**Programming with Sonic Pi**

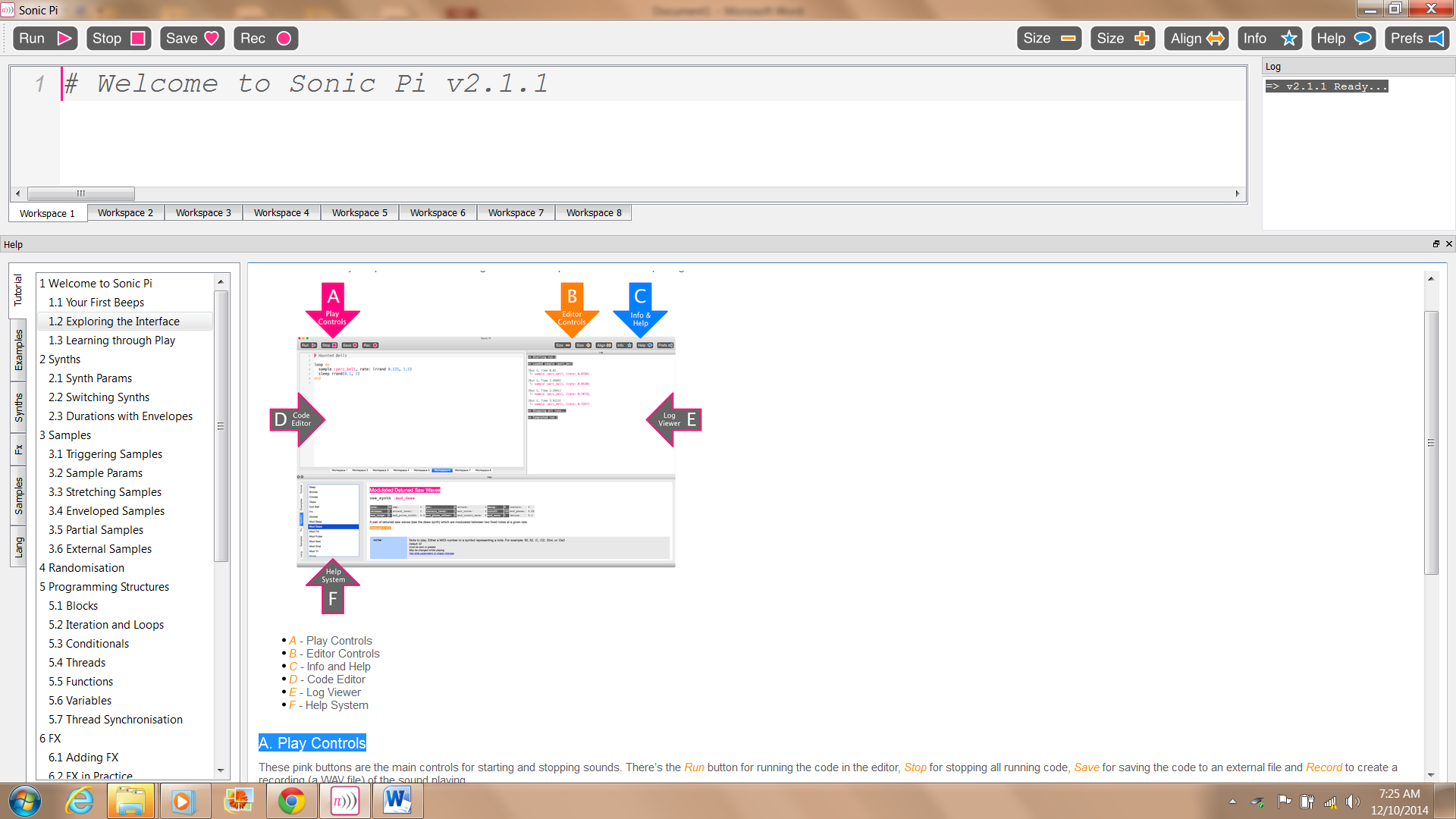
In this exercise, you will be programming the sound output from your Raspberry Pi. We will monitor the sound output with headphones. The sound output jack is what is known as “line level”. This is a very low voltage level similar to what would come out of an electric guitar or stereo turntable. Normally this would be hooked up to a stereo amplifier line input. In order to make that sound so we can hear it with our headphones we need to program the audio chip to be at a higher level. This should not be a problem as our 2.1 amp (2100 mA) power supply gives us a lot of room to grow. Remember the Pi draws about 0.7 amps (700 mA). We can do this by entering the following command in the terminal:

amixer set PCM -- 400

The amixer command will access the Alsa sound chip. PCM stands for Pulse Coded Modulation which is a method of digitally encoding an analog sample. The value 400 refers to decibels related to some reference value. 400 will set the Alsa sound chip to 100% output which should be enough for our purposes.

**The Sonic Pi Application**

Once you have started the Sonic Pi app, you should see a simple application window with three panels:



**Your first sound**

The first command to learn is play this takes a number as a parameter. For example:

play 60

Type play 60 into the text pane and pressing the play button. You should hear a simple bell-like pling sound! Remember to plug your headphones or a speaker into the RPi!

Try changing 60 to other numbers. What happens if you choose a number too low or high? You might be interested to know that these numbers are MIDI numbers. See the following link for how MIDI relates to a piano and more standard note names: http://www.phys.unsw.edu.au/jw/notes.html

**Your first piece**

The next command to learn is sleep which also takes a number as a parameter. In this case the number doesn't represent pitch, it represents time:

sleep 1

This will force the program to pause for one second. You can use sleep to add delays in your composition allowing you to create a melody. Try writing a bunch of play and sleep commands such as:

play 60

sleep 0.25

play 62

sleep 0.5

play 66

Go ahead and start writing your own pieces!

Here are some examples for you to try out.

play 42

sleep 0.5

play 45

sleep 0.85

play 54

sleep 1

play 54

sleep 0.7

play 45

sleep 0.2

play 49

sleep 1

Now let’s put this in a loop.

5.times do

play 42

sleep 0.5

... rest of code ...

sleep 1

end

Now let’s try these experiments. Put each one on its own worksheet

Using random number generation

3.times do

play 60 + rand(10)

sleep 0.5

end

Decision making structure

5.times do

if rand < 0.5

play 60

sleep 0.5

play 62

else

play 72

sleep 0.25

play 71

sleep 0.25

play 70

end

end

Data structure examples

play\_pattern [40,25,45,25,25,50,50]

play\_pattern [40,25,45,25,25,50,50].sort

play\_pattern [40,25,45,25,25,50,50].shuffle

play\_chord [60,65,67]

**Demo Program**

use\_bpm 350

2.times do

play\_pattern [60,55,65,55,55,70,70]

play\_pattern [55,70,55,60,65,50,55,70]

play\_pattern [55,70,55,60,65,50,55,70].reverse

end

2.times do

use\_synth :pretty\_bell

play\_pattern [25,50,25,30,35,40,45,50].shuffle

play\_pattern [25,50,25,30,35,40,45,50].reverse

end

in\_thread do

use\_synth :prophet

10.times do

if rand < 0.5

play 57

else

play 68

end

sleep 2

end

end

### Haunted Bells

loopdo

sample:perc\_bell,rate: (rrand0.125,1.5)

sleeprrand(0,2)

end

**Running Examples**

From the Help window, try copying and pasting some of the example programs into a workspace and running them