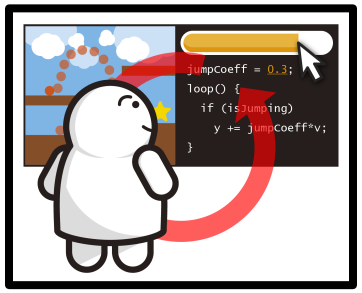
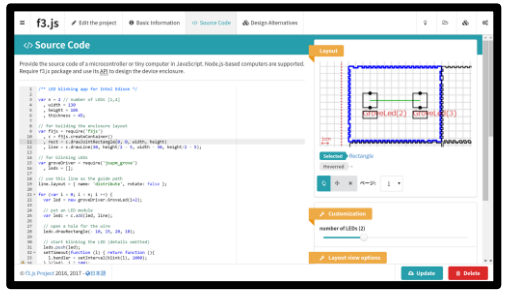
User Interfaces for Live Programming

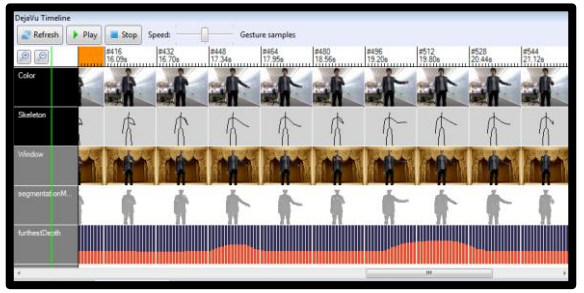
Jun Kato

https://junkato.jp

Researcher, AIST







LIVE 2017 Keynote, 10/24/2017

Jun Kato

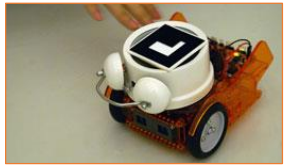
@ junkato

https://junkato.jp

Research Topic

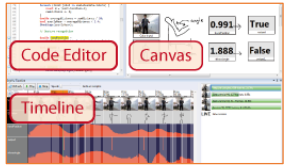
Computer Science (Human-Computer Interaction, Programming Language)

Phybots



ACM DIS'12

DejaVu



ACM UIST'12

Picode

ACM CHI'13

It's Alive!

ACM PLDI'13

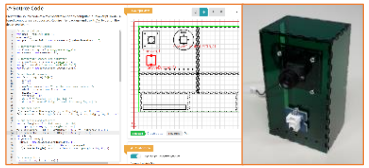
VisionSketch

GI'14

TextAlive

ACM CHI'15

f3.js



ACM DIS'17

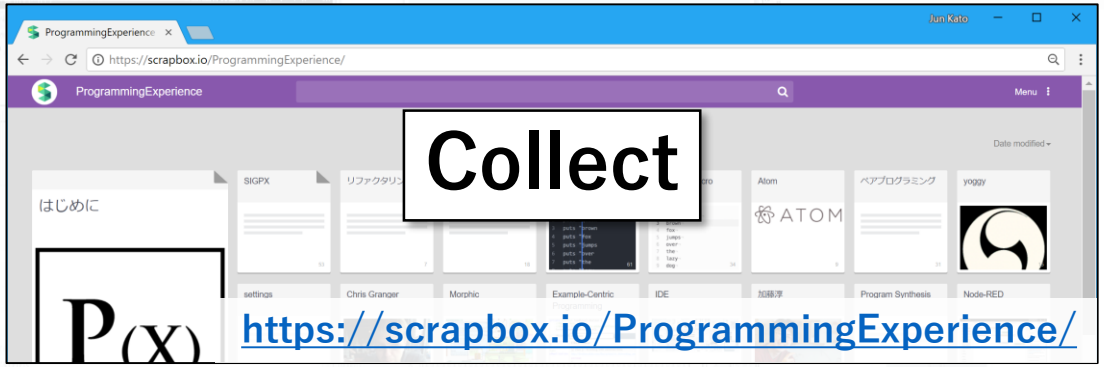
- Created **Tools and Environments** for **Creativity/Productivity Support**
- **Application Domains**: Prototyping, Physical Computing, Computer Vision, Robots, Internet of Things, Animation Authoring, …
- Founded **SIGPX** (SIG on Programming Experience) https://sigpx.org/en

SIGPX

https://sigpx.org/en

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

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



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

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

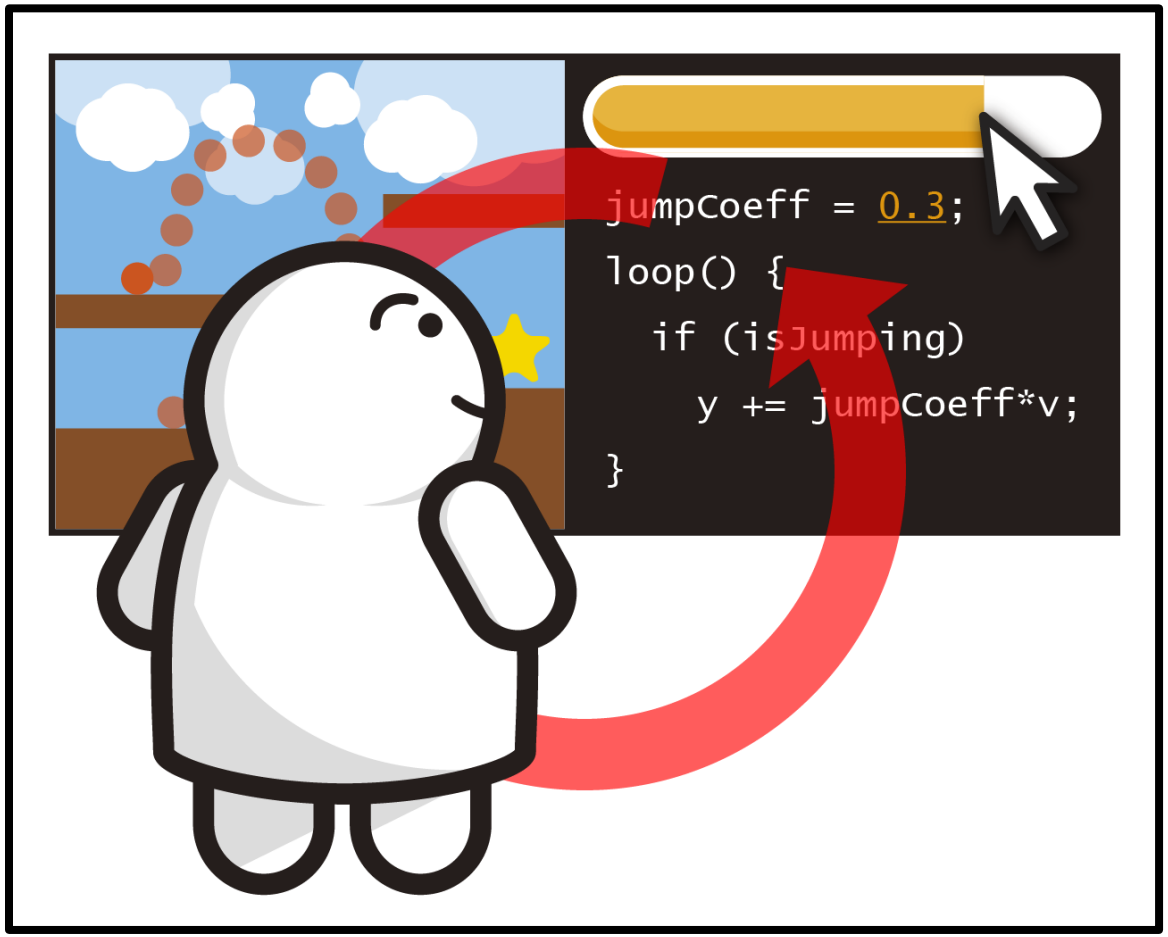
It is not about …

- No λ 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?



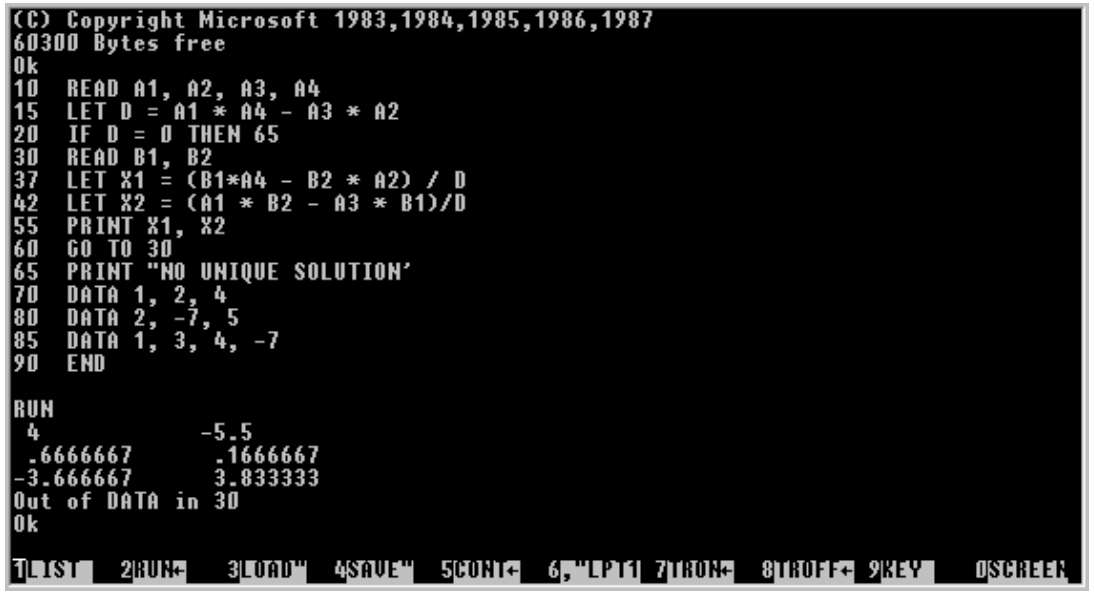
- Programming experience
- Continuous feedback
- Concrete information

- Early examples in VPL and OOP
- Attracting much attention these days

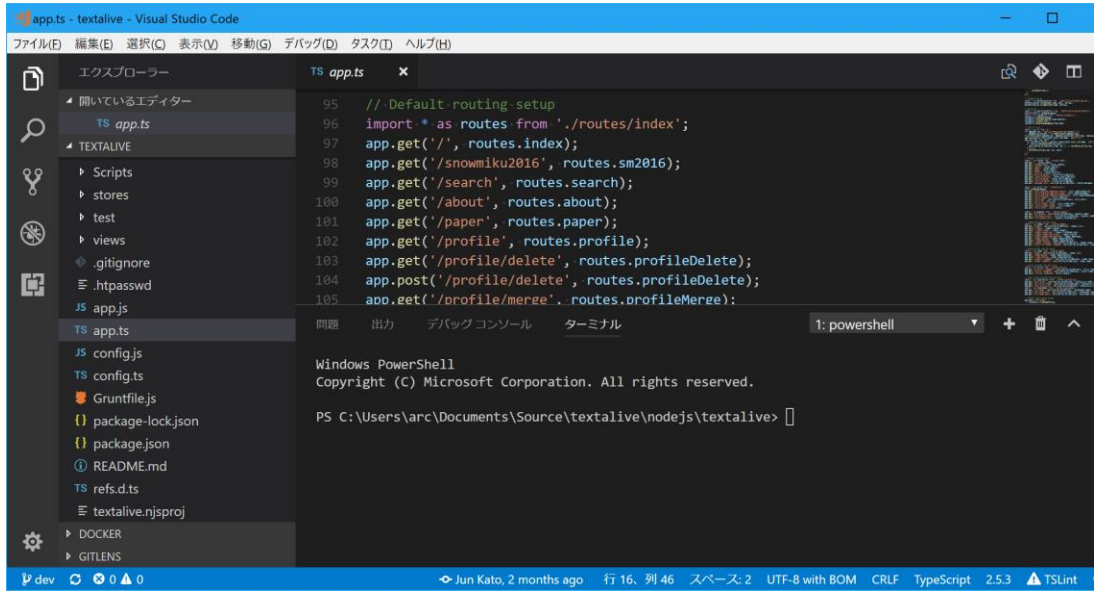
Not new but hot!

Text-based Programming

Dartmouth BASIC [1964]



Visual Studio Code [as of today]



- Text-based editor
- Text-based debugger
- Text-based …

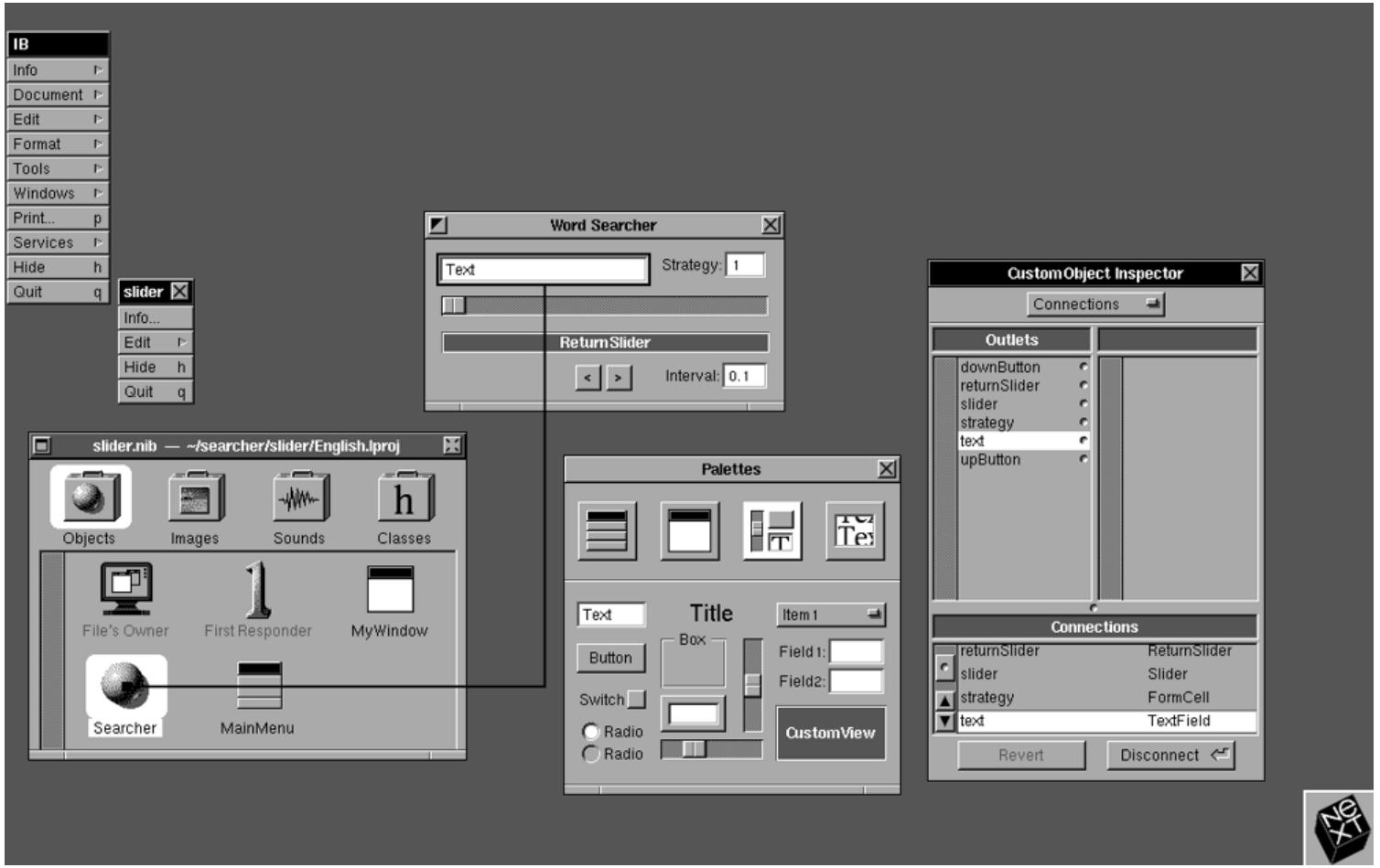
IDEs haven’t changed much

With some exceptions …

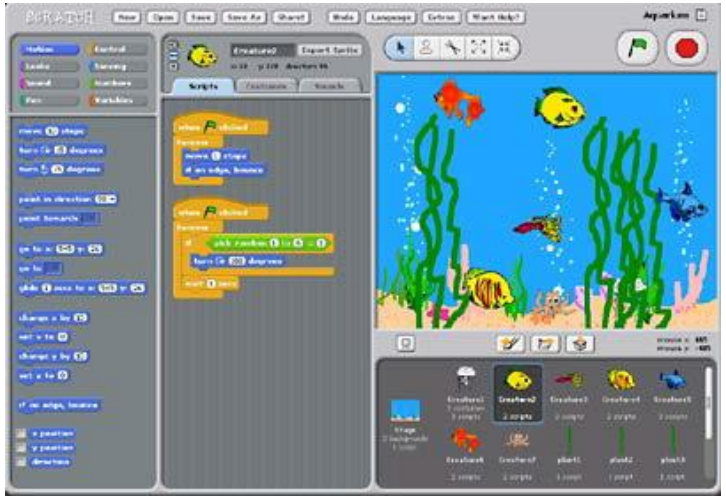
[since 1986]

Interface Builder

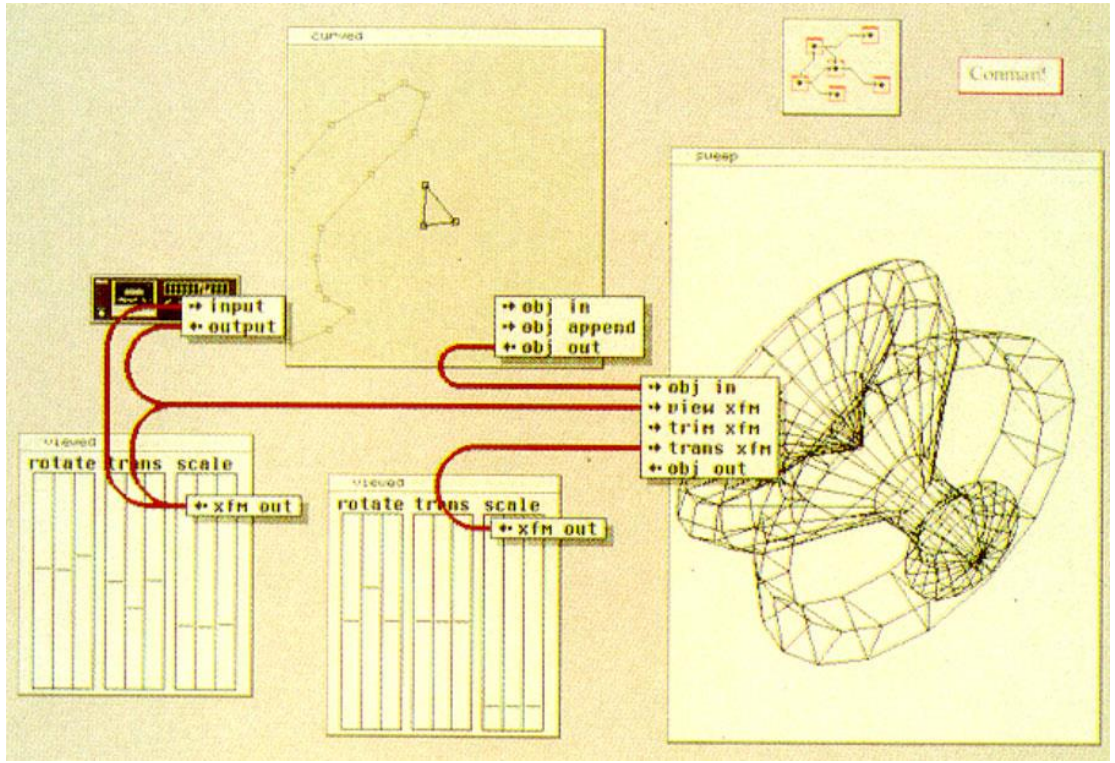
- A tool for NeXT UI development
- Later integrated into Xcode
- Many IDEs have similar built-in tools



Visual Programming



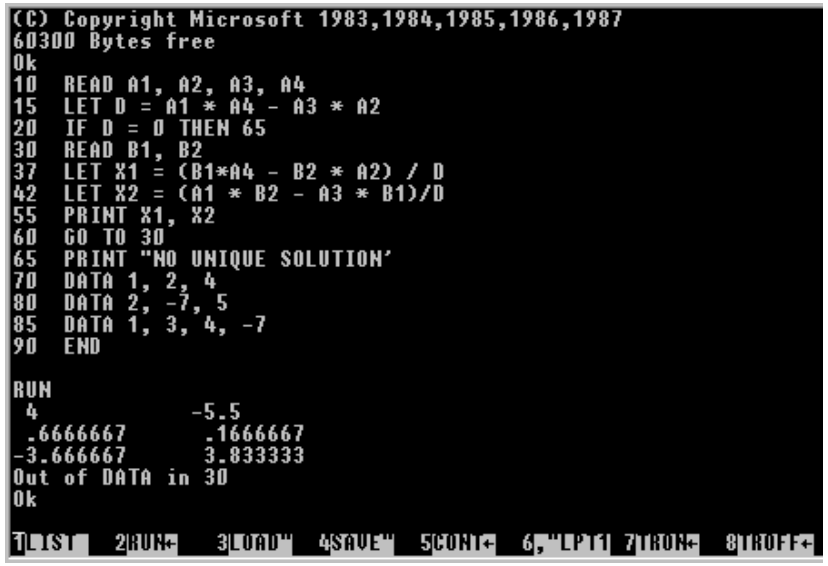
Scratch
MIT



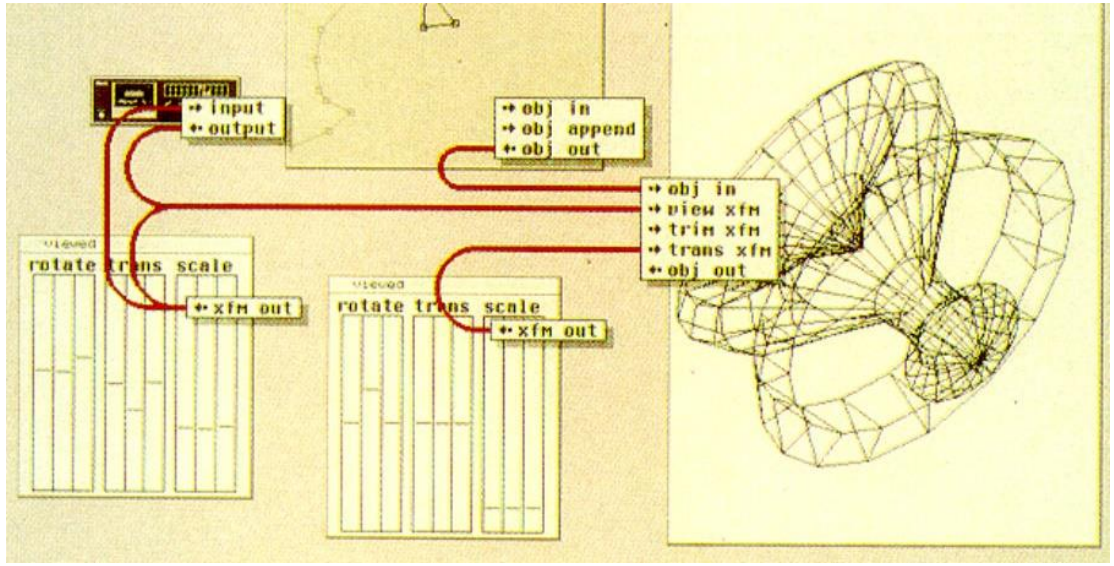
ConMan
Haberli
[SIGGRAPH 1988]

- 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?



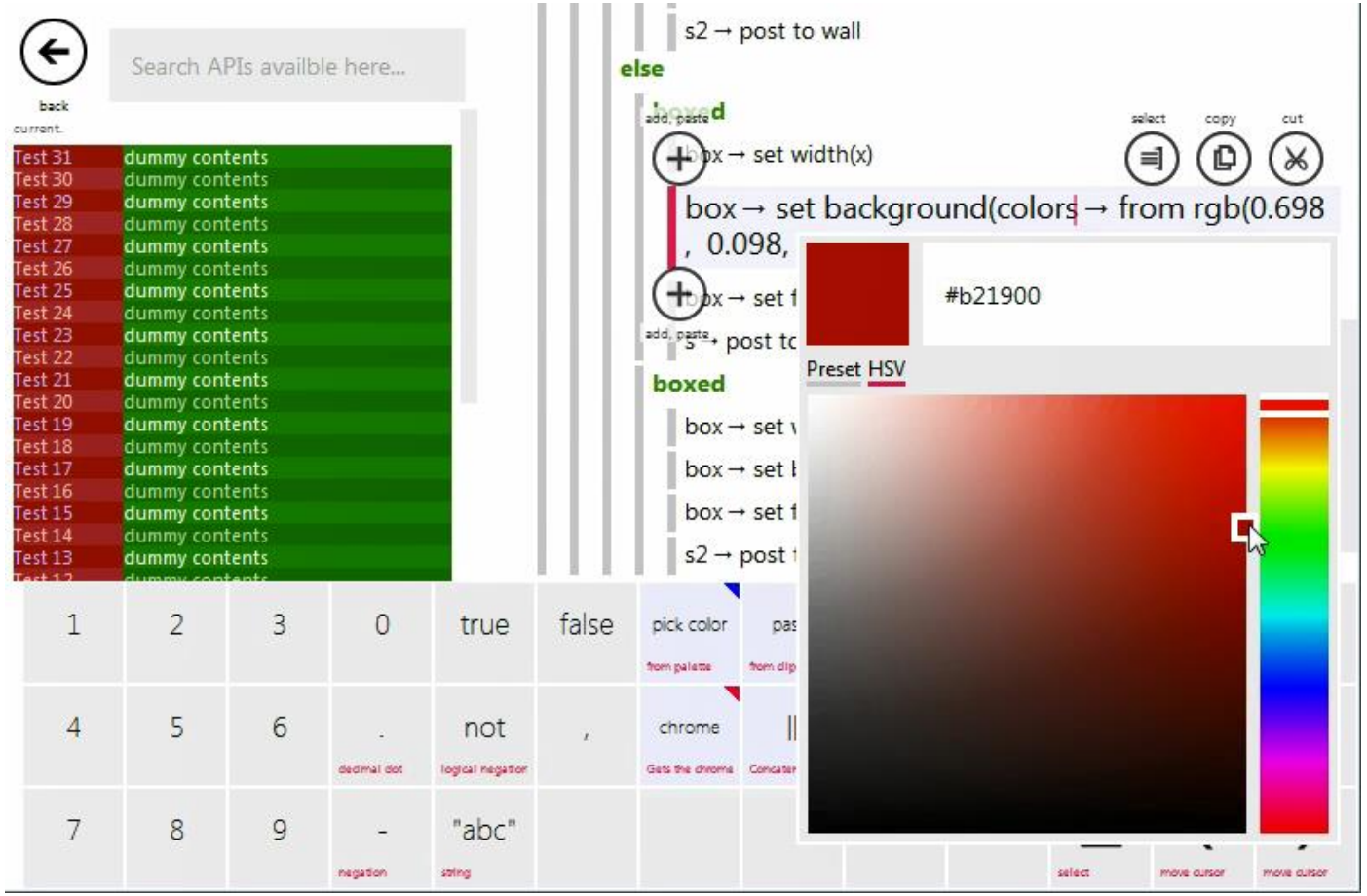
or



Modern

Live Programming as a hybrid approach

User Interfaces with Text **and** Graphics

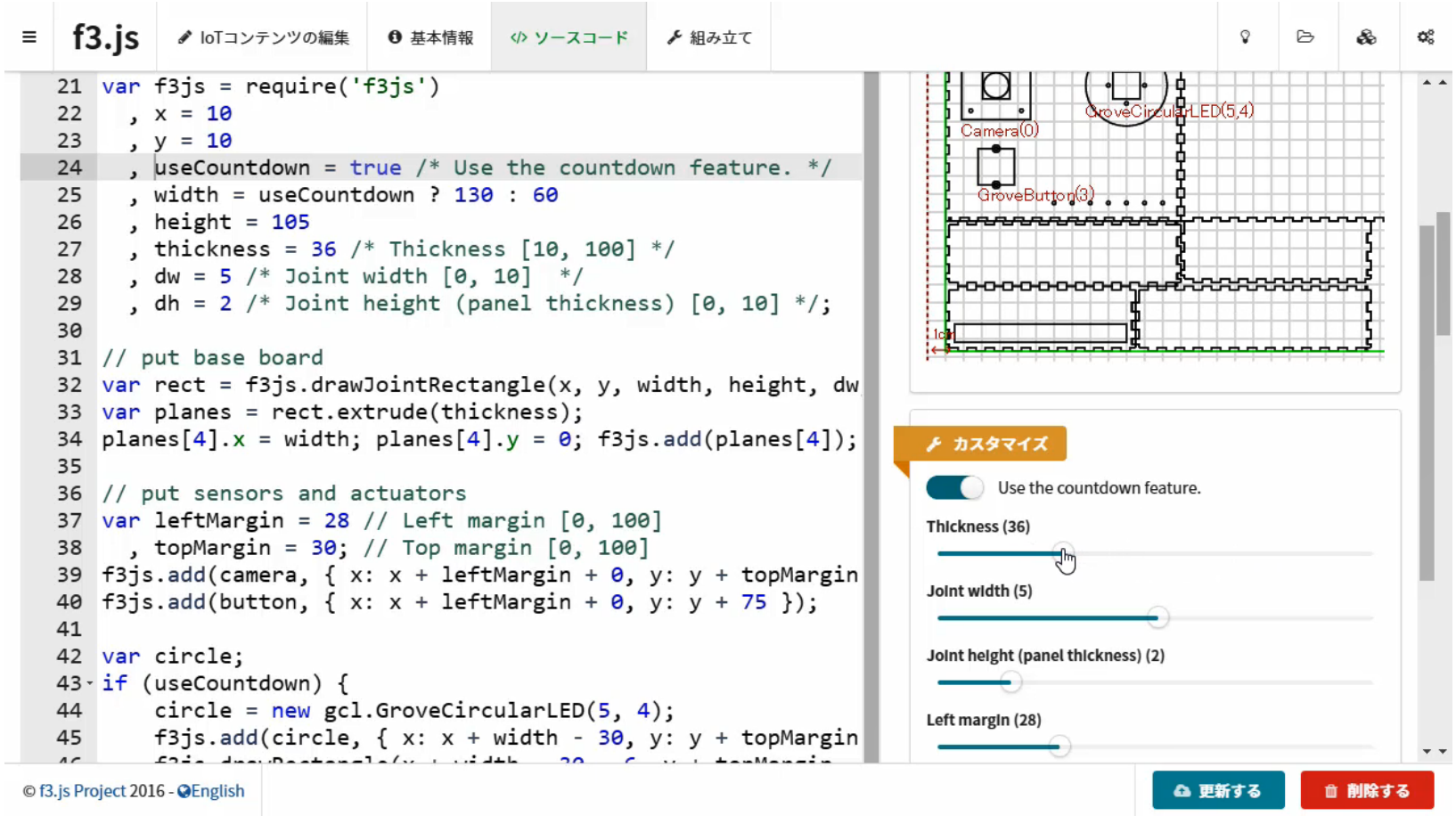


TouchDevelop

[PLDI 2014]

https://touchdevelop.com

User Interfaces with Text **and** Graphics



f3.js

[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 **Daisuke Sakamoto** **Takeo Igarashi**
The University of Tokyo, Tokyo, Japan – {jun.kato | d.sakamoto | takeo}@acm.org

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

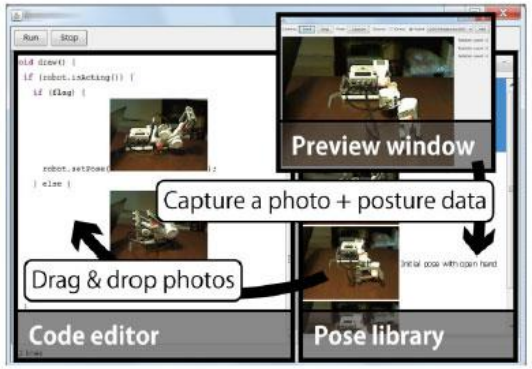


Figure 1. Overview of *Picode*

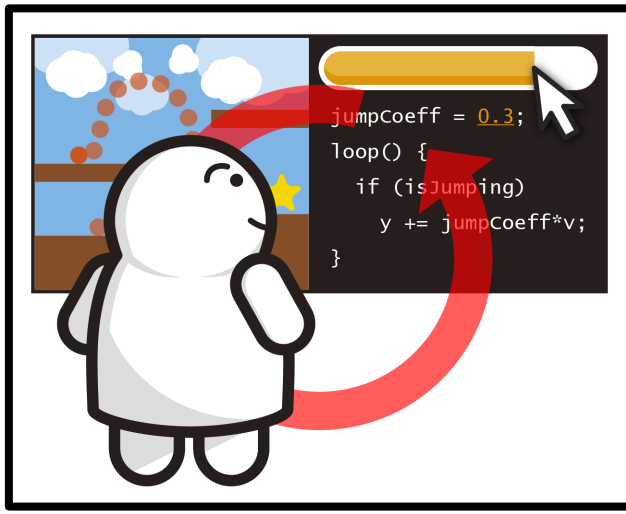
method by including an image that shows a concrete example of what the method does. Yeh et al. presented a development environment named "Sikuli," with which the programmer can take a screenshot of a GUI element and paste the image into a text editor [12]. In Sikuli, the image serves as an argument of 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

In Live Programming systems, we …

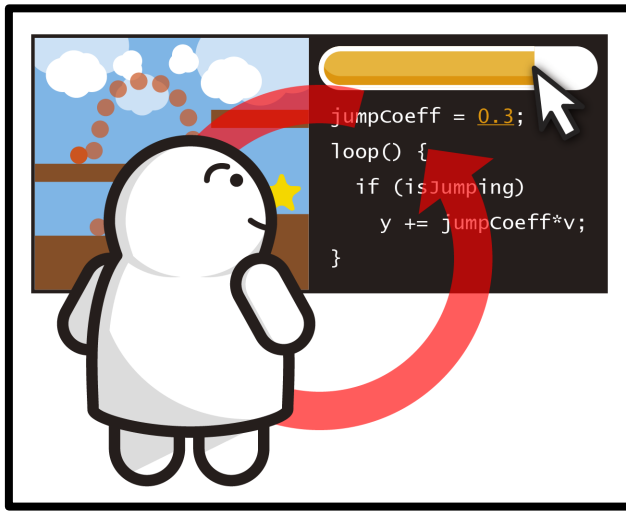
- first have **vague ideas**
- then explore the ideas with **concrete examples**
- gradually start turning the ideas into **programs**



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 …

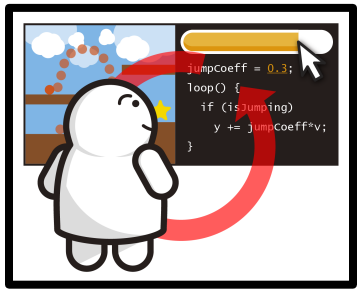
Don’t be afraid to be domain-specific

- 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

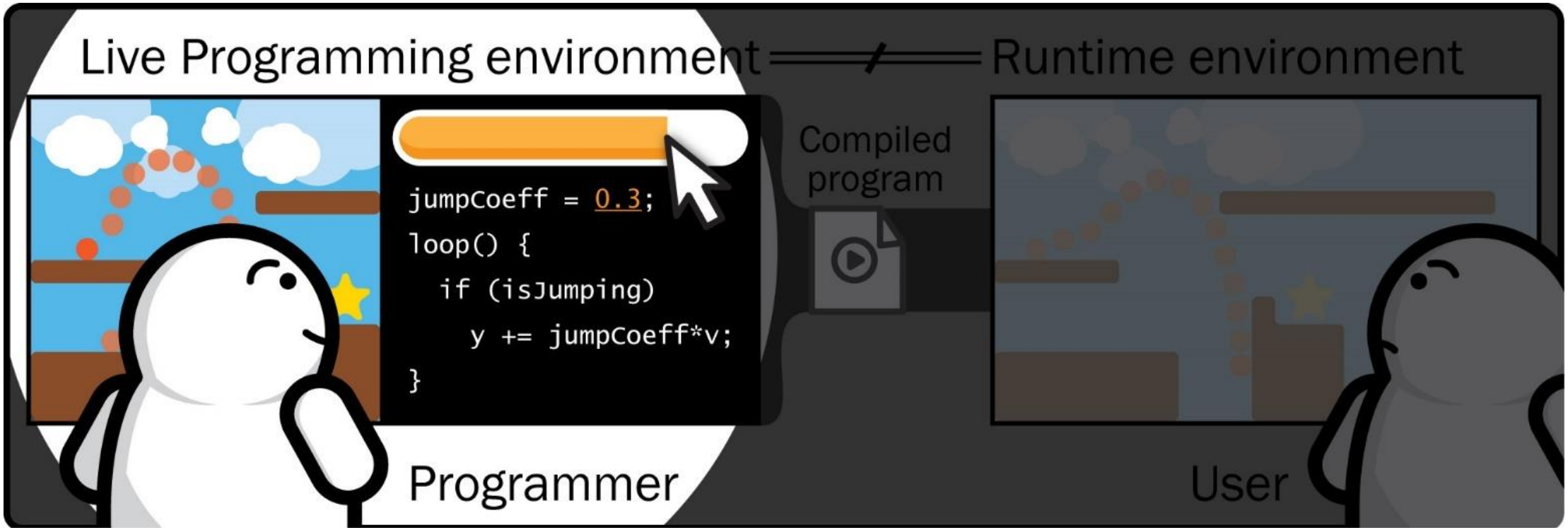
Today, I'm going to talk about …

- 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



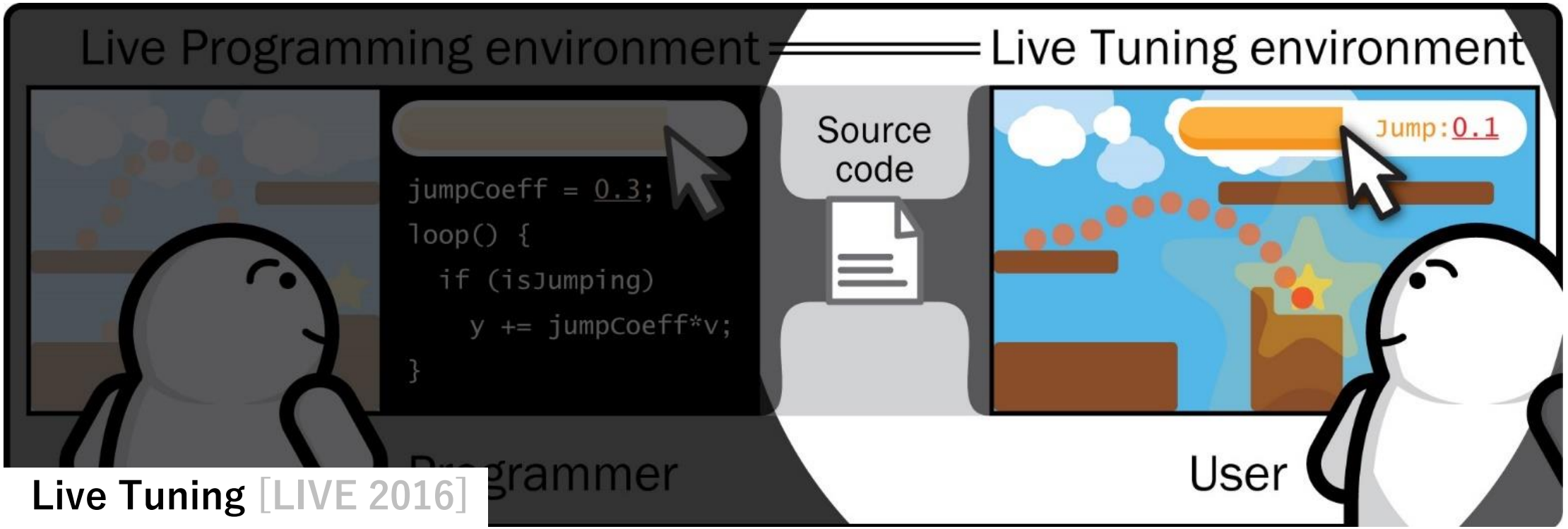
UIs for Live Programming

Good mixture of text-based and graphical user interfaces



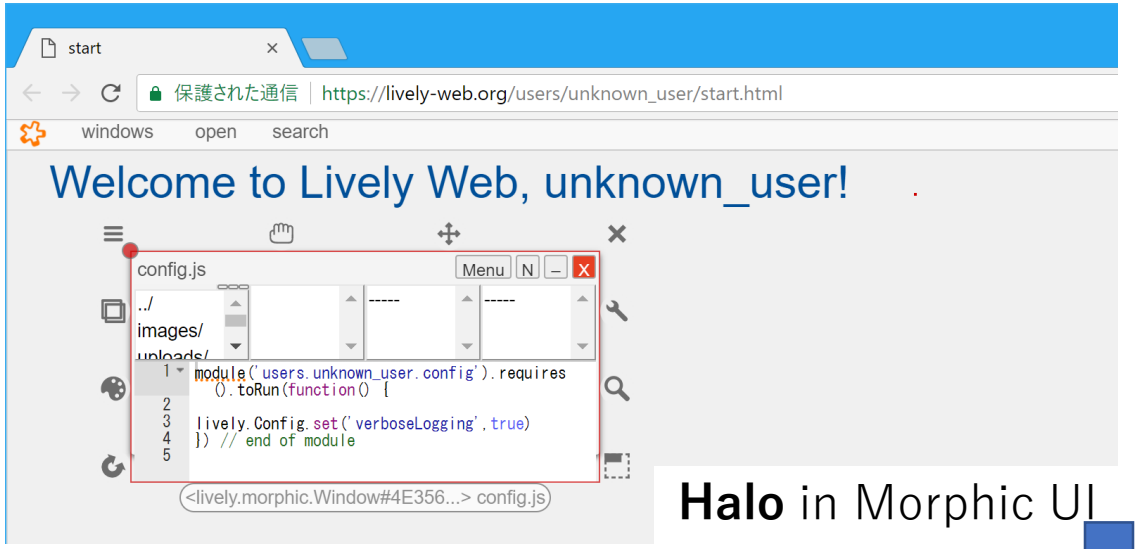
UIs for Live Programming

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

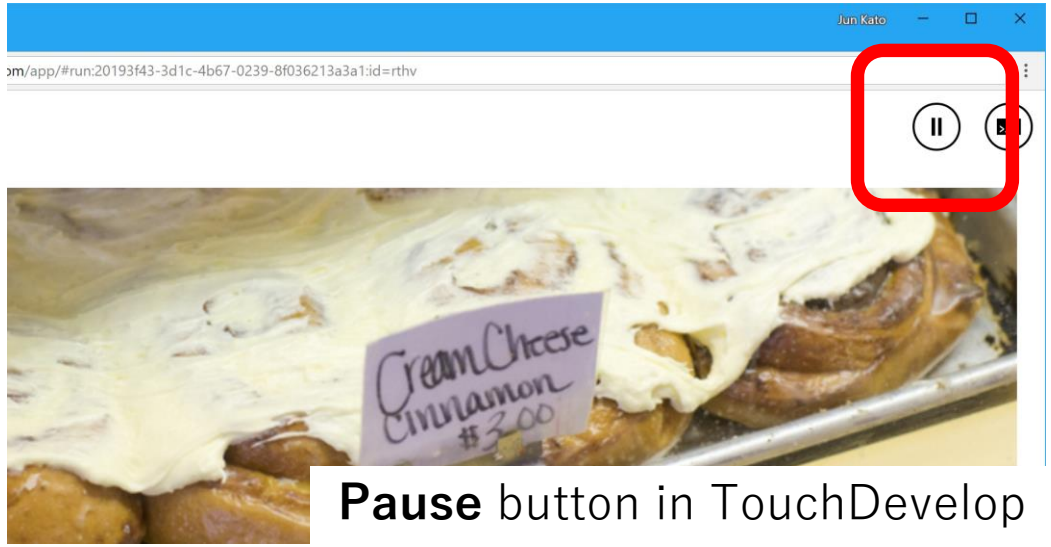


Live Tuning [LIVE 2016]

Mode Switch between “Use” and “Build”



Halo in Morphic UI

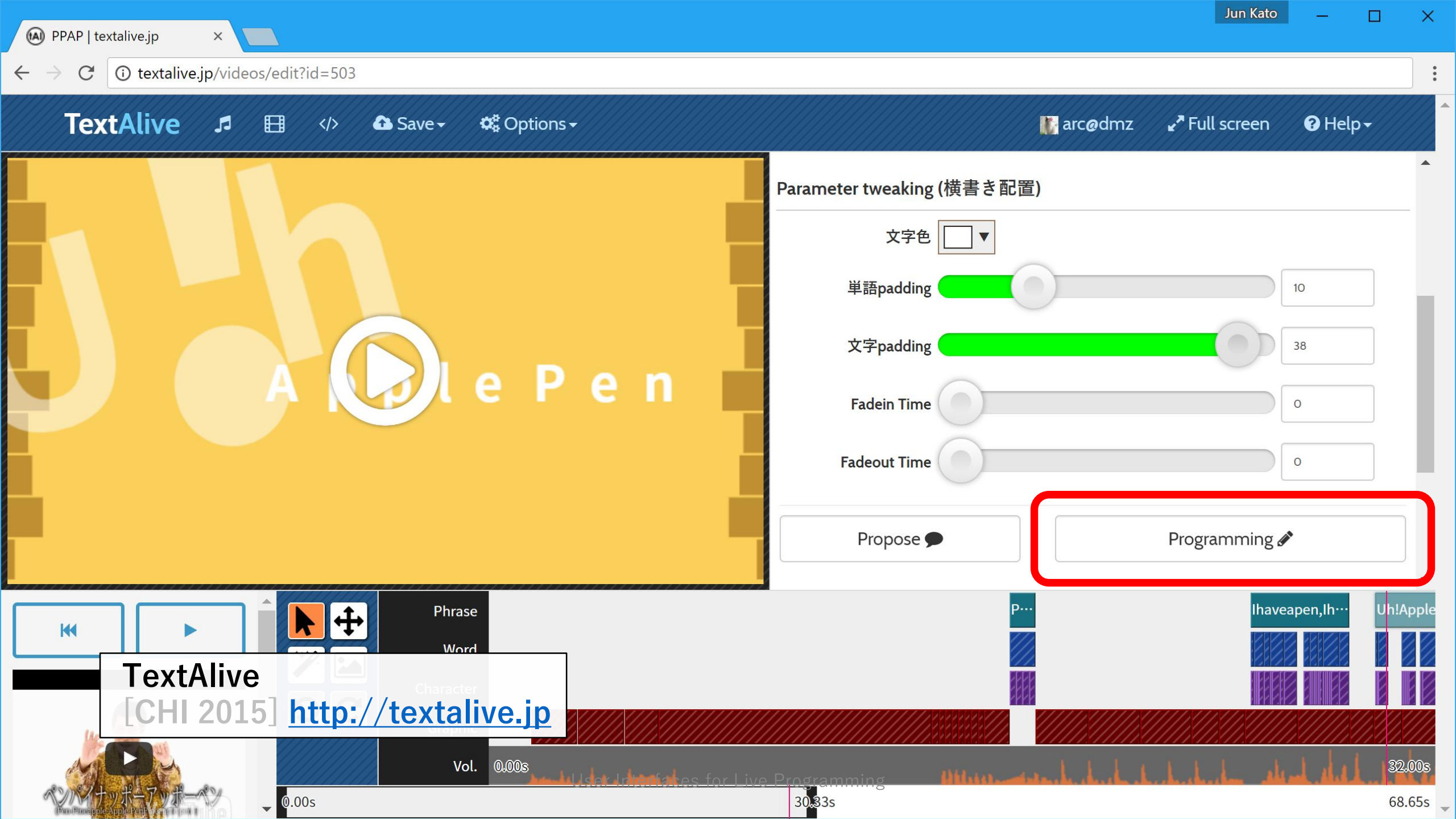


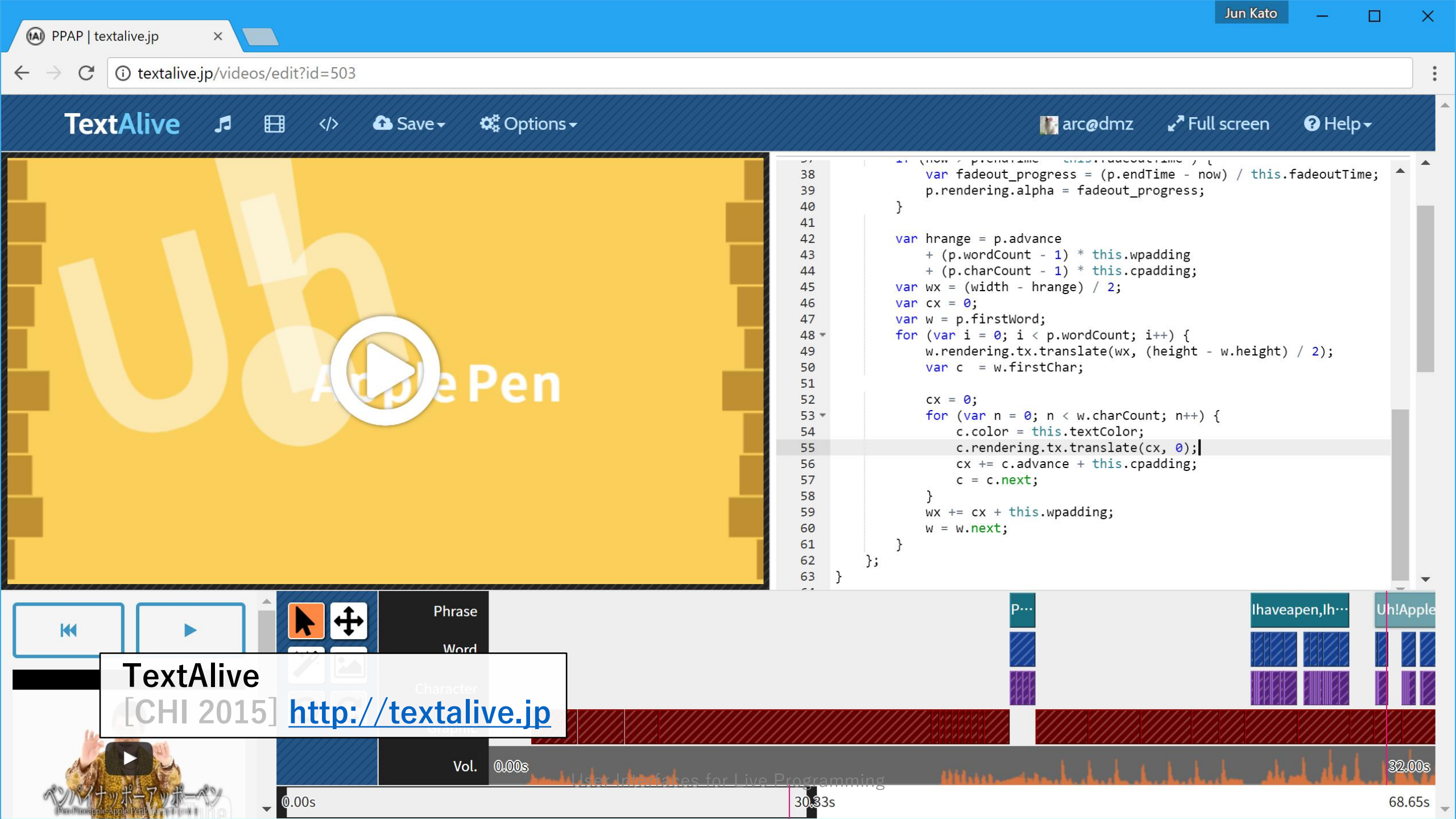
Pause button in TouchDevelop

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

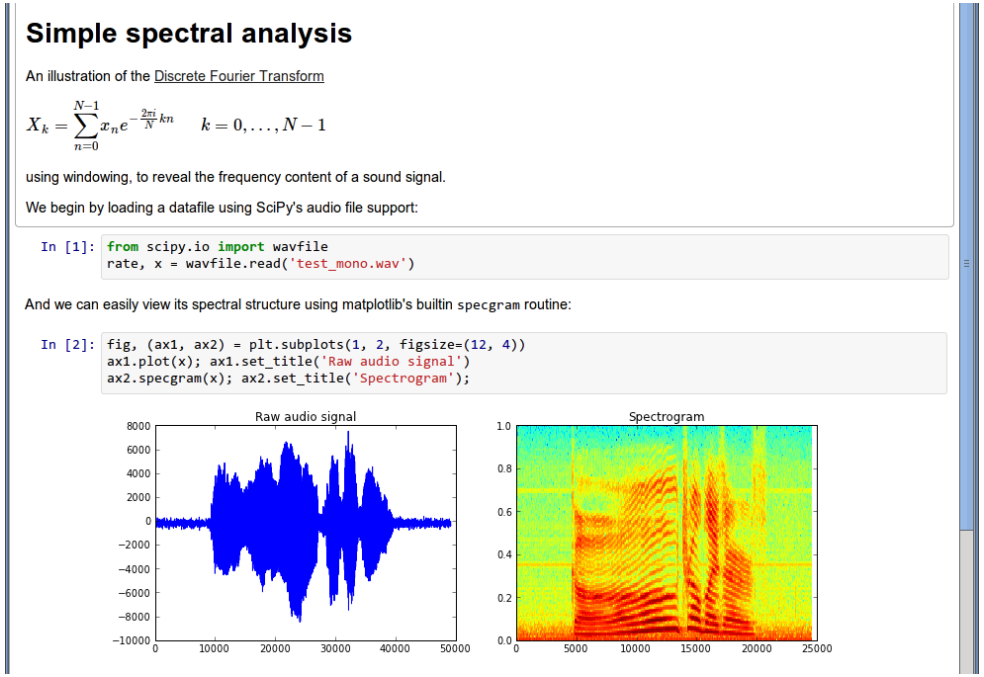
Promoting universal usability with multi-layer interface design
Ben Shneiderman [2002]


Jun Kato
PPAP | textalive.jp
textalive.jp/videos/503
TextAlive
Songs
Videos
Templates
Edit
Login
Full screen
Help
Pen
TextAlive
[CHI 2015] http://textalive.jp
User Interfaces for Live Programming

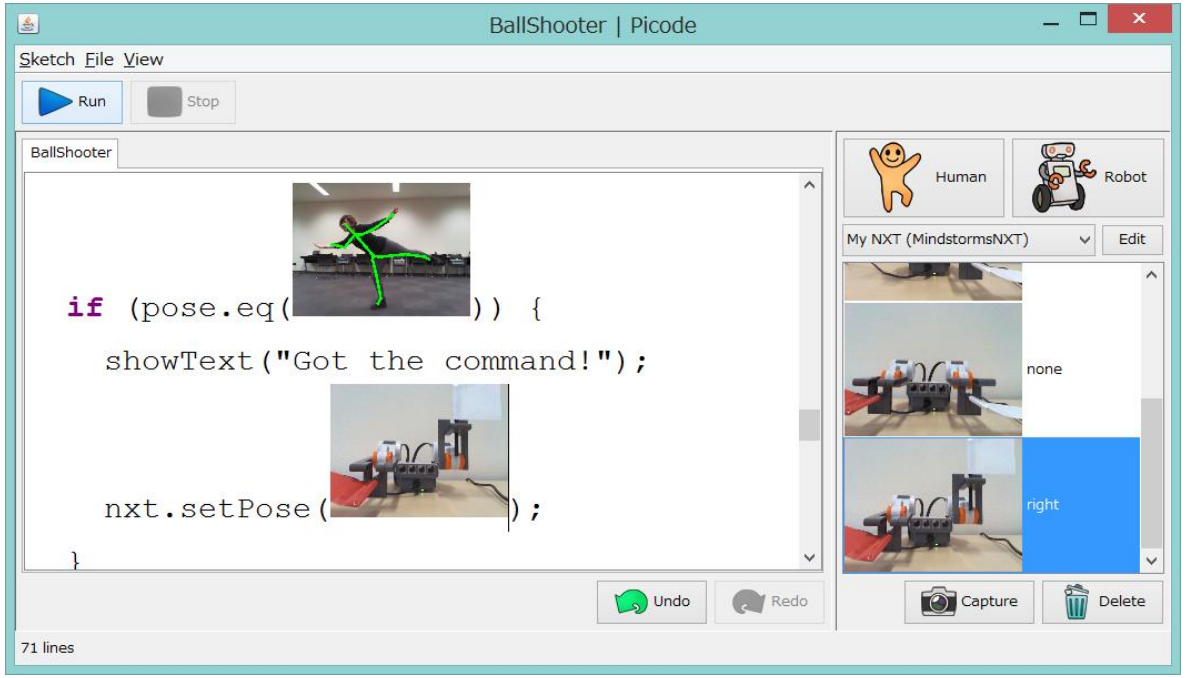

Jun Kato
PPAP | textalive.jp
textalive.jp/videos/edit?id=503
TextAlive
Save
Options
arc@dmz
Full screen
Help
Parameter tweaking (横書き配置)
文字色
単語padding
10
文字padding
38
Fadein Time
0
Fadeout Time
0
Propose
Programming
Phrase
Word
Character
Vol.
0.00s
32.00s
0.00s
30.33s
68.65s
TextAlive
[CHI 2015] http://textalive.jp
User Interfaces for Live Programming


Jun Kato
PPAP | textalive.jp
textalive.jp/videos/edit?id=503
TextAlive
Save
Options
arc@dmz
Full screen
Help
Uh Apple Pen
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
Vol.
P···
Ihaveapen,Ih···
Uh!Apple
0.00s
32.00s
0.00s
30.33s
68.65s
TextAlive
[CHI 2015] http://textalive.jp
User Interfaces for Live Programming

Co-hosting UIs for programmers and users

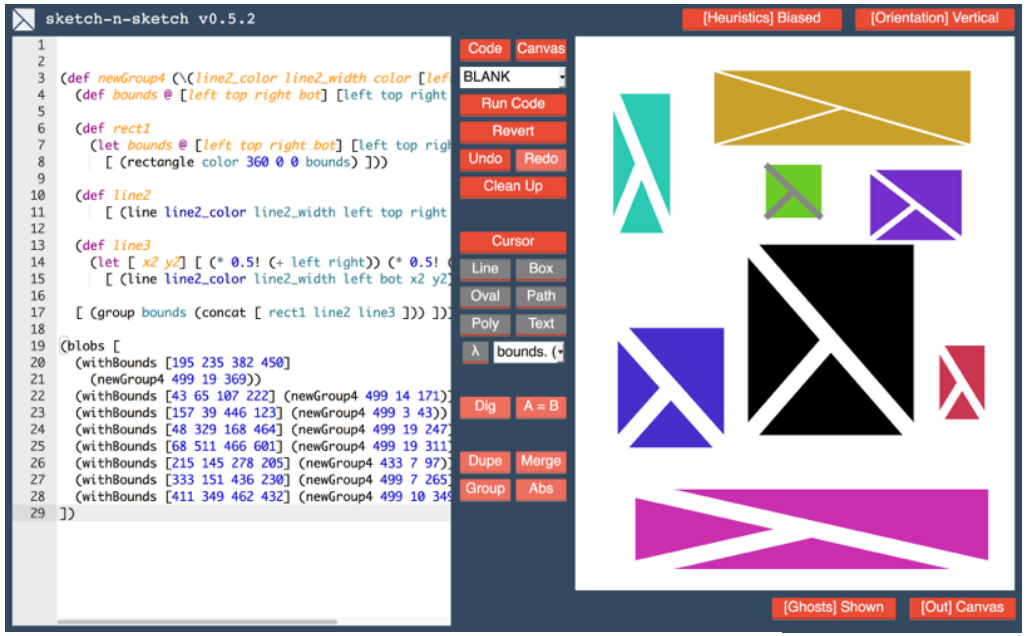


Literate Programming
in Jupyter (Ipython) Notebook

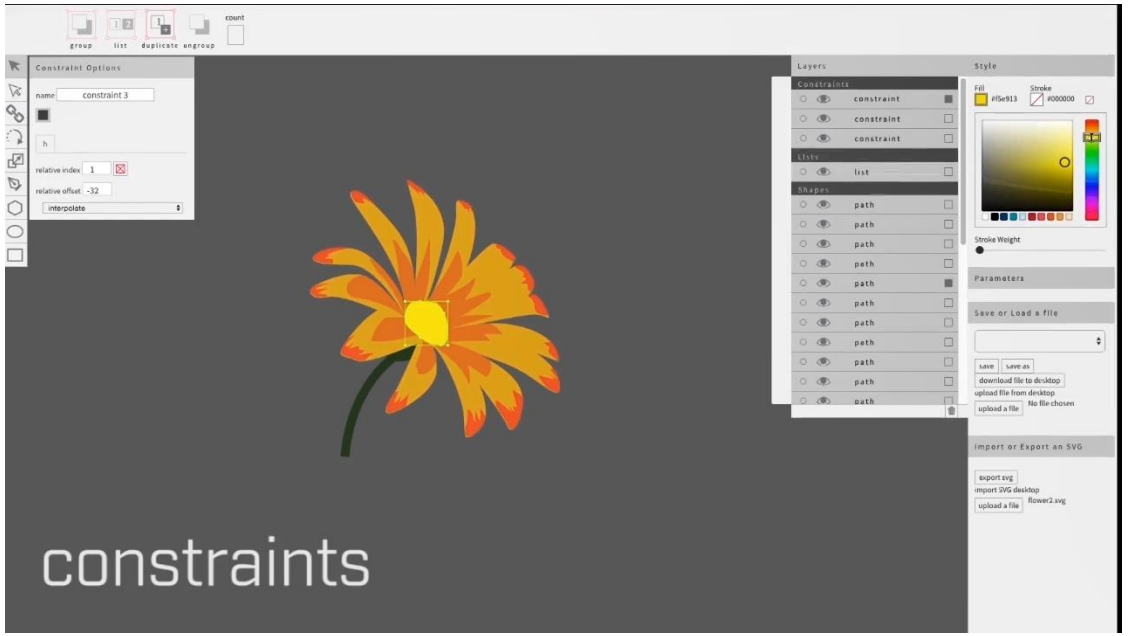


Inline Photos
in Picode [CHI 2013]

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



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

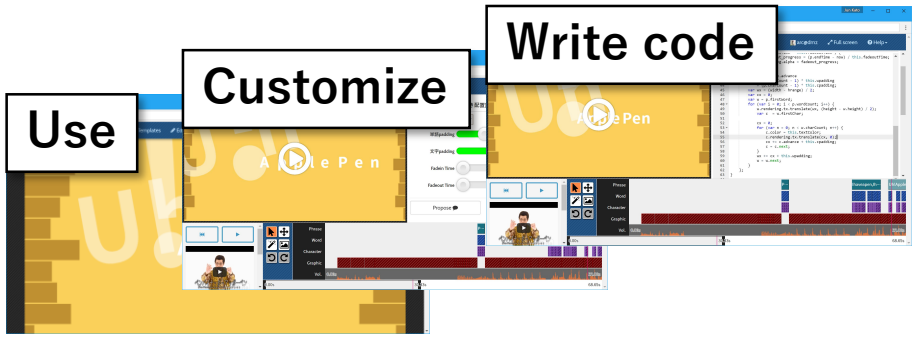


Para, Jacobs et al.
[CHI 2017]

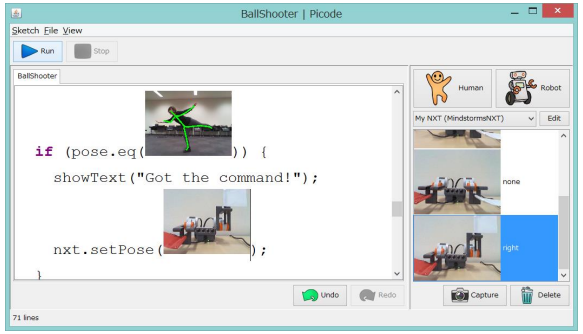
When designing live programming systems …

How about making the ladder of expertise?

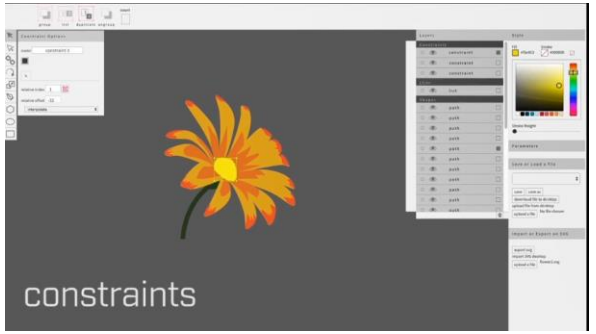
- From live programming for programmers
- To live programming for the community of people



Multi-layer



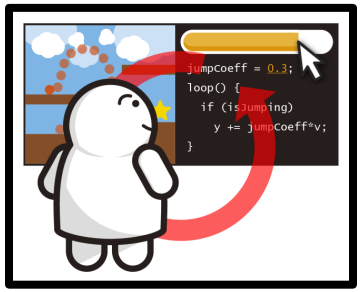
Co-host



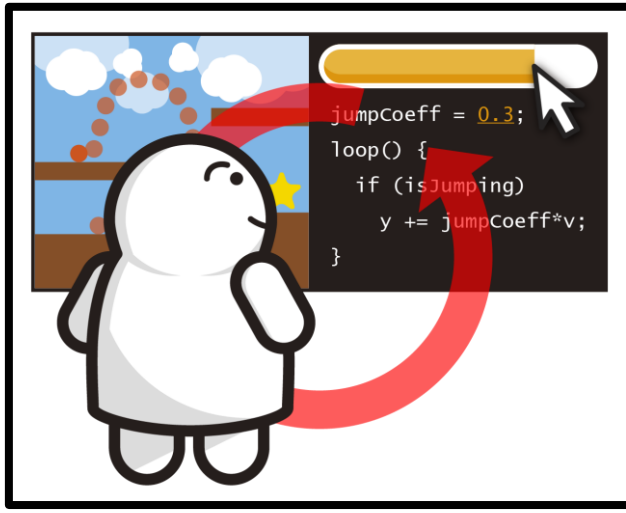
Merge

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

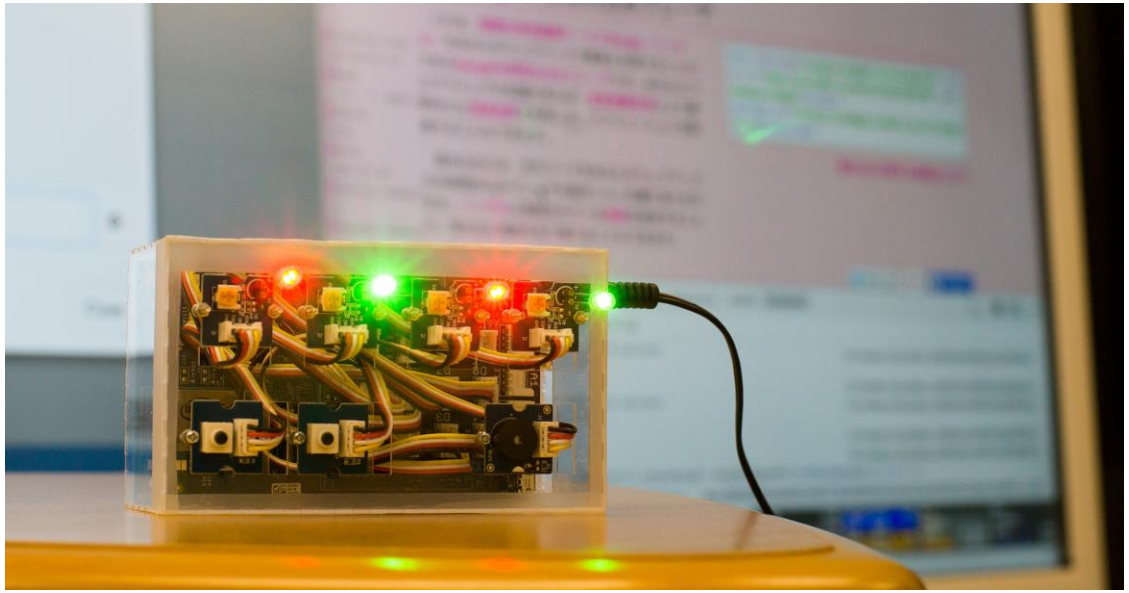
How Live are Live Programming Systems?
Rein et al. [PX 2016]

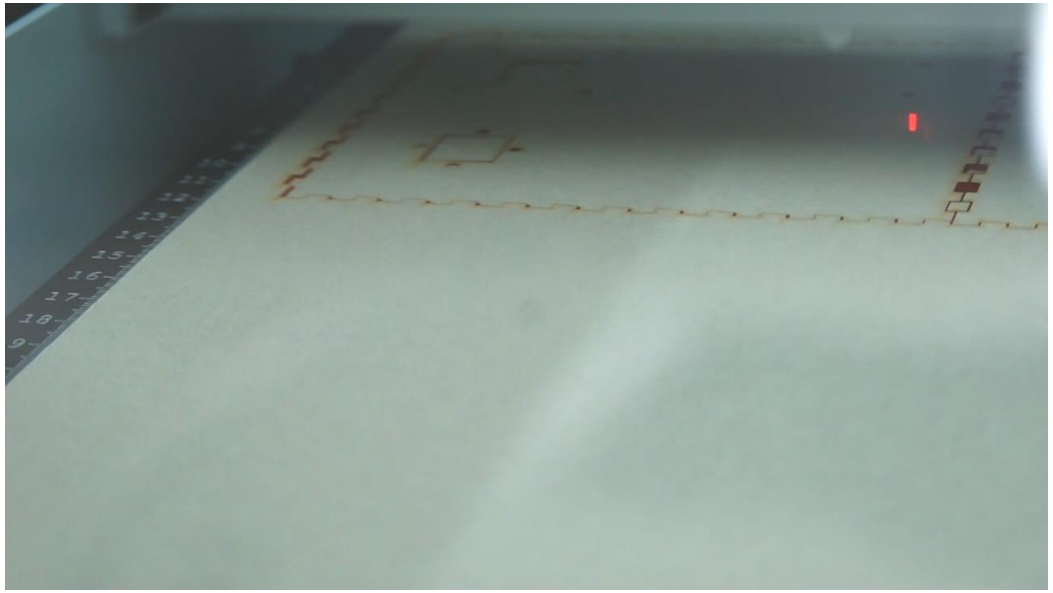
- Reflex time:
  - Visual 0.25s
  - Audio 0.17s
  - Touch 0.15s
- and more …
  - 3D printers and laser cutters
  - Shape changing devices and robots
  - Taste/smell interfaces

Various kinds of “latencies”

Printing physical computing devices

f3.js [DIS 2017]

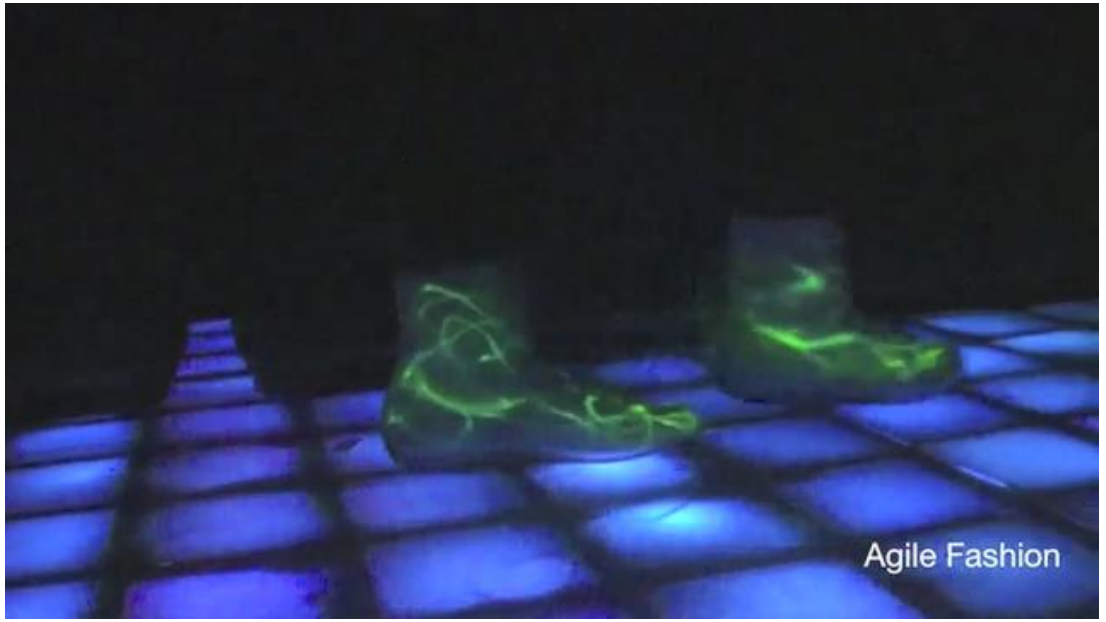




Slow “framerates” prevent live feedback

Slow display

Daniel Saakes et al. [SIGGRAPH Etech 2010]



Slow “framerates” can be useful, too

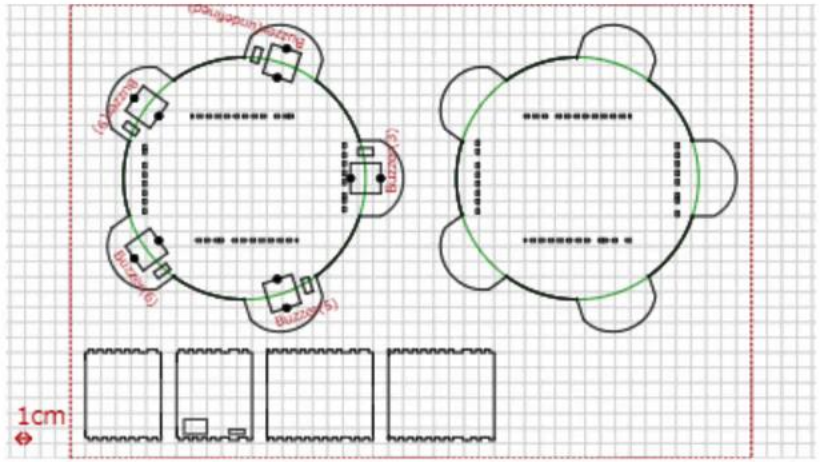
Source Code

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.

```
var WebSocket = require('ws');
var serverAddr = 'ws://192.168.10.100:8080/entry';
var ws = new WebSocket(serverAddr);
var upmBuzzer = require('jsupm_buzzer');

var numBuzzers = 5; // Number of buzzers [1, 5]
var inputs = [3, 5, 6, 9];
var buzzers = [];
var volume = 0.1;
var r = 80;

var f3js = require('f3js');
var c = f3js.createContainer();
c.x = 50;
c.y = 50;

var c1 = 43
  , c2 = 36
  , x = 70
  , y = 140
  , width = 70;
var a = c.createPath();

var ps = a.extrude(60);
ps[0].y = 50;
ps[0].x = 270;
f3js.add(ps[0]);

function addPetal(layout, ...
    var cc = f3js.createContainer();
    var pp = cc.createPath()
```

f3.js: IoT apps with enclosures from single code base
[DIS 2017] http://f3js.org

Layout



Selected -

Hoverred -

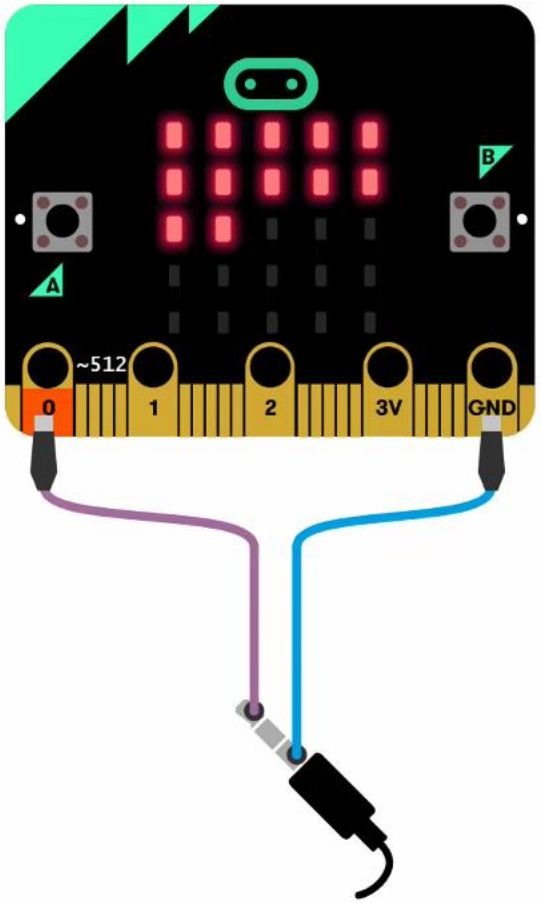
ページ: 1

Customization

Number of buzzers (5)

Layout view options

Update | Delete



micro:bit Projects Share Blocks JavaScript Microsoft

Search...

- Basic
- Input
- Music
- Led
- Radio
- Loops
- Logic
- Variables
- Math
- Advanced

Getting Started

```
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

BallShooter | Picode

Picode: inline photos representing posture data in source code [CHI 2013]

Sketch

BallShooter

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

Human | Robot

My NXT (MindstormsNXT) | Edit

none

right

Undo | Redo | Capture | Delete

71 lines

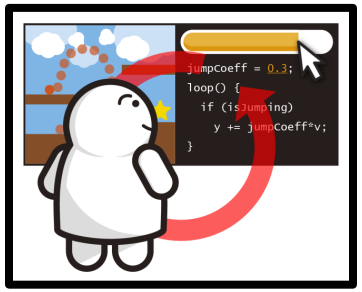
When designing live programming systems …

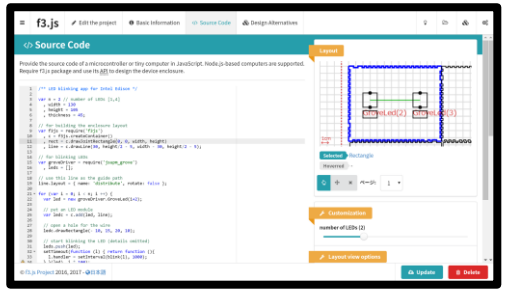
Deceiving users' perception is a good thing

- 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 …

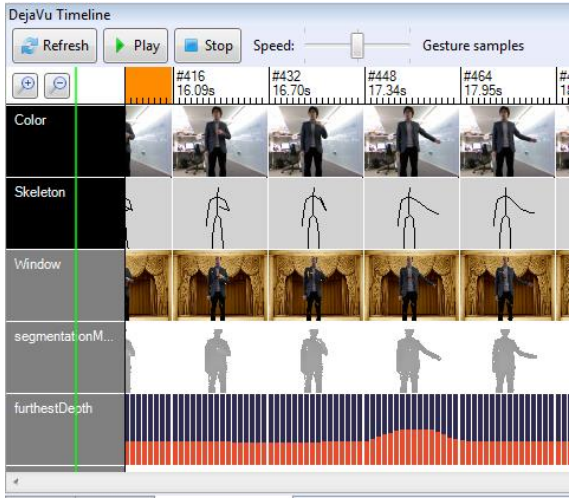
- What is Live Programming?
- UIs for Live Programming with end-users
- UIs for Live Programming of this material world
- **UIs for Live Programming with time travel**
- Live Programming as User Interface research





“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

KinectDress.Window1 | OnKinectFramesReady(object sender, KinectI

```
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

KinectDress

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 | #22 8.8

Color

Window

KinectDress

userDistance

DejaVu
[UIST 2012]

User Interfaces for Live Programming

Session 17, 580 frames, 23.00s

LIVE New session

Window1.xaml.cs

KinectDress.Window1 | OnKinectFramesReady(object sender, KinectI

```
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
```

DejaVu Canvas

Color | Depth | Skeleton | Window

Color input

Window output

userDistance

Properties | Classes | DejaVu Canvas

DejaVu Timeline

Refresh | Play | Stop | Speed: | Session 17

#64 2.65s | #128 5.15s | #192 7.61s | #256 10.14s | #320 12.63s | #384 15.16s | #448 17.82s

Color

Window

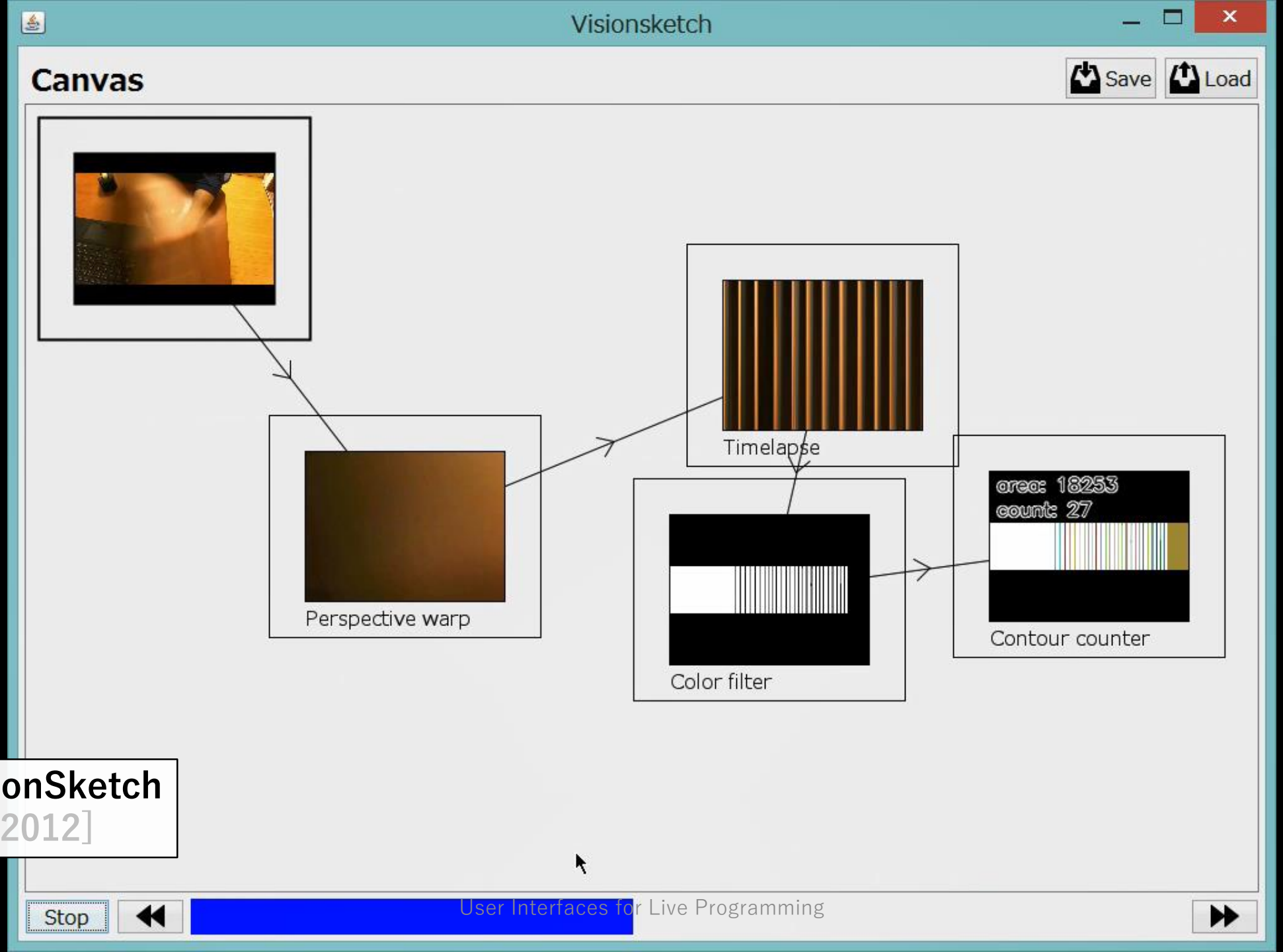
KinectDress KinectDress

userDistance

DejaVu
[UIST 2012]

Session 17, 580 frames, 23.00s

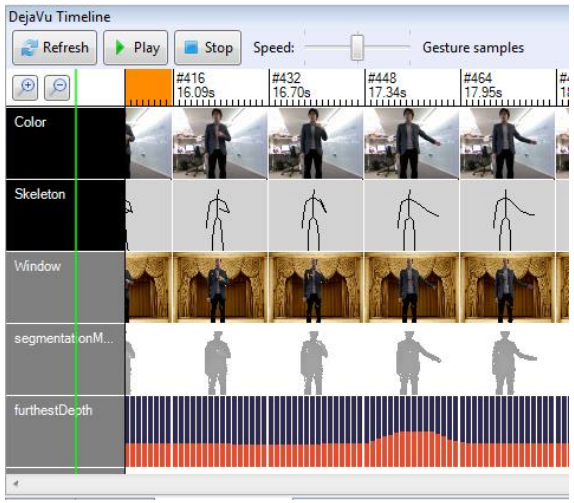
LIVE New session

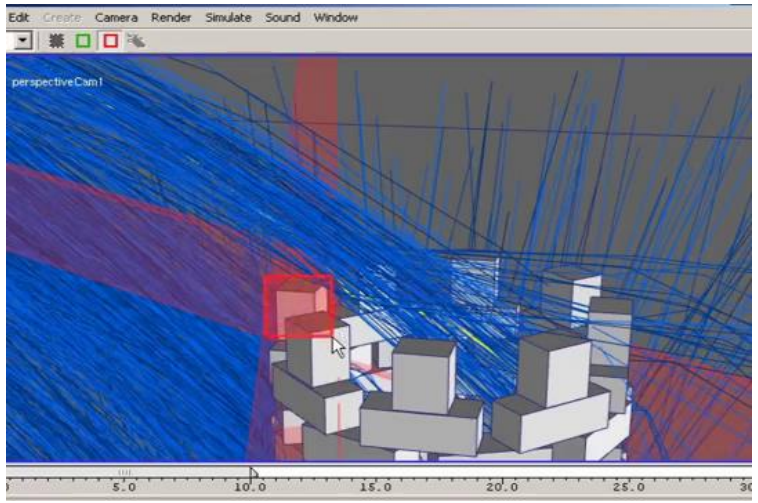


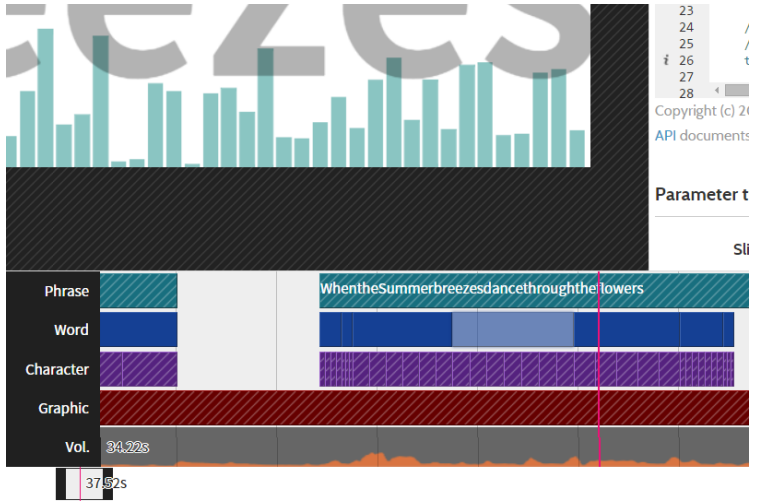
VisionSketch
[GI 2012]

“Live” is not always about “now”

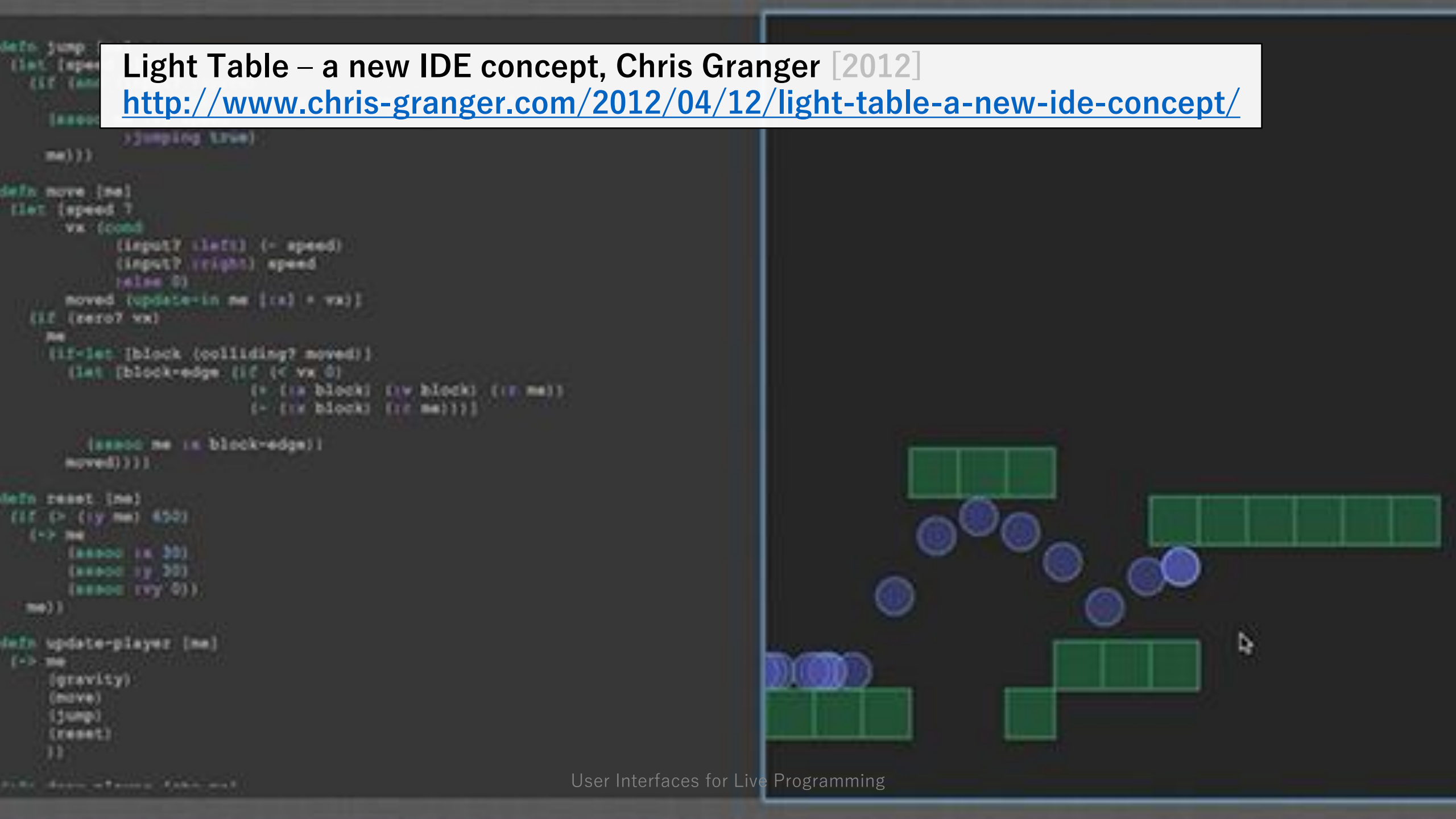
- UIs for exploring and modifying the past
- **UIs for predicting and choosing the future**
- Absolute vs semantic timeline

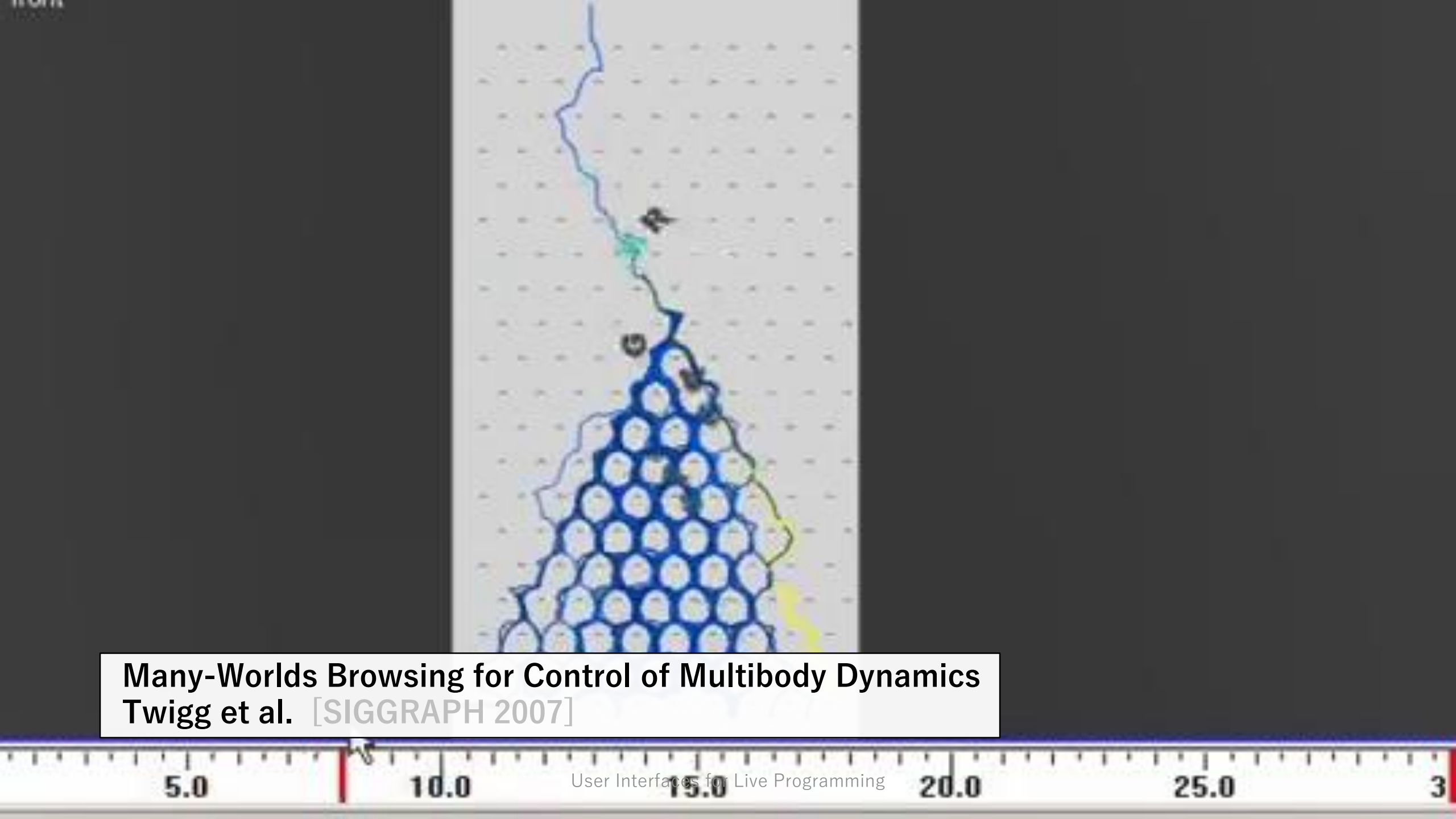






Light Table – a new IDE concept, Chris Granger [2012]
http://www.chris-granger.com/2012/04/12/light-table-a-new-ide-concept/



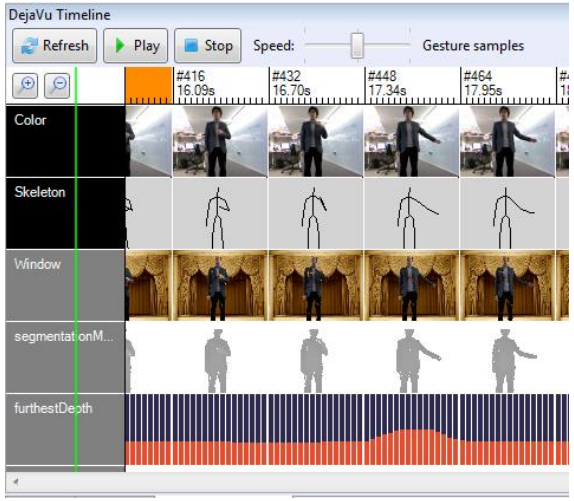


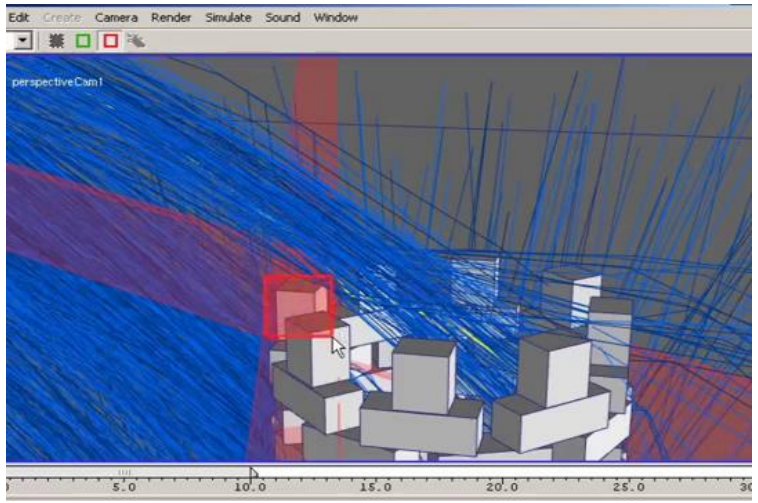
Many-Worlds Browsing for Control of Multibody Dynamics
Twigg et al. [SIGGRAPH 2007]

Rktcr, McCann (TCHOW)
[2013]

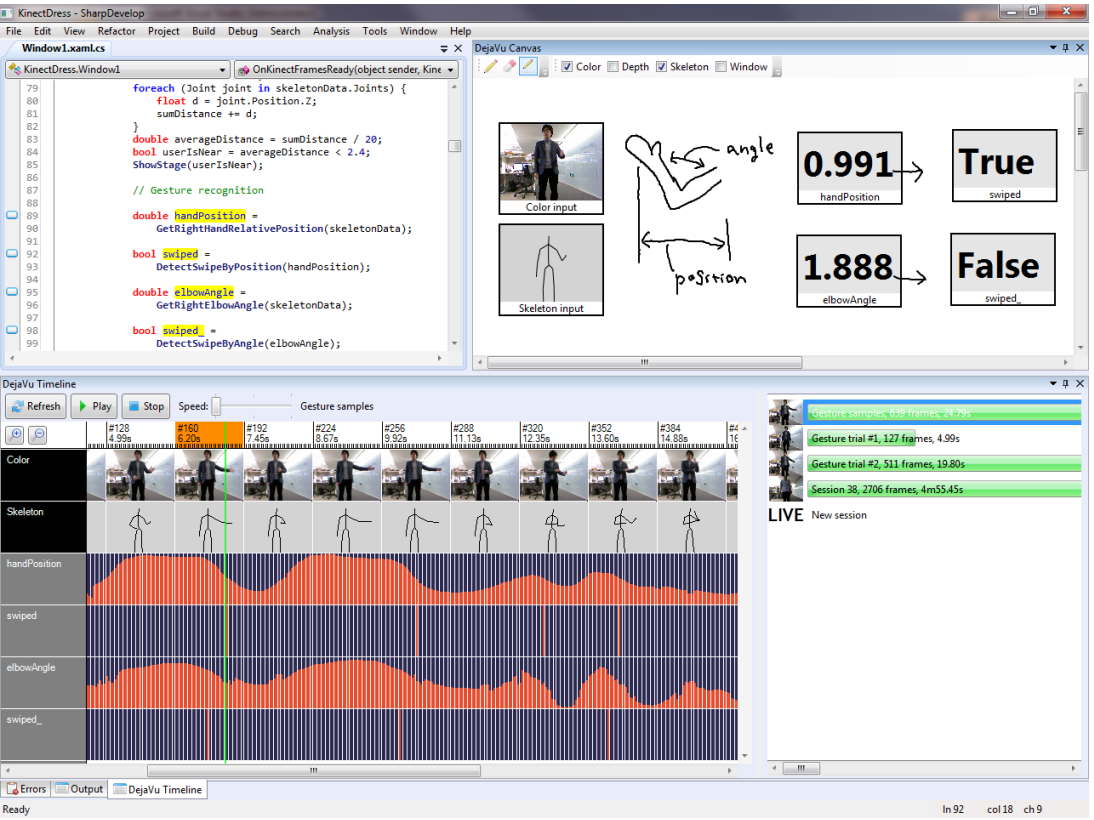
“Live” is not always about “now”

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

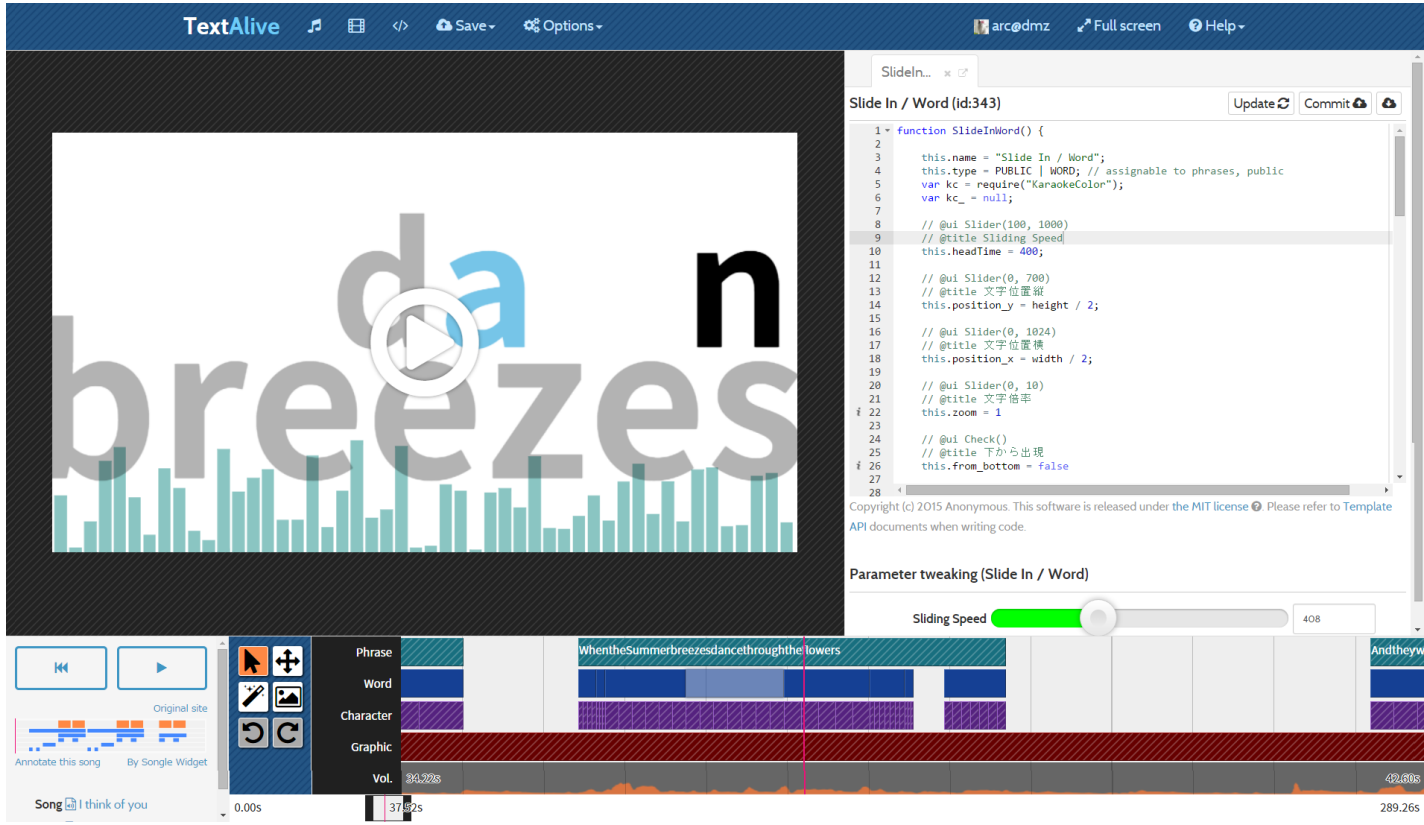




Absolute time vs semantic time



DejaVu [UIST 2012]

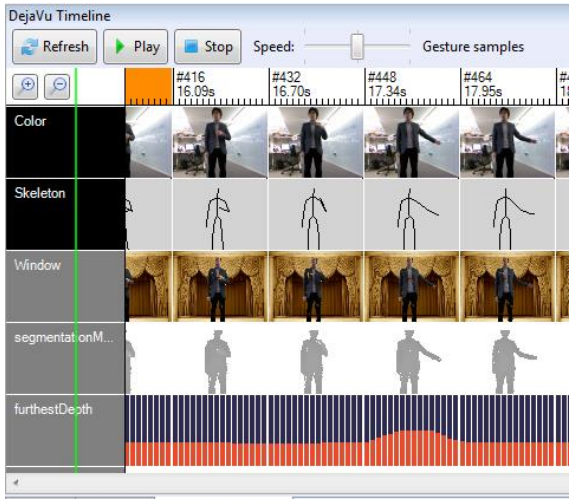


TextAlive [CHI 2015]

When designing live programming systems …

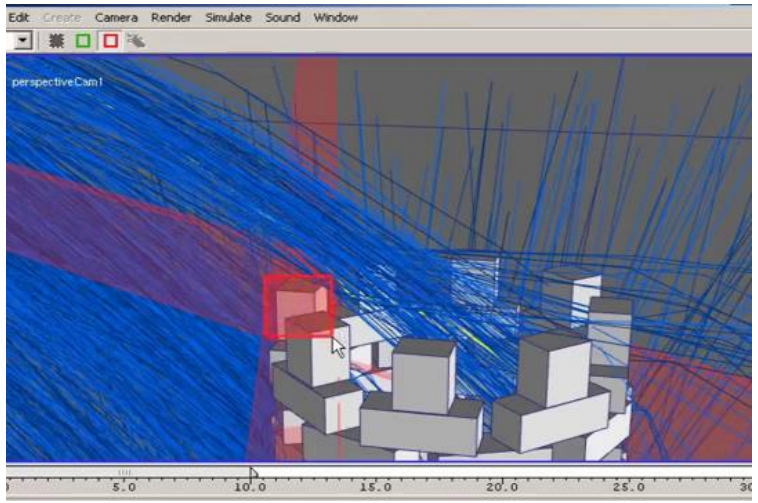
Try providing good sense of time

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



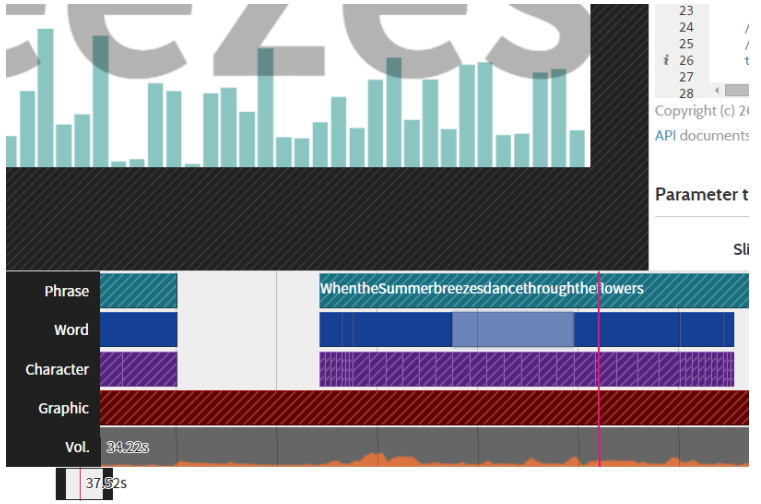
Replay & Refresh

Superspeed & slowmo



“Many worlds”

Stroboscopic visualization

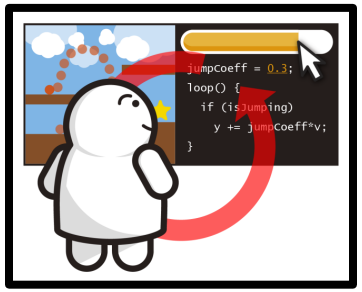


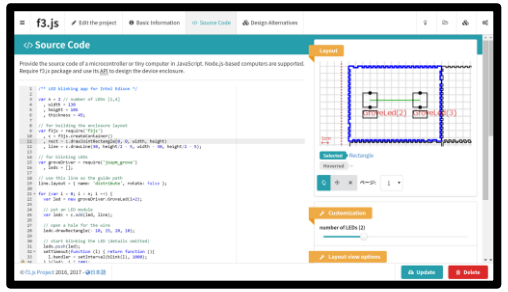
Timeline

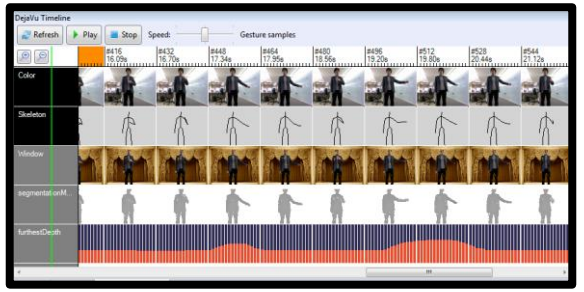
for absolute/semantic time

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

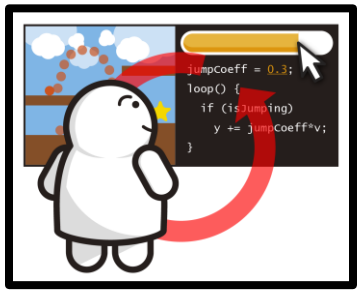
It’s **not only** about language design,
a single user, a single UI,
but about **designing the whole experience**

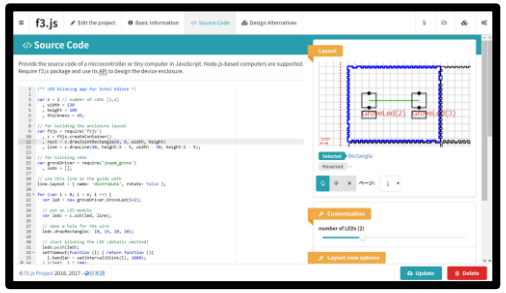
User Interfaces for Live Programming

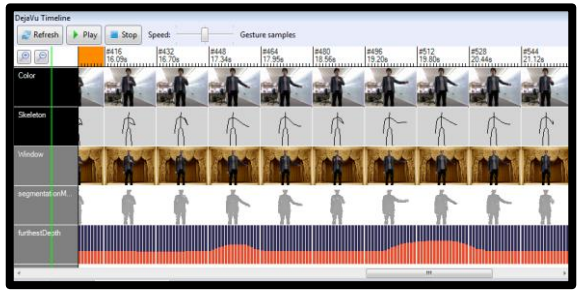
Jun Kato

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.