Verifying and timing concurrent instruments

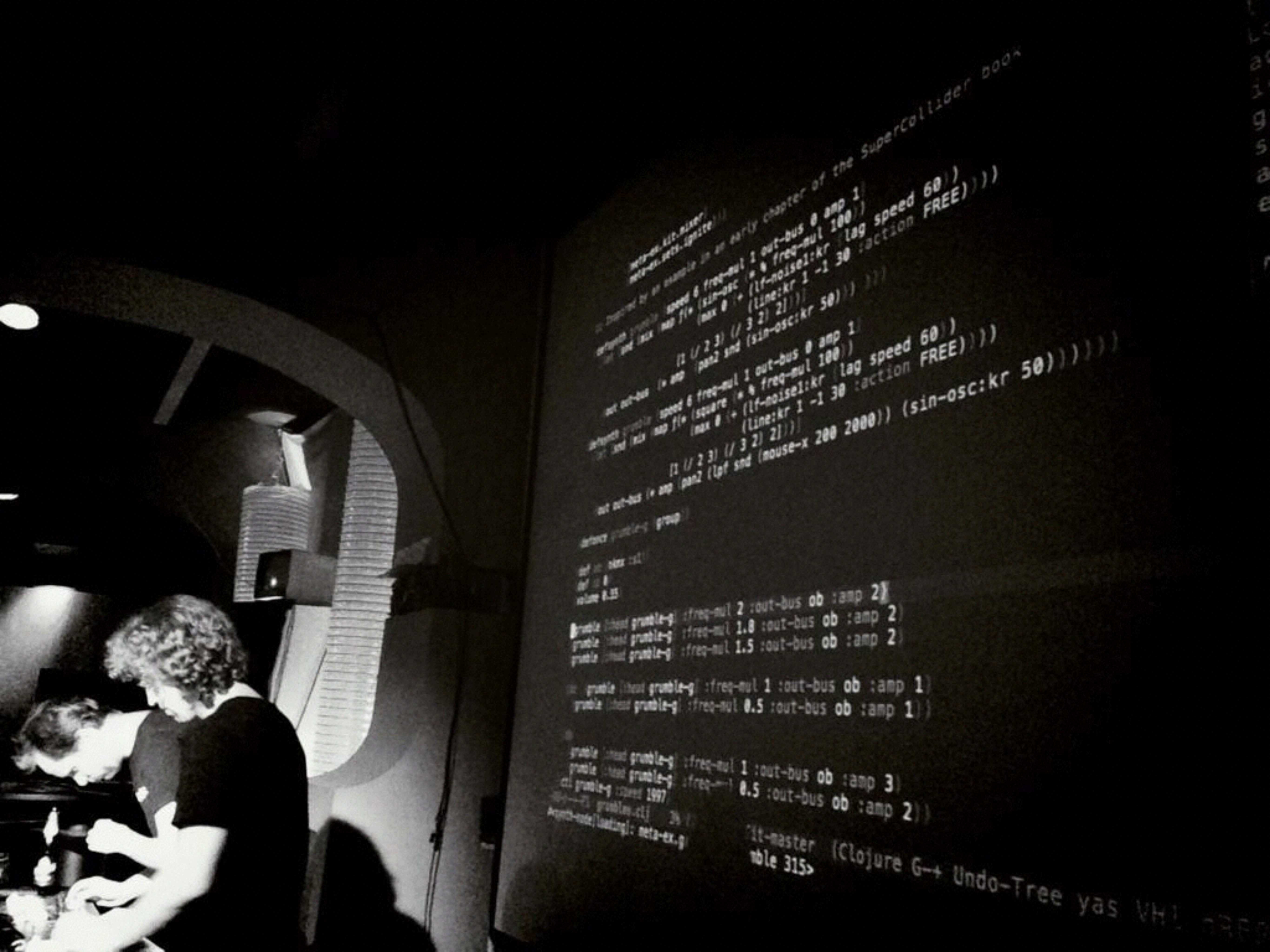
Dominic Orchard

thanks to Sam Aaron for some of these slides & inspiration

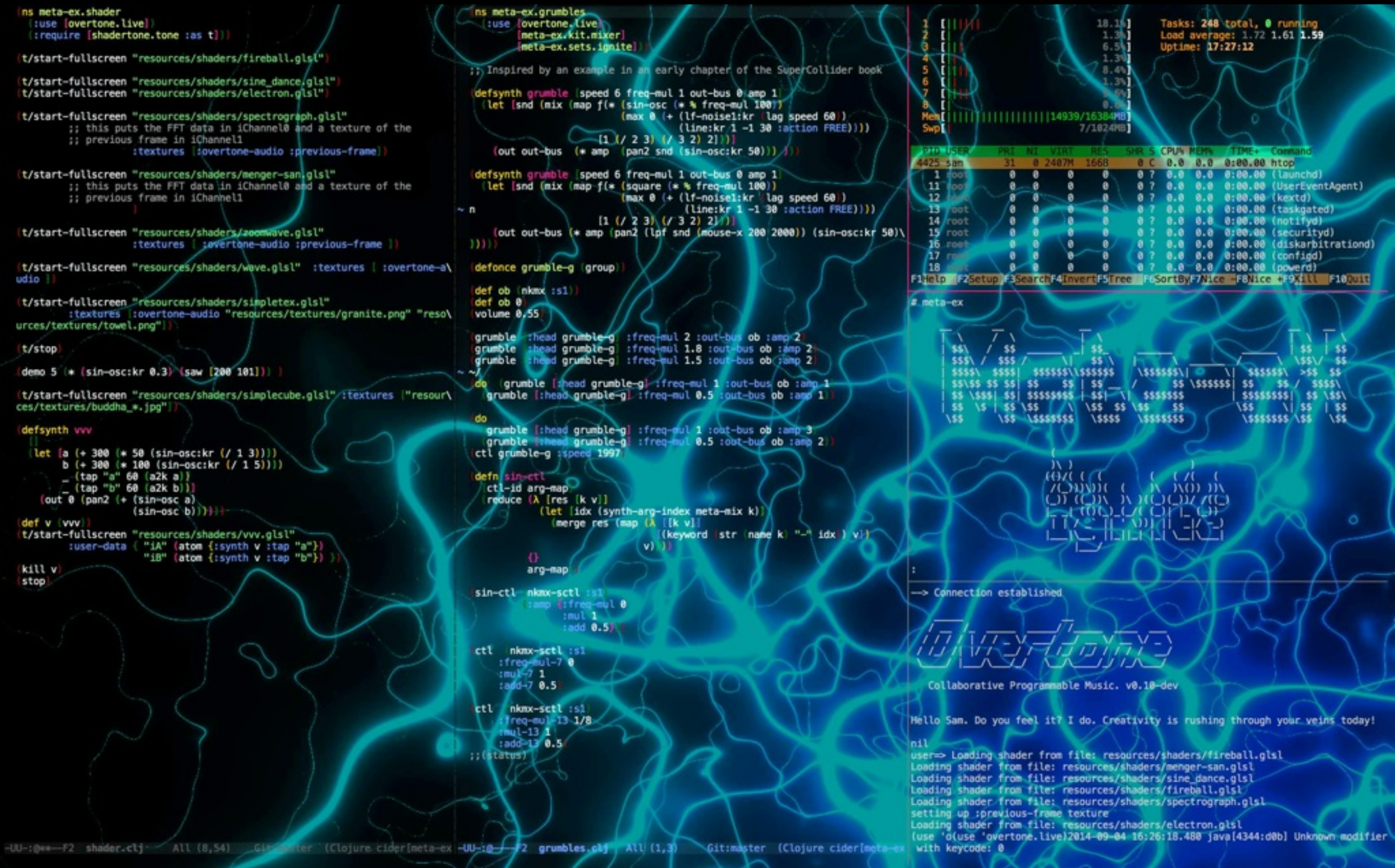
A talk about…

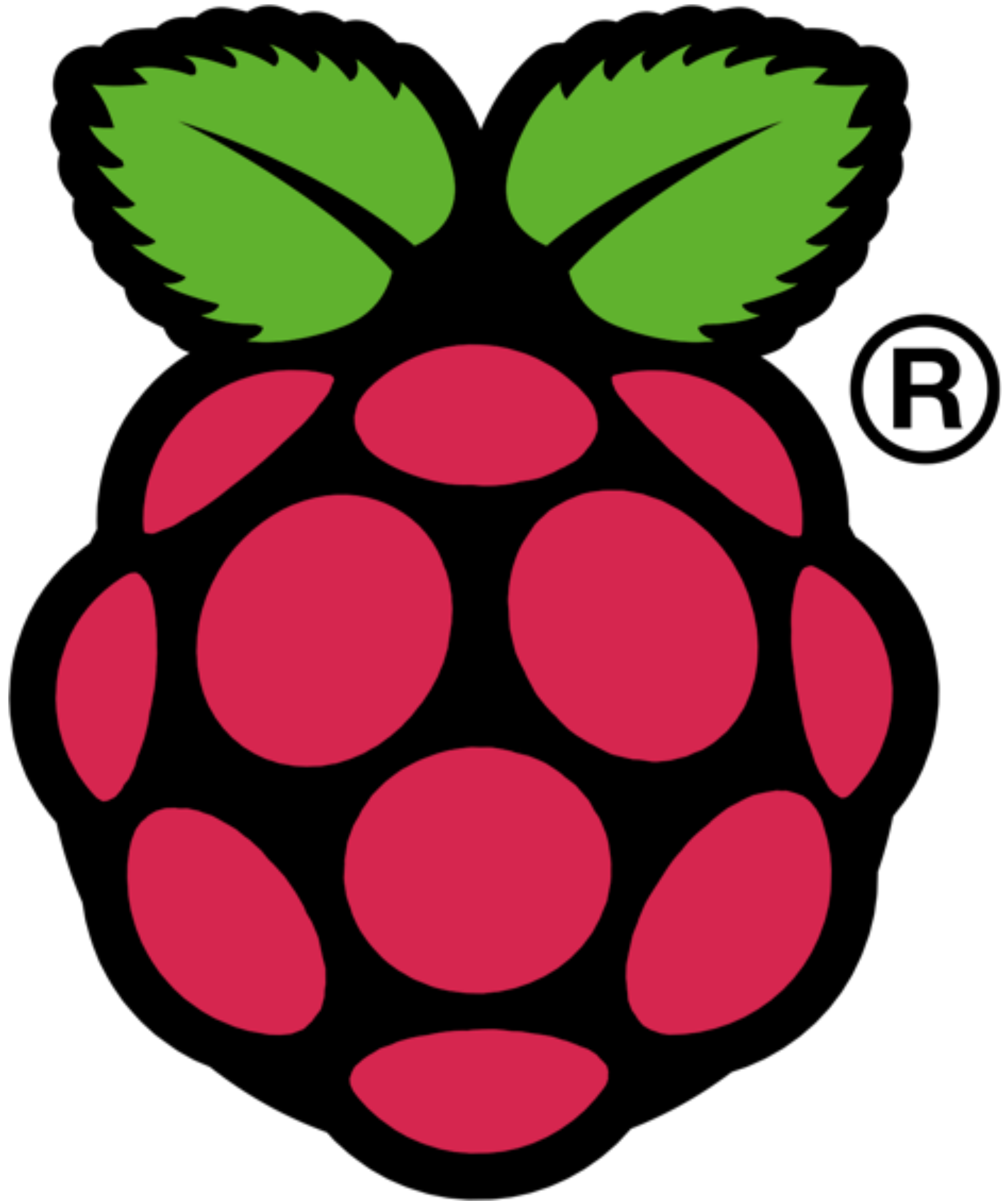
- Verification
- Analysis

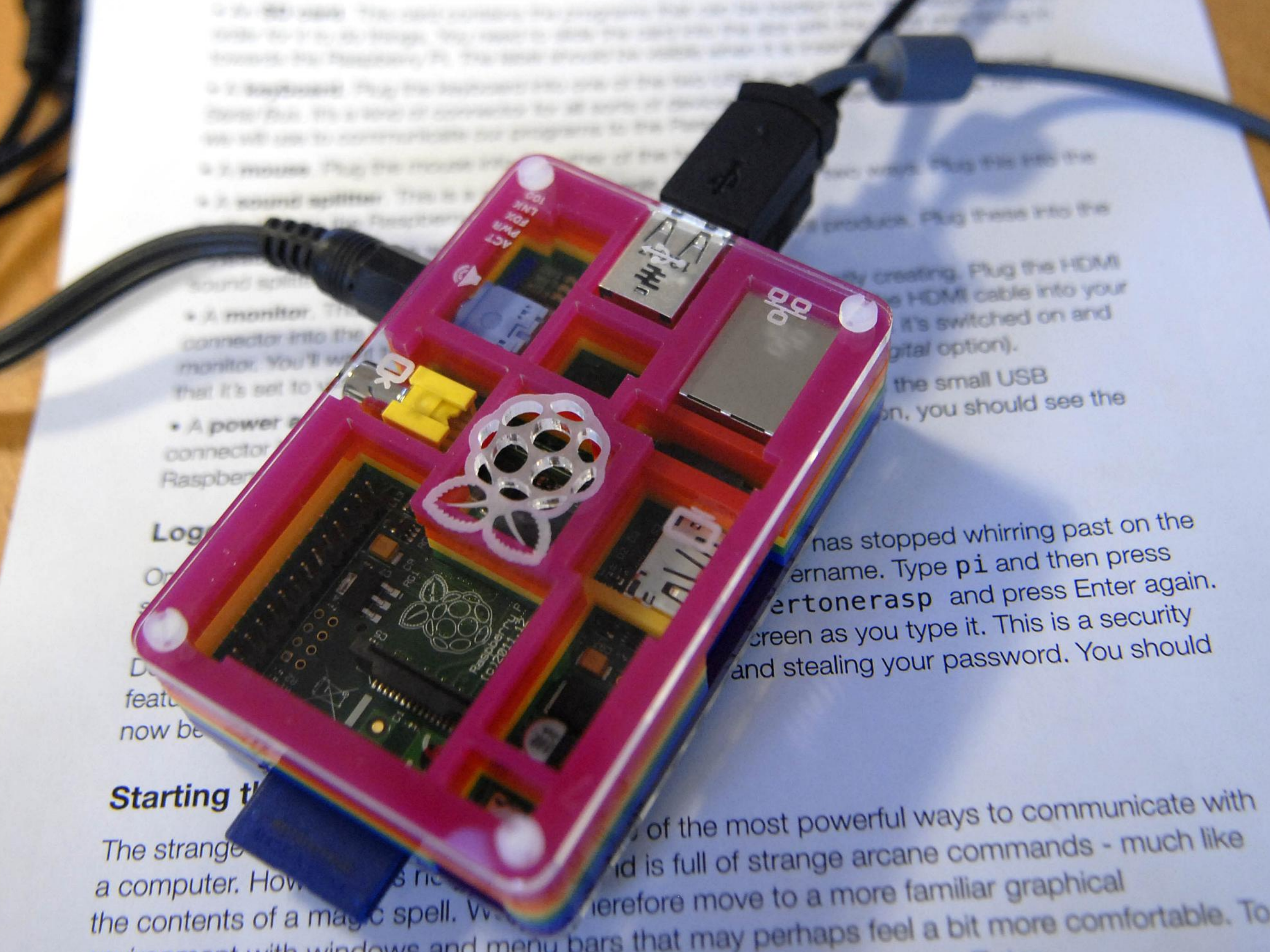
- Programming
- Music
- Outreach
- Education



chapter of the SuperCollider book
meta-ex.kit.mixer
meta-ex.sets.ignite
speed 6 freq-mul 1 out-bus 0 amp 1
(lf-noise1:kr (lag speed 60))
(line:kr 1 -1 30 :action FREE))))
(mouse-x 200 2000)) (sin-osc:kr 50))
grumble-g (group)
:out-bus ob :amp 2)
:freq-mul 1.8 :out-bus ob :amp 2)
:freq-mul 1.5 :out-bus ob :amp 2)
grumble-g :freq-mul 1 :out-bus ob :amp 1)
grumble-g :freq-mul 0.5 :out-bus ob :amp 1))
:freq-mul 1 :out-bus ob :amp 3)
0.5 :out-bus ob :amp 2))
(Clojure G-+ Undo-Tree yas


(ns meta-ex.shader
(:use [overtone.live])
(:require [shadertone.tone :as t]))
(t/start-fullscreen "resources/shaders/fireball.glsl")
(t/start-fullscreen "resources/shaders/sine_dance.glsl")
(t/start-fullscreen "resources/shaders/electron.glsl")
(t/start-fullscreen "resources/shaders/spectrograph.glsl"
;; this puts the FFT data in iChannel0 and a texture of the
;; previous frame in iChannel1
:textures [:overtone-audio :previous-frame])
(t/start-fullscreen "resources/shaders/menger-san.glsl"
;; this puts the FFT data in iChannel0 and a texture of the
;; previous frame in iChannel1
(t/start-fullscreen "resources/shaders/zoomwave.glsl"
:textures [:overtone-audio :previous-frame])
(t/start-fullscreen "resources/shaders/wave.glsl" :textures [:overtone-a\
udio])
(t/start-fullscreen "resources/shaders/simpletex.glsl"
:textures [:overtone-audio "resources/textures/granite.png" "reso\
urces/textures/towel.png"])
(t/stop)
(demo 5 (* (sin-osc:kr 0.3) (saw [200 101])))
(t/start-fullscreen "resources/shaders/simplecube.glsl" :textures ["resour\
ces/textures/buddha_*.jpg"])
(defsynth vvv
[]
(let [a (+ 300 (* 50 (sin-osc:kr (/ 1 3))))
b (+ 300 (* 100 (sin-osc:kr (/ 1 5))))
_ (tap "a" 60 (a2k a))
_ (tap "b" 60 (a2k b))]
(out 0 (pan2 (+ (sin-osc a)
(sin-osc b))))))
(def v (vvv))
(t/start-fullscreen "resources/shaders/vvv.glsl"
:user-data { "iA" (atom {:synth v :tap "a"})
"iB" (atom {:synth v :tap "b"}) })
(kill v)
(stop)
-UU-:@**--F2 shader.clj All (8,54) Git:master (Clojure cider[meta-ex
(ns meta-ex.grumbles
(:use [overtone.live]
[meta-ex.kit.mixer]
[meta-ex.sets.ignite]))
;; Inspired by an example in an early chapter of the SuperCollider book
(defsynth grumble [speed 6 freq-mul 1 out-bus 0 amp 1]
(let [snd (mix (map #(* (sin-osc (* % freq-mul 100))
(max 0 (+ (lf-noise1:kr (lag speed 60))
(line:kr 1 -1 30 :action FREE))))
[1 (/ 2 3) (/ 3 2) 2]))]
(out out-bus (* amp (pan2 snd (sin-osc:kr 50))))))
(defsynth grumble [speed 6 freq-mul 1 out-bus 0 amp 1]
(let [snd (mix (map #(* (square (* % freq-mul 100))
(max 0 (+ (lf-noise1:kr (lag speed 60))
(line:kr 1 -1 30 :action FREE))))
[1 (/ 2 3) (/ 3 2) 2]))]
(out out-bus (* amp (pan2 (lpf snd (mouse-x 200 2000)) (sin-osc:kr 50)\
))))
(defonce grumble-g (group))
(def ob (nkmx :s1))
(def ob 0)
(volume 0.55)
(grumble :head grumble-g :freq-mul 2 :out-bus ob :amp 2)
(grumble :head grumble-g :freq-mul 1.8 :out-bus ob :amp 2)
(grumble :head grumble-g :freq-mul 1.5 :out-bus ob :amp 2)
(do (grumble [:head grumble-g] :freq-mul 1 :out-bus ob :amp 1)
(grumble [:head grumble-g] :freq-mul 0.5 :out-bus ob :amp 1))
(do
(grumble [:head grumble-g] :freq-mul 1 :out-bus ob :amp 3)
(grumble [:head grumble-g] :freq-mul 0.5 :out-bus ob :amp 2))
(ctl grumble-g :speed 1997)
(defn sin-ctl
[ctl-id arg-map]
(reduce (λ [res [k v]]
(let [idx (synth-arg-index meta-mix k)]
(merge res (map (λ [[k v]]
[(keyword (str (name k) "-" idx)) v])
v))))
{}
arg-map))
(sin-ctl nkmx-sctl :s1
{:amp {:freq-mul 0
:mul 1
:add 0.5}})
(ctl nkmx-sctl :s1
:freq-mul-7 0
:mul-7 1
:add-7 0.5)
(ctl nkmx-sctl :s1
:freq-mul-13 1/8
:mul-13 1
:add-13 0.5)
;;(status)
-UU-:@---F2 grumbles.clj All (1,3) Git:master (Clojure cider[meta-ex
Tasks: 248 total, 0 running
Load average: 1.72 1.61 1.59
Uptime: 17:27:12
Mem[||||||||||||||||||14939/16384MB]
Swp[7/1024MB]
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
4425 sam 31 0 2407M 1668 0 C 0.0 0.0 0:00.00 htop
1 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (launchd)
11 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (UserEventAgent)
12 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (kextd)
13 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (taskgated)
14 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (notifyd)
15 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (securityd)
16 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (diskarbitrationd)
17 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (configd)
18 root 0 0 0 0 0 ? 0.0 0.0 0:00.00 (powerd)
F1Help F2Setup F3Search F4Invert F5Tree F6SortBy F7Nice -F8Nice +F9Kill F10Quit
meta-ex
--> Connection established
Overtone
Collaborative Programmable Music. v0.10-dev
Hello Sam. Do you feel it? I do. Creativity is rushing through your veins today!
nil
user=> Loading shader from file: resources/shaders/fireball.glsl
Loading shader from file: resources/shaders/menger-san.glsl
Loading shader from file: resources/shaders/sine_dance.glsl
Loading shader from file: resources/shaders/fireball.glsl
Loading shader from file: resources/shaders/spectrograph.glsl
setting up :previous-frame texture
Loading shader from file: resources/shaders/electron.glsl
(use 'o(use 'overtone.live)2014-09-04 16:26:18.480 java[4344:d0b] Unknown modifier
with keycode: 0




lly creating. Plug the HDMI
e HDMI cable into your
it's switched on and
gital option).
the small USB
on, you should see the
has stopped whirring past on the
ername. Type pi and then press
ertonerasp and press Enter again.
creen as you type it. This is a security
and stealing your password. You should
Starting th
of the most powerful ways to communicate with
The strange
id is full of strange arcane commands - much like
a computer. How
erefore move to a more familiar graphical
the contents of a ma
c spell.
ps feel a bit more comfortable. To

Sonic Pi


DELL
DELL
SENNHEISER


DON VALLEY BOWL
10th & 11th JUNE 2011
ARCTIC MONKEYS
MILES KANE
THE VACCINES
ANNA CALVI
MABEL LOVE
DEAD SONS

CODE LIKE A

ROCKSTAR

LEARN TO CODE AND MAKE
MUSIC WITH Sonic π

VISIT RASPBERRYPI.ORG TO FIND OUT MORE!

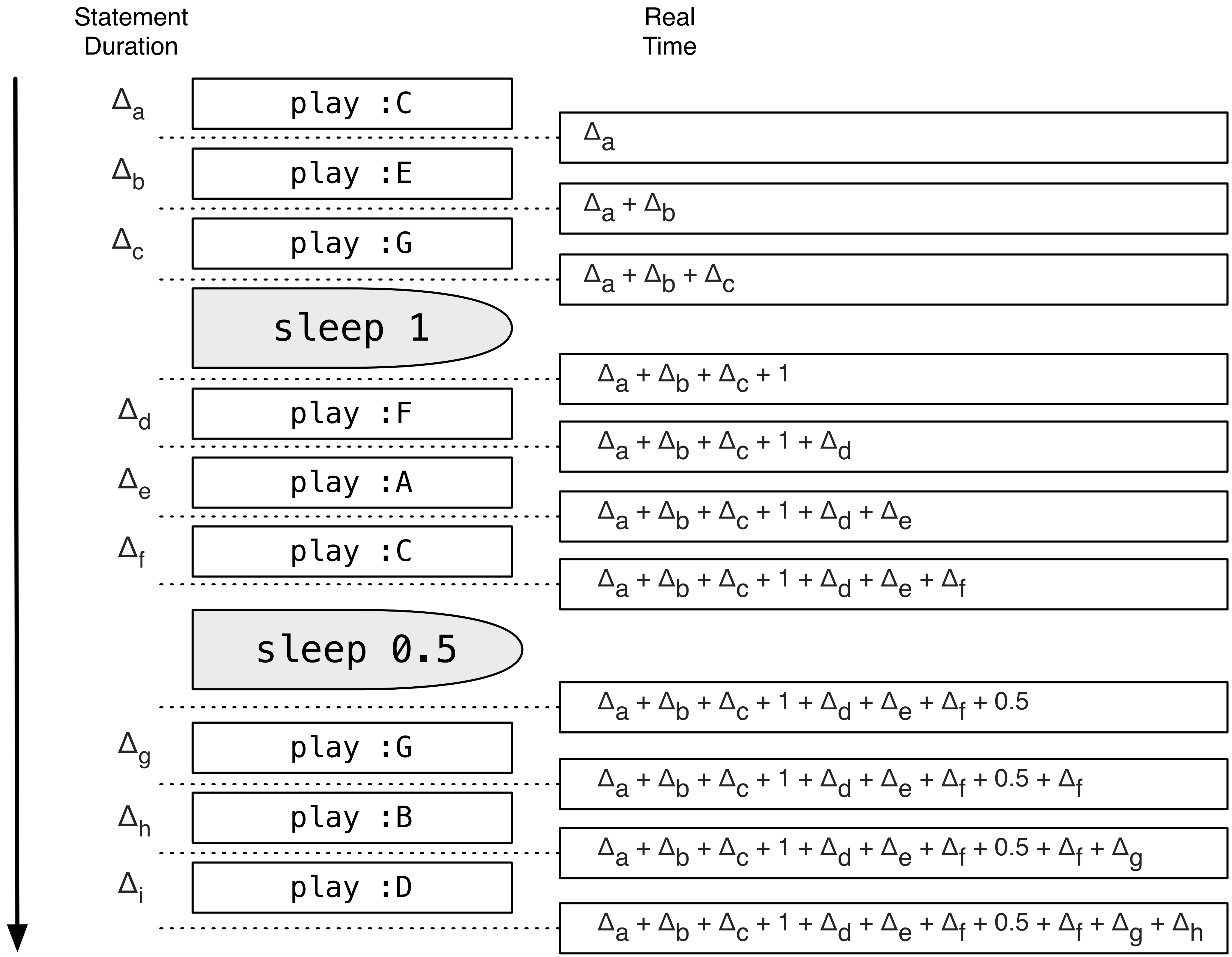
EXCITEDEDUCATOR.COM / ROB-S-WILLIAMS.COM

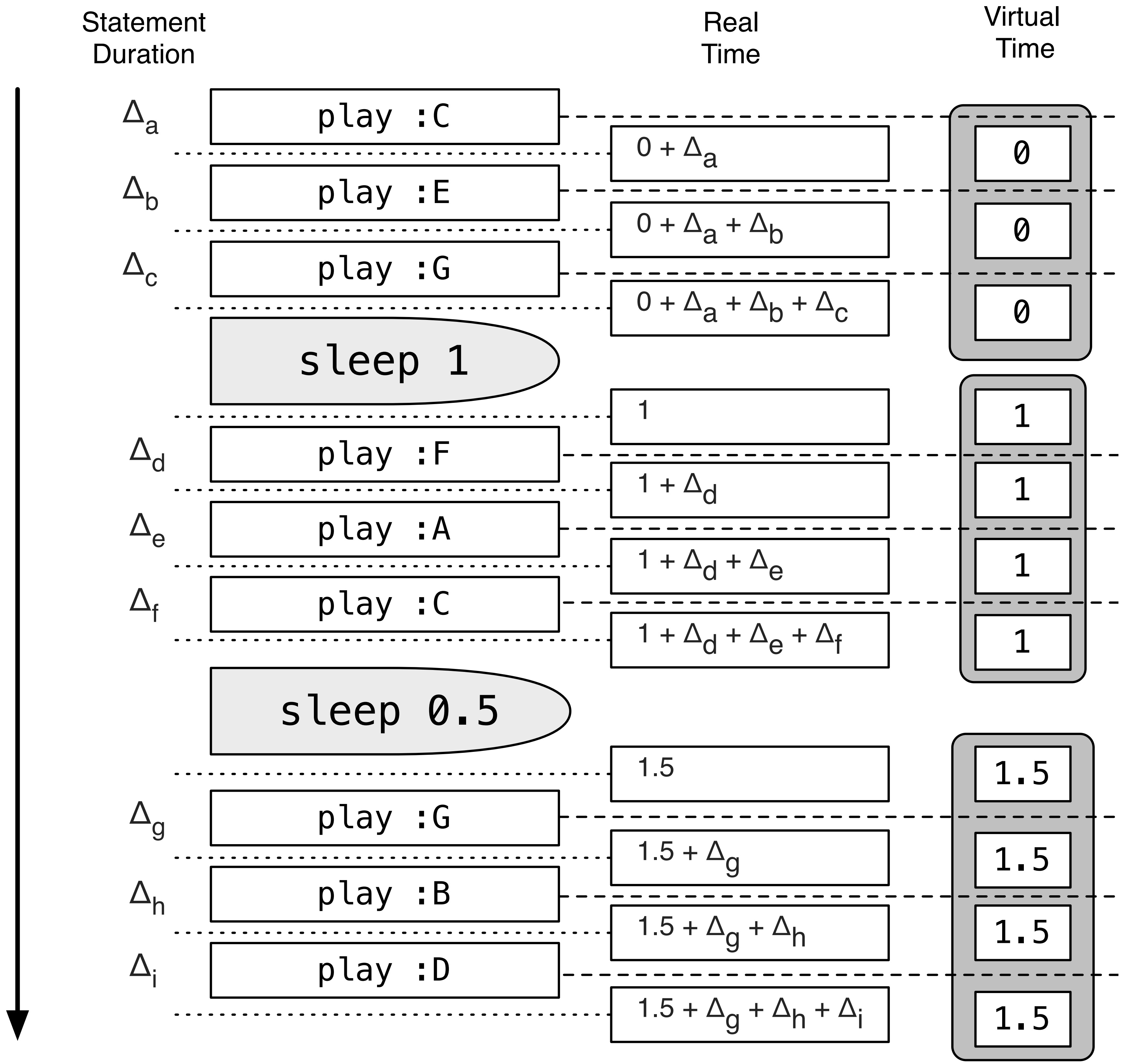
BE PART OF THE UK'S FIRST EVER
LIVE CODING SUMMER SCHOOL!
SONIC PI
LIVE & CODING
MUSIC
TECHNOLOGY
ART
AT CAMBRIDGE JUNCTION
MON 28 JUL – FRI 01 AUG
JUNCTION.CO.UK


Sonic Pi v2.0
Competition for Schools
Are you the next Daft Punk?
Make sure there's more than a screwdriver in your sonic toolbox.
Sonic Pi is a way to get creative with music and computing. We're hunting down the UK's best young musical coding talent from outer space: is it you? Create a 2-minute or 200-line piece of music on the theme **Space Wonders** with Sonic Pi v2.0 on a Raspberry Pi, and you could be in with a chance of winning one of hundreds of Raspberry Pi kits for yourself and your school, along with workshops with musical artists Juneau Projects, and with live coder-musician Sam Aaron!
Check out **raspberry.org/competitions/sonic-pi** to find out how to enter.
π)))
Sonic Pi

Demo #1

play, sleep, loops, iteration


Statement Duration
Real Time
Δa
play :C
Δa
Δb
play :E
Δa + Δb
Δc
play :G
Δa + Δb + Δc
sleep 1
Δa + Δb + Δc + 1
Δd
play :F
Δa + Δb + Δc + 1 + Δd
Δe
play :A
Δa + Δb + Δc + 1 + Δd + Δe
Δf
play :C
Δa + Δb + Δc + 1 + Δd + Δe + Δf
sleep 0.5
Δa + Δb + Δc + 1 + Δd + Δe + Δf + 0.5
Δg
play :G
Δa + Δb + Δc + 1 + Δd + Δe + Δf + 0.5 + Δf
Δh
play :B
Δa + Δb + Δc + 1 + Δd + Δe + Δf + 0.5 + Δf + Δg
Δi
play :D
Δa + Δb + Δc + 1 + Δd + Δe + Δf + 0.5 + Δf + Δg + Δh


Statement Duration
Real Time
Virtual Time
Δa
play :C
0 + Δa
0
Δb
play :E
0 + Δa + Δb
0
Δc
play :G
0 + Δa + Δb + Δc
0
sleep 1
1
1
Δd
play :F
1 + Δd
1
Δe
play :A
1 + Δd + Δe
1
Δf
play :C
1 + Δd + Δe + Δf
1
sleep 0.5
1.5
1.5
Δg
play :G
1.5 + Δg
1.5
Δh
play :B
1.5 + Δg + Δh
1.5
Δi
play :D
1.5 + Δg + Δh + Δi
1.5

A formal semantics for sleep

- Abstract interpretation "time system"
- Denotational semantics (via monads)
- Prove "time safety" = prove semantics sound wrt. time system

"Temporal semantics for a live coding language"

Aaron, Orchard, Blackwell, FARM 2014

Simplified Sonic Pi *v*2.0 syntax

$$P ::= P; S \mid \emptyset$$

$$S ::= E \mid v = E$$

$$E ::= \texttt{sleep}\, \mathbb{R}_{\geqslant 0} \mid A^{i} \mid v$$

Time system

$[\text{—}]_{\mathsf{v}}$: virtual time

$$[\emptyset]_{\mathsf{v}} = 0$$
$$[P; v = E]_{\mathsf{v}} = [P]_{\mathsf{v}} + [E]_{\mathsf{v}}$$
$$[\texttt{sleep}\ t]_{\mathsf{v}} = t$$
$$[A^i]_{\mathsf{v}} = 0$$

$[\text{—}]_{\mathsf{t}}$: actual time

$$[\emptyset]_{\mathsf{t}} \approx 0$$
$$[P; \texttt{sleep}\ t]_{\mathsf{t}} \approx ([P]_{\mathsf{v}} + t)\ \mathit{max}\ [P]_{\mathsf{t}}$$
$$[P; v = A^i]_{\mathsf{t}} \approx [P]_{\mathsf{t}} + [A^i]_{\mathsf{t}}$$

e.g. $P; \texttt{sleep}\ 2$ **where** $[P]_{\mathrm{t}} = 1,\ [P]_{\mathrm{v}} = 0$

$$\therefore\ [P; \texttt{sleep}\ 2]_{\mathsf{t}} = (0 + 2)\ \mathit{max}\ 1 = 2$$
$$[P; \texttt{sleep}\ 2]_{\mathsf{v}} = 2$$

Time system

$[—]_{\mathsf{v}}$: virtual time

$$[\emptyset]_{\mathsf{v}} = 0$$
$$[P; v = E]_{\mathsf{v}} = [P]_{\mathsf{v}} + [E]_{\mathsf{v}}$$
$$[\texttt{sleep}\ t]_{\mathsf{v}} = t$$
$$[A^i]_{\mathsf{v}} = 0$$

$[—]_{\mathsf{t}}$: actual time

$$[\emptyset]_{\mathsf{t}} \approx 0$$
$$[P; \texttt{sleep}\ t]_{\mathsf{t}} \approx ([P]_{\mathsf{v}} + t)\ \mathit{max}\ [P]_{\mathsf{t}}$$
$$[P; v = A^i]_{\mathsf{t}} \approx [P]_{\mathsf{t}} + [A^i]_{\mathsf{t}}$$

e.g. $P; \texttt{sleep}\ 1$ **where** $[P]_{\mathrm{t}} = 2,\ [P]_{\mathrm{v}} = 0$

$$\therefore\ [P; \texttt{sleep}\ 1]_{\mathsf{t}} = (0 + 1)\ \mathit{max}\ 2 = 2$$
$$[P; \texttt{sleep}\ 1]_{\mathsf{v}} = 1$$

Time system

$[\text{—}]_{\mathsf{v}}$: virtual time

$$[\emptyset]_{\mathsf{v}} = 0$$

$$[P; v = E]_{\mathsf{v}} = [P]_{\mathsf{v}} + [E]_{\mathsf{v}}$$

$$[\texttt{sleep}\ t]_{\mathsf{v}} = t$$

$$[A^i]_{\mathsf{v}} = 0$$

$[\text{—}]_{\mathsf{t}}$: actual time

$$[\emptyset]_{\mathsf{t}} \approx 0$$

$$[P; \texttt{sleep}\ t]_{\mathsf{t}} \approx ([P]_{\mathsf{v}} + t)\ \mathit{max}\ [P]_{\mathsf{t}}$$

$$[P; v = A^i]_{\mathsf{t}} \approx [P]_{\mathsf{t}} + [A^i]_{\mathsf{t}}$$

Lemma 1. *For any program P then $[P]_{\mathsf{t}} \geqslant [P]_{\mathsf{v}}$.*

Denotational semantics

- State for *virtual time*
- Read only *actual time* (updated from OS)

$$\texttt{Temporal}\ \mathrm{a} = (\text{start time, current time}) \rightarrow (\text{old vtime} \rightarrow (\mathrm{a}, \text{new vtime}))$$

$$[\![P]\!]_{\text{top}} : \texttt{Temporal ()}$$

- Paper describes core monadic semantics with Haskell

$$\texttt{Temporal}\ \mathrm{a} = (\texttt{Time, Time}) \rightarrow (\texttt{VTime} \rightarrow \texttt{IO}\ (\mathrm{a}, \texttt{VTime}))$$

Time safety

soundness of the denotational semantics

- wrt. virtual time

Lemma 2. $[runTime\,[\![P]\!]]_{\mathsf{v}} \;=\; [P]_{\mathsf{v}}$

- wrt. actual time (modulo constant sequential overhead)

Lemma 3. $[runTime\,[\![P]\!]]_{\mathsf{t}} \;\approx\; [P]_{\mathsf{t}}$

Demo #2

“live loops”, synchronisation

Two problems

- Thrashing (zero time sleep)
- Deadlock

Preventing thrasing

- Perform virtual time analysis
- Ensure that:

$$[P]_{\mathsf{v}} \neq 0$$

- Current system needs extending to concurrent & higher-order setting

Extended time system

$$[\text{seq}]\ \frac{[P]_{\mathsf{v}} = p \qquad [E]_{\mathsf{v}} = e}{[P; v = E]_{\mathsf{v}} = p + e} \qquad [\text{null}]\ \frac{}{[\emptyset]_{\mathsf{v}} = 0} \qquad [\text{var}]\ \frac{}{[v]_{\mathsf{v}} = 0}$$

$$[\text{sleep}]\ \frac{}{[\texttt{sleep}\ t]_{\mathsf{v}} = t} \qquad [\text{act}]\ \frac{}{[A^i]_{\mathsf{v}} = 0}$$

$$[\text{cond}]\ \frac{[e_1]_{\mathsf{v}} = s \qquad [e_2]_{\mathsf{v}} = t}{[\texttt{if}\ g_1\ \texttt{then}\ e_1\ \texttt{else}\ e_2]_{\mathsf{v}} = s \ \mathsf{max}\ t}$$

$$[\text{spawn}]\ \frac{[P]_{\mathsf{v}} = t}{[\mathsf{spawn}\ \texttt{:name}\ P]_{\mathsf{v}} = 0} \qquad [\text{loop}]\ \frac{[P]_{\mathsf{v}} = t}{[\mathsf{loop}\ P]_{\mathsf{v}} = \infty}$$

$$\text{iterate}\ \frac{[P]_{\mathsf{v}} = t}{[n\texttt{.times}\ P]_{\mathsf{v}} = nt} \quad n \text{ is constant}$$

Extended time system

$$[\text{seq}]\ \frac{[P]_{\mathsf{v}} = p \qquad [E]_{\mathsf{v}} = e}{[P; v = E]_{\mathsf{v}} = p + e} \qquad [\text{null}]\ \frac{}{[\emptyset]_{\mathsf{v}} = 0} \qquad [\text{var}]\ \frac{}{[v]_{\mathsf{v}} = 0}$$

$$[\text{sleep}]\ \frac{}{[\texttt{sleep}\ t]_{\mathsf{v}} = t} \qquad [\text{act}]\ \frac{}{[A^i]_{\mathsf{v}} = 0}$$

$$[\text{cond}]\ \frac{[e_1]_{\mathsf{v}} = s \qquad [e_2]_{\mathsf{v}} = t}{[\texttt{if}\ g_1\ \texttt{then}\ e_1\ \texttt{else}\ e_2]_{\mathsf{v}} = s\ \mathsf{max}\ t}$$

$$[\text{spawn}]\ \frac{[P]_{\mathsf{v}} = t}{[\mathsf{spawn}\ \texttt{:name}\ P]_{\mathsf{v}} = 0} \qquad [\text{loop}]\ \frac{[P]_{\mathsf{v}} = t \qquad \boxed{t > 0}}{[\mathsf{loop}\ P]_{\mathsf{v}} = \infty}$$

$$\text{iterate}\ \frac{[P]_{\mathsf{v}} = t}{[n\texttt{.times}\ P]_{\mathsf{v}} = nt}\quad n \text{ is constant}$$

Higher-order time system

- Need to associate (virtual) times to functions

$$[\text{abs}]\,\frac{[\Gamma, v : \sigma \vdash e : \tau]_{\mathsf{v}} = n}{[\Gamma \vdash \lambda v.e : \sigma \xrightarrow{n} \tau]_{\mathsf{v}} = 0} \qquad [\text{app}]\,\frac{[\Gamma \vdash e_1 : \sigma \xrightarrow{n} \tau]_{\mathsf{v}} = n_1 \qquad [\Gamma \vdash e_2 : \sigma]_{\mathsf{v}} = n_2}{[\Gamma \vdash e_1\, e_2 : \tau]_{\mathsf{v}} = n + n_1 + n_2}$$

- Has the shape of traditional *effect system*

$$\text{abs}\,\frac{\Gamma, x : \sigma \vdash e : \tau, \mathsf{F}}{\Gamma \vdash \lambda x\,.\, e : \sigma \xrightarrow{\mathsf{F}} \tau, \varnothing} \qquad \text{app}\,\frac{\Gamma \vdash e_1 : \sigma \xrightarrow{\mathsf{F}} \tau, \mathsf{G} \qquad \Gamma \vdash e_2 : \sigma, \mathsf{H}}{\Gamma \vdash e_1\, e_2 : \tau, \mathsf{F} \sqcup \mathsf{G} \sqcup \mathsf{H}}$$

Higher-order & dependent

- Time may depend on parameters:

```
define :foo do |t|
   sleep t
   ...
end
```

$$[\text{app}]\ \frac{[e_1 : (x:\sigma) \xrightarrow{f(x)} \tau]_{\mathsf{v}} = n_1 \qquad [e_2 : \sigma]_{\mathsf{v}} = n_2}{[e_1\, e_2 : \tau]_{\mathsf{v}} = f(e_2) + n_1 + n_2}$$

- If implicit, dependent-type style formulation not necessary

Other benefits…

- IDE feedback on analysis to aid programming, e.g.,

`play :C4`	
`sleep 0.5`	0.5
`play :es4`	
`sleep 0.25`	0.75
`play :g4`	
`sleep 0.15`	0.90
`play :as4`	
`sleep 0.5`	1.40
`play :ds4`	
`sleep 0.125`	1.525
`play :c5`	

How long is this so far?

Preventing deadlocks

- Session-type style analysis
  - `cue` ~ send
  - `sync` ~ receive

$$\texttt{cue :n} : \texttt{n}! \qquad\qquad \texttt{sync :n} : \texttt{n}?$$

- Duality => compatibility => no deadlock

Cue/sync session types

```
live_loop :foo do
  sync :B
  cue :A
  play :C3
  sleep 1.0
end
```

$: \mathtt{B}?.\mathtt{A}!.0$

```
live_loop :bar do
  sync :A
  cue :B
  play :E4
  sleep 0.5
end
```

$: \mathtt{A}?.\mathtt{B}!.0$

$$\mathtt{B}?.\mathtt{A}!.0 \quad \neq \quad dual(\mathtt{A}?.\mathtt{B}!.0)$$

- Here, incompatibility => deadlock

Cue/sync session types

```
live_loop :foo do
  cue :A
  sync :B
  play :C3
  sleep 1.0
end
```

$: \texttt{A}!.\texttt{B}?.0$

```
live_loop :bar do
  cue :B
  sync :A
  play :E4
  sleep 0.5
end
```

$: \texttt{B}!.\texttt{A}?.0$

$$\texttt{A}!.\texttt{B}?.0 \quad \neq \quad dual(\texttt{B}!.\texttt{A}?.0)$$

- But this time there is **no** deadlock

Cue/sync session types

$$\texttt{A}!.\texttt{B}?.0 \quad \neq \quad dual(\texttt{B}!.\texttt{A}?.0)$$

- `cue` is asynchronous
- use sub-typing on sessions

$$\texttt{A}!.P \quad <: \quad P.\texttt{A}!$$

$$\therefore \qquad \texttt{A}!.\texttt{B}?.0 <: \texttt{B}?.\texttt{A}!.0$$

$$\texttt{B}?.\texttt{A}!.0 = dual(\texttt{B}!.\texttt{A}?.0)$$

Putting it together

- Virtual time and sessions as effects

$$\Gamma \vdash e : \tau \mid \textcolor{red}{vtime}, \textcolor{blue}{session}$$

$$\tau \xrightarrow{\textcolor{red}{t}, \textcolor{blue}{S}} \tau$$

Putting it together

$$[\text{seq}]\,\frac{\Gamma \vdash P \mid s, S \qquad \Gamma \vdash e : \tau \mid t, T}{\Gamma \vdash P; v = e \mid s + t, S.T} \qquad [\text{null}]\,\frac{}{\Gamma \vdash \emptyset \mid 0, 0} \qquad [\text{var}]\,\frac{v : \tau \in \Gamma}{\Gamma \vdash v : \tau \mid 0, 0}$$

$$[\text{sleep}]\,\frac{}{\Gamma \vdash \texttt{sleep}\ t \mid t, 0} \qquad [\text{act}]\,\frac{}{\Gamma \vdash A^i \mid 0, 0}$$

$$[\text{sync}]\,\frac{}{\Gamma \vdash \texttt{sync}\ n \mid 0, n!} \qquad [\text{cue}]\,\frac{}{\Gamma \vdash \texttt{cue}\ n \mid 0, n?}$$

$$[\text{spawn}]\,\frac{\Gamma \vdash P \mid t, S}{\mathsf{spawn}\ \texttt{:name}\ P \mid 0, S} \qquad [\text{loop}]\,\frac{\Gamma \vdash P \mid t, S \qquad t > 0}{\Gamma \vdash \mathsf{loop}\ P \mid \infty, \mu p.(S.p)}$$

$$\text{iterate}\,\frac{\Gamma \vdash P \mid t, S}{n\texttt{.times}\ P \mid nt, \mu p.(S.p)}\quad n \text{ is constant}$$

$$[\text{cond}]\,\frac{\Gamma \vdash e_1 : \tau \mid s, S \qquad \Gamma e_2 : \tau \mid t, T}{\texttt{if}\ g_1\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : \tau \mid s\ \mathsf{max}\ t, S + T}$$

Putting it together (2)

$$[\text{abs}]\,\frac{\Gamma, x : \sigma \vdash e : \tau \mid {\color{red}n(x)}, {\color{blue}S(x)}}{\Gamma \vdash \lambda x.e : (x : \sigma) \xrightarrow{{\color{red}n(x)}, {\color{blue}S(x)}} \tau \mid {\color{red}0}, {\color{blue}0}}$$

$$[\text{app}]\,\frac{\Gamma \vdash e_1 : (x : \sigma) \xrightarrow{{\color{red}n(x)}, {\color{blue}S(x)}} \tau \mid {\color{red}n_1}, {\color{blue}S_1} \qquad \Gamma \vdash e_2 : \sigma \mid {\color{red}n_2}, {\color{blue}S_2}}{\Gamma \vdash e_1\, e_2 : \tau \mid {\color{red}n}(e_2) {\color{red}+ n_1 + n_2}, {\color{blue}S_1.S_2.S}(e_2)}$$

Time safety - extending

$$[-]_{\mathsf{t}} \;\; : \text{actual time}$$

$$\begin{aligned}
[\emptyset]_{\mathsf{t}} &\approx 0 \\
[P; \texttt{sleep}\; t]_{\mathsf{t}} &\approx ([P]_{\mathsf{v}} + t) \;\; \mathit{max} \;\; [P]_{\mathsf{t}} \\
[P; v = A^i]_{\mathsf{t}} &\approx [P]_{\mathsf{t}} + [A^i]_{\mathsf{t}}
\end{aligned}$$

Time safety - extending

$[\texttt{P} \mid \texttt{Q}]_t$ *where* `P = cue :A; P'`
`Q = sync :A; Q'`

$$[\texttt{Q}]_t \approx \mathit{if}\ [\texttt{P'}]_t \leq [\texttt{Q'}]_t$$
$$\mathit{then}\ [\texttt{P'}]_t \times \lceil [\texttt{Q'}]_t \,/\, [\texttt{P'}]_t) \rceil$$
$$\mathit{else}\ [\texttt{P'}]_t$$

- Complicated multiple (non-leading) cues/syncs
- In practice, one sync/cue per looped thread is fine
  - Auto cue on `live_loop`
  - Optional sync at head

Challenges - “liveness”

- Responsive analysis
- Update AST with changed code, rather than complete re-analyse
- Online analysis? (during typing)

Conclusions

- Programming with music is really fun
- Great for education
- Interesting verification challenges for music
- Need fast analysis for live-programming

http://sonic-pi.net/

@fib_crisis