VA336/546 interactive sound

Week 11

Advanced Audio Programming

Instructor: Assist. Prof. Dr. Selcuk ARTUT

Email: sartut@sabanciuniv.edu

Web: selcukartut.com/teaching

Supercollider

SuperCollider is a platform for audio synthesis and algorithmic composition, used by musicians, artists, and researchers working with sound. It is free and open source software available for Windows, macOS, and Linux.

Ref: https://supercollider.github.io

SuperCollider features three major components:

scsynth, a real-time audio server, forms the core of the platform. It features 400+ unit generators (“UGens”) for analysis, synthesis, and processing. Its granularity allows the fluid combination of many known and unknown audio techniques, moving between additive and subtractive synthesis, FM, granular synthesis, FFT, and physical modeling. You can write your own UGens in C++, and users have already contributed several hundred more to the sc3-plugins repository.

sclang, an interpreted programming language. It is focused on sound, but not limited to any specific domain. sclang controls scsynth via Open Sound Control. You can use it for algorithmic composition and sequencing, finding new sound synthesis methods, connecting your app to external hardware including MIDI controllers, network music, writing GUIs and visual displays, or for your daily programming experiments. It has a stock of user-contributed extensions called Quarks.

scide is an editor for sclang with an integrated help system.

Download and Install SuperCollider

Before we can make any sound, we need to start or 'boot' a server application. The easiest way to do this is to use the shortcut Ctrl-B (Cmd-B on the mac). There is also a Menu entry for "Boot Server".

Try

```
{ SinOsc.ar(440, 0, 0.2) }.play;
```

To stop it platying CTRL (cmd) + .

In order to have sth stereo

```
{ [SinOsc.ar(440, 0, 0.2), SinOsc.ar(442, 0, 0.2)] }.play;
```

Download and Install SuperCollider

Before we can make any sound, we need to start or 'boot' a server application. The easiest way to do this is to use the shortcut Ctrl-B (Cmd-B on the mac). There is also a Menu entry for "Boot Server".

Try

```
{ SinOsc.ar(440, 0, 0.2) }.play;
```

To stop it platying CTRL (cmd) + .

In order to have sth stereo

```
{ [SinOsc.ar(440, 0, 0.2), SinOsc.ar(442, 0, 0.2)] }.play;
```

For getting more serious

I recommend you to follow these tutorials

https://www.youtube.com/watch?v=yRzsOOiJ_p4

How about live coding?

def: (wiki) Live coding, sometimes referred to as on-the-fly programming, just in time programming and conversational programming, makes programming an integral part of the running program.

Common platforms

- ChucK
- COLT
- Extempore
- Fluxus
- Impromptu
- ixi lang
- LiveCode
- Lua
- Max
- Pharo
- Pure Data
- Scratch
- Sonic Pi
- SuperCollider
- TidalCycles

TOPLAP is an organisation founded in 2004, to explore and promote live coding.

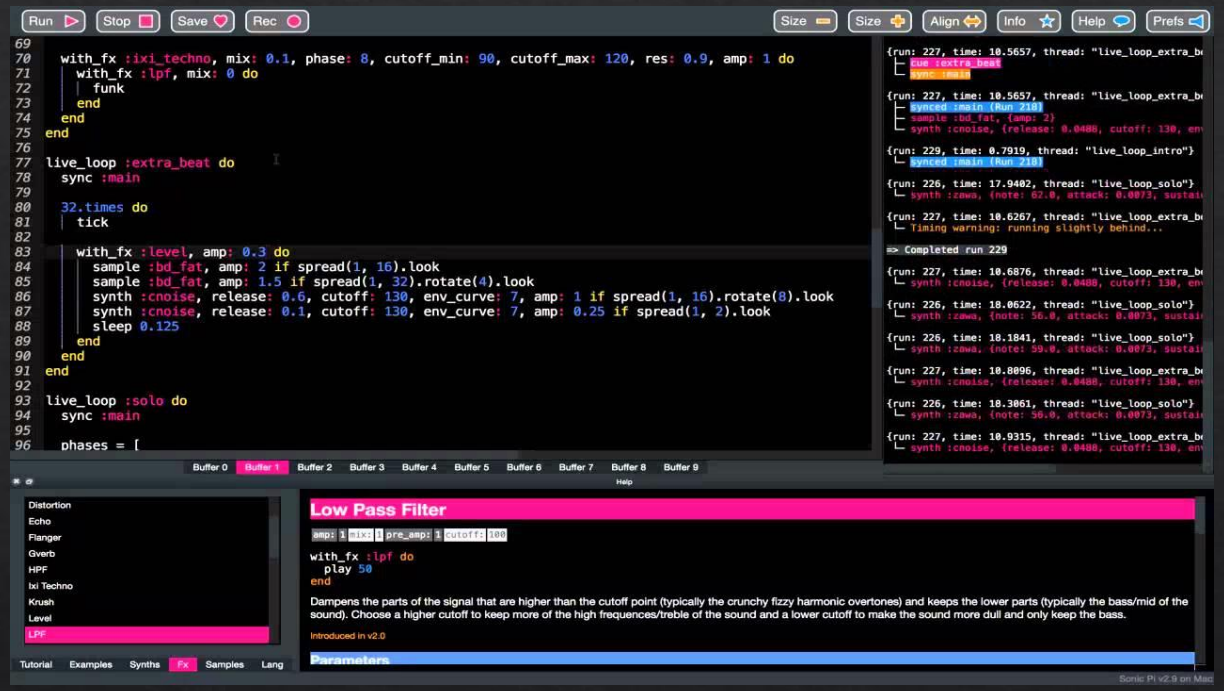
Live coding is a new direction in electronic music and video, and is getting somewhere interesting. Live coders expose and rewire the innards of software while it generates improvised music and/or visuals. All code manipulation is projected for your pleasure. Live coding works across musical genres, and has been seen in concert halls, late night jazz bars, as well as algoraves. There is also a strong movement of video-based live coders, writing code to make visuals, and many environments can do both sound and video, creating synaesthetic experiences.

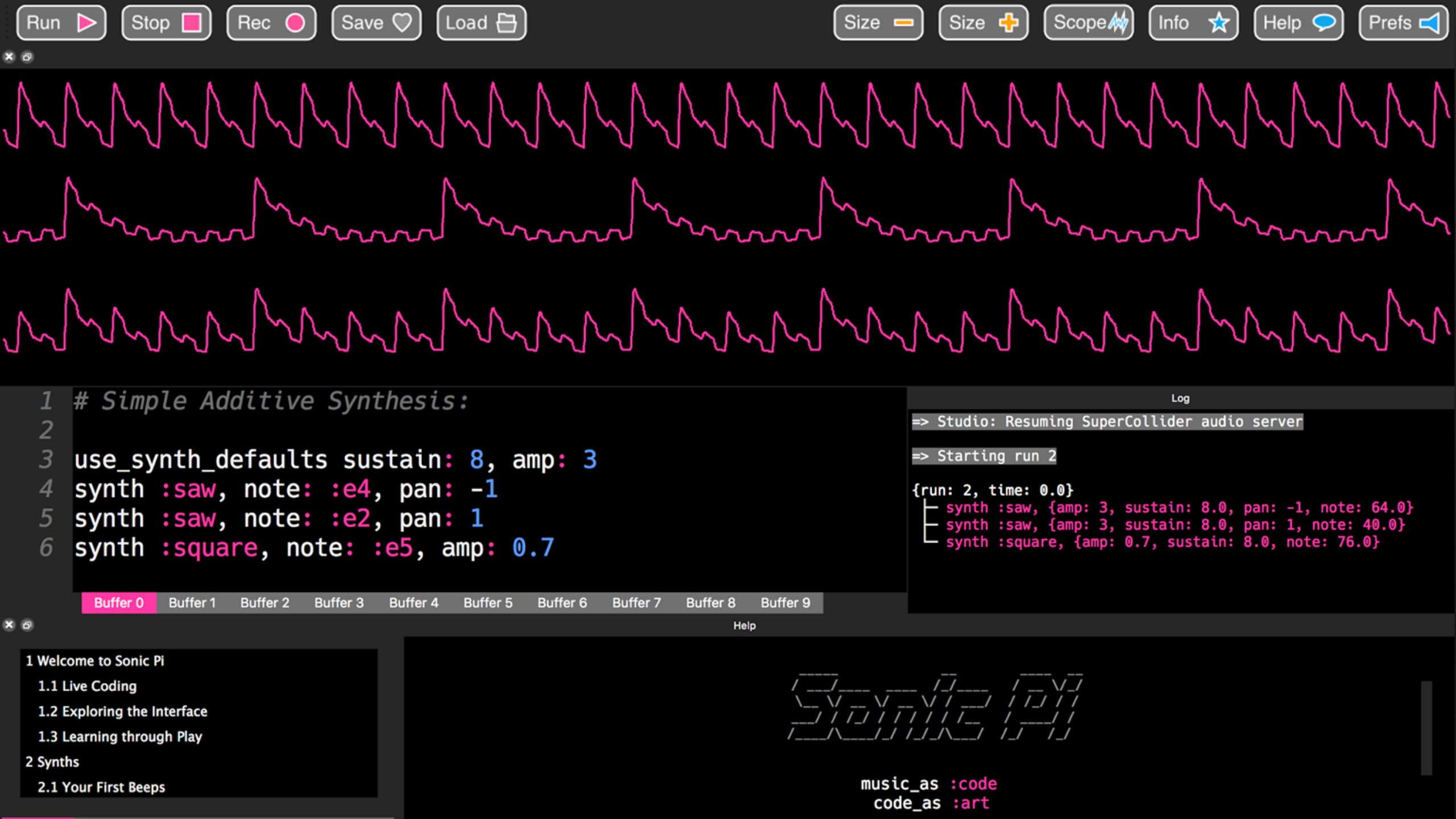
- https://toplap.org

Examples

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

Sonic Pi


Run
Stop
Save
Rec
Size
Size
Align
Info
Help
Prefs
with_fx :ixi_techno, mix: 0.1, phase: 8, cutoff_min: 90, cutoff_max: 120, res: 0.9, amp: 1 do
with_fx :lpf, mix: 0 do
funk
end
end
end
live_loop :extra_beat do
sync :main
32.times do
tick
with_fx :level, amp: 0.3 do
sample :bd_fat, amp: 2 if spread(1, 16).look
sample :bd_fat, amp: 1.5 if spread(1, 32).rotate(4).look
synth :cnoise, release: 0.6, cutoff: 130, env_curve: 7, amp: 1 if spread(1, 16).rotate(8).look
synth :cnoise, release: 0.1, cutoff: 130, env_curve: 7, amp: 0.25 if spread(1, 2).look
sleep 0.125
end
end
end
live_loop :solo do
sync :main
phases = [
Buffer 0
Buffer 1
Buffer 2
Buffer 3
Buffer 4
Buffer 5
Buffer 6
Buffer 7
Buffer 8
Buffer 9
Help
Distortion
Echo
Flanger
Gverb
HPF
Ixi Techno
Krush
Level
LPF
Low Pass Filter
amp: 1 mix: 1 pre_amp: 1 cutoff: 100
with_fx :lpf do
play 50
end
Dampens the parts of the signal that are higher than the cutoff point (typically the crunchy fizzy harmonic overtones) and keeps the lower parts (typically the bass/mid of the sound). Choose a higher cutoff to keep more of the high frequences/treble of the sound and a lower cutoff to make the sound more dull and only keep the bass.
Introduced in v2.0
Parameters
Tutorial
Examples
Synths
Fx
Samples
Lang


Run
Stop
Rec
Save
Load
Size
Size
Scope
Info
Help
Prefs
Simple Additive Synthesis:
use_synth_defaults sustain: 8, amp: 3
synth :saw, note: :e4, pan: -1
synth :saw, note: :e2, pan: 1
synth :square, note: :e5, amp: 0.7
Buffer 0
Buffer 1
Buffer 2
Buffer 3
Buffer 4
Buffer 5
Buffer 6
Buffer 7
Buffer 8
Buffer 9
Log
=> Studio: Resuming SuperCollider audio server
=> Starting run 2
{run: 2, time: 0.0}
synth :saw, {amp: 3, sustain: 8.0, pan: -1, note: 64.0}
synth :saw, {amp: 3, sustain: 8.0, pan: 1, note: 40.0}
synth :square, {amp: 0.7, sustain: 8.0, note: 76.0}
Help
1 Welcome to Sonic Pi
1.1 Live Coding
1.2 Exploring the Interface
1.3 Learning through Play
2 Synths
2.1 Your First Beeps
music_as :code
code_as :art

Sonic Pi

Run | Stop | Rec | Save | Load | Size | Size | Align | Info | Help | Prefs

```
1  # Hello Stir Trek!
2  # This is file 1 of 5
3  # You can find it at
4  # https://github.com/sfradkin/presentations/tree/master/sonic-pi-p
5  # after the conference
6
7  ##| Overview of the Sonic Pi
8  ## environment, especially the prefs and help
9
10
11 ##| Play (midi note number or note + octave)
12 ##| Play 60
13 ##| The numbers are midi notes
14
15
16
17 ##| You can also use: play :c4
18 ##| Symbols for notes + octave
19 ##| Use 's' for sharps or 'b' for flats, :cs4
```

Buffer 0 | Buffer 1 | Buffer 2 | Buffer 3 | Buffer 4 | Buffer 5 | Buffer 6 | Buffer 7 | Buffer 8 | Buffer 9

Log

```
=> Welcome to Sonic Pi
=> Session 9df4d07b
=> Friday 6th May, 2016
=> 07:46, CDT
=> Hello, it's lovely to see
   you again. I do hope that
   you're well.
=> v2.10 Ready...
=> Studio: Initializing...
=> Starting run 1
=> Completed run 1
```

Help

1 Welcome to Sonic Pi
- 1.1 Live Coding
- 1.2 Exploring the Interface
- 1.3 Learning through Play

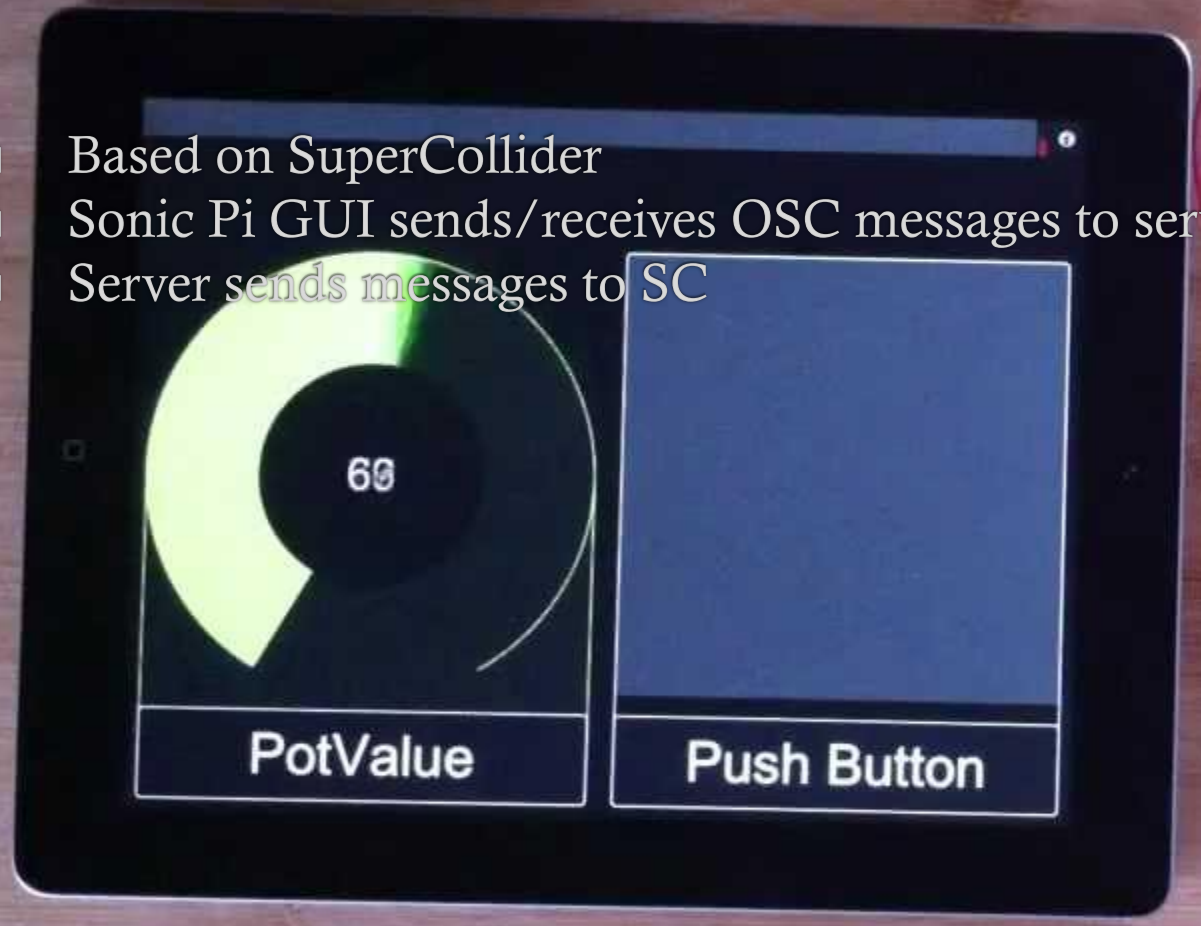
2 Synths

Sonic Pi

music as :code

How it works?

- Based on SuperCollider
- Sonic Pi GUI sends/receives OSC messages to server
- Server sends messages to SC

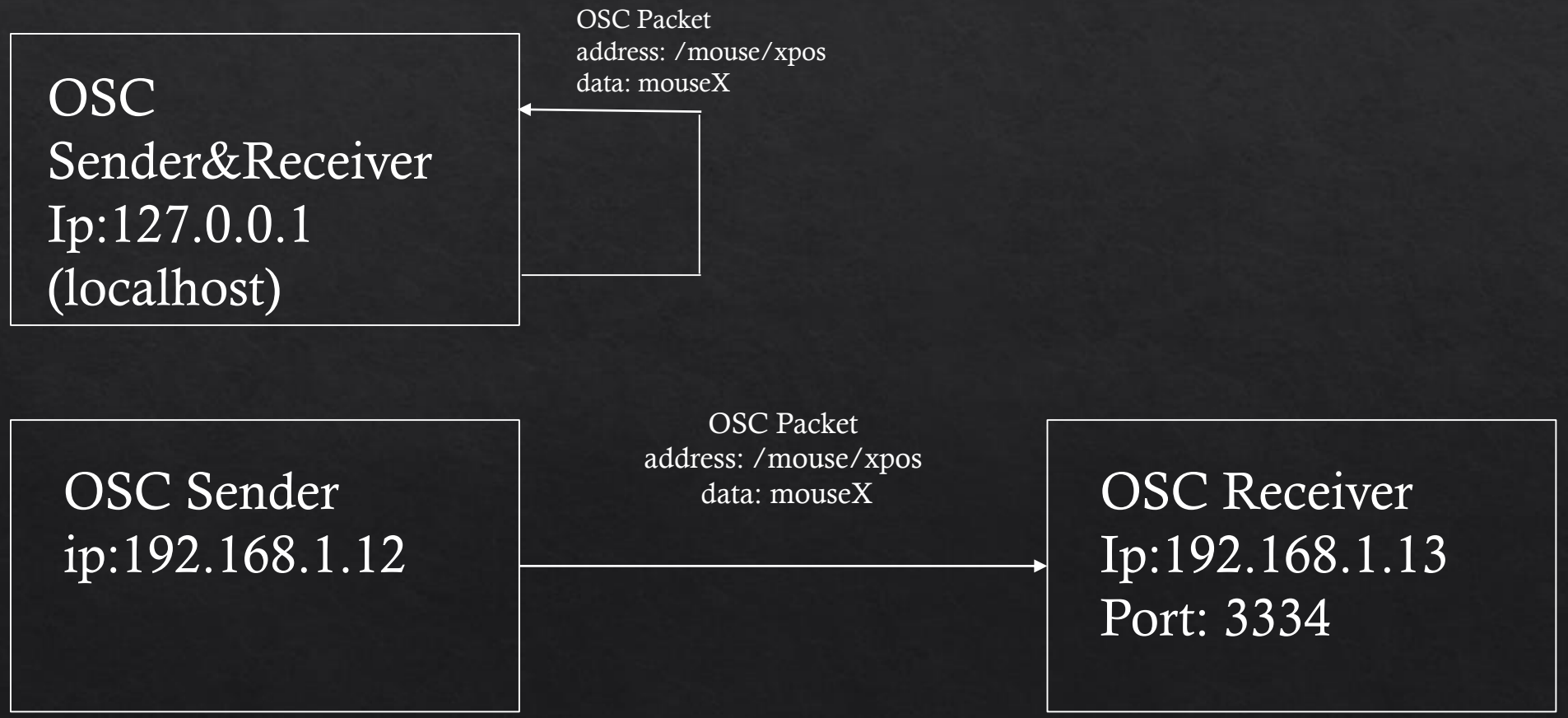


OSC (Open Sound Control)

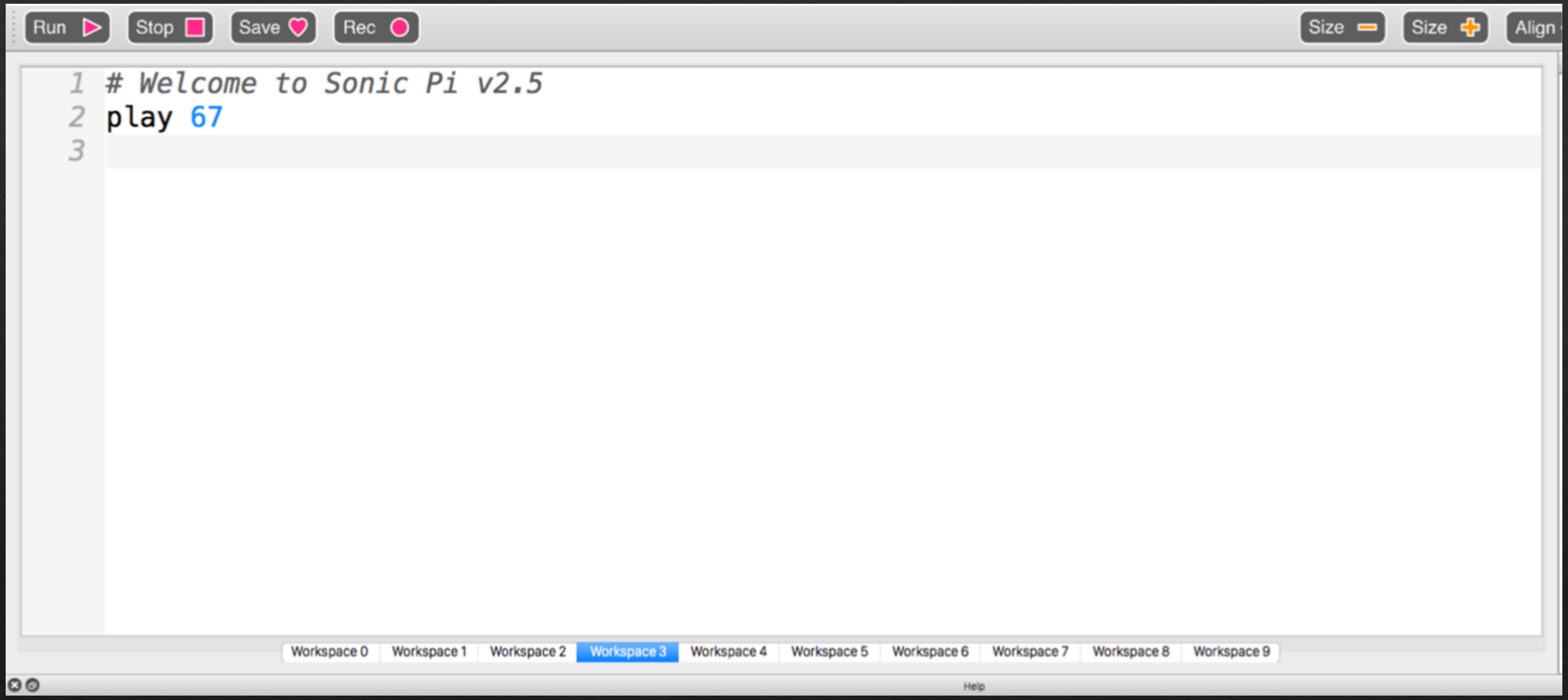
- Ongoing research project by Berkeley Center for New Music and Audio Technology (CNMAT)
- Open Sound Control (OSC) is an open,
- Transport-independent,
- Message-based protocol based on UDP
- Developed for communication among computers, sound synthesizers, and other multimedia devices.

OSC Communication

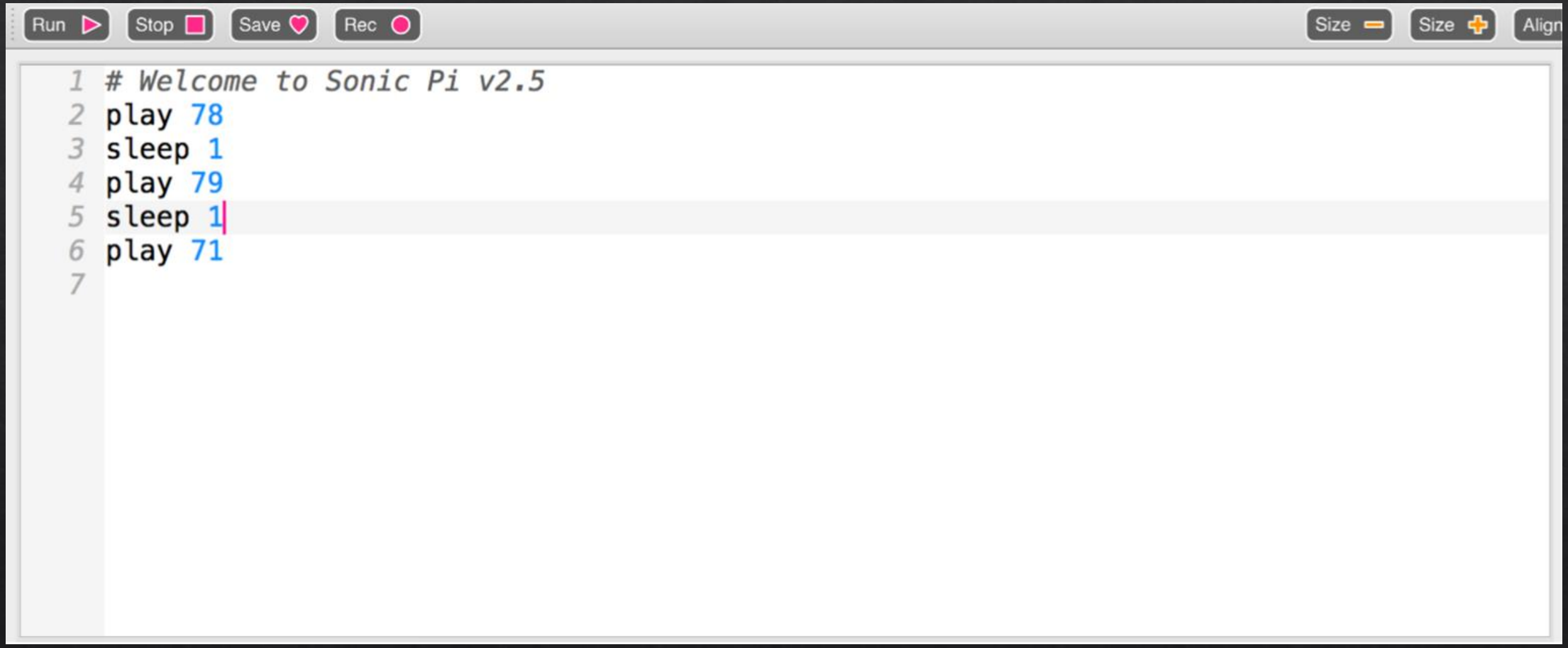
For more than one devices, you need a local network, each device need to be delivering unique ip's..
Be aware if there is any firewall restriction



https://sonic-pi.net


Run
Stop
Save
Rec
Size
Size
Align
1 # Welcome to Sonic Pi v2.5
2 play 67
3
Workspace 0
Workspace 1
Workspace 2
Workspace 3
Workspace 4
Workspace 5
Workspace 6
Workspace 7
Workspace 8
Workspace 9
Help


Run
Stop
Save
Rec
Size
Size
Align
1 # Welcome to Sonic Pi v2.5
2 play 72
3 play 75
4 play 79
5

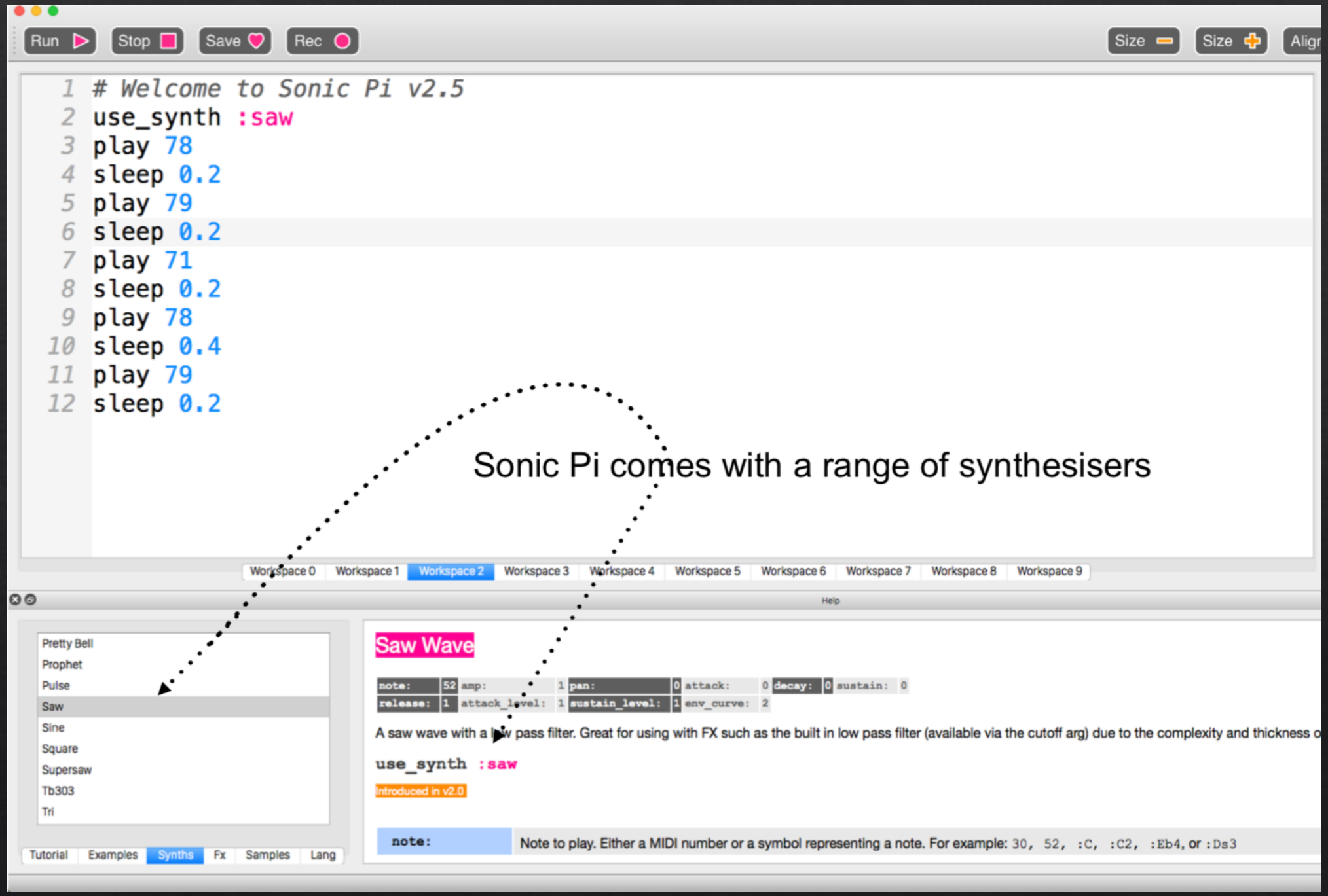

Run
Stop
Save
Rec
Size
Size
Align
1 # Welcome to Sonic Pi v2.5
2 play 78
3 sleep 1
4 play 79
5 sleep 1
6 play 71
7

Run Stop Save Rec Size Size Align

```
# Welcome to Sonic Pi v2.5
play :Fs5
sleep 0.2
play :G5
sleep 0.2
play :B4
```

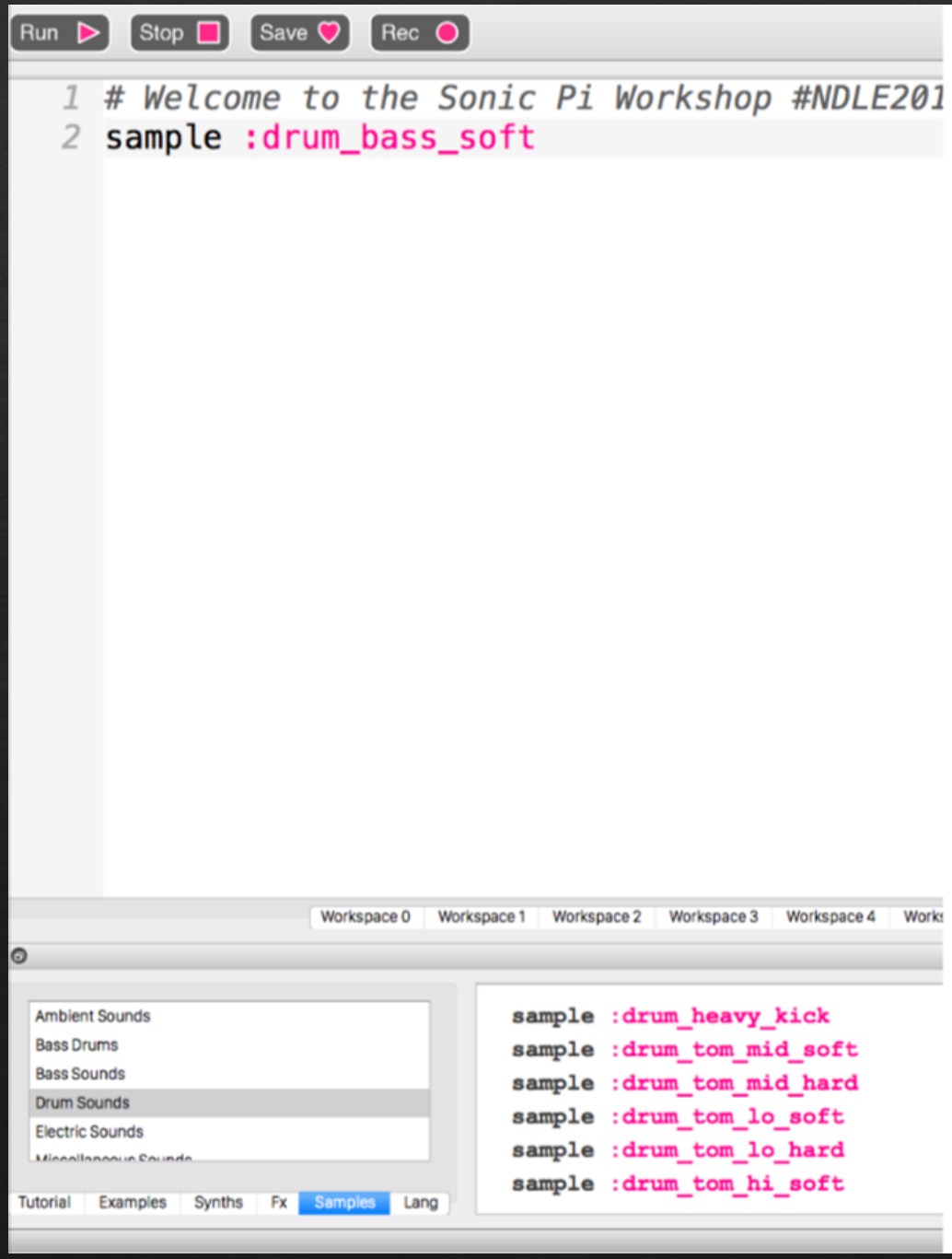
Octave	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				


Run
Stop
Save
Rec
Size
Size
Welcome to Sonic Pi v2.5
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 78
sleep 0.4
play 79
sleep 0.2


Run
Stop
Save
Rec
Size
Size
1 # Welcome to Sonic Pi v2.5
2 use_synth :saw
3 play 78
4 sleep 0.2
5 play 79
6 sleep 0.2
7 play 71
8 sleep 0.2
9 play 78
10 sleep 0.4
11 play 79
12 sleep 0.2
Sonic Pi comes with a range of synthesisers
Workspace 0
Workspace 1
Workspace 2
Workspace 3
Workspace 4
Workspace 5
Workspace 6
Workspace 7
Workspace 8
Workspace 9
Help
Pretty Bell
Prophet
Pulse
Saw
Sine
Square
Supersaw
Tb303
Tri
Tutorial
Examples
Synths
Fx
Samples
Lang
Saw Wave
note: 52 amp: 1 pan: 0 attack: 0 decay: 0 sustain: 0
release: 1 attack_level: 1 sustain_level: 1 env_curve: 2
A saw wave with a low pass filter. Great for using with FX such as the built in low pass filter (available via the cutoff arg) due to the complexity and thickness o
use_synth :saw
Introduced in v2.0
note:
Note to play. Either a MIDI number or a symbol representing a note. For example: 30, 52, :C, :C2, :Eb4, or :Ds3

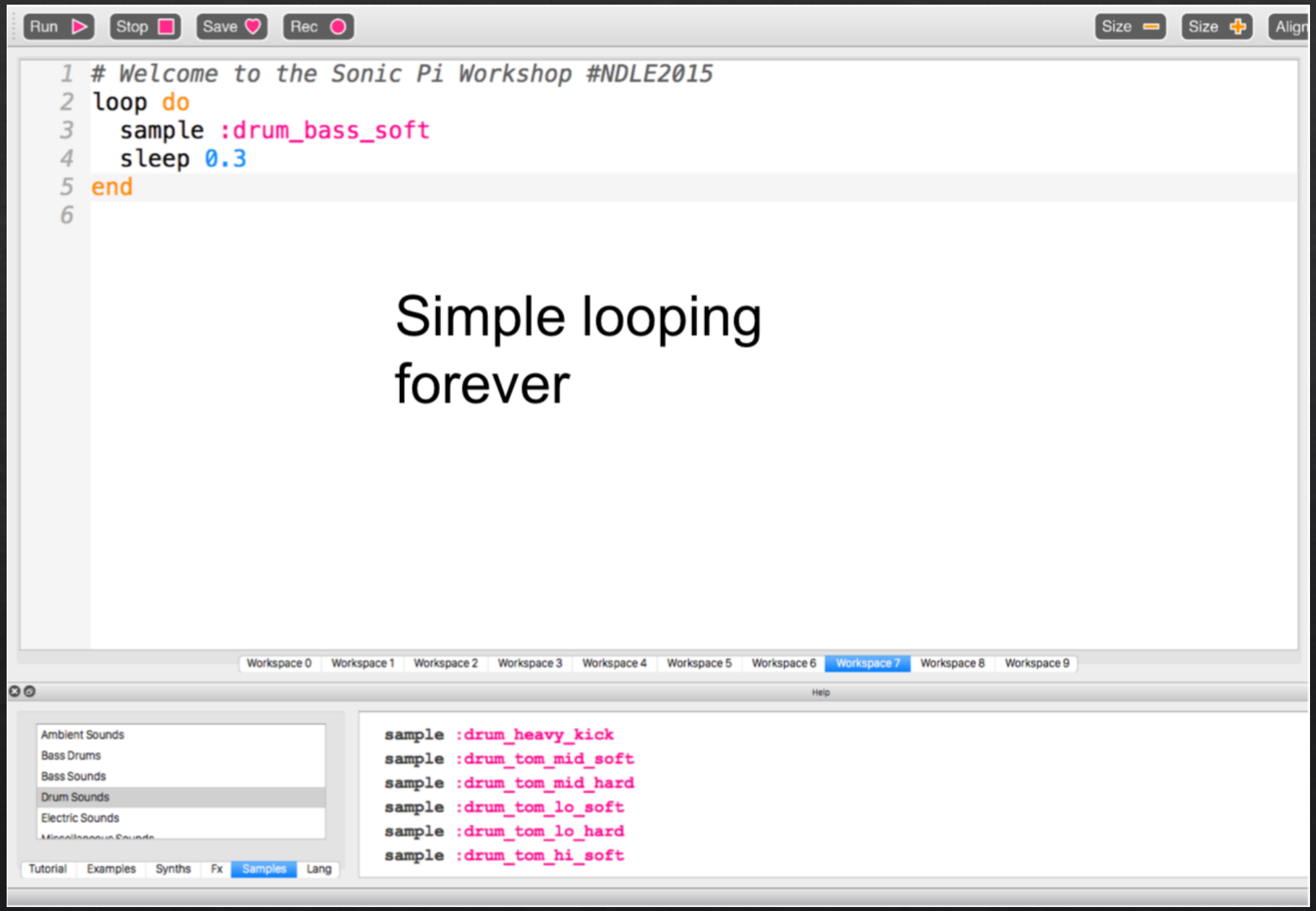
Run Stop Save Rec Size Size Align

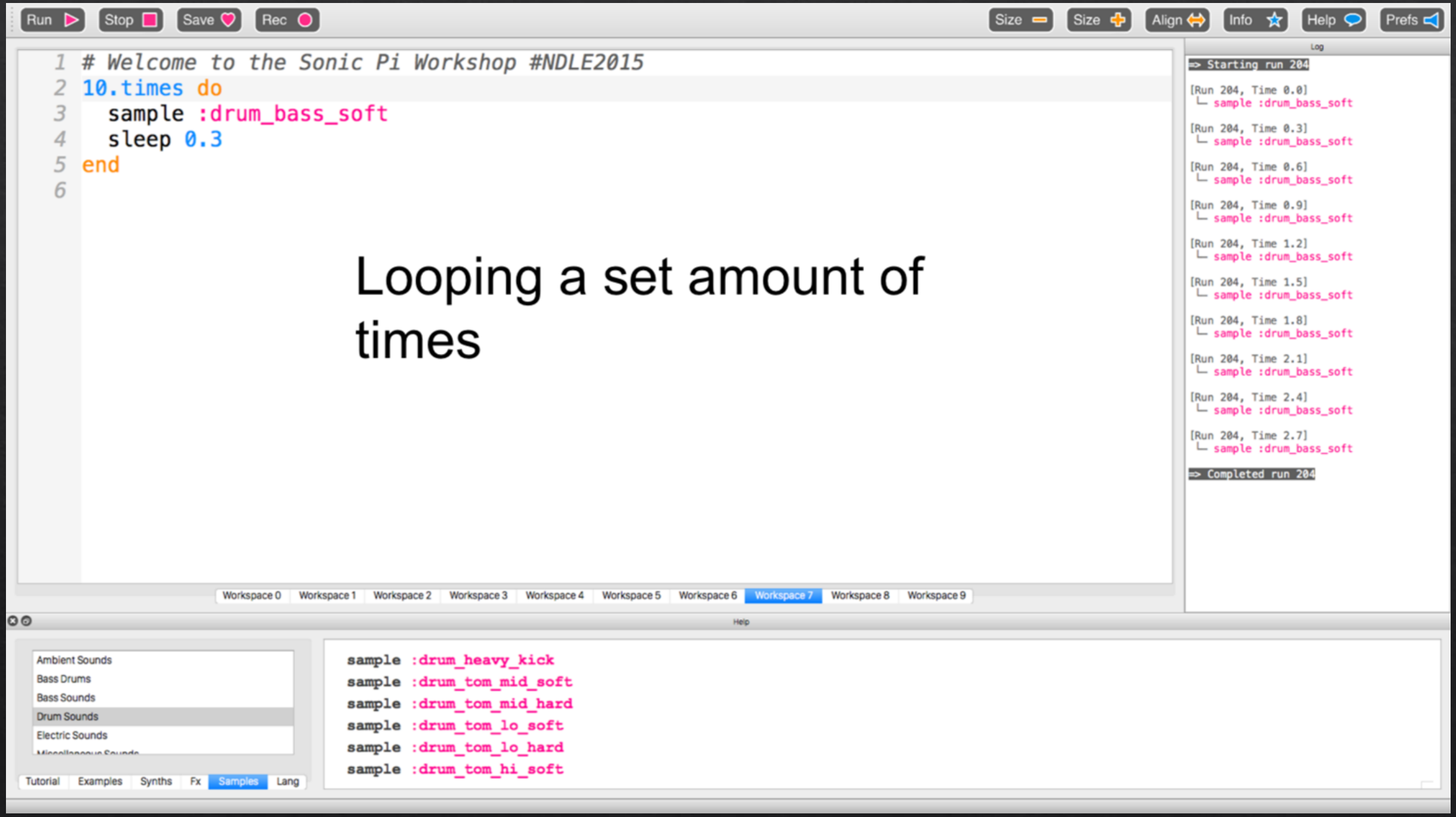
```
# Welcome to the Sonic Pi Workshop #NDLE2015
use_synth :saw
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 78
sleep 0.4
play 79
sleep 0.8
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 79
```



Open a new workspace and try some samples

There are lots listed in the sample section, it also shows you how to write them in your code


Run
Stop
Save
Rec
Size
Size
Align
1 # Welcome to the Sonic Pi Workshop #NDLE2015
2 loop do
3 sample :drum_bass_soft
4 sleep 0.3
5 end
6
Simple looping forever
Workspace 0
Workspace 1
Workspace 2
Workspace 3
Workspace 4
Workspace 5
Workspace 6
Workspace 7
Workspace 8
Workspace 9
Help
Ambient Sounds
Bass Drums
Bass Sounds
Drum Sounds
Electric Sounds
Tutorial
Examples
Synths
Fx
Samples
Lang
sample :drum_heavy_kick
sample :drum_tom_mid_soft
sample :drum_tom_mid_hard
sample :drum_tom_lo_soft
sample :drum_tom_lo_hard
sample :drum_tom_hi_soft


Run
Stop
Save
Rec
Size
Size
Align
Info
Help
Prefs
1 # Welcome to the Sonic Pi Workshop #NDLE2015
2 10.times do
3 sample :drum_bass_soft
4 sleep 0.3
5 end
6
Looping a set amount of times
Log
=> Starting run 204
[Run 204, Time 0.0]
└ sample :drum_bass_soft
[Run 204, Time 0.3]
└ sample :drum_bass_soft
[Run 204, Time 0.6]
└ sample :drum_bass_soft
[Run 204, Time 0.9]
└ sample :drum_bass_soft
[Run 204, Time 1.2]
└ sample :drum_bass_soft
[Run 204, Time 1.5]
└ sample :drum_bass_soft
[Run 204, Time 1.8]
└ sample :drum_bass_soft
[Run 204, Time 2.1]
└ sample :drum_bass_soft
[Run 204, Time 2.4]
└ sample :drum_bass_soft
[Run 204, Time 2.7]
└ sample :drum_bass_soft
=> Completed run 204
Workspace 0
Workspace 1
Workspace 2
Workspace 3
Workspace 4
Workspace 5
Workspace 6
Workspace 7
Workspace 8
Workspace 9
Help
Ambient Sounds
Bass Drums
Bass Sounds
Drum Sounds
Electric Sounds
Tutorial
Examples
Synths
Fx
Samples
Lang
sample :drum_heavy_kick
sample :drum_tom_mid_soft
sample :drum_tom_mid_hard
sample :drum_tom_lo_soft
sample :drum_tom_lo_hard
sample :drum_tom_hi_soft

Live Loops

```
live_loop :foo do
 play 60
 sleep 1
end
```

Live Loops + synth + sound samples

```
live_loop :foo do
 use_synth :prophet
 play :c1, release:
8, cutoff: rrand(70,
130)
 sleep 8
end

live_loop :bar do
 sample :bd_haus
 sleep 0.5
end
```

Load external sound samples

```
# 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
```

Live Loops to try

```
live_loop :arp do
 play (scale :e3, :minor_pentatonic).tick,
release: 0.1
 sleep 0.125
end
```

List & Arrays

play 52

play 55

play 59

try...

play [52, 55, 59] or play [:E3, :G3, :B3]

Chords

play chord(:E3, :minor)