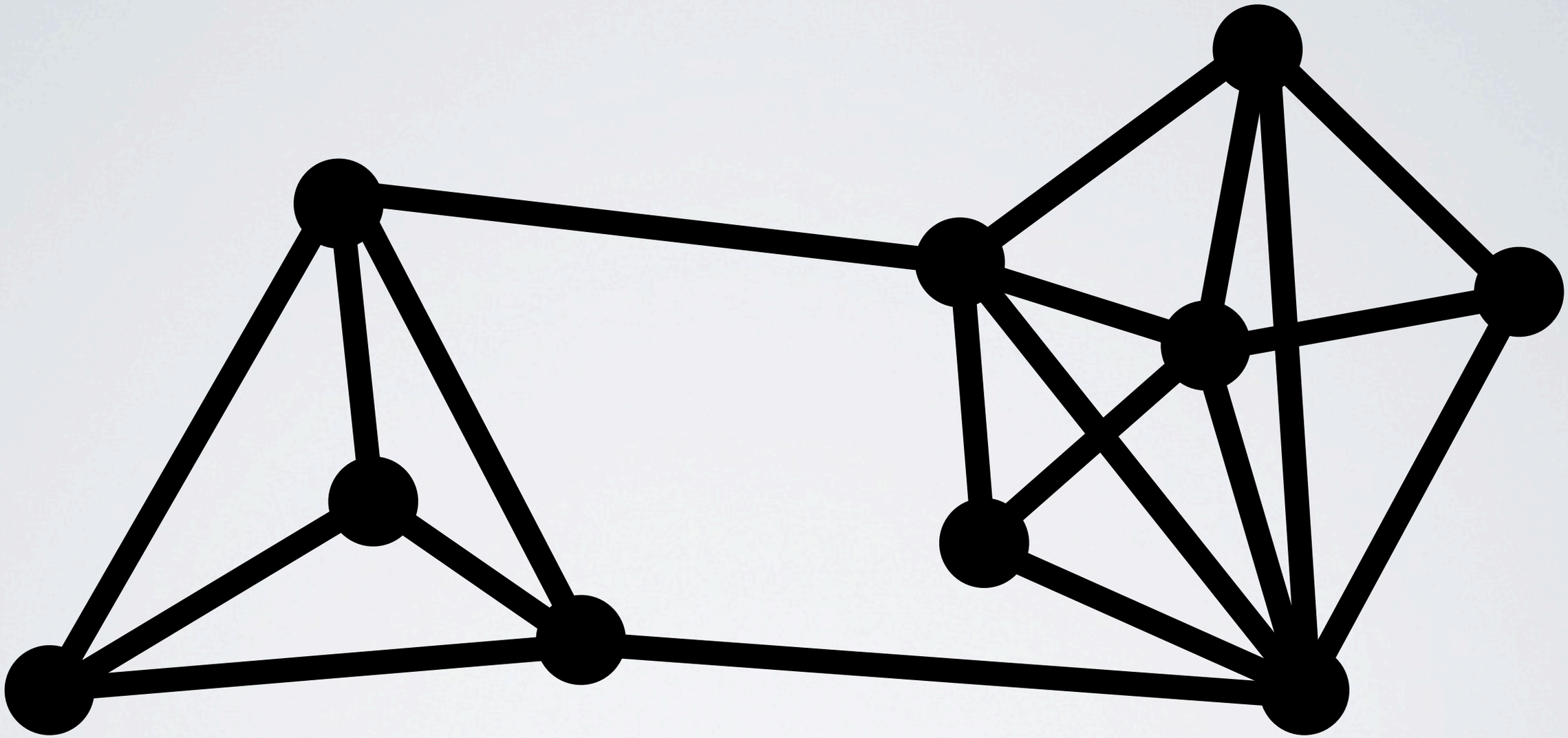
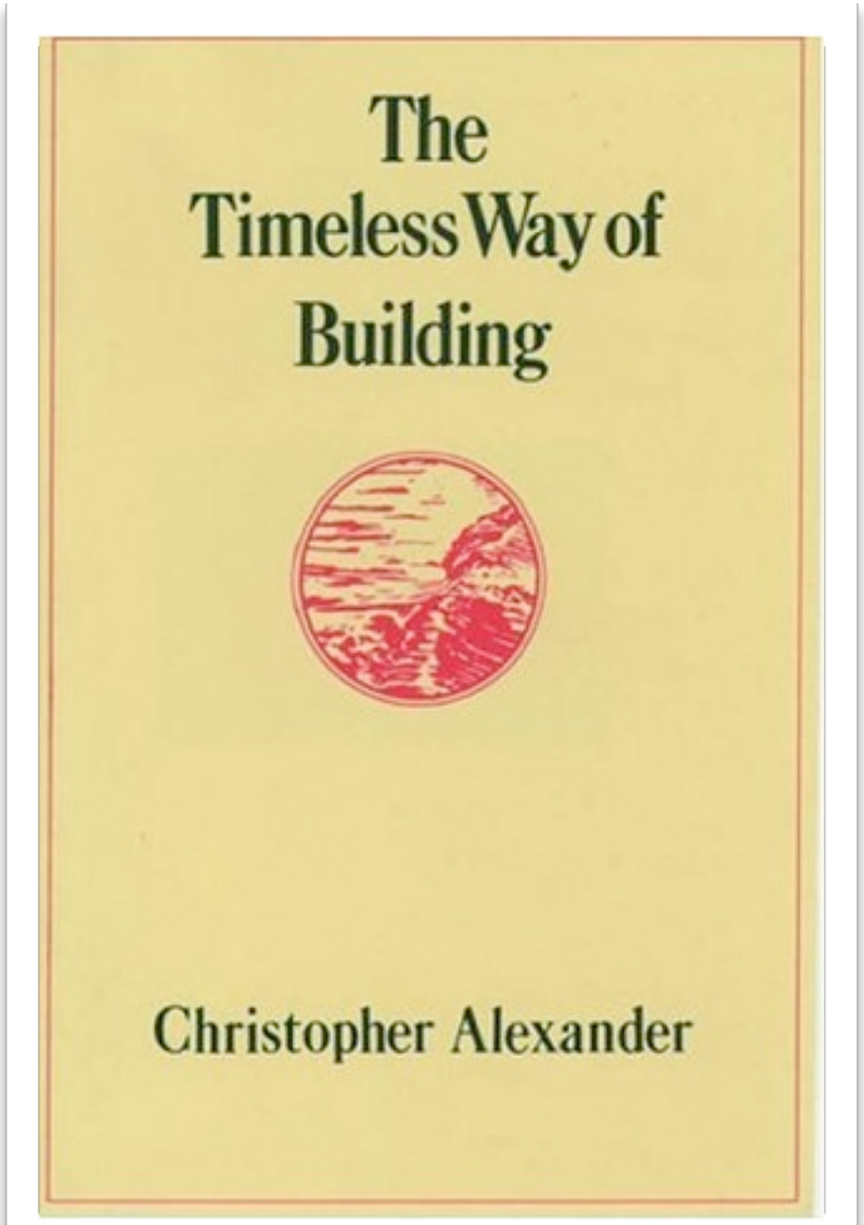
NOTES ON THE SYNTHESIS OF MUSIC

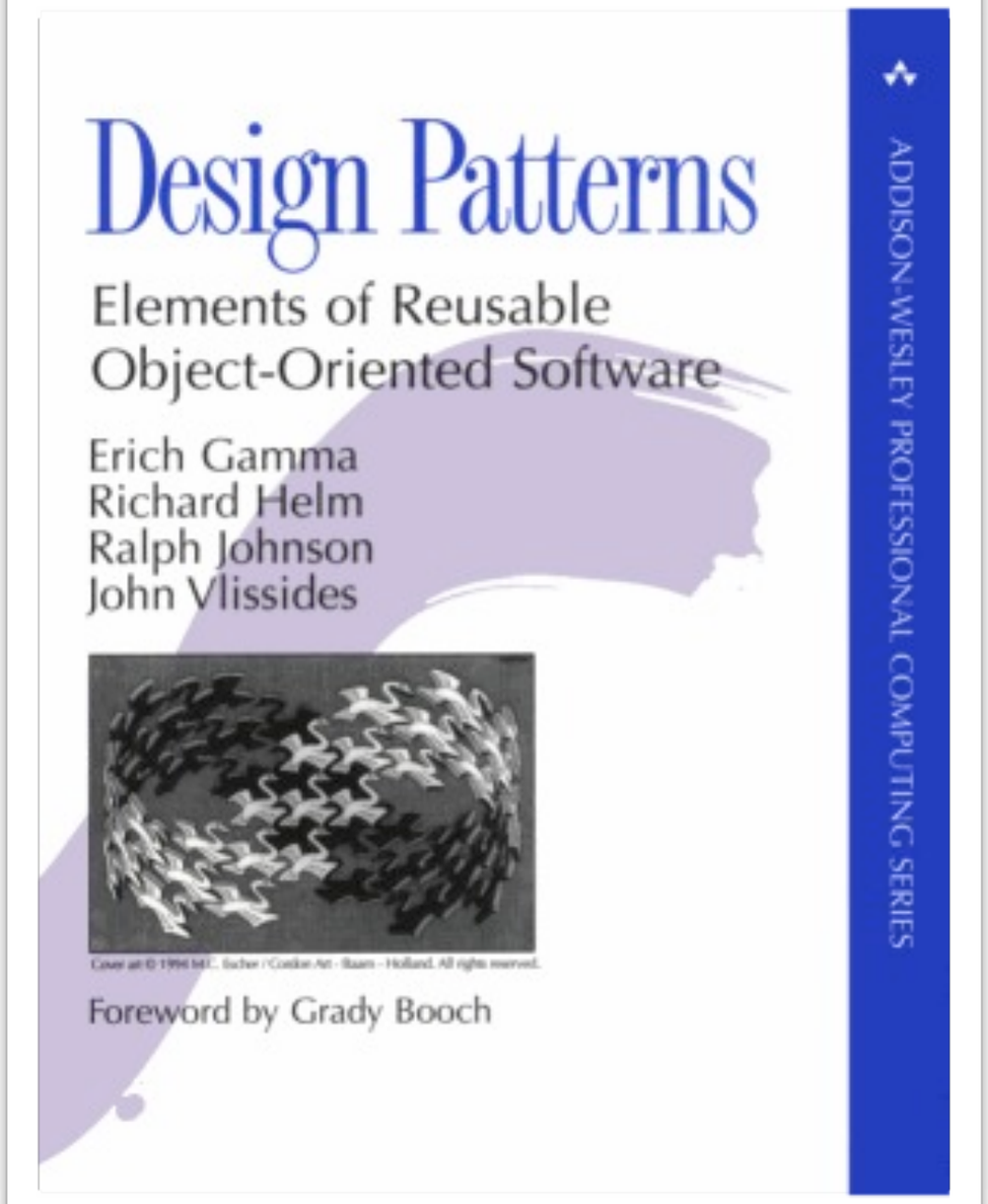
SAMUEL JAMES AARON

NOTES ON THE SYNTHESIS OF FORM

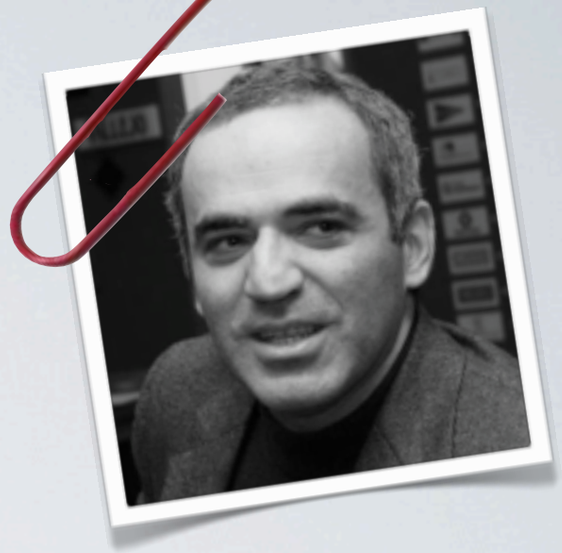
CHRISTOPHER ALEXANDER




The
Timeless Way of
Building
Christopher Alexander


Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


KASPAROV VS
DEEP BLUE
the rematch
1:57
1:58



*"**Weak human** + **machine** + **better process***

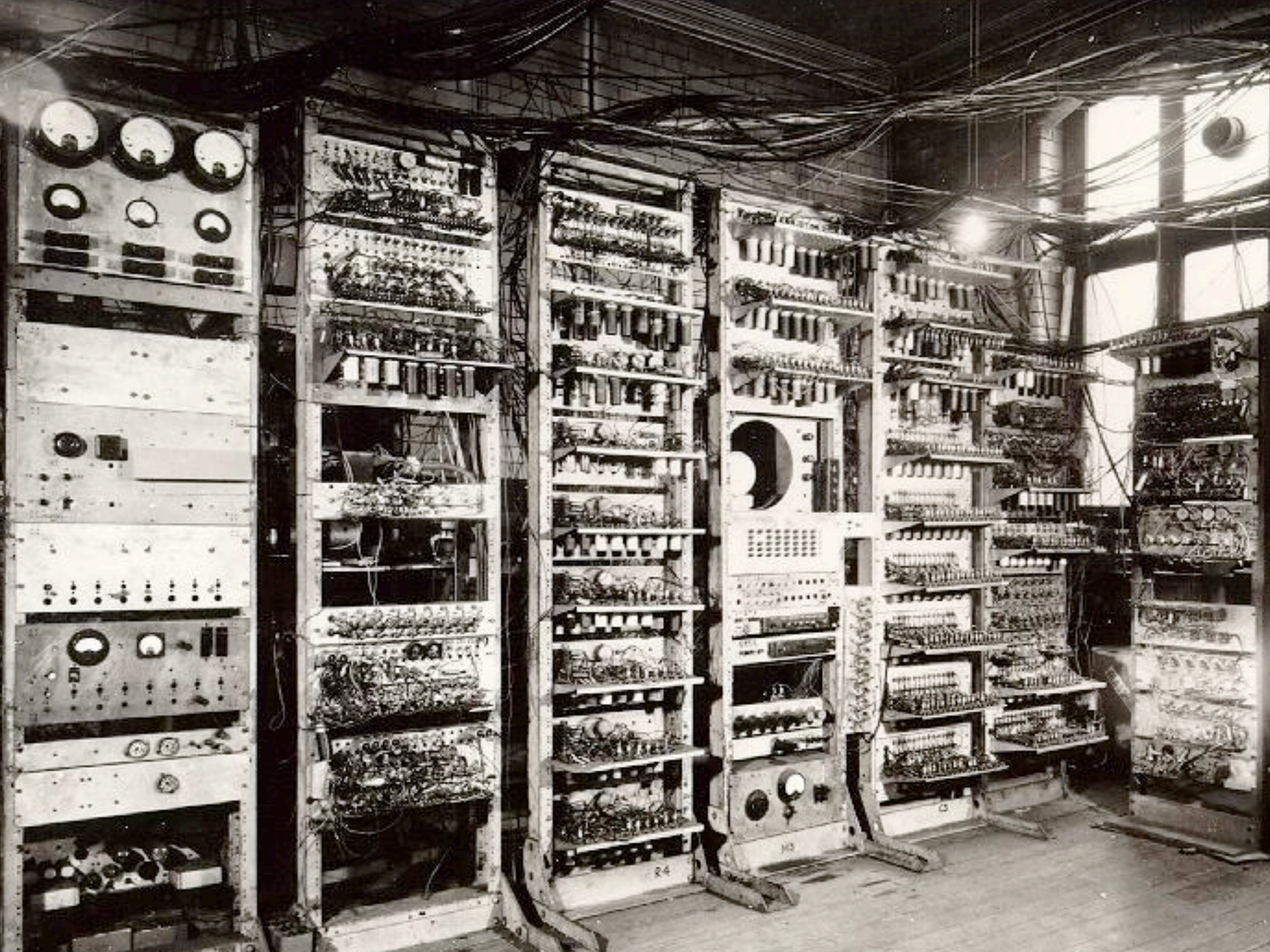
*was superior to a **strong computer** alone*

and, more remarkably, superior to a

***strong human** + **machine** + **inferior process**."*

Gary Kasparov

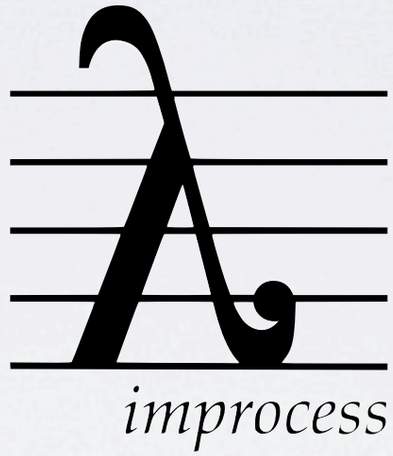
http://www.nybooks.com/articles/archives/2010/feb/11/the-chess-master-and-the-computer/



improcess

Alan Blackwell
Chris Nash
Rainbow Group

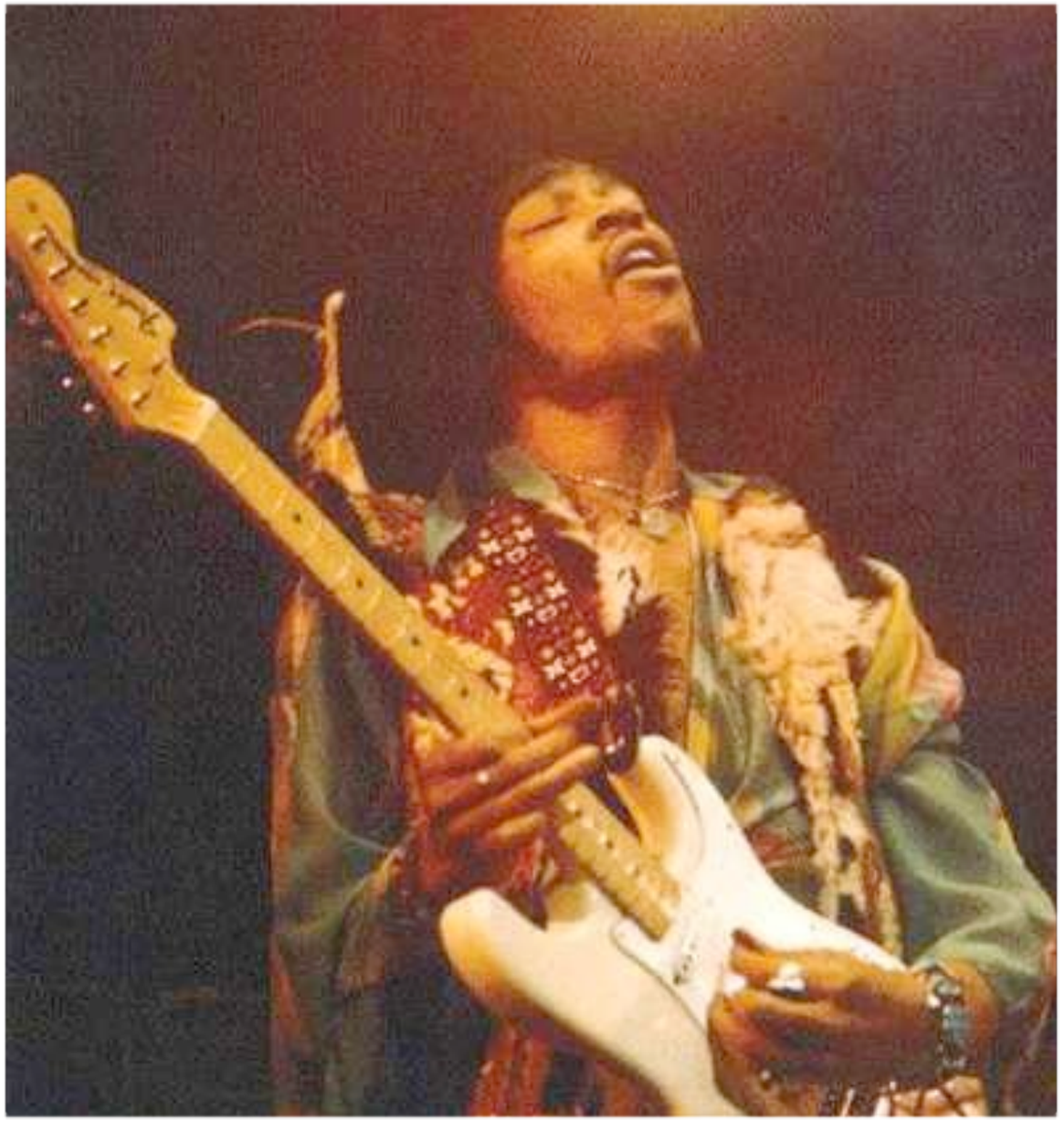
Ian Cross
Centre for Music & Science



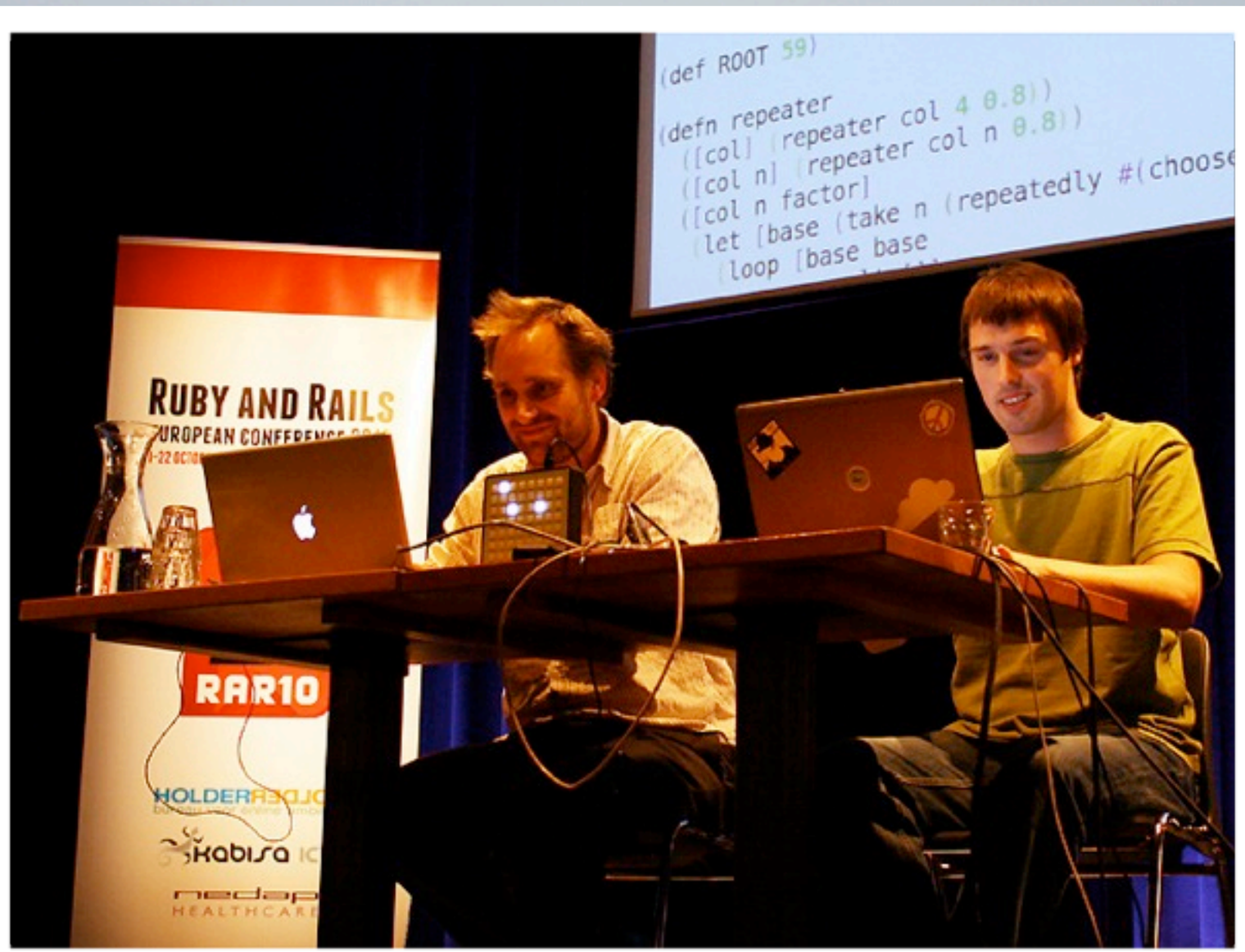
Richard Hoadley
Tom Hall
Digital Performance Lab

Tim Regan
Stuart Taylor
Microsoft Research Cambridge

Performance



Performance



Improvisation

Improvisation

Communication

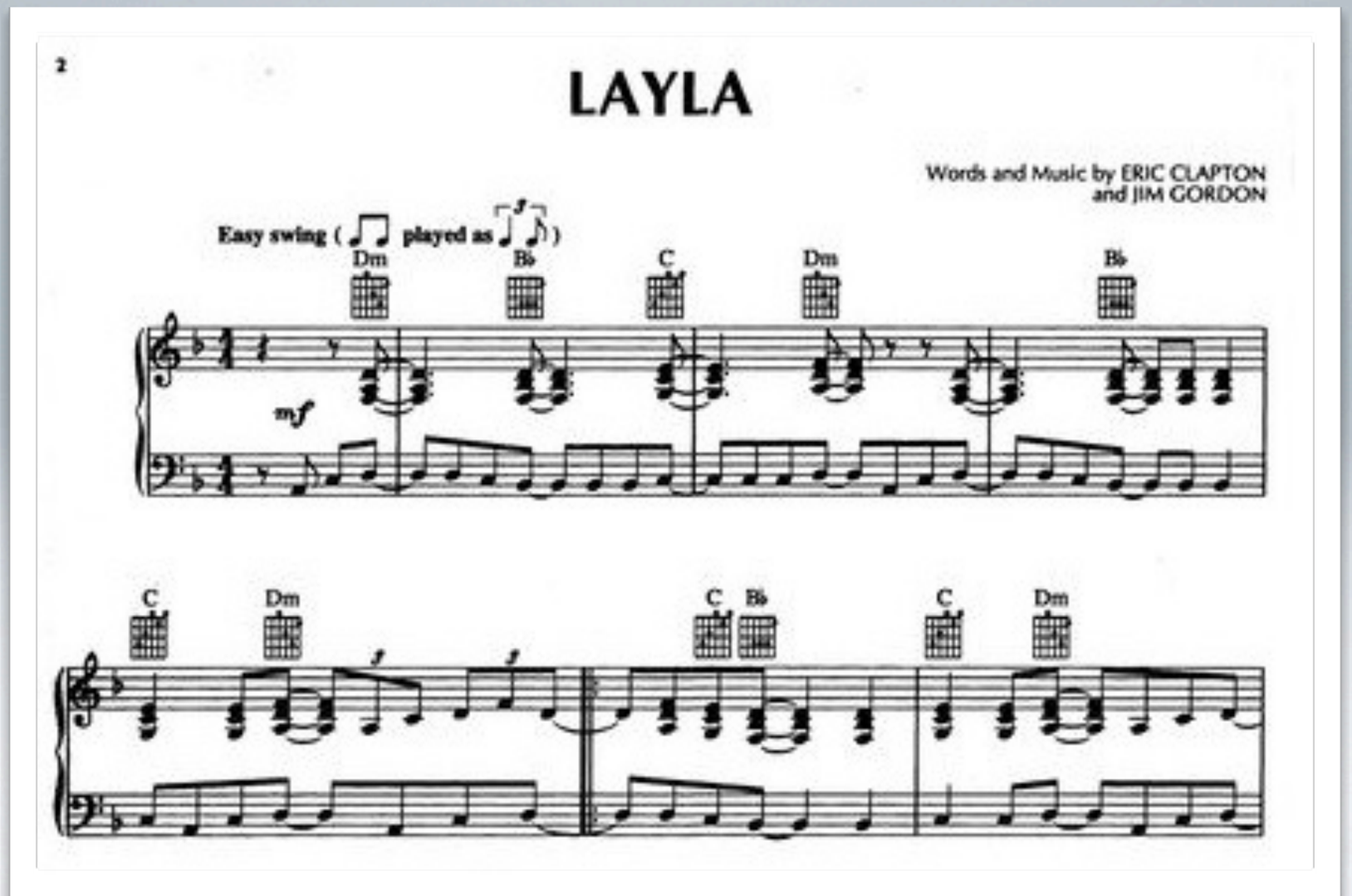
Communication

Interface

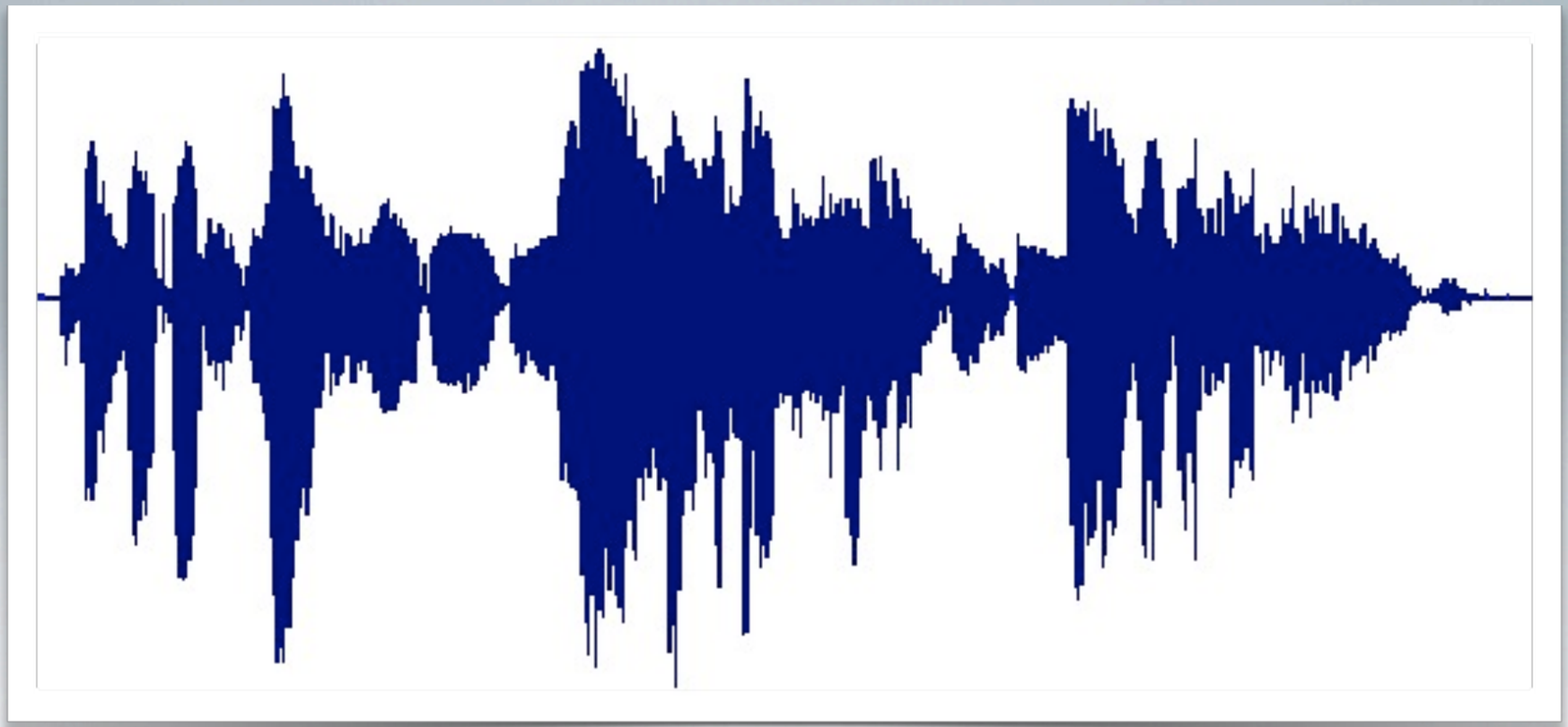
Interface


TAP
123.00
4 / 4
EXT
S
25 . 2 . 3
None
25 . 1 . 1
3 . 2 . 0
KEY
MIDI
64 %
Desktop
Testtones
pulse
sine
1kHz-12dB.aif
440Hz-0.1dB.aif
440Hz-12dB.aif
440Hz-30dB.aif
440Hz-60dB.aif
Static
1sec_ramp.aif
440-train
fadeout
fadeoutPT
fadeoutPT2.R
fadeoutPT3.R
fpt4
Live-PT-puls.aif
pan_LR_6dB
pan_R_6dB
pt5
sweep.aif
tri
tri2.R
VSTPlugins
Mitarbeiter
...bass
...drum ride
...funki guitar
...BD
...piano
...lo dr126
...fpt4
sweep
Track 1
Track 2
Track 3
Track 4
Track 5
Track 6
Track 7
Track 8
Track 9
Master
Resonators
Compressor II
EQ Three

Notation


2
LAYLA
Words and Music by ERIC CLAPTON
and JIM GORDON
Easy swing (♩♪ played as ♩♪)
Dm
B♭
C
Dm
B♭
mf
C
Dm
C
B♭
C
Dm

Notation



FINGERPICKING BEATLES

CD edition

Learn as you play

Drums

Chris Barron

LEARN TO PLAY

GUITAR

GARY TURNER

LEARN AS YOU PLAY SAXOPHONE

BY PETER WASTALL

LEARN AS YOU PLAY FRENCH HORN

BY PETER WASTALL

LEARN AS YOU PLAY OBOE

BY PETER WASTALL

PIANO ACCOMPANIMENT

CONCERT PIECES

LEARN AS YOU PLAY TUBA

BY PETER WASTALL

LEARN AS YOU PLAY FLUTE

BY PETER WASTALL

LEARN AS YOU PLAY BASSOON

BY PETER WASTALL

USC | THORNTON

SCHOOL OF MUSIC

GUITAR INSTRUCTIONAL SERIES

ARPEGGIOS

Evolving Guitarist

by Pat Kelley

JIMI HENDRIX

BAND OF GYPSYS

HAL LEONARD

LEARN AS YOU PLAY TROMBONE & EUPHONIUM

BY PETER WASTALL

ALSO FOR E♭ TENOR HORN & BARITONE

LEARN AS YOU PLAY FRENCH HORN

BY PETER WASTALL

LEARN AS YOU PLAY OBOE

BY PETER WASTALL

CASSETTE AVAILABLE

BOOSEY & HAWKES

(Score in C)

Commissioned by The Mobile Symphony, Scott Speck, Music Director
In honor of the Tricentennial of the City of Mobile, Alabama, 2002

Southern Exposure

for orchestra

JOHN DAVID EARNEST
(2002)

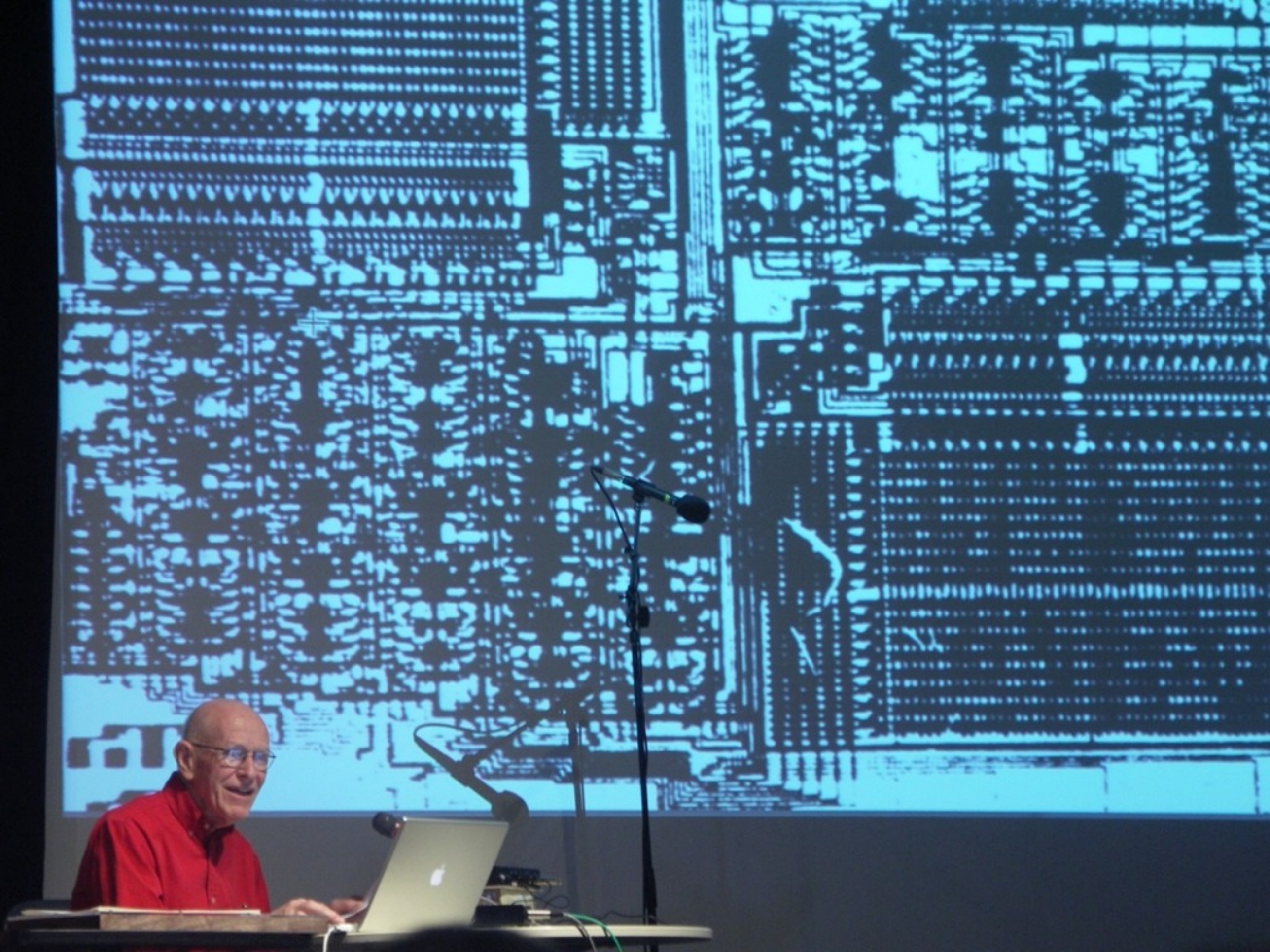
Joyous and vigorous (♩= 120 ca.)

1, 2 Flutes
3 / Piccolo
1, 2 Oboes
3 / English Horn
1, 2 Clarinets
3
1, 2 Bassoons
3
1, 3 Horns
2, 4
1, 2 Trumpets
3
Trombones 1, 2
Bass Trombone
Tuba
Timpani
Glsp.
1
Xyln.
2
Percussion
Vibes
3
Bs. Dr.
4
Harp
Piano

Joyous and vigorous (♩= 120 ca.)

1 Violin
2
Viola
Violoncello
Contrabass

Music engraved by Ipolitov Music Services, New York – www.ipolitovmusic.com



The Digital Computer as a Musical Instrument

A computer can be programmed to play "instrumental" music, to aid the composer, or to compose unaided.

M. V. Mathews

With the aid of suitable output equipment, the numbers which a modern digital computer generates can be directly converted to sound waves. The process is completely general, and any perceivable sound can be so produced. This potentiality of the computer has been of considerable use at the Bell Telephone Laboratories in generating stimuli for experiments in the field of hearing, and for generating speech sounds and connected speech in investigations of the factors which contribute to the intelligibility and naturalness of speech.

The quality of sound is of great importance in two fields—that of speech and communication and that of music. Our studies at the Bell Laboratories in the first of these fields have led us, over the past few years, to related studies in the production of musical sounds and their organization into mu-

chine for composing music. It can either compose pieces based entirely on random numbers generated by itself or it can cooperate with a human composer. It can play its own compositions.

Here I first describe the process for converting numbers to sounds, then I describe a program for playing music. Next I consider a psychoacoustic problem which is typical of those posed in attempts to make more interesting sounds. Finally, I look to the future, to the time when the computer is itself the composer.

Sound from Numbers

How can the numbers with which a computer deals be converted into sounds the ear can hear? The most general conversion is based upon the use of the numbers as samples of the

example, by running our computer at a rate of 30,000 numbers per second, we can generate sound waves with frequencies from 0 to 15,000 cycles per second. Waves in this frequency range are about the only ones the human ear can perceive.

The signal-to-quantizing-noise ratio of the sound wave depends on the accuracy with which the amplitudes of the pulses are represented. Computers deal with a finite number of digits and, hence, have limited accuracy. However, the computer limits are more than sufficient acoustically. For example, amplitudes represented by four-digit decimal numbers, are accurate to within 1 part in 10,000, an accuracy which represents a signal-to-noise ratio of 80 decibels; this is less noise than the ear can hear, and less noise than would be introduced by any audio equipment, such as the best tape recorder.

The sampling process just described is theoretically unrestricted, but the generation of sound signals requires very high sampling rates. The question should immediately be asked, "Are computers of the type now available capable of generating numbers at these rates?" The answer is "Yes," with some qualifications. A high-speed machine such as the I.B.M. 7090, using the programs described later in this article, can compute only about 5000 numbers per second when generating a reasonably complex sound. However, the numbers can be temporarily stored on one of the computer's digital magnetic tapes, and this tape can subsequently be replayed at rates up to 30,000 numbers per second (each number being a 12-bit binary number).

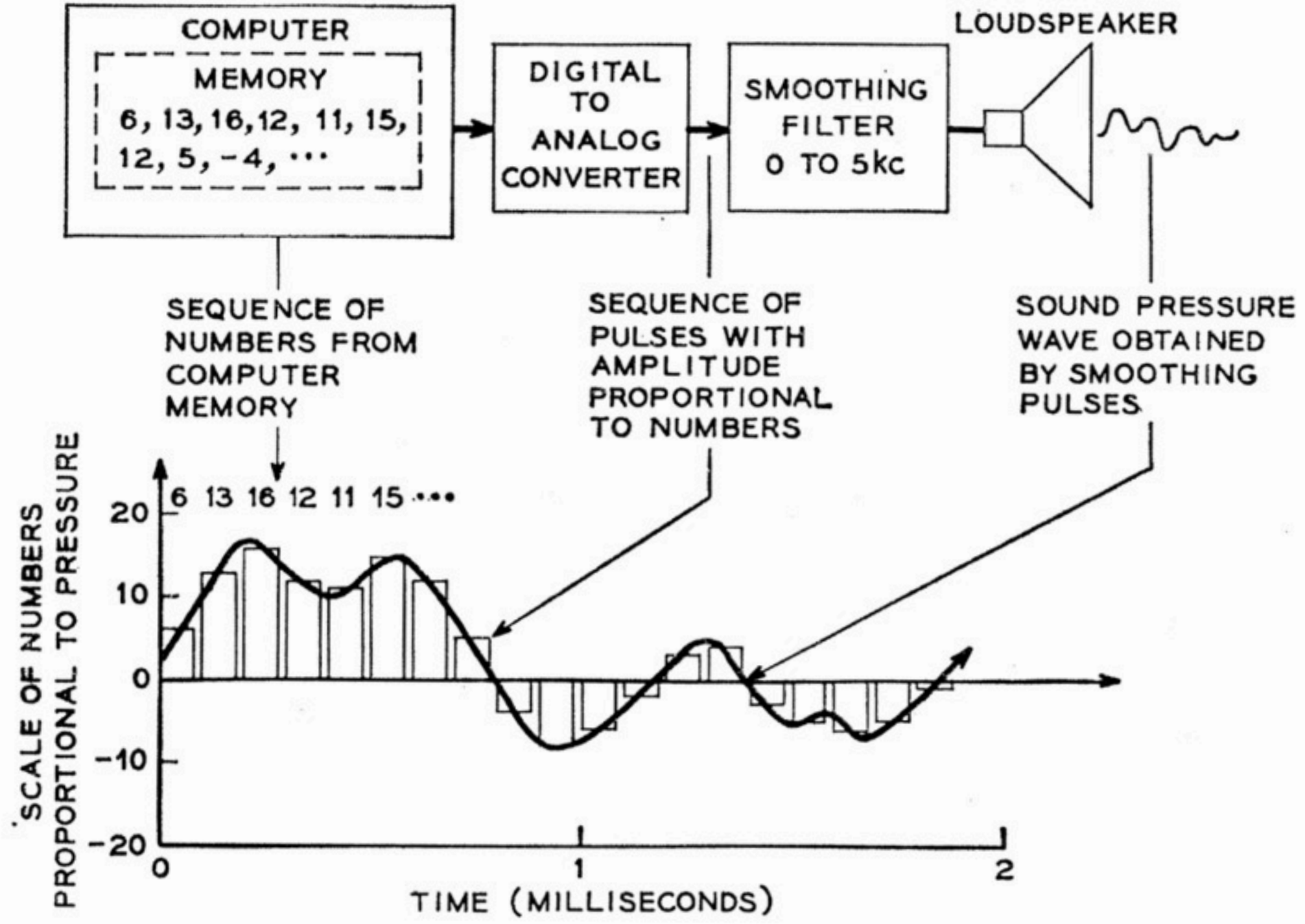


Fig. 1. Schematic diagram depicting the conversion of a sequence of numbers stored in a computer memory to a sound pressure wave form. The sampling rate is 10,000 numbers per second to yield a bandwidth of 5000 cycles per second for the sound wave.

Table 1. A typical computer score. The corresponding conventional score is shown in Fig. 3.

Operation code	Instrument No.	Starting time (sec)	Duration (sec)	Loudness (arbitrary units)	Frequency (cy/sec)	Periodic vibrato		Random vibrato	
						Amplitude (cy/sec)	Frequency (cy/sec)	Amplitude (cy/sec)	Bandwidth (cy/sec)
Play	1	0.0	0.25	1	466	0	0	7.0	6
Play	1	.5	.25	3	698	0	0	10.5	7
Play	1	1.0	.125	5	698	0	0	10.5	7.5
Play	1	1.5	.125	7	698	0	0	10.5	8
Play	1	2.0	.25	9	932	0	0	14.0	8.5
Play	1	2.25	.125	10	784	0	0	11.7	9
Play	2	0.5	.50	1	116.5	1.7	6	0	0
Play	2	1.5	.25	5	156	2.3	7	0	0
Play	2	2.0	.125	10	233	3.5	8	0	0

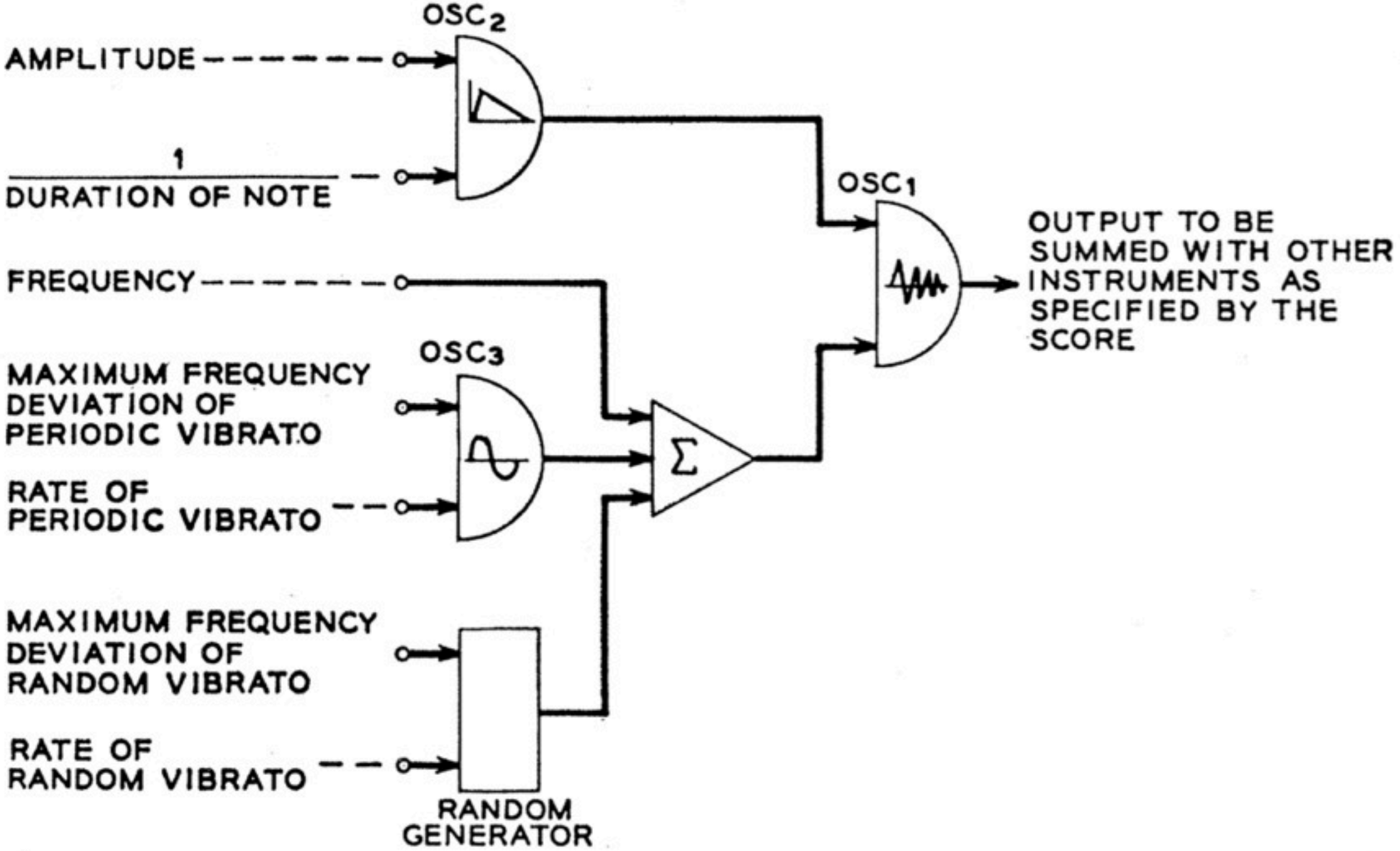
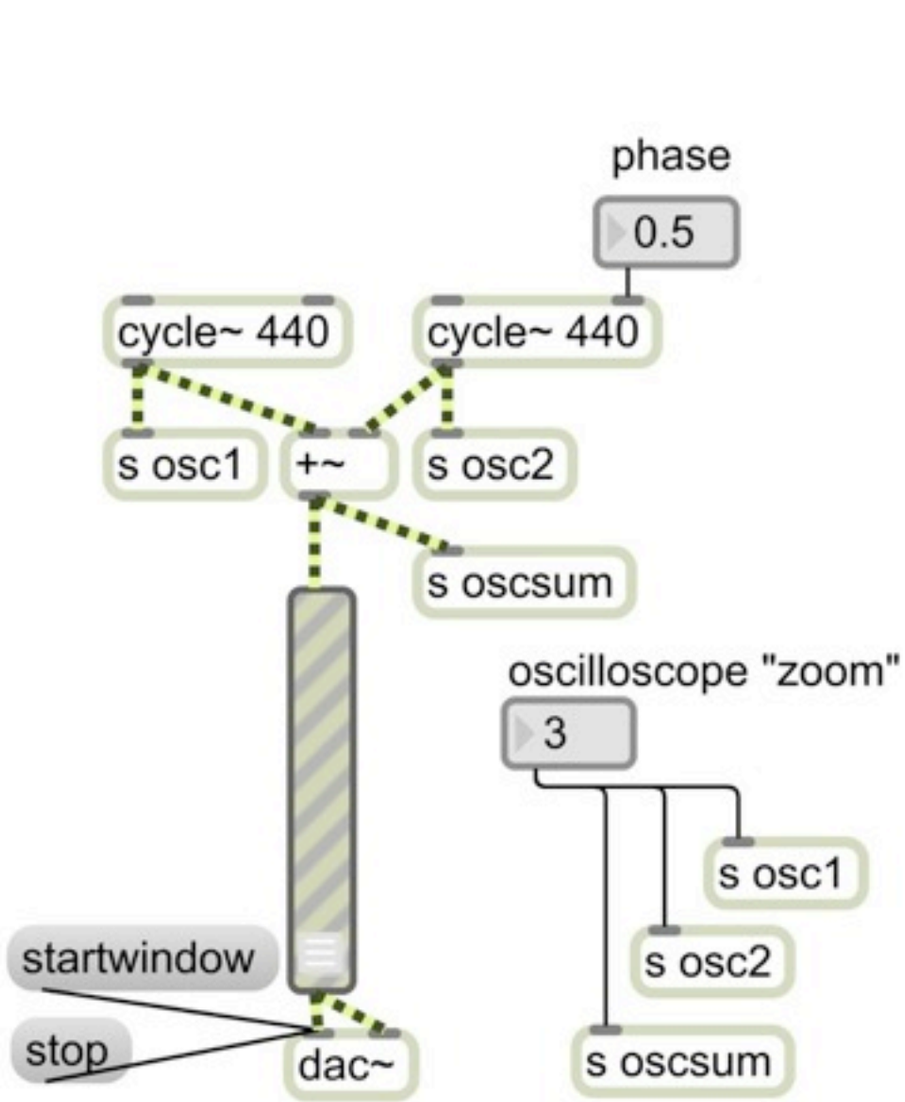
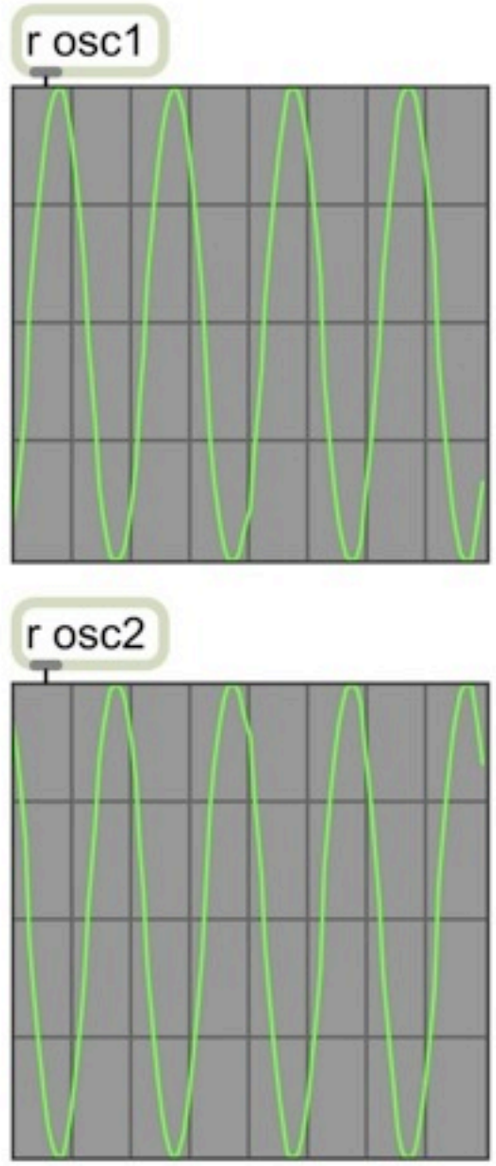
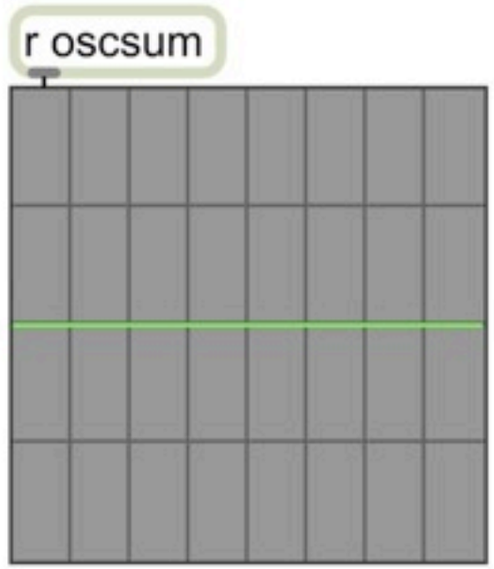


Fig. 2. Schematic diagram of a typical instrument-unit in the computer orchestra. The diagram represents a section of the computer program. In order for the computer to produce a note, numerical values for the note parameters shown at the left of the diagram are stored in the program. The program then generates samples of the sound pressure wave form.


phase
0.5
cycle~ 440
cycle~ 440
s osc1
+~
s osc2
s oscsum
oscilloscope "zoom"
3
s osc1
s osc2
s oscsum
startwindow
stop
dac~


r osc1
r osc2


r oscsum

sampler9

oliver leaman

8-track random playback sampler variables - volume jump & slide, speed, pitch, loop time, reverb rate & time, pan & pan fade rate

all on/off

loops samples~

MASTER VOLUME

random on/off

metro 500

p randomiser

random 8

all random volumes jump on/off

all random volumes slider on/off

all random loop start/end time on/off

all reverb on/off (max)

dry/wet (max)

reverb level random

dry/wet & backwards

p f-b-high

p f-b-med

p f-b-low

p f-b-v-low

p f-high

p f-med

p f-low

p f-v-low

fade-selector

random volume jump

random-volume-slider

random-pan

random-pan-fade

randomiser

random 3000

random 6000

select start ms

select end ms

load sample-1

load sample-2

load sample-3

load sample-4

load sample-5

load sample-6

load sample-7

load sample-8

replace

buffer~ sample-1

buffer~ sample-2

buffer~ sample-3

buffer~ sample-4

buffer~ sample-5

buffer~ sample-6

buffer~ sample-7

buffer~ sample-8

double-click to open buffer~ window

s buffer_out

name of most recently loaded file

info~ sample-1

info~ sample-2

info~ sample-3

info~ sample-4

info~ sample-5

info~ sample-6

info~ sample-7

info~ sample-8

piano-chord1.aif

piano-chord2.aif

piano-chord3.aif

piano-chord7.aif

piano-chord8.aif

prepend set

p pitch~mapper

groove~ sample-1

groove~ sample-2

groove~ sample-3

groove~ sample-4

groove~ sample-5

groove~ sample-6

groove~ sample-7

groove~ sample-8

edit only one end

undo

stop

record~ sample-1

record~ sample-2

record~ sample-3

record~ sample-4

record~ sample-5

record~ sample-6

record~ sample-7

record~ sample-8

bop 1

offset

set sample-1

set sample-2

set sample-3

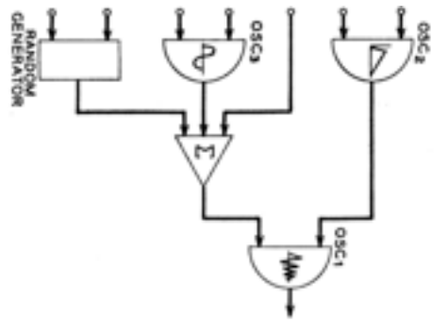
set sample-4

set sample-5

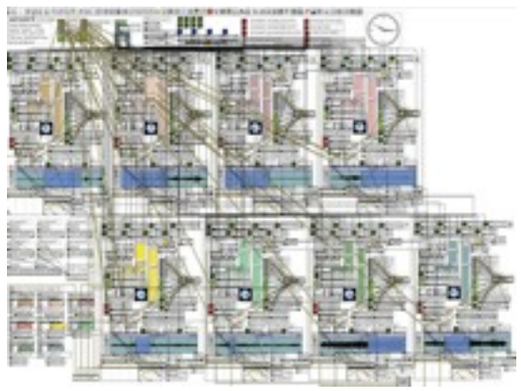
set sample-6

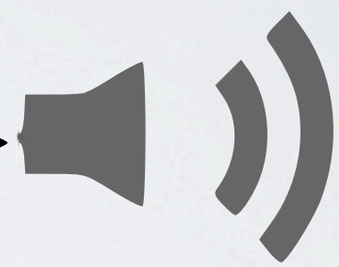
set sample-7

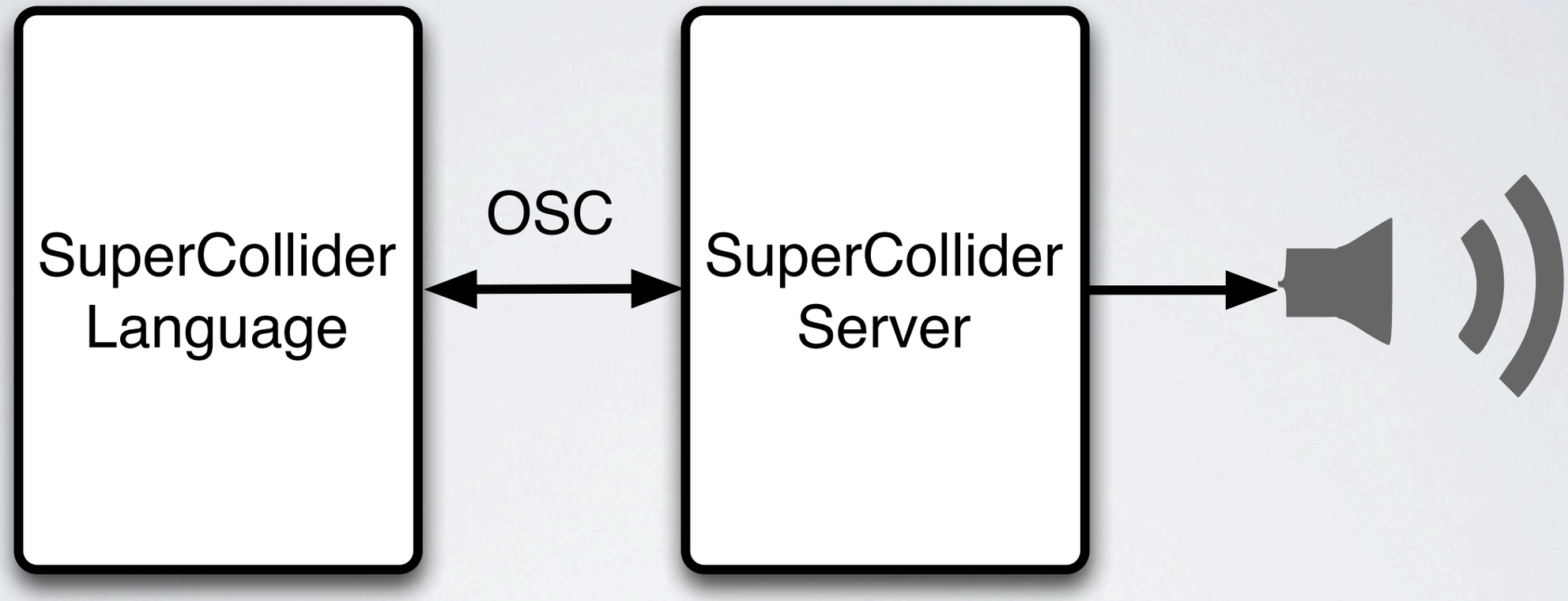
set sample-8



SuperCollider



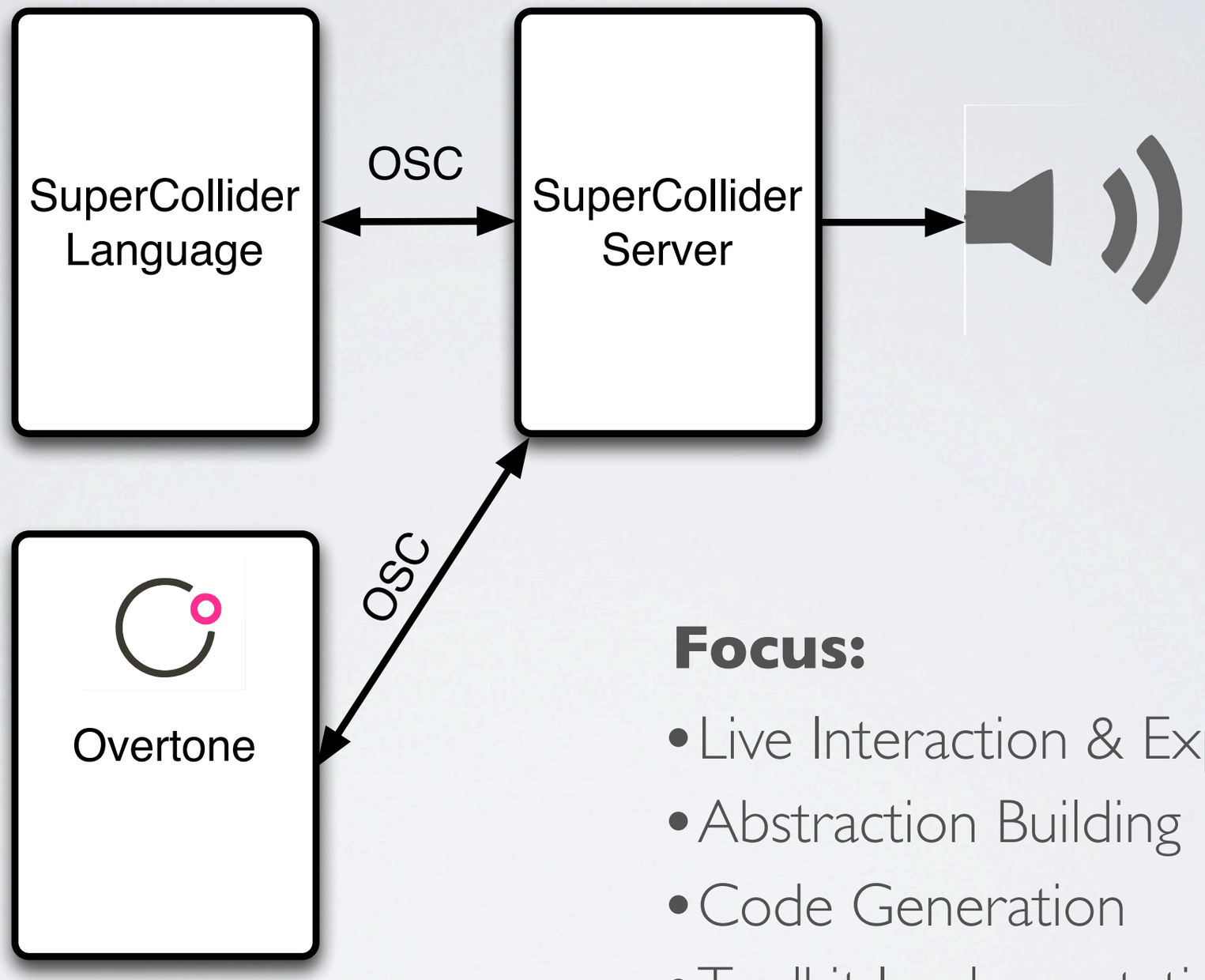



SuperCollider
Language
OSC
SuperCollider
Server

"One goal of separating the synthesis engine and the language in SC Server is to make it possible to explore implementing in other languages the concepts expressed in the SuperCollider language and class library"

James McCartney

Rethinking the Computer Music Language SuperCollider.
Computer Music Journal, 26(4):61-68, Dec. 2002



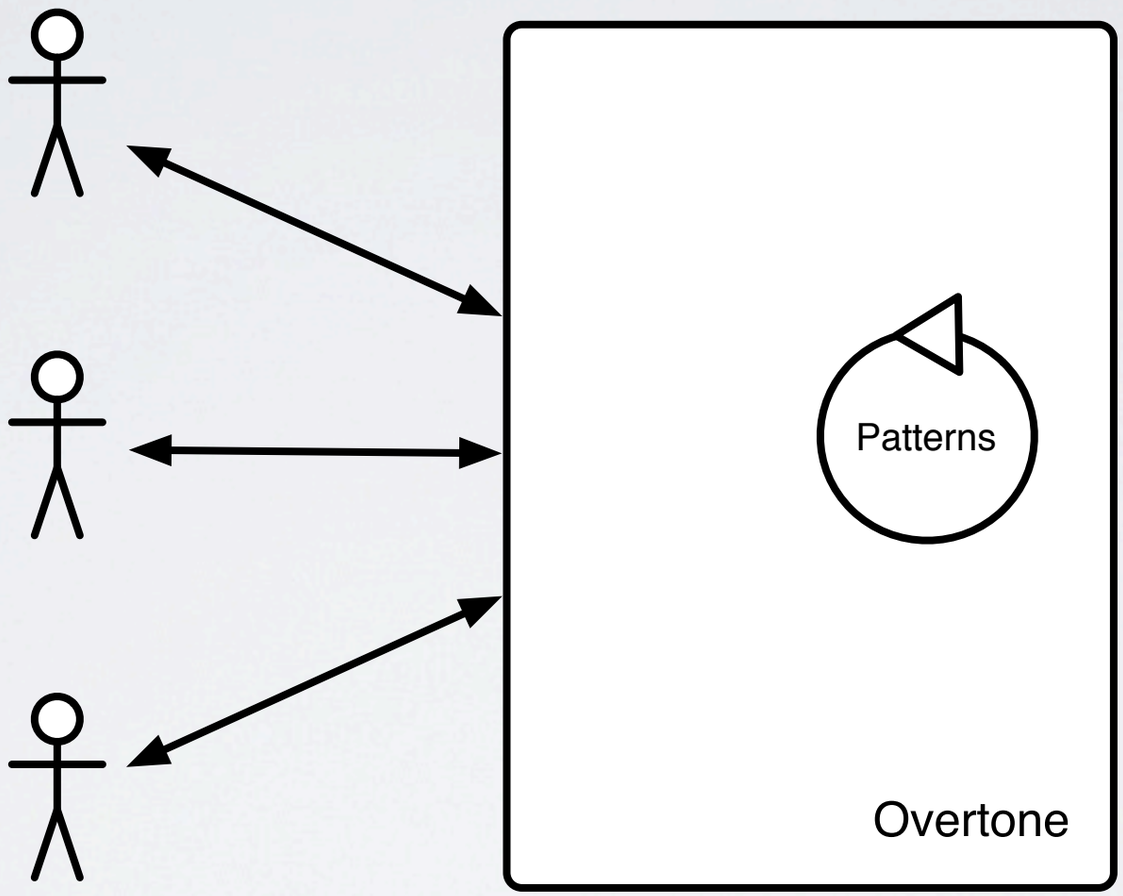
Focus:

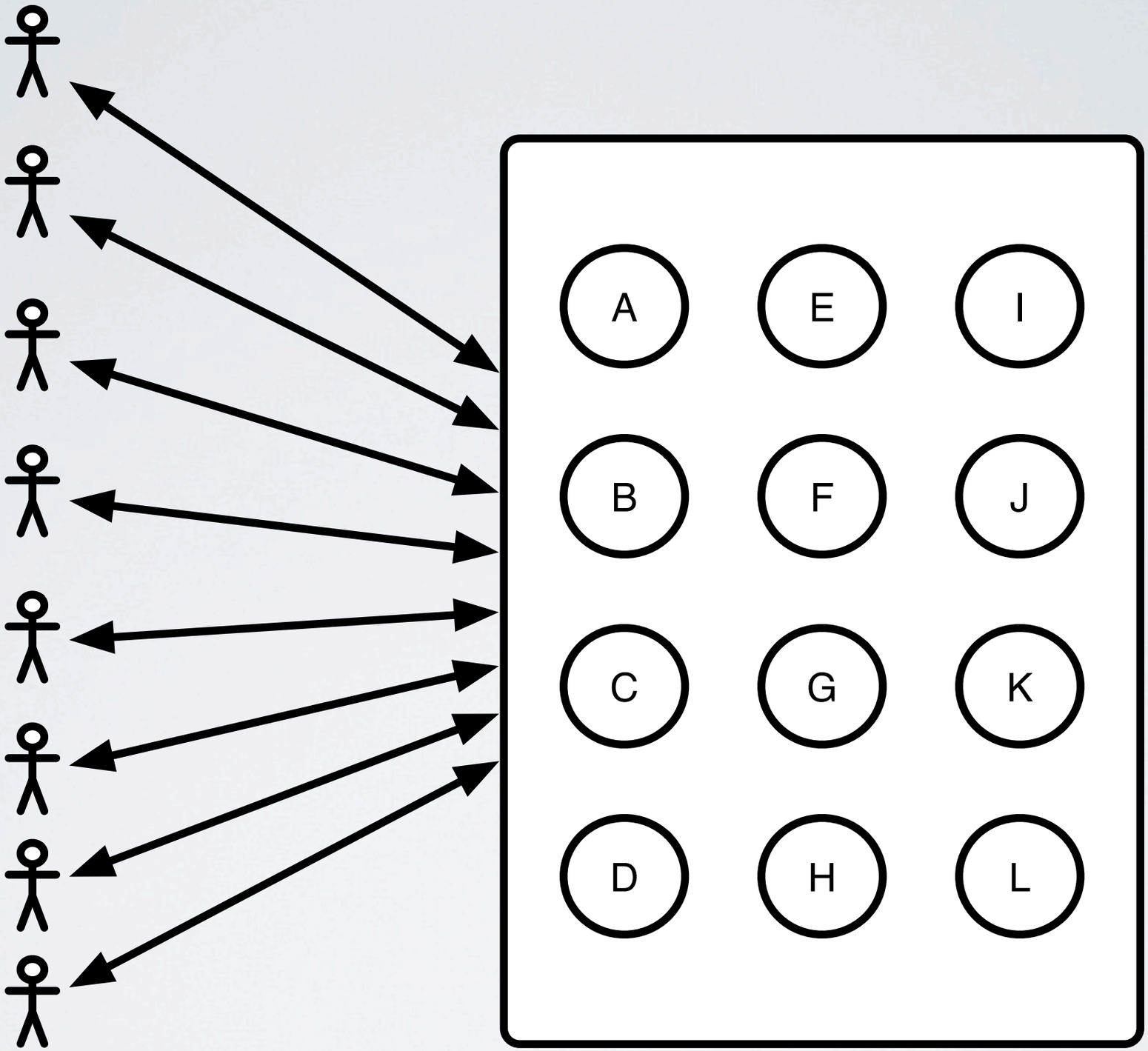
- Live Interaction & Exploration
- Abstraction Building
- Code Generation
- Toolkit Implementation
- Real World Connectivity

```
(defun eval. (e a)
  (cond
    ((atom e) (assoc. e a))
    ((atom (car e))
     (cond
       ((eq (car e) 'quote) (cadr e))
       ((eq (car e) 'atom)  (atom   (eval. (cadr e) a)))
       ((eq (car e) 'eq)    (eq     (eval. (cadr e) a)
                                    (eval. (caddr e) a)))
       ((eq (car e) 'car)   (car    (eval. (cadr e) a)))
       ((eq (car e) 'cdr)   (cdr    (eval. (cadr e) a)))
       ((eq (car e) 'cons)  (cons   (eval. (cadr e) a)
                                    (eval. (caddr e) a)))
       ((eq (car e) 'cond)  (evcon. (cdr e) a))
       ('t (eval. (cons (assoc. (car e) a)
                        (cdr e))
                  a))))
    ((eq (caar e) 'label)
     (eval. (cons (caddar e) (cdr e))
            (cons (list. (cadar e) (car e)) a)))
    ((eq (caar e) 'lambda)
     (eval. (caddar e)
            (append. (pair. (cadar e) (evlis. (cdr e) a))
                     a)))))
```

eq
car
cdr
atom
cond
quote
cons

Collaboration


Patterns
Overtone


A
E
I
B
F
J
C
G
K
D
H
L

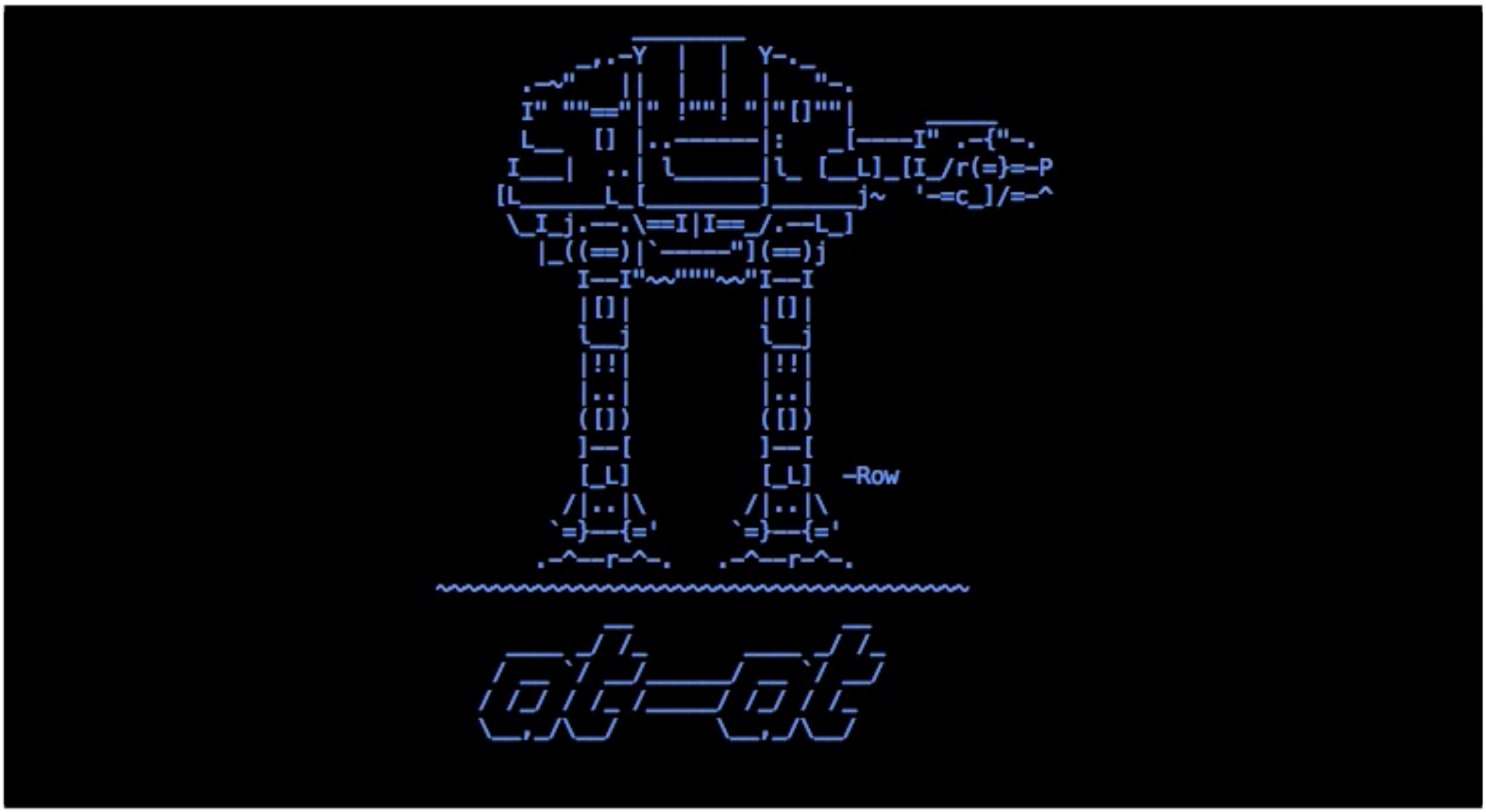
http://github.com/overtone/overtone

Programmable Music.

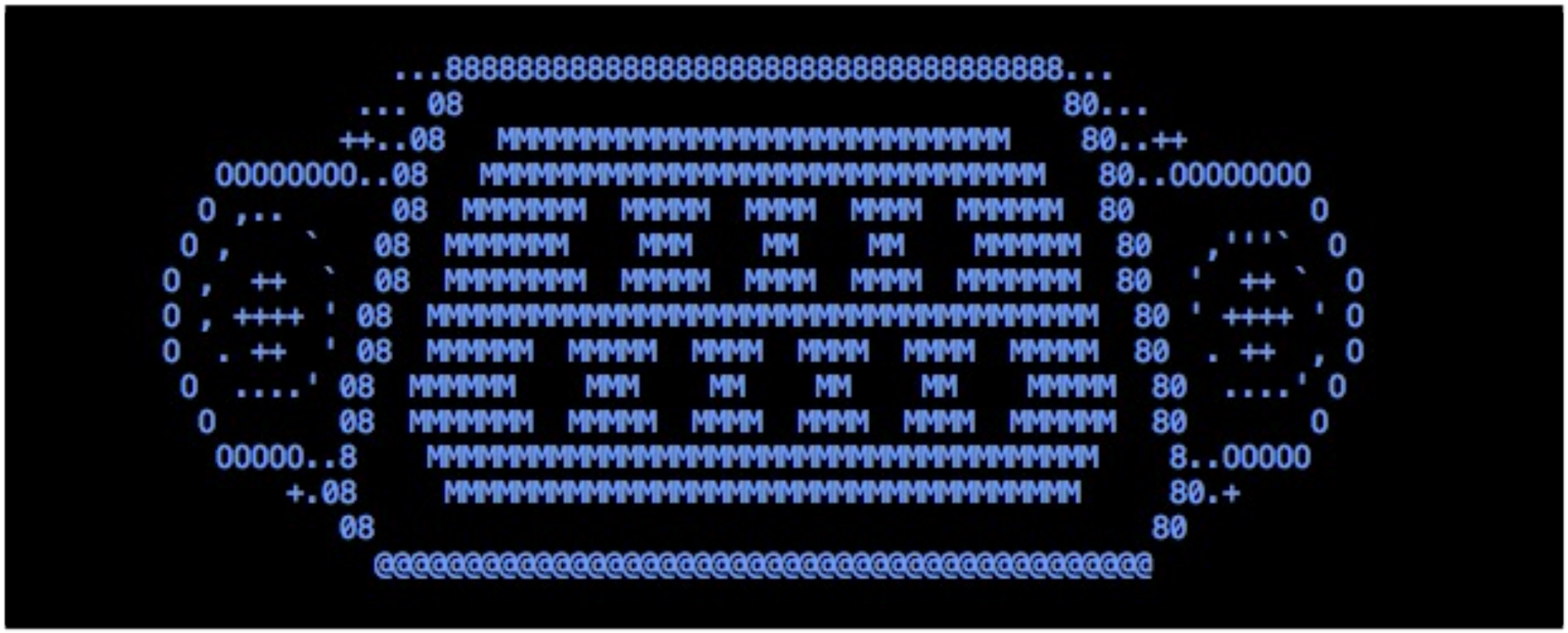
Live-coding & musical exploration

Overtone is an Open Source toolkit for creating synthesizers and making music. It provides:

* A Clojure API to the SuperCollider synthesis engine
* A growing library of musical functions (scales, chords, rhythms, arpeggiators, etc.)
* Metronome and timing system to support live-coding and sequencing
* Plug and play midi device I/O
* Simple Open Sound Control (OSC) message handling



(Ascii art borrowed from http://www.sanitarium.net/jokes/getjoke.cgi?132)




,~~~
+~~~~~
?~~~~+~
8~~~~~+~,
O~~~~~=+==
8?~~~~=+=~~
NI=~~~~~~~~~
DO?~~=7I~~~~=
Z8?~~~~~~:+~?$I,
ZNOI~~~~~~~~~~.~7$$Z7:~~ZZZOSCZZ+,
ZDO7:~~~=~~~~~~
ZDOOI~~?~~~~=====
~NDOO?~~~~~~~~~~
:I OO$+~~~~=~~=~
DDDD8D?~~~~~~~=====~
D DDDN8Z=~~~~~===~~~~
? DDDDN8O7~~~~~~~~=====.7
N888DNMOO?~~~~~===~~~$?
O88+DM ~NNN88OI~~~~~=~~~+~~?/$
D OVERTONE88I~~~~~===~~~,~
D ~ INC.NNN888I~~~~~=~~==,
:I ND~ DDN8880~~~?=======,
8 8DDD ~ M88887:I~~======
D 8DDDDND D 88$~~~===+z
:D8MDDDNN DD87~~I~+=
88DDDDDN DD7?~~
$CLOJURE DD`
$8DSDDDD
DDDCDDDN
::7OLIBRARY$$$77I$IZ7
.d88b. .d8888b .d8888b .d8888b
d88""88b 88K d88P" d88P"
888 888 "Y8888b. 888 888888 888
Y88..88P X88 Y88b. Y88b.
"y88p" 88888P' "Y8888P "Y8888P
888 d8b
888 Y8P
888
888 8888
888 "888
888 888
888 888
888 888
888
d88P
888P"

```clojure
(on-sync-event :reset
  (λ []
    (clear-msg-queue)
    (group-clear @synth-group*))
  ::reset-base)
```

```
(on-deps [:server-connected :core-groups-created]
         ::signal-server-ready
  f(satisfy-deps :server-ready))
```

```
(def OSC-TYPE-SIGNATURES
  {
   ;;Master Controls
   "/quit"                    []
   "/notify"                  [:zero-or-one]
   "/status"                  []
   "/cmd"                     [:int :anything*]
   "/dumpOSC"                 [:zero-to-three]
   "/sync"                    [:int]
   "/clearSched"              []
   "/error"                   [:minus-two-to-one]

   ;;Synth Definition Commands
   "/d_recv"                  [:bytes]
   "/d_load"                  [:pathname]
   "/d_loadDir"               [:pathname]
   "/d_free"                  [:synthdef-name]
```

```clojure
(def SCALE
  (let [ionian-sequence [2 2 1 2 2 2 1]
        ionian-len    (count ionian-sequence)
        rotate-ionian (λ [offset]
                        (drop offset (take (+ ionian-len offset)
                                           (cycle ionian-sequence))))]
    {:diatonic          ionian-sequence
     :ionian            (rotate-ionian 0)
     :major             (rotate-ionian 0)
     :dorian            (rotate-ionian 1)
     :phrygian          (rotate-ionian 2)
     :lydian            (rotate-ionian 3)
     :mixolydian        (rotate-ionian 4)
     :aeolian           (rotate-ionian 5)
     :minor             (rotate-ionian 5)
     :lochrian          (rotate-ionian 6)
     :pentatonic        [2 3 2 2 3]
     :major-pentatonic  [2 2 3 2 3]
     :minor-pentatonic  [3 2 2 3 2]
     :whole-tone        [2 2 2 2 2 2]
     :chromatic         [1 1 1 1 1 1 1 1 1 1 1 1]
     :harmonic-minor    [2 1 2 2 1 3 1]
     :melodic-minor-asc [2 1 2 2 2 2 1]
     :hungarian-minor   [2 1 3 1 1 3 1]
     :octatonic         [2 1 2 1 2 1 2 1]
     :messiaen1         [2 2 2 2 2 2]
     :messiaen2         [1 2 1 2 1 2 1 2]
     :messiaen3         [2 1 1 2 1 1 2 1 1]
     :messiaen4         [1 1 3 1 1 1 3 1]
     :messiaen5         [1 4 1 1 4 1]
     :messiaen6         [2 2 1 1 2 2 1 1]
     :messiaen7         [1 1 1 2 1 1 1 1 2 1]}))
```

```clojure
(def CHORD
  (let [major  #{0 4 7}
        minor  #{0 3 7}
        major7 #{0 4 7 11}
        dom7   #{0 4 7 10}
        minor7 #{0 3 7 10}
        aug    #{0 4 8}
        dim    #{0 3 6}
        dim7   #{0 3 6 9}]
    {:1          #{0}
     :5          #{0 7}
     :+5         #{0 4 8}
     :m+5        #{0 3 8}
     :sus2       #{0 2 7}
     :sus4       #{0 5 7}
     :6          #{0 4 7 9}
     :m6         #{0 3 7 9}
     :7sus2      #{0 2 7 10}
     :7sus4      #{0 5 7 10}
     :7-5        #{0 4 6 10}
     :m7-5       #{0 3 6 10}
     :7+5        #{0 4 8 10}
     :m7+5       #{0 3 8 10}
     :9          #{0 4 7 10 14}
     :m9         #{0 3 7 10 14}
     :maj9       #{0 4 7 11 14}
     :9sus4      #{0 5 7 10 14}
     :6*9        #{0 4 7 9 14}
     :m6*9       #{0 3 9 7 14}
     :7-9        #{0 4 7 10 13}
     :m7-9       #{0 3 7 10 13}
     :7-10       #{0 4 7 10 15}
     :9+5        #{0 10 13}
     :m9+5       #{0 10 14}
     :7+5-9      #{0 4 8 10 13}
     :m7+5-9     #{0 3 8 10 13}
     :11         #{0 4 7 10 14 17}
     :m11        #{0 3 7 10 14 17}
     :maj11      #{0 4 7 11 14 17}
     :11+        #{0 4 7 10 14 18}
     :m11+       #{0 3 7 10 14 18}
     :13         #{0 4 7 10 14 17 21}
     :m13        #{0 3 7 10 14 17 21}
     :major       major
     :M           major
     :minor       minor
     :m           minor
     :major7      major7
     :dom7        dom7
     :7           dom7
     :M7          major7
     :minor7      minor7
     :m7          minor7
     :augmented   aug
     :a           aug
     :diminished  dim
     :dim         dim
     :i           dim
     :diminished7 dim7
     :dim7        dim7
     :i7          dim7}))
```

```clojure
(definst small-hat [ffreq 200 rq 0.5
                    attack 0 release 0.025 amp 0.3
                    pan 0]
  (let [snd (white-noise)
        snd (hpf snd ffreq)
        snd (rhpf snd (* ffreq 2) rq)
        snd (* snd (env-gen (perc attack release 1 -10) :action FREE))]
    (* 2 snd amp)))
```

Interface

```
%w[tempfile uri].map{|l|require l};class Object;def meta_def m,&b;(class<<self
self end).send:define_method,m,&b end end;module Camping;C=self
S=IO.read(__FILE__)rescue nil;P="<h1>Cam\\ping Problem!</h1><h2>#{e
class H<Hash
def method_missing n,*a;n.to_s=~/=$/?self[$`]=a[0]:a==[]?self[n.to_s]:super end
alias u merge!;undef id,type;end;module Helpers;def R c,*g
p,h=/\(.+?\)/,g.grep(Hash);g-=h;raise"bad route"unless u=c.urls.find{|x|
break x if x.scan(p).size==g.size&&/^#{x}\/?$/=~(x=g.inject(x){|x,a|
x.sub p,C.escape((a[a.class.primary_key]rescue a))})}
h.any?? u+"?"+h[0].map{|x|x.map{|z|C.escape z}*"="}*"&":u end;def / p
p[/^\//]?@root+p:p end;def URL c='/',*a;c=R(c, *a) if c.respond_to?:urls
c=self/c;c="//"+@env.HTTP_HOST+c if c[/^\//];URI c end end;module Base
attr_accessor:input,:cookies,:env,:headers,:body,:status,:root;Z="\r\n"
def method_missing *a,&b;a.shift if a[0]==:render;m=Mab.new({},self)
s=m.capture{send(*a,&b)};s=m.capture{send(:layout){s}}if/^_/!~a[0].to_s and
m.respond_to?:layout;s end;def r s,b,h={};@status=s;headers.u(h);@body=b
end;def redirect *a;r 302,'','Location'=>URL(*a)end;def r404 p=env.PATH
r 404,P%"#{p} not found"end;def r500 k,m,x
r 500,P%"#{k}.#{m}"+"<h3>#{x.class} #{x.message}: <ul>#{x.
backtrace.map{|b|"<li>#{b}</li>"}}</ul></h3>"end;def r501 m=@method
r 501,P%"#{m.upcase} not implemented"end;def to_a
[status,body,headers]end;def initialize r,e,m;@status,@method,@env,@headers,
@root=200,m,e,H['Content-Type','text/html'],e.SCRIPT_NAME.sub(/\/$/,'')
@k=C.kp e.HTTP_COOKIE;q=C.qsp e.QUERY_STRING;@in=r;case e.CONTENT_TYPE
when%r|\Amultipart/form-.*boundary=\"?([^\";,]+)|n
b=/(?:\r?\n|\A)#{Regexp.quote"--#$1"}(?:--)?\r$/;until@in.eof?;fh=H[]
for l in@in;case l;when Z;break;when/^Content-D.+?: form-data;/
fh.u H['*$'.scan(/(?:\s(\w+)="([^"]+)")/).flatten]
when/^Content-Type: (.+?)(\r$|\Z)/m: fh.type = $1 end end;fn=fh.name
o=if fh.filename;c=fh.tempfile=Tempfile.new(:C);o.binmode;else;fh="";end;s=8192
k='';l=@in.read(s*2);while l;if(k<<l)=~b;o<<$`.chomp
@in.seek(-$'.size,IO::SEEK_CUR);break end;o<<k.slice!(0...s);l=@in.read(s) end
C.qsp(fn,'&;',fh,q)if fn;fh.tempfile.rewind if fh.is_a?H end;when
"application/x-www-form-urlencoded": q.u(C.qsp(@in.read))end
@cookies,@input=@k.dup,q.dup end;def service *a;@body=send @method,*a
headers['Set-Cookie']=cookies.map{|k,v|"#{k}=#{C.escape(v)}; path=#{self/
"/"}"if v!=@k[k]}-[nil];self end;def to_s;"Status: #@status#{Z+(headers.inject([
])){|a,o|["*o[1]].map{|v|a<<[o[0],v]*": "if v&&v.to_s.any?};a}*Z}#{Z+Z}#@body"end
end;X=module Controllers;@r=[];class<<self;def r;@r end;def R *u;r=@r
Class.new{meta_def(:urls){u};meta_def(:inherited){|x|r<<x}}end
def D p,m;r.map{|k|k.urls.map{|x|return(k.instance_method(m)rescue nil)?
[k,m,*$~[1..-1]]:[I,'r501',m]if p=~/^#{x}\/?$/}};[I,'r404',p]
end;def M;def M;end;constants.map{|c|k=const_get(c)
k.send:include,C,Base,Helpers,Models;@r=[k]+r if r-[k]==r
k.meta_def(:urls){["/#{c.downcase}"]}if !k.respond_to?:urls}end end;class I<R()
end;self end;class<<self;def goes m
eval S.gsub(/Camping/,m.to_s),TOPLEVEL_BINDING end;def escape s
s.to_s.gsub(/[^ \w.-]+/n){'%'+($&.unpack('H2'*$&.size)*'%').upcase}.tr' ','+'
end;def un s;s.tr('+',' ').gsub(/%([\da-f]{2})/in){[$1].pack'H*'}end
def qsp q,d='&;',y=nil,z=H[];m=proc{|_,o,n|o.u(n,&m)rescue([*o]<<n)}
(q.to_s.split(/[#{d}]+ */n)-[""]).inject((b,z=z,H[])[0]){|h,p|k,v=un(p).
split'=',2;h.u k.split(/[\]\[]+/).reverse.inject(y||v){|x,i|H[i,x]},&m}end
def kp s;c=qsp s,';','end;def run r=$stdin,e=ENV;X.M;e=H[e.to_hash];k,m,*a=X.D e.
PATH_INFO=un("/#{e.PATH_INFO}".gsub(/\/+/,'/')),
(e.REQUEST_METHOD||'get').downcase
k.new(r,e,m).Y.service(*a);rescue=>x;X::I.new(r,e,'r500').service k,m,x
end;def method_missing m,c,*a;X.M;k=X.const_get(c).new StringIO.new,
H['HTTP_HOST','','SCRIPT_NAME','','HTTP_COOKIE',''],m.to_s
H[a.pop].each{|e,f|k.send"#{e}=",f}if Hash===a[-1];k.service(*a)end end
module Views;include X,Helpers end;module Models;autoload:Base,'camping/db'
def Y;self;end end;autoload:Mab,'camping/mab'end
```

```
%w[tempfile uri].map{|l|require l};class Object;def meta_def m,&b;(class<<self
self end).send:define_method,m,&b end end;module Camping;C=self
S=IO.read(__FILE__)rescue nil;P="<h1>Cam\\ping Problem!</h1><h2>%s</h2>"
class H<Hash
 method_missing m,*a;m.to_s=~/=$/?self[$`]=a[0]:a==[]?self[m.to_s]:super end
alias u merge!;undef id,type;end;module Helpers;def R c,*g
p,h=/\(.+?\)/,g.grep(Hash);g-=h;raise"bad route"unless u=c.urls.find{|x|
break x if x.scan(p).size==g.size&&/^#{x}\/?$/=~(x=g.inject(x){|x,a|
x.sub p,C.escape((a[a.class.primary_key]rescue a))})}
h.any?? u+"?"+h[0].map{|x|x.map{|z|C.escape z}*"="}*"&":u end;def / p
 p[/^\//]?@root+p:p end;def URL c='/',*a;c=R(c, *a) if c.respond_to?:urls
c=self/c:c="//"+@env.HTTP_HOST+c if c[/^\//];URI c end end;module Base
 attr_accessor:input,:cookies,:env,:headers,:body,:status,:root;Z="\r\n"
 def method_missing *a,&b;a.shift if a[0]==:render;m=Mab.new({},self)
s=m.capture{send(*a,&b)};s=m.capture{send(:layout){s}}if/^_/!~a[0].to_s and
 m.respond_to?:layout;s end;def r s,b,h={};@status=s;headers.u(h);@body=b
 end;def redirect *a;r 302,'','Location'=>URL(*a)end;def r404 p=env.PATH
 r 404,P%"#{p} not found"end;def r500 k,m,x
 r 500,P%"#{k}.#{m}"+"<h3>#{x.class} #{x.message}: <ul>#{x.
backtrace.map{|b|"<li>#{b}</li>"}}</ul></h3>"end;def r501 m=@method
  r 501,P%"#{m.upcase} not implemented"end;def to_a
 [status,body,headers]end;def initialize r,e,m;@status,@method,@env,@headers,
@root=200,m,e,H['Content-Type','text/html'],e.SCRIPT_NAME.sub(/\/$/,'')
@k=C.kp e.HTTP_COOKIE;q=C.qsp e.QUERY_STRING;@in=r;case e.CONTENT_TYPE
 when%r|\Amultipart/form-.*boundary=\"?([^\";,]+)|n
 b=/(?:\r?\n|\A)#{Regexp.quote"--#$1"}(?:--)?\r$/;until@in.eof?;fh=H[]
for l in@in;case l;when Z;break;when/^Content-D.+?: form-data;/
 fh.u H[*$'.scan(/(?:\s(\w+)="([^"]+)")/).flatten]
 when/^Content-Type: (.+?)(\r$|\Z)/m: fh.type = $1 end end;fn=fh.name
o=if fh.filename;o=fh.tempfile=Tempfile.new(:C);o.binmode;else;fh="";end;s=8192
k='';l=@in.read(s*2);while l;if(k<<l)=~b;o<<$`.chomp
@in.seek(-$'.size,IO::SEEK_CUR);break end;o<<k.slice!(0...s);l=@in.read(s) end
C.qsp(fn,'&;',fh,q)if fn;fh.tempfile.rewind if fh.is_a?H end;when
 "application/x-www-form-urlencoded": q.u(C.qsp(@in.read))end
@cookies,@input=@k.dup,q.dup end;def service *a;@body=send @method,*a
headers['Set-Cookie']=cookies.map{|k,v|"#{k}=#{C.escape(v)}; path=#{self/
"/"}"if v!=@k[k]}-[nil];self end;def to_s;"Status: #@status#{Z+(headers.inject([
]){|a,o|[*o[1]].map{|v|a<<[o[0],v]*": "if v&&v.to_s.any?};a}*Z)+Z+Z}#@body"end
end;X=module Controllers;@r=[];class<<self;def r;@r end;def R *u;r=@r
Class.new{meta_def(:urls){u};meta_def(:inherited){|x|r<<x}}end
 def D p,m;r.map{|k|k.urls.map{|x|return(k.instance_method(m)rescue nil)?
[k,m,*$~[1..-1]]:[I,'r501',m]if p=~/^#{x}\/?$/}};[I,'r404',p]
end;def M;def M;end;constants.map{|c|k=const_get(c)
k.send:include,C,Base,Helpers,Models;@r=[k]+r if r-[k]==r
k.meta_def(:urls){["/#{c.downcase}"]}if !k.respond_to?:urls}end end;class I<R()
 end;self end;class<<self;def goes m
eval S.gsub(/Camping/,m.to_s),TOPLEVEL_BINDING end;def escape s
s.to_s.gsub(/[^ \w.-]+/n){'%'+($&.unpack('H2'*$&.size)*'%').upcase}.tr' ','+'
 end;def un s;s.tr('+',' ').gsub(/%([\da-f]{2})/in){[$1].pack'H*'}end
def qsp q,d='&;',y=nil,z=H[];m=proc{|_,o,n|o.u(n,&m)rescue([*o]<<n)}
q.to_s.split(/[#{d}]+ */n)-[""]).inject((b,z=z,H[])[0]){|h,p|k,v=un(p).
split'=',2;h.u k.split(/[\]\[]+/).reverse.inject(y||v){|x,i|H[i,x]},&m}end
def kp s;c=qsp s,';','end;def run r=$stdin,e=ENV;X.M;e=H[e.to_hash];k,m,*a=X.D e.
```

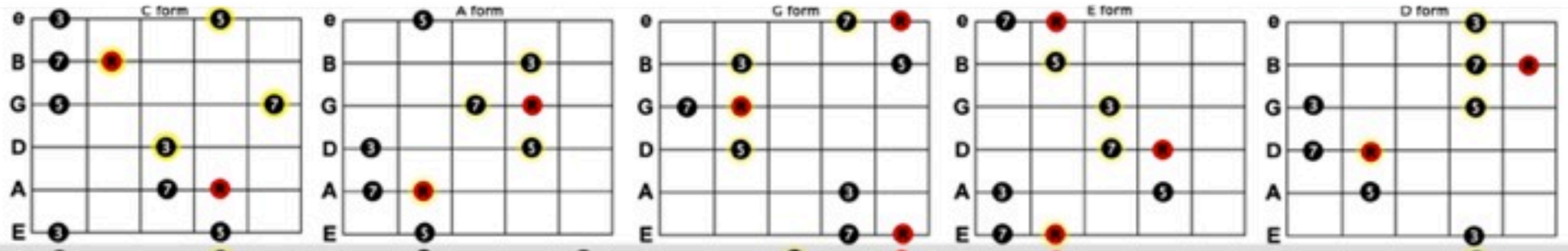
TALK TO ME

Design and the Communication between People and Objects

MoMA.org/talktome

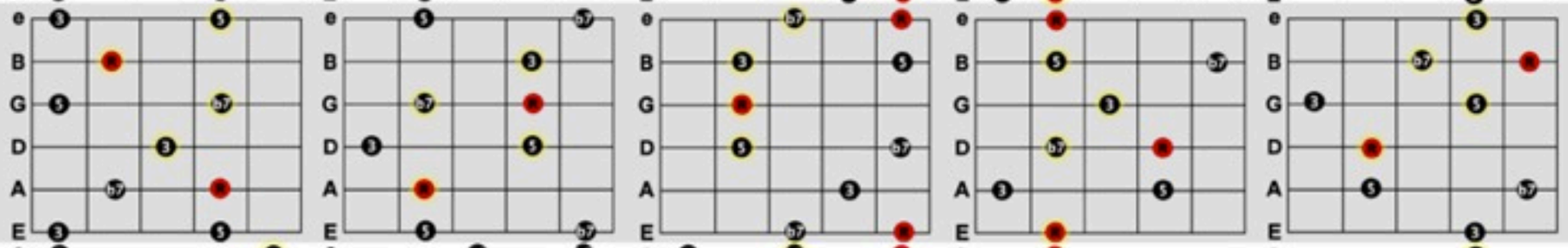
Major 7th

R W 3 W h 5 W 7 W W



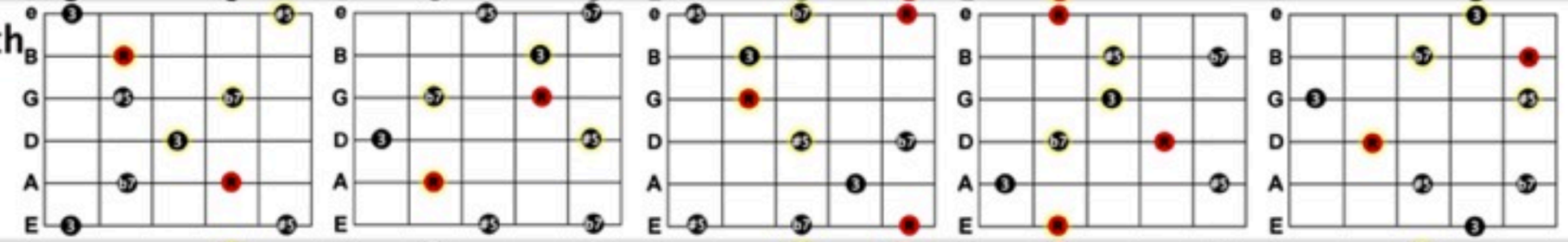
Dominant 7th

R W 3 W h 5 W ♭7 h



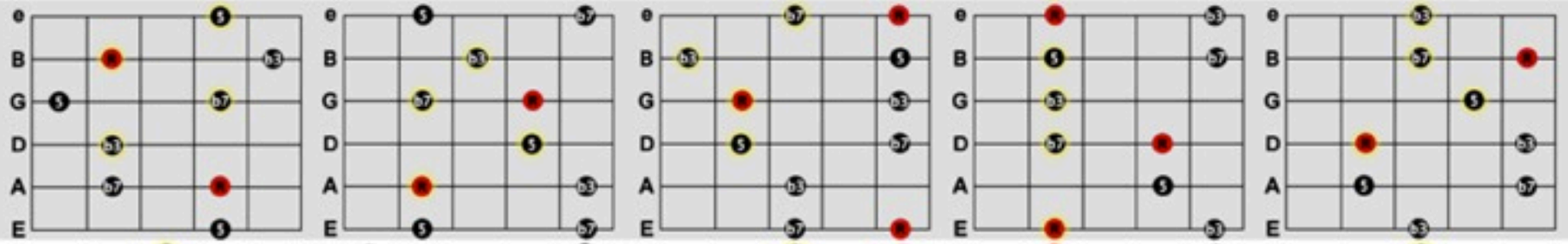
Augmented 7th

R W 3 W ♯5 W ♭7 W



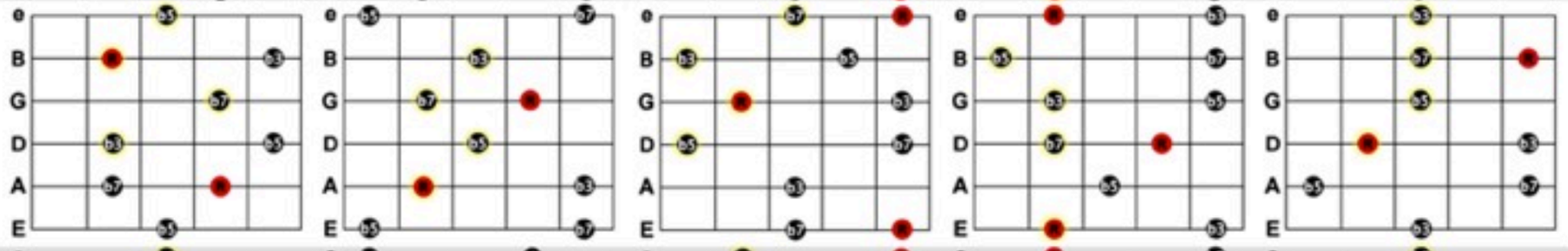
Minor 7th

R W h ♭3 W ♭5 W ♭7 h



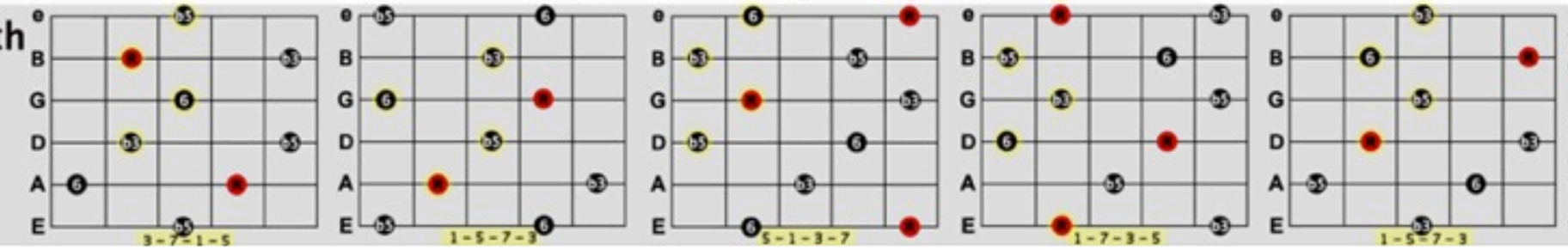
Minor 7th b5

R W h ♭3 W ♭5 h ♭7 W

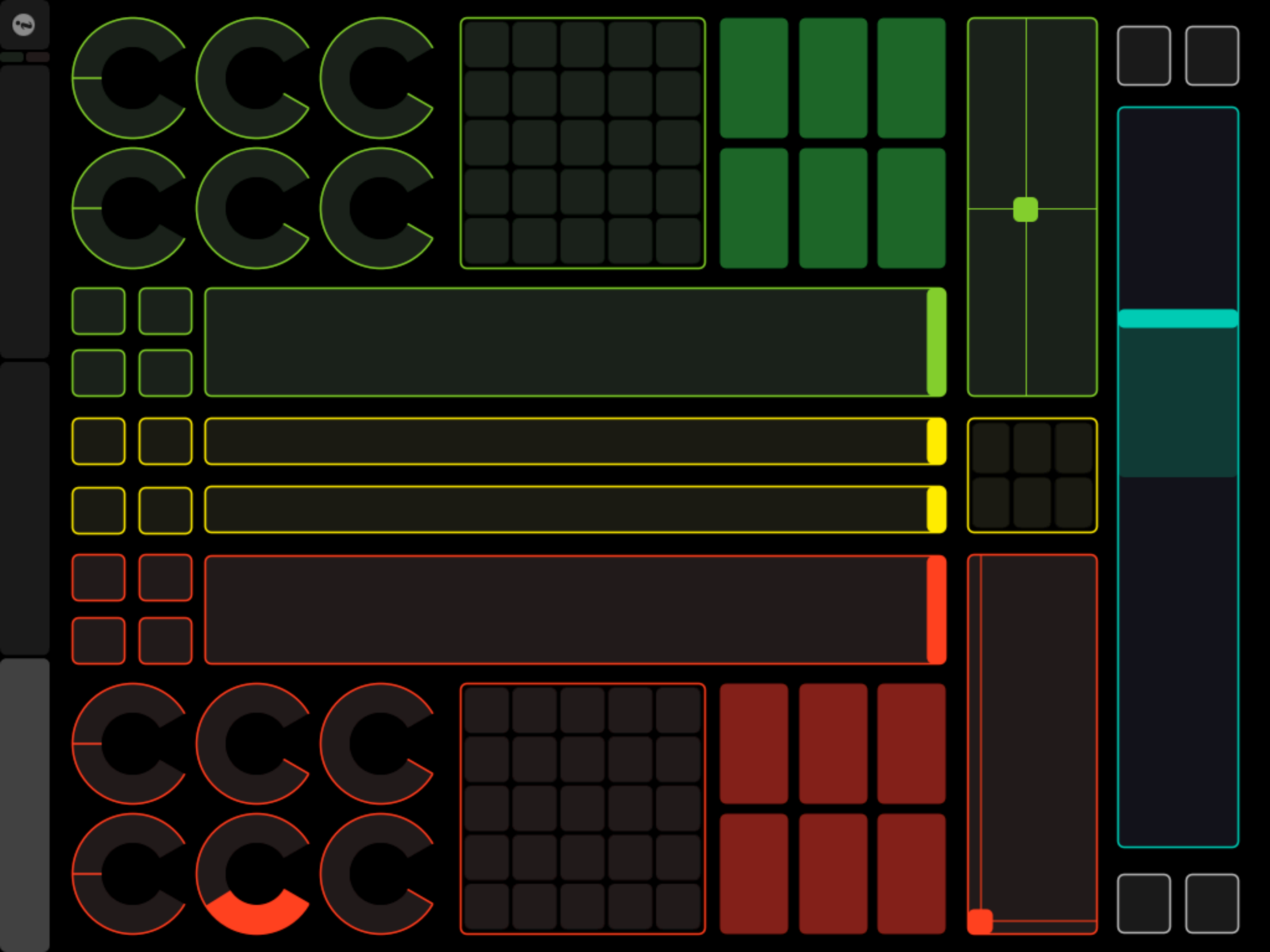


Diminished 7th

R W h ♭3 W ♭5 6 W h



Seventh Chords v1 by Dan Becker



scratch

bend

scroll stripe

fx 1 select

fx 2 select | d/w

TT fx set

gater on

brake

spinback

play / pause

Perform

Prepare

/32 /16 /8 /4 /2 1 2 4 8 16 32

Mash | << | >> | Loop | Snap | Loop | << | >> | Mash

Mash | LFO | Gater | Lo-Fi | Flanger | Flanger | Lo-Fi | Gate | LFO | Mash

Mash | Mash | Mash | Echo | Shift | Echo | Mash | Mash | Mash

Juggle | Sync | Key | Brake | Juggle | Sync | Key | Brake

Scratch | Filter | Filter | Scratch

Demo

Thank you

@samaaron

http://sam.aaron.name