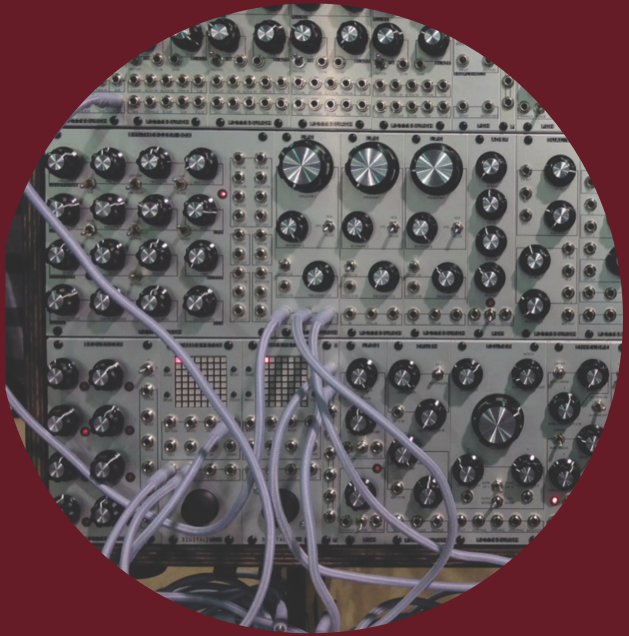
Software Architectures for Coding Music

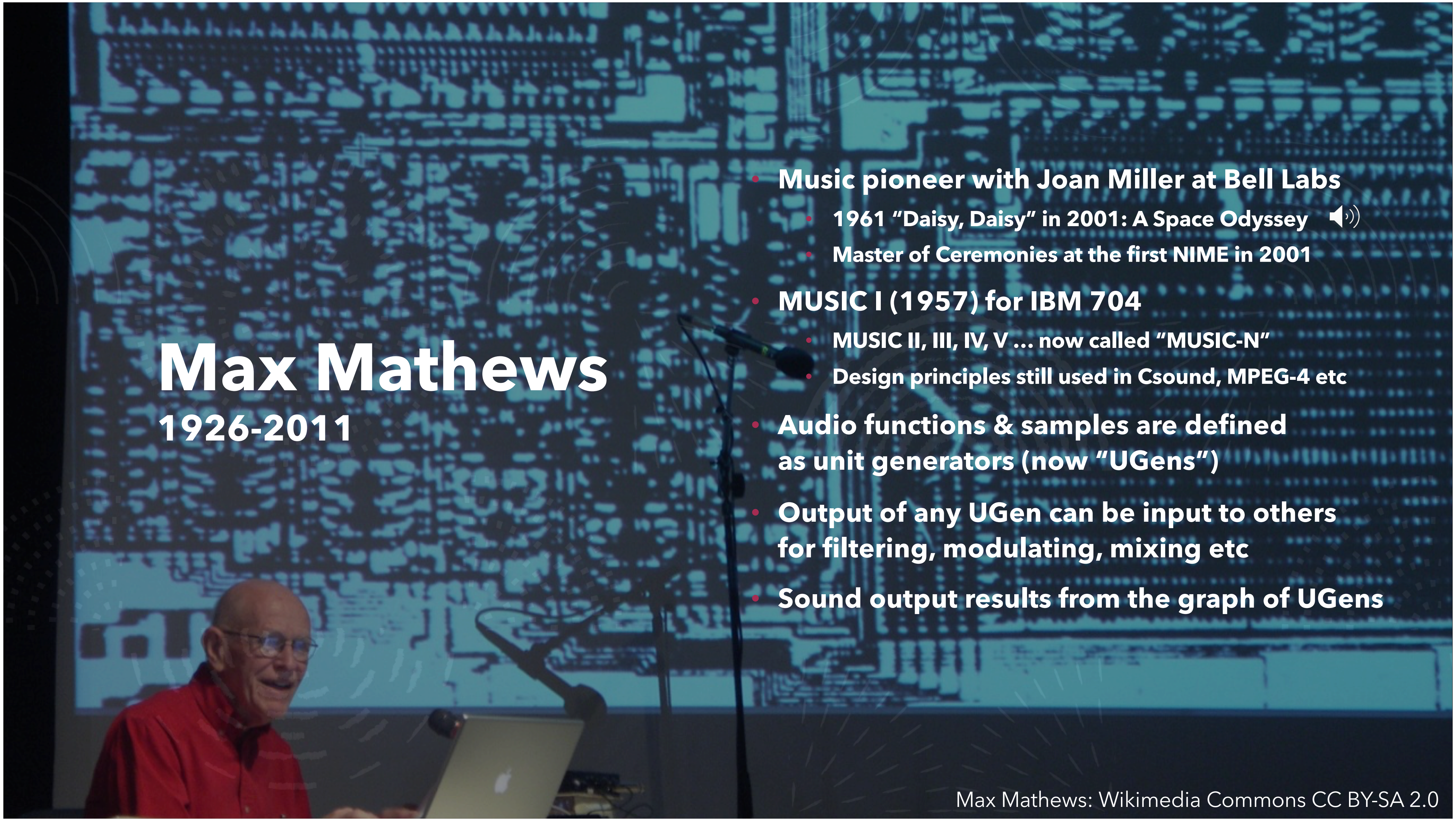
Alan Blackwell

Modular instrument architectures

1. Signal-based: waveforms, filters, modulators, mixers



2. Event-based: MIDI “piano roll” : pitch + velocity



Max Mathews

1926-2011

- Music pioneer with Joan Miller at Bell Labs
  - 1961 "Daisy, Daisy" in 2001: A Space Odyssey
  - Master of Ceremonies at the first NIME in 2001
- MUSIC I (1957) for IBM 704
  - MUSIC II, III, IV, V … now called "MUSIC-N"
  - Design principles still used in Csound, MPEG-4 etc
- Audio functions & samples are defined as unit generators (now "UGens")
- Output of any UGen can be input to others for filtering, modulating, mixing etc
- Sound output results from the graph of UGens

Max Mathews: Wikimedia Commons CC BY-SA 2.0

Architectures follow interface standards

- MIDI – Musical Instrument Digital Interface (1983)
  - Designed for point to point control, not networked
  - Basic abstraction is note on/off events (live or sequenced)
  - Instrument ID and some control signals
- OSC – Open Sound Control (2002)
  - Network address space (UDP/IP)
  - Time-tagged messages
  - Supports both numeric and symbolic data

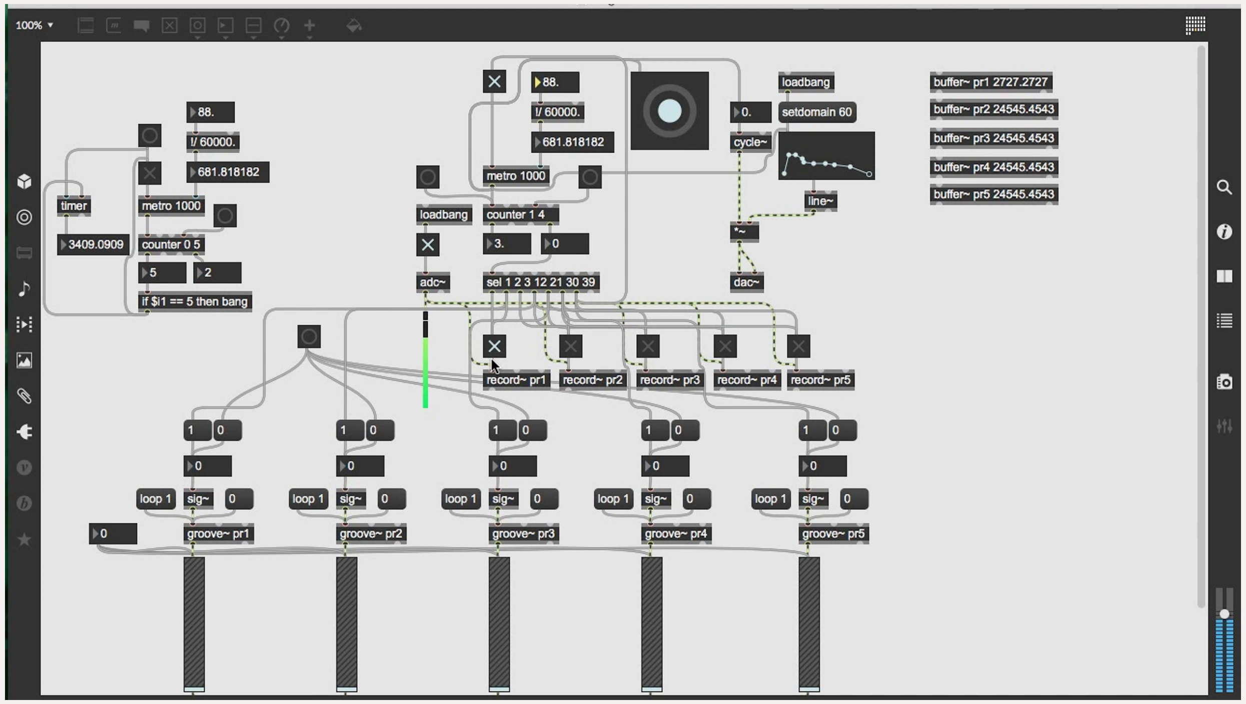
Miller S. Puckette

Max/MSP

(also Pure Data, Pd)

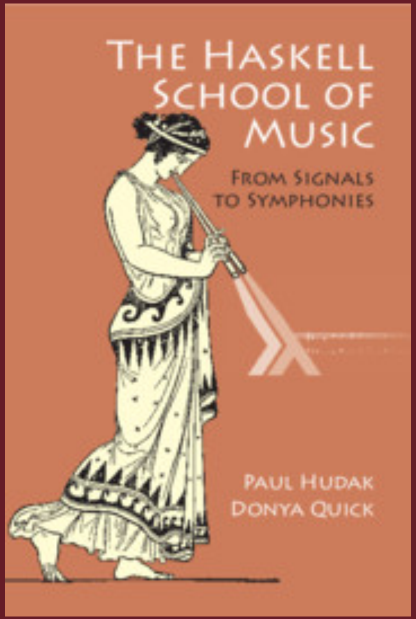
for Max Mathews

- Miller Puckette's work at IRCAM (1985)
- Originally MIDI "patches" only
- Commercialised by Cycling '74
- Open source version maintained as "Pd"



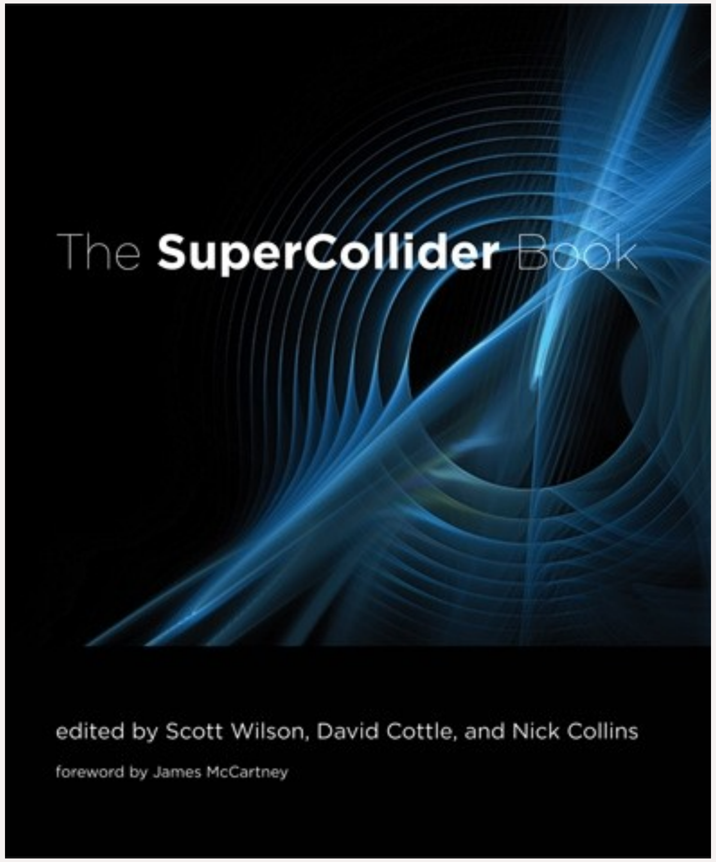
Functional Reactive Programming

- Defined by Paul Hudak (1952-2015)
  - dataflow / event-based paradigm
- FARM series
  - ACM SIGPLAN International Workshop on Functional Art, Music, Modeling and Design
- Haskell School of Music
  - Euterpea language dialect
  - Textbook available online from CUP



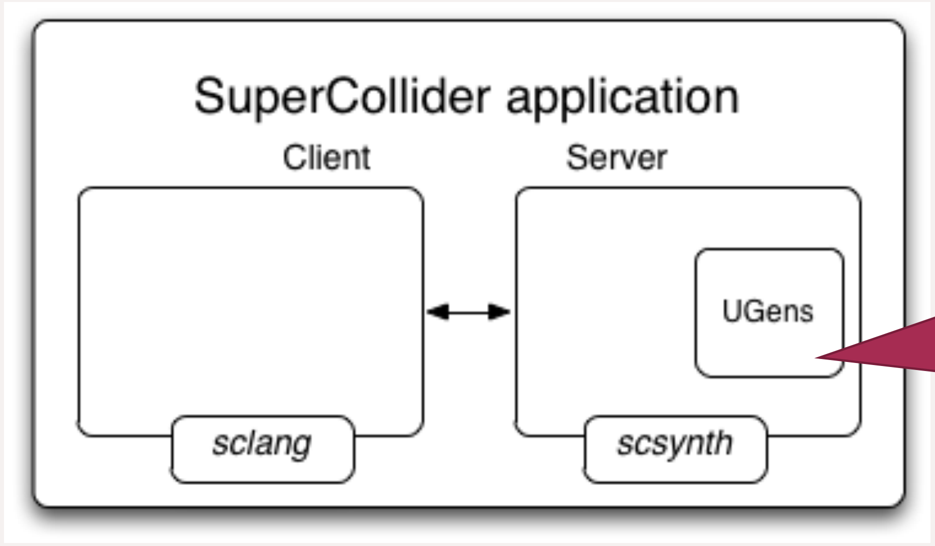
James McCartney's SuperCollider (1996 -)

- UGen-based language presented at ICMC in 1996
- Version 2 reimplemented as Smalltalk-like object-oriented language
  - UGens defined as objects
  - Released as open source in 2002
- Version 3 decoupled the architecture …

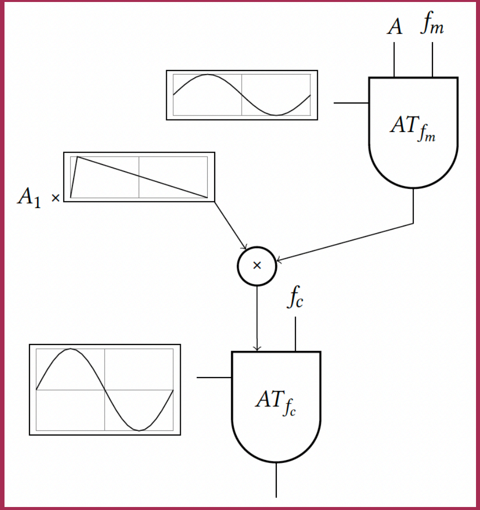

The SuperCollider Book
edited by Scott Wilson, David Cottle, and Nick Collins
foreword by James McCartney

SC architecture

- Network interface via OSC
- Client *defines* the synth graph
- scsynth UGens communicate (along graph edges) via internal control & audio buses



e.g. Christophe Rhodes' lecture:



Live Coding

- Dynamic sound modification: Ron Kuivila's demonstration of synthesis using FORTH at STEIM, Amsterdam 1985
- Code as performance art: SLUB (Alex McLean and Adrian Ward) using PERL at Public Life, London 2000
- Julian Rohrhuber's SuperCollider hot swap "trick" in 2003
- Liveness in modifying a process as it is executing
  - So coding becomes gesture, interpretation, improvisation

The TOPLAP manifesto

Temporary Organisation for the Promotion of Live Art Programming (2004)

- *We demand:* [note this is *still* a “draft” manifesto]
  - Give us access to the performer's mind, to the whole human instrument.
  - Obscurantism is dangerous. Show us your screens.
  - Programs are instruments that can change themselves
  - The program is to be transcended - Artificial language is the way.
  - Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome.
  - Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws.

(some) Live Coding languages

ChucK
from
Ge Wang

ixi lang
from Thor
Magnusson

Tidal Cycles
from Alex
McLean

Impromptu &
Extempore
from Andrew
Sorensen

Overtone
(+ EMACS Live)
from Sam Aaron

Sonic Pi
from
Sam Aaron

SuperCollider clients

Sam Aaron's Sonic Pi

- Developed in Cambridge Computer Lab, sponsored by Raspberry Pi foundation
- Goal to provide creative experiences with computing
- Focused on UK Computer Science curriculum, used in schools from outset
- Change from Clojure-based Overtone to Ruby DSP because JVM too slow on R-Pi
- Audio implementation as fixed scsynth graph with controllable samples, synths & effects

Open-source product with over 3 million users

Used in schools, arts commissions, community programmes

IDE with built-in language reference, tutorials and examples

Used by Sam as a live performance language

(nearly) funded by performance fees and Patreon supporters

The problem of time

- Rohrhuber and McLean are intensely concerned with execution time *vs* musical time *vs* creation time
- Sorensen's *temporal recursion* in Extempore is an elegant technical abstraction
- Standard musical questions push the bounds of "real-time"
    - e.g. Sam's redefinition of Ruby "sleep" to schedule future sc events via OSC time, not simply pausing code execution
    - Note that rhythm is driven by note onset, not (variable) note decay

The problem of richness

- Simple specifications are often boring to listen to
  - 4/4 rhythms, major scales, the "Amen" break
- So many live coded performances include stochastic noise generators, jitter in rhythm, random walks within a key …
- Random numbers offer stimulating creativity impetus …
  - … but also frustrating when something great can't be reproduced
- Sonic Pi hacks "random" to be a repeatable generative seed

Sonic Pi

created by

Sam Aaron

Code. Music. Live.

Love and thanks to all the
kind people who supported this release on Patreon:
https://patreon.com/samaaron

Version 3.3.1

Demo

See also Sam's keynote talk
Beating Threads - live coding with real time
https://youtu.be/YlRTTzlhquo

… and the Pop Pi video commissions
https://vimeo.com/user33572687