Aesthetics and Narrative:

Programming What Cannot Be Programmed

Clojure/conj 2016

🐦 @dschmudde

💻 http ://schmud.de

Functional Programming

Reason About Code

💵 Finance (Quantitative)

💾 Data (Quantitative)

Quantitative Reasoning

Soft: Cogitating

- Behavior
- Improvements

Hard: Mathematically Provable

- Correctness
- Performance

Functional Programming

Long History in Abstract Domains

🎨 Aesthetics

📖 Story

Art

Artist: Mostly Qualitative

🌝 Chaos and Surrender

🌞 Decisiveness and Agency

The Decisive Artist

Know the Rules Before You Can Break Them

The Decisive Programmer

Know the Rules Before You Can Write Them

We Don't Always Know the Rules

AARON

Development: 1968 - 21st Century

Fortran →

C →

Lisp (CLOS)

1983

Arnolfini
1983

1992

📙 📗 Color 📘 📕

Composing a Small Number of Primitives

Programming Autonomy

Aesthetic Choices

Harold Cohen

(1 May 1928 – 27 April 2016)

Paintings by AARON

(1968 - ?)

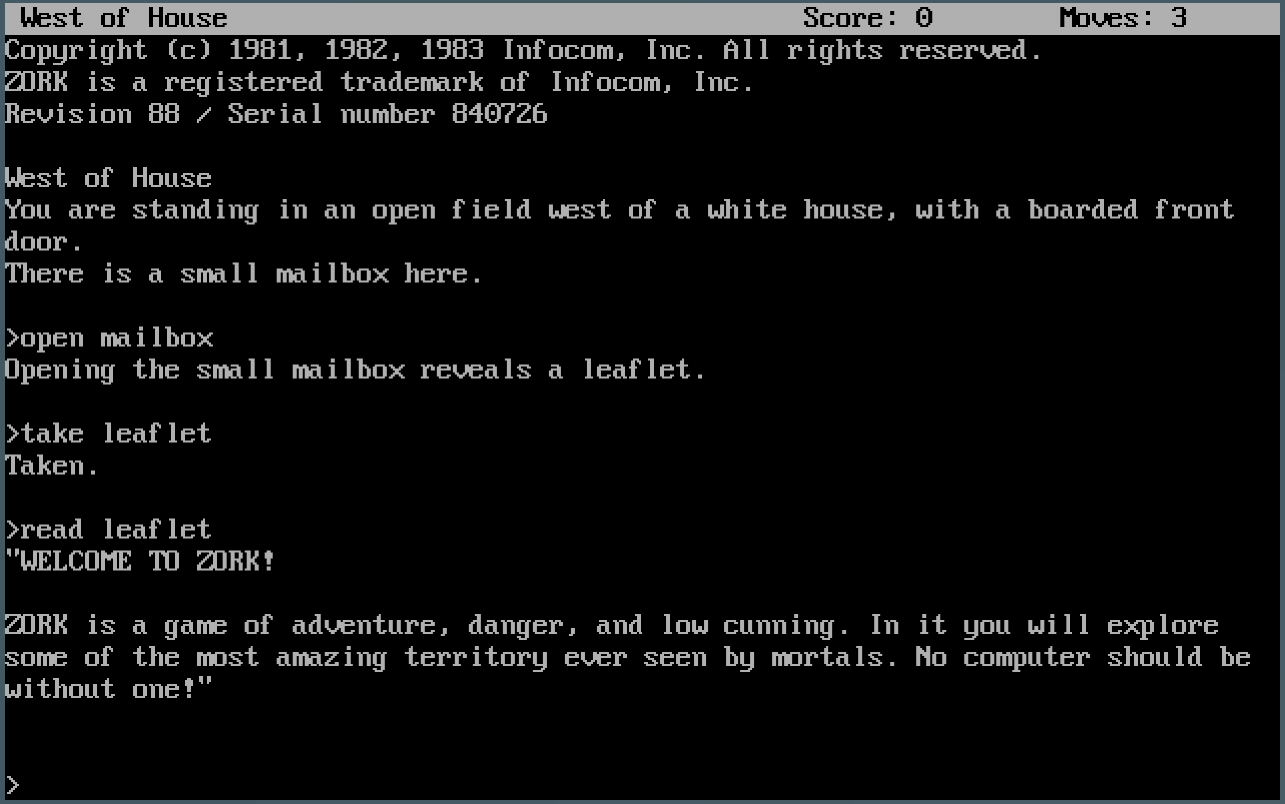
Zork

Development: 1975 - 1982

Fortran (*Adventure*) →

MDL (PDP-10) →

ZIL (Z-Machine running *Zork I-III*)



```
 West of House                                    Score: 0        Moves: 3
Copyright (c) 1981, 1982, 1983 Infocom, Inc. All rights reserved.
ZORK is a registered trademark of Infocom, Inc.
Revision 88 / Serial number 840726

West of House
You are standing in an open field west of a white house, with a boarded front
door.
There is a small mailbox here.

>open mailbox
Opening the small mailbox reveals a leaflet.

>take leaflet
Taken.

>read leaflet
"WELCOME TO ZORK!

ZORK is a game of adventure, danger, and low cunning. In it you will explore
some of the most amazing territory ever seen by mortals. No computer should be
without one!"


>
```

1975

Colossal Cave Adventure

Verb-Noun Commands

go west

1977

Zork

Prepositions and Conjunctions

Direct and Indirect Objects

fill the bottle with water

MDL

```
<DEFINE AXE-FUNCTION ()
  <COND (<VERB? "TAKE">
         <TELL "The troll's axe seems white-hot.
                You can't hold on to it.">
         T)>>
```

ZIL

```
<OBJECT LANTERN
    (LOC LIVING-ROOM)
    (SYNONYM LAMP LANTERN LIGHT)
    (ADJECTIVE BRASS)
    (DESC "brass lantern")
    (FLAGS TAKEBIT LIGHTBIT)
    (ACTION LANTERN-F)
    (FDESC "A battery-powered lantern is on the
            trophy case.")
    (LDESC "There is a brass lantern
            (battery-powered) here.")
    (SIZE 15)>
```

📃 Parsing and Language

Small Number of Primitives

Autonomy

Narrative Guide

Natural Language

Qualitative Reasoning

❓ Unknown Quantities

💡 Exploration of the *Idea*

Exploration

Agent Autonomy

$\longrightarrow$ **Intelligence?**

$\longrightarrow$ **Creativity?**

🎥 Making a Movie

Film Script:

A Technical Document

- Clarity and precision for interpretation:
    - Production breakdowns
    - Camera shooting scripts
    - Direction for actors and directors
    - Basis for novelizations

blueprint ≠ bridge

code ≠ execution

script ≠ film

Script

Pure Narrative

Film

Result of Autonomous Agents Making Aesthetic Choices

Creating Agency

Pure Functions

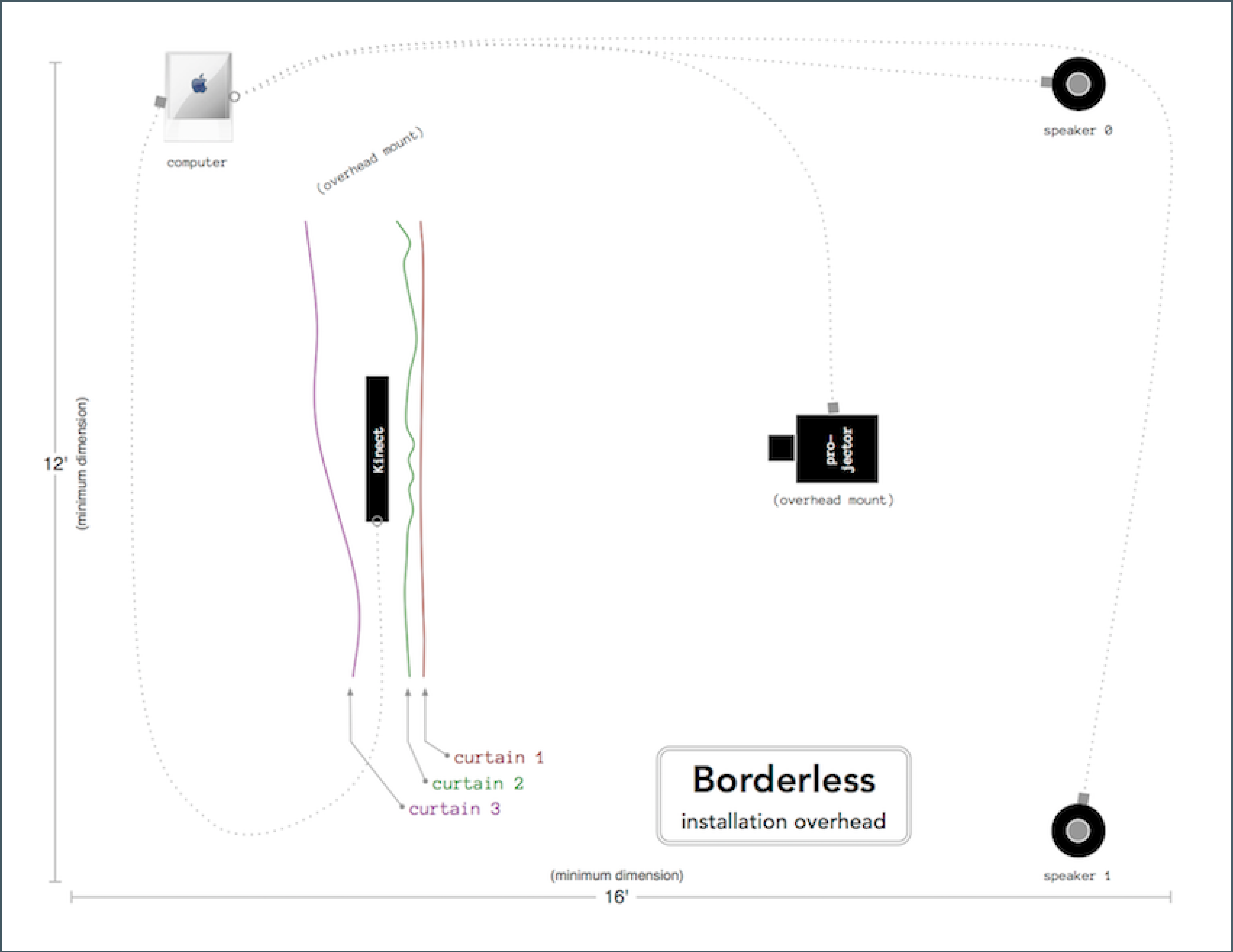
Composition

Borderless

👯 **Kinect**

🎶 **Overtone**

Abstract Domain



computer
speaker 0
(overhead mount)
Kinect
pro-jector
(overhead mount)
curtain 1
curtain 2
curtain 3
Borderless
installation overhead
speaker 1
12'
(minimum dimension)
(minimum dimension)
16'

Transducers

→ Input

- personEntered
- personUpdated
- personWillLeave

→ Input

- 0: pid;
- 1: oid;
- 2: age;
- 3: centroid.x;
- 4: centroid.y;
- 5: velocity.x;
- 6: velocity.y;
- 7: depth;
- 8: boundingRect.x;

→ Input

- 9: boundingRect.y;
- 10: boundingRect.width;
- 11: boundingRect.height;
- 12: highest.x
- 13: highest.y
- 14: haarRect.x; - will be 0 if hasHaar == false
- 15: haarRect.y; - will be 0 if hasHaar == false
- 16: haarRect.width; - will be 0 if hasHaar == false
  ...

Output →

- vca (loudness)
- reverb (timbre)
- vco (pitch)
- attack (timbre)
- sustain (timbre)
- release (timbre)
- gate (loudness)
- silence (loudness)

A/V Flow

→

Input: Kinect Vision

Δ

Internally: Clojure: Operate on Primitives

→

Output: Overtone Sound

Δ Operations

`osc/person-enter` ∈

`@person-sound{id-1 "vowel-1" id-2 "vowel-2" ...}`

`(osc/person-updated '(id age))`

∃ `@person-sound{id ...}`

× `(vowel (map #(* age %) [amp verb osc])`

Complexity

```
(definst drone-ae-sus
  "I make the 'ae' vowel sound at a given frequency.
   I start/stop with the gate set to 1 or 0."
  [freq   100
   gate   (synth-defaults ::gate)
   amp    (synth-defaults ::vca)
   verb   (synth-defaults ::reverb)
   kr-mul (synth-defaults ::vco)]

  (let [kr-mul     (:value kr-mul)
        eq-freq    [270 2290 3010]
        hpf-rlpf   [600 8000 0.6]
        q          0.1
        synth-unit (synth-unit-layered freq
                    eq-freq q kr-mul)]

    (synth-filter-chain synth-unit amp verb gate hpf-rlpf)
```

Art

Qualitative Reasoning

Know The Rules Before You Can Break Them

Constraints Breed Creativity

Trust Your Instinct

(REPL is God)

spec

Scientific Constraints

```
(s/def ::frequency (s/and number?
                          #((control-range 20 20000) %) ))
```

spec

Aesthetic Constraints

```
(s/def ::vca (s/and number?
                    #((control-range 0.4 1) %)))

(s/def ::reverb (s/and integer?
                       #((control-range 0 1000) %)))

(s/def ::vco (s/and integer?
                    #((control-range 0 25) %)))
```

Autonomous Decision Making

~~testable~~ experiment-able with `spec`

Autonomous Decision Making

Designing its own timbre over long periods of uptime

Autonomy

&

Authorship

📊 ~~Objectivity~~ 📉

🎨 Subjectivity 📖

Art

Aesthetics and Narrative:

Programming What ~~Cannot~~ Can Be Programmed

@dschmudde

http ://schmud.de