattern_timed** chord command can be used to break down chords into their constituent arpeggios. For example, the coding below will give a two-octave pattern on the chords of E minor and A minor. The sleep values are added in brackets and, in this case, give a quaver-semiquaver-semiquaver pattern.

```
play_pattern_timed chord(:E3, :minor), [0.5, 0.25, 0.25]
play_pattern_timed chord(:E4, :minor), [0.5, 0.25, 0.25]
play_pattern_timed chord(:A3, :minor), [0.5, 0.25, 0.25]
play_pattern_timed chord(:A4, :minor), [0.5, 0.25, 0.25]
```

ACTIVITY 10

Using the example above, create a satisfying arpeggio progression for a sequence of chords. Enclose in a loop command to keep repeating the sequence. Add tempo and synth commands if required.

Attempt to play along on any instrument, or improvise a melodic line over the top.

Working with scales

As with chords, we can programme Sonic Pi to play all the notes of a scale without having to programme the notes in individually:

```
play_pattern_timed scale(:c3, :major, num_octaves: 3), 0.125, release: 0.1
```

In the coding above, we have used a **num_octaves** command to instruct the computer to play the scale over three octaves. Adjust the values for the various parameters to give different outcomes. For example:

- Change the starting note from c3 to c2. This will begin the sequence an octave lower.
- Change the scale from major to say, **whole_tone**.
- Increase or decrease the number of octaves.
- Change the sleep setting from 0.125 to 0.25 to double the length of the notes.
- Increase the **release** value to give notes of longer duration.

ACTIVITY 11

The coding included above can be used to create play-along routines for practising scales – a great idea if a grade examination is looming! Use the above coding to create routines for scale practice. Instrumentalists can modify it to any required scale. Enclose in a loop command for repetition of the pattern. Add a **use_bpm** command to set the tempo.

Chance procedures

Most of the musical procedures we have covered so far can also be achieved using standard sequencer technology. However, it's the ability to add chance procedures into the mix that makes using Sonic Pi such an interesting proposition.

For example, let's take a look at the coding in the previous section for playing scales. This is useful in helping explain certain theory concepts, aural tests and scale practice, but of more limited use in music making. Listening to a scale going up and down is just not that interesting. Using chance procedures changes all that, however. Consider the coding below:

```
use_synth :hoover
loop do
 play choose(chord(:D3, :minor)), release: 0.3, cutoff: rrand(60, 120)
 sleep 0.25
end
```

The **play choose** command selects the notes from the chord at random.

The **cutoff: rrand** command chooses values between 60 and 120 for the frequency settings of the synth used. Lower values will give a duller, bassy tone – higher values more crisp and trebly. This command can significantly add timbral interest.

ACTIVITY 12

We can adapt the coding above to produce interesting ostinato-style patterns that might work well in a soundscape or mood-type piece. Change the synth to **dark_ambience** and edit the other settings to produce something workable. Consider adding live instruments to produce something that might work as a film or dance soundtrack.

An alternative is to press the **Rec** button before hitting **Run** and **Stop**. The output will be saved as a wav file, which can be burnt to disc or imported into a sequencer programme for further music creating.

Live looping

Sonic Pi has an **Examples** folder, accessed from the **Help** tab. Copying these into a spare buffer and experimenting with them is another way to gain some valuable insights into the potential of this program. Live loops are really useful for live performance using Sonic Pi as they allow the user to make changes to sound parameters 'on the fly', without having to stop and restart the program.

This next example is adapted from Darin Wilson's *Ambient Experiment*, which has three long loops of different lengths playing together. A random command is added to provide continuous pitch variations to the layers of sound. The choice of synth and use of reverb add to the atmospheric ambience:

```
use_synth :hollow
with_fx :reverb, mix: 0.8 do
 live_loop :note1 do
 play choose([:C4,:E4]), attack: 7, release: 5
 sleep 9
 end
 live_loop :note2 do
 play choose([:E3,:G4]), attack: 3, release: 6
 sleep 10
 end
 live_loop :note3 do
 play choose([:C2, :C5]), attack: 5, release: 8
 sleep 7
 end
end
```

The coding shows that we have three loops playing simultaneously. Each **live loop** has to be given a unique name – in this case, the loops have been named note 1, note 2 and note 3. They could just as easily be labelled red, blue and green, for example. Each loop has a **choose** command, which can randomly choose between one of two notes. So the first loop is set to choose between C4 and E4. The **sleep** values are set to different lengths, which will vary the duration of respective loops. **Attack** and **release** control the fade in/fade out of notes.

In this instance, the pitches are all set to notes from a C major chord, so running the program will result in shifting voicings of a C major chord. Try changing some of the notes to other notes from C major at various octaves to generate even more variations in voicings of the chord.

Experiment also with different **attack**, **release** and **sleep** times to make the piece sound more energetic or more restful.

The changes will come into effect each time you press **Run**.

ACTIVITY 13

Now let's think about changing the harmony. We can use this as a whole-class listening and music theory exercise.

Change one of the notes in one of the loops to a B flat (Bb). This will, of course, change the chord to a C7. When you press **Run**, the change will not happen immediately. It will come into effect when that particular loop decides to choose the B flat. Ask the class to close their eyes and to raise their hands when they can hear the change to a C7 chord.

Ask the class which chord the C7 is aiming towards in a typical diatonic chord progression context. Hopefully one of them will suggest a dominant to tonic, C7 to F major. Now ask what are the smallest pitch changes we can make to the loops to bring about an F major chord. These are as follows:

- Cs can remain as they are.
- Es up a semitone to F.
- Gs up a tone to A.
- B flats drop a semitone to A.

Make these changes then press **Run**. Of course the notes will change in their own time as dictated by the **choose** commands. So there will be some interesting layers of harmony before the chord finally comes to rest on an F major.

This harmonic exploration can continue. For example, what changes need to be made to change the chord from F major to F minor?

ACTIVITY 14

Running live loops in the way described above is not just a useful classroom learning activity. They can also provide the basis for compelling live performances. You may have noticed that the layers of sound generated in the previous activity can sound very interesting musically – evoking the sound worlds of Brian Eno or possibly Arvo Pärt.

Set students the task of adapting and modifying the above coding to create an interesting performance of ambient music, lasting between five and ten minutes. They need to think about different chord progressions and different rates of change. Further live loops can be added to create even thicker textures of sound.

Note that to switch off a layer of sound, simply add a hashtag in front of the play choose command:

```
#play choose([:C2, :C5]), attack: 5, release: 8
```

Remove the hashtag to have that loop start playing again.

This activity can be used as the basis for live performance, solo or with additional conventional instruments. It could also be used for creating a film soundtrack.

BRINGING IT ALL TOGETHER

Over the course of this resource we've covered quite a number of routines. Let's round off by bringing a few of these together for an algorave! Algoraves have been described as events where people dance to music generated from algorithms, often using live coding techniques. Algoraves can include a range of EDM (Electronic Dance Mix) styles, and represent an ever-growing meeting point of hacker philosophy, geek culture and clubbing.

Let's look at the following coding, which provides a template for a techno performance. It comprises a collection of loops that can be switched on and off and modified during the course of a performance:

```
live_loop :riff do
 notes = scale(:c2, :minor_pentatonic, num_octaves:1 )
 #play notes.choose, release:0.3, amp:rrand(0.2,0.4), pan: -0.5
 sleep 0.25
end
live_loop :drum1 do
 sync :riff
 #sample :bd_zum, amp:rrand(0.13,0.25)
 sleep 0.5
end
live_loop :drum2 do
 sync :riff
 #sample :loop_mika, amp:rrand(0.13,0.25)
 sleep 4
end
live_loop :mystery do
 sync :riff
 sample :ambi_haunted_hum, rate:rrand(0.5,0.7), amp:rrand(0.13,0.14), pan:rand
 sleep 6
end
live_loop :drone do
 sync :riff
 #sample :bass_trance_c, amp:rrand(0.1,0.3), pan:0.5
 sleep 2
end
```

Some performance notes and suggestions:

- Notice that all loops except *mystery* have a hashtag in front of the playing instruction. This means that they will be muted at the start. Remove them one at a time to gradually bring in layers of additional sound.
- Some of the loops have **amp:rrand** commands. This has the effect of varying the volumes throughout the performance, boosting and fading sounds. Change these number settings to alter the dynamic range.
- Some loops have panning instructions to help spread the sound over the stereo mix. Positive numbers pan to the right; negative numbers pan left. The **pan:rand** command will result in random panning.
- The mystery loop also has a **rate:rrand** to add a bit of variation.
- The drum2 loop has a sleep value of 4. Raise or lower this number to vary the length of the drum loop.
- The first loop, riff, has a **num_octaves** setting of 1. This restricts it to just playing low notes, which will be barely perceptible as a throbbing sound when everything else is happening. Increase the number of octave setting to 3, for example, and you'll notice higher notes emerging, which will often work together to provide a melodic part.
- Sometimes make subtle changes, just adding enough variation to retain interest. At other times go for more extreme contrasts. For example, switch all loops off apart from the two drum loops to produce a percussion-only section.

WHERE NEXT?

The procedures we've covered in this resource represent only the tip of a very large iceberg. But for teachers and students who want to take these ideas further, there are a range of support options:

- Sonic Pi Live & Coding has downloadable lesson plans, links to a research report about using Sonic Pi in education, and a YouTube playlist of live coding performances.
- The main Sonic Pic site is the host website for downloading this free software for any platform. It also has a link to inThread – the online support community – and an introductory article and video from Sonic Pi creator Sam Aaron.
- The Help section of the program itself has comprehensive tutorials and PDF reprint of articles on Sonic Pi from the MagPi magazine.
- *Sonic Pi in the Music Classroom* and *Making the most of Sonic Pi: Creative Music Making with Computers* are both publications I have written, available from Amazon.