Towards More Natural Programming for Mobile and Touch

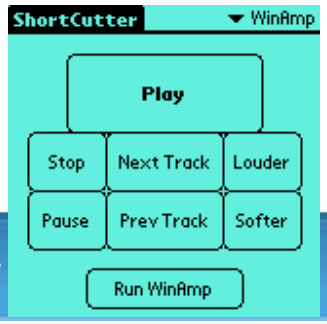
Brad A. Myers

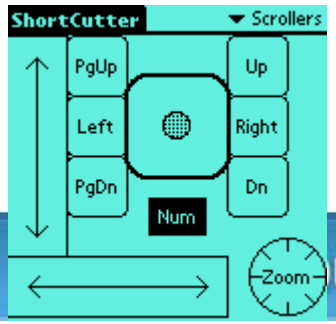
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
http://www.cs.cmu.edu/~bam
bam@cs.cmu.edu

Follow @bradamyers

Former Project: Pebbles

- **P**DAs for **E**ntry of **B**oth **B**ytes and **L**ocations from **E**xternal **S**ources
- http://www.pebbles.hcii.cmu.edu/
- One of the first to investigate Personal Digital Assistants (PDAs), 1997-2002
  - Starting with original Palm Pilot, Windows CE 2.1
- Key research – using PDAs *with* PCs
- Provided end-user programming of panels



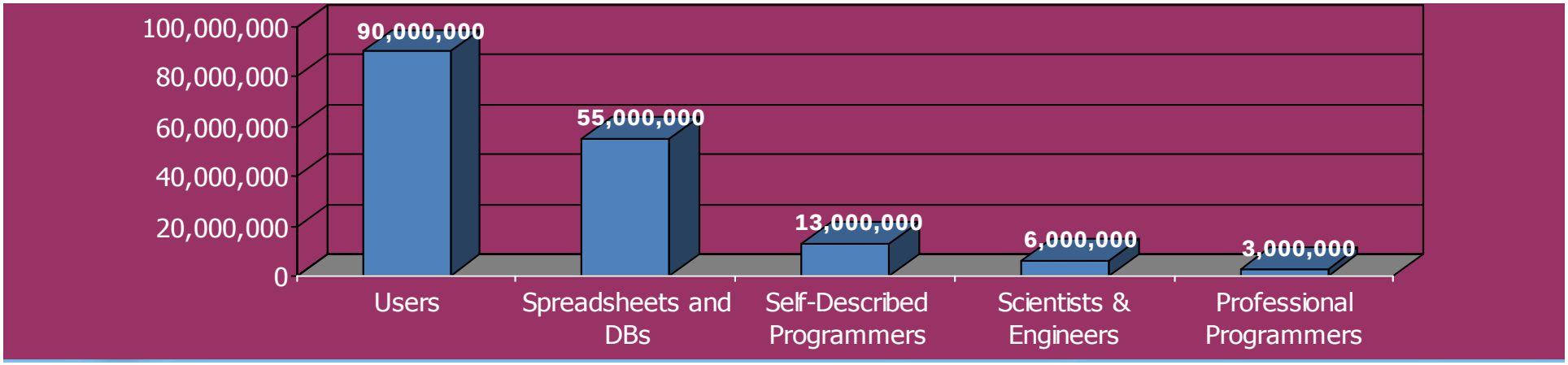


Natural Programming Project

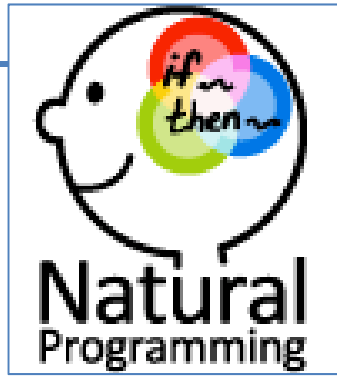
- Researching better tools for programming since 1978
- Natural Programming project started in 1995
- Make programming easier and more correct by making it more *natural*
  - Closer to the way that people think about algorithms and solving their tasks
- Methodology – human-centered approach
  - Perform *studies* to inform design
    - Provide new knowledge about what people do and think, & barriers
  - Guide the designs from the data
    - Design of programming *languages* and *environments*
  - Iteratively evaluate and improve the tools
- Target novice, expert and end-user programmers

“End-User Programmers”

- Programming tools are not just used by highly-trained professional programmers
- End-User Programmers = People whose primary job is *not* programming
- In 2012 in USA at work: — *[Scaffidi, Shaw and Myers 2005]*
  - 3 million professional programmers
  - 6 million scientists & engineers
  - 13 million will describe themselves as programmers
  - 55 million will use spreadsheets or databases at work
  - 90 million computer users at work in US



Why Would Being Natural be Good?



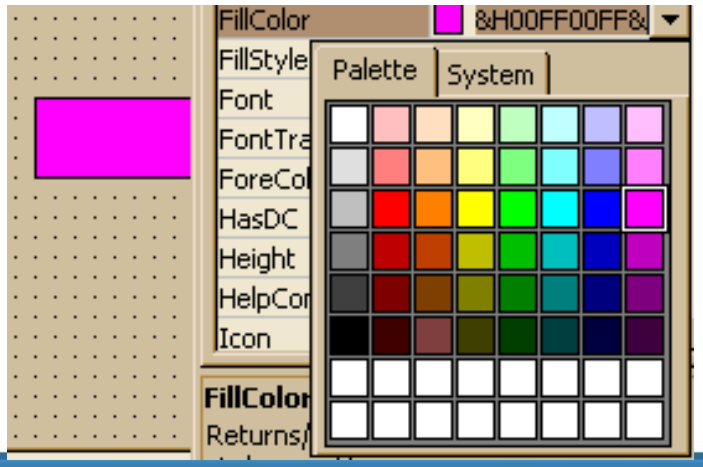
- Programmers are People Too
  - Take the human into account
- Language should be close to user's plan
  - "Programming is the process of transforming a mental plan into one that is compatible with the computer."
    — *Jean-Michel Hoc*
- *Closeness of mapping*
  - "The closer the programming world is to the problem world, the easier the problem-solving ought to be.... Conventional textual languages are a long way from that goal." — *Green and Petre*

Not so Natural!

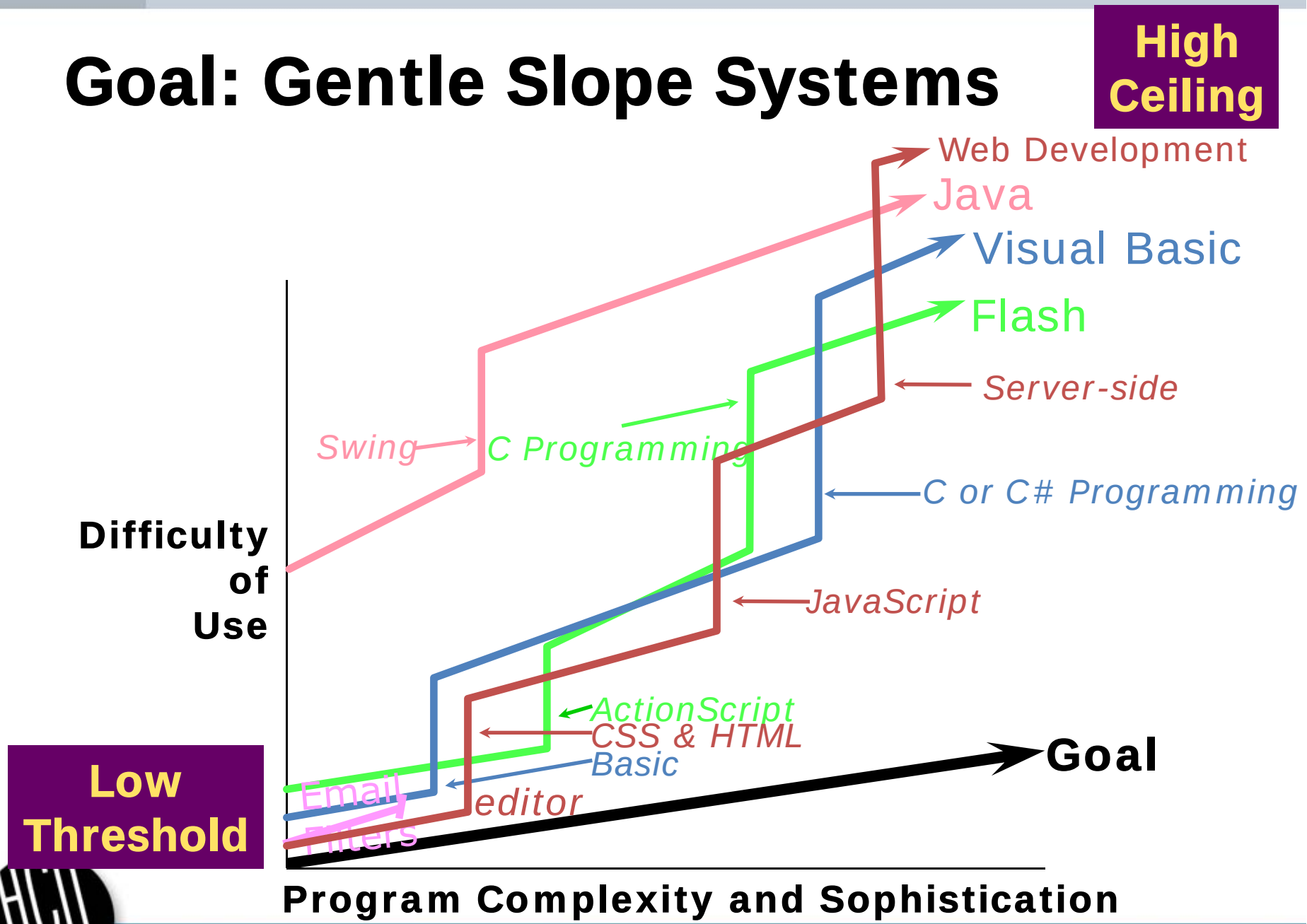
```
class HelloWorldApp {
      public static void main(String[] args) {
            System.out.println("Hello World!");
      }
}
```

- 3 kinds of parentheses and 9 special words!
- Compared to click and type: "Hello World!"

```
Let Shape1.FillColor
  = &H00FF00FF&
```



Goal: Gentle Slope Systems



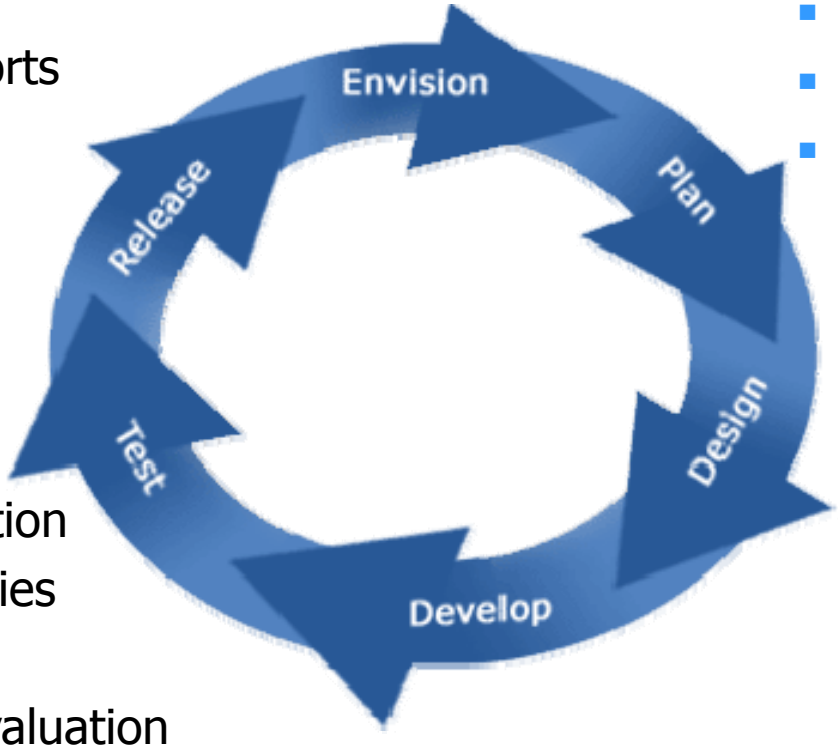
UX Techniques to Improve Programming

Exploratory Studies

- Contextual Inquiries
- Surveys
- Lab Studies
- Corpus data mining

Field Studies

- Logs & error reports



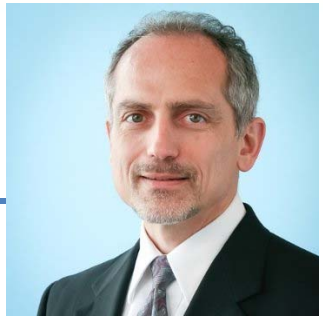
Evaluative Studies

- Usability Evaluation
- Formal Lab studies
- Expert analyses
  - Heuristic Evaluation
  - Cognitive Walkthroughs
  - Cognitive Dimensions

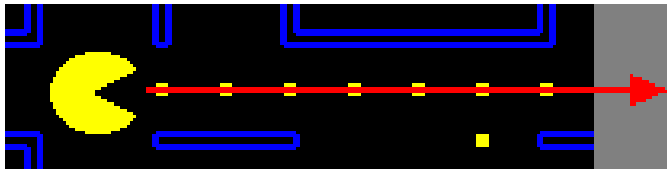
Design Practices

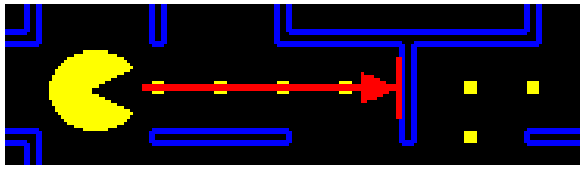
- "Natural programming" elicitation
- Graphic & Interaction Design
- Paper Prototyping

First Natural Programming Studies



- John Pane, PhD 2002
- Studies:
  - How people naturally express programming concepts and algorithms
    - 1) Nine scenes from PacMan
    - 2) Transforming and calculating data in a spreadsheet





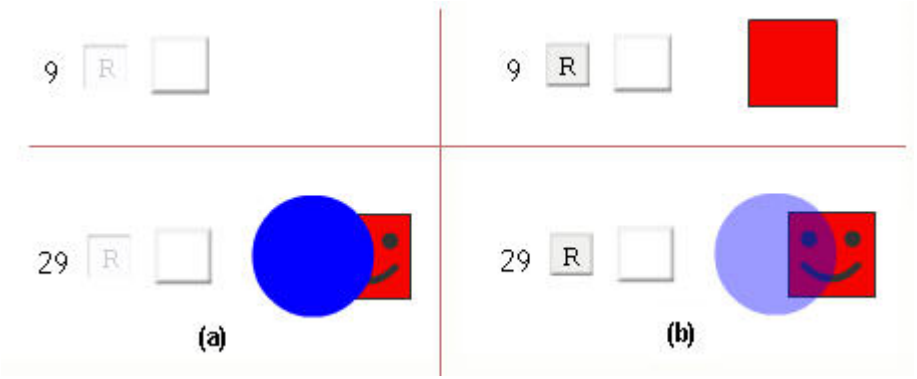
  - Specific issue of language design
    - 3) Selecting specific objects from a group ("and", "or", "not")
  - Lots of interesting results

Examples of Results

- Rule-based style
  "If PacMan loses all his lives, its game over."
- "And", "Or", "Not" don't match computer interpretation
  - … left-handed and right-handed people
  - … (not an apple) or pear
- Operations suggest data as lists, not arrays
  - People don't make space before inserting
- Objects normally moving
  "If PacMan hits a wall, he stops."
  - so objects remember their own state

Interactive Behaviors

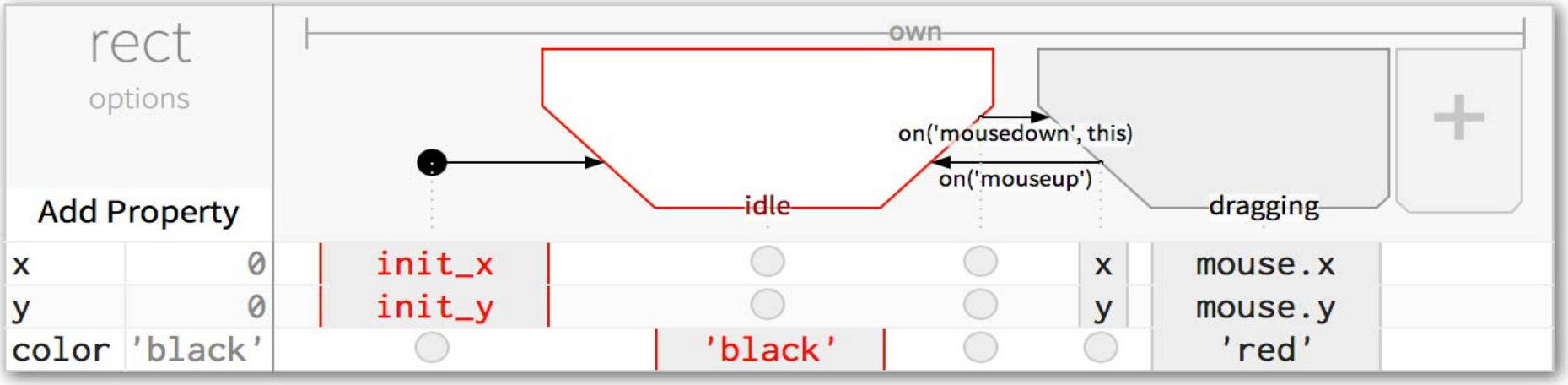
- *(VL/HCC'08)*
- Studied natural expression for interactive behaviors & animations
- Before and after pictures of primitives of interactive behaviors
- More use of constraints
- Consistent wording -- "appears", "fades out"



InterState

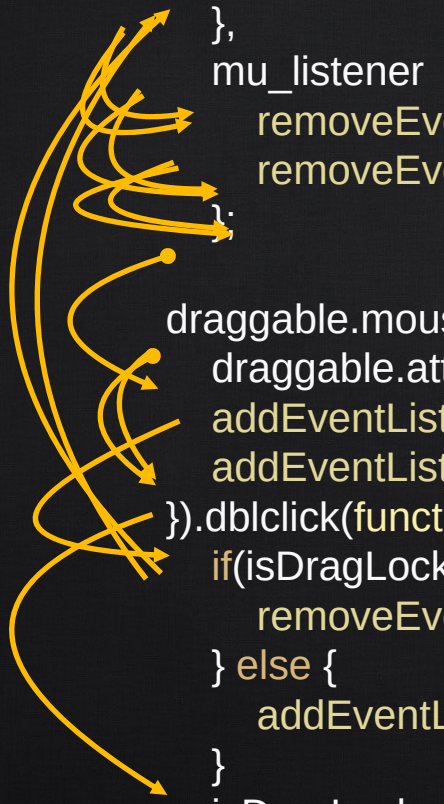
- PhD work of Stephen Oney (PhD 2015)
  - Now faculty at Univ. Michigan
  - http://interstate.from.so/
- Visual Programming Language for expressing behaviors
- Aimed at Interaction Designers (EUPs) who have some experience with programming
- Spreadsheet-like tables for object properties with constraints
- Columns are state machines to control when applied
- Many innovations in language, inheritance model, etc.

(UIST'14)
Video (3:36)



Equivalent drag-lock JavaScript code

```
var isDragLocked = false,
    mm_listener  = function(mm_event) {
        draggable.attr({ x: mm_ev.x, y: mm_ev.y });
    },
    mu_listener  = function(mu_event) {
        removeEventListener("mousemove", mm_listener);
        removeEventListener("mouseup",   mu_listener);
    };

draggable.mousedown(function(md_ev) {
    draggable.attr({ x: md_ev.x, y: md_ev.y });
    addEventListener("mousemove", mm_listener);
    addEventListener("mouseup",   mu_listener);
}).dblclick(function(md_event) {
    if(isDragLocked) {
        removeEventListener("mousemove", mm_listener);
    } else {
        addEventListener   ("mousemove", mm_listener);
    }
    isDragLocked = !isDragLocked;
});
```



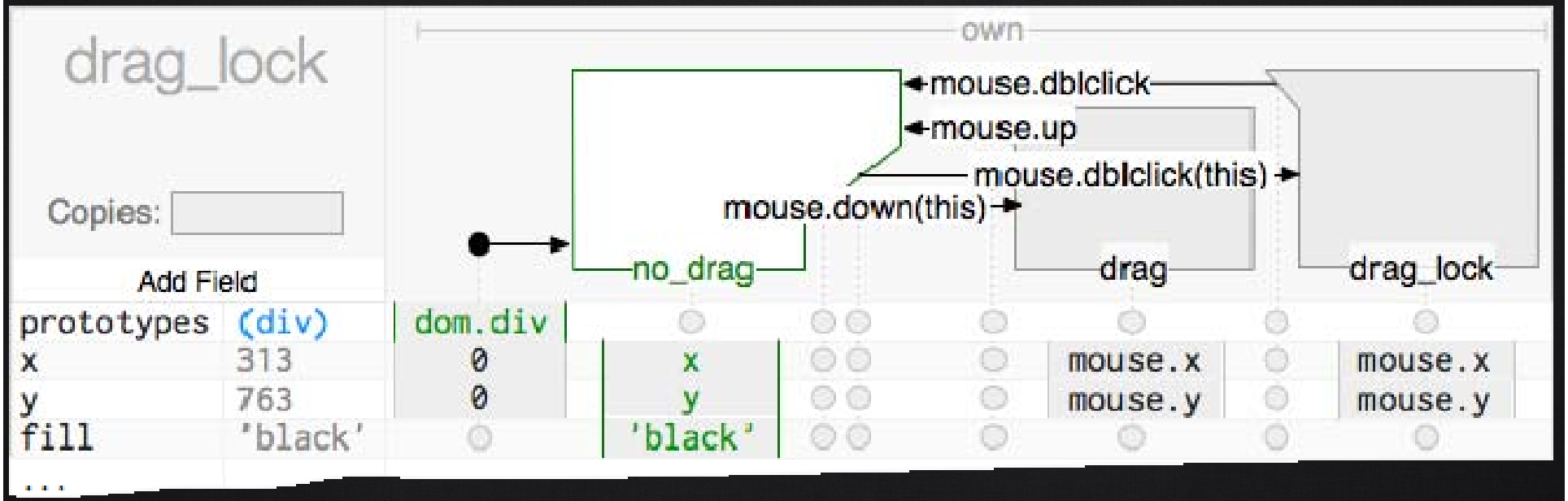
Control flow on double-click

Equivalent drag-lock JavaScript code

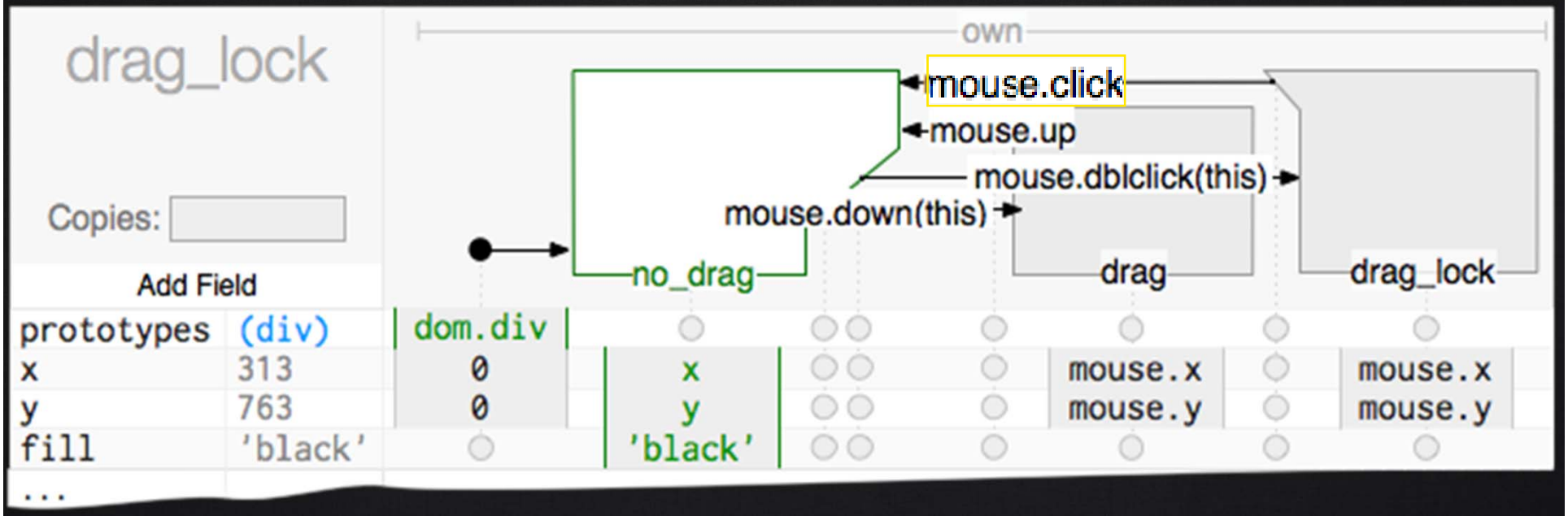
```
var isDragLocked = false,
    mm_listener  = function(mm_event) {
        draggable.attr({ x: mm_ev.x, y: mm_ev.y });
    },
    mu_listener  = function(mu_event) {
        removeEventListener("mousemove", mm_listener);
        removeEventListener("mouseup",   mu_listener);
    };

draggable.mousedown(function(md_ev) {
    draggable.attr({ x: md_ev.x, y: md_ev.y });
    addEventListener("mousemove", mm_listener);
    addEventListener("mouseup",   mu_listener);
}).dblclick(function(md_event) {
    if(isDragLocked) {
        removeEventListener("mousemove", mm_listener);
    } else {
        addEventListener   ("mousemove", mm_listener);
    }
    isDragLocked = !isDragLocked;
});
```

Control flow when click again

Most of the InterState code



Changes required for single-click to exit



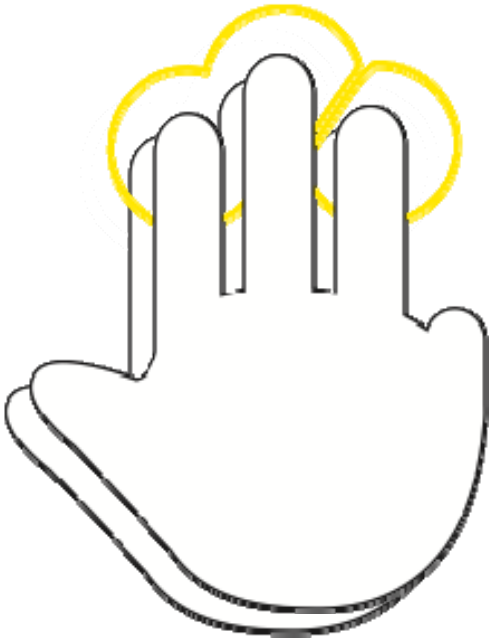
Changes required for single-click to exit

```
var isDragLocked = false,
    mm_listener  = function(mm_ev) {
        draggable.attr({ x: mm_ev.x, y: mm_ev.y });
    },
    mu_listener  = function(mu_event) {
        if(!isDragLocked) {
            removeEventListener("mousemove", mm_listener);
            removeEventListener("mouseup",   mu_listener);
        }
    };
draggable.mousedown(function(md_ev) {
    draggable.attr({ x: md_ev.x, y: md_ev.y });
    if(!isDragLocked) {
        addEventListener("mousemove", mm_listener);
        addEventListener("mouseup",   mu_listener);
    }
}).dblclick(function(md_event) {
    if(!isDragLocked) {
        removeEventListener("mousemove", mm_listener);
    } else {
                addEventListener("mousemove", mm_listener);
        isDragLocked = !isDragLocked;
    }
}).click(function(c_event) {
    if(isDragLocked) {
        removeEventListener("mousemove", mm_listener);
        isDragLocked = !isDragLocked;
    }
});
```

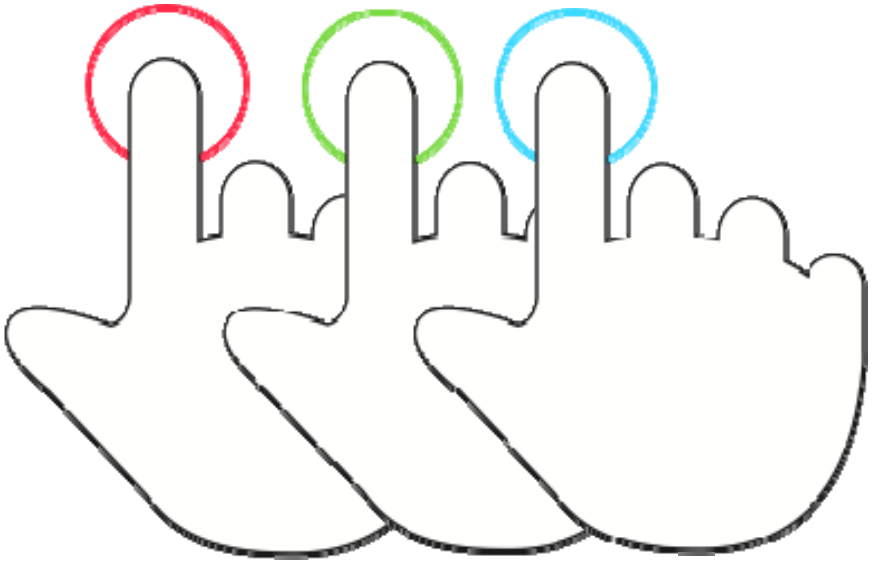
InterState: touch clusters

- Newest work – better ways to describe touch events and resulting behaviors
- Developers specify number of fingers, where pressed, etc.
- Outputs: location, scale, rotation
- Resolving conflicts:
  - optional delay & priority for events
  - touch clusters can determine whether other clusters can use same touches

Disambiguation



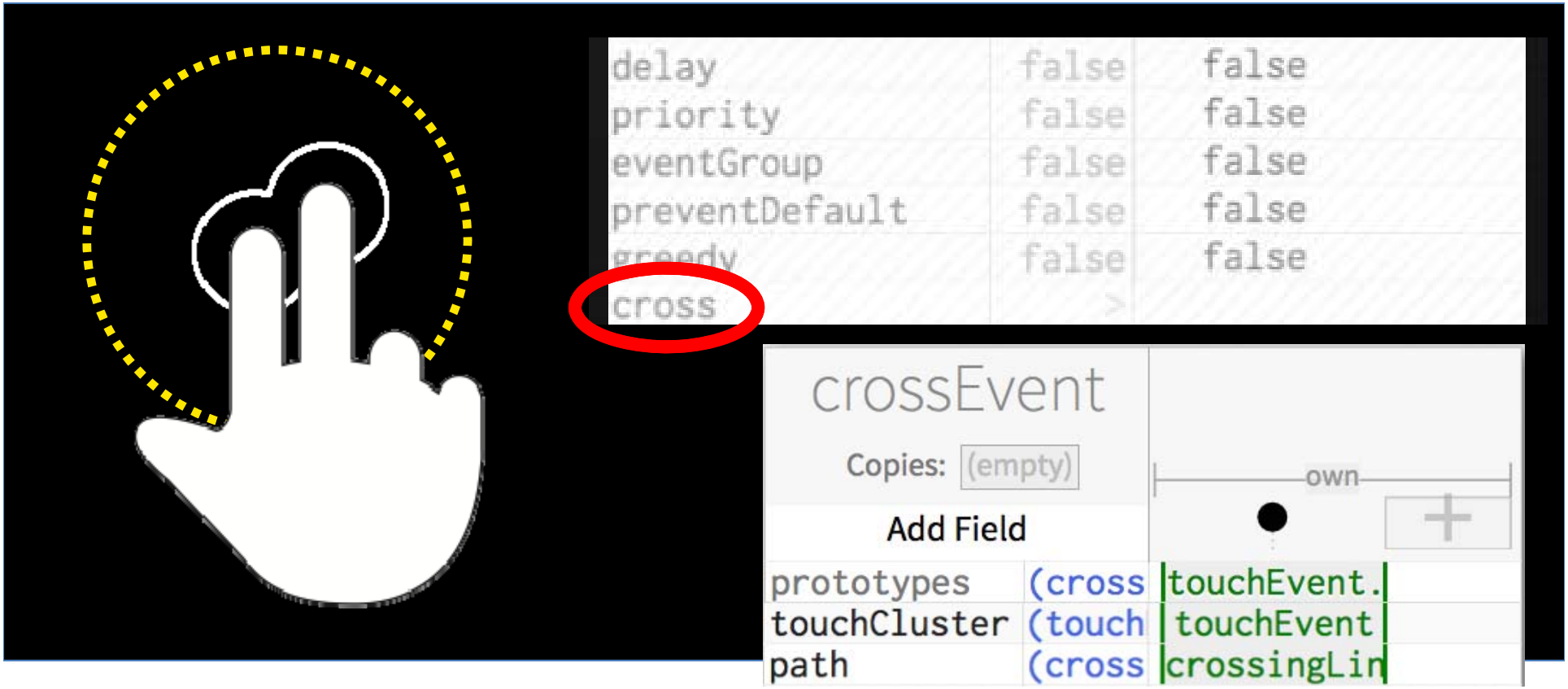
three-finger cluster fires



three single-touch clusters fire

delay	false	false
priority	false	false
eventGroup	false	false
preventDefault	false	false
greedy	false	false
cross		

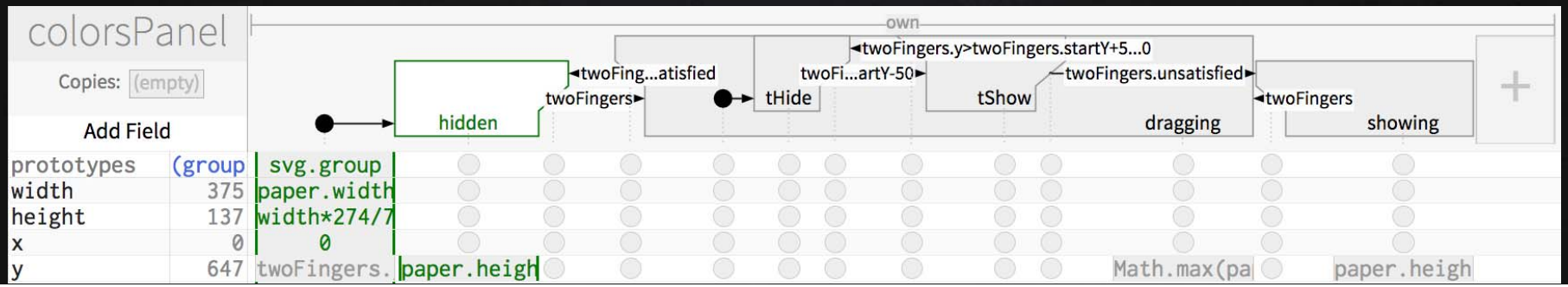
Crossing events



- Lines, circles, rects
- Can be calculated with formulas

Putting these together

- One-finger swipe up for tools
- Two-finger swipe up for colors
- Crossing invisible rectangle at the bottom



Gneiss: Extending Spreadsheet Programming

- PhD work of HCII student Kerry Chang (in progress)
- Gneiss: Gathering Novel End-user Internet Services using Spreadsheets
- Extend spreadsheet model so spreadsheet can calculate using web service data, streaming data, and web user interfaces
  - Lists of restaurants, movies, cars, stock prices, RSS feeds, Twitter feeds, … (almost anything!)
- Can also create user interfaces that use and control the values

(VL/HCC'14, UIST'14, CHI'15)

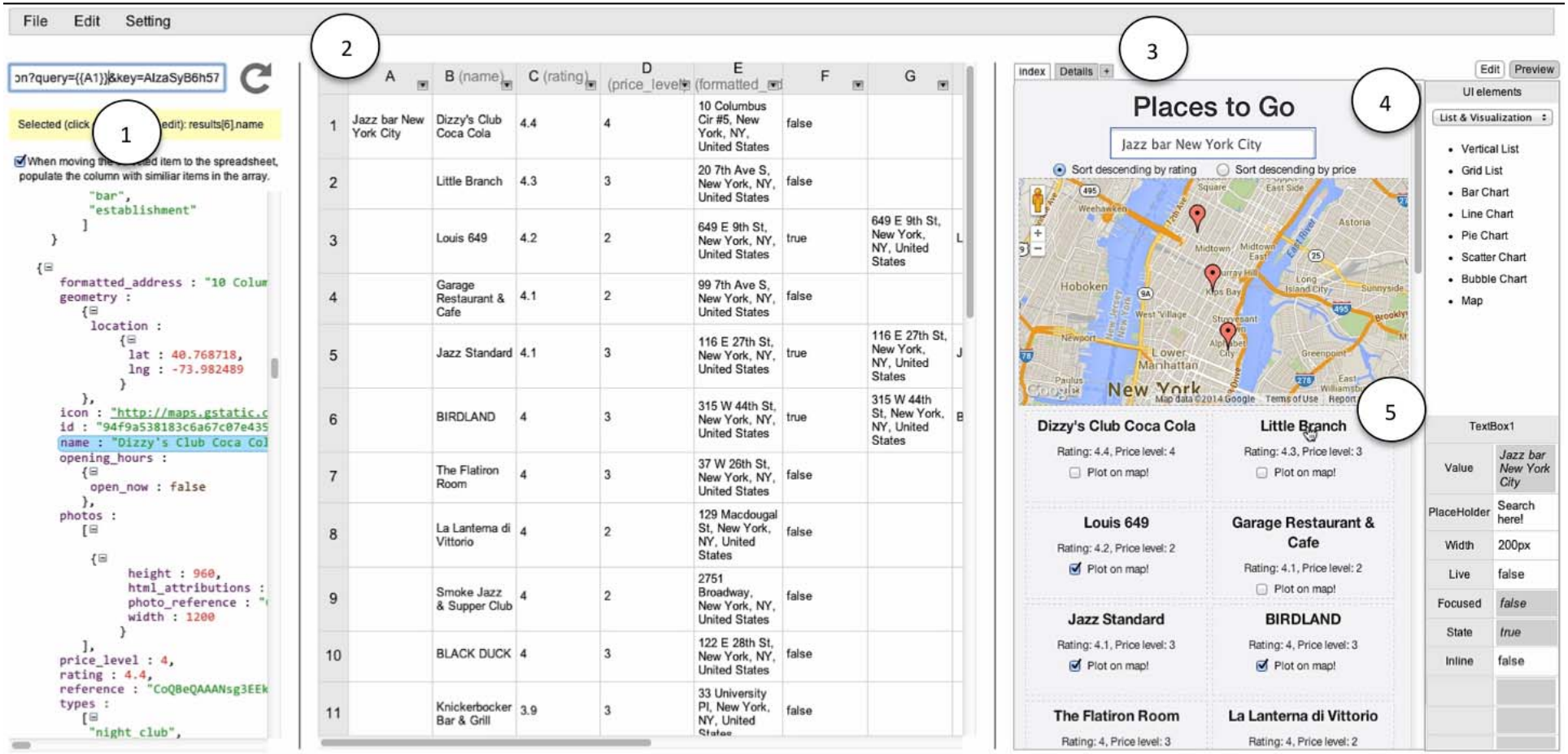
Gneiss Language

- Code using familiar spreadsheet language
  - Innovation: pull (formula) semantics even for user interface elements (instead of events)
- Interface builder to drag in UI elements
  - Connect to spreadsheet cells using formulas
  - Including lists – Autofill-down to populate
- Multiple pages – transitions based on input events and formulas

Gneiss Video

- Right pane could be on mobile device

[Video](#)
(5:00min)



Gneiss New Features

- Newest work – handle hierarchical data using spreadsheet UIs – e.g., JSON data
- Submitted for publication
- Drag columns to restructure
- Spreadsheet language can refer to cells at multiple levels

1

	A (businesses.name)	B (businesses.categories.cate	
1	Coca Cafe	1.1	breakfast_brunch
		1.2	newamerican
2	Waffles Incaffeinated	2.1	breakfast_brunch
		2.2	newamerican
		2.3	tradamerican
3	Point Brugge Café	3.1	belgian
4	The Dor-Stop Restaurant	4.1	breakfast_brunch
		4.2	diners
5	Deluca's Diner	5.1	breakfast_brunch

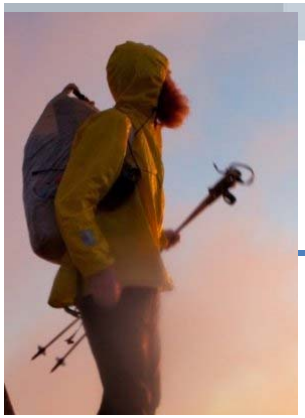
2

	A (businesses.name)	B (businesses.categories.cate	
	Column B		
1	Coca Cafe	1.1	breakfast_brunch
		1.2	newamerican
2	Waffles Incaffeinated	2.1	breakfast_brunch
		2.2	newamerican

3

	A (businesses.categori	B (businesses.name)
1	belgian	Point Brugge Café
2	belgian	Park Bruges
3	breakfast_brunch	Coca Cafe
4	breakfast_brunch	Waffles Incaffeinated

(1) a screenshot of our tool showing a list of restaurants and their categories retrieved from Yelp's JSON web service. Nested tables are used to represent the hierarchical structure. By dragging column B to the front (2), the user reshapes the data and views the restaurants by categories (3).



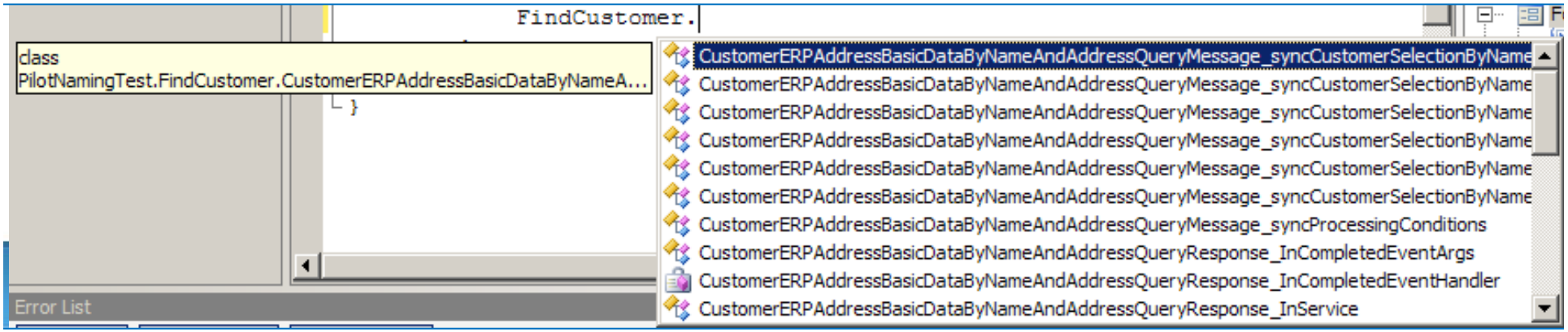
Study of APIs

- Started as PhD work of Jeff Stylos, PhD, 2009
  - Inspired by Steven Clarke, Microsoft Visual Studio group
- Application Programming Interface
  - Libraries, frameworks, SDKs, …
- Which programming patterns are most usable?
- Barriers to use of APIs
- Measures: learnability, errors, preferences
- Expert and novice programmers
- Studied:
  - Default parameters in constructors
  - Factory pattern
  - Object design
  - SAP's Web Services APIs
- See www.apiusability.org

Study of APIs for SAP

- Study APIs for Enterprise Service-Oriented Architectures ("Web Services")
- Naming problems:
  - Too long
  - Not understandable
  - Differences in *middle* are frequently missed

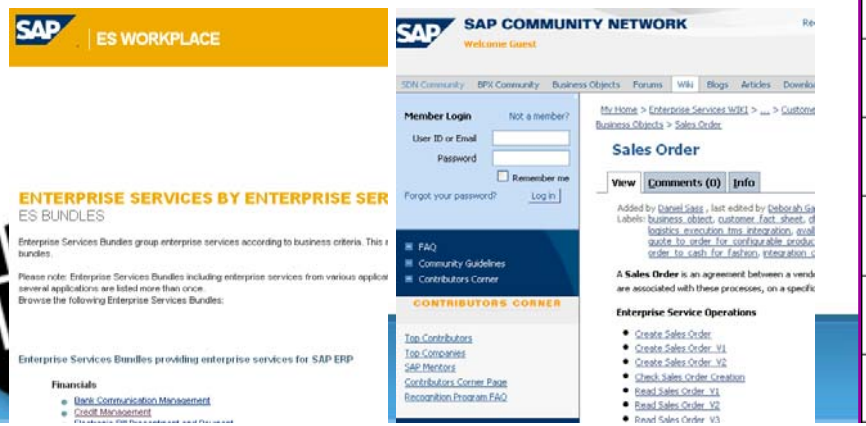
CustomerAddressBasicDataByNameAndAddressRequestMessageCustomerSelectionCommonName
CustomerAddressBasicDataByNameAndAddressResponseMessageCustomerSelectionCommonName

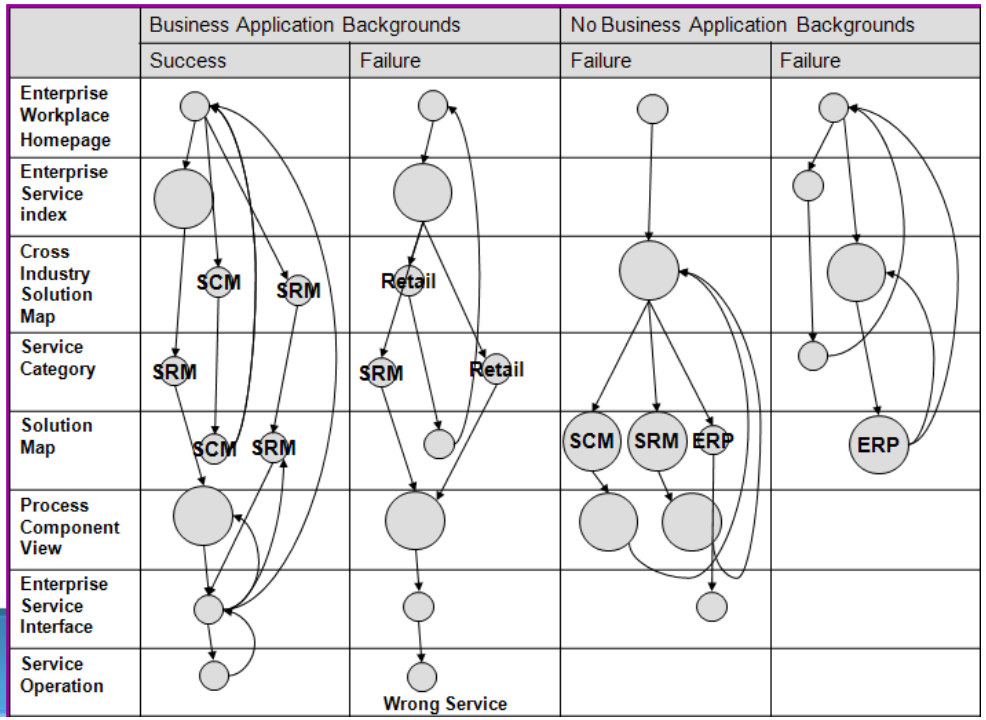


eSOA Documentation Results

- Multiple paths: unclear which one to use
- Some paths were dead ends
- Inconsistent look and feel caused immediate abandonment of paths
- Hard to find required information
- Business background helped

(IS-EUD'2009)





Required Constructors

- Compared create-set-call (default constructor)

```
var foo = new FooClass();
foo.Bar = barValue;
foo.Use();
```

- vs. required constructors (*immutable* classes):

```
var foo = new FooClass(barValue);
foo.Use();
```

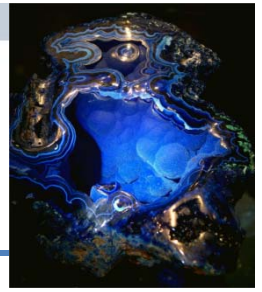
- All participants assumed there would be a default constructor
- Required constructors interfered with learning
  - Users wanted to experiment with what kind of object to use first
- Preferred to *not* use temporary variables

(Stylos & Clarke, ICSE '2007)

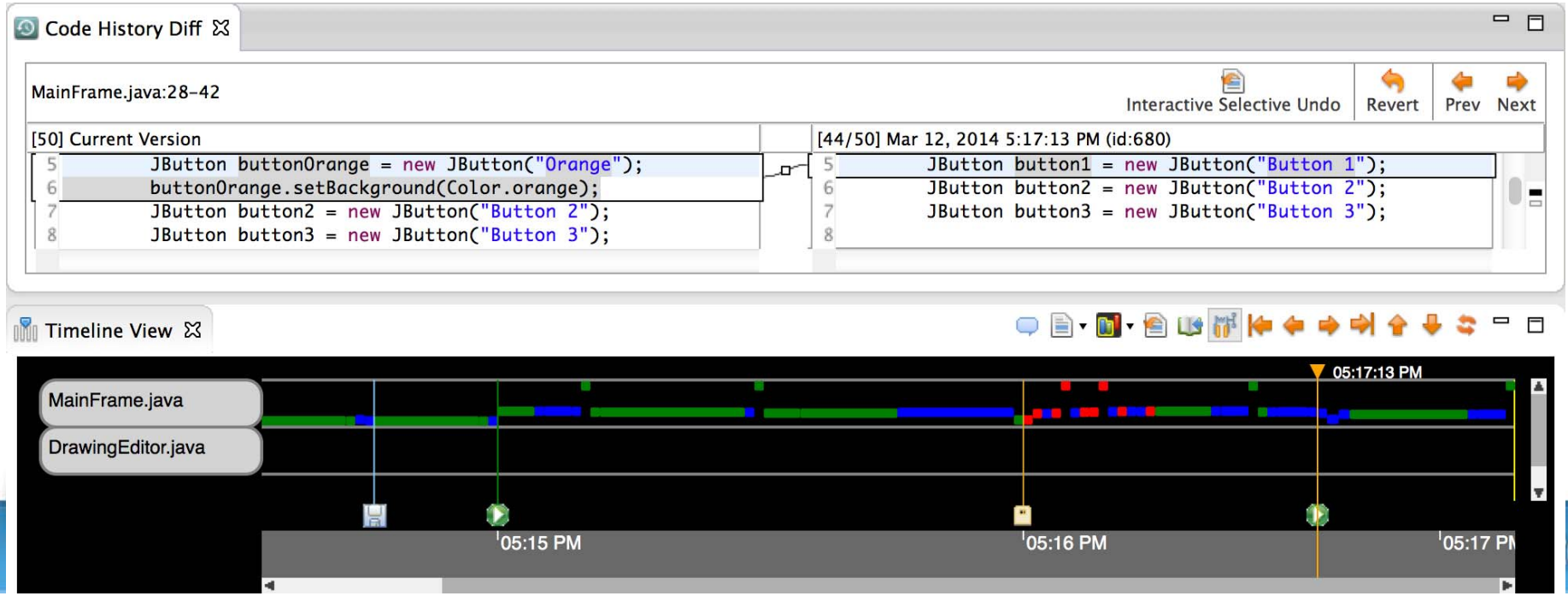
New Project: API Usability & Security

- Collaboration with CMU's Software Engineering Institute (SEI)
- New NSF grant & SEI grant
- Sometimes usability ≅ security
  - More usable → fewer mistakes
  - E.g., Android and iOS apps misused Secure Sockets Layer (SSL) or Transport Layer Security (TLS) due to difficulties with using the APIs and had vulnerabilities [Fahl, CCS 2013]
- But sometimes usability ≠ security
  - Mutability better for usability, worse for security
- How can usability research inform API design for security?
  - Current study: Immutability in APIs – (PhD student Michael Coblenz)
    - Interviews showed `const`, `final`, `readonly`, etc. are inadequate
- Future study: Error and exception handling

Azurite: Exploring Selective Undo



- PhD work of ISR student YoungSeok Yoon (May'2015)
- Azurite: Adding Zest to Undoing and Restoring Improves Textual Exploration **http://www.cs.cmu.edu/~azurite**
- Work out meaning of selective undo for code
  - Conflicting edits of same region of code
- Time-line visualization of all past operations
- Search through history (time) to find appropriate points *(VL/HCC'13 & '15, ICSE'15)*



Summary of Insights

- Field and lab studies can reveal the real issues
  - Addressing these issues creates tools that are actually useful
- Researcher's intuitions about what might be useful are often wrong
- Our experience highlights:
  - Understanding the barriers can lead to more effective tools
  - Many user-centered methods can be successfully applied to help understand developers and create better tools.
  - Completely different ways to program mobile applications are possible

There are lots of Gemstones!

- *And acronyms are fun!*

Fluorite:
Full of
Low-level
User
Operations
Recorded In
The
Editor

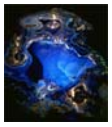
Apatite:
Associative
Perusing of
APIs
That
Identifies
Targets
Easily

Jadeite:
Java
API
Documentation with
Extra
Information
Tacked-on for
Emphasis

Aquamarine:
Allowing
Quick
Undoing of
Any
Marks
And
Repair
Improving
Novel
Editing

Jasper:
Java
Aid with
Sets of
Pertinent
Elements for
Recall

Azurite:
Adding
Zest to
Undoing and
Restoring
Improves
Textual
Exploration



Euclase:
End
User
Centered
Language,
APIs
System and
Environment

Crystal:
Clarifications
Regarding
Your
Software using a
Toolkit,
Architecture and
Language

GARNET
Generating an
Amalgam of
Real-time,
Novel
Editors and
Toolkits

Euklas:
Eclipse
Users'
Keystrokes
Lessened by
Attaching from
Samples

Graphite:
GRAphical
Palettes
Help
Instantiate
Types in the
Editor

Calcite:
Construction
And
Language
Completion
Integrated
Throughout

GNEISS:
Gathering
Novel
End-user
Internet
Services using
Spreadsheets

PEBBLES
PDAs for
Entry of
Both
Bytes and
Locations from
External
Sources

C32
CMU's
Clever and
Compelling
Contribution to
Computer Science in
CommonLisp which is
Customizable and
Characterized by a
Complete
Coverage of
Code and
Contains a
Cornucopia of
Creative
Constructs, because it
Can
Create
Complex,
Correct
Constraints that are
Constructed
Clearly and
Concretely, and
Communicated using
Columns of
Cells, that are
Constantly
Calculated so they
Change
Continuously, and
Cancel
Confusion

For more, see: www.cs.cmu.edu/~bam/acronyms.html

Thanks to:

- Funding:
  - NSF under CNS-1423054, HCC-1314356, IIS-1116724, IIS-0329090, CCF-0811610, IIS-0757511 (Creative-IT), and NSF ITR CCR-0324770 as part of the EUSES Consortium
  - SAP
  - Adobe
  - IBM
  - Microsoft Research RISE

SAP®

Adobe

EUSES
End Users Shaping Effective Software

IBM®

Microsoft Research
RiSE

- >32 students:
  - Htet Htet Aung
  - Jack Beaton
  - Ruben Carbonell
  - John R. Chang
  - Kerry S. Chang
  - Polo Chau
  - Luis J. Cota
  - Michael Coblenz
  - Dan Eisenberg
  - Brian Ellis
  - Andrew Faulring
  - Aristiwidya B. (Ika) Hardjanto
  - Erik Harpstead
  - Sae Young (Sophie) Jeong
  - Andy Ko
  - Sebon Koo
  - Ashley Lai
  - Thomas LaToza
  - Tam Minh Le
  - Joonhwan Lee
  - Leah Miller
  - Mathew Mooty
  - Gregory Mueller
  - Yoko Nakano
  - Stephen Oney
  - John Pane
  - Sunyoung Park
  - Chotirat (Ann) Ratanamahatana
  - Christopher Scaffidi
  - Jeff Stylos
  - David A. Weitzman
  - Yingyu (Clare) Xie
  - Zizhuang (Zizzy) Yang
  - YoungSeok Yoon

Towards More Natural Programming for Mobile and Touch

Brad A. Myers

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
http://www.cs.cmu.edu/~bam
bam@cs.cmu.edu

Follow @bradamyers