
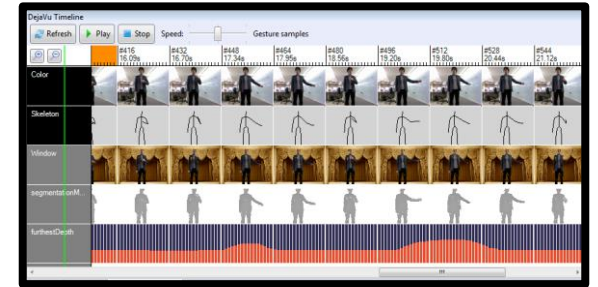
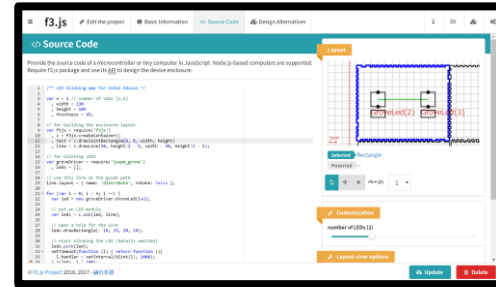
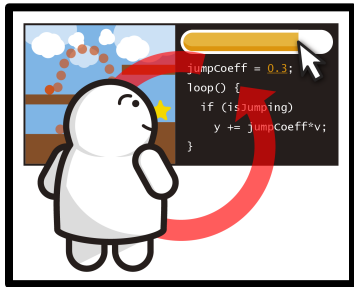


# User Interfaces for Live Programming

Jun Kato

<https://junkato.jp>

Researcher,  **AIST**



LIVE 2017 Keynote, 10/24/2017



## Computer Science (Human-Computer Interaction, Programming Language)

Phybots



ACM DIS'12

DejaVu



ACM UIST'12

Picode



ACM CHI'13

It's Alive!



ACM PLDI'13

VisionSketch



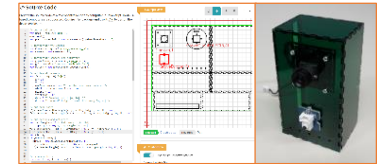
GI'14

TextAlive



ACM CHI'15

f3.js



ACM DIS'17

- Created **Tools and Environments** for **Creativity/Productivity Support**
- **Application Domains:** Prototyping, Physical Computing, Computer Vision, Robots, Internet of Things, Animation Authoring, ...
- Founded **SIGPX** (SIG on Programming Experience) <https://sigpx.org/en>

# $P(x)$ SIGPX

<https://sigpx.org/en>

A group of **researchers/engineers/teachers** in **Japan**, studying ...

**Programming Experience** in the intersection of **HCI/PL/SE**

**Meet & Discuss**

1<sup>st</sup> meetup, 2/27/2016

**Collect**

$P(x)$

<https://scrapbox.io/ProgrammingExperience/>

**Publish**

**PX Special Issue in IPSJ Journal (Nov 2017)**  
**Emerging Research on Programming Experience:**  
**From Language Design to Industrial Applications**

User Interfaces for Live Programming

# Today, I'm going to talk about ...

- What is Live Programming?
- UIs for Live Programming with end-users
- UIs for Live Programming of this material world
- UIs for Live Programming with time travel
- Live Programming as User Interface research

---

## **It is about ...**

- Showcase of user interfaces for programming
- Not only my work but also others' notable work
- Discussion on live programming system design

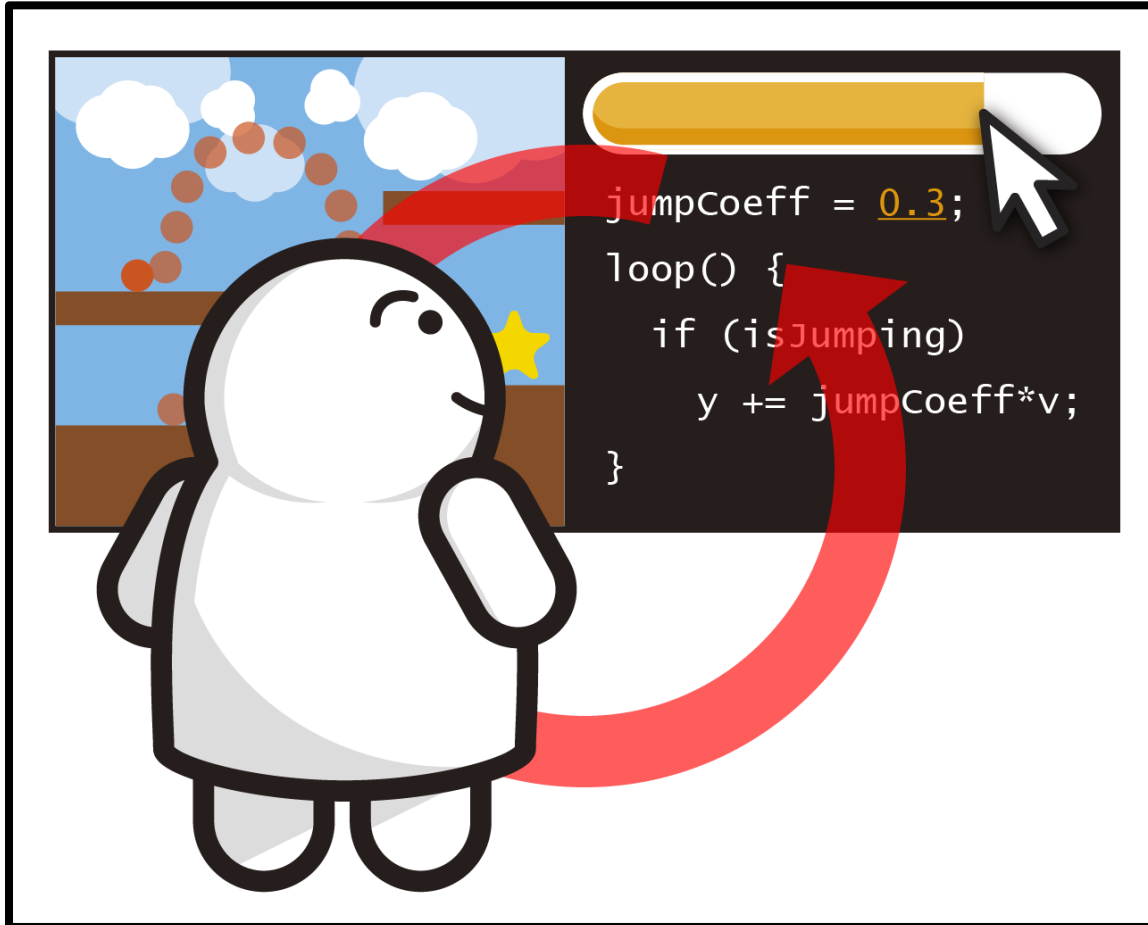
## **It is not about ...**

- No  $\lambda$  or greek symbols in slides
- Not a consensus in the field (it's ongoing!)
- No peer review involved (my personal view)

# Today, I'm going to talk about ...

- **What is Live Programming?**
  - UIs for Live Programming with end-users
  - UIs for Live Programming of this material world
  - UIs for Live Programming with time travel
  - Live Programming as User Interface research
-

# What is Live Programming?



- Programming experience
- Continuous feedback
- Concrete information
- Early examples in VPL and OOP
- Attracting much attention these days



Not new but hot!



# Text-based Programming

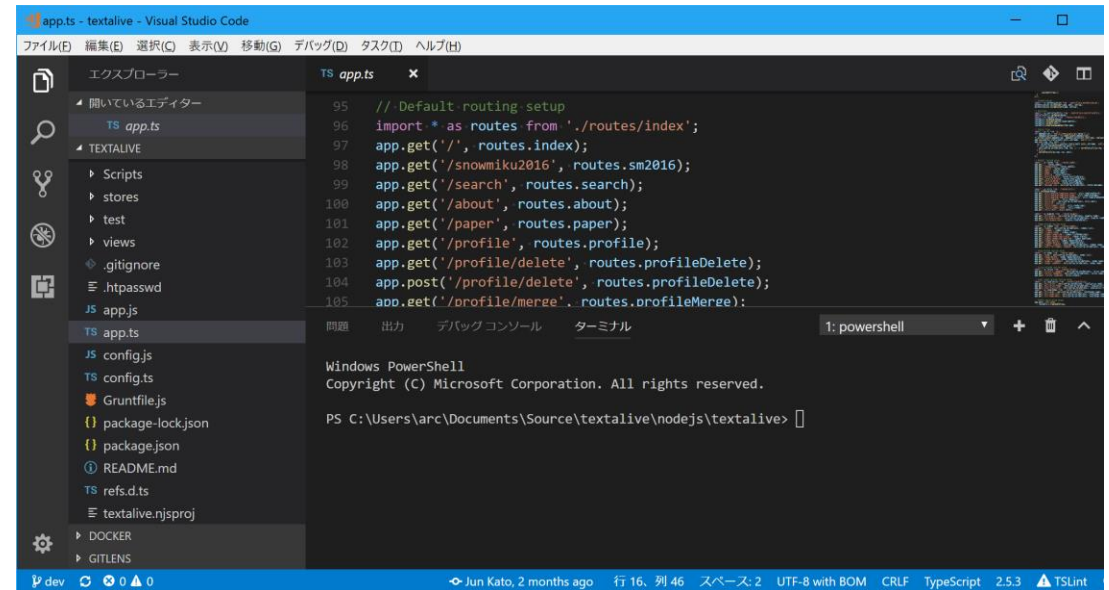
## Dartmouth BASIC [1964]

```
(C) Copyright Microsoft 1983,1984,1985,1986,1987
60300 Bytes free
Ok
10 READ A1, A2, A3, A4
15 LET D = A1 * A4 - A3 * A2
20 IF D = 0 THEN 65
30 READ B1, B2
37 LET X1 = (B1*A4 - B2 * A2) / D
42 LET X2 = (A1 * B2 - A3 * B1)/D
55 PRINT X1, X2
60 GO TO 30
65 PRINT "NO UNIQUE SOLUTION"
70 DATA 1, 2, 4
80 DATA 2, -7, 5
85 DATA 1, 3, 4, -7
90 END

RUN
4          -5.5
.6666667   .1666667
-3.666667  3.833333
Out of DATA in 30
Ok
T|LIST 2|RUN+ 3|LOAD+ 4|SAVE+ 5|CONT+ 6|"LPT1 7|IRON+ 8|TRUFF+ 9|KEY 0|SCREEN
```

- Text-based editor
- Text-based debugger
- Text-based ...

## Visual Studio Code [as of today]



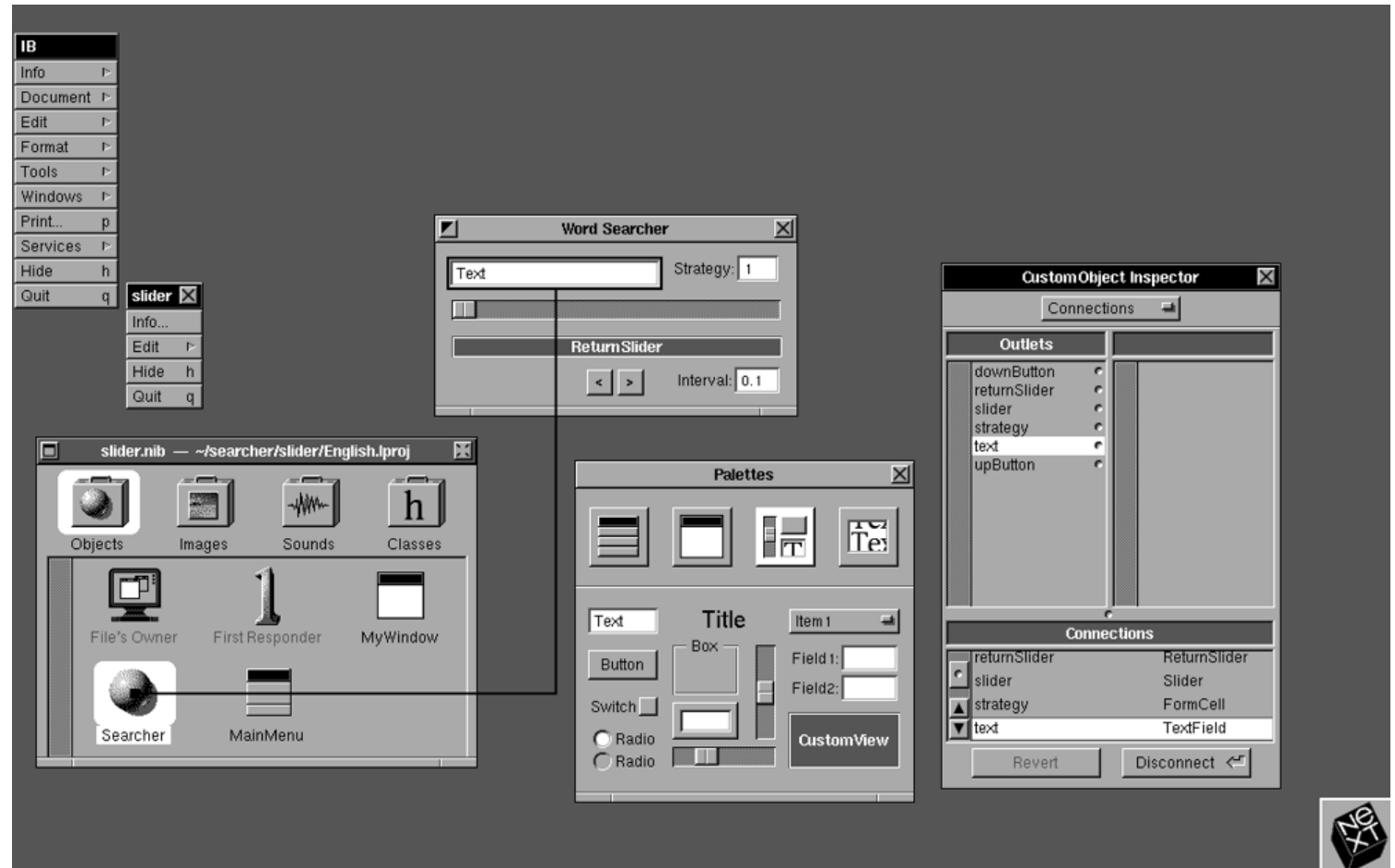
IDEs haven't changed much

# With some exceptions ...

[since 1986]

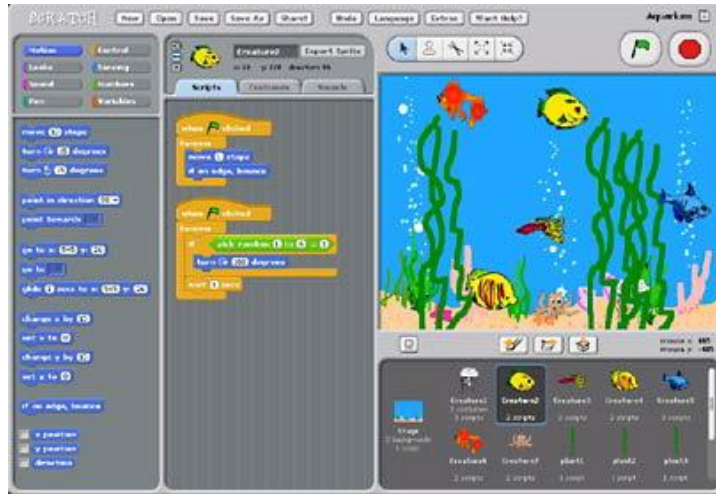
## Interface Builder

- A tool for NeXT UI development
- Later integrated into Xcode
- Many IDEs have similar built-in tools



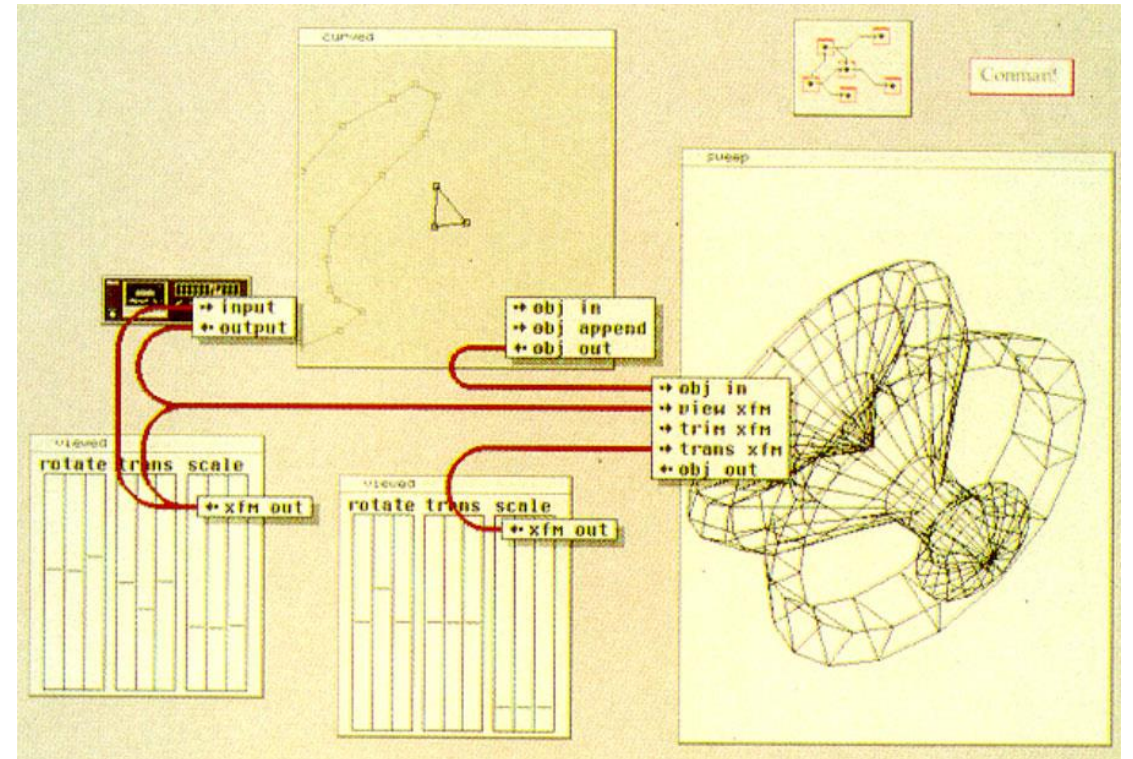


# Visual Programming



**Scratch**  
MIT

**ConMan**  
Haberli  
[SIGGRAPH 1988]



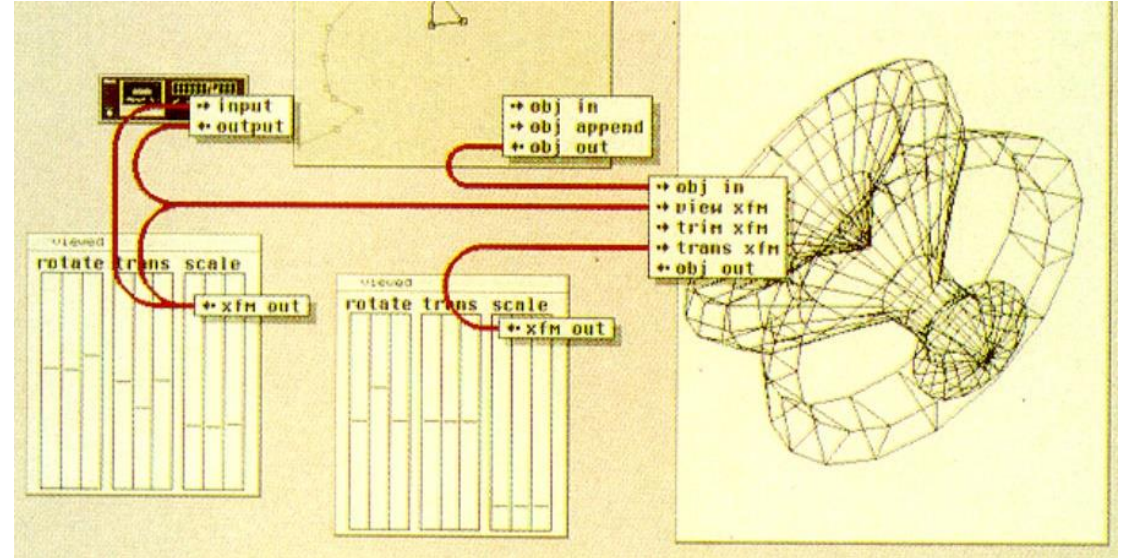
- Mostly dealing with symbolic representations of programs
- Often considered as tools for novices and good for education
- Dataflow languages: early examples of live programming

# Character-based UIs **or** Graphical UIs?

```
(C) Copyright Microsoft 1983,1984,1985,1986,1987
60300 Bytes free
Ok
10 READ A1, A2, A3, A4
15 LET D = A1 * A4 - A3 * A2
20 IF D = 0 THEN 65
30 READ B1, B2
37 LET X1 = (B1*A4 - B2 * A2) / D
42 LET X2 = (A1 * B2 - A3 * B1)/D
55 PRINT X1, X2
60 GO TO 30
65 PRINT "NO UNIQUE SOLUTION"
70 DATA 1, 2, 4
80 DATA 2, -7, 5
85 DATA 1, 3, 4, -7
90 END

RUN
4          -5.5
.6666667   .1666667
-3.666667  3.833333
Out of DATA in 30
Ok
LIST 2RUN 3LOAD 4SAVE 5CONT 6, "LP11 7IRON 8TROFF
```

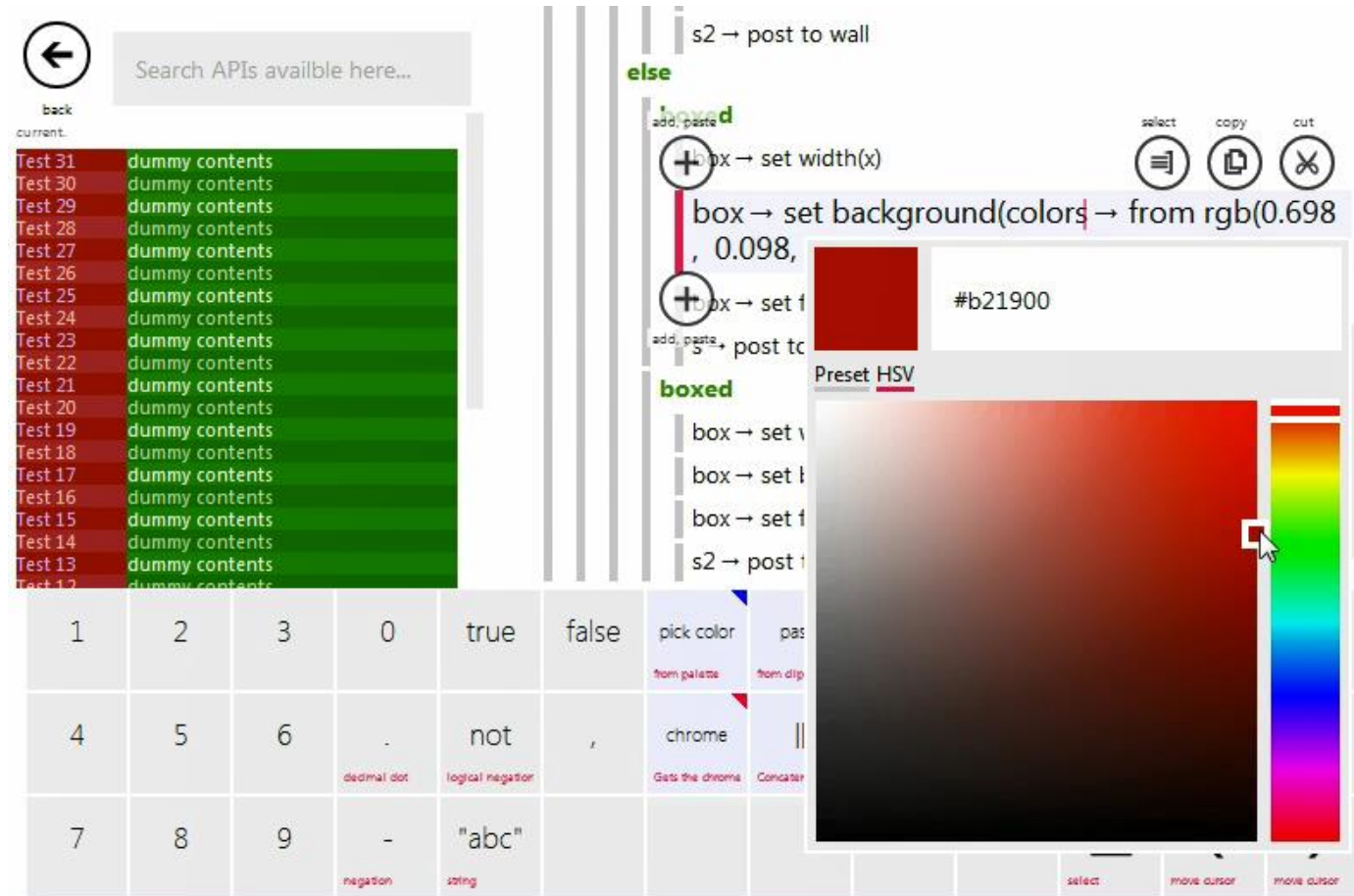
or



Modern

Live Programming as a hybrid approach

# User Interfaces with Text **and** Graphics



**TouchDevelop**

[PLDI 2014]

<https://touchdevelop.com>



# User Interfaces with Text **and** Graphics

≡ f3.js IoTコンテンツの編集 ① 基本情報 <> ソースコード 組み立て

```
21 var f3js = require('f3js')
22   , x = 10
23   , y = 10
24   , useCountdown = true /* Use the countdown feature. */
25   , width = useCountdown ? 130 : 60
26   , height = 105
27   , thickness = 36 /* Thickness [10, 100] */
28   , dw = 5 /* Joint width [0, 10] */
29   , dh = 2 /* Joint height (panel thickness) [0, 10] */;
30
31 // put base board
32 var rect = f3js.drawJointRectangle(x, y, width, height, dw);
33 var planes = rect.extrude(thickness);
34 planes[4].x = width; planes[4].y = 0; f3js.add(planes[4]);
35
36 // put sensors and actuators
37 var leftMargin = 28 // Left margin [0, 100]
38   , topMargin = 30; // Top margin [0, 100]
39 f3js.add(camera, { x: x + leftMargin + 0, y: y + topMargin
40 f3js.add(button, { x: x + leftMargin + 0, y: y + 75 });
41
42 var circle;
43 if (useCountdown) {
44   circle = new gcl.GroveCircularLED(5, 4);
45   f3js.add(circle, { x: x + width - 30, y: y + topMargin
46   f3js.drawRectangle(x + width - 30, y + topMargin
```

Camera(0)

GroveCircularLED(5,4)

GroveButton(3)

100

カスタマイズ

☒ Use the countdown feature.

Thickness (36)

Joint width (5)

Joint height (panel thickness) (2)

Left margin (28)

更新する

削除する

© f3.js Project 2016 - English

f3.js

[DIS 2017]

<http://f3js.org>

# Character-based UIs **and** Graphical UIs

- It's like text **and** figures in research papers
- **Text** is good at abstraction
- **Graphics** are good at presenting concrete information

## Integrated Graphical Representations [2014, dissertation] [2016]



They complement each other

## Picode: Inline Photos Representing Posture Data in Source Code

Jun Kato

The University of Tokyo, Tokyo, Japan – {jun.kato | d.sakamoto | takeo}@acm.org

Daisuke Sakamoto

Takeo Igarashi

### ABSTRACT

Current programming environments use textual or symbolic representations. While these representations are appropriate for describing logical processes, they are not appropriate for representing raw values such as human and robot posture data, which are necessary for handling gesture input and controlling robots. To address this issue, we propose *Picode*, a text-based development environment augmented with inline visual representations: photos of human and robots. With *Picode*, the user first takes a photo to bind it to posture data. She then drag-and-drops the photo into the code editor, where it is displayed as an inline image. A preliminary user study revealed positive effects of taking photos on the programming experience.

### Author Keywords

Development Environment; Inline Photo; Posture Data.

### ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces – GUI; D.2.6. Software Engineering: Programming Environments – Integrated environments.

### INTRODUCTION

A programming language is an interface for the programmer to input procedures into a computer. As with other user interfaces, there have been many attempts to improve its usability. Such attempts include visual programming languages to visualize the control flow of the program, structured editors to prevent syntax errors, and enhancement to code completion that visualizes possible inputs [8]. However, programming languages usually consist of textual or symbolic representations. While these representations are appropriate for precisely describing logical processes, they are not appropriate for representing the posture of a human or a robot. In such a case, the programmer has to list raw numeric values or to maintain a reference to the datasets stored in a file or a database.

To address this issue, Ko and Myers presented a framework called “Barista” for implementing code editors which are capable of showing text and visual representations [5]. This framework enhances comments for an image processing

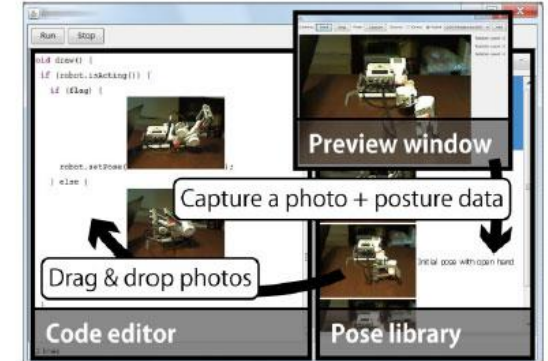


Figure 1. Overview of *Picode*

method by including an image that shows a concrete example of what the method does. Yeh et al. presented a development environment named “Sikuli,” with which the programmer can take a screenshot of a GUI element and paste the image into a text editor [12]. In Sikuli, the image serves as an argument of the API functions. Our goal was to apply a similar idea to facilitate the programming of applications that handle human and robot postures.

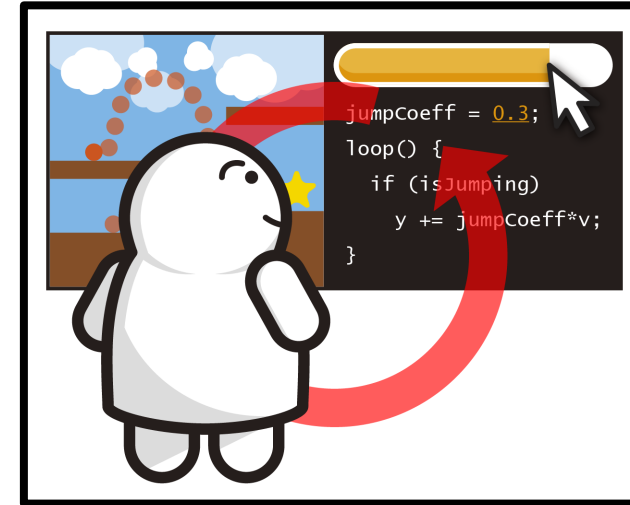
We propose a development environment named *Picode* that uses photographs of human and robots to represent their posture data in a text editor (Figure 1). It helps the development process of applications for handling gesture input and/or controlling robots. The programmer is first asked to take a photo of a human or a robot to bind it to the posture data. She then drag-and-drops the photo into the code editor, where it is shown as an inline image. Our environment provides a built-in API which methods take photos as arguments. It allows the user to easily understand when the photo was taken and what the code is meant to do.

### RELATED WORK

After the Microsoft Kinect and its Software Development Kit (SDK) hit the market, many interactive applications have been developed that handle human posture. At the same time, some toolkits and libraries have been proposed that support the development of such applications. They can typically recognize preset poses and gestures. When the programmer wants to recognize her own poses and gestures, however, she has to record the examples outside the development environment. On the other hand, our development environment is designed to support the entire prototyping process of application development. It fully

# In Live Programming systems, we ...

- first have **vague ideas**
- then explore the ideas with **concrete examples**
- gradually start turning the ideas into **programs**

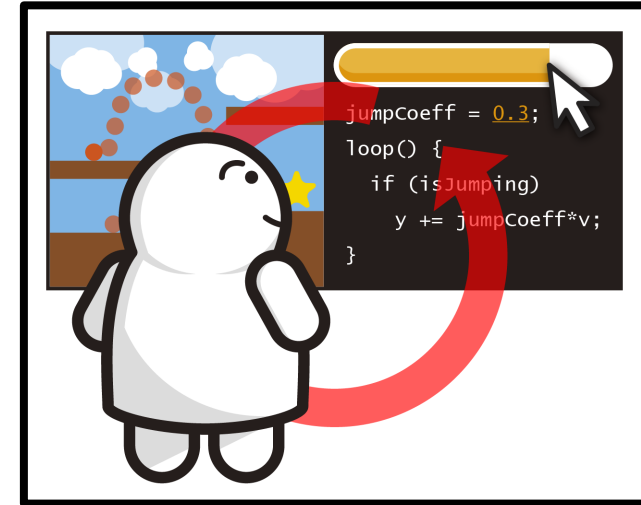


Live Programming requires  
decent UIs for exploring the problem space



# UIs for Live Programming should ...

- avoid sudden changes in the program behavior
- keep the program and its output relevant
- allow continuously exploring the problem space



Appropriate user interface design differs  
from application to application

When designing live programming systems ...

# Don't be afraid to be domain-specific

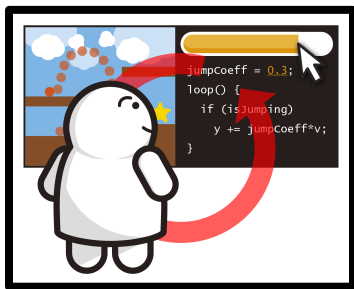
- Good UI is always specifically designed for the target domain
- It might be like replaying the history of end-user computing in the domain of programming
- We might need PX workbench (cf. language workbench)

**Cf. Programming eXperience Toolkit (PXT)**

<https://github.com/microsoft/pxt>

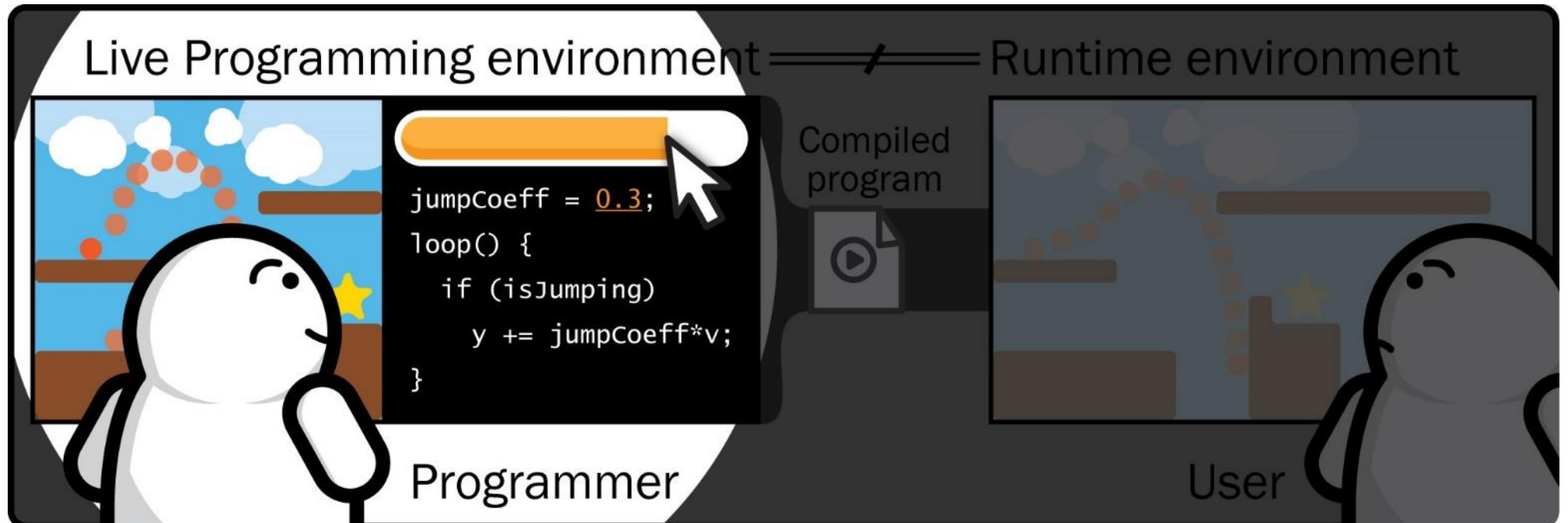
# Today, I'm going to talk about ...

- What is Live Programming?
- **UIs for Live Programming with end-users**
- UIs for Live Programming of this material world
- UIs for Live Programming with time travel
- Live Programming as User Interface research



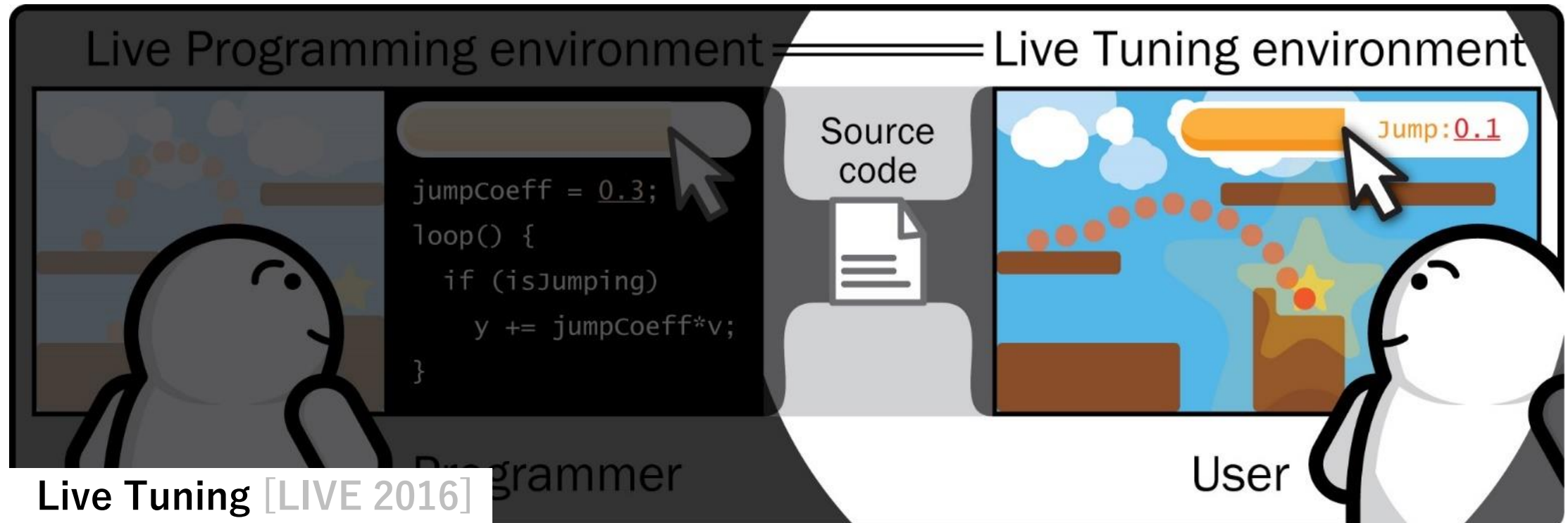
# UIs for Live Programming

Good mixture of text-based and graphical user interfaces

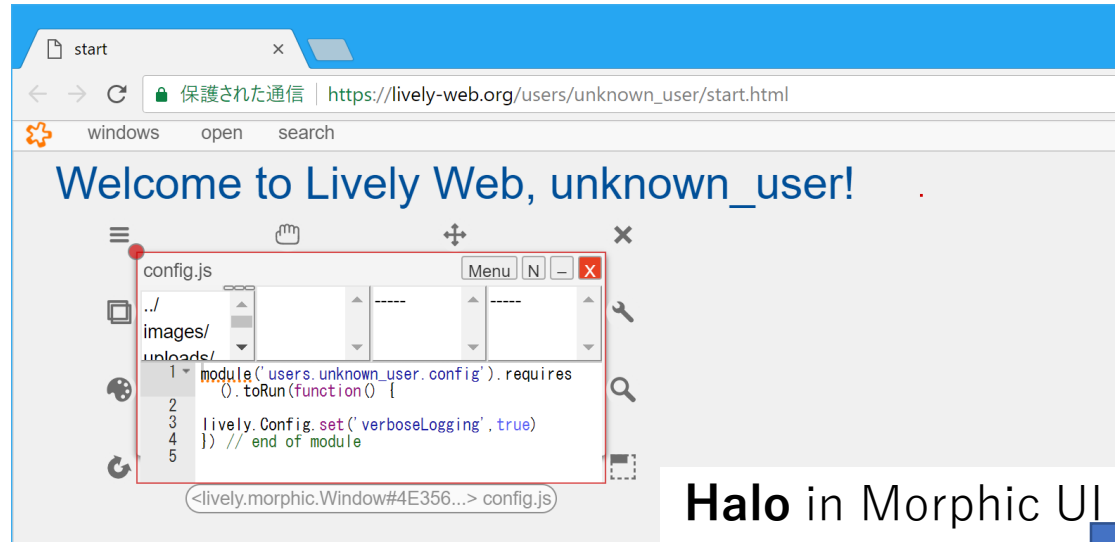


# UIs for Live Programming

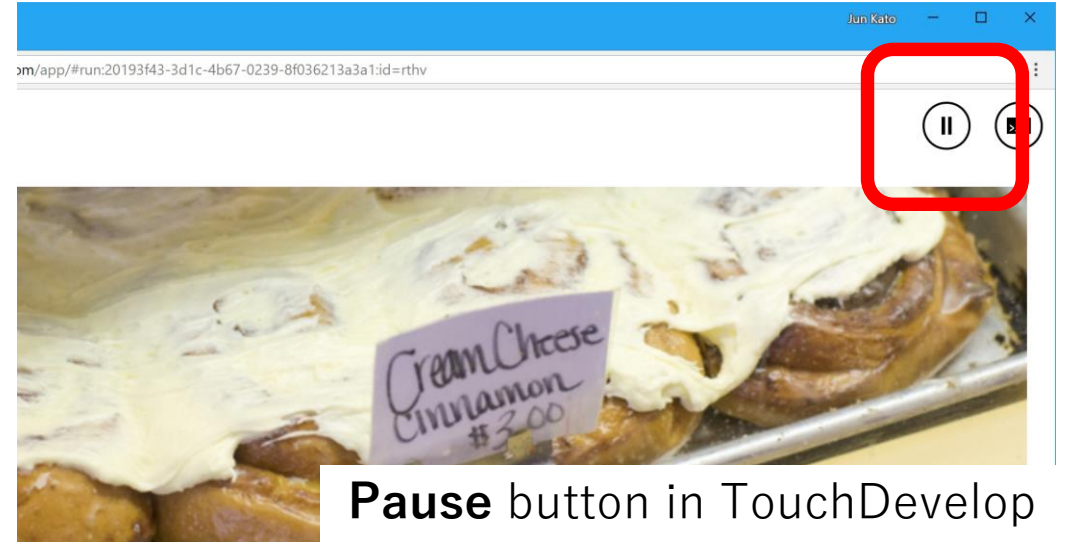
Why not expose GUI to users so that they can edit programs?



# Mode Switch between “Use” and “Build”



**Halo** in Morphic UI



**Pause** button in TouchDevelop

What if we add another layer for **users**?

**Promoting universal usability with multi-layer interface design**

Ben Shneiderman [2002]



TextAlive

Songs

Videos

Templates

Edit

Login

Full screen

Help

# Ukulele Pen

**TextAlive**[CHI 2015] <http://textalive.jp>

TextAlive



Save



Options

arc@dmz

Full screen

Help

## Parameter tweaking (横書き 配置)

文字色



単語padding



10

文字padding



38

Fadein Time



0

Fadeout Time



0

Propose

Programming

TextAlive

[CHI 2015]

<http://textalive.jp>

Phrase

Word

Character

Vol. 0.00s

User Interfaces for Live Programming

30.33s

32.00s

68.65s

TextAlive



Save



Options

arc@dmz

Full screen



Help



Apple Pen

```
37 (new / preTime this.fadeoutTime / {
38   var fadeout_progress = (p.endTime - now) / this.fadeoutTime;
39   p.rendering.alpha = fadeout_progress;
40 }
41
42 var hrange = p.advance
43   + (p.wordCount - 1) * this.wpadding
44   + (p.charCount - 1) * this.cpadding;
45 var wx = (width - hrange) / 2;
46 var cx = 0;
47 var w = p.firstWord;
48 for (var i = 0; i < p.wordCount; i++) {
49   w.rendering.tx.translate(wx, (height - w.height) / 2);
50   var c = w.firstChar;
51
52   cx = 0;
53   for (var n = 0; n < w.charCount; n++) {
54     c.color = this.textColor;
55     c.rendering.tx.translate(cx, 0);
56     cx += c.advance + this.cpadding;
57     c = c.next;
58   }
59   wx += cx + this.wpadding;
60   w = w.next;
61 }
62 };
63 }
```



Phrase

Word

Character

TextAlive

[CHI 2015]

<http://textalive.jp>

Vol. 0.00s

User Interfaces for Live Programming

30.33s

32.00s

68.65s

# Co-hosting UIs for programmers and users

## Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

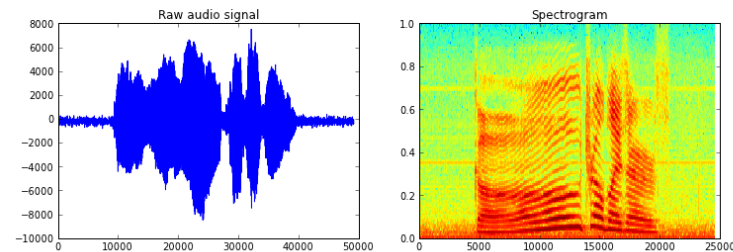
using windowing, to reveal the frequency content of a sound signal.

We begin by loading a datafile using SciPy's audio file support:

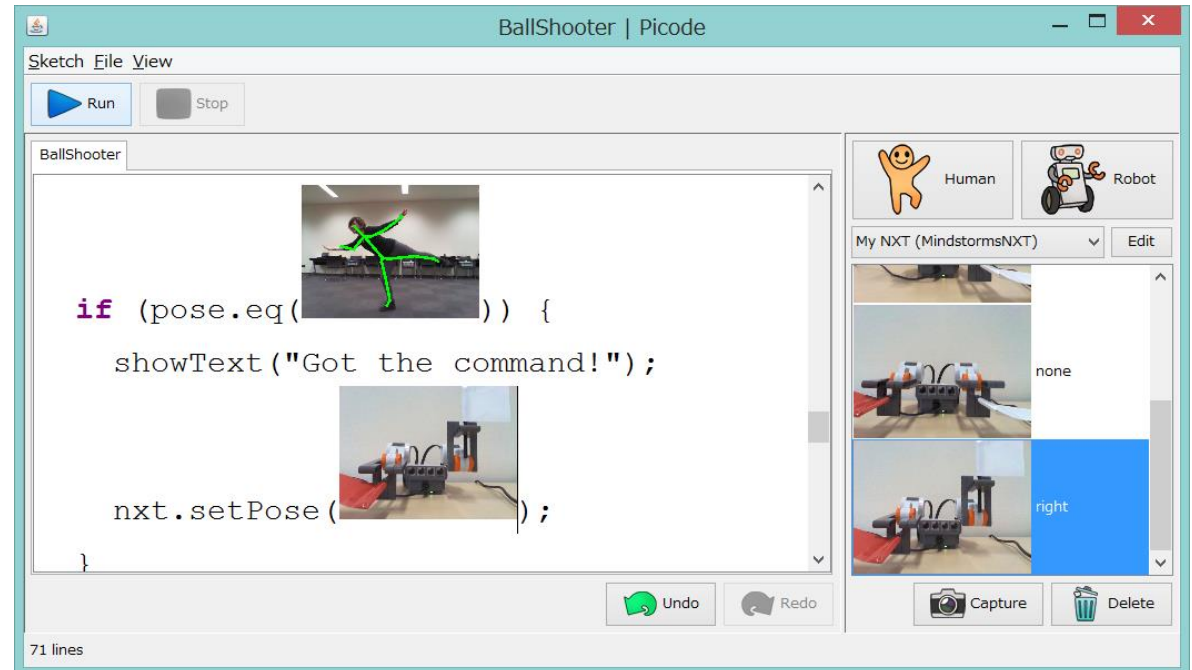
```
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin spectrogram routine:

```
In [2]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.spectrogram(x); ax2.set_title('Spectrogram');
```



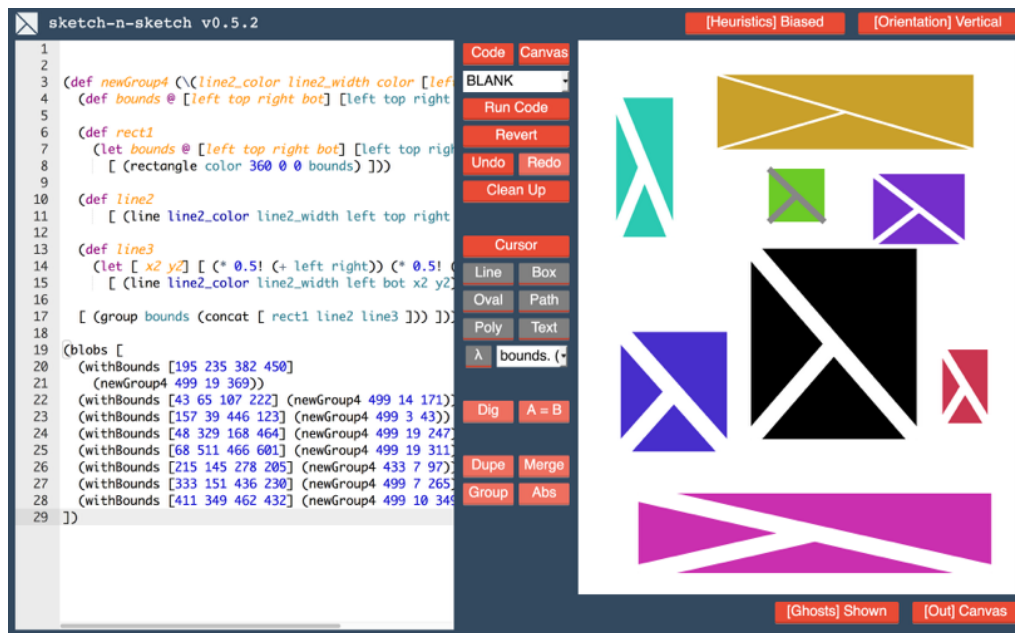
**Literate Programming**  
in Jupyter (Ipython) Notebook



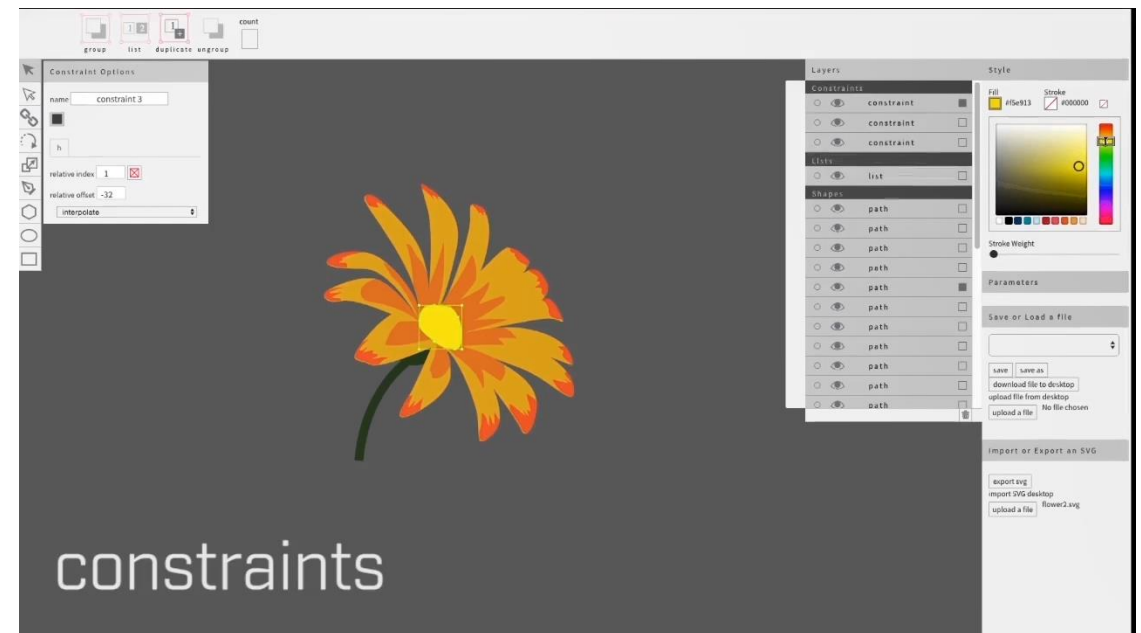
**Inline Photos**  
in Picode [CHI 2013]



# Merging UIs for programmers and users (direct manipulation on program output)



**Sketch-n-Sketch, Hempel et al.**  
[UIST 2016 etc.]

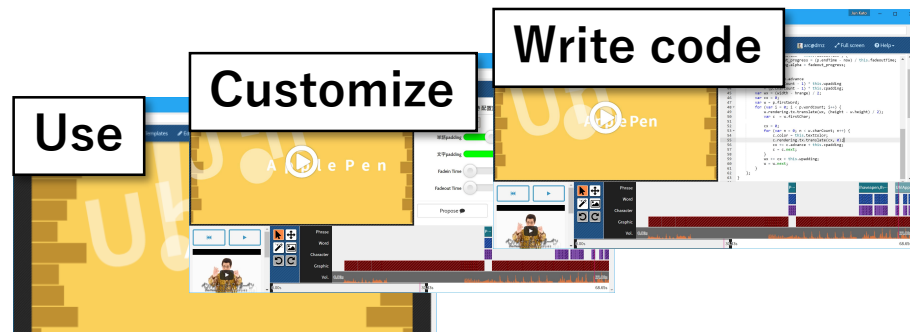


**Para, Jacobs et al.**  
[CHI 2017]

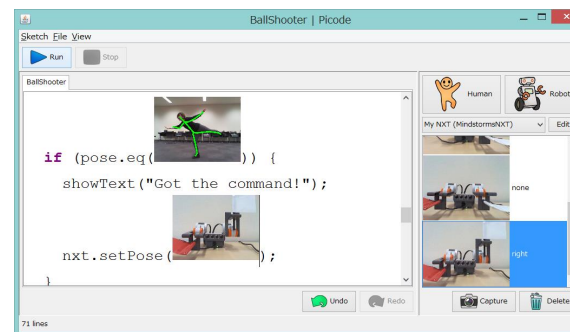
When designing live programming systems ...

# How about making the ladder of expertise?

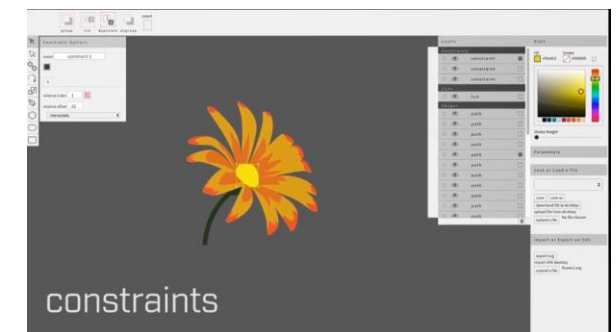
- From live programming for programmers
- To live programming for the community of people



**Multi-layer**



**Co-host**



**Merge**



# Today, I'm going to talk about ...

- What is Live Programming?
- UIs for Live Programming with end-users
- **UIs for Live Programming of this material world**
- UIs for Live Programming with time travel
- Live Programming as User Interface research



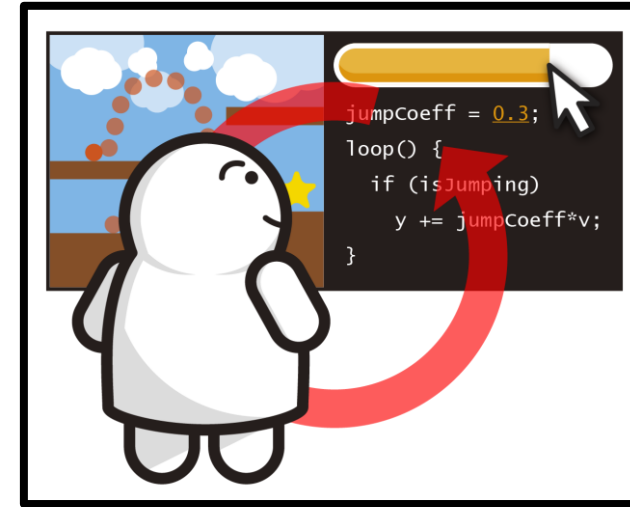
# What is “live” and what not?

- System response time:
  - Computation
  - Network
  - Touch display response

## How Live are Live Programming Systems?

Rein et al. [PX 2016]

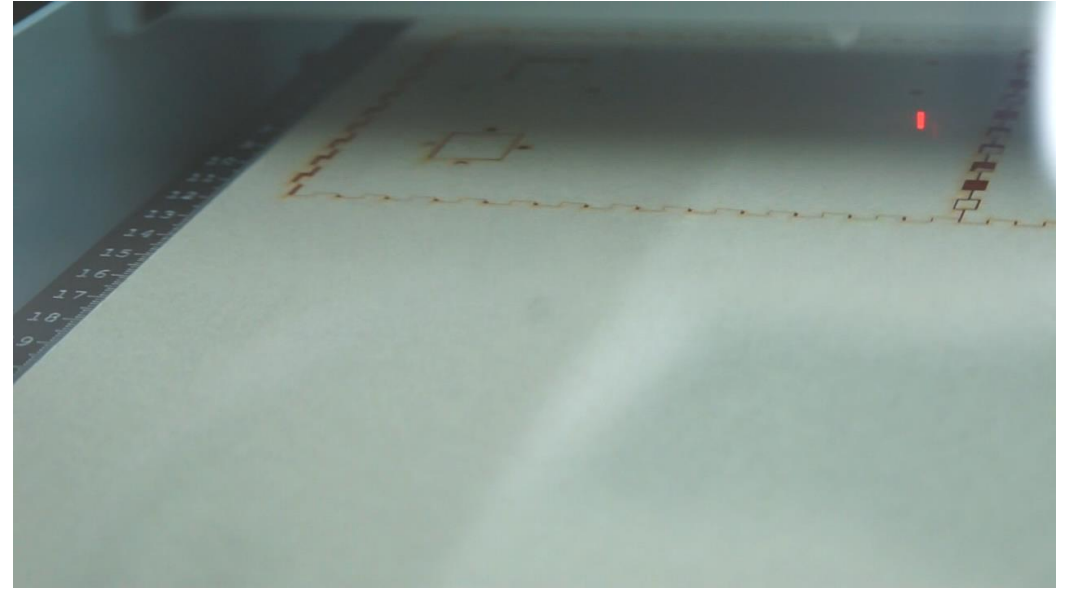
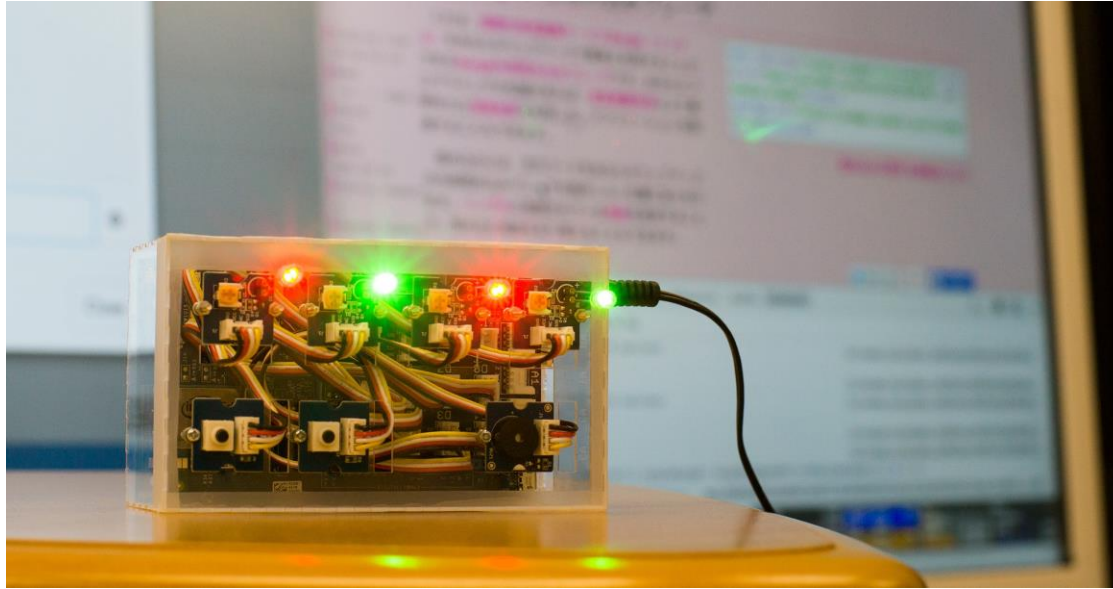
- Reflex time:
  - Visual 0.25s
  - Audio 0.17s
  - Touch 0.15s
- and more ...
  - 3D printers and laser cutters
  - Shape changing devices and robots
  - Taste/smell interfaces



Various kinds of “latencies”

# Printing physical computing devices

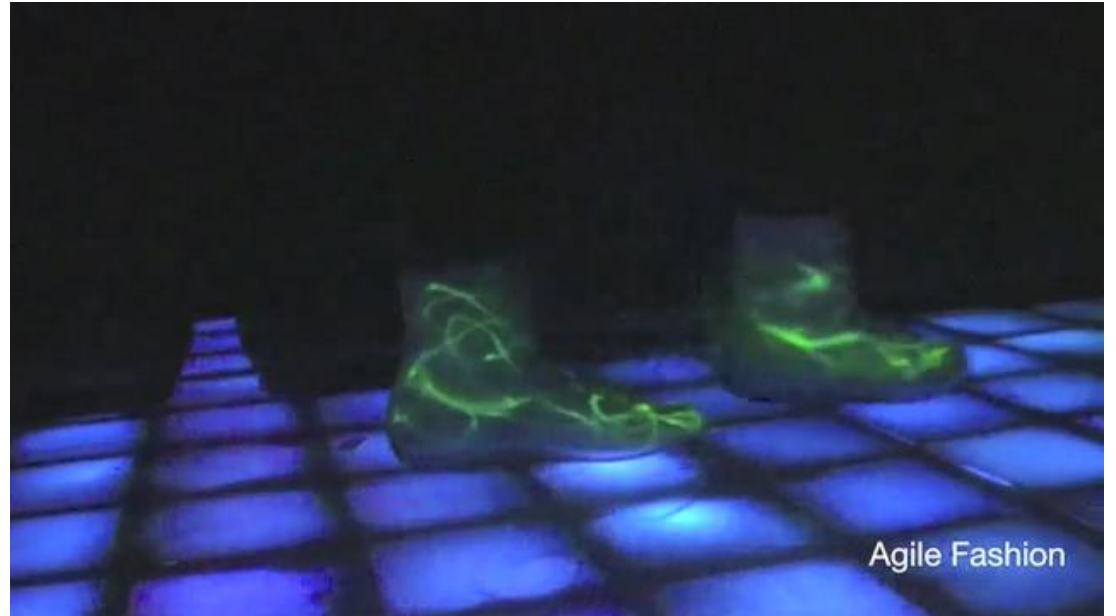
f3.js [DIS 2017]



Slow “framerates” prevent live feedback

# Slow display

Daniel Saakes et al. [SIGGRAPH Etech 2010]



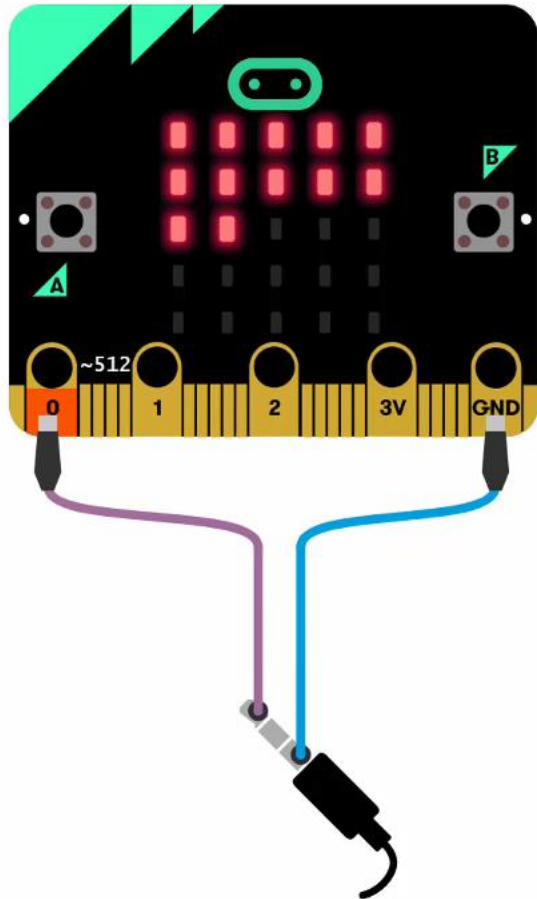
Slow “framerates” can be useful, too

## &lt;/&gt; Source Code

Provide the source code of a microcontroller or tiny computer in JavaScript. Node.js-based computers are supported. Require f3js package and use its API to design the device enclosure.

```
1 var WebSocket = require('ws');
2 var serverAddr = 'ws://192.168.10.100:8080/entry';
3 var ws = new WebSocket(serverAddr);
4 var upmBuzzer = require('jsupm_buzzer');
5
6 var numBuzzers = 5; // Number of buzzers [1, 5]
7 var inputs = [3, 5, 6, 9];
8 var buzzers = [];
9 var volume = 0.1;
10 var r = 80;
11
12 var f3js = require('f3js');
13 var c = f3js.createContainer();
14 c.x = 50;
15 c.y = 50;
16
17 var c1 = 43
18   , c2 = 36
19   , x = 70
20   , y = 140
21   , width = 70;
22 var a = c.createPath();
23
24 var ps = a.extrude(60);
25 ps[0].y = 50;
26 ps[0].x = 270;
27 f3js.add(ps[0]);
28
29 var d1 = f3js.createPath();
30 d1.add([
31   [0, 0],
32   [1, 0],
33   [1, 1],
34   [0, 1],
35   [0, 0]
36 ]);
37 f3js.add(d1);
38
39 var d2 = f3js.createPath();
40 d2.add([
41   [0, 0],
42   [1, 0],
43   [1, 1],
44   [0, 1],
45   [0, 0]
46 ]);
47 f3js.add(d2);
48
49 var d3 = f3js.createPath();
50 d3.add([
51   [0, 0],
52   [1, 0],
53   [1, 1],
54   [0, 1],
55   [0, 0]
56 ]);
57 f3js.add(d3);
58
59 var d4 = f3js.createPath();
60 d4.add([
61   [0, 0],
62   [1, 0],
63   [1, 1],
64   [0, 1],
65   [0, 0]
66 ]);
67 f3js.add(d4);
68
69 var d5 = f3js.createPath();
70 d5.add([
71   [0, 0],
72   [1, 0],
73   [1, 1],
74   [0, 1],
75   [0, 0]
76 ]);
77 f3js.add(d5);
78
79 var d6 = f3js.createPath();
80 d6.add([
81   [0, 0],
82   [1, 0],
83   [1, 1],
84   [0, 1],
85   [0, 0]
86 ]);
87 f3js.add(d6);
88
89 var d7 = f3js.createPath();
90 d7.add([
91   [0, 0],
92   [1, 0],
93   [1, 1],
94   [0, 1],
95   [0, 0]
96 ]);
97 f3js.add(d7);
98
99 var d8 = f3js.createPath();
100 d8.add([
101   [0, 0],
102   [1, 0],
103   [1, 1],
104   [0, 1],
105   [0, 0]
106 ]);
107 f3js.add(d8);
108
109 var d9 = f3js.createPath();
110 d9.add([
111   [0, 0],
112   [1, 0],
113   [1, 1],
114   [0, 1],
115   [0, 0]
116 ]);
117 f3js.add(d9);
118
119 var d10 = f3js.createPath();
120 d10.add([
121   [0, 0],
122   [1, 0],
123   [1, 1],
124   [0, 1],
125   [0, 0]
126 ]);
127 f3js.add(d10);
128
129 var d11 = f3js.createPath();
130 d11.add([
131   [0, 0],
132   [1, 0],
133   [1, 1],
134   [0, 1],
135   [0, 0]
136 ]);
137 f3js.add(d11);
138
139 var d12 = f3js.createPath();
140 d12.add([
141   [0, 0],
142   [1, 0],
143   [1, 1],
144   [0, 1],
145   [0, 0]
146 ]);
147 f3js.add(d12);
148
149 var d13 = f3js.createPath();
150 d13.add([
151   [0, 0],
152   [1, 0],
153   [1, 1],
154   [0, 1],
155   [0, 0]
156 ]);
157 f3js.add(d13);
158
159 var d14 = f3js.createPath();
160 d14.add([
161   [0, 0],
162   [1, 0],
163   [1, 1],
164   [0, 1],
165   [0, 0]
166 ]);
167 f3js.add(d14);
168
169 var d15 = f3js.createPath();
170 d15.add([
171   [0, 0],
172   [1, 0],
173   [1, 1],
174   [0, 1],
175   [0, 0]
176 ]);
177 f3js.add(d15);
178
179 var d16 = f3js.createPath();
180 d16.add([
181   [0, 0],
182   [1, 0],
183   [1, 1],
184   [0, 1],
185   [0, 0]
186 ]);
187 f3js.add(d16);
188
189 var d17 = f3js.createPath();
190 d17.add([
191   [0, 0],
192   [1, 0],
193   [1, 1],
194   [0, 1],
195   [0, 0]
196 ]);
197 f3js.add(d17);
198
199 var d18 = f3js.createPath();
200 d18.add([
201   [0, 0],
202   [1, 0],
203   [1, 1],
204   [0, 1],
205   [0, 0]
206 ]);
207 f3js.add(d18);
208
209 var d19 = f3js.createPath();
210 d19.add([
211   [0, 0],
212   [1, 0],
213   [1, 1],
214   [0, 1],
215   [0, 0]
216 ]);
217 f3js.add(d19);
218
219 var d20 = f3js.createPath();
220 d20.add([
221   [0, 0],
222   [1, 0],
223   [1, 1],
224   [0, 1],
225   [0, 0]
226 ]);
227 f3js.add(d20);
228
229 var d21 = f3js.createPath();
230 d21.add([
231   [0, 0],
232   [1, 0],
233   [1, 1],
234   [0, 1],
235   [0, 0]
236 ]);
237 f3js.add(d21);
238
239 var d22 = f3js.createPath();
240 d22.add([
241   [0, 0],
242   [1, 0],
243   [1, 1],
244   [0, 1],
245   [0, 0]
246 ]);
247 f3js.add(d22);
248
249 var d23 = f3js.createPath();
250 d23.add([
251   [0, 0],
252   [1, 0],
253   [1, 1],
254   [0, 1],
255   [0, 0]
256 ]);
257 f3js.add(d23);
258
259 var d24 = f3js.createPath();
260 d24.add([
261   [0, 0],
262   [1, 0],
263   [1, 1],
264   [0, 1],
265   [0, 0]
266 ]);
267 f3js.add(d24);
268
269 var d25 = f3js.createPath();
270 d25.add([
271   [0, 0],
272   [1, 0],
273   [1, 1],
274   [0, 1],
275   [0, 0]
276 ]);
277 f3js.add(d25);
278
279 var d26 = f3js.createPath();
280 d26.add([
281   [0, 0],
282   [1, 0],
283   [1, 1],
284   [0, 1],
285   [0, 0]
286 ]);
287 f3js.add(d26);
288
289 var d27 = f3js.createPath();
290 d27.add([
291   [0, 0],
292   [1, 0],
293   [1, 1],
294   [0, 1],
295   [0, 0]
296 ]);
297 f3js.add(d27);
298
299 var d28 = f3js.createPath();
300 d28.add([
301   [0, 0],
302   [1, 0],
303   [1, 1],
304   [0, 1],
305   [0, 0]
306 ]);
307 f3js.add(d28);
308
309 var d29 = f3js.createPath();
310 d29.add([
311   [0, 0],
312   [1, 0],
313   [1, 1],
314   [0, 1],
315   [0, 0]
316 ]);
317 f3js.add(d29);
318
319 var d30 = f3js.createPath();
320 d30.add([
321   [0, 0],
322   [1, 0],
323   [1, 1],
324   [0, 1],
325   [0, 0]
326 ]);
327 f3js.add(d30);
328
329 var d31 = f3js.createPath();
330 d31.add([
331   [0, 0],
332   [1, 0],
333   [1, 1],
334   [0, 1],
335   [0, 0]
336 ]);
337 f3js.add(d31);
338
339 var d32 = f3js.createPath();
340 d32.add([
341   [0, 0],
342   [1, 0],
343   [1, 1],
344   [0, 1],
345   [0, 0]
346 ]);
347 f3js.add(d32);
348
349 var d33 = f3js.createPath();
350 d33.add([
351   [0, 0],
352   [1, 0],
353   [1, 1],
354   [0, 1],
355   [0, 0]
356 ]);
357 f3js.add(d33);
358
359 var d34 = f3js.createPath();
360 d34.add([
361   [0, 0],
362   [1, 0],
363   [1, 1],
364   [0, 1],
365   [0, 0]
366 ]);
367 f3js.add(d34);
368
369 var d35 = f3js.createPath();
370 d35.add([
371   [0, 0],
372   [1, 0],
373   [1, 1],
374   [0, 1],
375   [0, 0]
376 ]);
377 f3js.add(d35);
378
379 var d36 = f3js.createPath();
380 d36.add([
381   [0, 0],
382   [1, 0],
383   [1, 1],
384   [0, 1],
385   [0, 0]
386 ]);
387 f3js.add(d36);
388
389 var d37 = f3js.createPath();
390 d37.add([
391   [0, 0],
392   [1, 0],
393   [1, 1],
394   [0, 1],
395   [0, 0]
396 ]);
397 f3js.add(d37);
398
399 var d38 = f3js.createPath();
400 d38.add([
401   [0, 0],
402   [1, 0],
403   [1, 1],
404   [0, 1],
405   [0, 0]
406 ]);
407 f3js.add(d38);
408
409 var d39 = f3js.createPath();
410 d39.add([
411   [0, 0],
412   [1, 0],
413   [1, 1],
414   [0, 1],
415   [0, 0]
416 ]);
417 f3js.add(d39);
418
419 var d40 = f3js.createPath();
420 d40.add([
421   [0, 0],
422   [1, 0],
423   [1, 1],
424   [0, 1],
425   [0, 0]
426 ]);
427 f3js.add(d40);
428
429 var d41 = f3js.createPath();
430 d41.add([
431   [0, 0],
432   [1, 0],
433   [1, 1],
434   [0, 1],
435   [0, 0]
436 ]);
437 f3js.add(d41);
438
439 var d42 = f3js.createPath();
440 d42.add([
441   [0, 0],
442   [1, 0],
443   [1, 1],
444   [0, 1],
445   [0, 0]
446 ]);
447 f3js.add(d42);
448
449 var d43 = f3js.createPath();
450 d43.add([
451   [0, 0],
452   [1, 0],
453   [1, 1],
454   [0, 1],
455   [0, 0]
456 ]);
457 f3js.add(d43);
458
459 var d44 = f3js.createPath();
460 d44.add([
461   [0, 0],
462   [1, 0],
463   [1, 1],
464   [0, 1],
465   [0, 0]
466 ]);
467 f3js.add(d44);
468
469 var d45 = f3js.createPath();
470 d45.add([
471   [0, 0],
472   [1, 0],
473   [1, 1],
474   [0, 1],
475   [0, 0]
476 ]);
477 f3js.add(d45);
478
479 var d46 = f3js.createPath();
480 d46.add([
481   [0, 0],
482   [1, 0],
483   [1, 1],
484   [0, 1],
485   [0, 0]
486 ]);
487 f3js.add(d46);
488
489 var d47 = f3js.createPath();
490 d47.add([
491   [0, 0],
492   [1, 0],
493   [1, 1],
494   [0, 1],
495   [0, 0]
496 ]);
497 f3js.add(d47);
498
499 var d48 = f3js.createPath();
500 d48.add([
501   [0, 0],
502   [1, 0],
503   [1, 1],
504   [0, 1],
505   [0, 0]
506 ]);
507 f3js.add(d48);
508
509 var d49 = f3js.createPath();
510 d49.add([
511   [0, 0],
512   [1, 0],
513   [1, 1],
514   [0, 1],
515   [0, 0]
516 ]);
517 f3js.add(d49);
518
519 var d50 = f3js.createPath();
520 d50.add([
521   [0, 0],
522   [1, 0],
523   [1, 1],
524   [0, 1],
525   [0, 0]
526 ]);
527 f3js.add(d50);
528
529 var d51 = f3js.createPath();
530 d51.add([
531   [0, 0],
532   [1, 0],
533   [1, 1],
534   [0, 1],
535   [0, 0]
536 ]);
537 f3js.add(d51);
538
539 var d52 = f3js.createPath();
540 d52.add([
541   [0, 0],
542   [1, 0],
543   [1, 1],
544   [0, 1],
545   [0, 0]
546 ]);
547 f3js.add(d52);
548
549 var d53 = f3js.createPath();
550 d53.add([
551   [0, 0],
552   [1, 0],
553   [1, 1],
554   [0, 1],
555   [0, 0]
556 ]);
557 f3js.add(d53);
558
559 var d54 = f3js.createPath();
560 d54.add([
561   [0, 0],
562   [1, 0],
563   [1, 1],
564   [0, 1],
565   [0, 0]
566 ]);
567 f3js.add(d54);
568
569 var d55 = f3js.createPath();
570 d55.add([
571   [0, 0],
572   [1, 0],
573   [1, 1],
574   [0, 1],
575   [0, 0]
576 ]);
577 f3js.add(d55);
578
579 var d56 = f3js.createPath();
580 d56.add([
581   [0, 0],
582   [1, 0],
583   [1, 1],
584   [0, 1],
585   [0, 0]
586 ]);
587 f3js.add(d56);
588
589 var d57 = f3js.createPath();
590 d57.add([
591   [0, 0],
592   [1, 0],
593   [1, 1],
594   [0, 1],
595   [0, 0]
596 ]);
597 f3js.add(d57);
598
599 var d58 = f3js.createPath();
600 d58.add([
601   [0, 0],
602   [1, 0],
603   [1, 1],
604   [0, 1],
605   [0, 0]
606 ]);
607 f3js.add(d58);
608
609 var d59 = f3js.createPath();
610 d59.add([
611   [0, 0],
612   [1, 0],
613   [1, 1],
614   [0, 1],
615   [0, 0]
616 ]);
617 f3js.add(d59);
618
619 var d60 = f3js.createPath();
620 d60.add([
621   [0, 0],
622   [1, 0],
623   [1, 1],
624   [0, 1],
625   [0, 0]
626 ]);
627 f3js.add(d60);
628
629 var d61 = f3js.createPath();
630 d61.add([
631   [0, 0],
632   [1, 0],
633   [1, 1],
634   [0, 1],
635   [0, 0]
636 ]);
637 f3js.add(d61);
638
639 var d62 = f3js.createPath();
640 d62.add([
641   [0, 0],
642   [1, 0],
643   [1, 1],
644   [0, 1],
645   [0, 0]
646 ]);
647 f3js.add(d62);
648
649 var d63 = f3js.createPath();
650 d63.add([
651   [0, 0],
652   [1, 0],
653   [1, 1],
654   [0, 1],
655   [0, 0]
656 ]);
657 f3js.add(d63);
658
659 var d64 = f3js.createPath();
660 d64.add([
661   [0, 0],
662   [1, 0],
663   [1, 1],
664   [0, 1],
665   [0, 0]
666 ]);
667 f3js.add(d64);
668
669 var d65 = f3js.createPath();
670 d65.add([
671   [0, 0],
672   [1, 0],
673   [1, 1],
674   [0, 1],
675   [0, 0]
676 ]);
677 f3js.add(d65);
678
679 var d66 = f3js.createPath();
680 d66.add([
681   [0, 0],
682   [1, 0],
683   [1, 1],
684   [0, 1],
685   [0, 0]
686 ]);
687 f3js.add(d66);
688
689 var d67 = f3js.createPath();
690 d67.add([
691   [0, 0],
692   [1, 0],
693   [1, 1],
694   [0, 1],
695   [0, 0]
696 ]);
697 f3js.add(d67);
698
699 var d68 = f3js.createPath();
700 d68.add([
701   [0, 0],
702   [1, 0],
703   [1, 1],
704   [0, 1],
705   [0, 0]
706 ]);
707 f3js.add(d68);
708
709 var d69 = f3js.createPath();
710 d69.add([
711   [0, 0],
712   [1, 0],
713   [1, 1],
714   [0, 1],
715   [0, 0]
716 ]);
717 f3js.add(d69);
718
719 var d70 = f3js.createPath();
720 d70.add([
721   [0, 0],
722   [1, 0],
723   [1, 1],
724   [0, 1],
725   [0, 0]
726 ]);
727 f3js.add(d70);
728
729 var d71 = f3js.createPath();
730 d71.add([
731   [0, 0],
732   [1, 0],
733   [1, 1],
734   [0, 1],
735   [0, 0]
736 ]);
737 f3js.add(d71);
738
739 var d72 = f3js.createPath();
740 d72.add([
741   [0, 0],
742   [1, 0],
743   [1, 1],
744   [0, 1],
745   [0, 0]
746 ]);
747 f3js.add(d72);
748
749 var d73 = f3js.createPath();
750 d73.add([
751   [0, 0],
752   [1, 0],
753   [1, 1],
754   [0, 1],
755   [0, 0]
756 ]);
757 f3js.add(d73);
758
759 var d74 = f3js.createPath();
760 d74.add([
761   [0, 0],
762   [1, 0],
763   [1, 1],
764   [0, 1],
765   [0, 0]
766 ]);
767 f3js.add(d74);
768
769 var d75 = f3js.createPath();
770 d75.add([
771   [0, 0],
772   [1, 0],
773   [1, 1],
774   [0, 1],
775   [0, 0]
776 ]);
777 f3js.add(d75);
778
779 var d76 = f3js.createPath();
780 d76.add([
781   [0, 0],
782   [1, 0],
783   [1, 1],
784   [0, 1],
785   [0, 0]
786 ]);
787 f3js.add(d76);
788
789 var d77 = f3js.createPath();
790 d77.add([
791   [0, 0],
792   [1, 0],
793   [1, 1],
794   [0, 1],
795   [0, 0]
796 ]);
797 f3js.add(d77);
798
799 var d78 = f3js.createPath();
800 d78.add([
801   [0, 0],
802   [1, 0],
803   [1, 1],
804   [0, 1],
805   [0, 0]
806 ]);
807 f3js.add(d78);
808
809 var d79 = f3js.createPath();
810 d79.add([
811   [0, 0],
812   [1, 0],
813   [1, 1],
814   [0, 1],
815   [0, 0]
816 ]);
817 f3js.add(d79);
818
819 var d80 = f3js.createPath();
820 d80.add([
821   [0, 0],
822   [1, 0],
823   [1, 1],
824   [0, 1],
825   [0, 0]
826 ]);
827 f3js.add(d80);
828
829 var d81 = f3js.createPath();
830 d81.add([
831   [0, 0],
832   [1, 0],
833   [1, 1],
834   [0, 1],
835   [0, 0]
836 ]);
837 f3js.add(d81);
838
839 var d82 = f3js.createPath();
840 d82.add([
841   [0, 0],
842   [1, 0],
843   [1, 1],
844   [0, 1],
845   [0, 0]
846 ]);
847 f3js.add(d82);
848
849 var d83 = f3js.createPath();
850 d83.add([
851   [0, 0],
852   [1, 0],
853   [1, 1],
854   [0, 1],
855   [0, 0]
856 ]);
857 f3js.add(d83);
858
859 var d84 = f3js.createPath();
860 d84.add([
861   [0, 0],
862   [1, 0],
863   [1, 1],
864   [0, 1],
865   [0, 0]
866 ]);
867 f3js.add(d84);
868
869 var d85 = f3js.createPath();
870 d85.add([
871   [0, 0],
872   [1, 0],
873   [1, 1],
874   [0, 1],
875   [0, 0]
876 ]);
877 f3js.add(d85);
878
879 var d86 = f3js.createPath();
880 d86.add([
881   [0, 0],
882   [1, 0],
883   [1, 1],
884   [0, 1],
885   [0, 0]
886 ]);
887 f3js.add(d86);
888
889 var d87 = f3js.createPath();
890 d87.add([
891   [0, 0],
892   [1, 0],
893   [1, 1],
894   [0, 1],
895   [0, 0]
896 ]);
897 f3js.add(d87);
898
899 var d88 = f3js.createPath();
900 d88.add([
901   [0, 0],
902   [1, 0],
903   [1, 1],
904   [0, 1],
905   [0, 0]
906 ]);
907 f3js.add(d88);
908
909 var d89 = f3js.createPath();
910 d89.add([
911   [0, 0],
912   [1, 0],
913   [1, 1],
914   [0, 1],
915   [0, 0]
916 ]);
917 f3js.add(d89);
918
919 var d90 = f3js.createPath();
920 d90.add([
921   [0, 0],
922   [1, 0],
923   [1, 1],
924   [0, 1],
925   [0, 0]
926 ]);
927 f3js.add(d90);
928
929 var d91 = f3js.createPath();
930 d91.add([
931   [0, 0],
932   [1, 0],
933   [1, 1],
934   [0, 1],
935   [0, 0]
936 ]);
937 f3js.add(d91);
938
939 var d92 = f3js.createPath();
940 d92.add([
941   [0, 0],
942   [1, 0],
943   [1, 1],
944   [0, 1],
945   [0, 0]
946 ]);
947 f3js.add(d92);
948
949 var d93 = f3js.createPath();
950 d93.add([
951   [0, 0],
952   [1, 0],
953   [1, 1],
954   [0, 1],
955   [0, 0]
956 ]);
957 f3js.add(d93);
958
959 var d94 = f3js.createPath();
960 d94.add([
961   [0, 0],
962   [1, 0],
963   [1, 1],
964   [0, 1],
965   [0, 0]
966 ]);
967 f3js.add(d94);
968
969 var d95 = f3js.createPath();
970 d95.add([
971   [0, 0],
972   [1, 0],
973   [1, 1],
974   [0, 1],
975   [0, 0]
976 ]);
977 f3js.add(d95);
978
979 var d96 = f3js.createPath();
980 d96.add([
981   [0, 0],
982   [1, 0],
983   [1, 1],
984   [0, 1],
985   [0, 0]
986 ]);
987 f3js.add(d96);
988
989 var d97 = f3js.createPath();
990 d97.add([
991   [0, 0],
992   [1, 0],
993   [1, 1],
994   [0, 1],
995   [0, 0]
996 ]);
997 f3js.add(d97);
998
999 var d98 = f3js.createPath();
1000 d98.add([
1001   [0, 0],
1002   [1, 0],
1003   [1, 1],
1004   [0, 1],
1005   [0, 0]
1006 ]);
1007 f3js.add(d98);
1008
1009 var d99 = f3js.createPath();
1010 d99.add([
1011   [0, 0],
1012   [1, 0],
1013   [1, 1],
1014   [0, 1],
1015   [0, 0]
1016 ]);
1017 f3js.add(d99);
1018
1019 var d100 = f3js.createPath();
1020 d100.add([
1021   [0, 0],
1022   [1, 0],
1023   [1, 1],
1024   [0, 1],
1025   [0, 0]
1026 ]);
1027 f3js.add(d100);
1028
1029 var d101 = f3js.createPath();
1030 d101.add([
1031   [0, 0],
1032   [1, 0],
1033   [1, 1],
1034   [0, 1],
1035   [0, 0]
1036 ]);
1037 f3js.add(d101);
1038
1039 var d102 = f3js.createPath();
1040 d102.add([
1041   [0, 0],
1042   [1, 0],
1043   [1, 1],
1044   [0, 1],
1045   [0, 0]
1046 ]);
1047 f3js.add(d102);
1048
1049 var d103 = f3js.createPath();
1050 d103.add([
1051   [0, 0],
1052   [1, 0],
1053   [1, 1],
1054   [0, 1],
1055   [0, 0]
1056 ]);
1057 f3js.add(d103);
1058
1059 var d104 = f3js.createPath();
1060 d104.add([
1061   [0, 0],
1062   [1, 0],
1063   [1, 1],
1064   [0, 1],
1065   [0, 0]
1066 ]);
1067 f3js.add(d104);
1068
1069 var d105 = f3js.createPath();
1070 d105.add([
1071   [0, 0],
1072   [1, 0],
1073   [1, 1],
1074   [0, 1],
1075   [0, 0]
1076 ]);
1077 f3js.add(d105);
1078
1079 var d106 = f3js.createPath();
1080 d106.add([
1081   [0, 0],
1082   [1, 0],
1083   [1, 1],
1084   [0, 1],
1085   [0, 0]
1086 ]);
1087 f3js.add(d106);
1088
1089 var d107 = f3js.createPath();
1090 d107.add([
1091   [0, 0],
1092   [1, 0],
1093   [1, 1],
1094   [0, 1],
1095   [0, 0]
1096 ]);
1097 f3js.add(d107);
1098
1099 var d108 = f3js.createPath();
1100 d108.add([
1101   [0, 0],
1102   [1, 0],
1103   [1, 1],
1104   [0, 1],
1105   [0, 0]
1106 ]);
1107 f3js.add(d108);
1108
1109 var d109 = f3js.createPath();
1110 d109.add([
1111   [0, 0],
1112   [1, 0],
1113   [1, 1],
1114   [0, 1],
1115   [0, 0]
1116 ]);
1117 f3js.add(d109);
1118
1119 var d110 = f3js.createPath();
1120 d110.add([
1121   [0, 0],
1122   [1, 0],
1123   [1, 1],
1124   [0, 1],
1125   [0, 0]
1126 ]);
1127 f3js.add(d110);
1128
1129 var d111 = f3js.createPath();
1130 d111.add([
1131   [0, 0],
1132   [1, 0],
1133   [1, 1],
1134   [0, 1],
1135   [0, 0]
1136 ]);
1137 f3js.add(d111);
1138
1139 var d112 = f3js.createPath();
1140 d112.add([
1141   [0, 0],
1142   [1, 0],
1143   [1, 1],
1144   [0, 1],
1145   [0, 0]
1146 ]);
1147 f3js.add(d112);
1148
1149 var d113 = f3js.createPath();
1150 d113.add([
1151   [0, 0],
1152   [1, 0],
1153   [1, 1],
1154   [0, 1],
1155   [0, 0]
1156 ]);
1157 f3js.add(d113);
1158
1159 var d114 = f3js.createPath();
1160 d114.add([
1161   [0, 0],
1162   [1, 0],
1163   [1, 1],
1164   [0, 1],
1165   [0, 0]
1166 ]);
1167 f3js.add(d114);
1168
1169 var d115 = f3js.createPath();
1170 d115.add([
1171   [0, 0],
1172   [1, 0],
1173   [1, 1],
1174   [0, 1],
1175   [0, 0]
1176 ]);
1177 f3js
```





Search...



Basic

Input

Music

Led

Radio

Loops

Logic

Variables

Math

Advanced

forever

ring tone (Hz)

Middle C

on start

while

true

do

for index-y from 0 to 4

do

for index from 0 to 4

do

toggle x index y index-y

pause (ms)

100

MakeCode for BBC micro:bit, Microsoft Research [2017]

<http://makecode.microbit.org>



Download

blinking-leds



User Interfaces for Live Programming





# Picode: inline photos representing posture data in source code

[CHI 2013]

BallShooter

```
if (pose.eq()) {  
  showText("Got the command!");  
  
  ;  
  nxt.setPose(  
}  
}
```



Human



Robot

My NXT (MindstormsNXT)

Edit



none



right



Undo



Redo



Capture



Delete

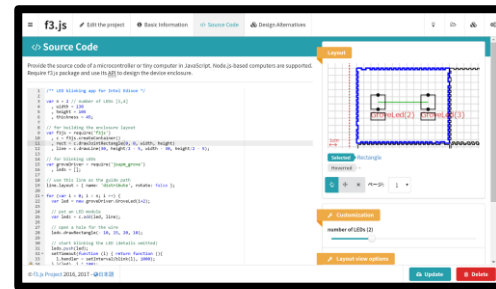
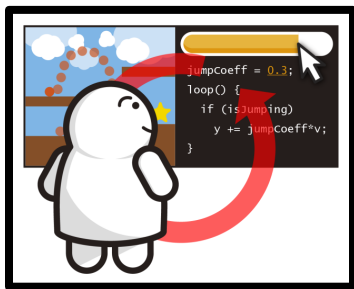
When designing live programming systems ...

# Deceiving users' perception is a good thing

- As long as **the lie is reasonable**
- The actual “**framerate**” can be very slow
- Emulating, or sometimes even pretending, is needed to provide the **continuous feedback**
- Make **full use of five senses** in programming environments

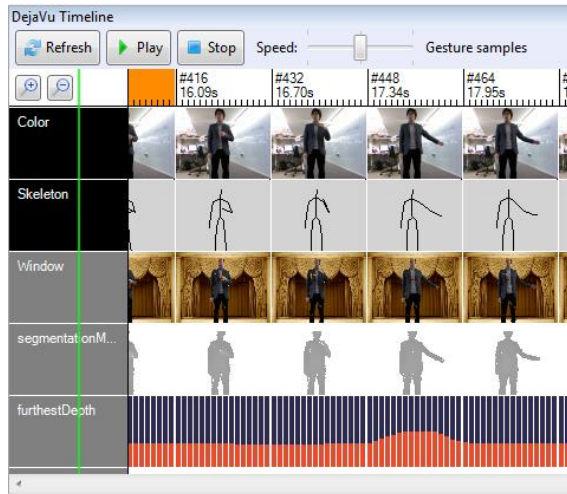
# Today, I'm going to talk about ...

- What is Live Programming?
- UIs for Live Programming with end-users
- UIs for Live Programming of this material world
- **UIs for Live Programming with time travel**
- Live Programming as User Interface research



# “Live” is not always about “now”

- **UIs for exploring and modifying the past**
- UIs for predicting and choosing the future
- Absolute vs semantic timeline



Window1.Xaml.cs

OnKinectFramesReady(object sender, Kinect)

```
86
87
88 float sumDistance = 0;
89 foreach (Joint joint in skeletonData.Joints) {
90     sumDistance += joint.Position.Z;
91 }
92 float userDistance = sumDistance / 20;
93
94 //If the user is close enough to the camera, show the vir
95 bool userIsNear = userDistance < 2.5;
96 ShowStage(userIsNear);
97
98
99
100
101
102
```

DejaVu Canvas

Color Depth Skeleton Window

Color input

Window output

userDistance

Properties Classes DejaVu Canvas

DejaVu Timeline

Refresh Play Stop Speed: Session 17

#0 0.00s #32 1.34s #64 2.65s #96 3.87s #128 5.15s #160 6.36s #192 7.61s #224 8.80s

Color

Window

userDistance

DejaVu [UIST 2012]

Session 17, 580 frames, 23.00s

LIVE New session

KinectDress

User interfaces for Live Programming

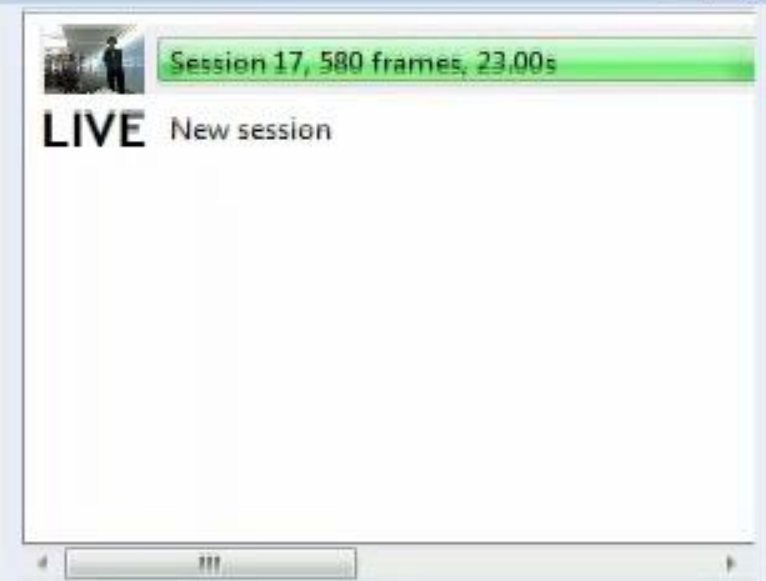
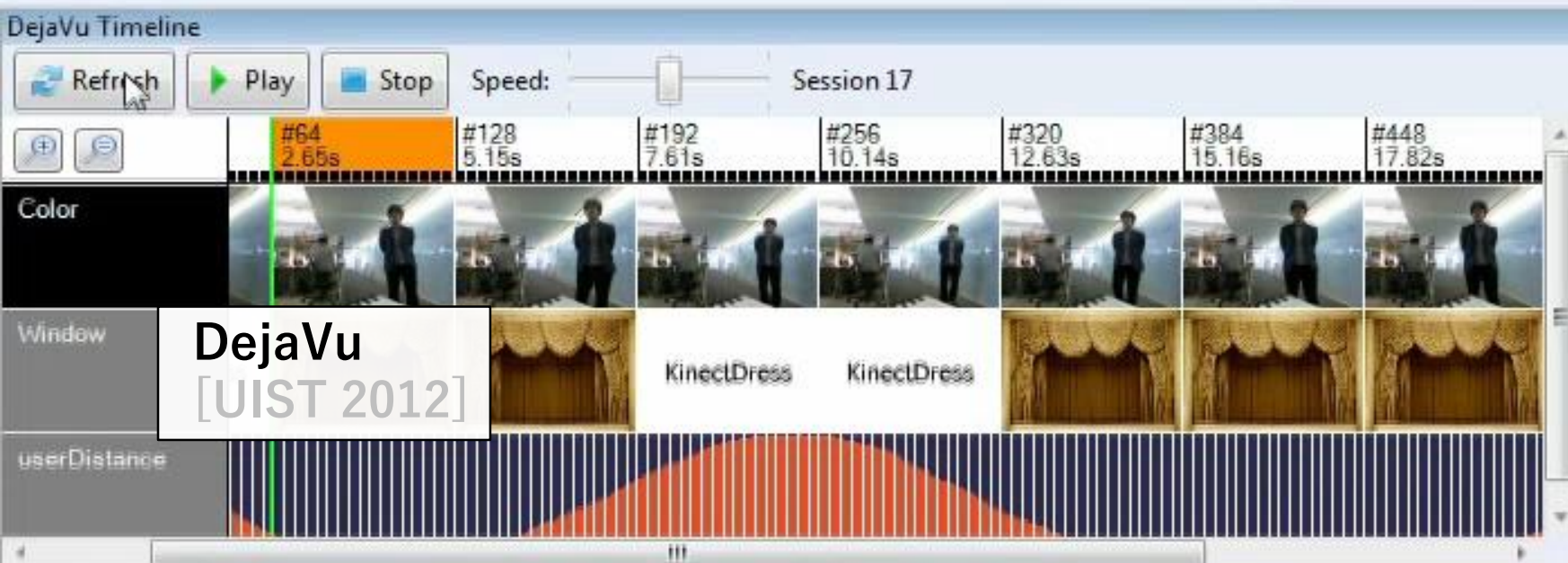


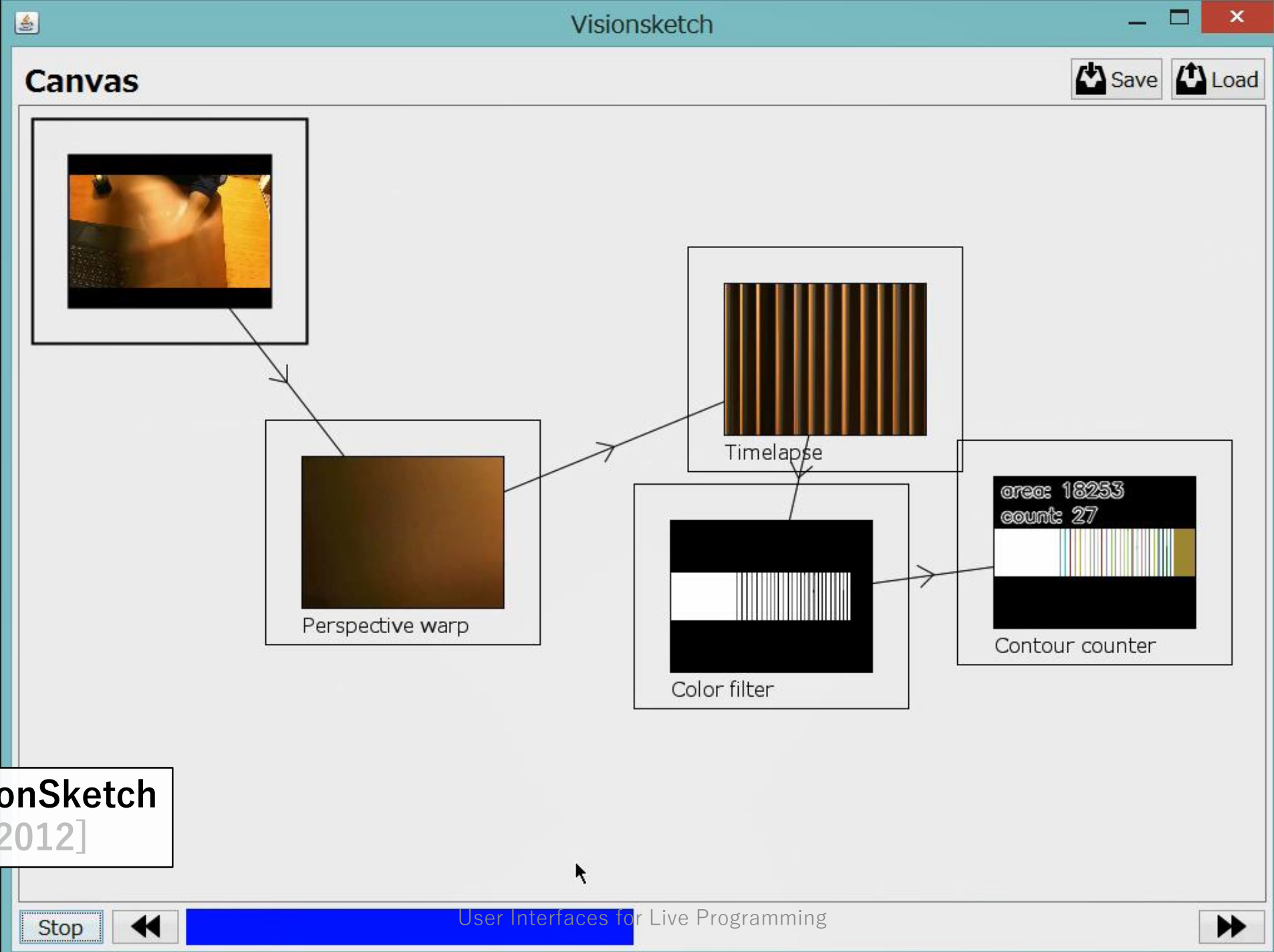
Window1.xaml.cs

KinectDress.Window1

OnKinectFramesReady(object sender, Kinect

```
86
87
88     float sumDistance = 0;
89     foreach (Joint joint in skeletonData.Joints) {
90         sumDistance += joint.Position.Z;
91     }
92     float userDistance = sumDistance / 20;
93
94     //If the user is close enough to the camera, show the vir
95     bool userIsNear = userDistance < 3;
96     ShowStage(userIsNear);
97
98
99
100
101
102
```

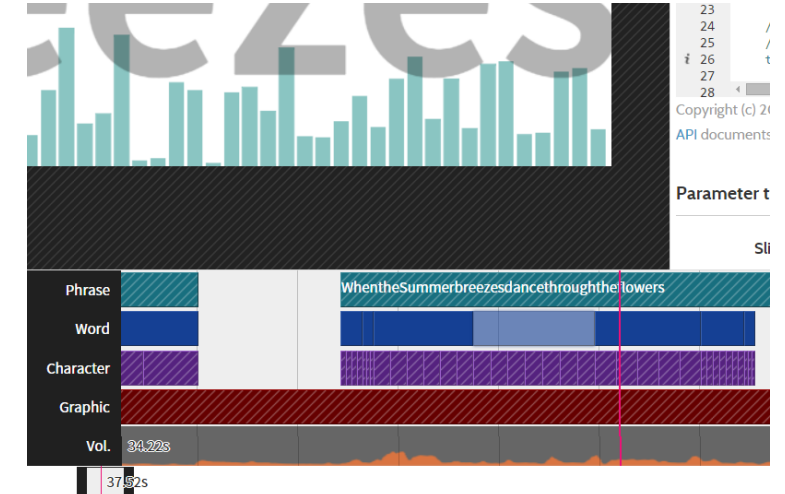
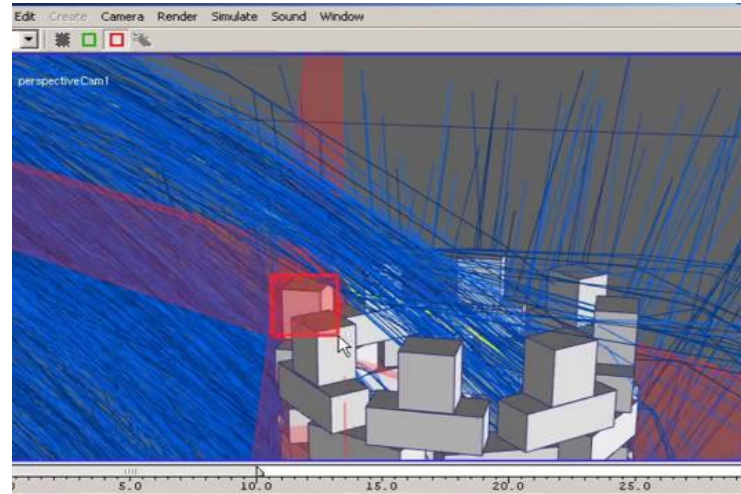
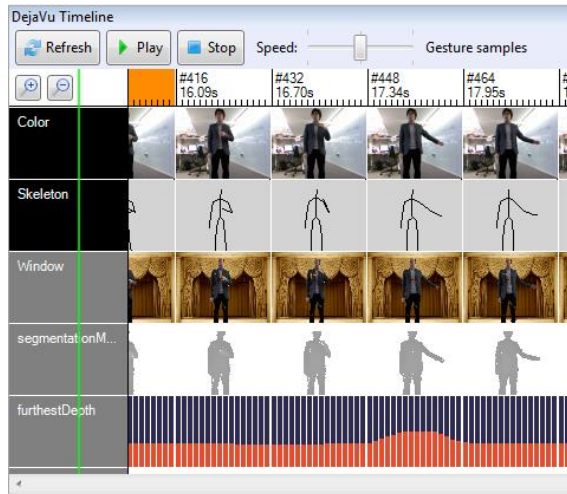




**VisionSketch**  
[GI 2012]

# “Live” is not always about “now”

- UIs for exploring and modifying the past
- **UIs for predicting and choosing the future**
- Absolute vs semantic timeline



## Light Table – a new IDE concept, Chris Granger [2012]

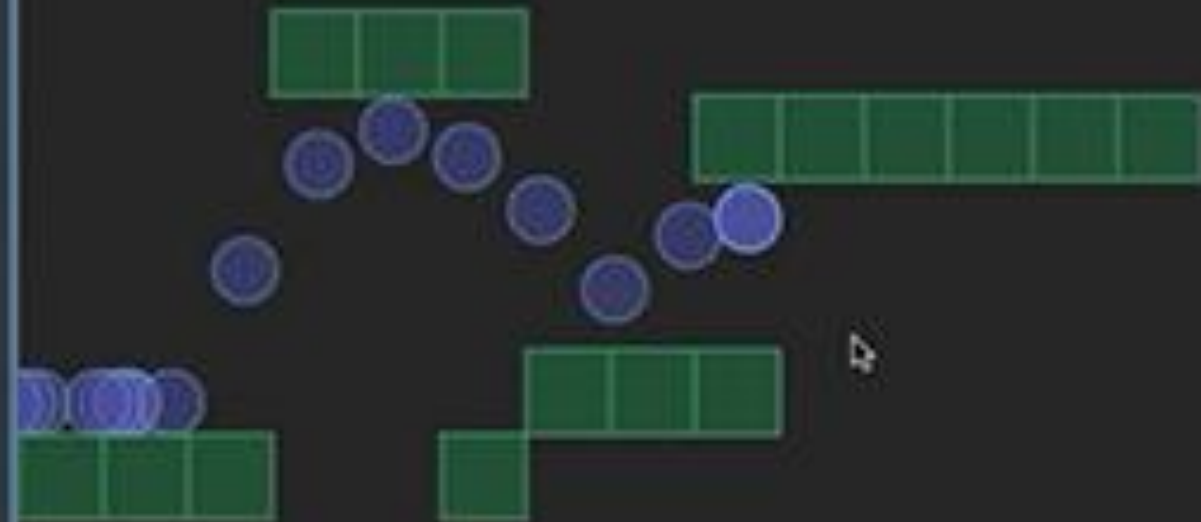
<http://www.chris-granger.com/2012/04/12/light-table-a-new-ide-concept/>

```
defn jump [me]
  (let [speed 7]
    (if (and
        (input? :left)
        (input? :right))
        (assoc me :jumping true)
        me)))

defn move [me]
  (let [speed 7
        vx (cond
              (input? :left) (- speed)
              (input? :right) speed
              :else 0)]
    (moved (update-in me [:x] + vx))
    (if (zero? vx)
        me
        (if-let [block (colliding? moved)]
            (let [block-edge (if (< vx 0)
                                (= [x block] [y block] [z me])
                                (= [x block] [z me]))]
              (assoc me :x block-edge))
            moved))))

defn reset [me]
  (if (> (y me) 450)
    (-> me
        (assoc :x 30)
        (assoc :y 30)
        (assoc :vy 0))
    me)

defn update-player [me]
  (-> me
      (gravity)
      (move)
      (jump)
      (reset)
      ))
```







**Many-Worlds Browsing for Control of Multibody Dynamics**  
Twigg et al. [SIGGRAPH 2007]

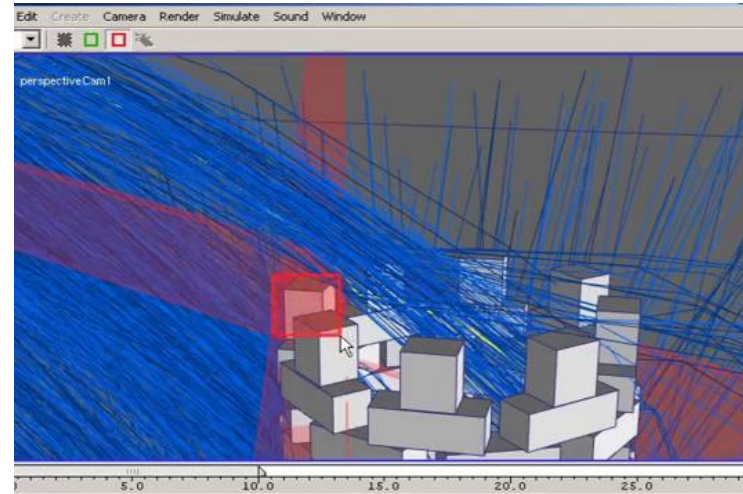
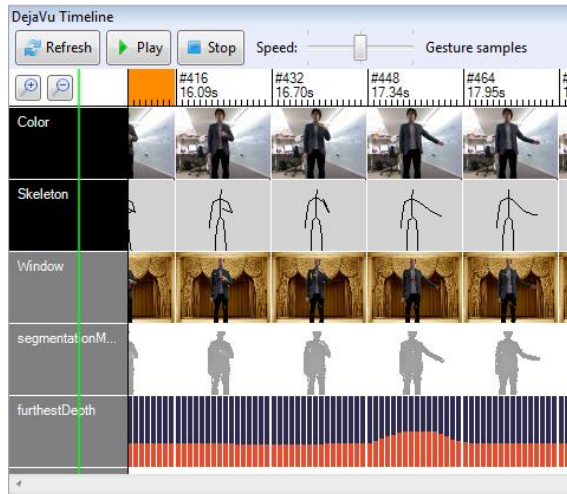




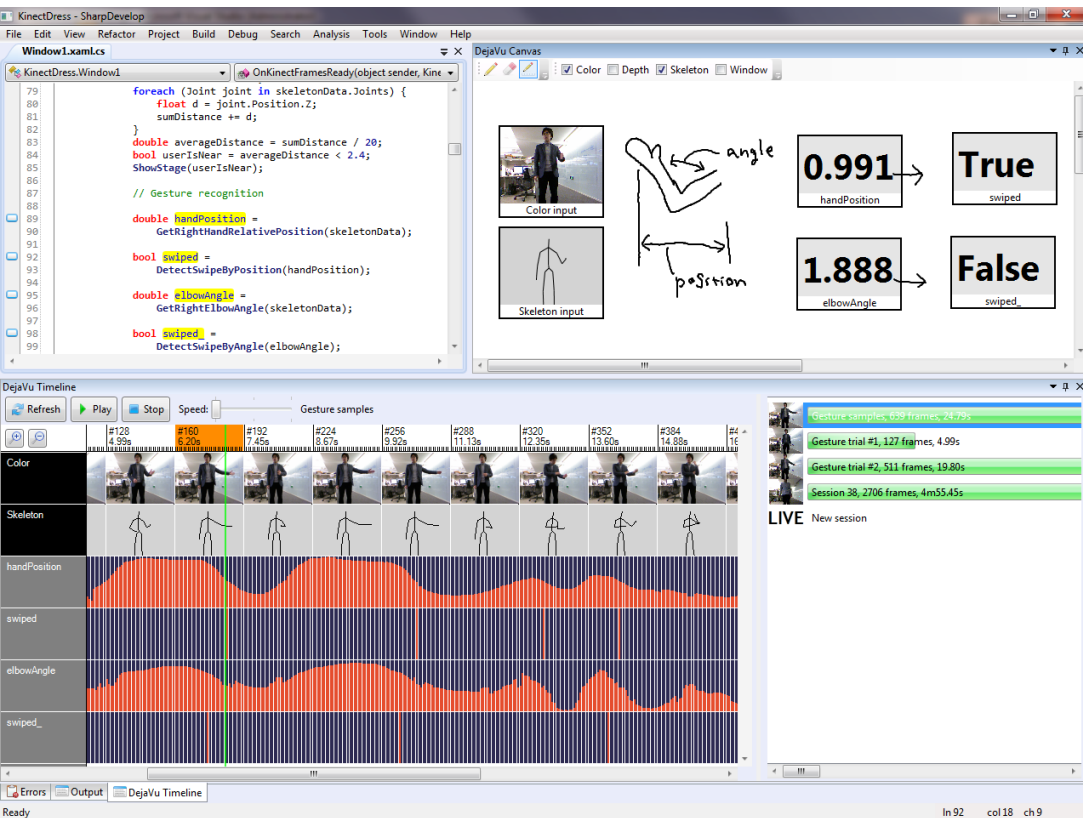
Rkter, McCann (TCHOW)  
[2013]

# “Live” is not always about “now”

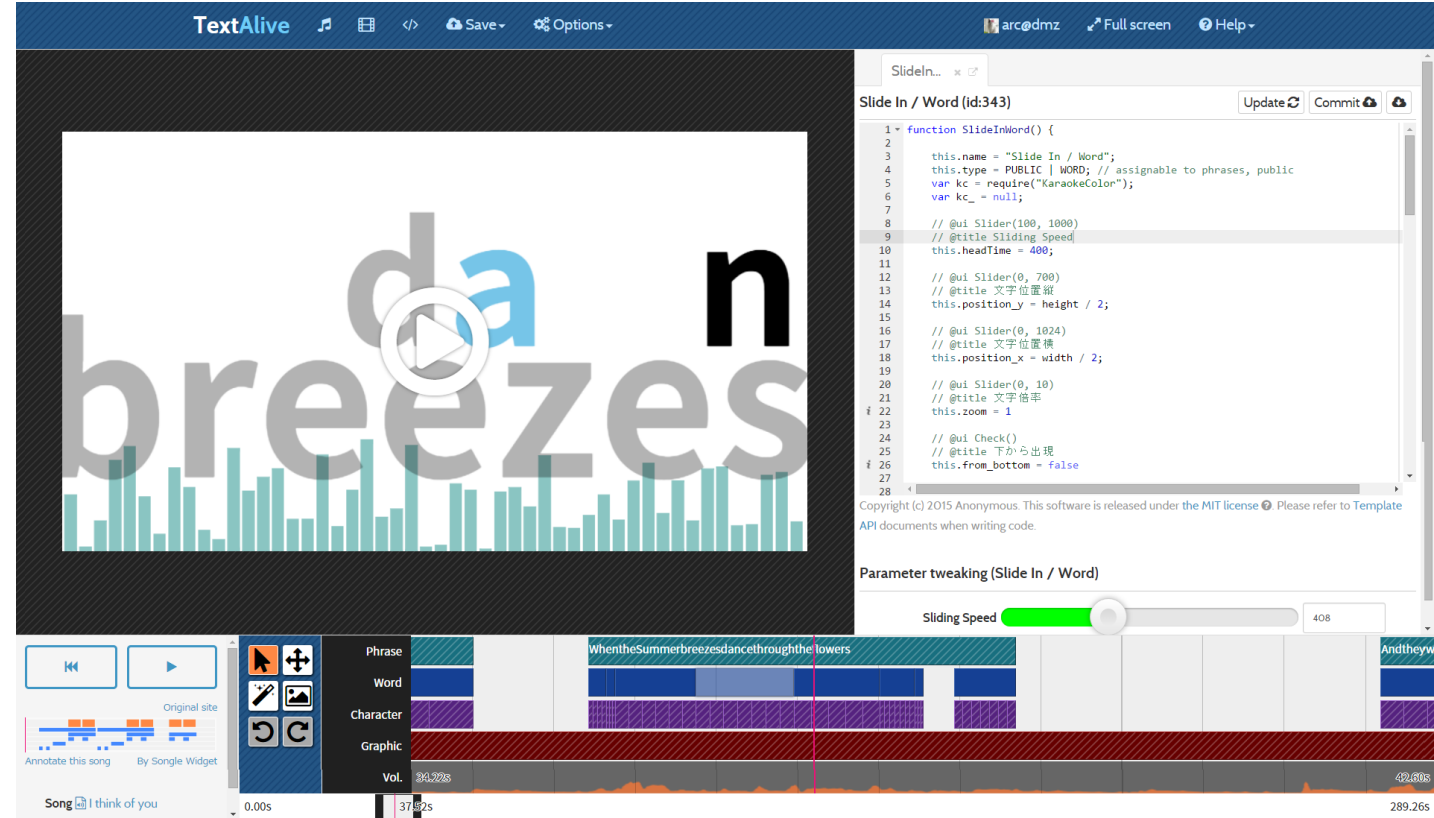
- UIs for exploring and modifying the past
- UIs for predicting and choosing the future
- **Absolute vs semantic timeline**



# Absolute time vs semantic time



DeJaVu [UIST 2012]



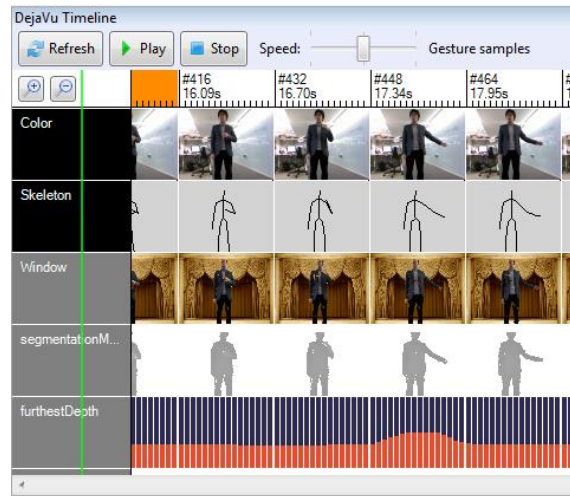
TextAlive [CHI 2015]



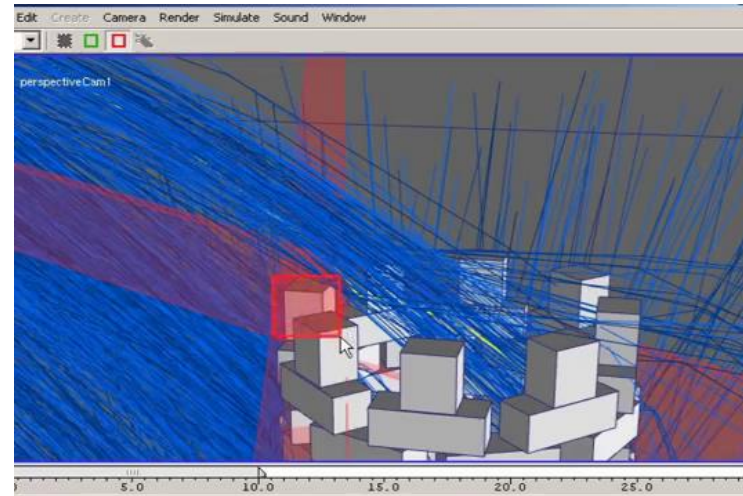
When designing live programming systems ...

# Try providing good sense of time

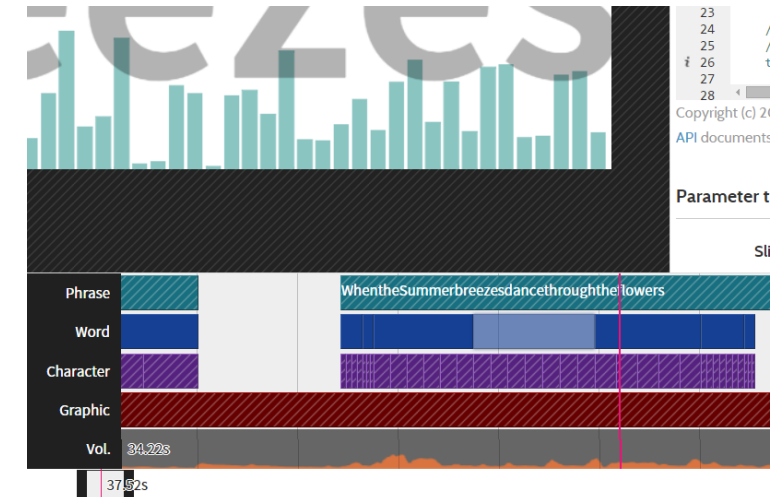
- Enable time travel to find critical timings in the history
- Allow editing the code and program input to explore futures



**Replay & Refresh**  
Superspeed & slowmo



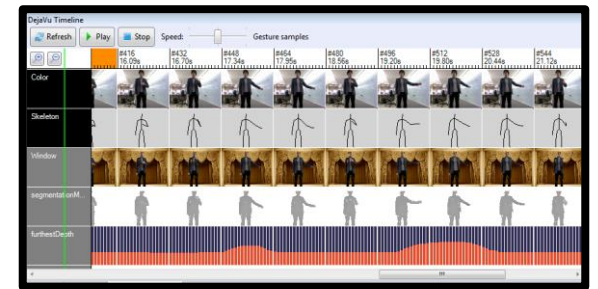
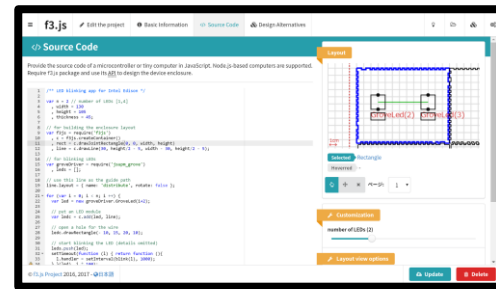
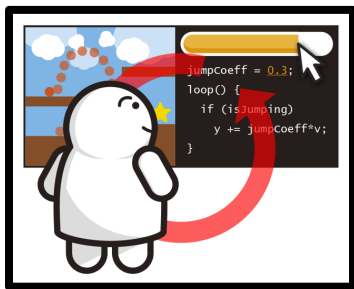
**“Many worlds”**  
Stroboscopic visualization



**Timeline**  
for absolute/semantic time

# Today, I'm going to talk about ...

- What is Live Programming?
- UIs for Live Programming with end-users
- UIs for Live Programming of this material world
- UIs for Live Programming with time travel
- **Live Programming as User Interface research**





# Live Programming research as User Interface research

- Don't be afraid to be domain-specific
- How about making the ladder of expertise?
- Deceiving users' perception is a good thing
- Try providing good sense of time




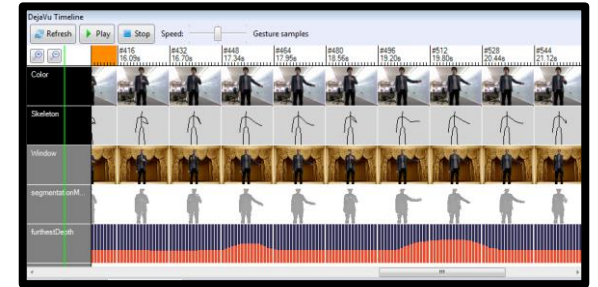
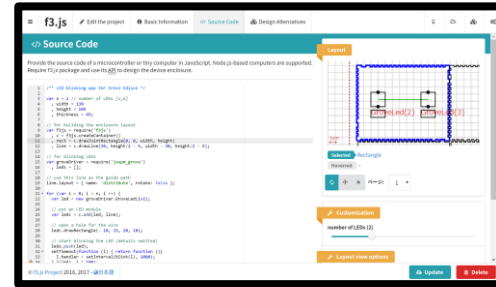
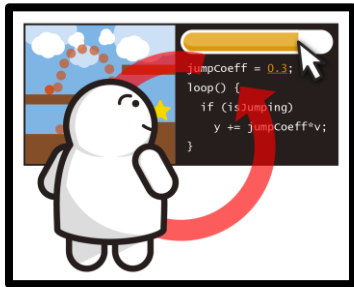
It's **not only** about language design,  
a single user, a single UI,  
but about **designing the whole experience**

# User Interfaces for Live Programming

Jun Kato

<https://junkato.jp>

Researcher,  **AIST**



LIVE 2017 Keynote, 10/24/2017

# References

1. Jun Kato, Masataka Goto, "**f3.js: A Parametric Design Tool for Physical Computing Devices for Both Interaction Designers and End-users**", In *Proceedings of the 2017 Conference on Designing Interactive Systems*. pp.1099-1110, 2017.
2. Jun Kato, Masataka Goto, "**Live Tuning: Expanding Live Programming Benefits to Non-Programmers**", In *Proceedings of the Second Workshop on Live Programming Systems*. 2016.
3. Jun Kato, Takeo Igarashi, Masataka Goto, "**Programming with Examples to Develop Data-Intensive User Interfaces**", In *Computer* 49(7). pp.34-42, Jul. 2016. **Special Issue on 21st User Interfaces**.
4. Jun Kato, Tomoyasu Nakano, Masataka Goto, "**TextAlive: Integrated Design Environment for Kinetic Typography**", In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. pp.3403-3412, 2015. **ACM CHI 2015 Best Paper Honorable Mention Award**.
5. Jun Kato, Takeo Igarashi, "**VisionSketch: Integrated Support for Example-centric Programming of Image Processing Applications**", In *Proceedings of the 2014 Graphics Interface Conference*. pp.115-122, 2014.
6. Sebastian Burckhardt, Manuel Fahndrich, Peli Halleux, Sean McDirmid, Michal Moskal, Nikolai Tillmann, Jun Kato, "**It's Alive! Continuous Feedback in UI Programming**", In *PLDI '13: Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*. pp.95-104, 2013.
7. Jun Kato, Daisuke Sakamoto, Takeo Igarashi, "**Picode: Inline Photos Representing Posture Data in Source Code**", In *CHI '13: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. pp.3097-3100, Apr. 2013. **ACM CHI 2013 Best Paper Honorable Mention Award**.
8. Jun Kato, Sean McDirmid, Xiang Cao, "**DejaVu: Integrated Support for Developing Interactive Camera-Based Programs**", In *UIST '12: Proceedings of the 25th annual ACM symposium on User Interface Software and Technology*. pp.189-196, Oct. 2012.