Live Coding for Creating Audio-Visual Experiences

Presenter: Assoc. Prof. Dr. Selcuk ARTUT

Visual Arts and Visual Communication Design

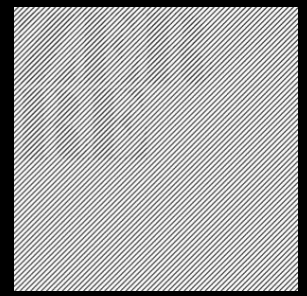
Sabanci University, Istanbul Turkiye

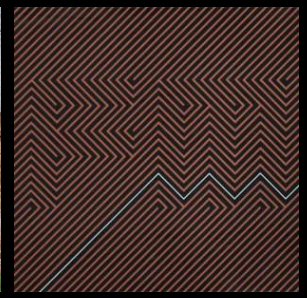
Musical Background

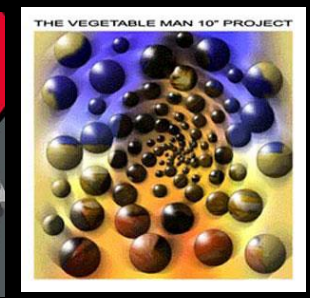
Passion for Music

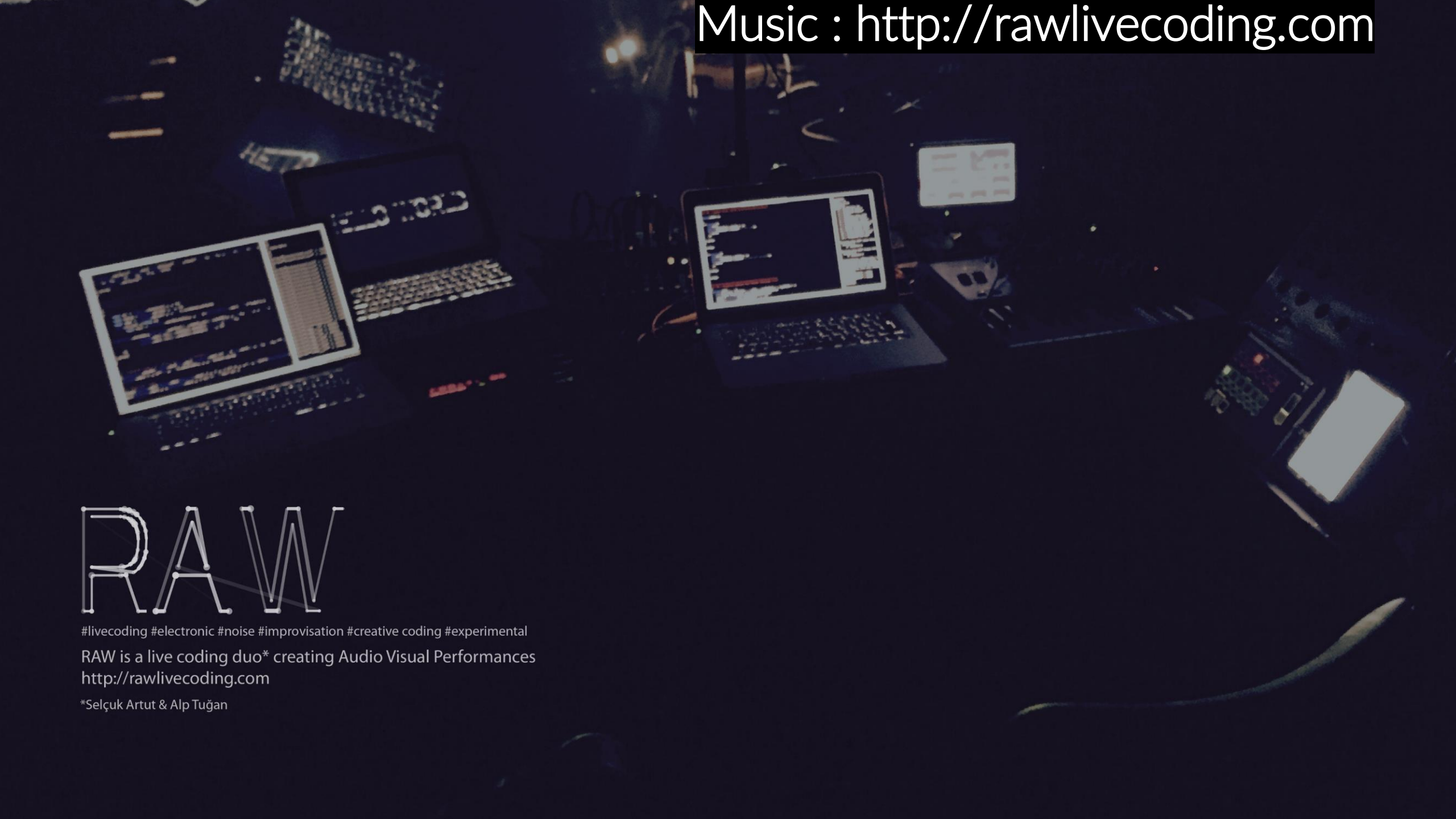
Replikas

Discography
Albums
Köledoyuran 2000
Dadaruhi 2002
Avaz 2005
Film Müzikleri 2006
Zerre 2008
Biz Burada Yok İken 2012
EP No: 1 2013
Box Set: Köledoyuran & Dadaruhi Remastered + EP No.1 2013
Alfred Hitchcock's Blackmail - Live at Istanbul Modern 2014








Music : http://rawlivecoding.com
RAW
#livecoding #electronic #noise #improvisation #creative coding #experimental
RAW is a live coding duo* creating Audio Visual Performances
http://rawlivecoding.com
*Selçuk Artut & Alp Tuğan


RAW

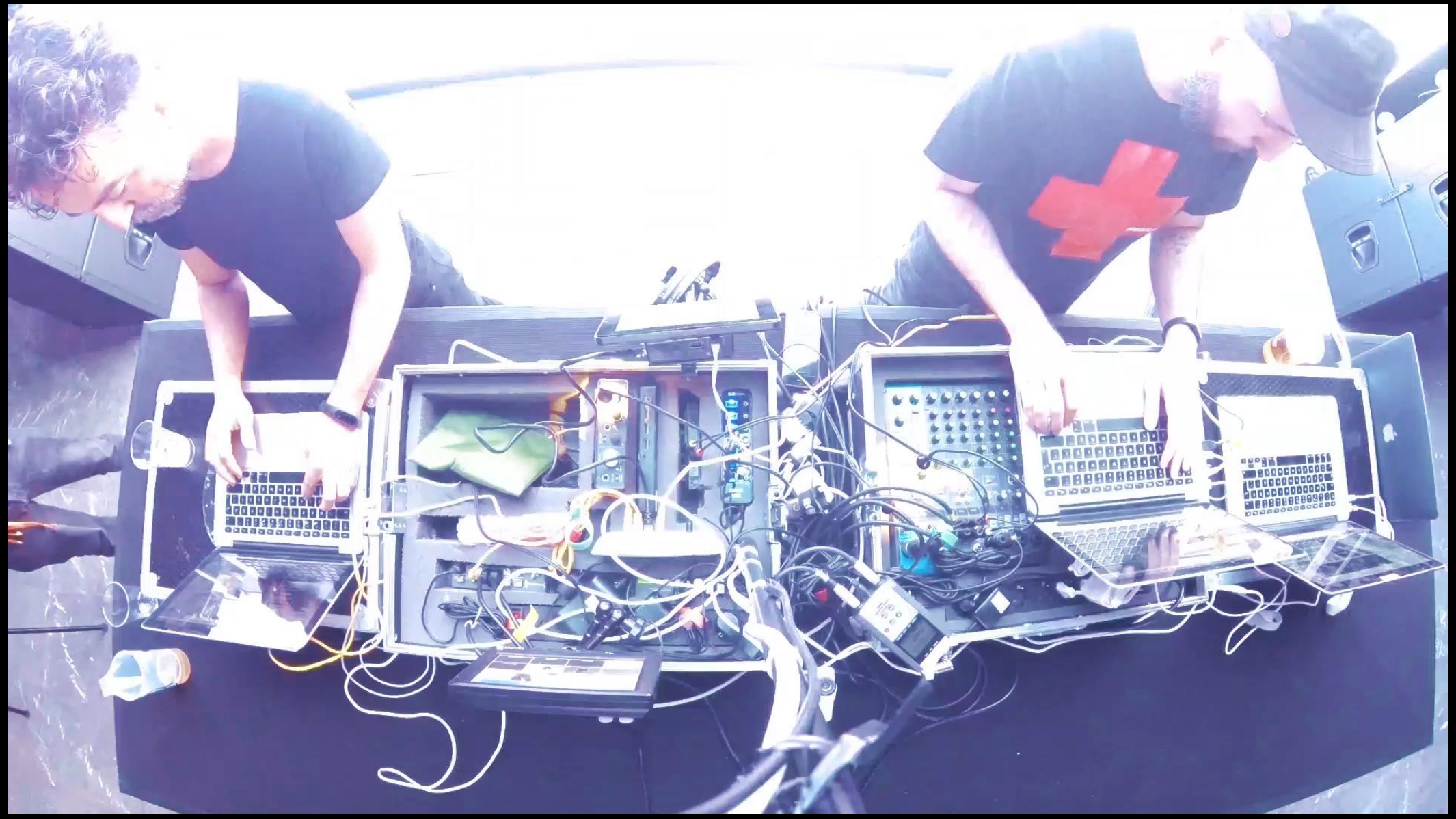
nate
rtive
nant

B

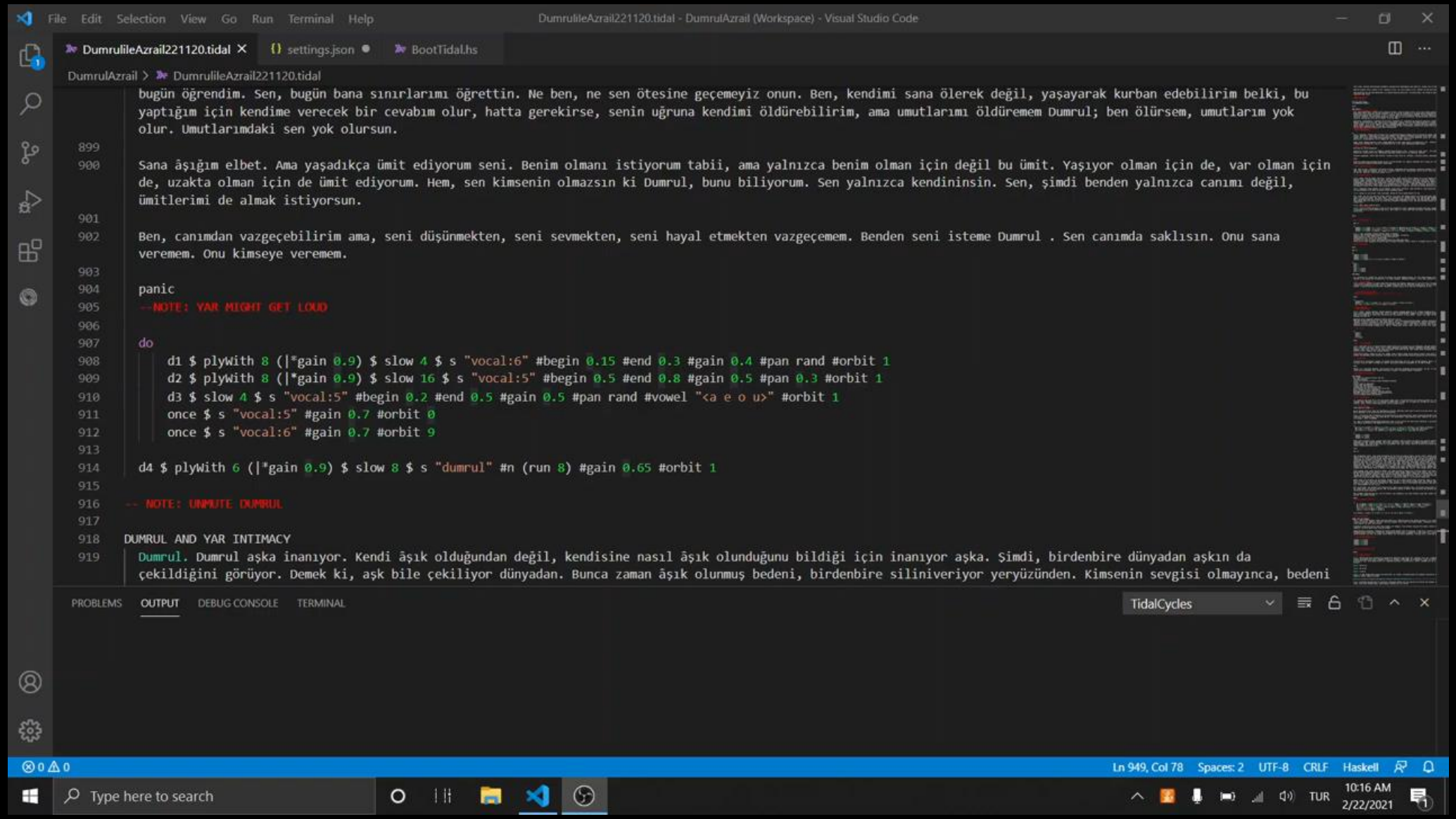
4. Mellifluous
5. Pernicious
6. Dynasia

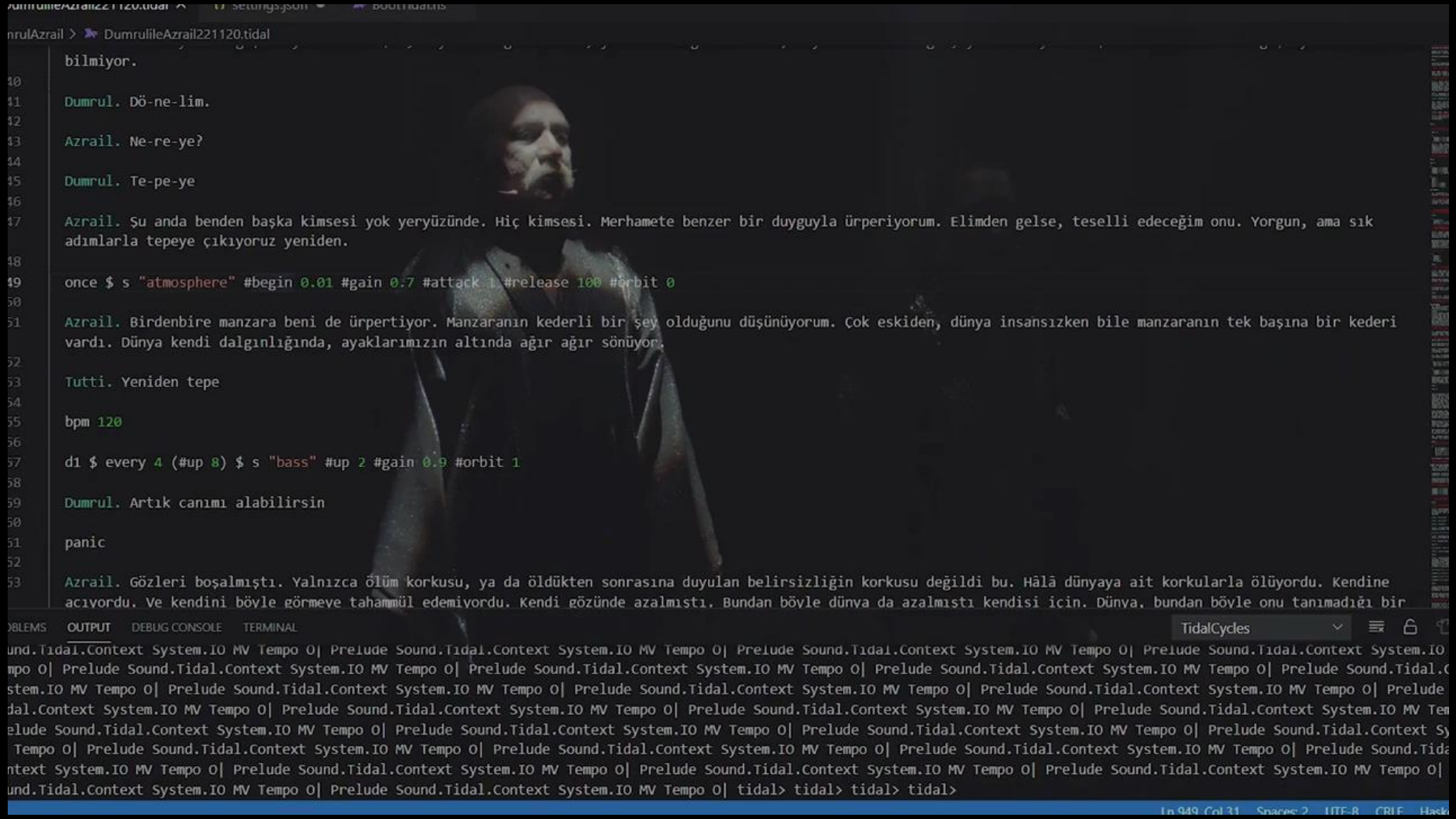
GATE

tidal.com/context/
tidal.com/context/




RAW
GATE


File Edit Selection View Go Run Terminal Help
DumrulileAzrail221120.tidal - DumrulAzrail (Workspace) - Visual Studio Code
DumrulileAzrail221120.tidal
settings.json
BootTidal.hs
DumrulAzrail > DumrulileAzrail221120.tidal
bugün öğrendim. Sen, bugün bana sınırlarımı öğrettin. Ne ben, ne sen ötesine geçemeyiz onun. Ben, kendimi sana ölerek değil, yaşayarak kurban edebilirim belki, bu yaptığım için kendime verecek bir cevabım olur, hatta gerekirse, senin uğruna kendimi öldürebilirim, ama umutlarımı öldüremem Dumrul; ben ölürsem, umutlarım yok olur. Umutlarımdaki sen yok olursun.
Sana âşığım elbet. Ama yaşadıkça ümit ediyorum seni. Benim olmanı istiyorum tabii, ama yalnızca benim olman için değil bu ümit. Yaşıyor olman için de, var olman için de, uzakta olman için de ümit ediyorum. Hem, sen kimsenin olmazsın ki Dumrul, bunu biliyorum. Sen yalnızca kendininsin. Sen, şimdi benden yalnızca canımı değil, ümitlerimi de almak istiyorsun.
Ben, canımdan vazgeçebilirim ama, seni düşünmekten, seni sevmekten, seni hayal etmekten vazgeçemem. Benden seni isteme Dumrul . Sen canımda saklısın. Onu sana veremem. Onu kimseye veremem.
panic
--NOTE: YAR MIGHT GET LOUD
do
d1 $ plyWith 8 (|*gain 0.9) $ slow 4 $ s "vocal:6" #begin 0.15 #end 0.3 #gain 0.4 #pan rand #orbit 1
d2 $ plyWith 8 (|*gain 0.9) $ slow 16 $ s "vocal:5" #begin 0.5 #end 0.8 #gain 0.5 #pan 0.3 #orbit 1
d3 $ slow 4 $ s "vocal:5" #begin 0.2 #end 0.5 #gain 0.5 #pan rand #vowel "<a e o u>" #orbit 1
once $ s "vocal:5" #gain 0.7 #orbit 0
once $ s "vocal:6" #gain 0.7 #orbit 9
d4 $ plyWith 6 (|*gain 0.9) $ slow 8 $ s "dumrul" #n (run 8) #gain 0.65 #orbit 1
-- NOTE: UNMUTE DUMRUL
DUMRUL AND YAR INTIMACY
Dumrul. Dumrul aşka inanıyor. Kendi âşık olduğundan değil, kendisine nasıl âşık olunduğunu bildiği için inanıyor aşka. Şimdi, birdenbire dünyadan aşkın da çekildiğini görüyor. Demek ki, aşk bile çekiliyor dünyadan. Bunca zaman âşık olunmuş bedeni, birdenbire siliniveriyor yeryüzünden. Kimsenin sevgisi olmayınca, bedeni
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
TidalCycles
Ln 949, Col 78 Spaces: 2 UTF-8 CRLF Haskell
Type here to search
TUR
10:16 AM
2/22/2021


nrulAzrail > DumrulileAzrail221120.tidal
bilmiyor.
Dumrul. Dö-ne-lim.
Azrail. Ne-re-ye?
Dumrul. Te-pe-ye
Azrail. Şu anda benden başka kimsesi yok yeryüzünde. Hiç kimsesi. Merhamete benzer bir duyguyla ürperiyorum. Elimden gelse, teselli edeceğim onu. Yorgun, ama sık adımlarla tepeye çıkıyoruz yeniden.
once $ s "atmosphere" #begin 0.01 #gain 0.7 #attack 1 #release 100 #orbit 0
Azrail. Birdenbire manzara beni de ürpertiyor. Manzaranın kederli bir şey olduğunu düşünüyorum. Çok eskiden, dünya insansızken bile manzaranın tek başına bir kederi vardı. Dünya kendi dalgınlığında, ayaklarımızın altında ağır ağır sönüyor.
Tutti. Yeniden tepe
bpm 120
d1 $ every 4 (#up 8) $ s "bass" #up 2 #gain 0.9 #orbit 1
Dumrul. Artık canımı alabilirsin
panic
Azrail. Gözleri boşalmıştı. Yalnızca ölüm korkusu, ya da öldükten sonrasına duyulan belirsizliğin korkusu değildi bu. Hâlâ dünyaya ait korkularla ölüyordu. Kendine acıyordu. Ve kendini böyle görmeye tahammül edemiyordu. Kendi gözünde azalmıştı. Bundan böyle dünya da azalmıştı kendisi için. Dünya, bundan böyle onu tanımadığı bir
OUTPUT
DEBUG CONSOLE
TERMINAL
TidalCycles
tidal> tidal> tidal> tidal>
Ln 949, Col 31
Spaces: 2
UTF-8
CRLF

PART I – Dealing with Audio

What is Live Coding?

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.

Live coding is inclusive and accessible to all. Many live coding environments can be downloaded and used for free, with documentation and examples to get you started and friendly on-line communities to help when you get problems. Popular live coding software includes ChucK, Cyril, extempore, fluxus, impromptu, overtone and supercollider. Environments designed for fast exploration of musical pattern include ixi lang and TidalCycles. Sonic Pi is designed for teaching both music and computer science in classrooms, as well as performing in algoraves. There are also impressively capable web-based live coding environments like gibber and livecodelab. Live patching is live coding with graph-based languages such as the venerable pure-data. It's also possible to livecode with a gamepad, e.g. with the robot oriented Al-Jazari.

Related Links

1. **Sonic Pi Essentials Book** - https://www.raspberrypi.org/magpi/issues/essentials-sonic-pi-v1/
2. **TOPLAP** - http://toplap.org
3. **Live Code Slack** - http://live-code-slack.herokuapp.com/
4. **Algorave** - http://algorave.com
5. **Sonic Pi on Github** - https://github.com/samaaron/sonic-pi
6. **Sonic Pi Google Group** - https://groups.google.com/forum/#!forum/sonic-pi
7. **Sonic Pi on Gitter.im** - https://gitter.im/samaaron/sonic-pi

Algorave

An algorave (from an algorithm and rave) is an event where people dance to music generated from algorithms, often using live coding techniques

https://sonic-pi.net

Intro | Community | Examples | News | Tutorial

Sonic Pi

Welcome to the future of music.

Sonic Pi is a code-based music creation and performance tool.

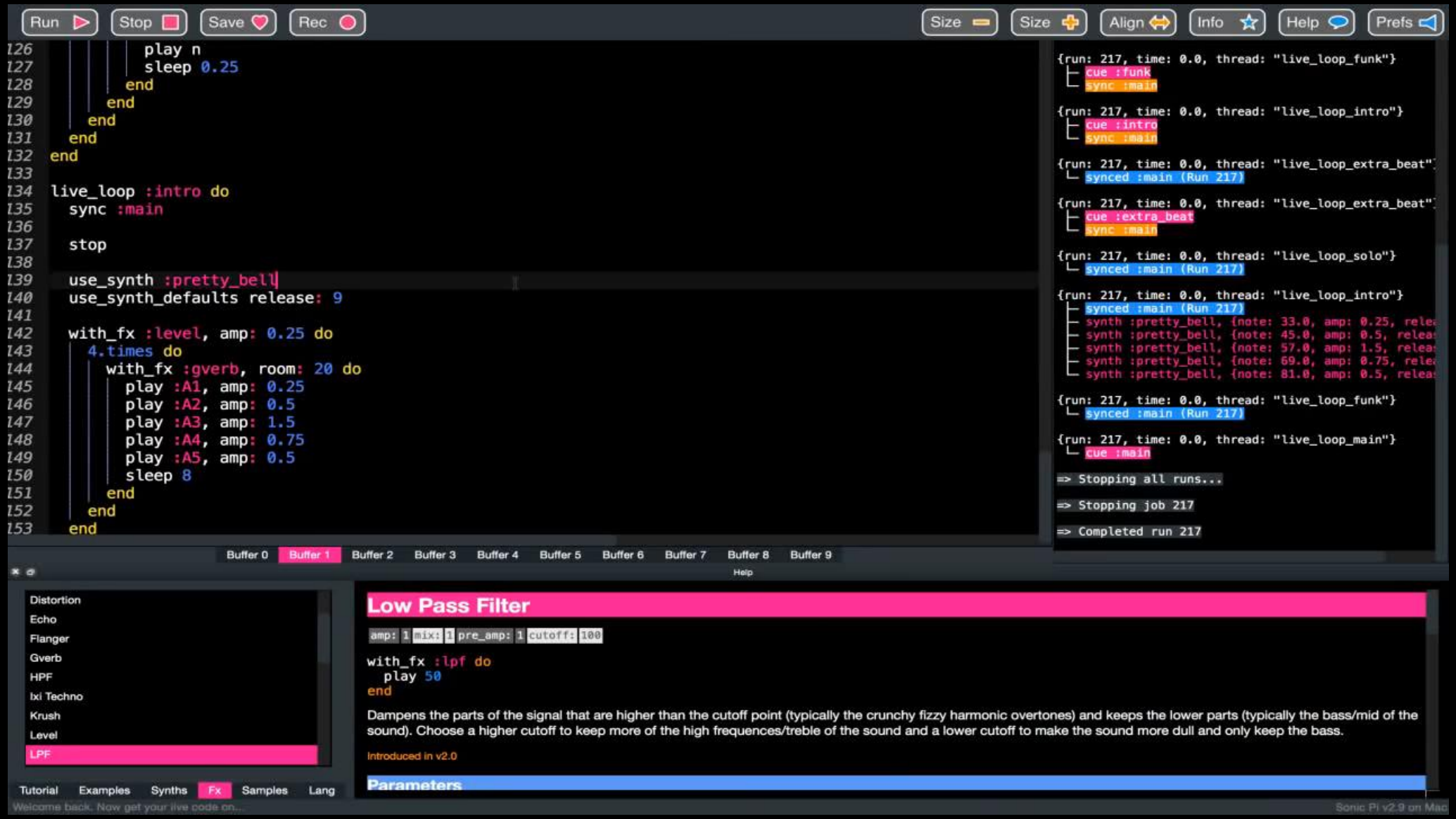
Simple enough for computing and music lessons.

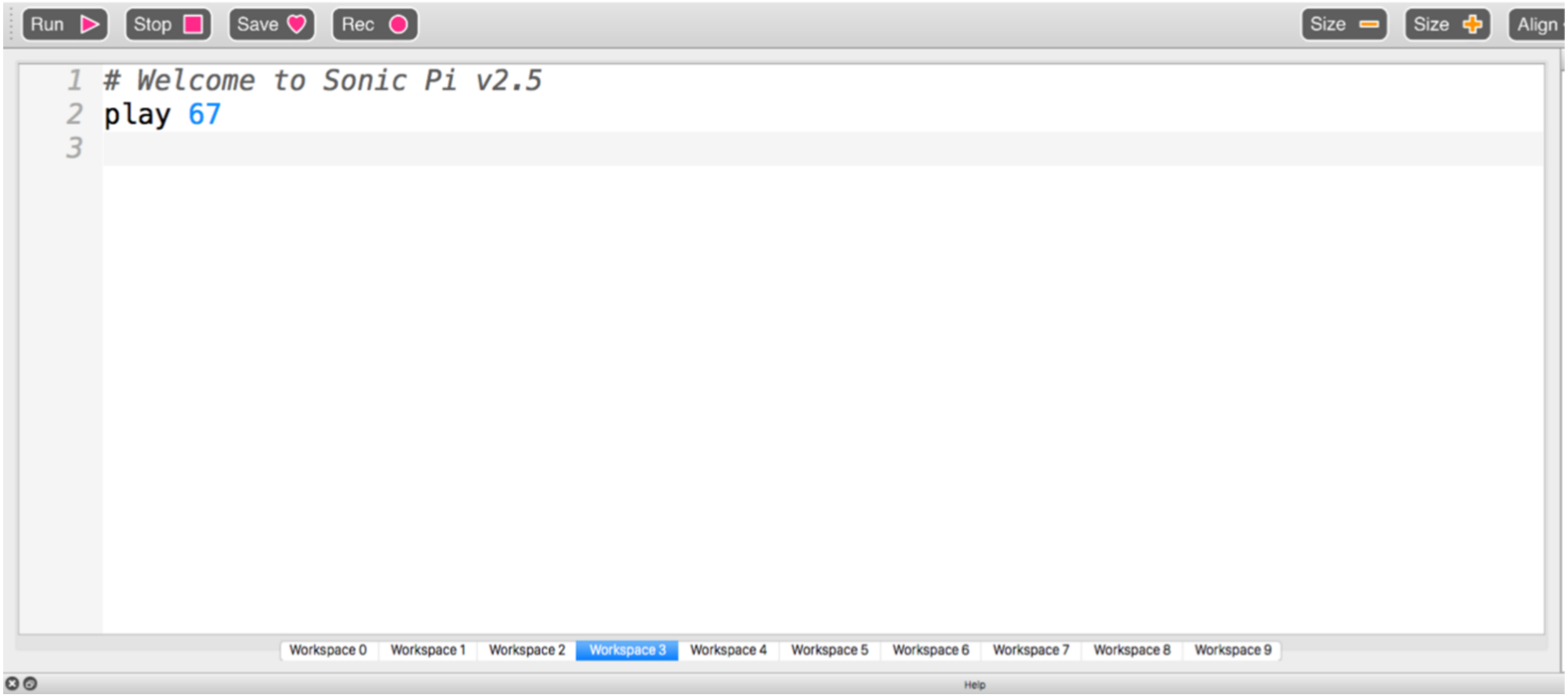
Powerful enough for professional musicians.

Free to download with a friendly tutorial.

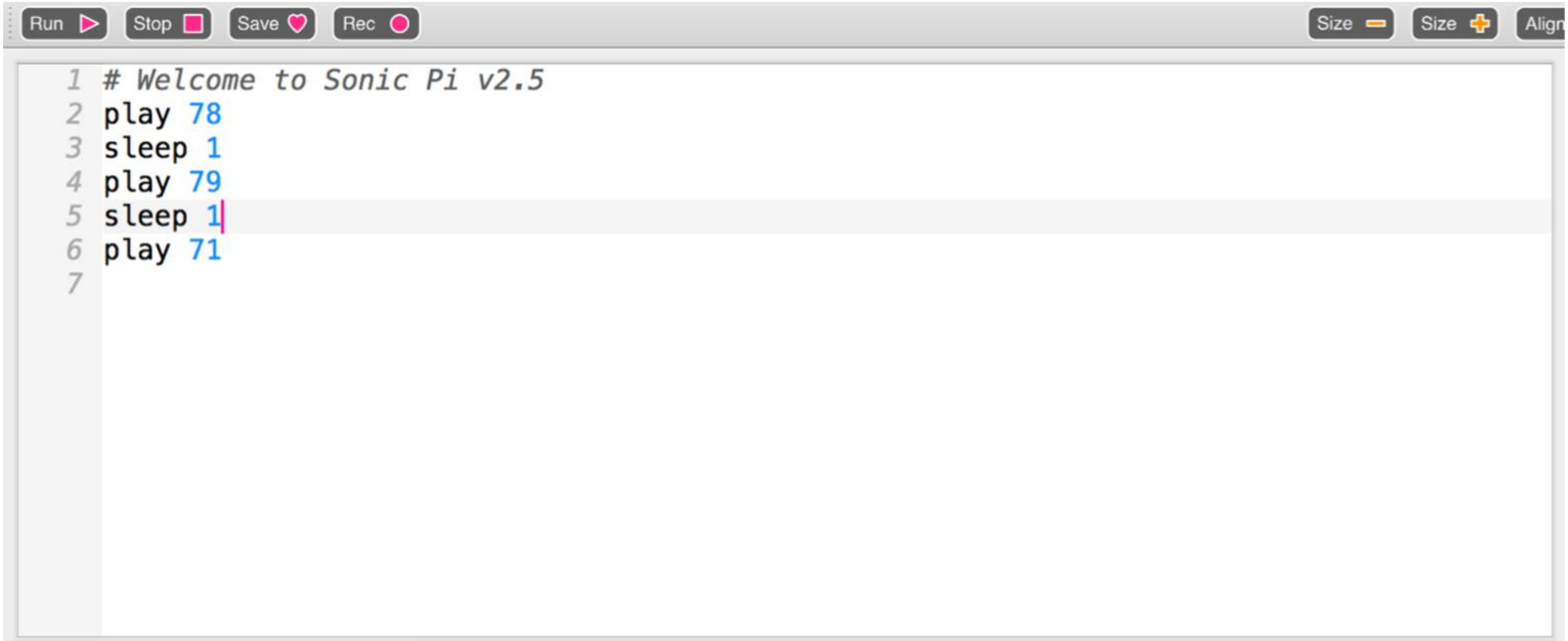
Diverse community of over 1.8 million live coders.

Windows | macOS | Raspberry Pi OS


Run
Stop
Save
Rec
Size
Size
Align
Info
Help
Prefs
play n
sleep 0.25
end
end
end
end
live_loop :intro do
sync :main
stop
use_synth :pretty_bell
use_synth_defaults release: 9
with_fx :level, amp: 0.25 do
4.times do
with_fx :gverb, room: 20 do
play :A1, amp: 0.25
play :A2, amp: 0.5
play :A3, amp: 1.5
play :A4, amp: 0.75
play :A5, amp: 0.5
sleep 8
end
end
end
{run: 217, time: 0.0, thread: "live_loop_funk"}
cue :funk
sync :main
{run: 217, time: 0.0, thread: "live_loop_intro"}
cue :intro
sync :main
{run: 217, time: 0.0, thread: "live_loop_extra_beat"}
synced :main (Run 217)
{run: 217, time: 0.0, thread: "live_loop_extra_beat"}
cue :extra_beat
sync :main
{run: 217, time: 0.0, thread: "live_loop_solo"}
synced :main (Run 217)
{run: 217, time: 0.0, thread: "live_loop_intro"}
synced :main (Run 217)
synth :pretty_bell, {note: 33.0, amp: 0.25, rele
synth :pretty_bell, {note: 45.0, amp: 0.5, relea
synth :pretty_bell, {note: 57.0, amp: 1.5, relea
synth :pretty_bell, {note: 69.0, amp: 0.75, rele
synth :pretty_bell, {note: 81.0, amp: 0.5, relea
{run: 217, time: 0.0, thread: "live_loop_funk"}
synced :main (Run 217)
{run: 217, time: 0.0, thread: "live_loop_main"}
cue :main
=> Stopping all runs...
=> Stopping job 217
=> Completed run 217
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
Welcome back. Now get your live code on...
Sonic Pi v2.9 on Mac


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

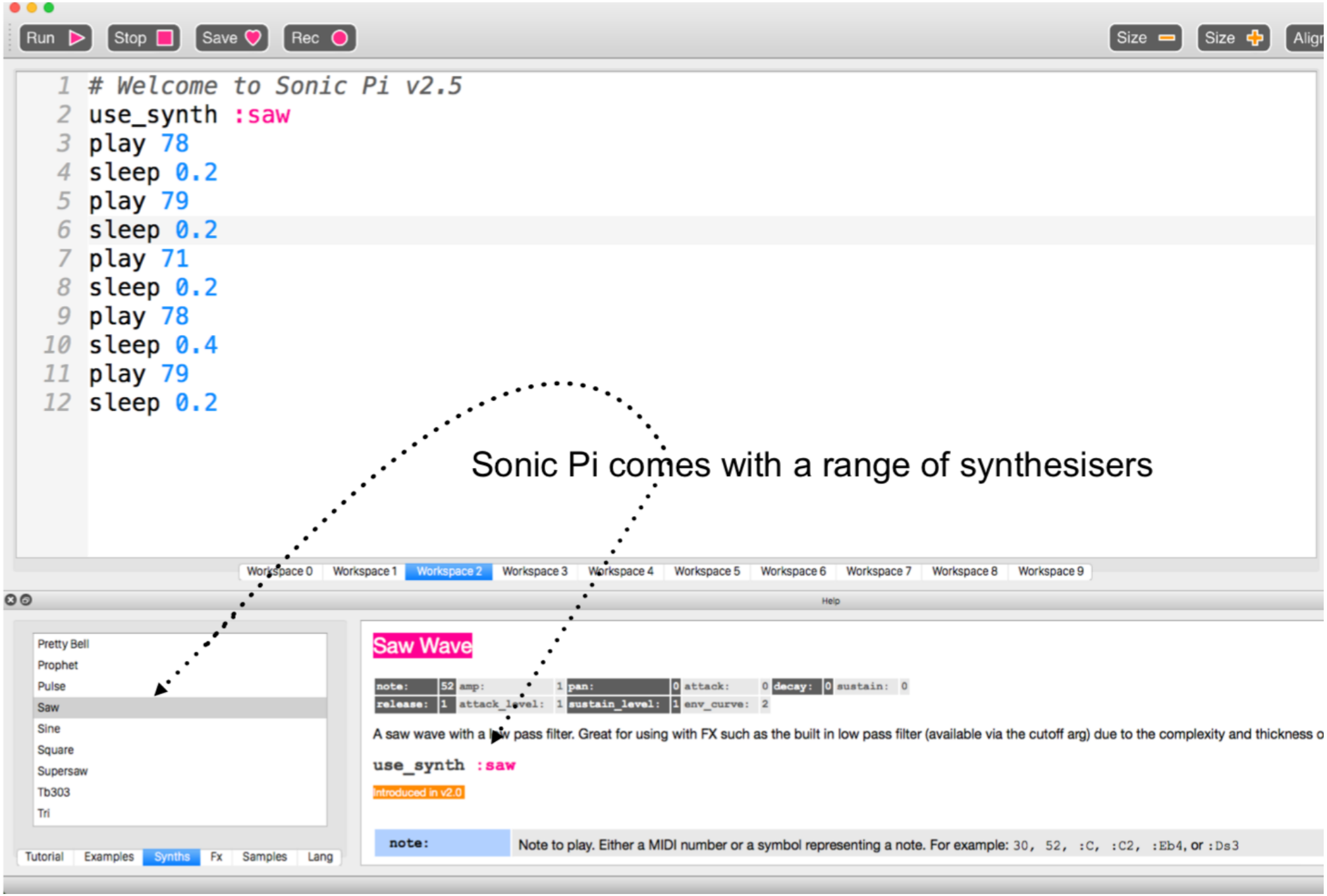
Run Stop Save Rec Size Size Align

```
# 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				

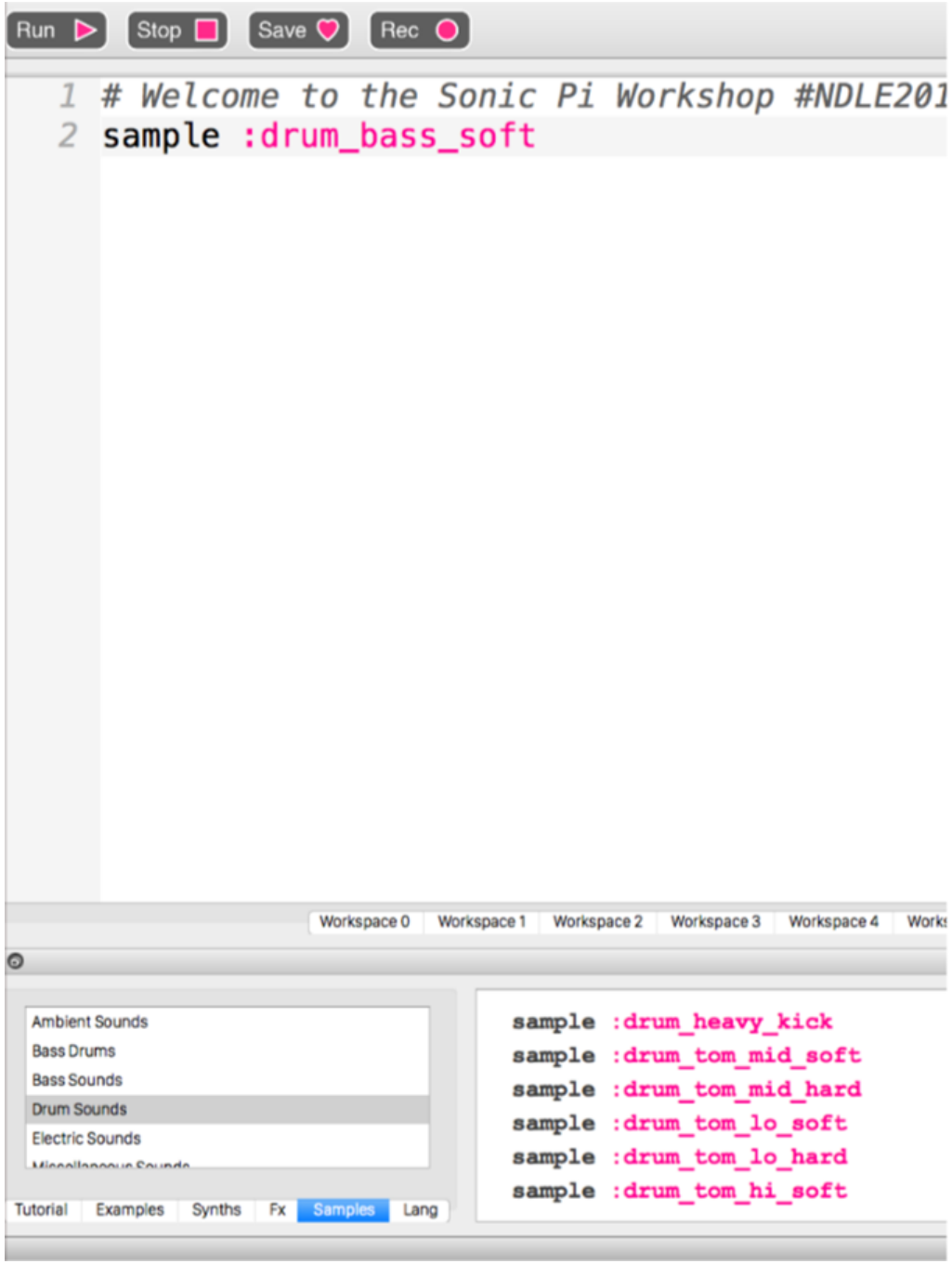
Run Stop Save Rec Size Size Alig

```
1  # Welcome to Sonic Pi v2.5
2  play 78
3  sleep 0.2
4  play 79
5  sleep 0.2
6  play 71
7  sleep 0.2
8  play 78
9  sleep 0.4
10 play 79
11 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

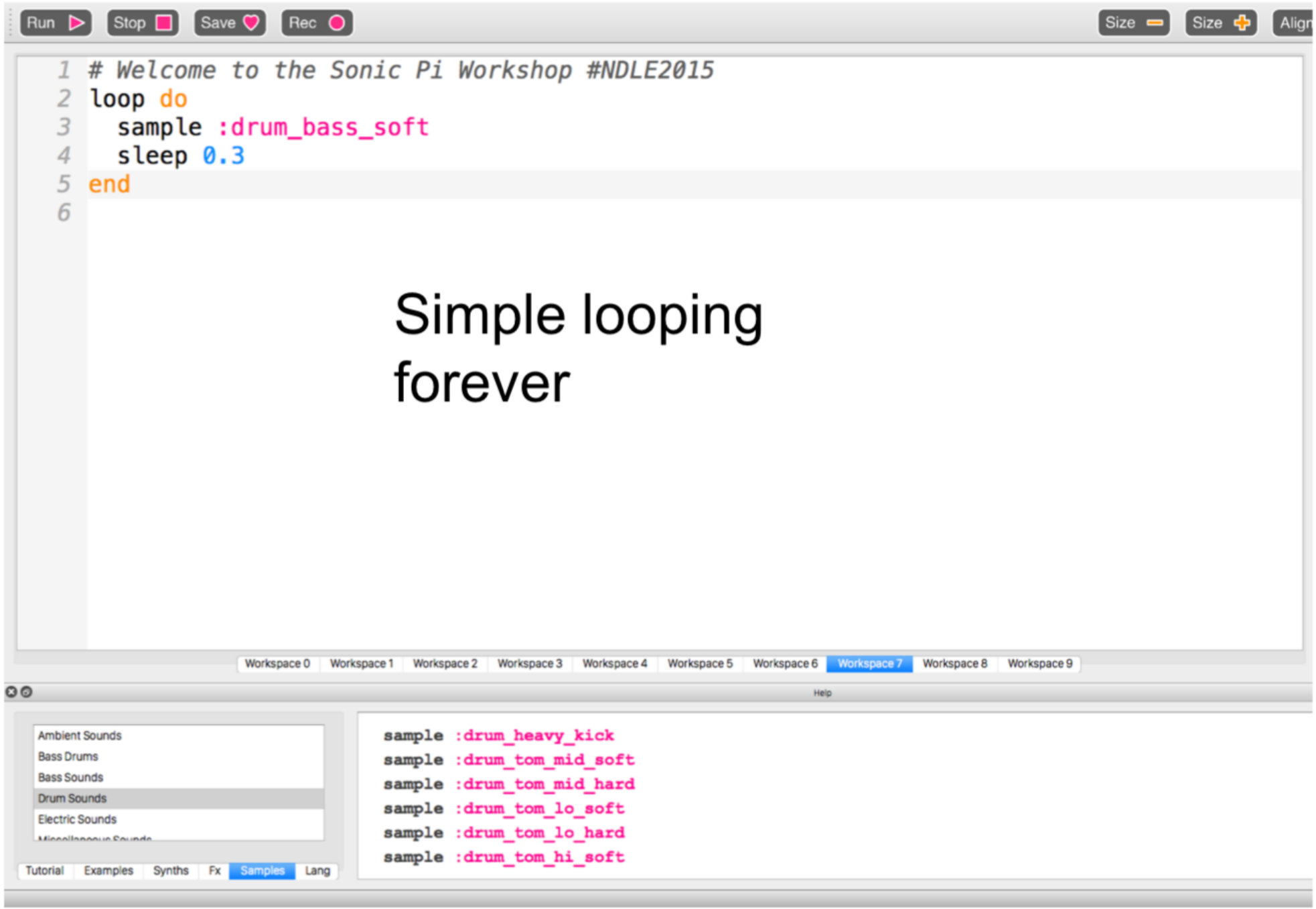
Run Stop Save Rec Size Size Align

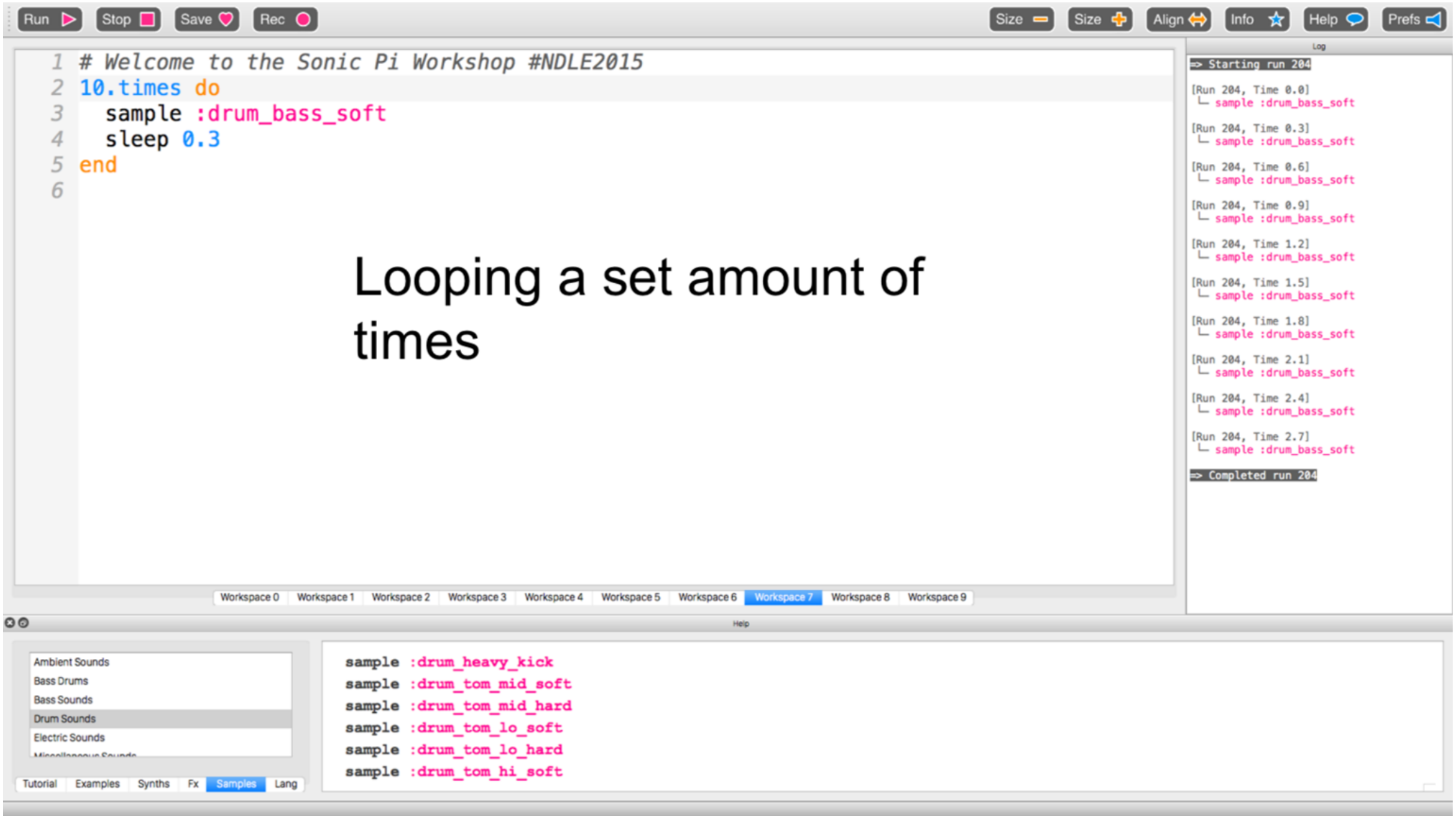
```
# Welcome to the Sonic Pi Workshop #NDLE2015
use_synth :saw
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 78
sleep 0.4
play 79
sleep 0.8
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 79
```



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)

PART II – Dealing with Visuals

Most Common Creative Coding Platforms

Processing

Openframeworks

Cinder

p5.js

Touchdesigner

MaxMSP/Jitter

Vvvv

Most Common Creative Coding Platforms

Processing

Openframeworks

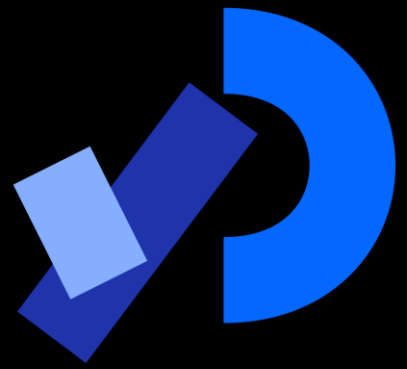
Cinder

p5.js

Touchdesigner

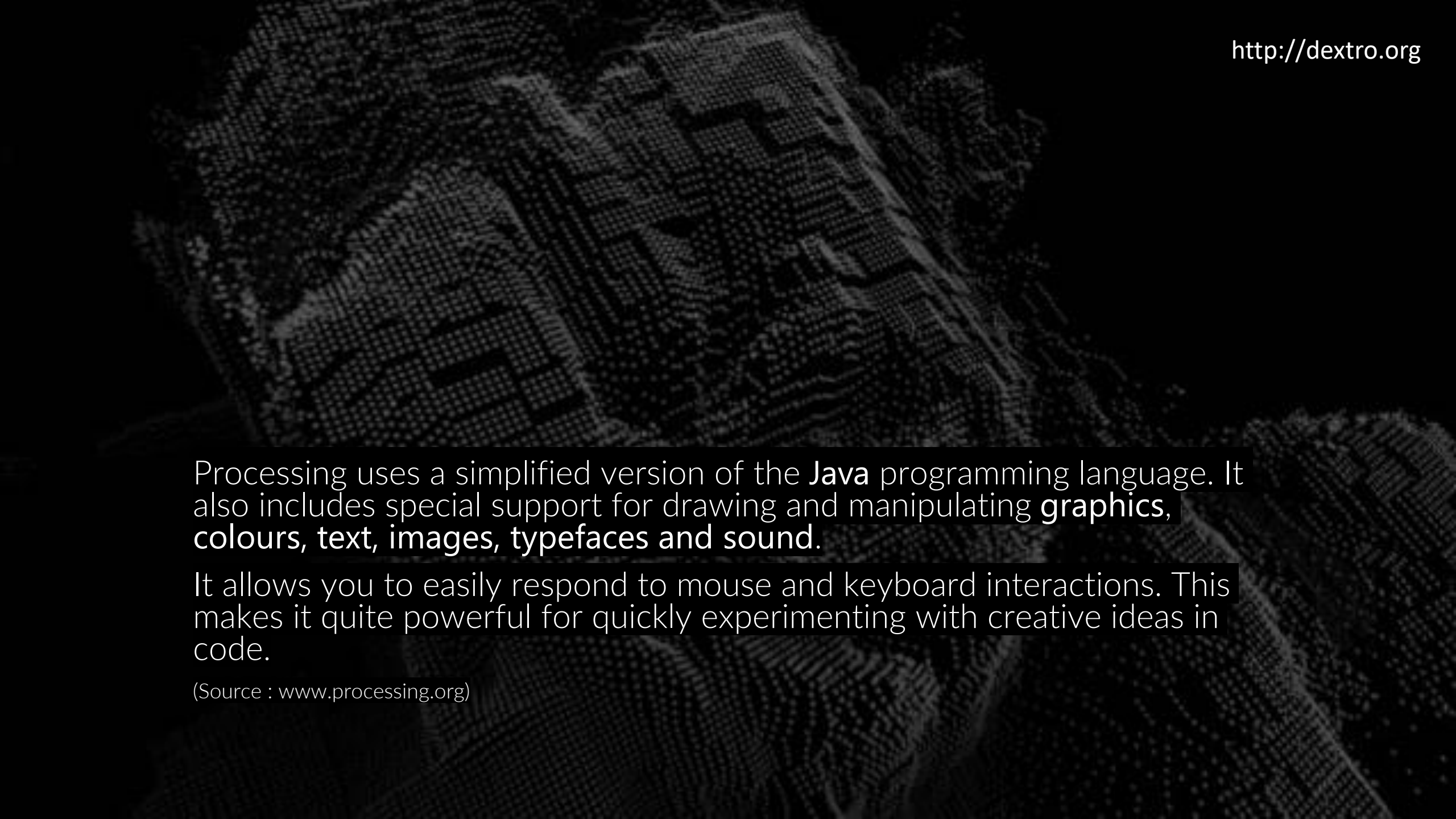
MaxMSP/Jitter

Vvvv



Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

www.processing.org

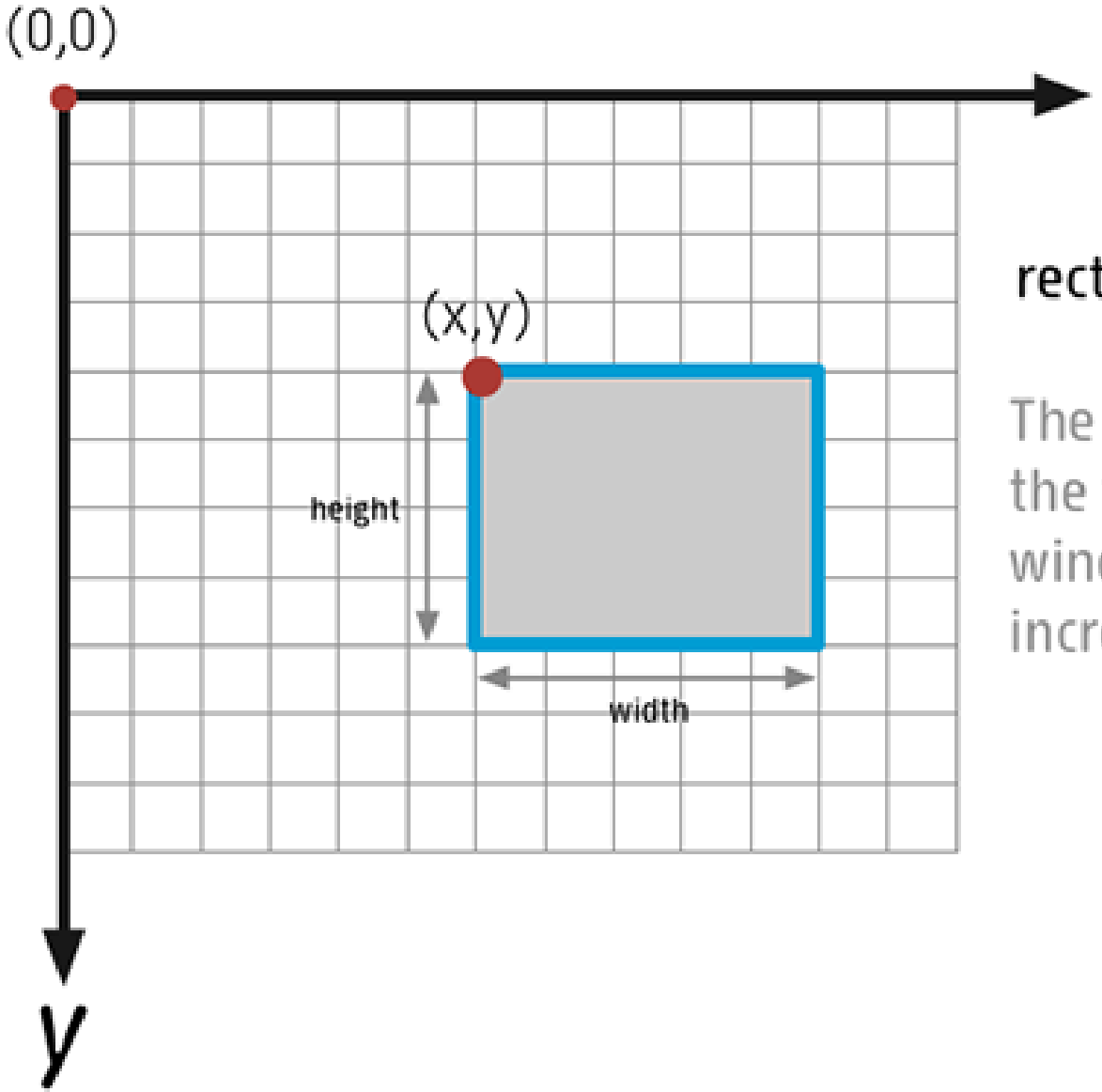


Processing uses a simplified version of the **Java** programming language. It also includes special support for drawing and manipulating **graphics**, **colours, text, images, typefaces and sound**.

It allows you to easily respond to mouse and keyboard interactions. This makes it quite powerful for quickly experimenting with creative ideas in code.

(Source : www.processing.org)

SCREEN COORDINATES IN PROCESSING



rect(x,y,width,height)

The coordinate origin (0,0) is at the top left of the display window. Increasing *x* moves right, increasing *y* moves down.

Analyzing Processing Coding Environment

```
void setup() {
        size(500,500);
}
void draw() {
        background(0);
        ellipse(250,250,100,100);
}
```

size(width, height, optional : Renderer)

size(640,480,JAVA2D);	The standard renderer, this is used if nothing else has been specified
size(640,480,P2D);	Processing 2D Renderer, this is quicker but less accurate
size(640,480,P3D);	Processing 3D Renderer, this is quick and its reliable on the Web
size(640,480, OPENGL);	OpenGL renderer, this uses OpenGL-compatible graphic hardware

Using Comments

Comments are to be used by humans only. So computer will ignore those lines, and compile the uncommented lines.

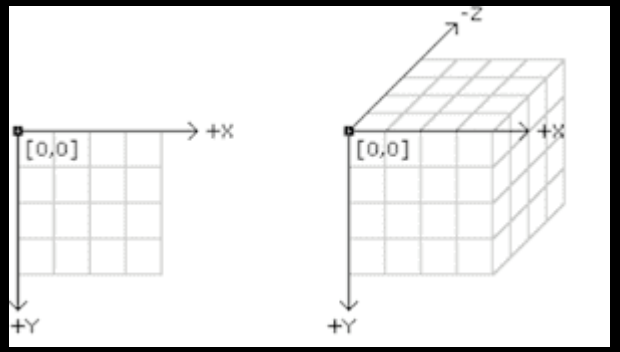
Comment Types

Block Comment : /* ..... */ (Using forward slash asterix .... asterix forward slash)

Line Comment : // (Using two forward slash)

Alternative usage : You may want to ignore a block or a line of code to check for debugging.

Drawing Basic Shapes



Some simple shapes;

point : https://processing.org/reference/point_.html

ellipse : https://processing.org/reference/ellipse_.html

rect : https://processing.org/reference/rect_.html

line : https://processing.org/reference/line_.html

arc : https://processing.org/reference/arc_.html

vertex : https://processing.org/reference/vertex_.html

Colors

color() : Creates colors for storing in variables of the color datatype. The parameters are interpreted as RGB or HSB values depending on the current colorMode(). The default mode is RGB values from 0 to 255 and, therefore, color(255, 204, 0) will return a bright yellow color.

Syntax

color(gray)

color(gray, alpha)

color(v1, v2, v3)

color(v1, v2, v3, alpha)

Data Types, Variables

In order to create dynamic structures, we are required to have data carriers that can change its value at a particular moment. So variables are **great** tools for **Creative Coding.**

When defining a variable, the user must specify its type, meaning what kind of information it will save.

Variable Types

boolean myBoolean = true;

int myInteger = 5;

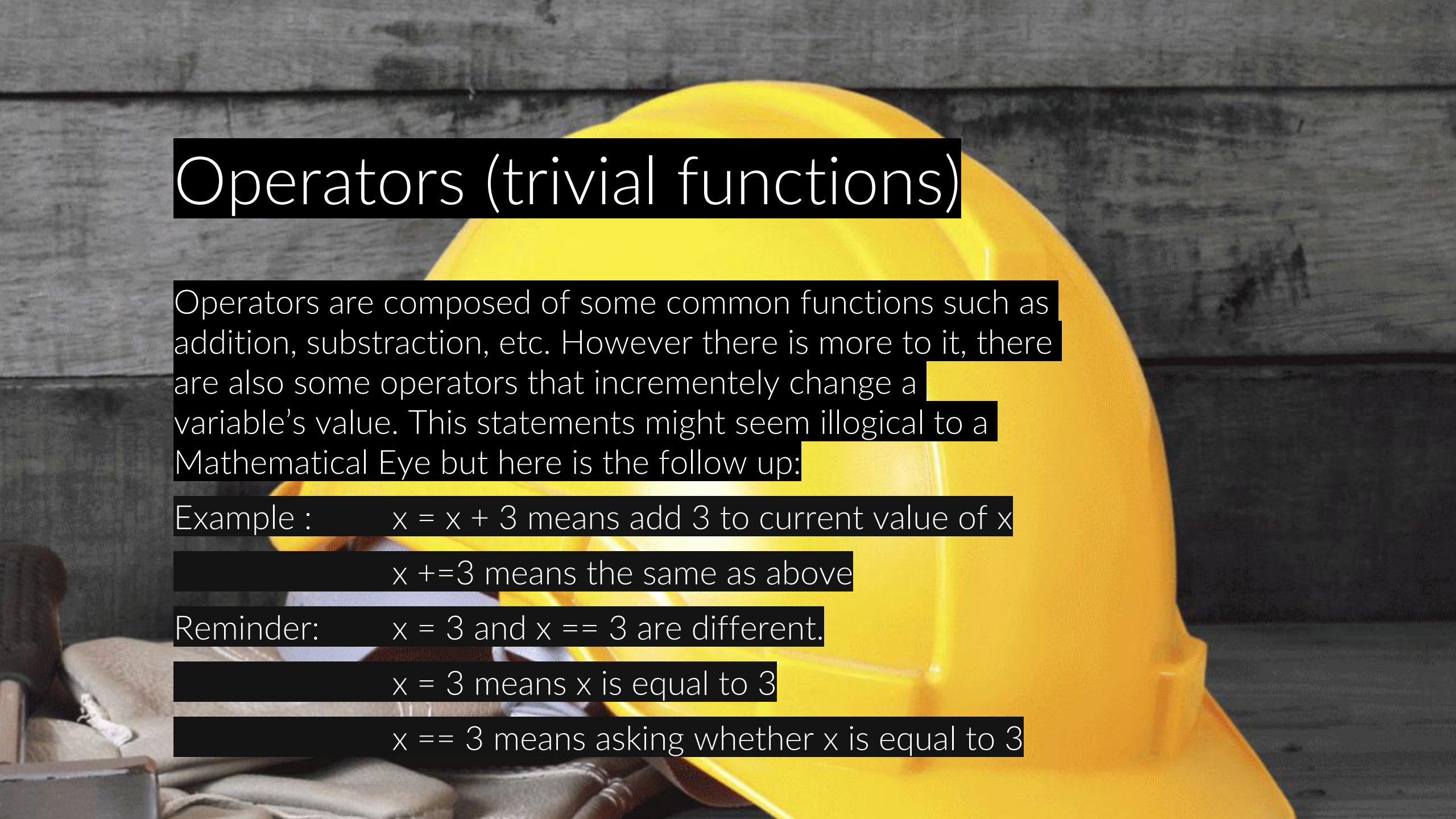
float myFloat = 3.14;

char myChar = ‘A’;

String myString = ‘’this is my string’’

Here are the basic types in Processing:

Type	Explanation	Example
`boolean`	boolean - has only two values: true or false	`boolean isOpen = true;`
`int`	integer (whole number, positive or negative, including 0)	`int lucky = 7;`
`float`	floating-point number (32 bit)	`float balance = -232.5149;`
`char`	single character	`char middleInitial = 'P';`
`String`	string of characters (text)	`String yourName = "Abigail";`



Operators (trivial functions)

Operators are composed of some common functions such as addition, substraction, etc. However there is more to it, there are also some operators that incrementely change a variable's value. This statements might seem illogical to a Mathematical Eye but here is the follow up:

Example : x = x + 3 means add 3 to current value of x

x +=3 means the same as above

Reminder: x = 3 and x == 3 are different.

x = 3 means x is equal to 3

x == 3 means asking whether x is equal to 3

result stored in `area`.

Here's a list of some useful operators:

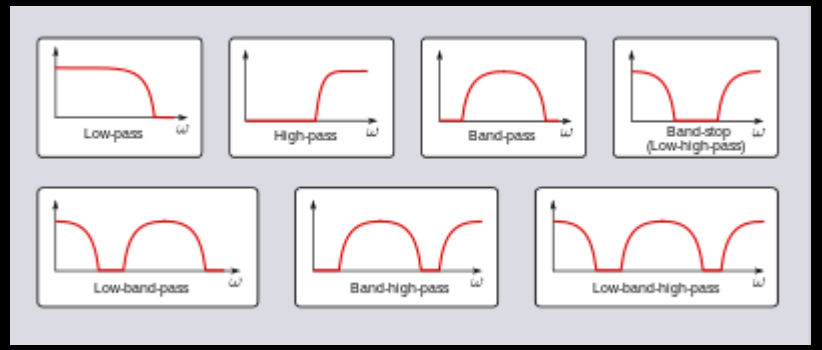
Operator	Explanation	Example
+	addition	`x + y`
-	subtraction	`x - 5`
*	multiplication	`radius * PI`
/	division	`rise / run`
%	modulus (remainder)	`8 % 3`
+=	add assign	`y += 4.3`
-=	subtract assign	`height -= x`
*=	multiply assign	`interest *= 6`

PART III – Establishing links between Audio and Visuals

PART III – Establishing links between Audio and Visuals

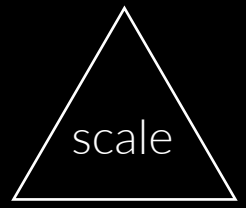
Two Approaches may apply
1-Frequency Mapping
2-Sound Mapping

1-Frequency Mapping

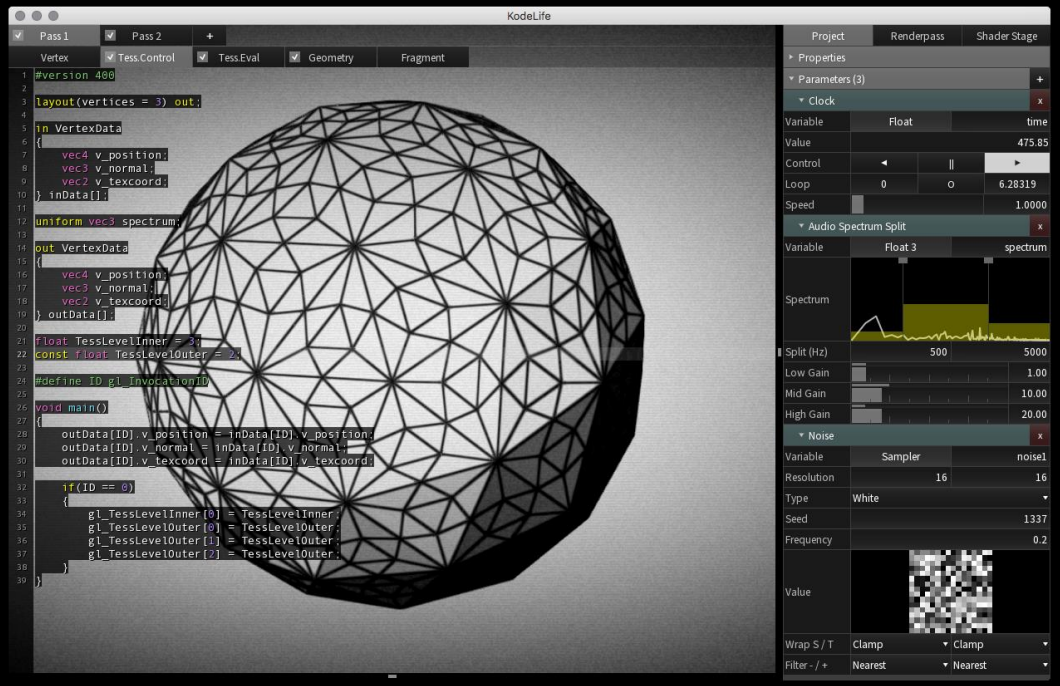


bass

treble



1-Frequency Mapping

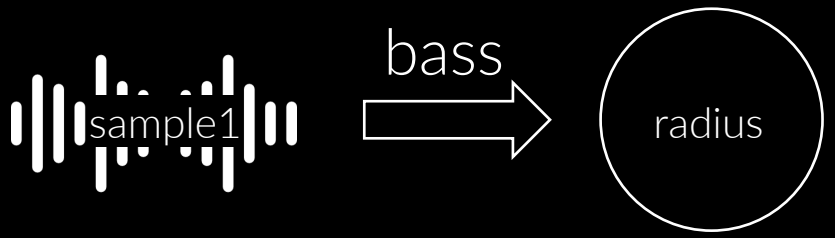


Example: https://hexler.net/kodelife

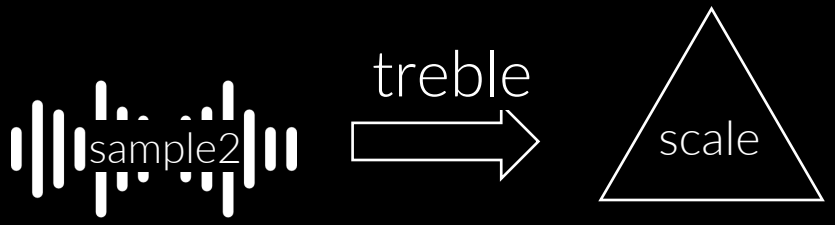
Brasil - Santé Les Amis (Official Music Video) – YouTube

BUENOS MUCHACHOS-Amanecer Buho [Full Album] - YouTube

2-Sound Mapping



Direct mapping between building sound blocks and visual elements

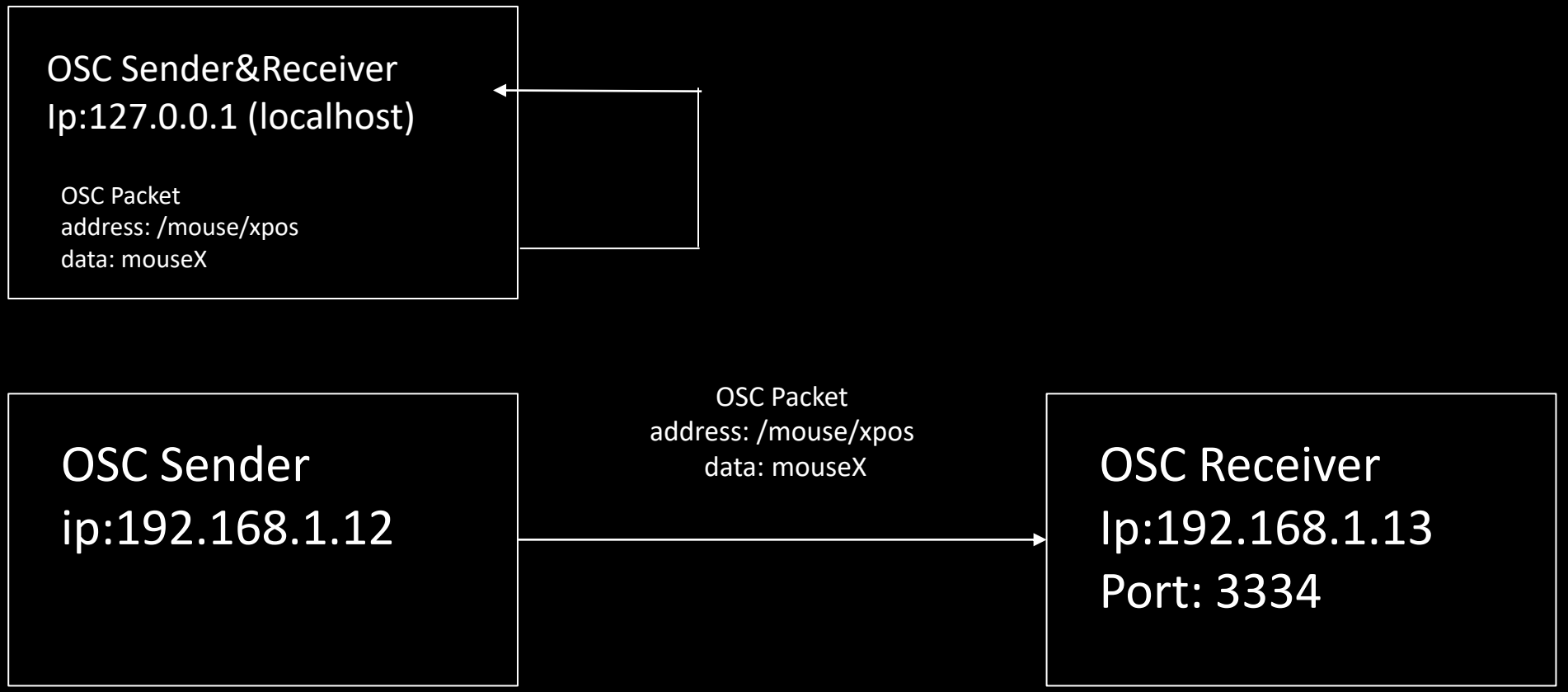


OSC (Open Sound Control)

- 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



All OSC data is composed of the following fundamental data types:

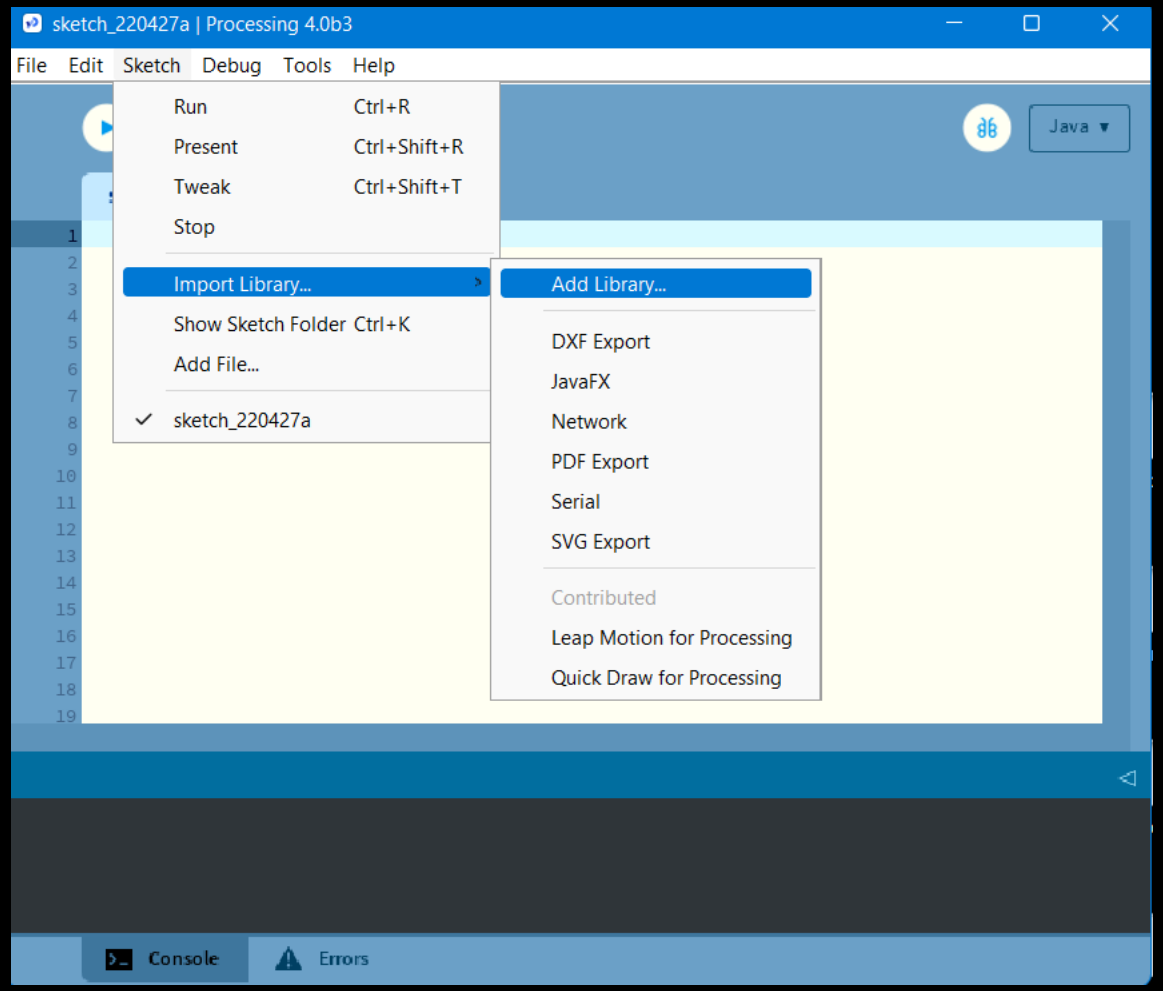
Int32 i.e. 3, 5, 188

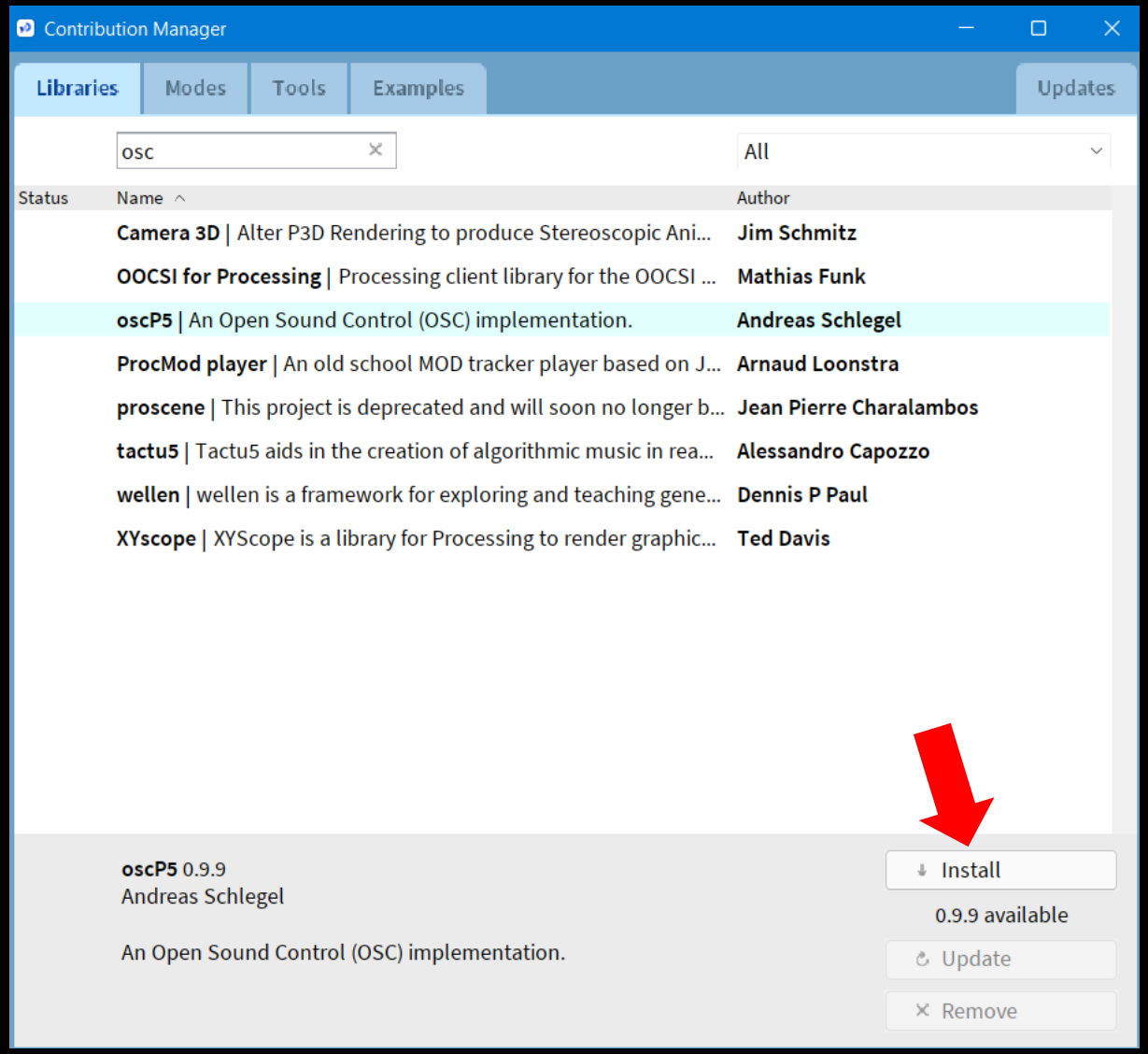
Float32 3.4, 2.7, 56.8

OSC-string “hello world”

The unit of transmission of OSC is an OSC Packet. Every OSC Packet requires an address and a data information.

Processing & OSC





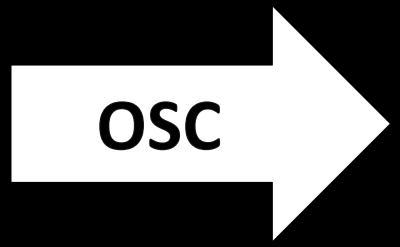
Processing & OSC

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;

void setup() {
  size(400, 400);
  frameRate(25);
  // start oscP5, listening for incoming
//messages at port 8000
  oscP5 = new OscP5(this, 8000);
}
```

```
/* incoming osc message are forwarded to the oscEvent method. */
void oscEvent(OscMessage theOscMessage) {
  /* print the address pattern and the typetag of the received OscMessage */
  print("### received an osc message.");
  print(" addrpattern: "+theOscMessage.addrPattern());
  println(" typetag: "+theOscMessage.typetag());
}
```

Mapping data

Send data from Sonic Pi such as:
- Amp
- Panning
- Pitch
- etc

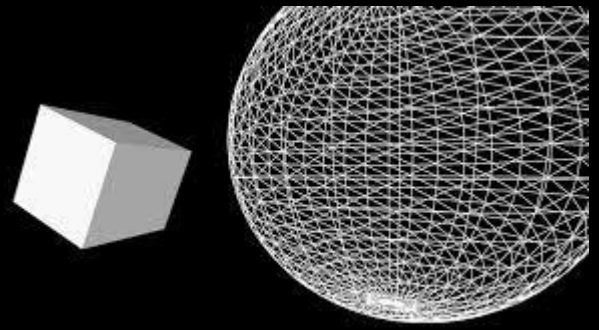


Receive data in Processing use such as:
- Radius
- Color
- Positioning
- etc

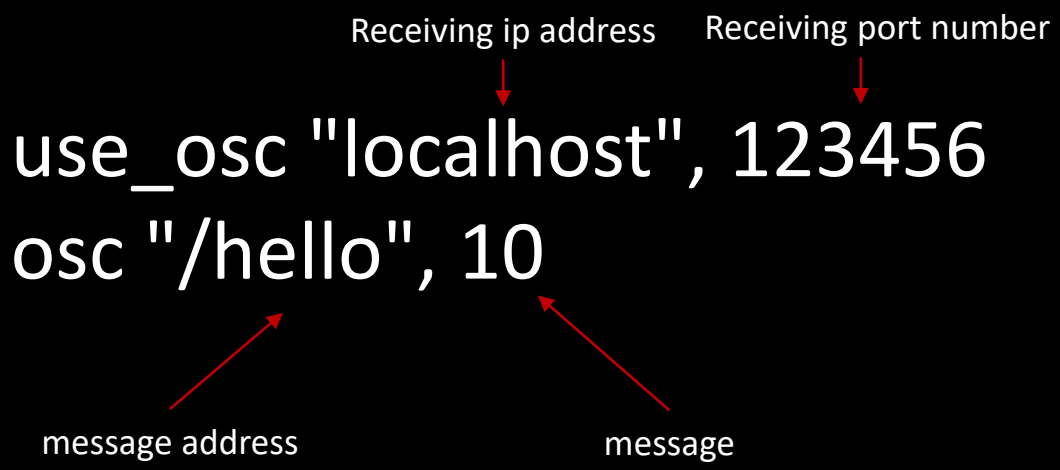
Processing – 3d primitives & Transformations

3d Primitives

`box()`	A box is an extruded `rectangle`
`sphereDetail()`	Controls the detail used to render a sphere by adjusting the number of vertices of the sphere mesh
`sphere()`	A sphere is a hollow ball made from tessellated triangles



Sonic Pi & OSC format



Sonic Pi & OSC

Let’s generate dynamic data

List : [50, 55, 62]

Example: choose([50, 55, 62])

Articulation

Example: Math.sin(tick)

Randomization

Example: rrand(0.5, 1)

SonicPi & Processing & OSC

Sender

```
# Welcome to Sonic Pi
use_osc "localhost",8000

live_loop :myLoop do
  ap = [0.75, 0.95, 0.85, 1].choose
  osc "/sphere", ap
  sample :bd_808, amp:ap
  sleep 1
end
```

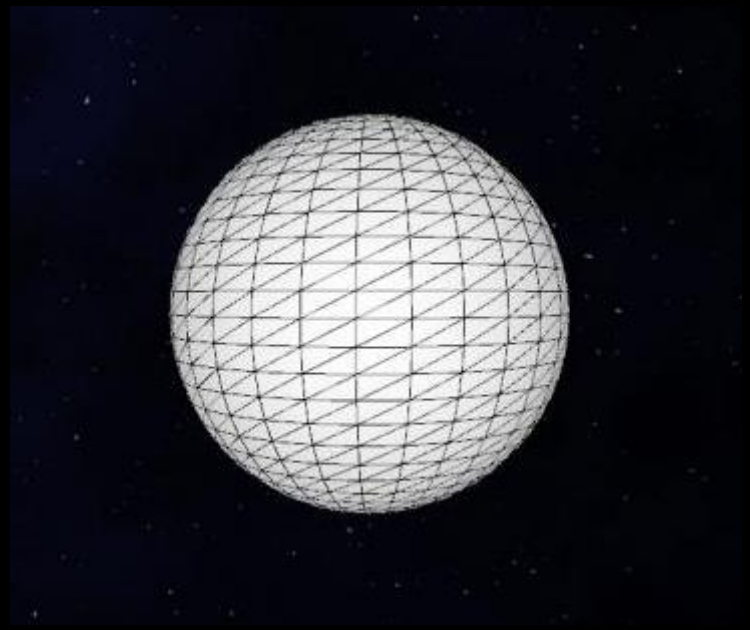
Receiver

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;

void setup() {
 size(400, 400);
 frameRate(25);
 // start oscP5, listening for incoming
//messages at port 8000
 oscP5 = new OscP5(this, 8000);
}

/* incoming osc message are forwarded to the oscEvent method. */
void oscEvent(OscMessage theOscMessage) {
 /* print the address pattern and the typetag of the received OscMessage */
 print("### received an osc message.");
 print(" addrpattern: "+theOscMessage.addrPattern());
 println(" typetag: "+theOscMessage.typetag());
}
```

SonicPi & Processing & OSC

Receiver

```
void oscEvent(OscMessage theOscMessage) {

if(theOscMessage.checkAddrPattern("/kick")==true) {
    kickAmp = theOscMessage.get(0).floatValue();
    rSphere = 100 * kickAmp; // scale upto 100 px
  }
}
```



Processing – Lineer Interpolation

Syntax	`lerp(start, stop, amt)`		
Parameters	**start**	(float)	first value
	stop	(float)	second value
	amt	(float)	float between 0.0 and 1.0
Return	`float`		

```
rSphere = lerp(rSphere, 50,0.05);
```

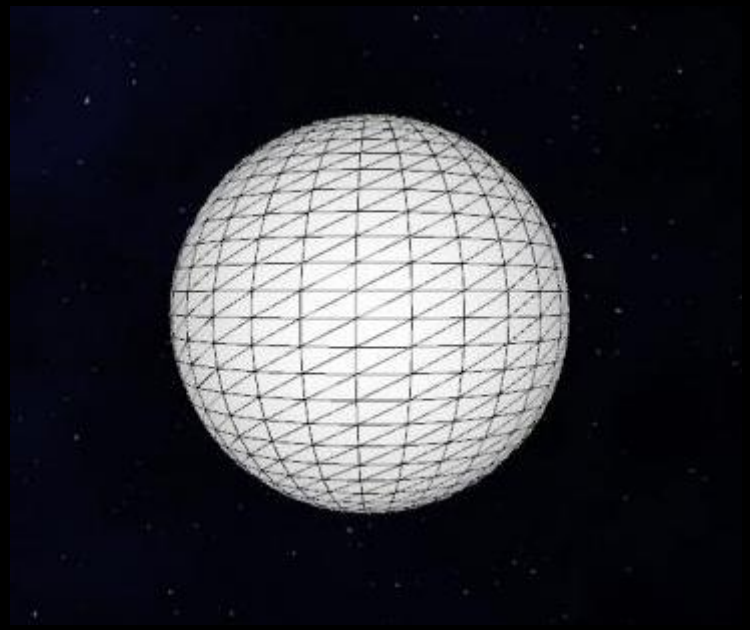
```
if(theOscMessage.checkAddrPattern("/kick")==true) {
   kickAmp = theOscMessage.get(0).floatValue();
}
```

Processing / p5.js Tutorial: What is lerp? (Linear Interpolation): https://www.youtube.com/watch?v=8uLVnM36XUc

SonicPi & Processing & OSC

Receiver

```
void draw() {
  background(0);
  pushMatrix();
  translate(width*0.5,height*0.5);
  noFill();
  rSphere = lerp(rSphere, 50,0.05);
  sphere(rSphere);
  popMatrix();
}
```



SonicPi & Processing & OSC

Receiver: use pan value

```
void oscEvent(OscMessage theOscMessage) {

  if(theOscMessage.checkAddrPattern("/hihat")==true) {
    hihatPan = theOscMessage.get(0).floatValue();
    xCube = width * 0.5 * hihatPan;
  }

}
```

```
void draw() {
  background(0);
  pushMatrix();
  translate(width*0.5,height*0.5);
  noFill();
  translate(xCube,0);
  box(60);
  popMatrix();
}
```

Thanks.
Be in touch!

Facebook/selcuk.artut
Instagram: selcukartut
Web: www.selcukartut.com
Email: selcukartut@gmail.com