Lecture 23:
Simple User Interface Toolkits and End-User Programming for UIs; Low-Code / NoCode

05-431/631 Software Structures for User Interfaces (SSUI)

Fall, 2021

Logistics

- HW6 due today

Overview

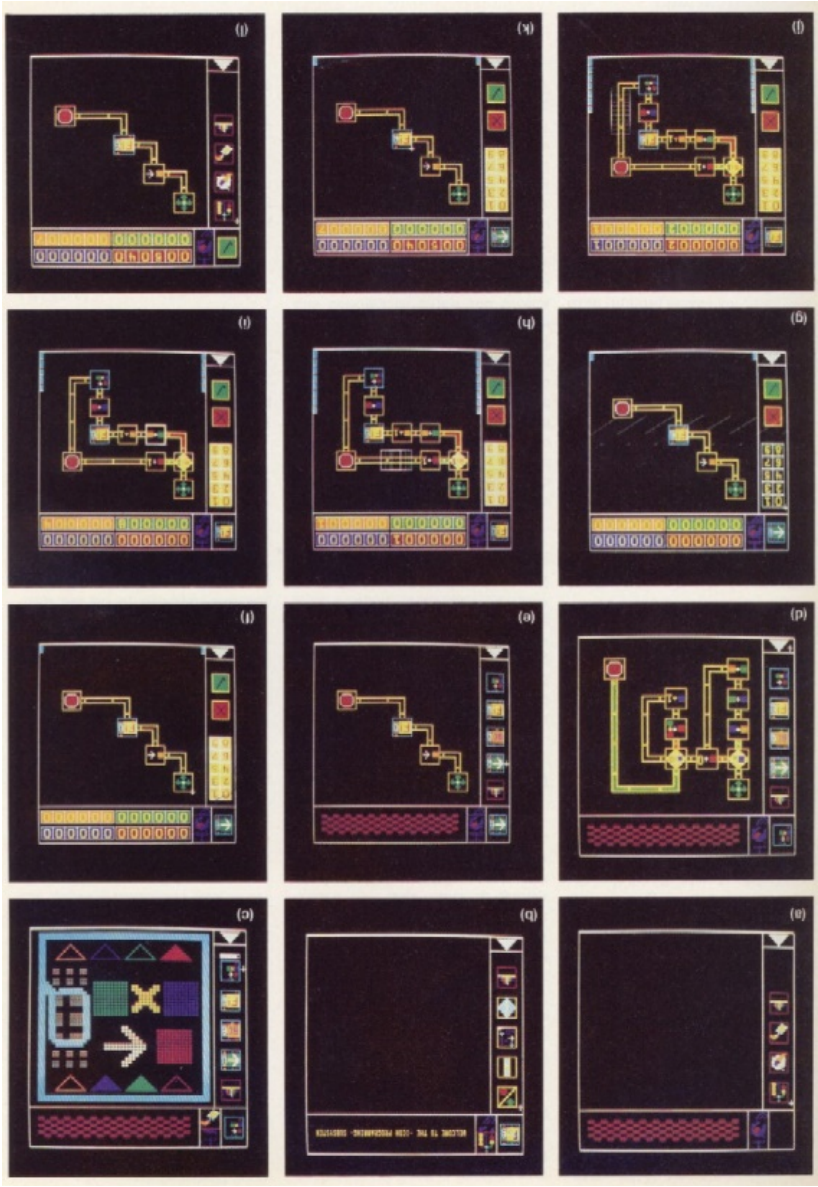
- Approaches to help novice programmers to be able to create dynamic interfaces
  - Static interfaces can just be drawn
- Typically, also easier to program in general
  - Most modern tools to make it easy to program focus on creating interactive software, like games, which have a UI
- EUP = end-user programmers
- Definition: Visual Programming = "Programming in which more than one dimension is used to convey semantics." *- [Myers, 1990]*

New terms: Low-Code, No-Code

- "Low-Code" – create app by "graphical user interfaces and configuration instead of traditional programming" – *Wikipedia*
  - term coined in 2014
  - Often use visual programming
  - Require some coding expertise
- "No-Code" – theoretically no coding at all, but often the same as "low code"
  - Aim to allow business people to create the whole app
  - Often declarative "model-driven", with fixed GUI

Older Approaches

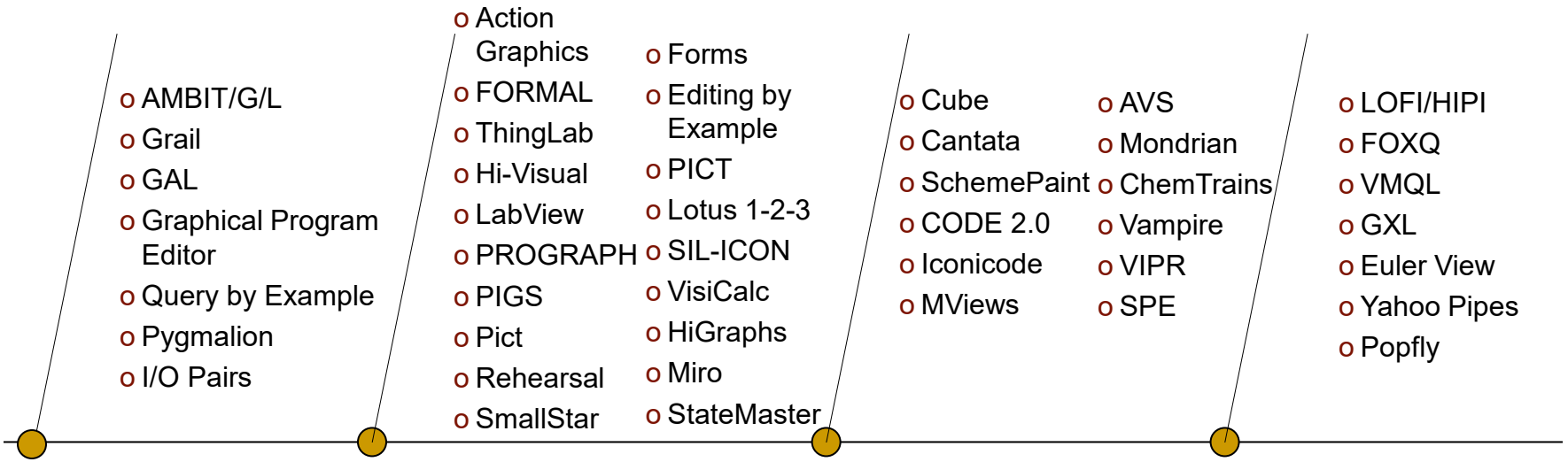
- Older visual language systems did not necessarily help with UIs
- E.g., Pict from Ephraim Glinert, 1984 uses conventional flowcharts to program algorithms
- Goal: easier to learn programming



Some Examples

- LabVIEW (1991 - present)
  - See also OutSystems (2001- present)
- SUIT [Pausch, 1992]
- Alice [Pausch, 1995]
- HANDS [Pane, 2002]
- Yahoo! Pipes (2007 – 2015)
- Scratch (2003-present)
- AppInventor (2009-present)
- Lego Mindstorms (NXT) Robot kits

Historical trends



1960

Techniques

- Graphs
- Flowcharts
- Flowchart derivatives
- FORMS
- Demonstrational

From: **Vishal Dwivedi**, 05-830 in 2013

1980

Techniques

- Graphs
- Flowcharts
- Flowchart derivatives
- FORMS
- Demonstrational
- Data Flows
- Spreadsheets
- Matrices
- Jigsaw Puzzles
- Petri nets
- Flowchart derivatives

1990

Techniques/Goals

- 3D Rendering
- Visual Hierarchy
- Procedures
- Control Structures
- Programmable Graphics
- Animations
- Video Imagery Exploitation
- General purpose, declarative language
- Audio, video and image processing
- Graphical models from behavioral models
- Learning and Cognitive abilities in vision processes
- Handling Scalability, typing, and imperative design
- Collaborative Software Development

2000

Techniques/Goals

- Child Learning
- Xquery by FORMS
- Spreadsheet Analysis
- Visual Model Query
- Layouts
- Specification and Interchange
- Mashups
- Web-based design
- Programming for end-users (non-Professionals

LABView

- One of the most successful visual programming systems
- Started about 1991 on Macintosh, still going
  - http://www.ni.com/labview/
  - J. Kodosky, J. MacCrisken and G. Rymar. "Visual programming using structured data flow," *Visual Languages, 1991., Proceedings. 1991 IEEE Workshop on, 8-11 Oct 1991, 1991. pp. 34-39.*
- Focused on scientists and lab equipment
- Wiring diagram backend with front panel
  - Drag and drop elements
  - Data flow programming

LabVIEW

- [Kodosky, 91]
- 2-view approach very influential

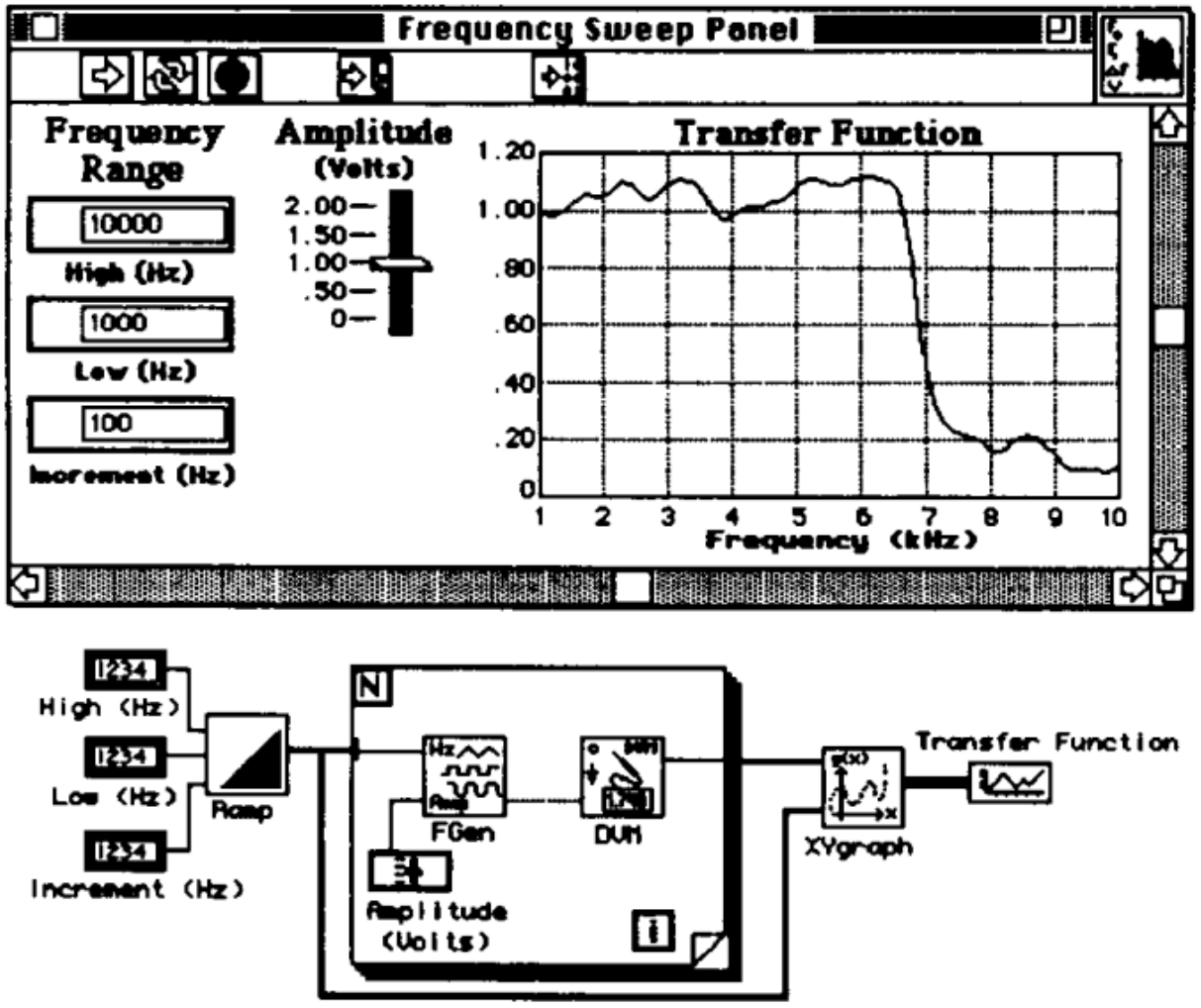
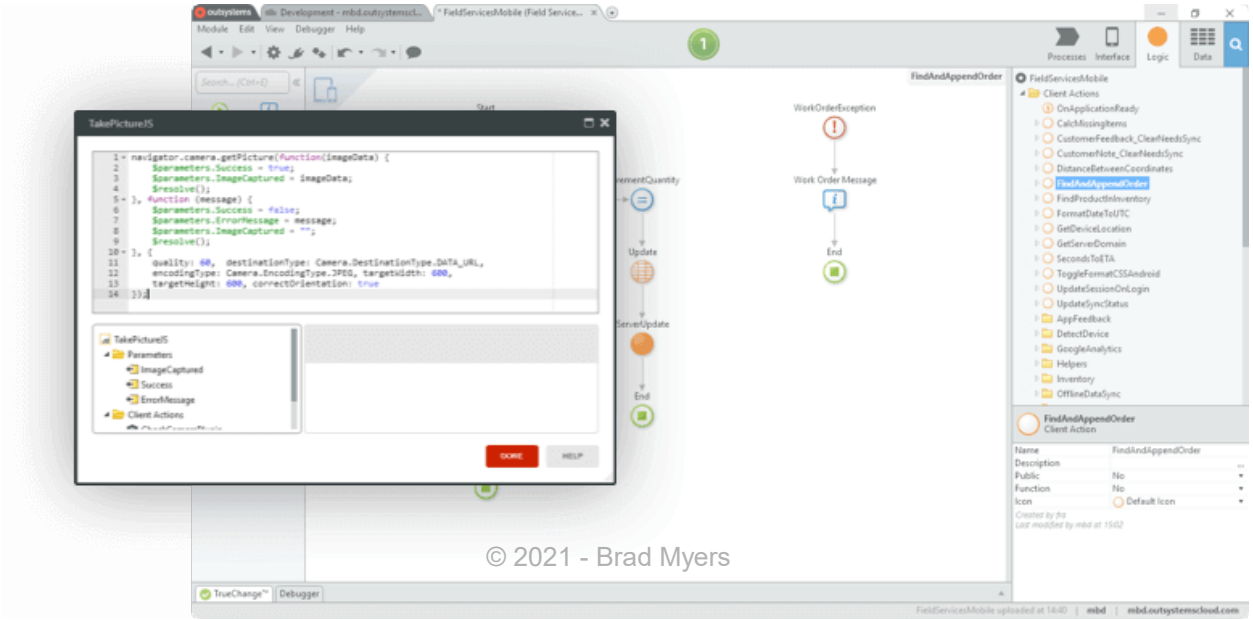


Figure 7. Frequency Response VI Panel and Diagram

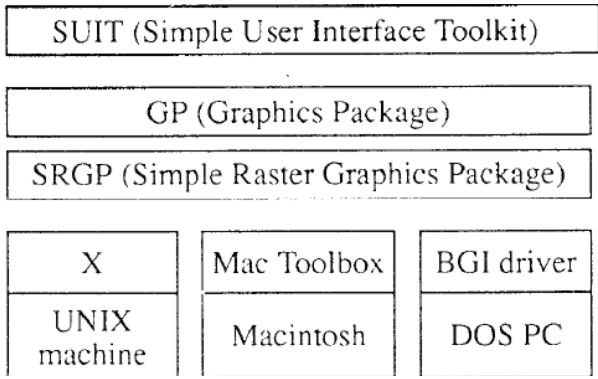
OutSystems

- 2001 – present
- https://www.outsystems.com/ - see video
- “low-code” platform
- Drag and drop, visual programming
- Focus on enterprise



SUIT (1992)

- Pausch, R., Conway, M., & DeLine, R. (1992). Lesson Learned from SUIT, the Simple User Interface Toolkit. *ACM Transactions on Information Systems*, 10(4), 320-344.
- Simple User Interface Toolkit
- Implemented in C
  - Portable across UNIX, Macintosh, and DOS
  - Requires only basic C programming skills
- Used in many courses at UVA
  - Become productive in 2½ hours, vs. weeks
- Iterative user testing
- Table of objects
  - No inheritance – just global or local
  - Property sheets
- Uses CTRL-SHIFT to avoid run/build mode
- Retained object model
- Data-linkages (constraints) through drag-and-drop



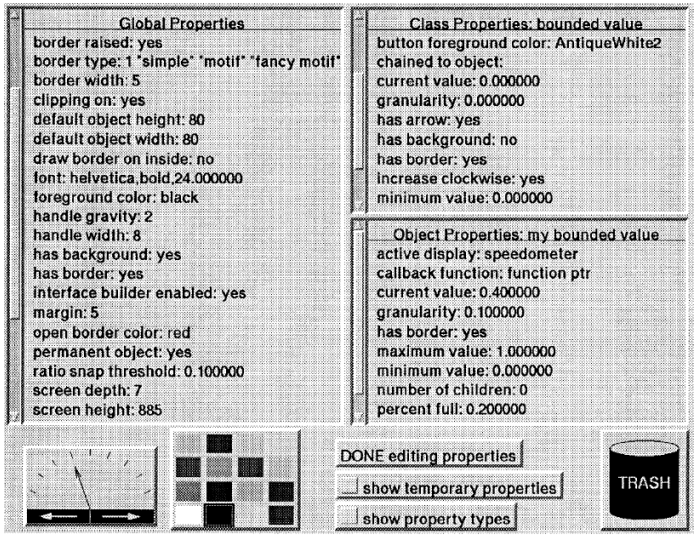


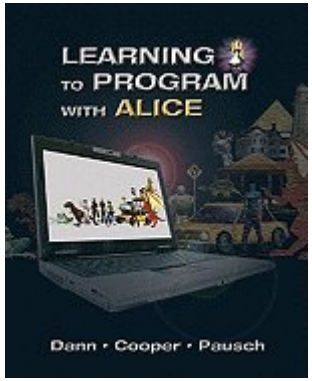
Figure 3: The SUIT Property Editor

Alice

Randy Pausch, Tommy Burnette, A.C. Capehart, Matthew Conway, Dennis Cosgrove, Rob DeLine, Jim Durbin, Rich Gossweiler, Suichi Koga and Jeff White. "Alice: A Rapid Prototyping System for 3D Graphics," *IEEE Computer Graphics and Applications. 1995. 15(3). pp. 8-11. May.*

Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, Kevin Christiansen, Rob Deline, Jim Durbin, Rich Gossweiler, Shuichi Koga, Chris Long, Beth Mallory, Steve Miale, Kristen Monkaitis, James Patten, Jeff Pierce, Joe Shochet, David Staack, Brian Stearns, Richard Stoakley, Chris Sturgill, John Viega, Jeff White, George Williams and Randy Pausch. "Alice: Lessons Learned from Building a 3D System For Novices," *Proceedings CHI'2000: Human Factors in Computing Systems, The Hague, The Netherlands, Apr 1-6, 2000. pp. 486-493. http://www.alice.org*

- Started as a 3D extension to SUIT
- PhD dissertation of Matthew Conway (1998)
- Grown to a large-scale system with books
  - Wanda Dann, Steven Cooper and Randy Pausch. *Learning to Program With Alice. Prentice-Hall. August, 2003.*
- Many more user studies of what students found easy and difficult



Alice

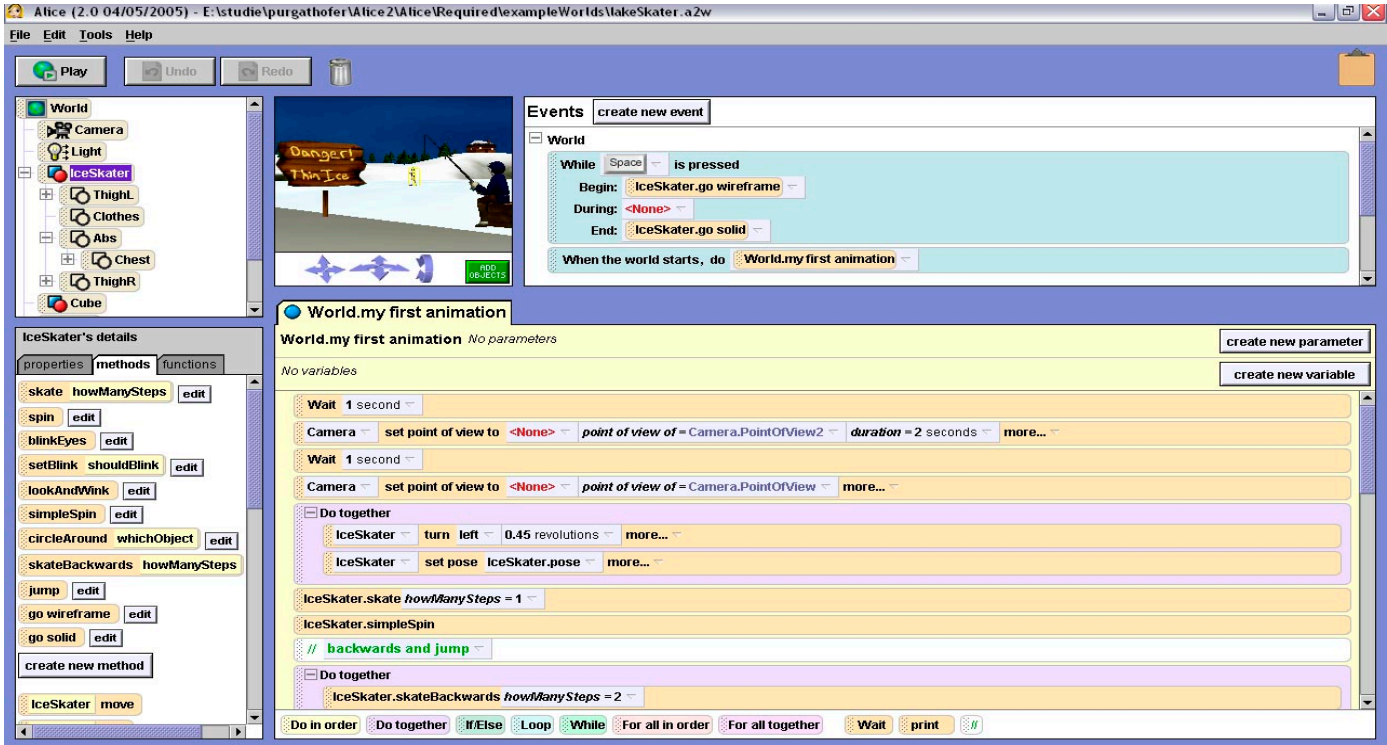
- Easy 3D with character-centered movement & rotation
  - "Bunny.move (up, 1)", "Turn Around Once" "Bunny.move(forward, 1, speed=4)"
  - No matrices! No X, Y and Z
- Camera control
  - "Point Camera At", "Get a Good Look At"
- Easy parallelism with "do-together"

```
ArmsOut = DoTogether(
            Bunny.Body.LeftArm.Turn(Left, 1/8),
            Bunny.Body.RightArm.Turn(Right, 1/8) )
```

- Create scene (by direct manipulation), then script
- All commands animated by default, so no sudden jumps, disappearing objects
- Early user of Python, switched to Java
- Lots of vocabulary fixes:
  - Resize, not Scale; Move, not Translate; Speed, not Rate; FrontToBack, not Depth:

Alice, cont.

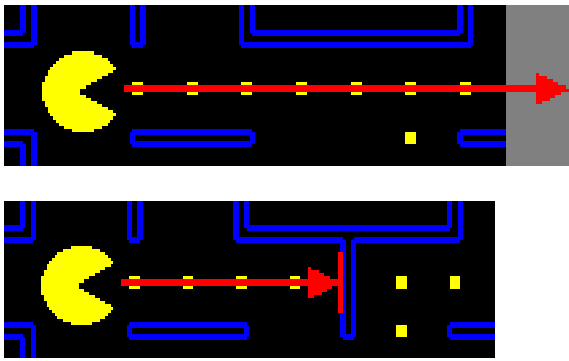
- Later versions: Avoid syntax issues with drag-and-drop editing
- Testing in classrooms showed significantly better learning and retention
- Tutorial video (from 2013) *6:44*



HANDS

- J.F. Pane, B.A. Myers and L.B. Miller. "Using HCI Techniques to Design a More Usable Programming System," *IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC 2002), Arlington, VA, September 3-6, 2002. 198-206.*

- PhD 2002 of John Pane (now at Rand in Pgh)
- 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."

- Set operations instead of iterations

  "When PacMan eats all of the dots, he goes to the next level."

- "And", "Or", "Not" don't match computer interpretation
- Most arithmetic used natural language style

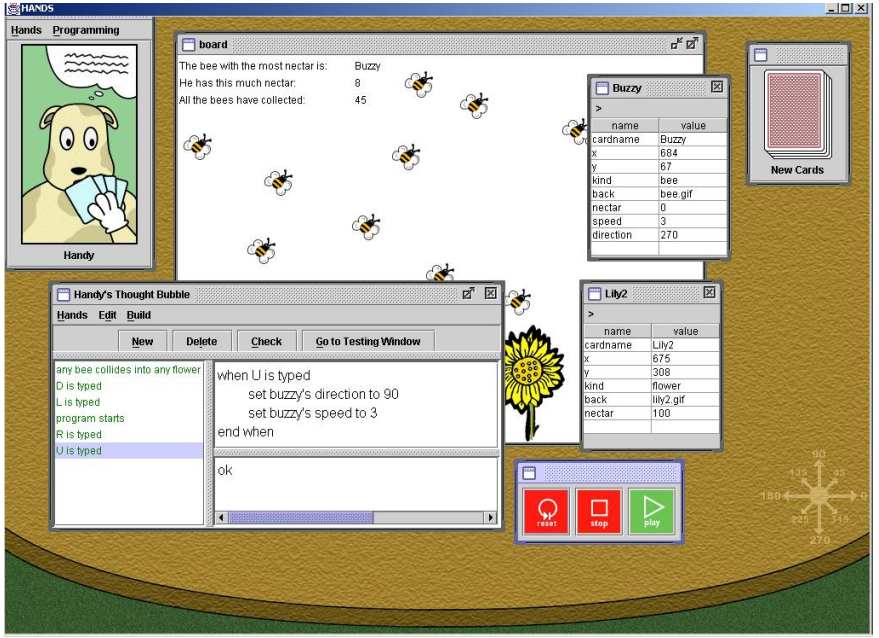
  "When PacMan eats a big dot, the score goes up 100."

- 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

New Language and System: HANDS



- Properties:
  - All data visible on *cards*
  - Metaphor of agent (Handy the dog) operating on cards
  - Natural language style for code
  - Domain-specific operations, like movement in a direction
  - All operations can operate on single items or sets of items
  - Sets can be dynamically constructed and used
    - "Set the speed of all bees to 0"
  - Event handlers: "when U is typed"
- See the video: *YouTube (7:36)*

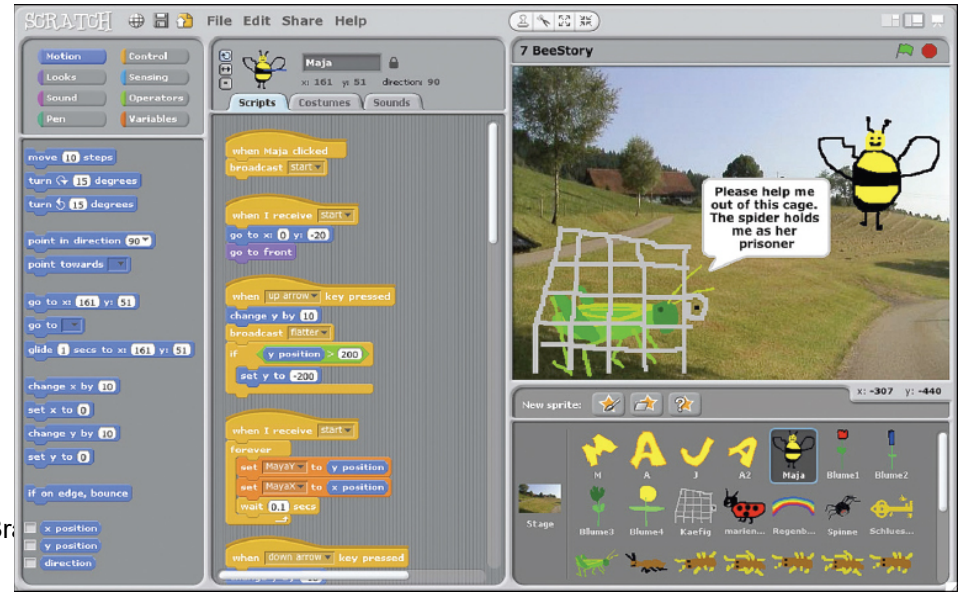
Scratch

- Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman and Yasmin Kafai. "Scratch: Programming for All," *Comm. ACM. 2009.* ***52(11). pp. 60-67. See also: http://scratch.mit.edu/.***

- MIT has long history of helping kids program
  - Logo (Seymour Papert, 1967)
  - Lego Mindstorms
  - "Constructionist" movement in education
- Scratch comes out of that program (MIT Media Lab) – started about 2003
  - https://scratch.mit.edu/
  - "Create stories, games, and animations Share with others around the world"

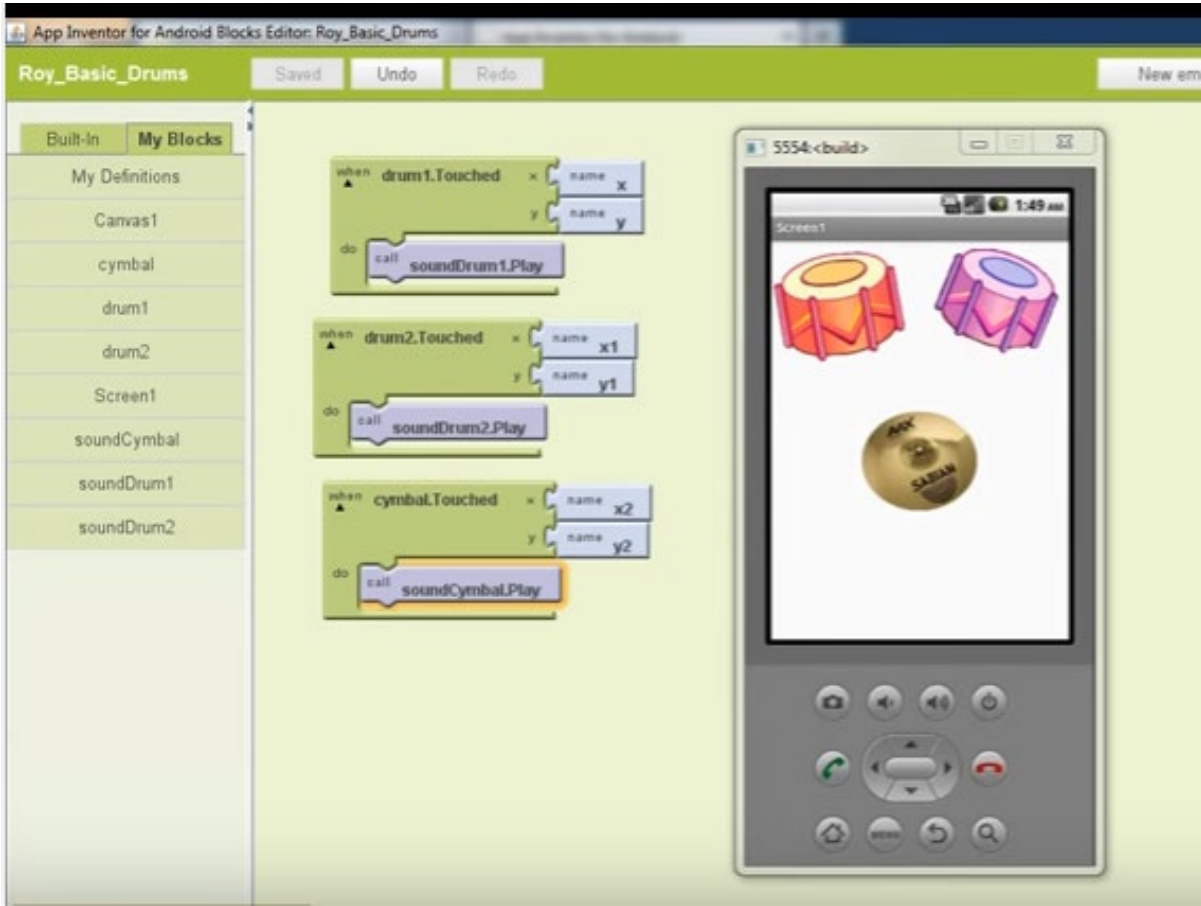
Scratch, cont.

- Metaphor of puzzle pieces with properties
  - Connectors shaped by type to eliminate type errors
  - Control structures wrap around
- Uses event handlers for behaviors
- https://vimeo.com/65583694, 1:37



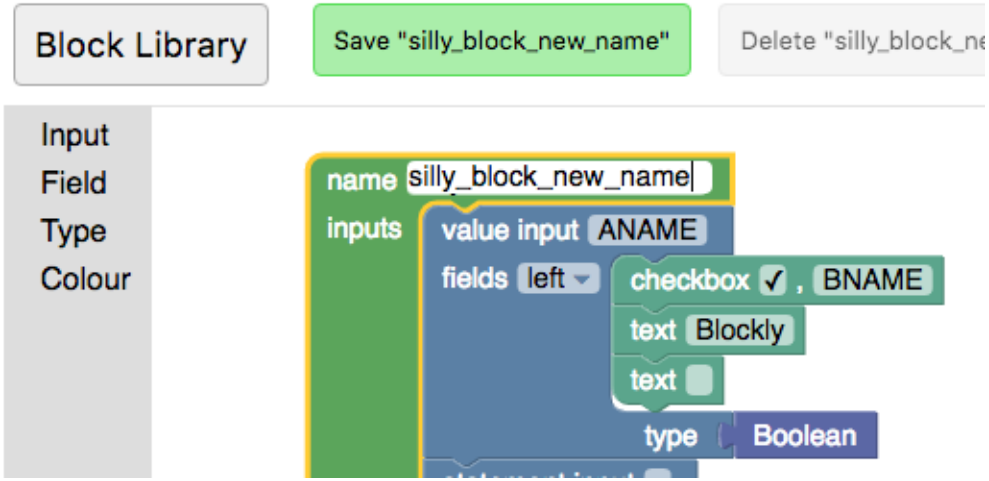
AppInventor

- Ideas from Scratch to build real Apps for Android phones
  - Briefly was a product from Google, while Hal Abelson was on sabbatical there (2009)
- http://appinventor.mit.edu/
- 2 panel view, like LabVIEW
  - Drag in elements for UI
  - Blocks view for code
  - event handlers using “when”

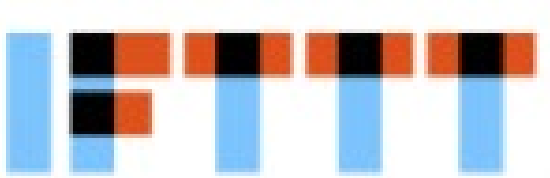


Many other “blocks” style languages

- Blockly Developer Tools from Google from AppInventor
- Used by Code.org, RoboBlockly, Wonder Workshop, etc.
- Define own set of primitive blocks

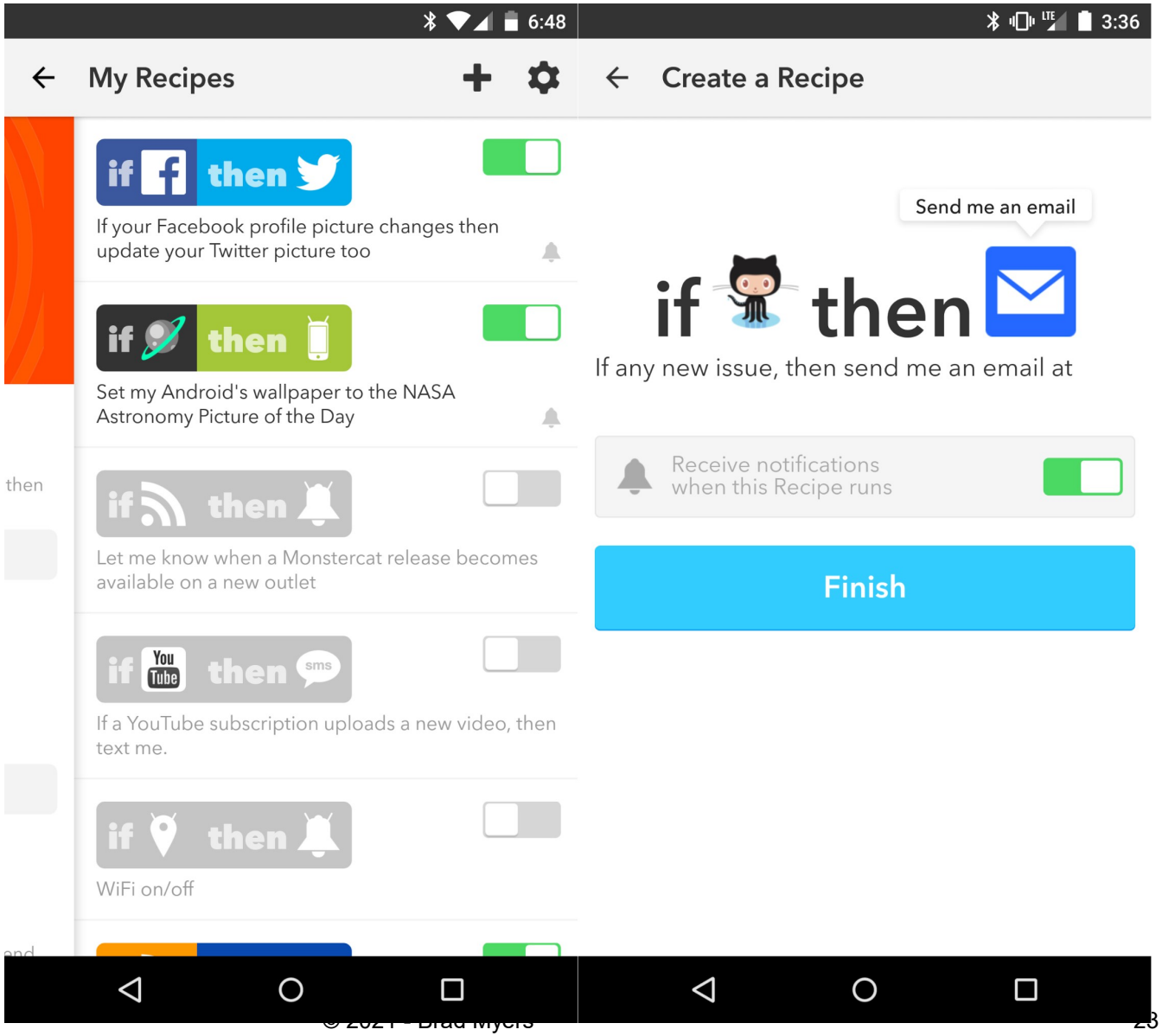


IFTTT.com



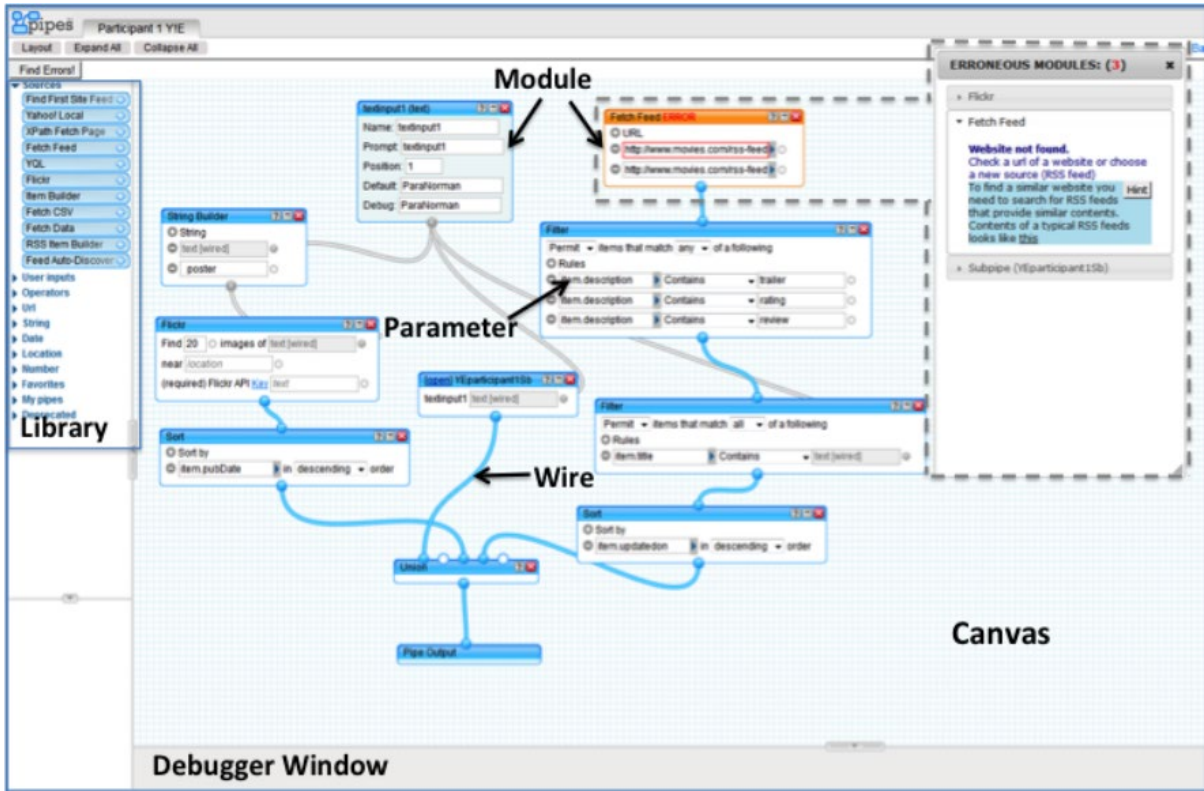
- Founded 2011
- "If this then that"
  - Condition-action rules (same as stimulus-response)
  - Web-based conditions
  - Often used with Internet of Things (IoT), "smart home" appliances
  - New services added with Ruby programming
- Single condition and action

IFTTT


6:48
My Recipes
if then
If your Facebook profile picture changes then update your Twitter picture too
if then
Set my Android's wallpaper to the NASA Astronomy Picture of the Day
if then
Let me know when a Monstercat release becomes available on a new outlet
if then
If a YouTube subscription uploads a new video, then text me.
if then
WiFi on/off
3:36
Create a Recipe
Send me an email
if then
If any new issue, then send me an email at
Receive notifications when this Recipe runs
Finish

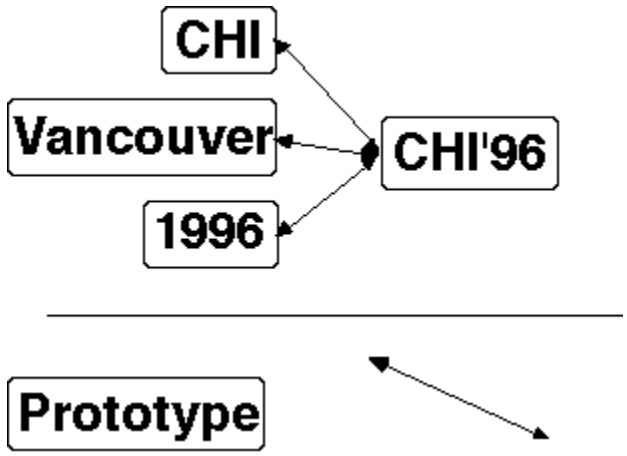
Yahoo! Pipes (2007 – 2015)

- Was a web application to process data feeds on the web
  - Originally focused on “RSS feeds”
- Visual data flow architecture, like LabVIEW
- Studies showed wasn’t easy for non-programmers to use
- Sandeep K Kuttal, A. Sarma, and G. Rothermel, "Debugging Support for End-User Mashup Programming", in *Proceedings of Computer and Human Interactions - CHI*, Paris, France,pages 1609 - 1618, April 2013.[pdf]
  - Issues with connections, parameters, debugging, etc.
- *Video (1:50) or tutorial (5:15)*



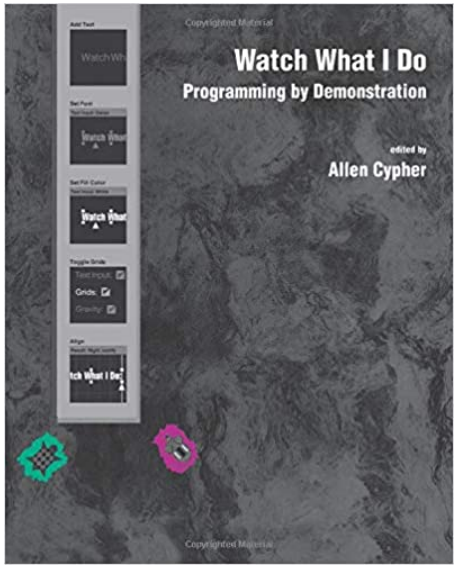
Another Approach: PBD

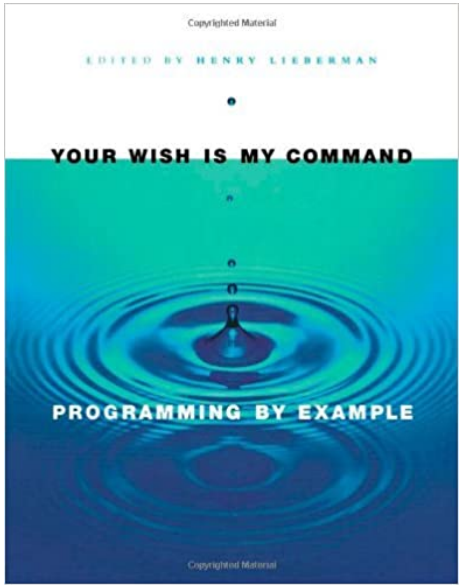
- Programming by Demonstration (PBD)
  - Also: Programming by Example (PBE)
- Give *examples* of desired input and output
  - Or of desired behavior
- For example:
  - Learns that size of boxes should match text from these examples
  - Arrows stay attached
- Like Machine Learning (ML) but only a few examples
  - E.g., gesture learning from 15 examples



Demonstrational Interfaces

- "Classic" Reference: Allen Cypher, ed. *Watch What I Do*, MIT Press. 1993.
- Later book: Henry Lieberman, ed. *Your Wish is My Command*. 2001: Morgan Kaufmann.
- My group has chapters in both



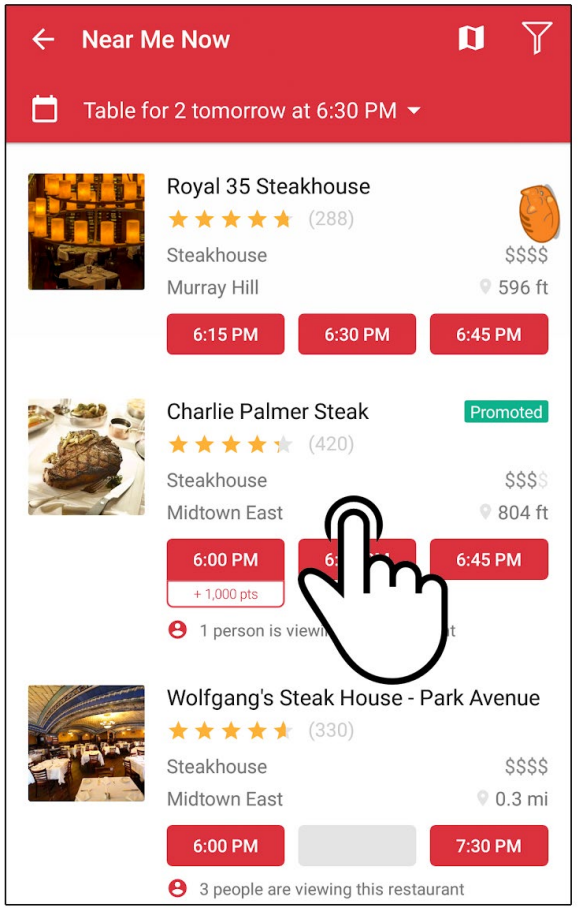


Motivation

- Demonstrational techniques expand how much of the interface can be specified interactively.
- And Interactive editors are much faster to use than programming with toolkits
  - Frameworks improve productivity by factors of 3 to 5, interactive tools by factors of 10 to 50!
  - It might take an hour to draw an interface interactively, compared to days to program it.
  - Because they are faster, this promotes rapid prototyping
- It is much more natural to specify the graphical parts of applications using a graphical editor.
- Because they do not require programming skills, graphic designers can design the graphical parts of the interface.

Key Challenges

- "Data description problem"
  - What does the reference mean?
    - Charlie Palmer Steak
    - The least expensive steakhouse near me
    - The closest one in Midtown East
    - The one with 1,000 bonus points
    - A promoted restaurant
    - The second restaurant in the list
    - ….
- (Operator is usually easy – like "click")
- Control structures
  - Conditionals and loops

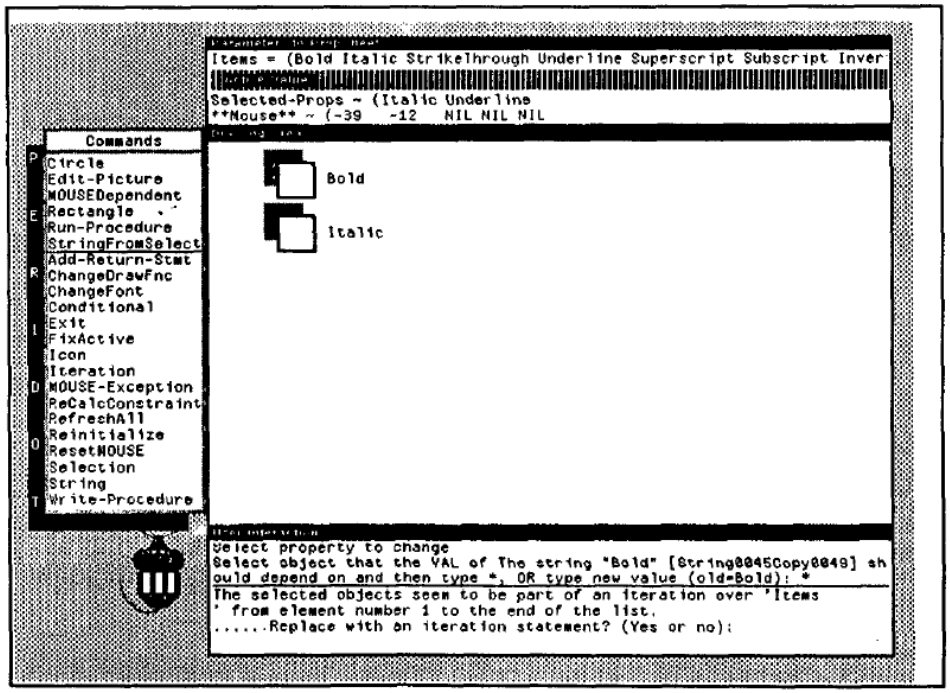


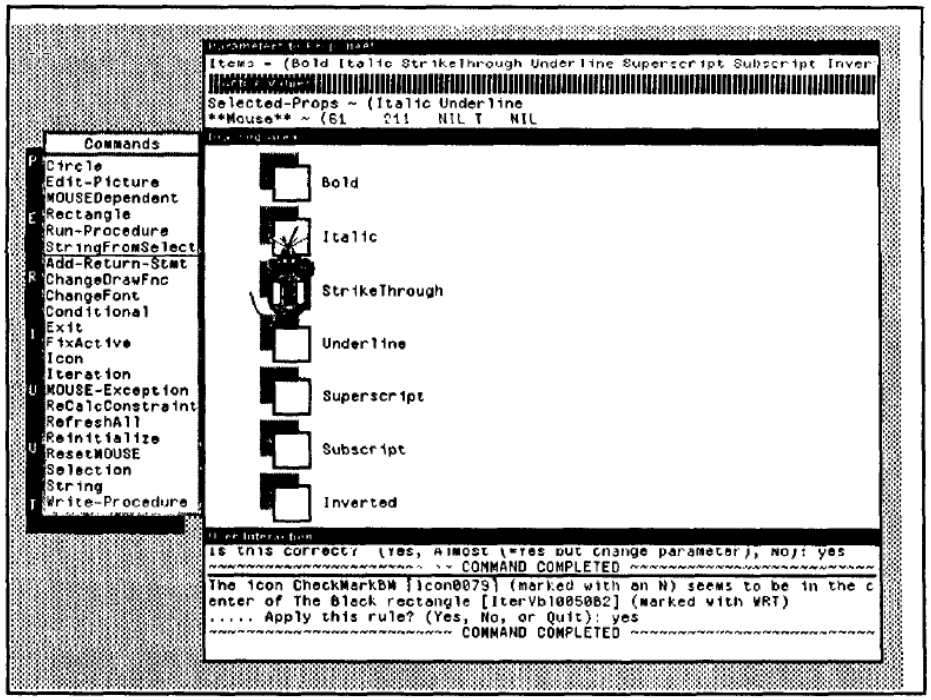
Examples (of uses to create UIs)

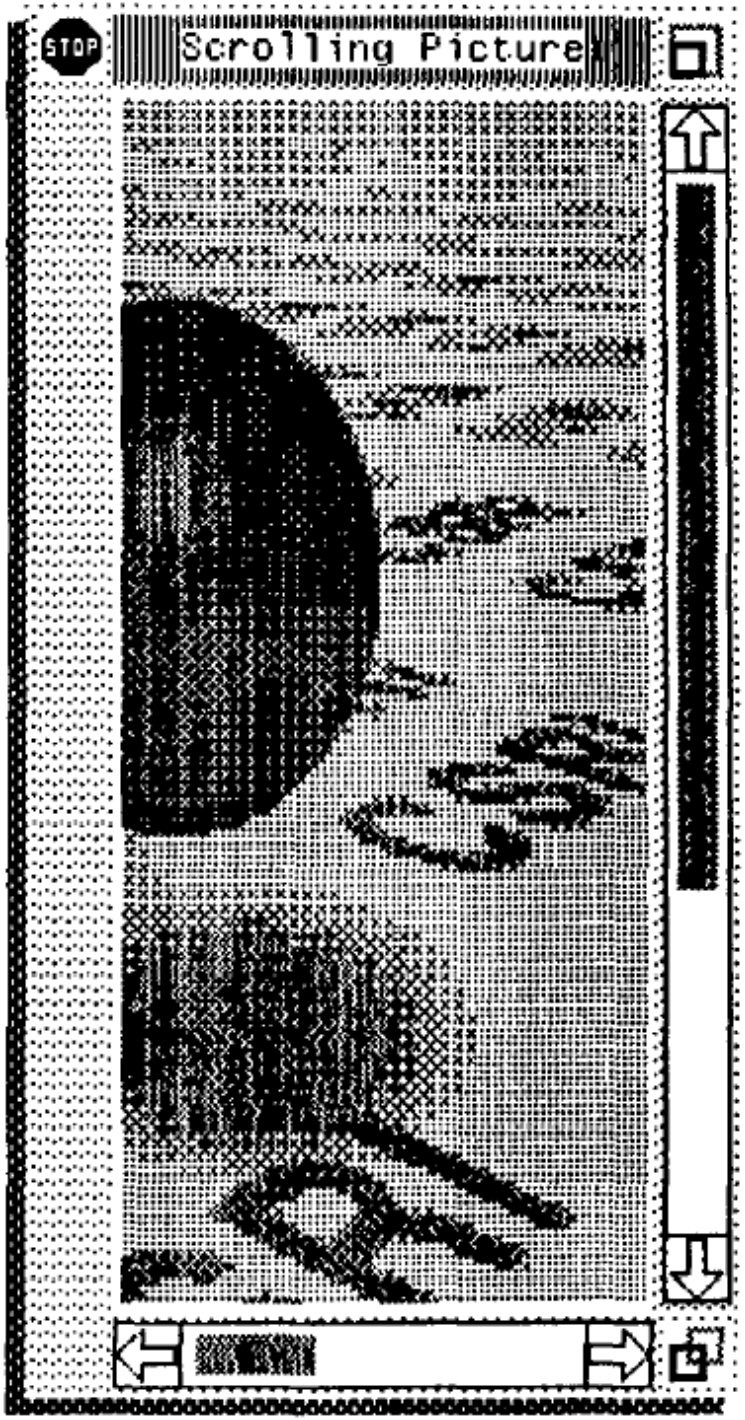
- (chronological order)

Peridot (1986-88)

- Myers B. "Creating User Interfaces Using Programming-by-Example, Visual Programming, and Constraints," ACM Transactions on Programming Languages and Systems. vol. 12, no. 2, April, 1990. pp. 143-177. (Peridot)
- Myers B., Creating User Interfaces by Demonstration, Academic Press, San Diego, 1988.
- Myers B., "Creating Interaction Techniques by Demonstration," IEEE Computer Graphics and Applications, Vol. 7, No. 9, IEEE, September 1987, pp. 51 - 60.
- First demonstrational tool, and it used by-example techniques to allow the creation of new widgets.
- From the drawings, it infers:
  - Graphical constraints among the objects, such as that the boxes should be the same size as the text.
  - control structures such as iteration over all the items in a menu
  - how the mouse affects the graphics, such as that the check mark should follow the mouse.
- feedback: question and answer
- video (8 min)


Items = (Bold Italic StrikeThrough Underline Superscript Subscript Inver
Selected-Props ~ (Italic Underline
Mouse ~ (-39 -12 NIL NIL NIL
Commands
Circle
Edit-Picture
MOUSEDependent
Rectangle
Run-Procedure
StringFromSelect
Add-Return-Stmt
ChangeDrawFnc
ChangeFont
Conditional
Exit
FixActive
Icon
Iteration
MOUSE-Exception
ReCalcConstraint
RefreshAll
Reinitialize
ResetMOUSE
Selection
String
Write-Procedure
Bold
Italic
Select property to change
Select object that the VAL of The string "Bold" [String0045Copy0049] sh
ould depend on and then type *, OR type new value (old=Bold): *
The selected objects seem to be part of an iteration over 'Items
' from element number 1 to the end of the list.
.....Replace with an iteration statement? (Yes or no):


Items = (Bold Italic StrikeThrough Underline Superscript Subscript Inver
Selected-Props ~ (Italic Underline
Mouse ~ (61 211 NIL T NIL
Commands
Circle
Edit-Picture
MOUSEDependent
Rectangle
Run-Procedure
StringFromSelect
Add-Return-Stmt
ChangeDrawFnc
ChangeFont
Conditional
Exit
FixActive
Icon
Iteration
MOUSE-Exception
ReCalcConstraint
RefreshAll
Reinitialize
ResetMOUSE
Selection
String
Write-Procedure
Bold
Italic
StrikeThrough
Underline
Superscript
Subscript
Inverted
Is this correct? (Yes, Almost (=Yes but change parameter), No): yes
~~~~~~~~~~~~~~~~~~~~ COMMAND COMPLETED ~~~~~~~~~~~~~~~~~~~~
The icon CheckMarkBW]Icon0079) (marked with an N) seems to be in the c
enter of The Black rectangle [IterVbl0050B2] (marked with WRT)
..... Apply this rule? (Yes, No, or Quit): yes
~~~~~~~~~~~~~~~~~~~~ COMMAND COMPLETED ~~~~~~~~~~~~~~~~~~~~

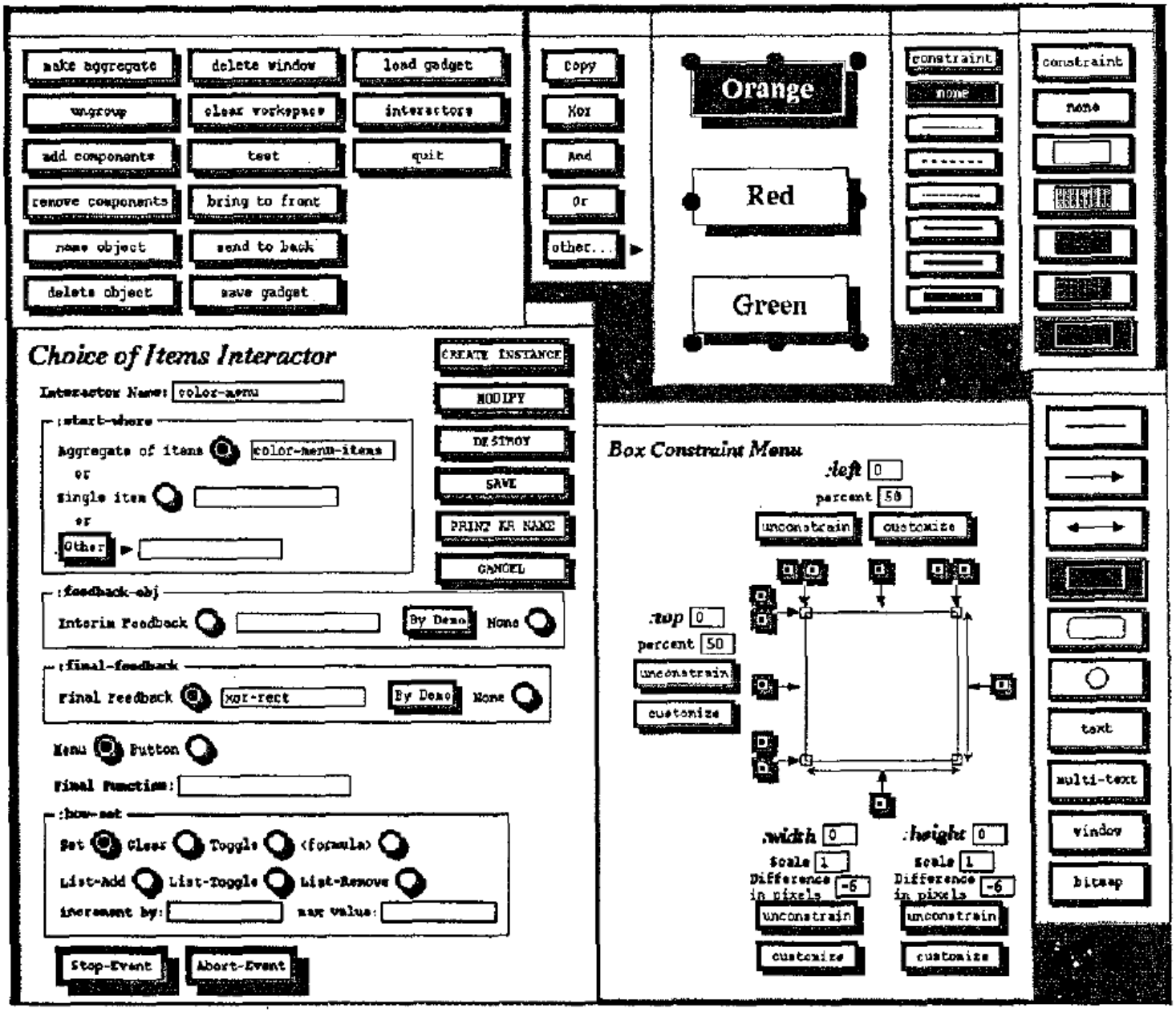

STOP
Scrolling Picture

Myers

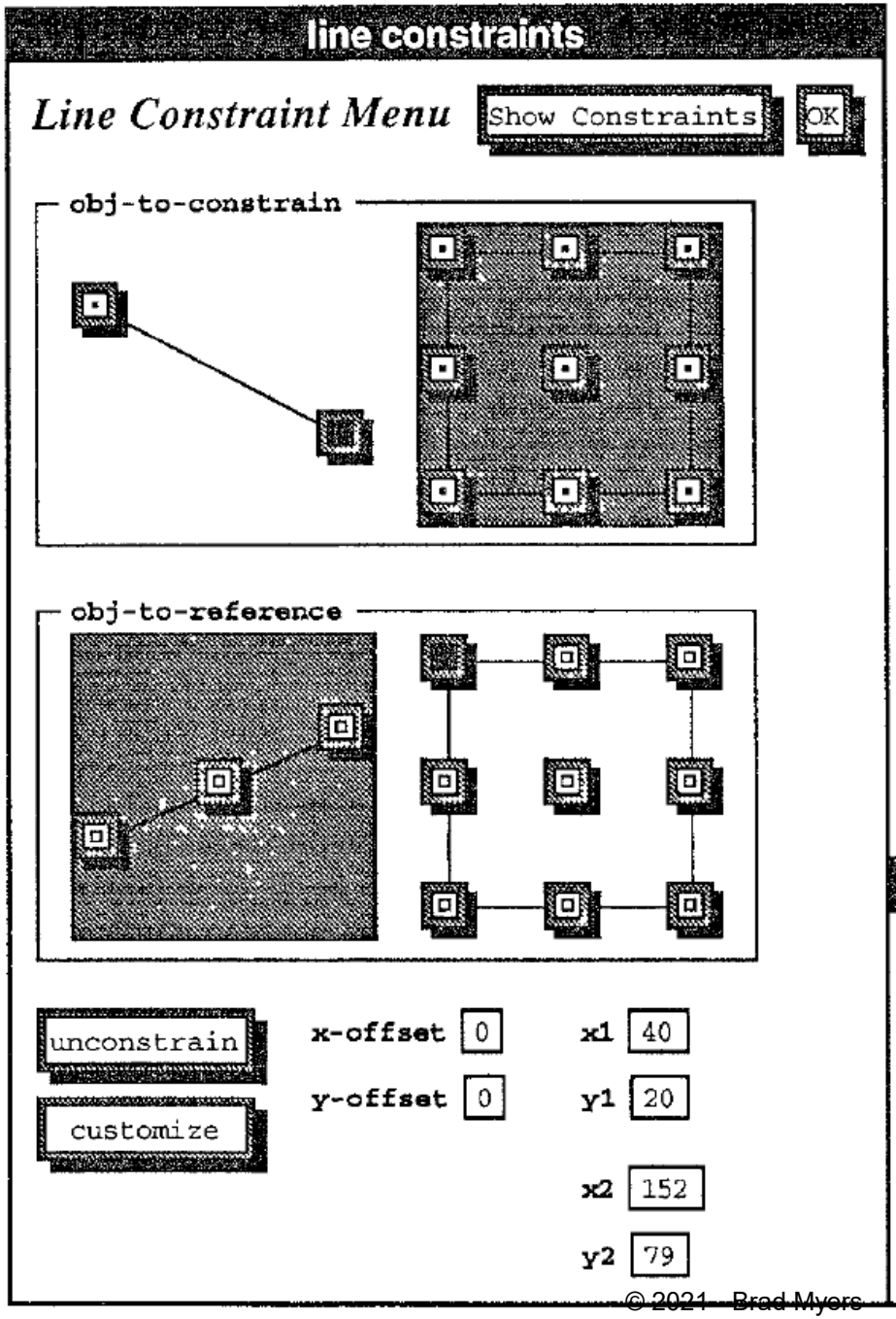
Lapidary (1989-1993)

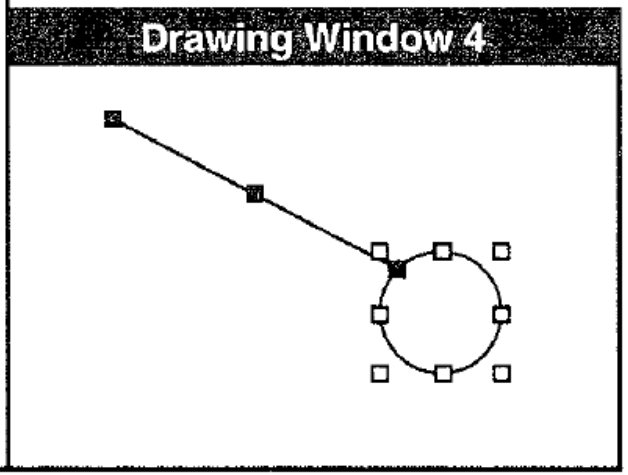
- Myers B., Vander Zanden B. and Dannenberg R., "Creating Graphical Interactive Application Objects by Demonstration," *Proceedings of the ACM Symposium on User Interface Software and Technology, UIST'89*, Williamsburg, November 1989, pp. 95 - 104.
- Brad Vander Zanden and Brad A. Myers. "Demonstrational and Constraint-Based Techniques for Pictorially Specifying Application Objects and Behaviors," *ACM Transactions on Computer-Human Interaction*. vol. 2, no. 4, Dec, 1995. pp. 308-356.
- Extends Peridot to allow the creation of application-specific graphical objects, like nodes in a graphics editor.
- Uses less inferencing and more dialog boxes
- Is "real" and you get it as part of the Garnet distribution
- Problems:
  - can only demonstrate "syntactic" parts of application
  - hard to set up correct constraints
- *video (12 min)*

Lapidary dialog boxes

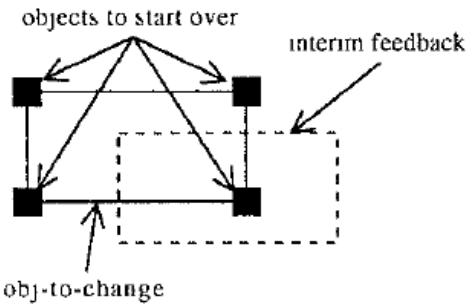


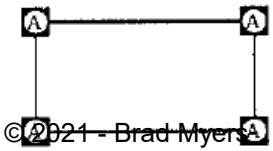
Lapidary, cont.

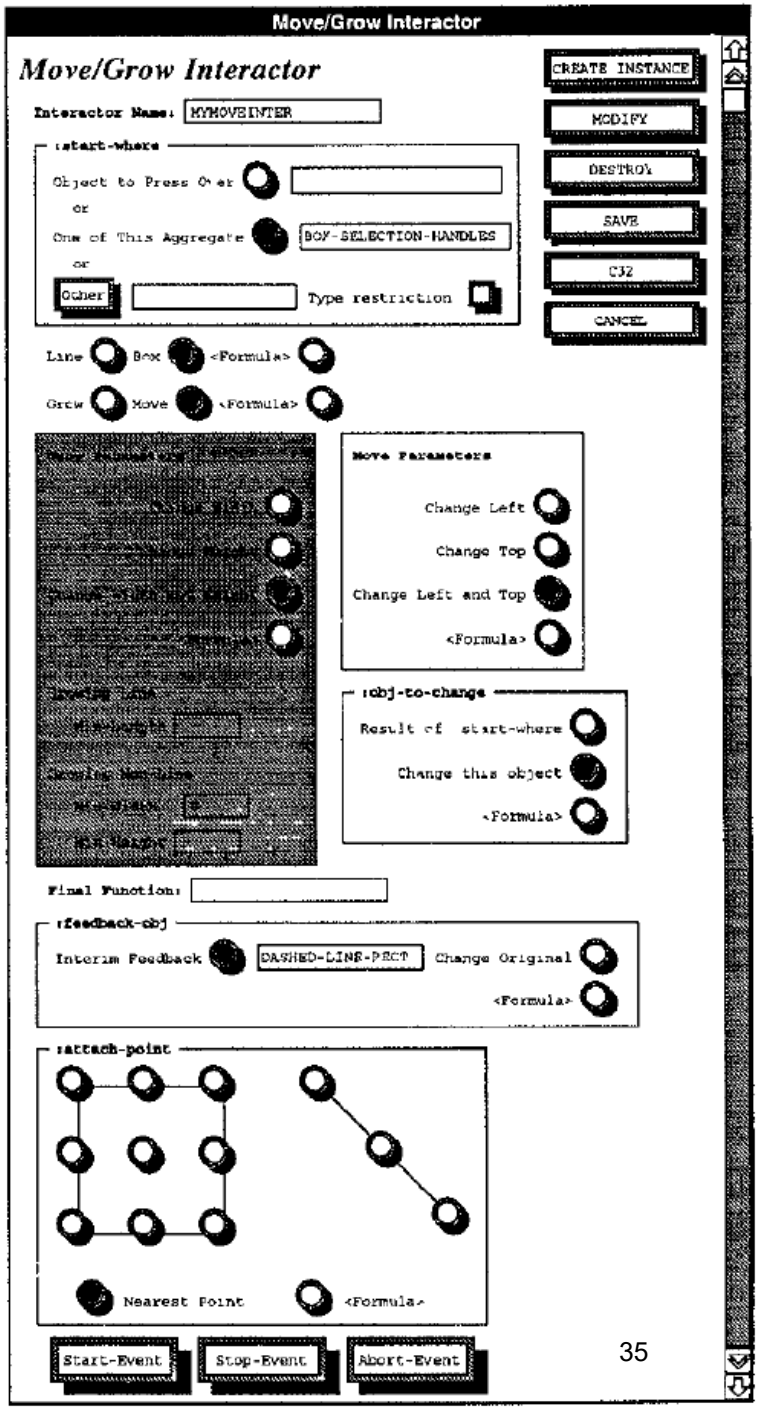




Lapidary, cont.



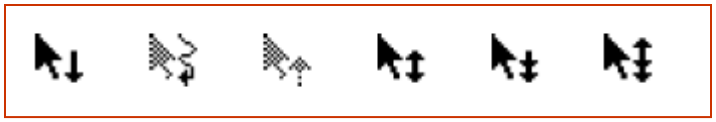


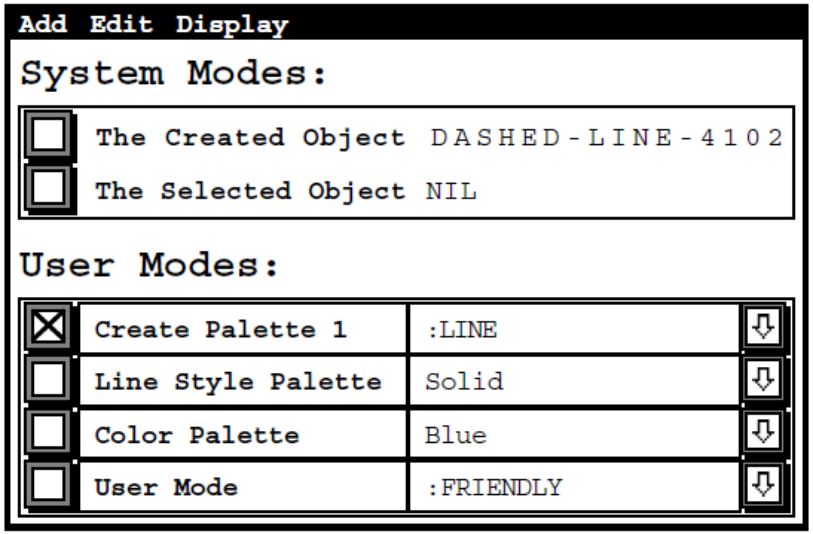


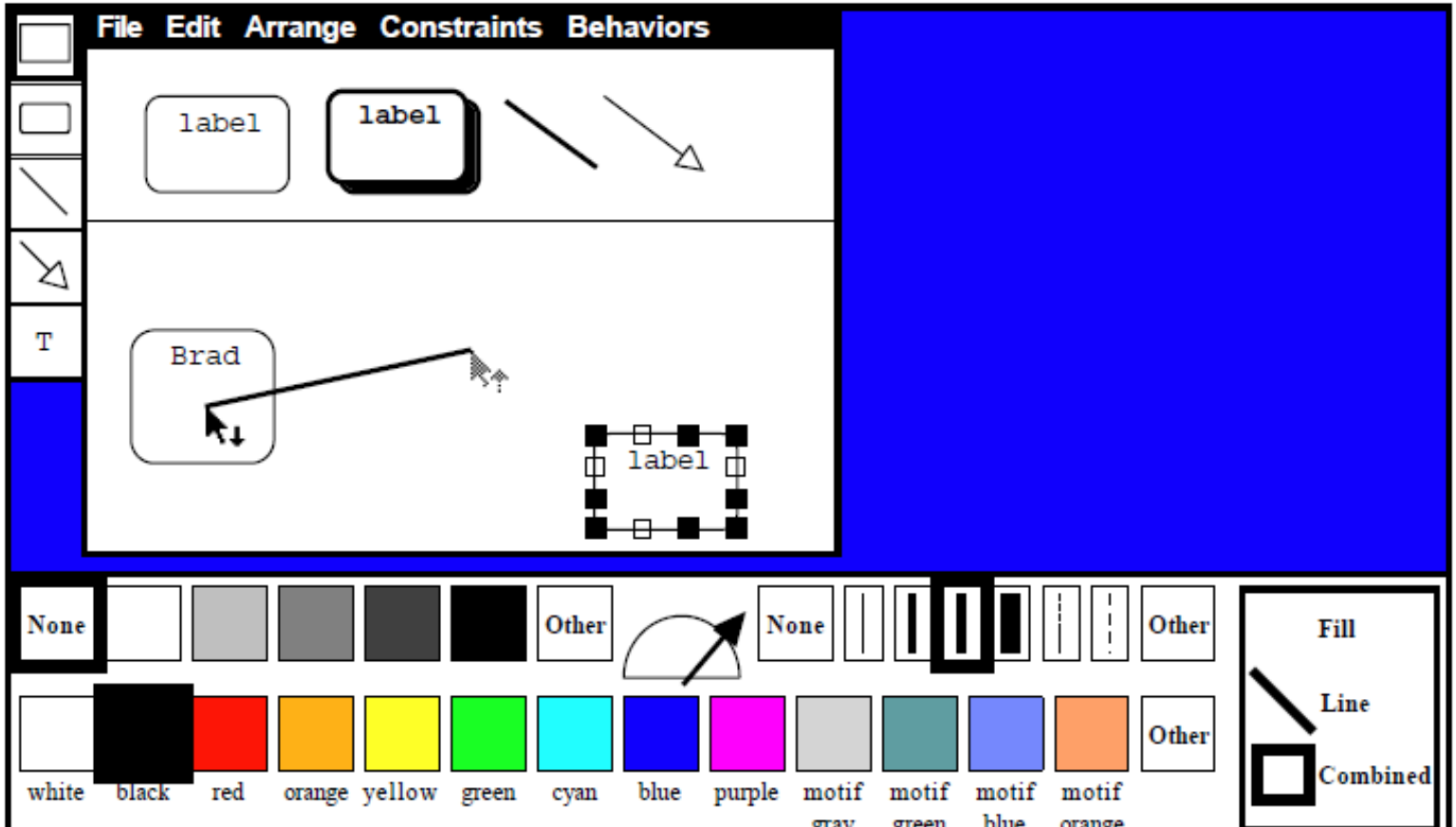
Marquise (1993-1994)

- Myers B., McDaniel, R. and Kosbie, D.. "Marquise: Creating Complete User Interfaces by Demonstration," *Proceedings CHI'94: Human Factors in Computing Systems*. Amsterdam, The Netherlands, April 24-29, 1993. pp. 293-300.
- Go back to doing more by demonstration, and just show the way that the interface should operate.
- In particular, demonstrate *when* the behaviors should start and what the feedback looks like.
  - mouse button does one of 10 things, depending on where press and global mode.
- Demonstrate both behavior and conditions
- Built-in support for palettes and modes.

Marquise windows

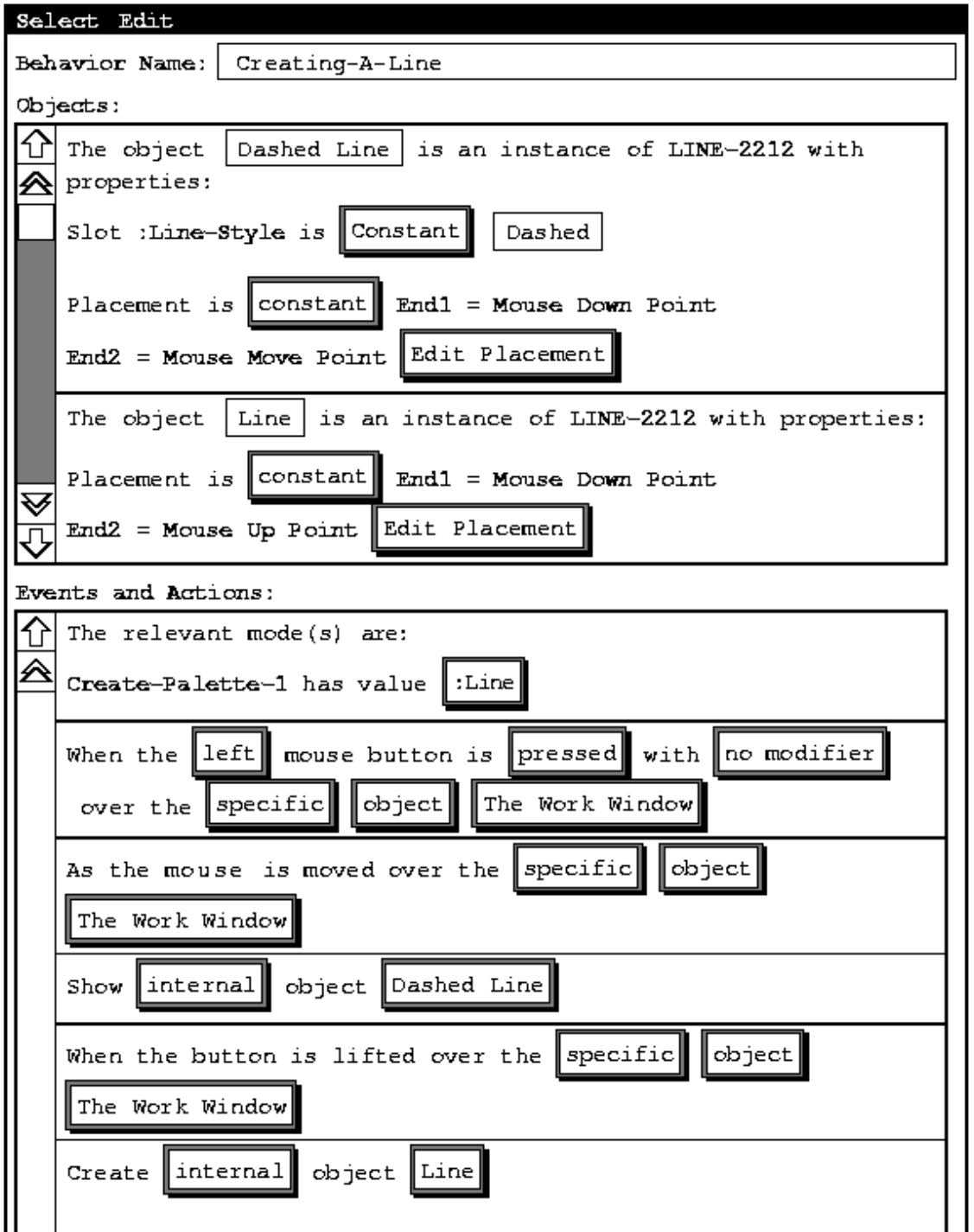






Marquise feedback window

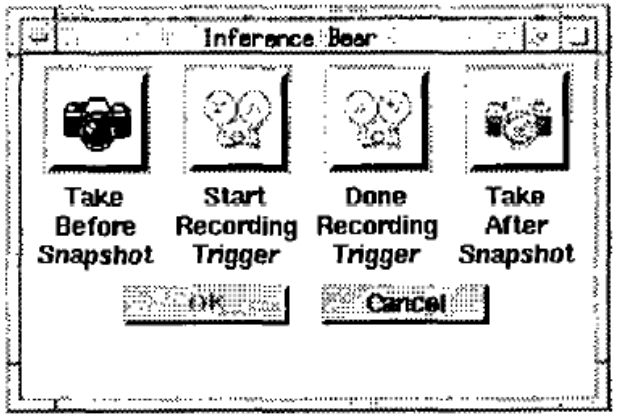
- *video (12 min)*

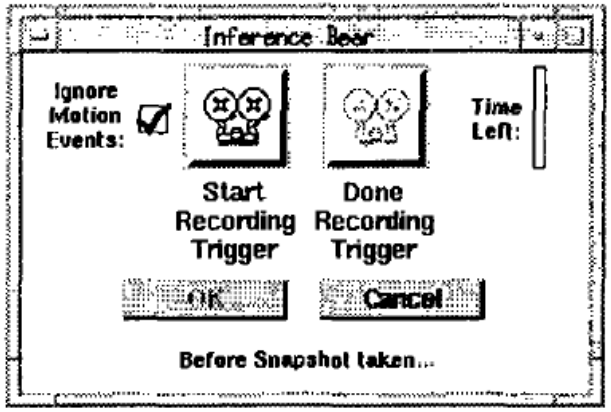


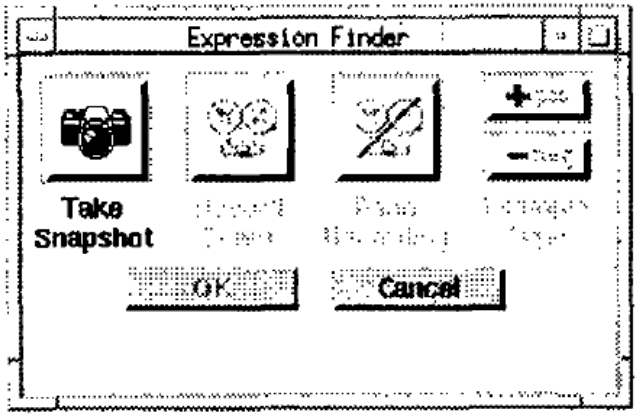
InferenceBear & Grizzly Bear (1994-1996)

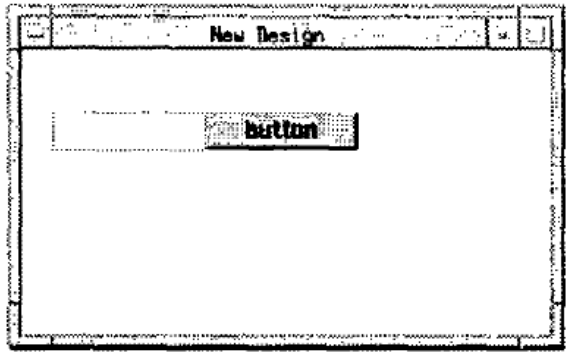
- Martin R. Frank, Piyawadee "Noi" Sukaviriya, James D. Foley. “Inference bear: designing interactive interfaces through before and after snapshots,” DIS’95. Ann Arbor, Michigan, pp. 167 – 175. pdf
- Martin Frank, Model-Based User Interface Design by Demonstration and By Interview. PhD Thesis, Georgia Tech, 1996.
- (Discussed his "Elements, Events & Transitions (EET) language in the event-language lecture)
- User control through dialog boxes, edit using textual language: EET
- Snapshots of before and after
- Multiple examples
  - More positive examples to cause generalization
  - Negative examples to specify exceptions
- Pictures – next slide

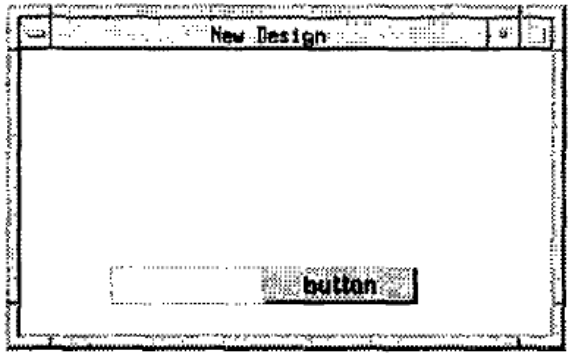
InferenceBear Pictures

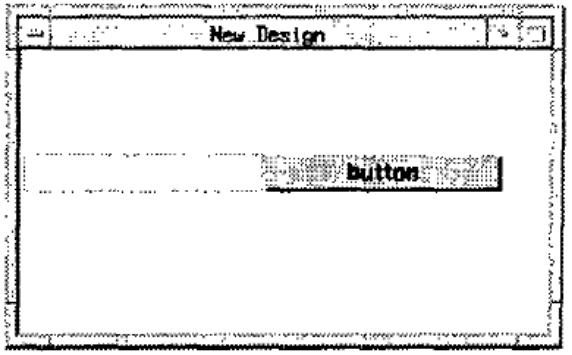


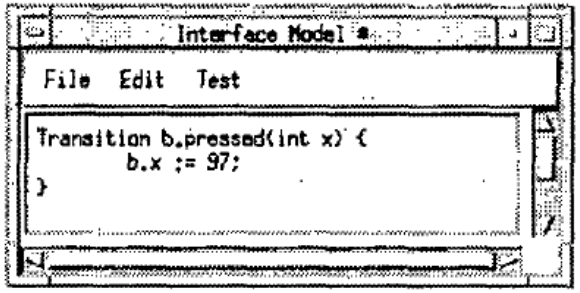


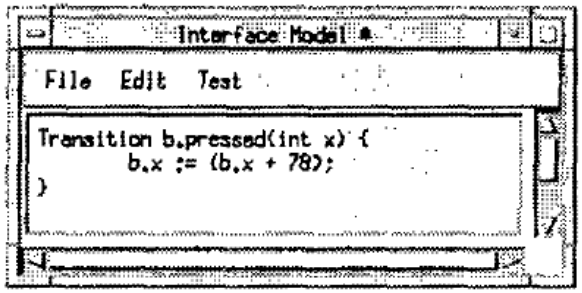


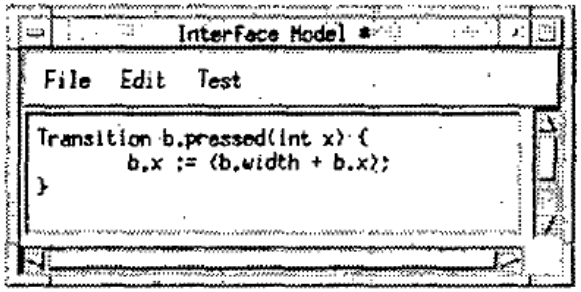






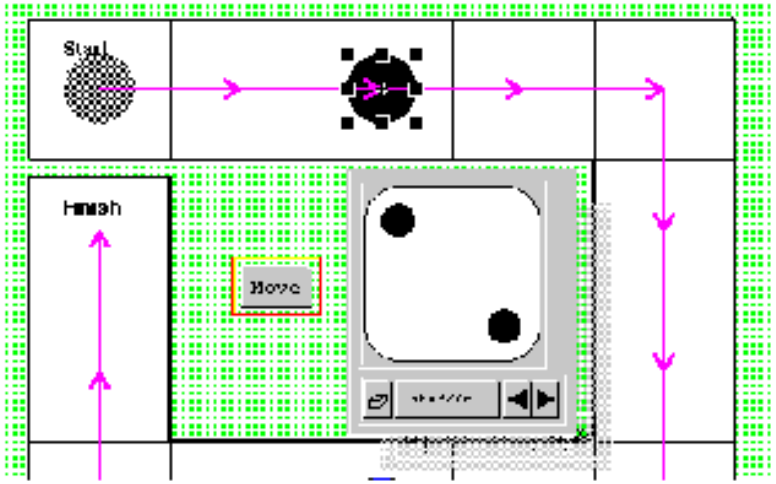






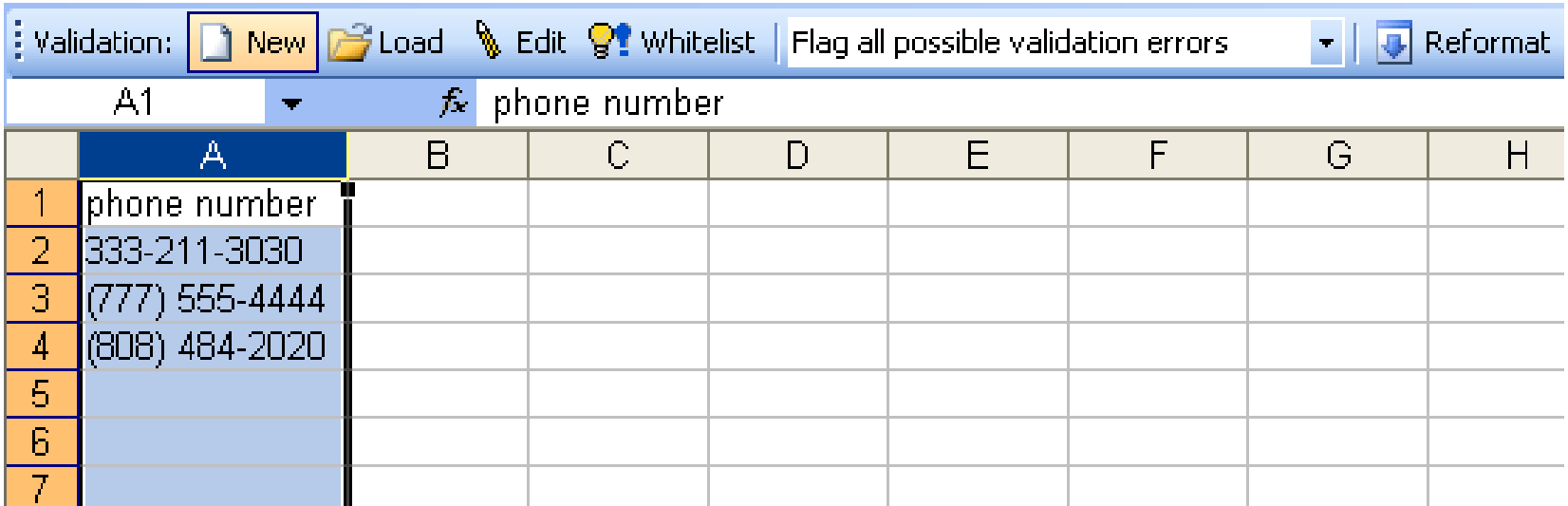
Gamut (1996 - 1999)

- PhD thesis of Rich McDaniel.
- Richard G. McDaniel and Brad A. Myers. "Building Applications Using Only Demonstration," IUI'98: 1998 International Conference On Intelligent User Interfaces, January 6-9, 1998, San Francisco, CA. pp. 109-116. pdf
- Richard G. McDaniel and Brad A. Myers, "Getting More Out Of Programming-By-Demonstration." Proceedings CHI'99: Human Factors in Computing Systems. Pittsburgh, PA, May 15-20, 1999. pp. 442-449. ACM DL Reference
- Domain: "board games" and educational software
- Goal: new interaction techniques so can infer more complex behaviors
- E.g., how a piece can move in Monopoly / Chess
- Reduce number of modes
- New interaction techniques to provide hints
  - "Do Something!", "Stop That", Hint highlighting, Temporal Ghosts, Guide objects, Deck of Playing Cards, etc.
- Better inferencing algorithms
- video (4.5 min)



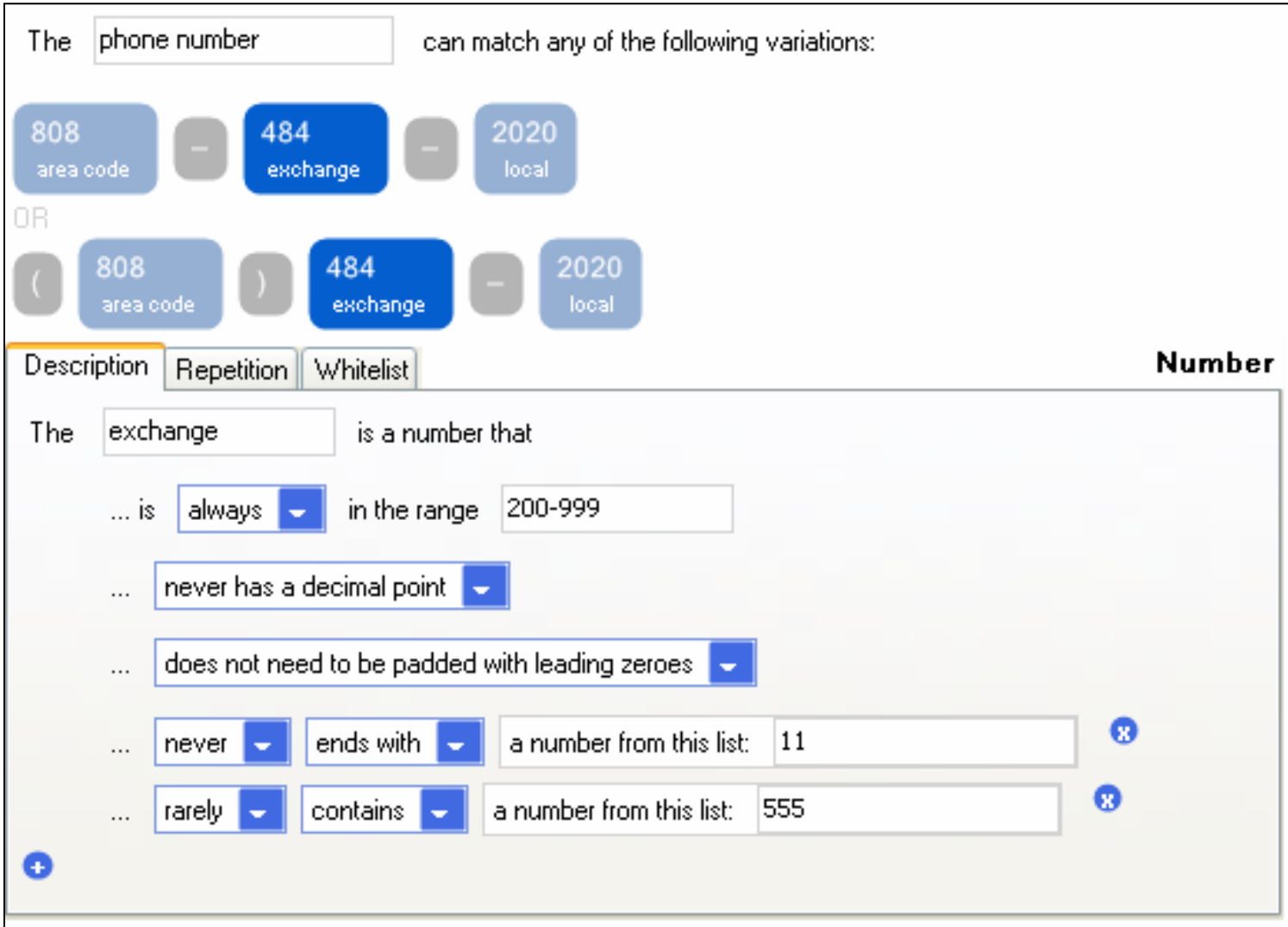
Topes (2004-2009)

- Chris Scaffidi's PhD thesis: 2009
  - Christopher Scaffidi, Brad Myers, Mary Shaw, "Topes: Reusable Abstractions for Validating Data." *ICSE'2008: 30th International Conference on Software Engineering*, Leipzig, Germany, 10 - 18 May 2008. pp. 1-10. IEEE DL pdf
- "topes" = user-level types for end-user programming (EUP)
- Create parsers, data-transformations
  - Infers topes from a list of examples
- Patterns in text input
  - Phone numbers, addresses, social security numbers, etc.



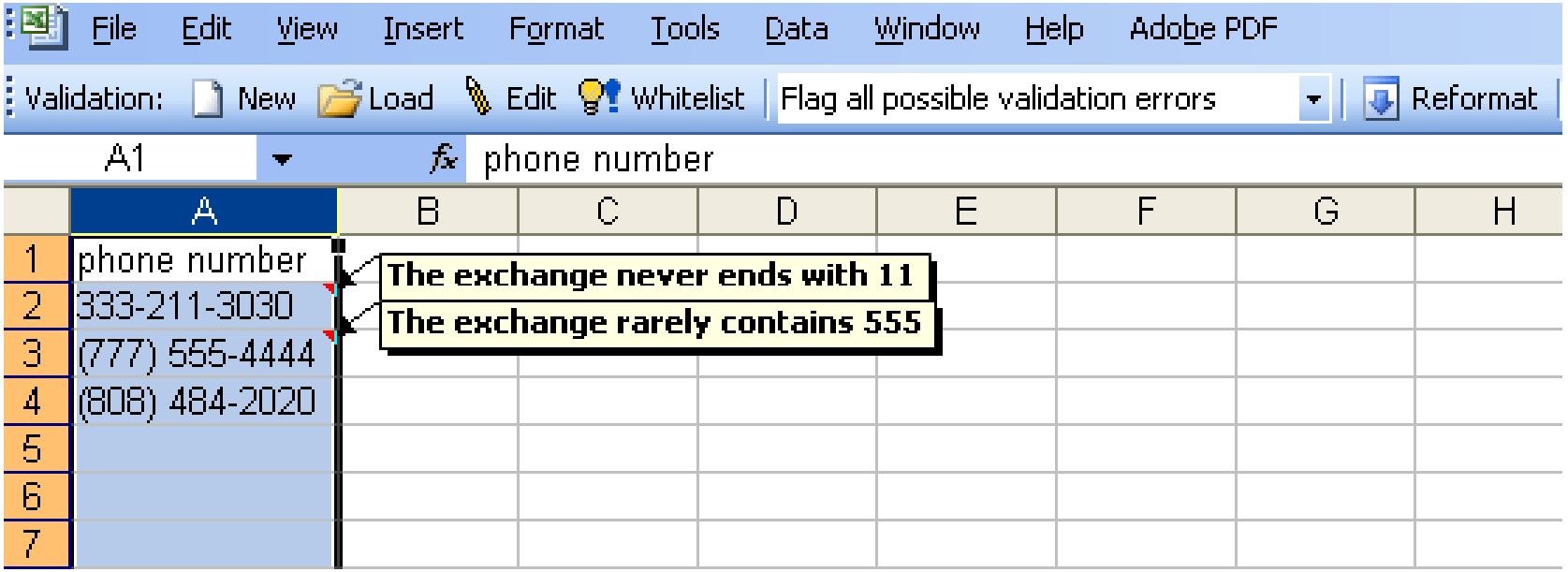
Topes, cont.

- Inferred pattern



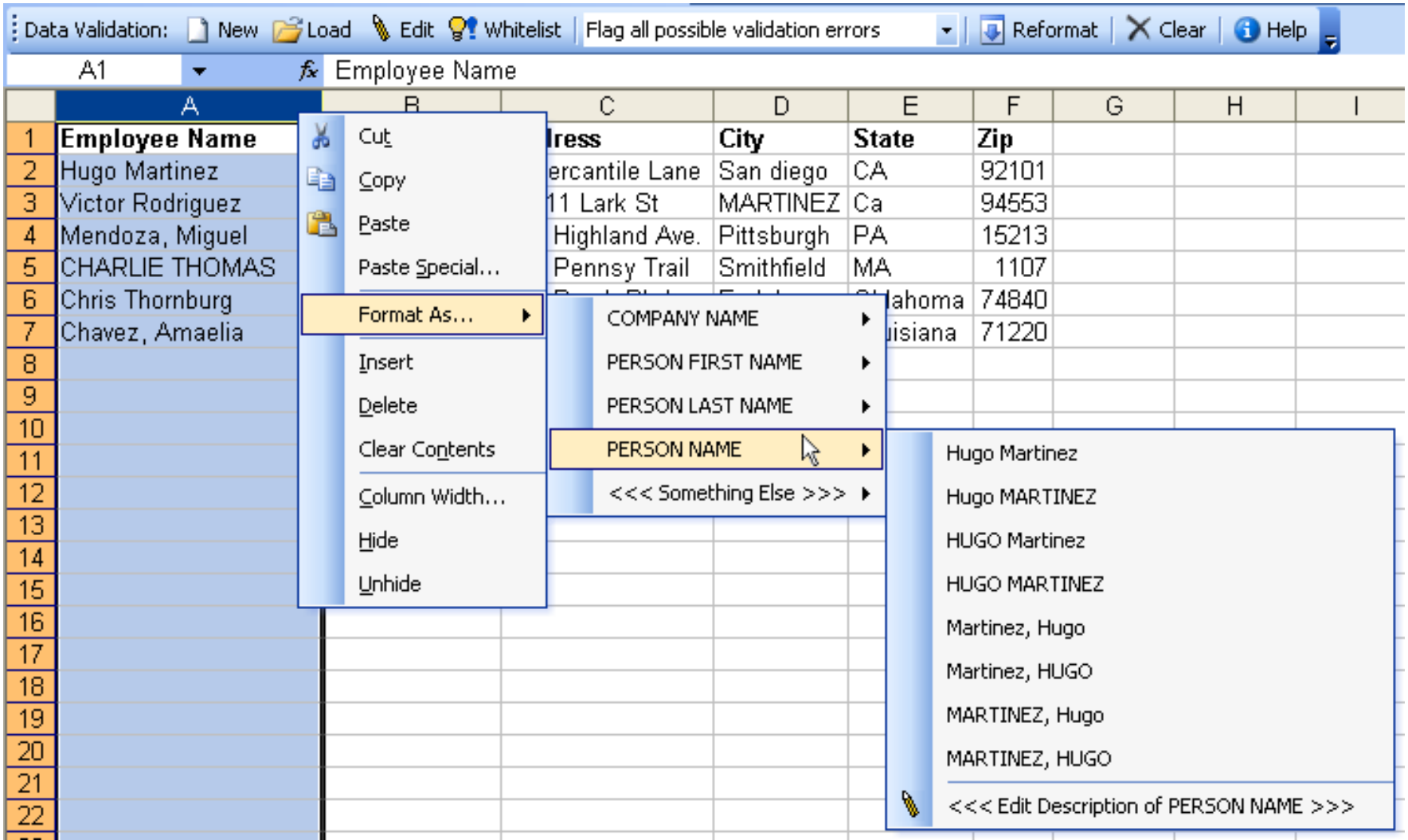
Topes, cont.

- Validator – never vs. rarely



Topes, cont.

- Converter



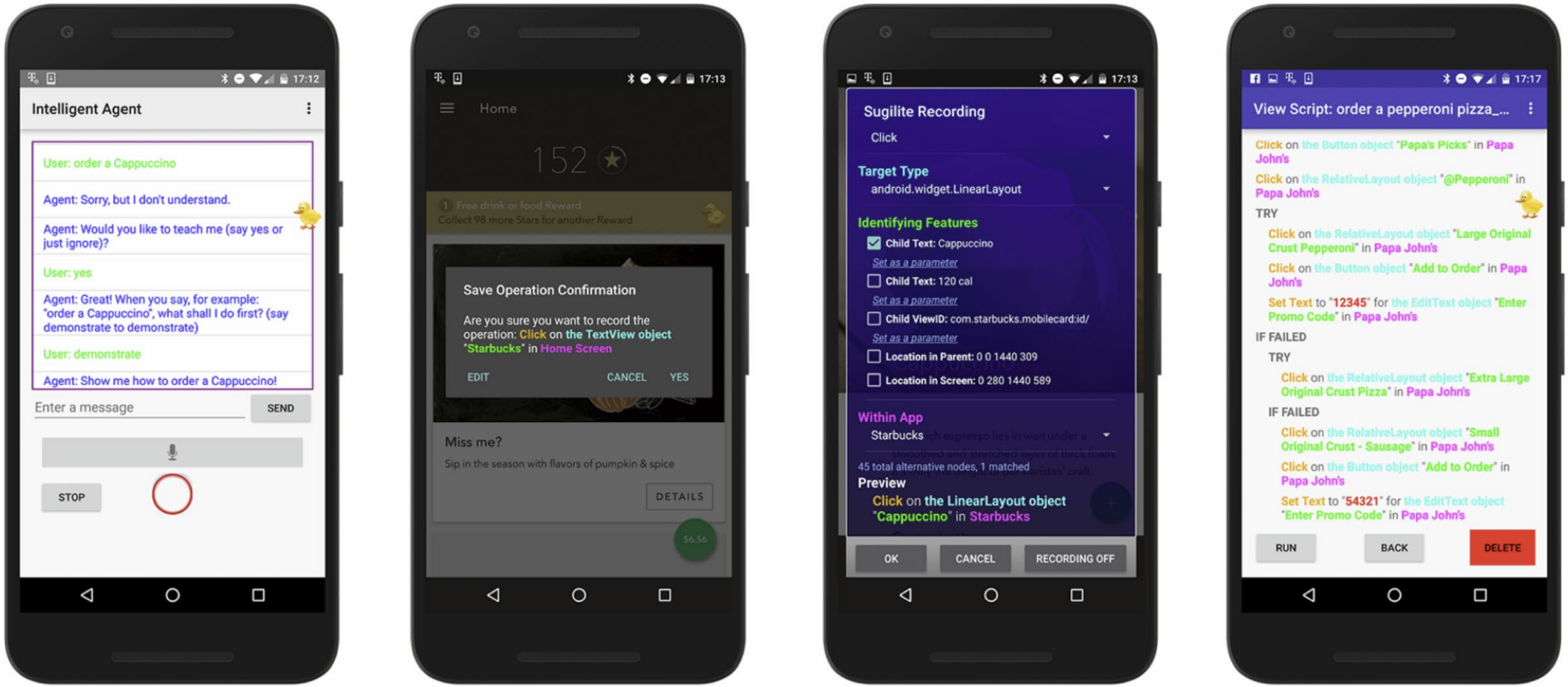
Draco, Skuid

- Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). 351-360. DOI: https://doi.org/10.1145/2556288.2556987
- Sketch to show animations and movements
- Augmented with dynamic animation effects
- Commercialized by AutoDesk
- *Video (4:57)!*

Toby Li's Sugilite

- Toby Li, Amos Azaria, and Brad Myers. "SUGILITE: Creating Multimodal Smartphone Automation by Demonstration", *Proceedings CHI'2017: Human Factors in Computing Systems*, Denver, CO, May 6-11, 2017. To appear. preprint pdf and video. **Best paper Honorable Mention award**.
- Programming by example for Android
- Scripts (macros) of common or repetitive tasks
- Uses the Android accessibility API
- Invoke using Speech or GUI
- Generalizes based on other menu items seen
- Currently, uses multiple examples only when script fails
  - Can replace or add fork
- *Video (6:48)*

Sugilite pictures



(a) (b) (c) (d)

Commercial Systems

- Excel Flashfill
- Adobe Catalyst
  - Create menus by giving examples of the items
  - Scroll bars by indicating the parts (thumb, track, etc.)
  - But discontinued ☹
- Adobe XD
  - Repeat grid
  - Component behaviors ??
- What else?

General Disadvantages of PBD

- People are actually **not** very good at coming up with concrete examples
  - examples tend to show the system the same thing over and over
  - people can't think of the edge cases and negative examples
- People need to be able to **edit** the code, so need a representation they can understand

Open Issues with PBD

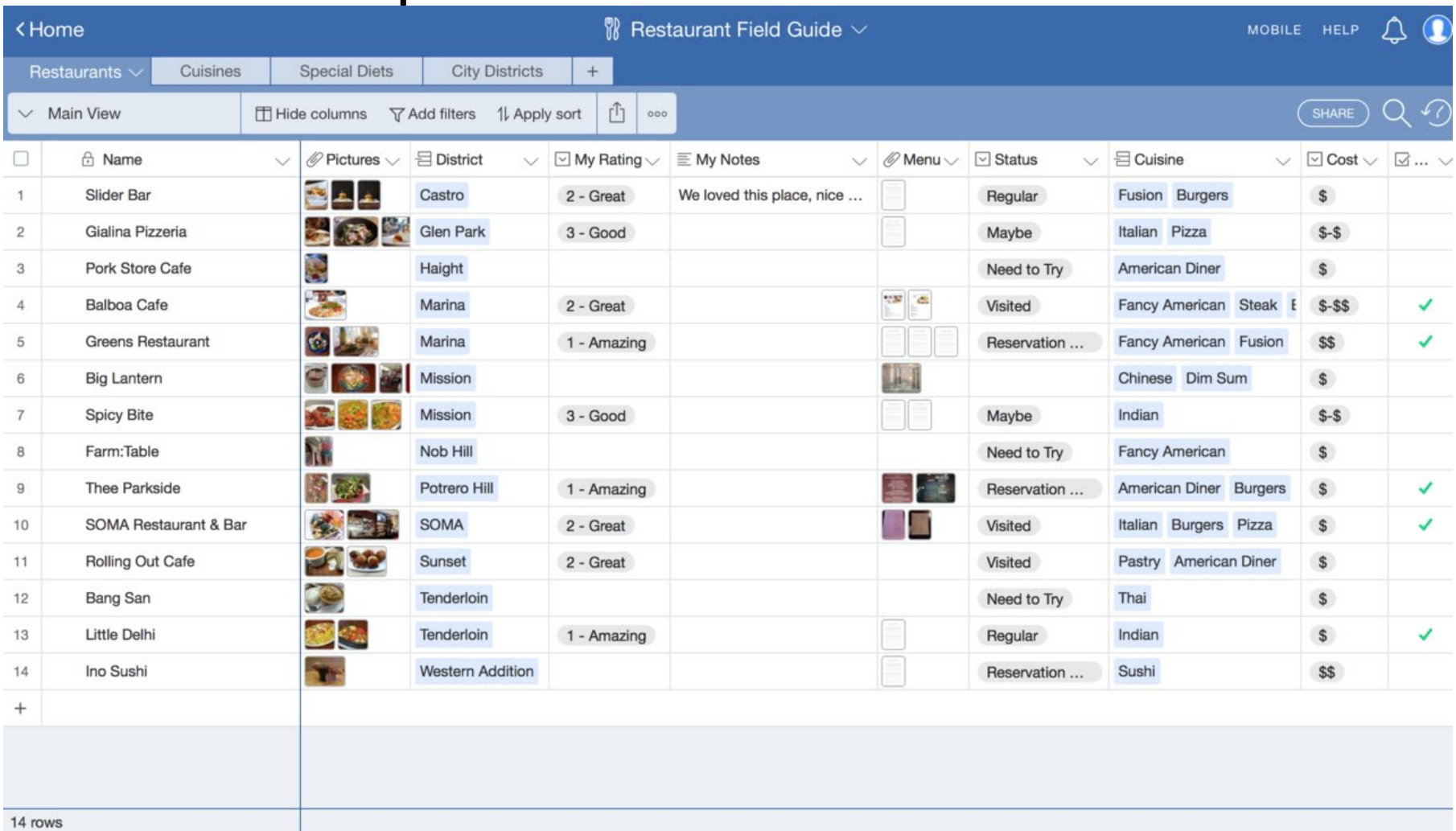
- Sometimes examples are harder than specifying
  - "and" vs. "or"
- How intelligent is enough?
  - Predictability
  - AI problem
- Techniques for feedback and editing
- Combining inferencing with direct editing of the code
- A "really" successful product using this technology

Some newer systems

- Claim to be: “Low code” or “No code”
  - Examples (all founded in 2012!):
    - AirTable – based on a spreadsheet model
    - Bubble.io – visual programming
    - Zapier – move data between web applications (automate repetitive tasks)
      - Like Yahoo! pipes!

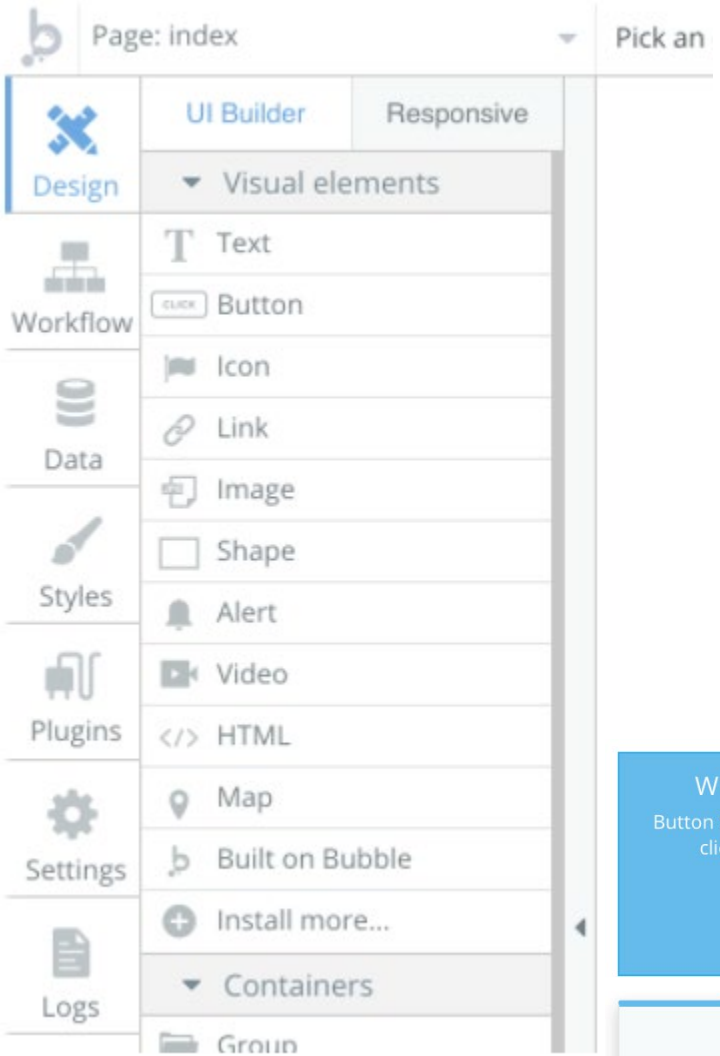
Airtable

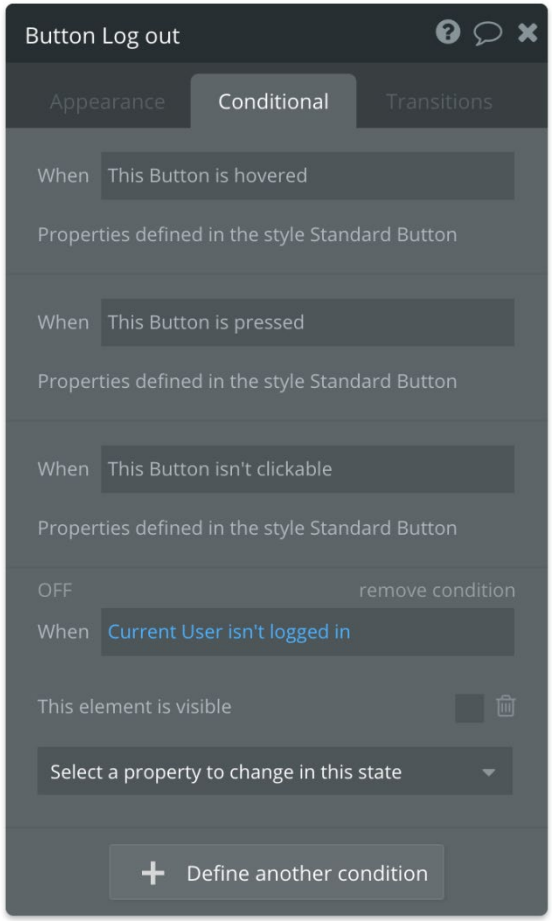
- From Wikipedia:

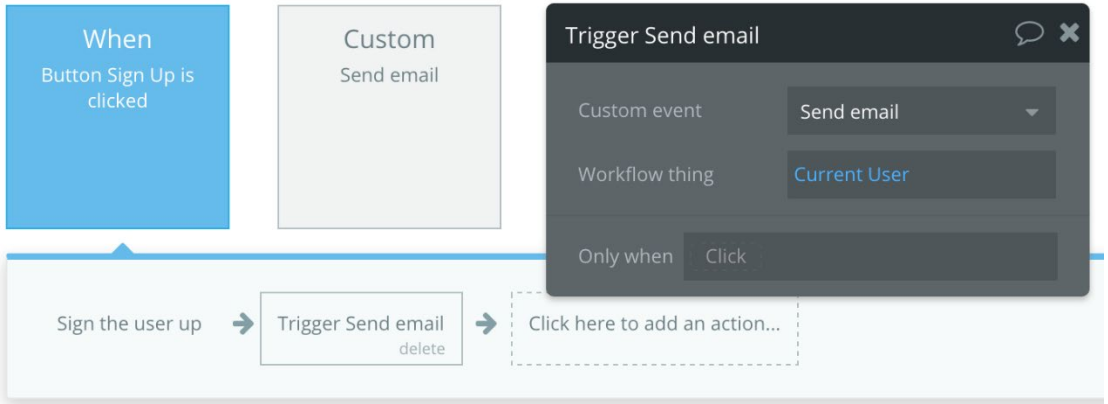


	Name	Pictures	District	My Rating	My Notes	Menu	Status	Cuisine	Cost	...
1	Slider Bar		Castro	2 - Great	We loved this place, nice ...		Regular	Fusion Burgers	$	
2	Gialina Pizzeria		Glen Park	3 - Good			Maybe	Italian Pizza	$-$	
3	Pork Store Cafe		Haight				Need to Try	American Diner	$	
4	Balboa Cafe		Marina	2 - Great			Visited	Fancy American Steak E	$-$$	✓
5	Greens Restaurant		Marina	1 - Amazing			Reservation ...	Fancy American Fusion	$$	✓
6	Big Lantern		Mission					Chinese Dim Sum	$	
7	Spicy Bite		Mission	3 - Good			Maybe	Indian	$-$	
8	Farm:Table		Nob Hill				Need to Try	Fancy American	$	
9	Thee Parkside		Potrero Hill	1 - Amazing			Reservation ...	American Diner Burgers	$	✓
10	SOMA Restaurant & Bar		SOMA	2 - Great			Visited	Italian Burgers Pizza	$	✓
11	Rolling Out Cafe		Sunset	2 - Great			Visited	Pastry American Diner	$	
12	Bang San		Tenderloin				Need to Try	Thai	$	
13	Little Delhi		Tenderloin	1 - Amazing			Regular	Indian	$	✓
14	Ino Sushi		Western Addition				Reservation ...	Sushi	$$	

Bubble.io


Page: index
Pick an
UI Builder
Responsive
Design
Visual elements
Text
Button
Icon
Link
Image
Shape
Alert
Video
HTML
Map
Built on Bubble
Install more...
Containers
Group
Workflow
Data
Styles
Plugins
Settings
Logs


Button Log out
Appearance
Conditional
Transitions
When This Button is hovered
Properties defined in the style Standard Button
When This Button is pressed
Properties defined in the style Standard Button
When This Button isn't clickable
Properties defined in the style Standard Button
OFF
remove condition
When Current User isn't logged in
This element is visible
Select a property to change in this state
Define another condition


When
Button Sign Up is clicked
Custom
Send email
Trigger Send email
Custom event
Send email
Workflow thing
Current User
Only when
Click
Sign the user up
Trigger Send email
delete
Click here to add an action...

Zapier