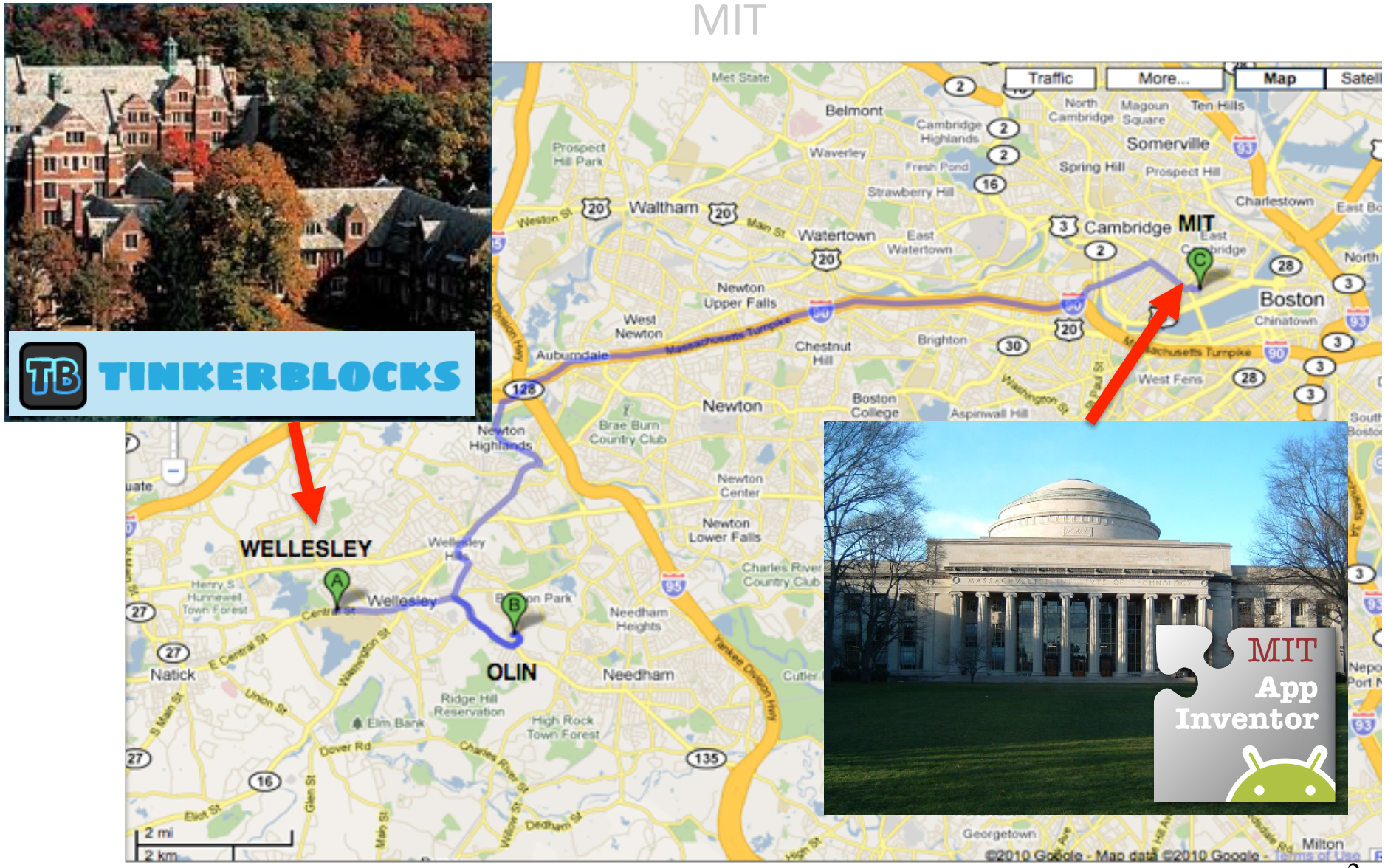
MIT App Inventor: Design and Implementation of a Blocks Programming Language

Franklyn Turbak
Wellesley College Computer Science Dept.

Lewis & Clark College
March 6, 2017

Wellesley & MIT



Talk Road Map

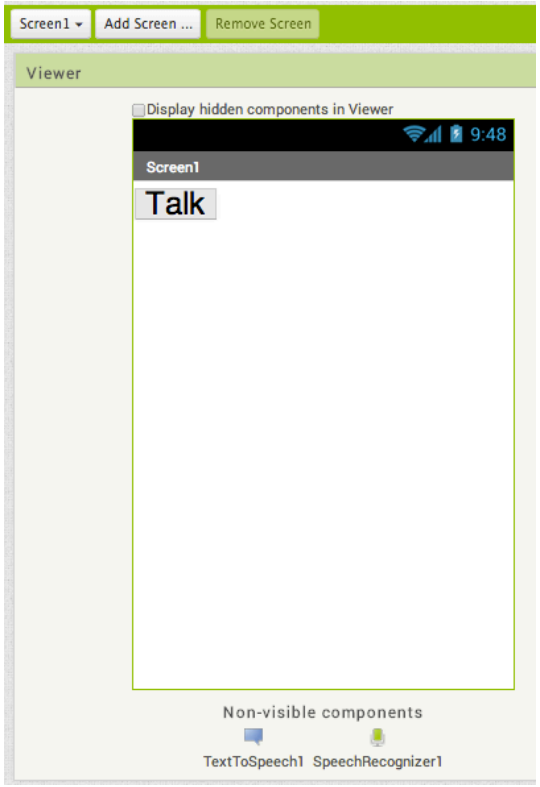
- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Talk Road Map

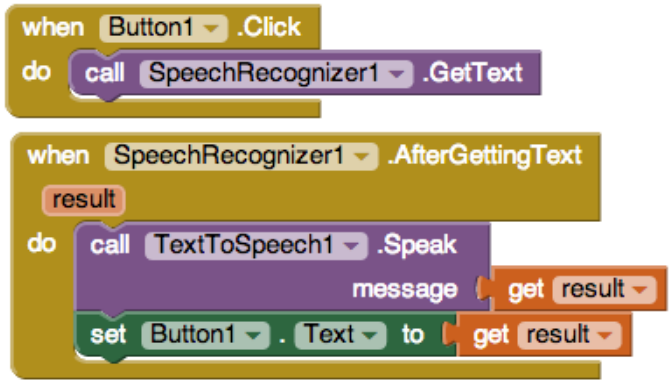
- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Simple App Inventor Example

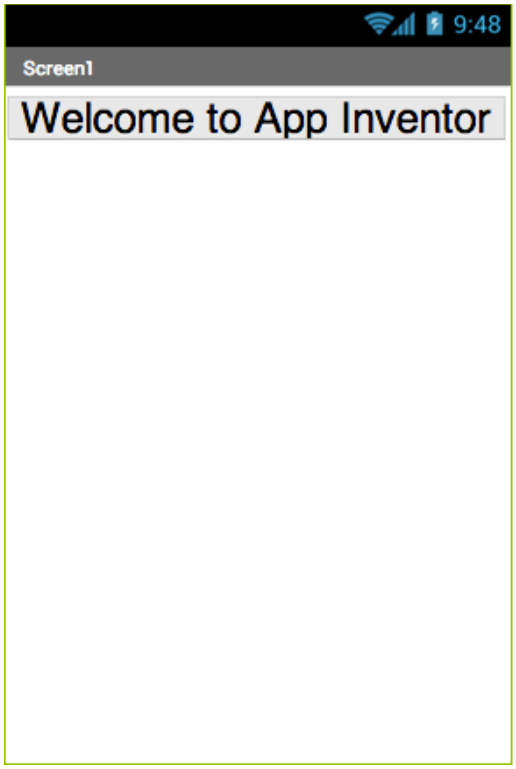
Designer Window



Blocks Editor



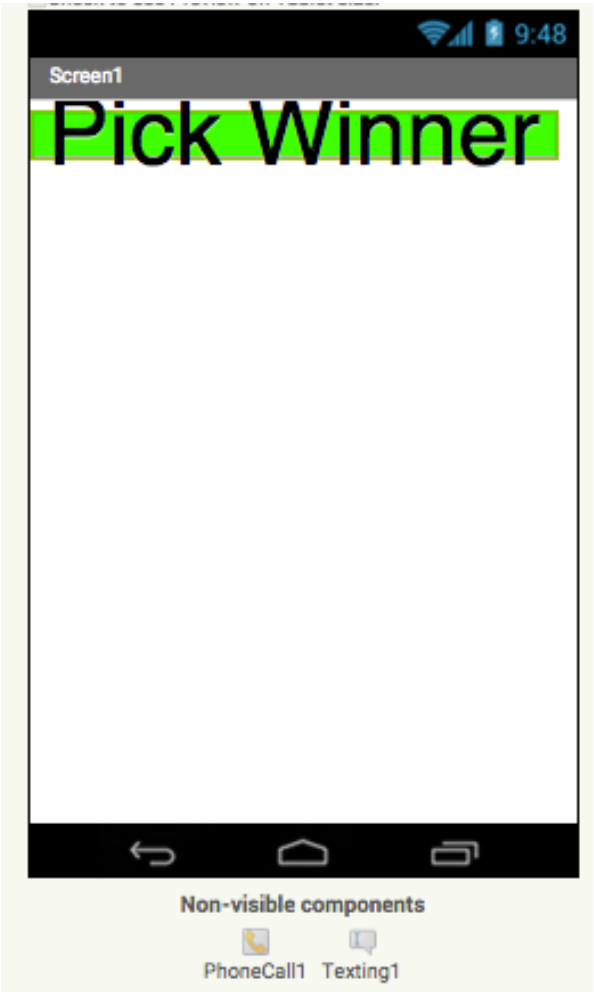
Android Device



Example: Raffle App In App Inventor

http://ai2.appinventor.mit.edu

Designer Window



Blocks Editor

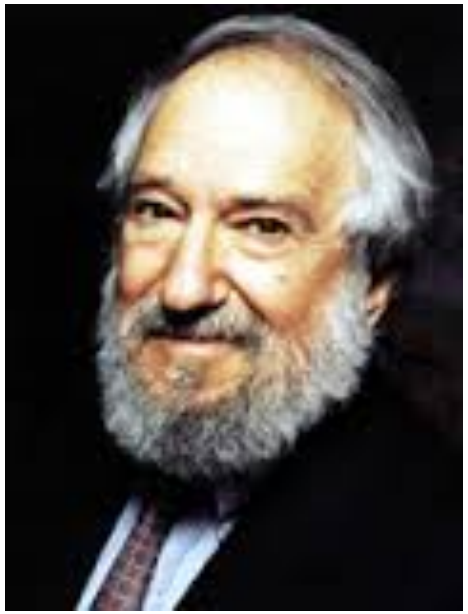
To enter the raffle, text me now with an empty message:
339-225-0287

Talk Road Map

- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

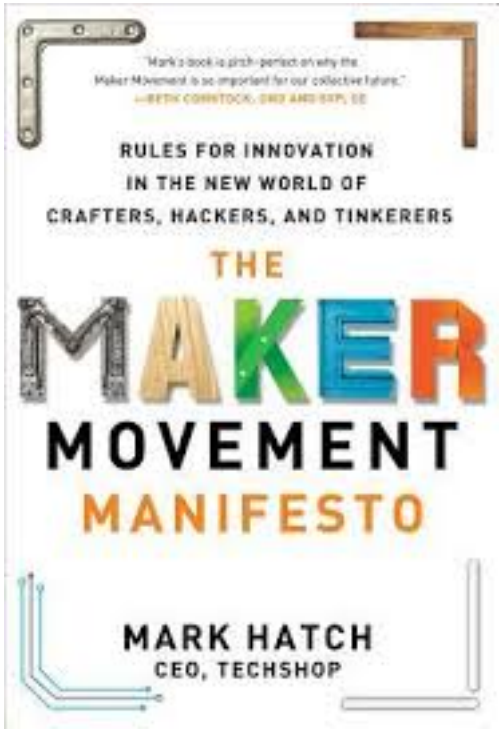
Papert on Constructionism

"The word **constructionism** is a mnemonic for two aspects of the theory of science education underlying this project … **learning is most effective when part of an activity the learner experiences as constructing is a meaningful product**." *Constructionism: A New Opportunity for Elementary Science Education (bolding mine)*

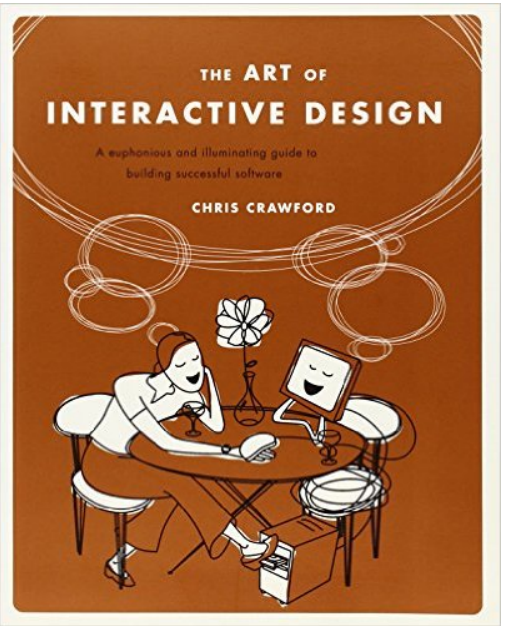


Maker Movement

"**You can innovate as a hobby. Imagine that: a nation of innovation hobbyists working to make their lives more meaningful and the world a better place.** Welcome to the maker revolution."
— Mark Hatch, The Maker Movement Manifesto: Rules for Innovation in the New World of Crafters, Hackers, and Tinkerers (bolding mine)

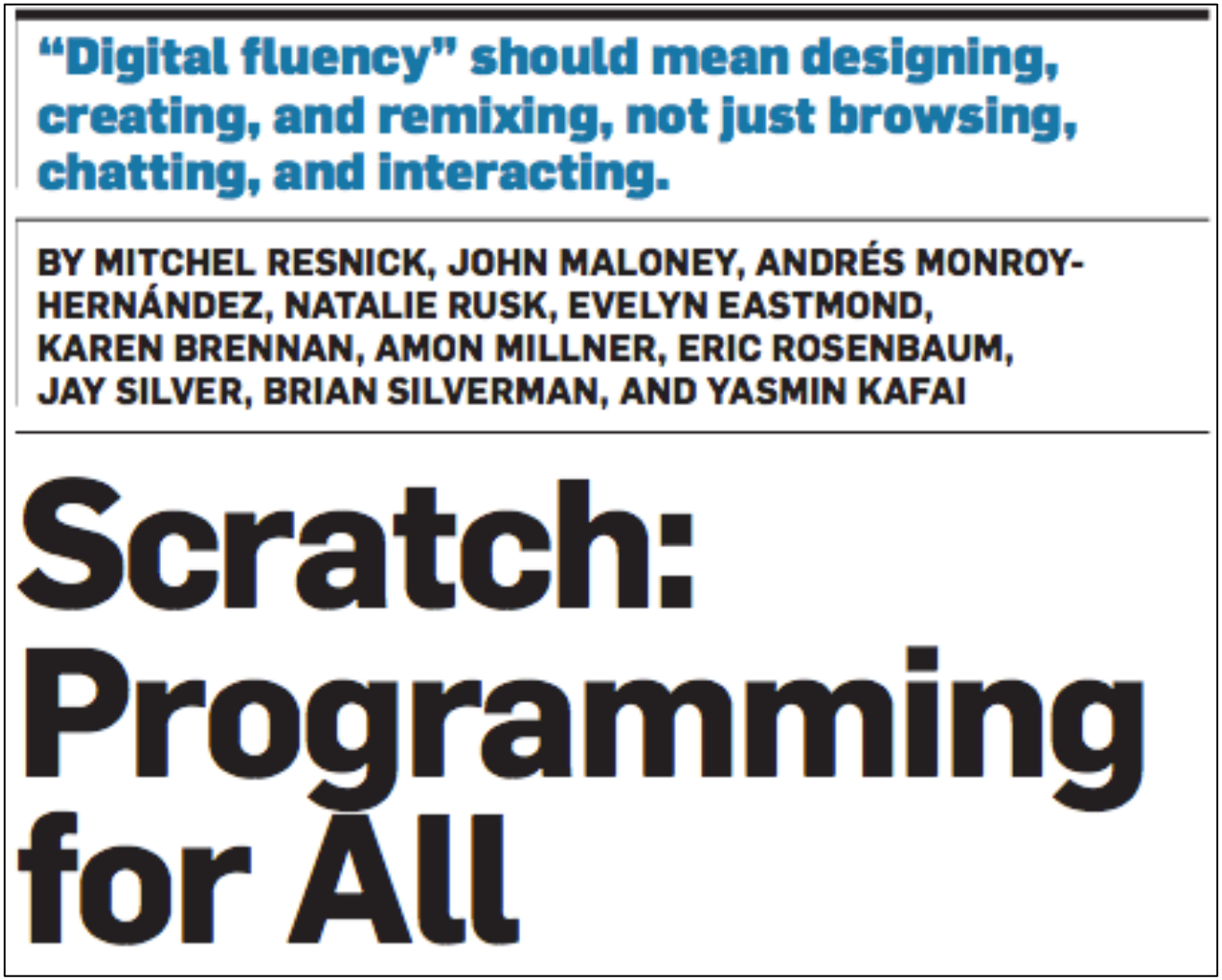


Democratizing Programming



"**What we need is a means of democratizing programming**, of taking it out of the soulless hands of the programmers and putting it into the hands of a wider range of talents."
Chris Crawford,
The Art of Interactive Design

Democratizing Programming



"Digital fluency" should mean designing, creating, and remixing, not just browsing, chatting, and interacting.

BY 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

CACM, Nov. 2009

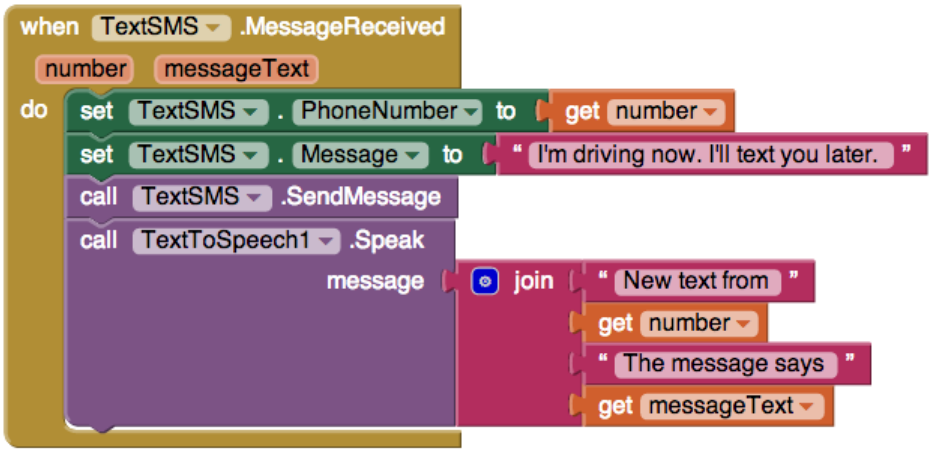
Democratizing Programming

MIT App Inventor mission statement:
The MIT App Inventor project seeks to **democratize** software development by empowering all people, especially young people, to transition from being consumers of technology to becoming creators of mobile technology.

No Texting While Driving App

Daniel Finnegan, English Major, developed the app in Dave Wolber's USF course *CS017: Computing, Mobile Apps, and the Web*

Daniel's code, translated into App Inventor 2:



Clive Thompson on Coding for the Masses

By Clive Thompson | November 29, 2010 | 12:00 pm | Wired December 2010

How do you stop people from texting while driving? Last spring, Daniel Finnegan had an idea. He realized that one of the reasons people type messages while they're in the car is that they don't want to be rude—they want to respond quickly so friends don't think they're being ignored.

So what if the phone knew you were driving—and responded on its own?

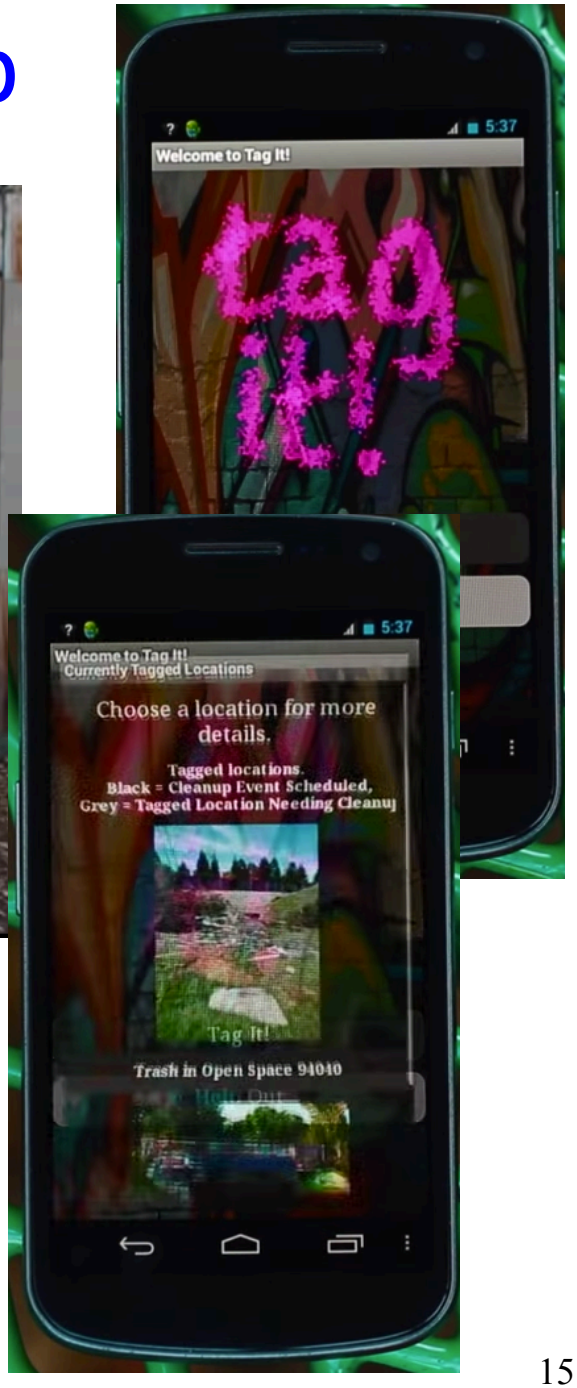
Normally, Finnegan wouldn't have been able to do anything with his insight. He was a creative-writing major at the University of San Francisco, not a programmer. But he'd enrolled in a class where students were learning to use Google's App Inventor, a tool that makes it pretty easy to hack together simple applications for Android phones by fitting bits of code together like Lego bricks.

App To Track Feral Hogs

Alabama's Lawrence County High School students used App Inventor to build an app that tracks feral hogs, which were causing economic damage to their community. Their app won a prize of $100K in technology for Samsung's 2012 Solve for Tomorrow contest.

http://www.forbes.com/sites/samsung/2013/11/25/high-school-students-battle-wild-hogs-with-stem-solutions/

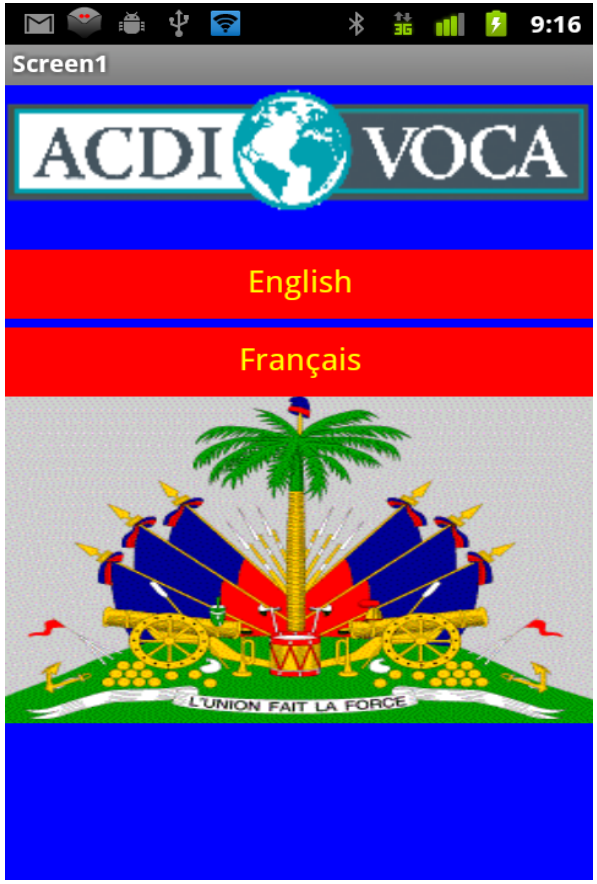
Trash & Graffiti Cleanup App



East Palo Alto girls created an app to tag the location of trash and create an event for cleaning it up. This app ranked highly in the Technovation Challenge competition.

http://appinventor.mit.edu/explore/stories/east-palo-alto-girls-create-app-clean-graffiti-trash.html

Commodity Tracker App for Haiti



Developed using App Inventor as part of Trinity College's Humanitarian Free and Open Source Software (HFOSS) project.

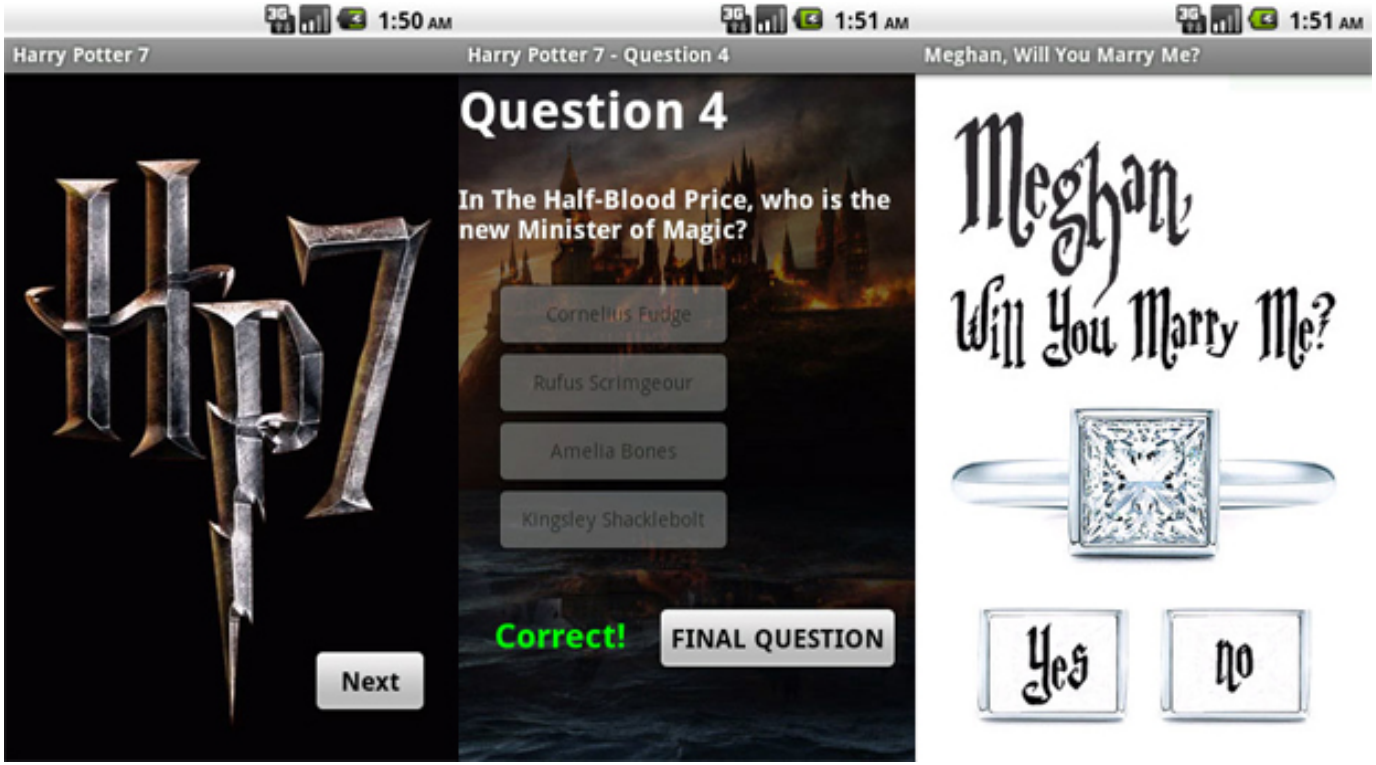
http://notes.hfoss.org/index.php/Haiti_Commodity_Collector

App to Destroy Mines Safely

Chris Metzger, United States Marine Corps Staff Sergeant, used App Inventor to create an app that helps other Marines destroy weaponry captured in the field. It calculates the amount of explosives necessary to safely destroy captured ammunition and mines.

http://appinventor.mit.edu/explore/stories/united-states-marines-use-app-inventor-field.html

Marriage Proposal App



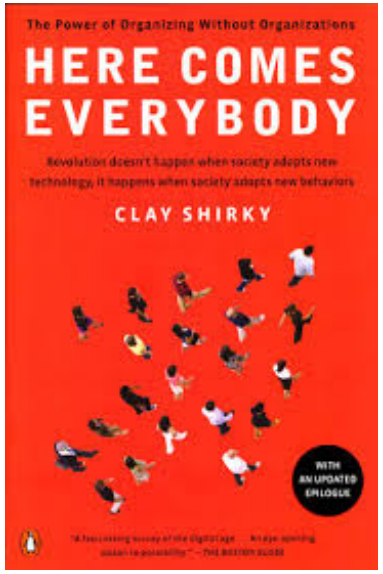
Hodgson didn't know how to develop an Android app. … "How the heck was I going to build this thing?" he recalls thinking. "I tried a couple of other rapid development tools, but they really had too much of a learning curve to let me do it in the time-frame I had in mind." That is, until a friend recommended App Inventor, a tool for amateur Android devs created by Google Labs. "It allowed me, with no java knowledge, to quickly get this thing whipped up," Hodgson says.

http://www.fastcompany.com/1754193/google-love-story-man-builds-android-app-propose-girlfriend

Clay Shirky on Situated Software vs. Web School (2004)

Target small population

- NYU ITP *Teachers on the Run*
  vs. RateMyProfessors.com
- scaling issues unimportant
- simple hardwired data vs. scalable databases
- software for your mom



Leverage small groups

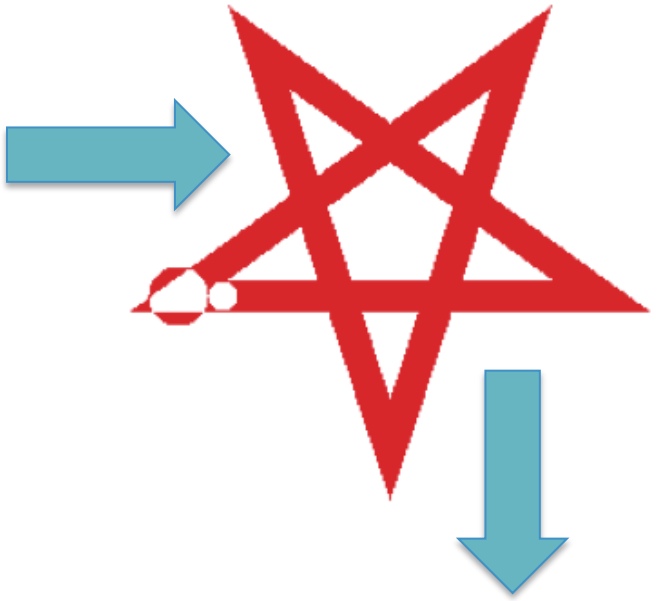
- local knowledge
- trust of other users
- publicly shame deadbeats in group purchase apps

http://shirky.com/writings/herecomeseverybody/situated_software.html

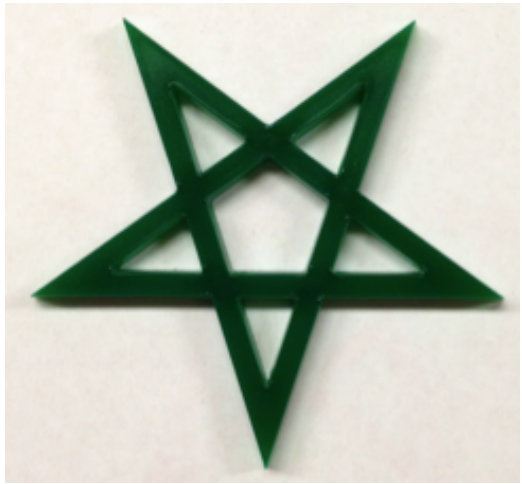
TurtleBlocks

TurtleBlocks program

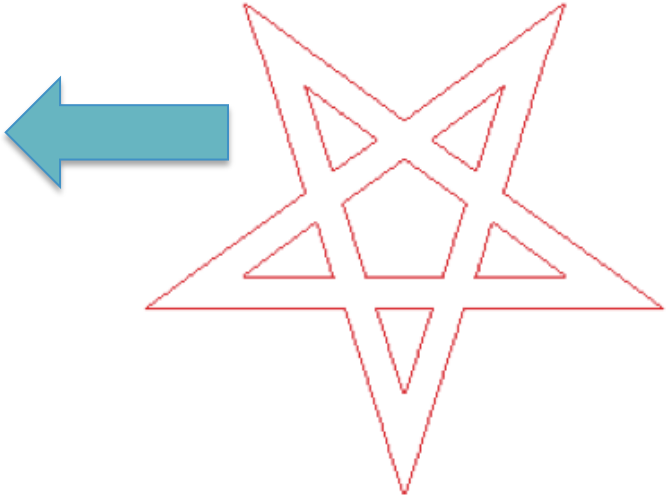
turtle drawing



cardstock

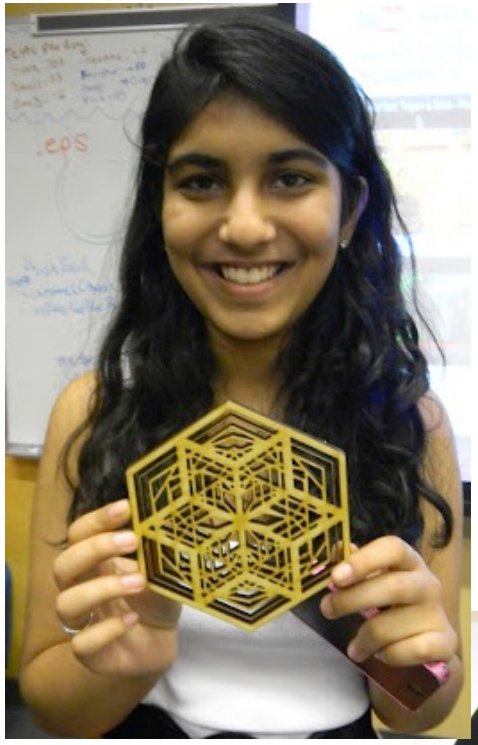


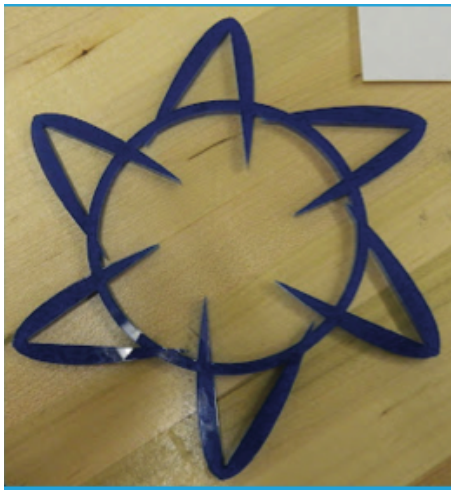
acrylic

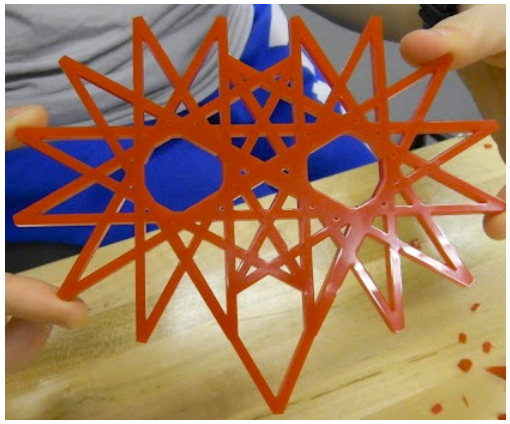


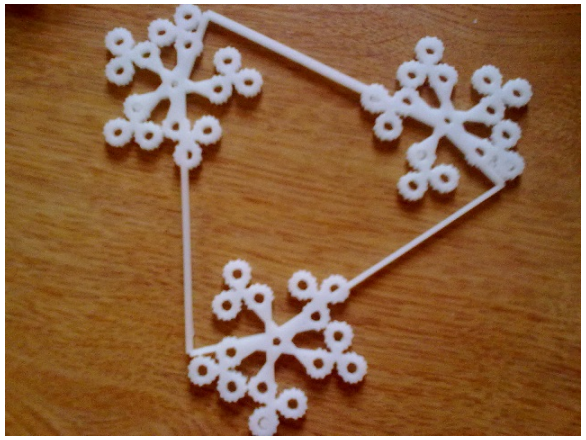
drawing boundary

TurtleBlocks Artifacts



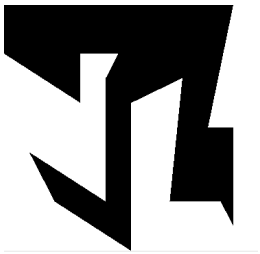




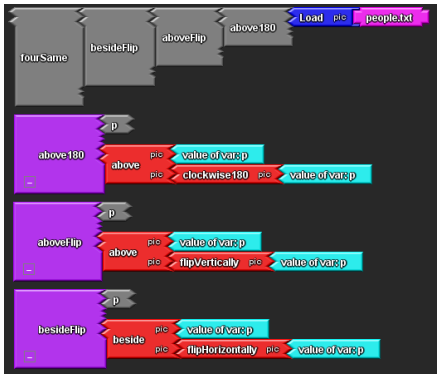


PictureBlocks: Sketching & Engraving

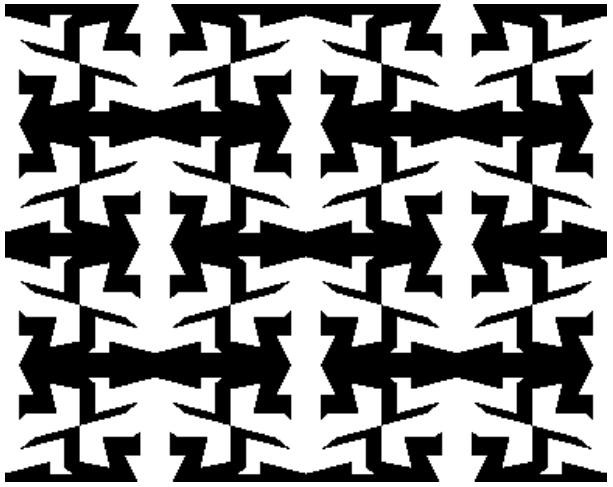
user sketch

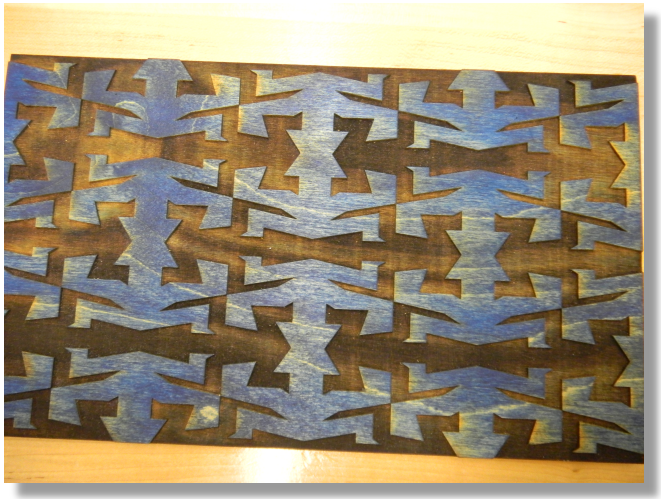


PictureBlocks program

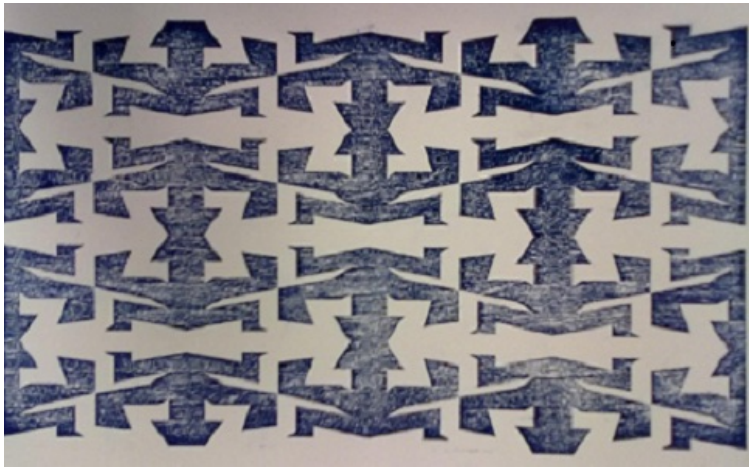


resulting picture



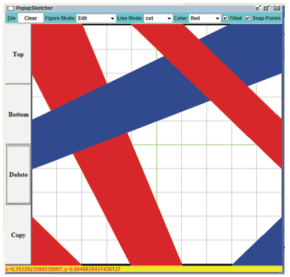


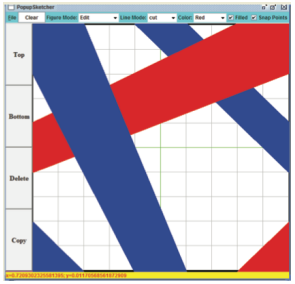
wood engraving

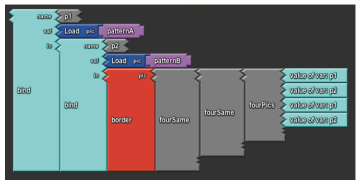


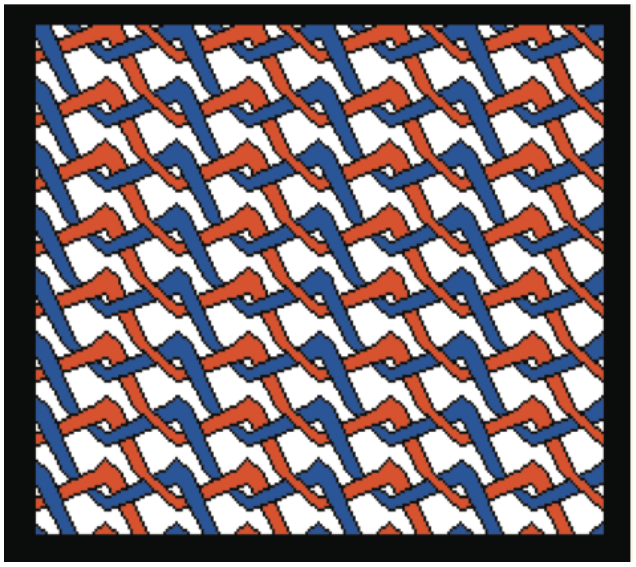
print from engraving

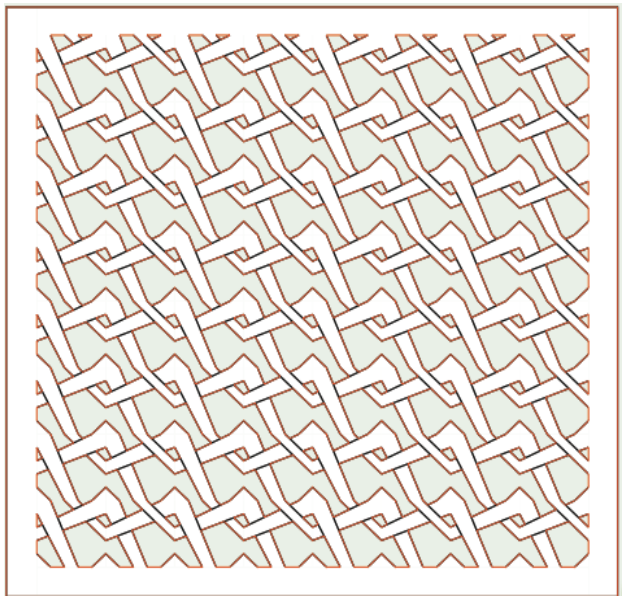
PictureBlocks: Engraving + Cutting

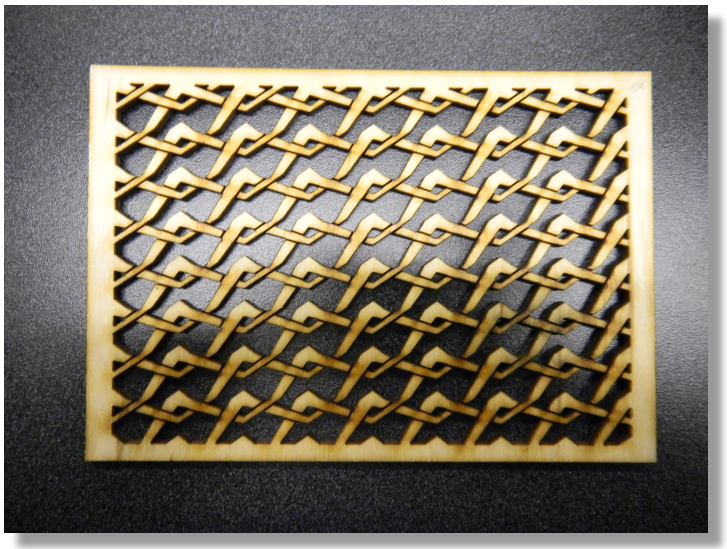




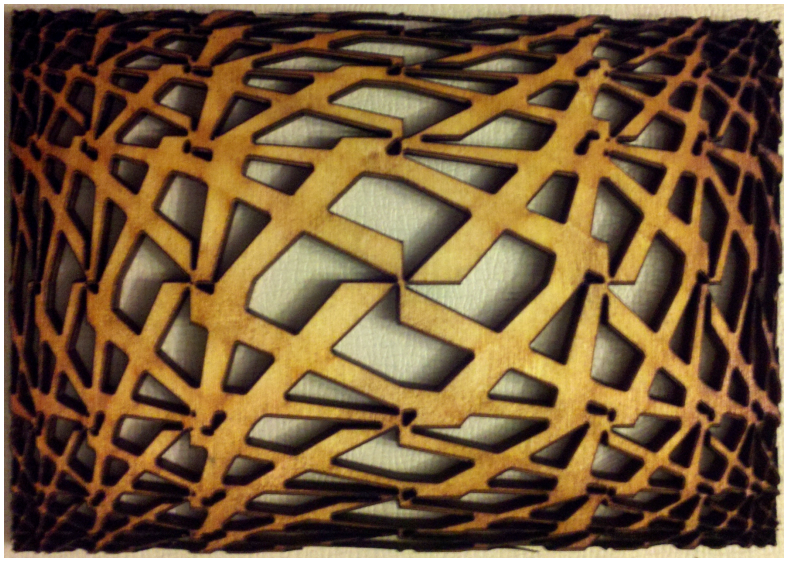






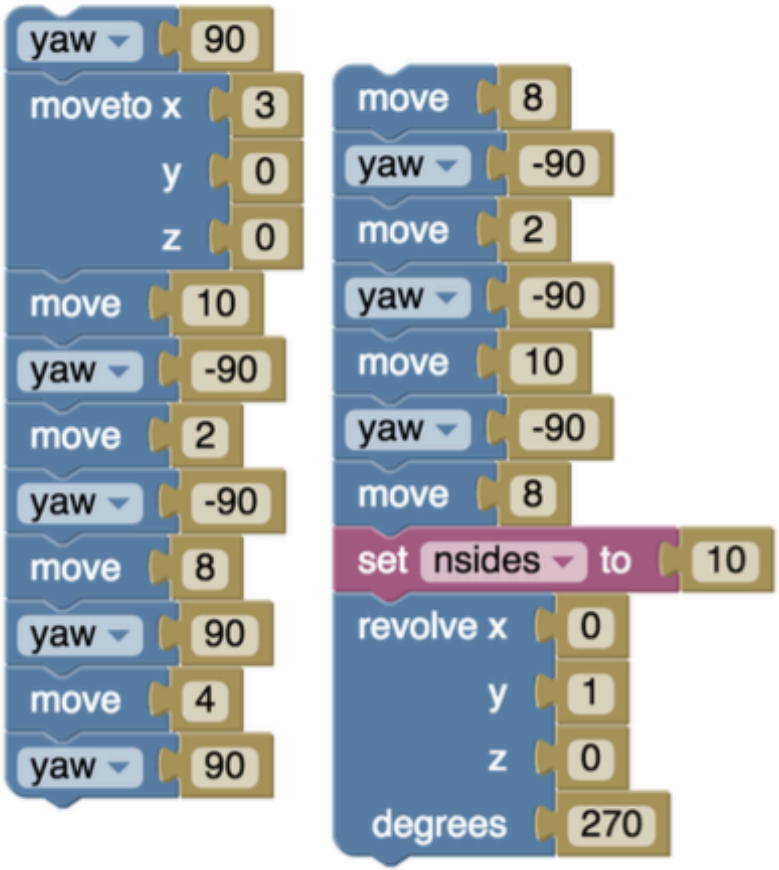


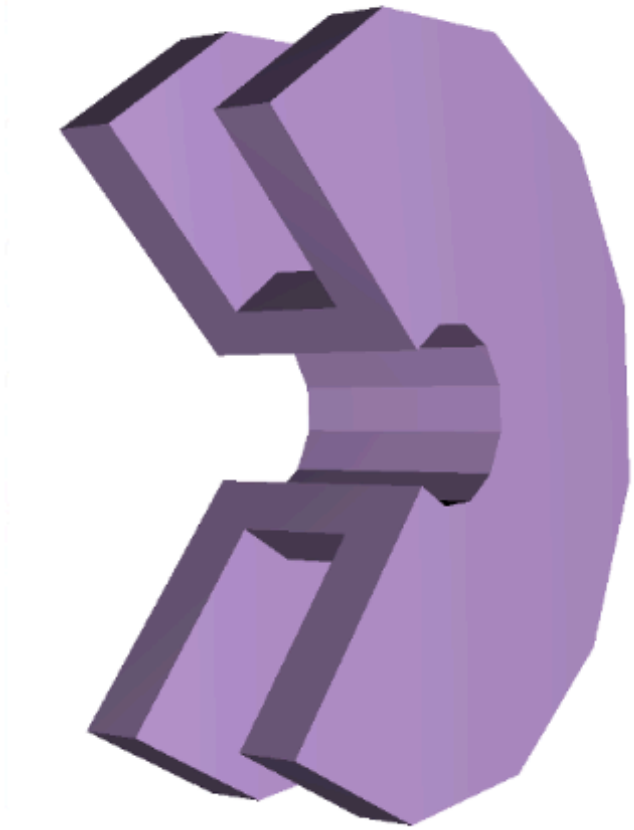
PictureBlocks Artifacts



Madeup: 3D Modeling with Blocks

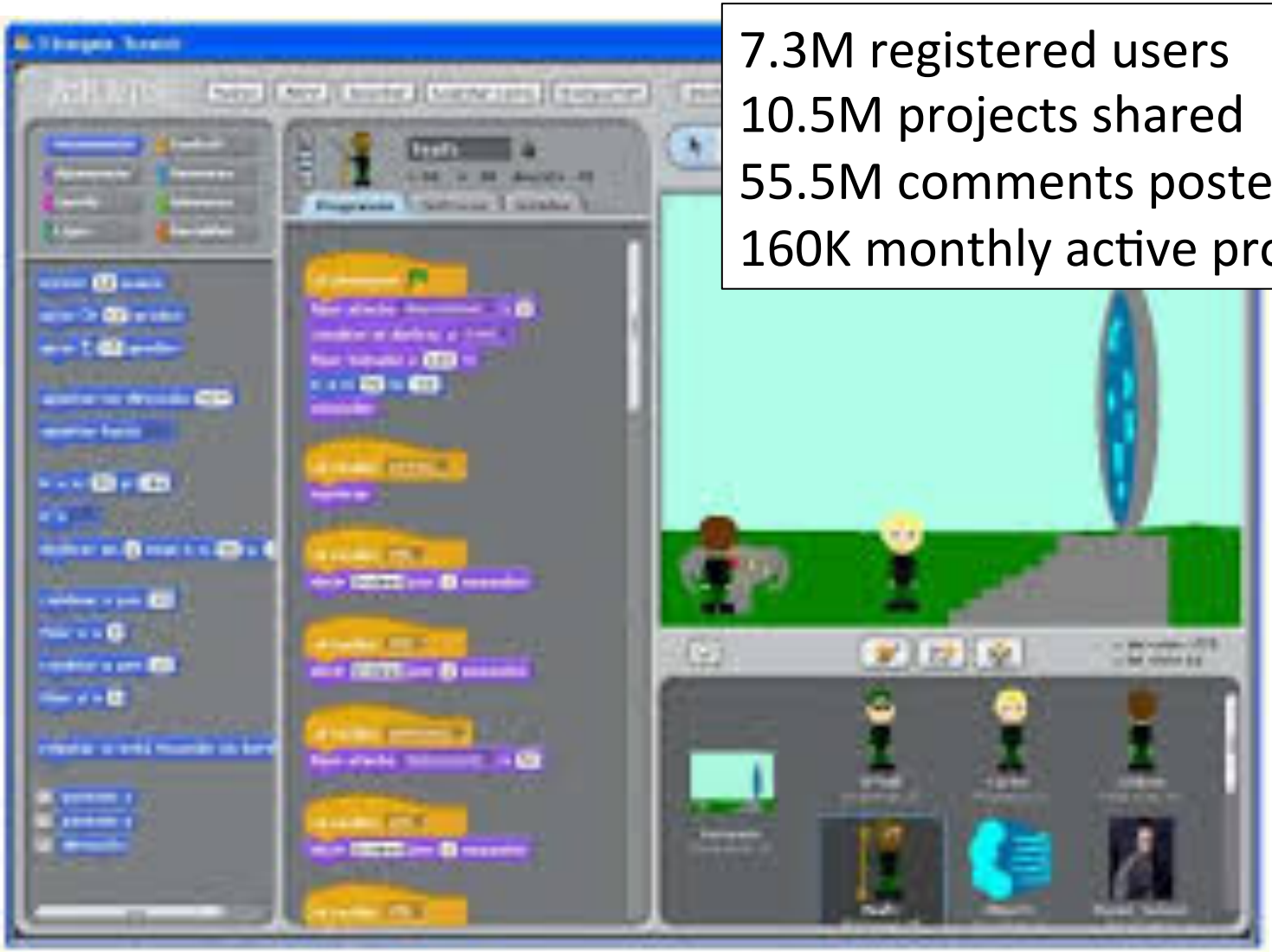
Chris Johnson, University of Wisconsin
Peter Bui, Notre Dame





Scratch

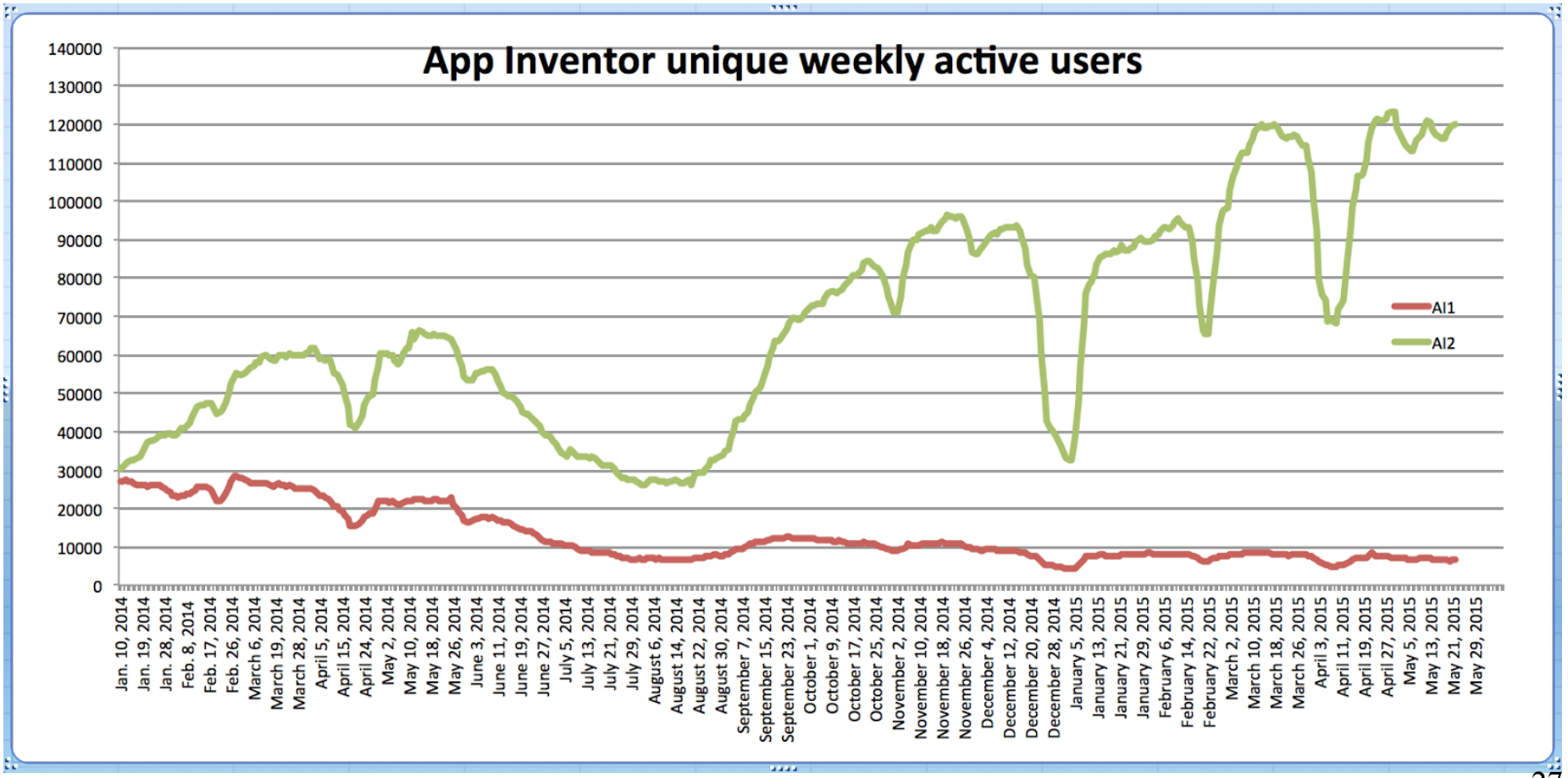
multi-media programs, animations, and games



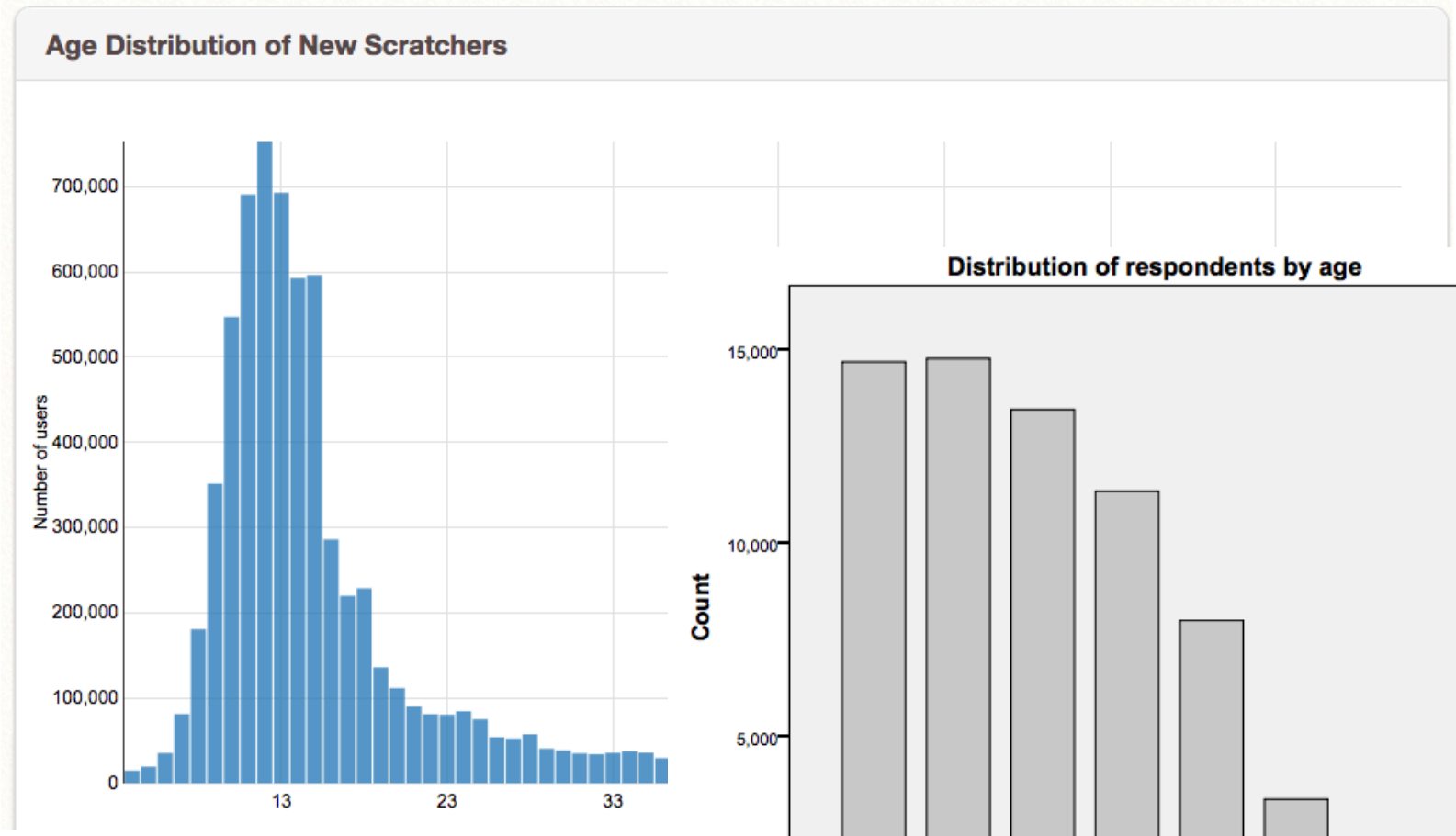
7.3M registered users
10.5M projects shared
55.5M comments posted
160K monthly active project creators

App Inventor Usage is Growing

- 3.3 million registered users
- 185 countries
- 8.9 million mobile apps created
- ~ 120K unique weekly users

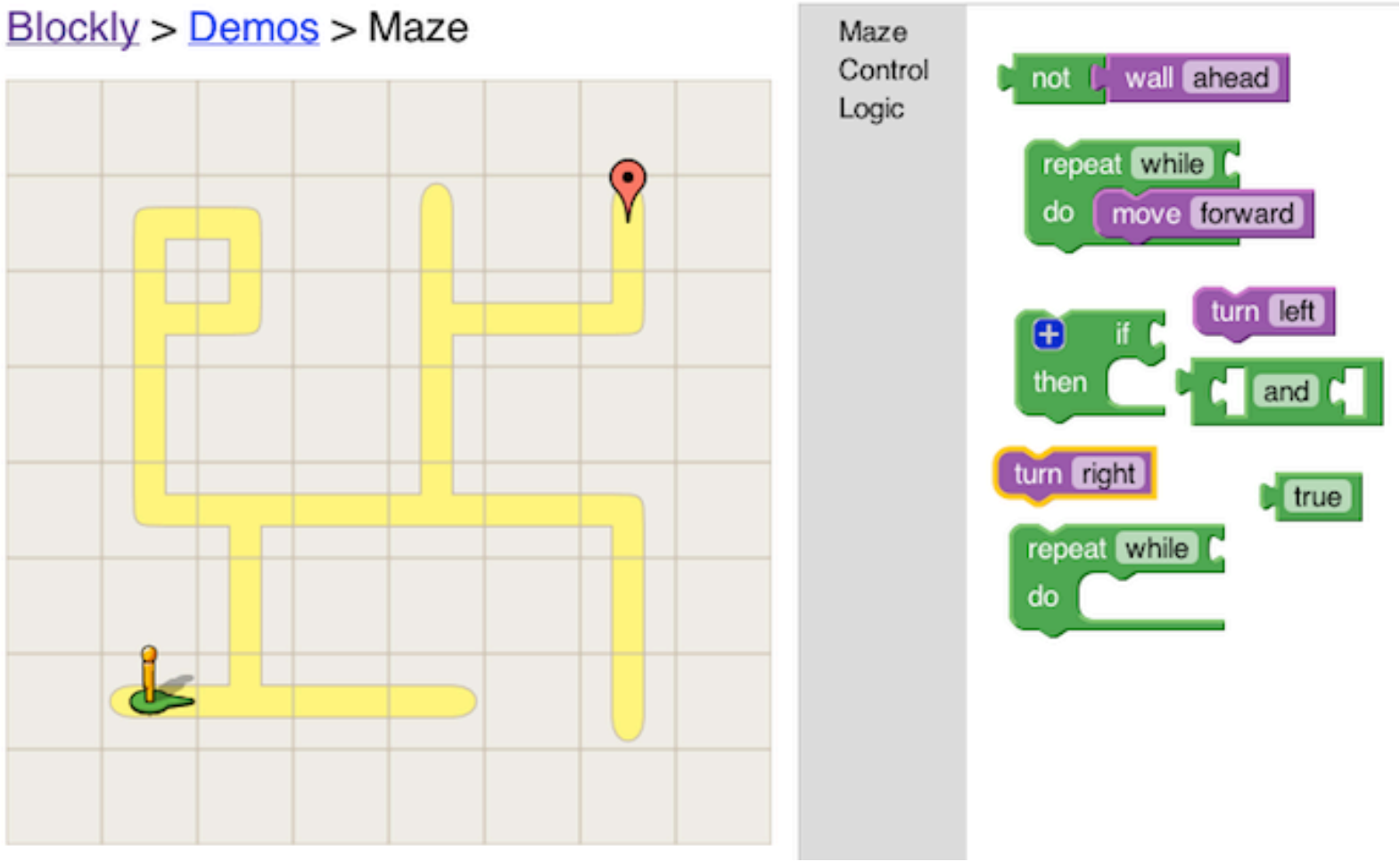


Age Distribution: Scratch vs. App Inventor



Blockly

Many blocks-based activities. Basis for early Code.org challenges.
Many other blocks environments, including App Inventor,
are based on Blockly.



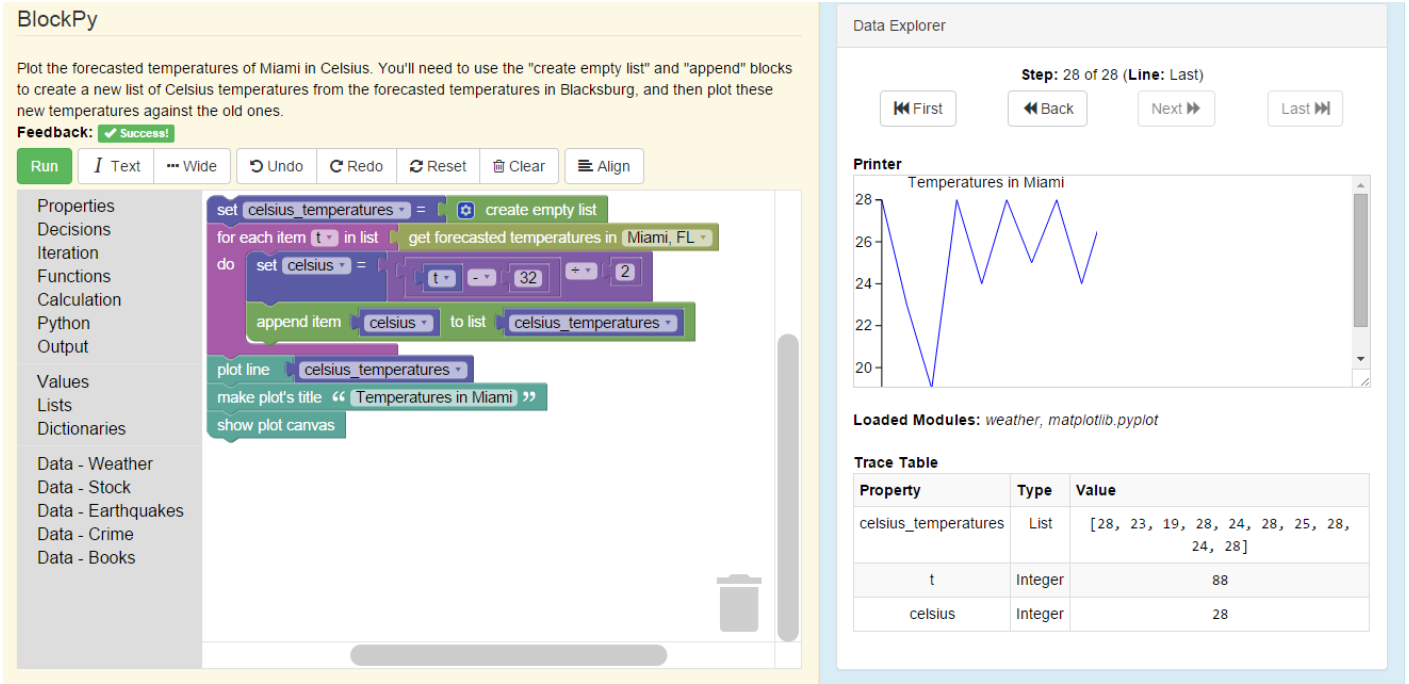
And many more …

Snap!: Scratch for Scheme, *Beauty and Joy of Computing curriculum* (Harvey, Monig, Garcia @ Berkeley)

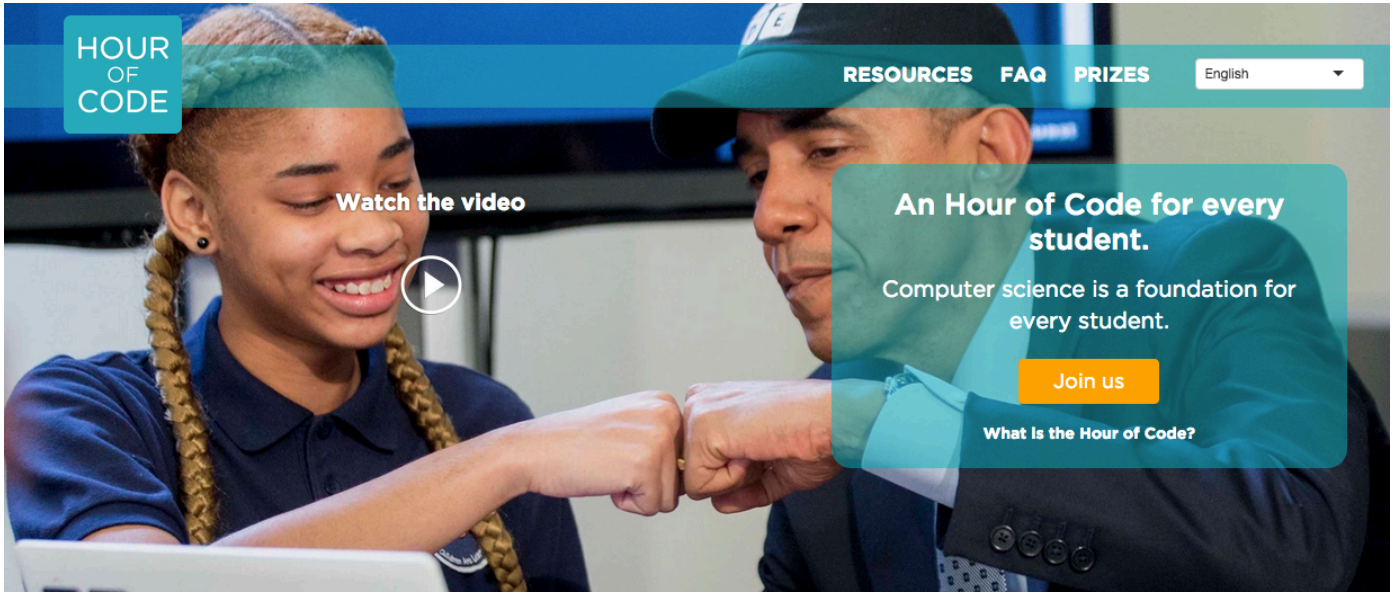
StarLogo Nova: multi-agent simulations (Wendel et al @ MIT)

Alice: 3D storytelling and gaming environment (CMU)

BlockPy: Blocks-based version of Python for teaching data science (Bart, Tilevitch, Shaffer, Kafura @ Virginia Tech)



Code.org Hour of Code

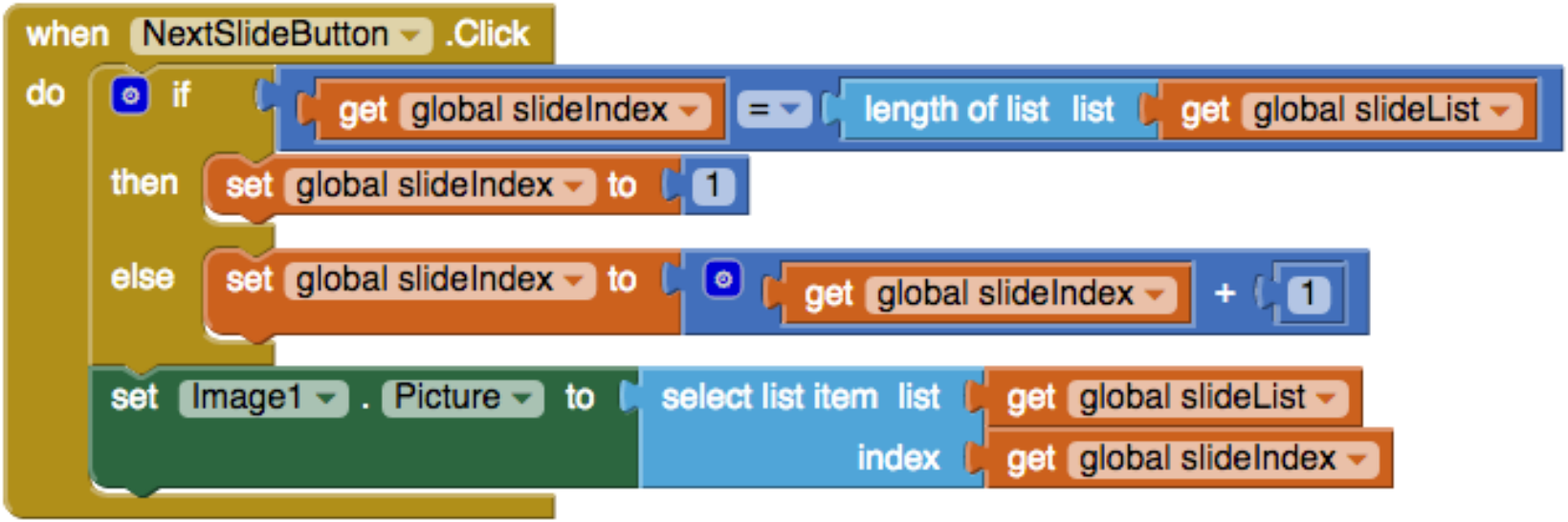


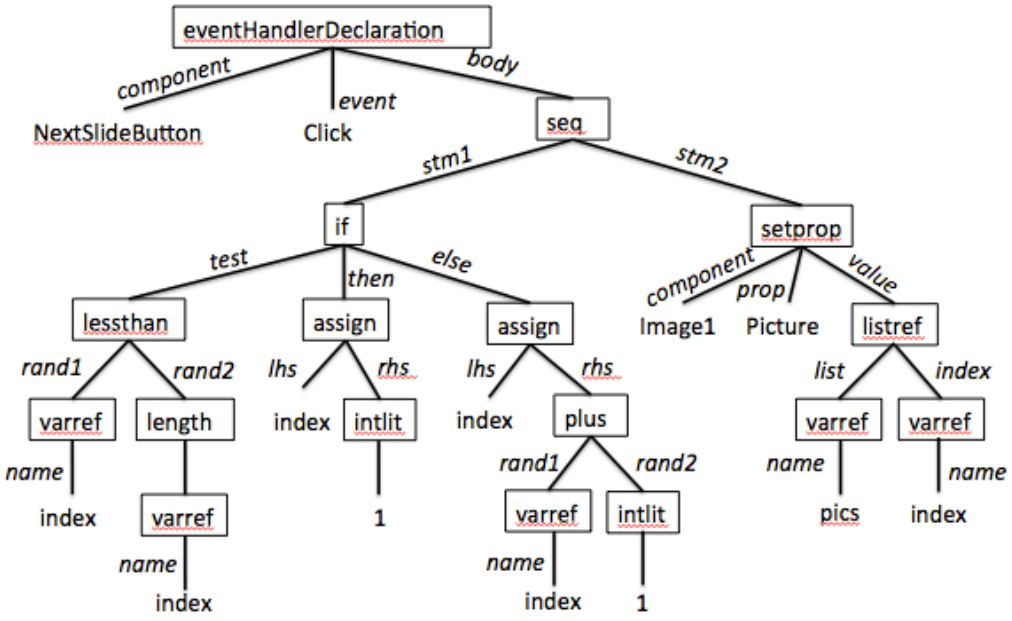
- Dec. 2013:
  - ✧ 26M participants spend an hour programming in one of ~24 programming environments
  - ✧ 74% of these use one of the 5 blocks languages
    - Code.org exercises based on Blockly
    - Scratch
    - App Inventor
    - Tynker
    - Hopscotch
- Dec. 2014 and beyond: claim > 100M participants total

Talk Road Map

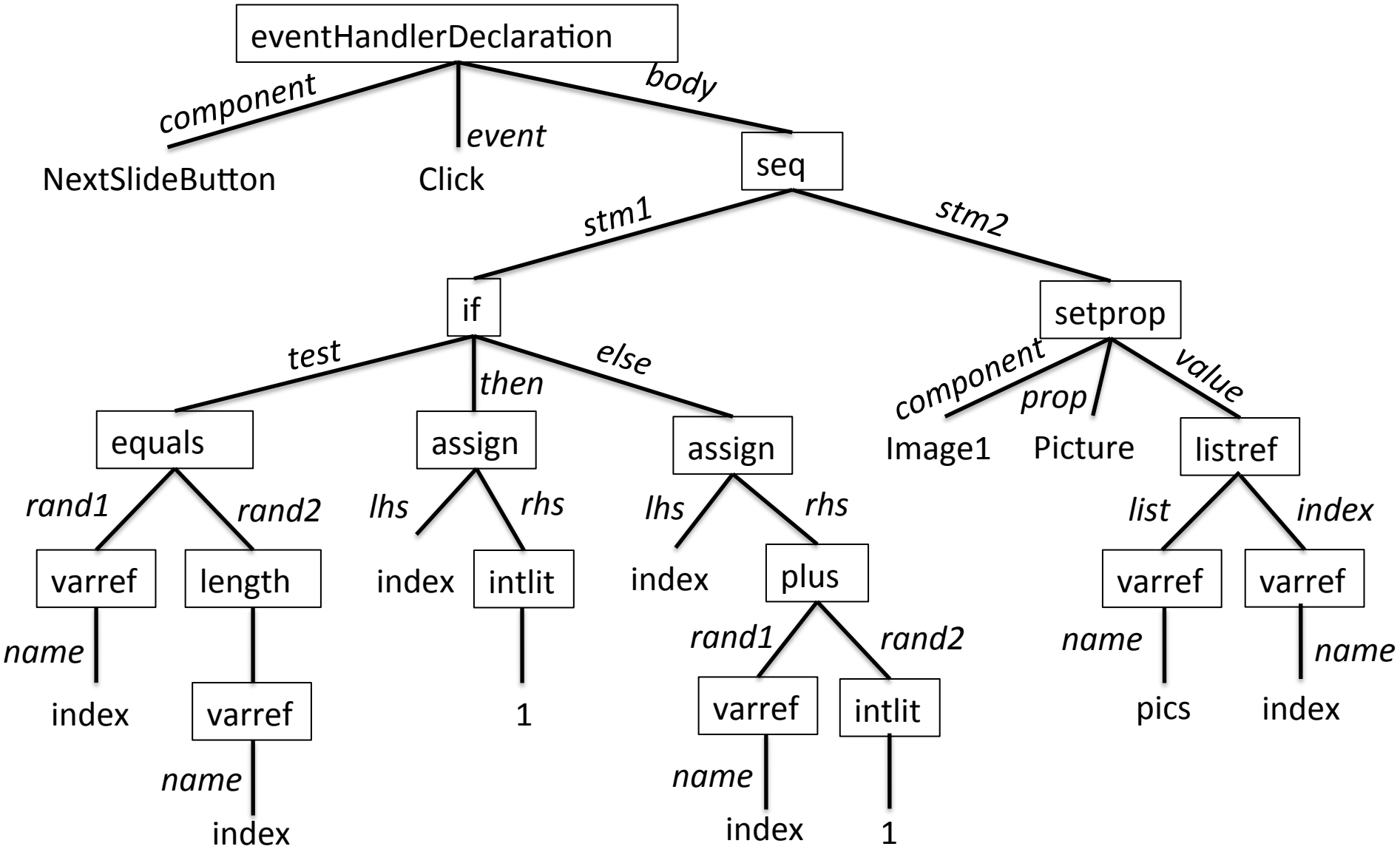
- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Blocks Represent Abstract Syntax Trees (ASTs)

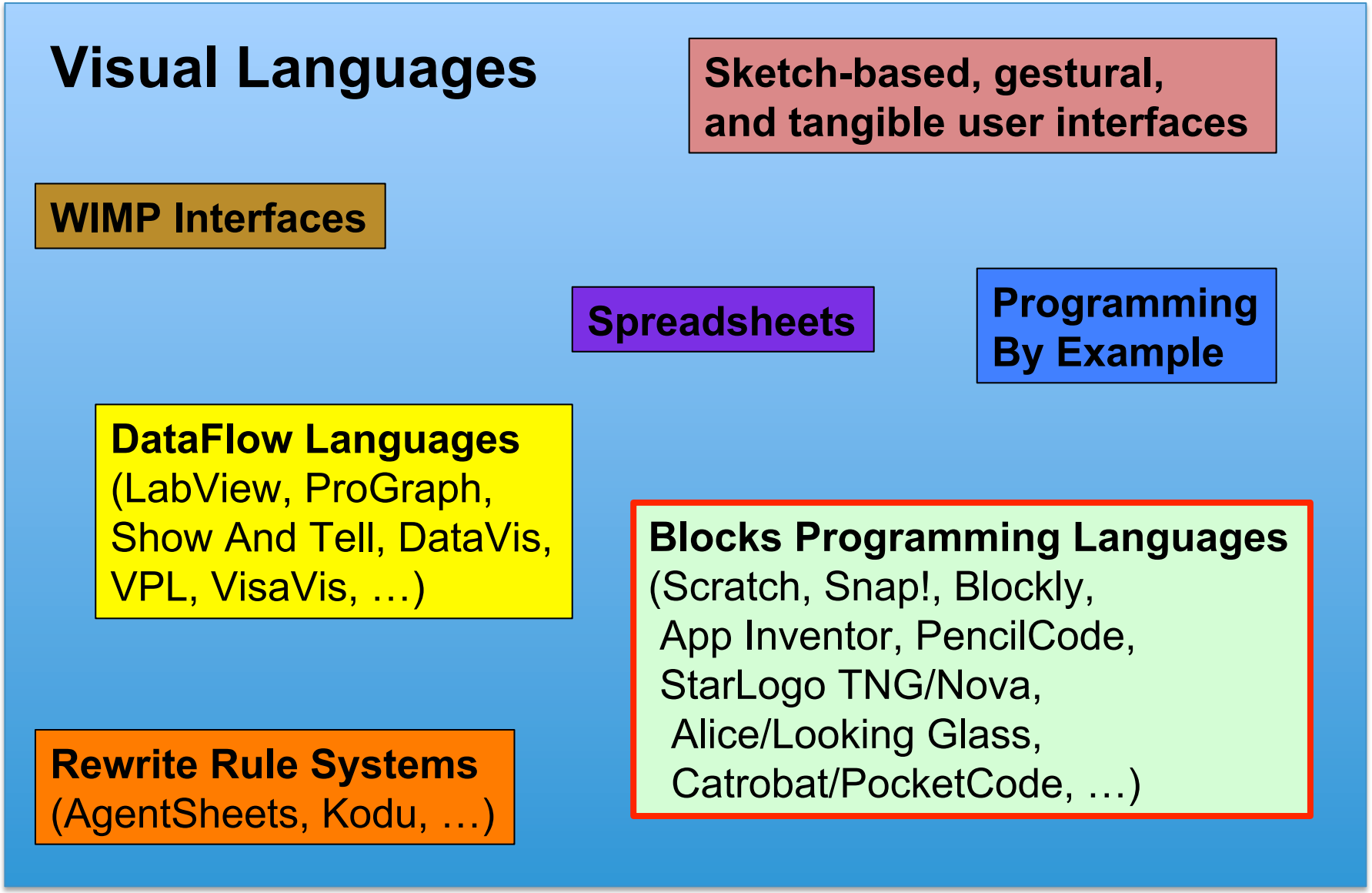




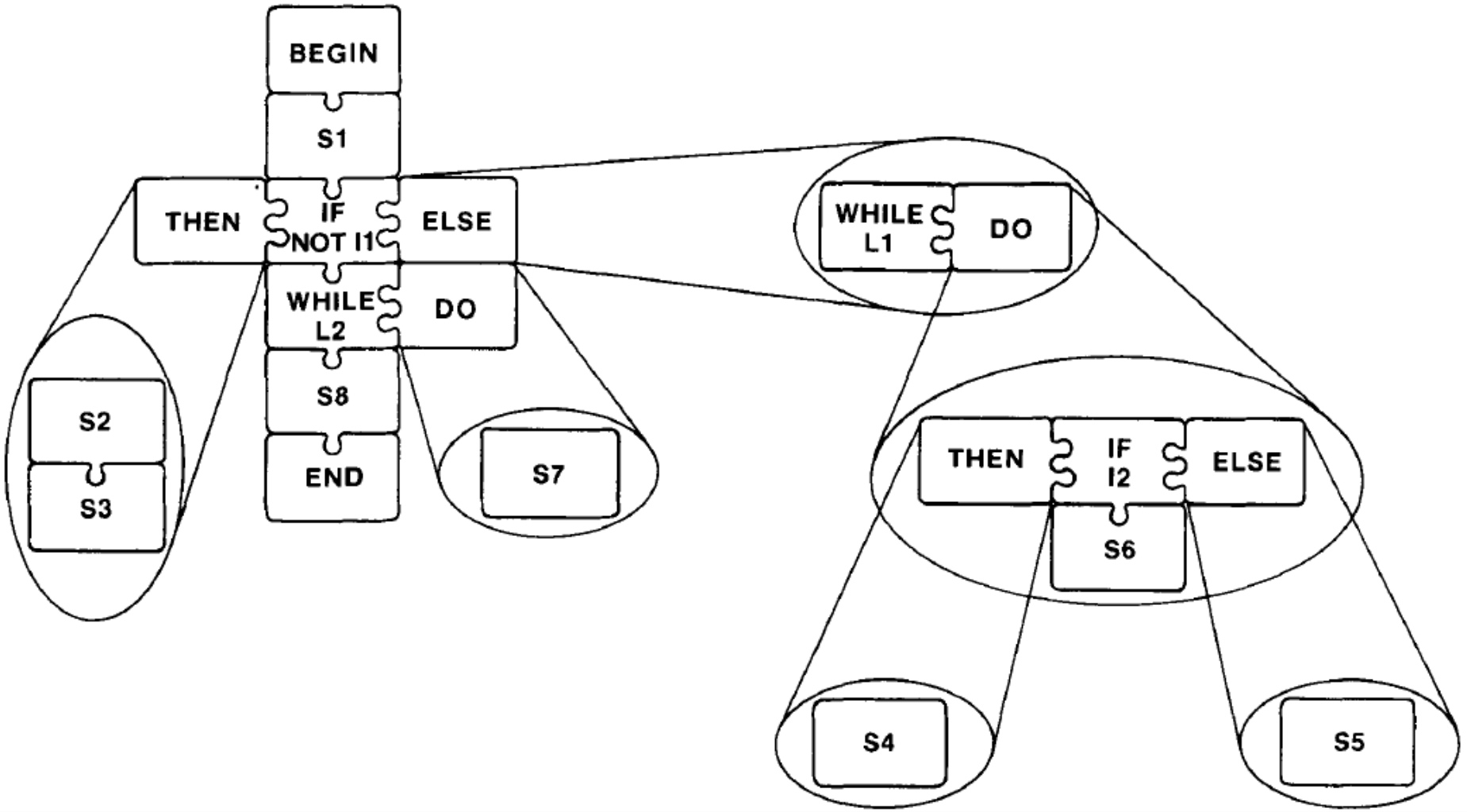
Blocks Represent Abstract Syntax Trees (ASTs)



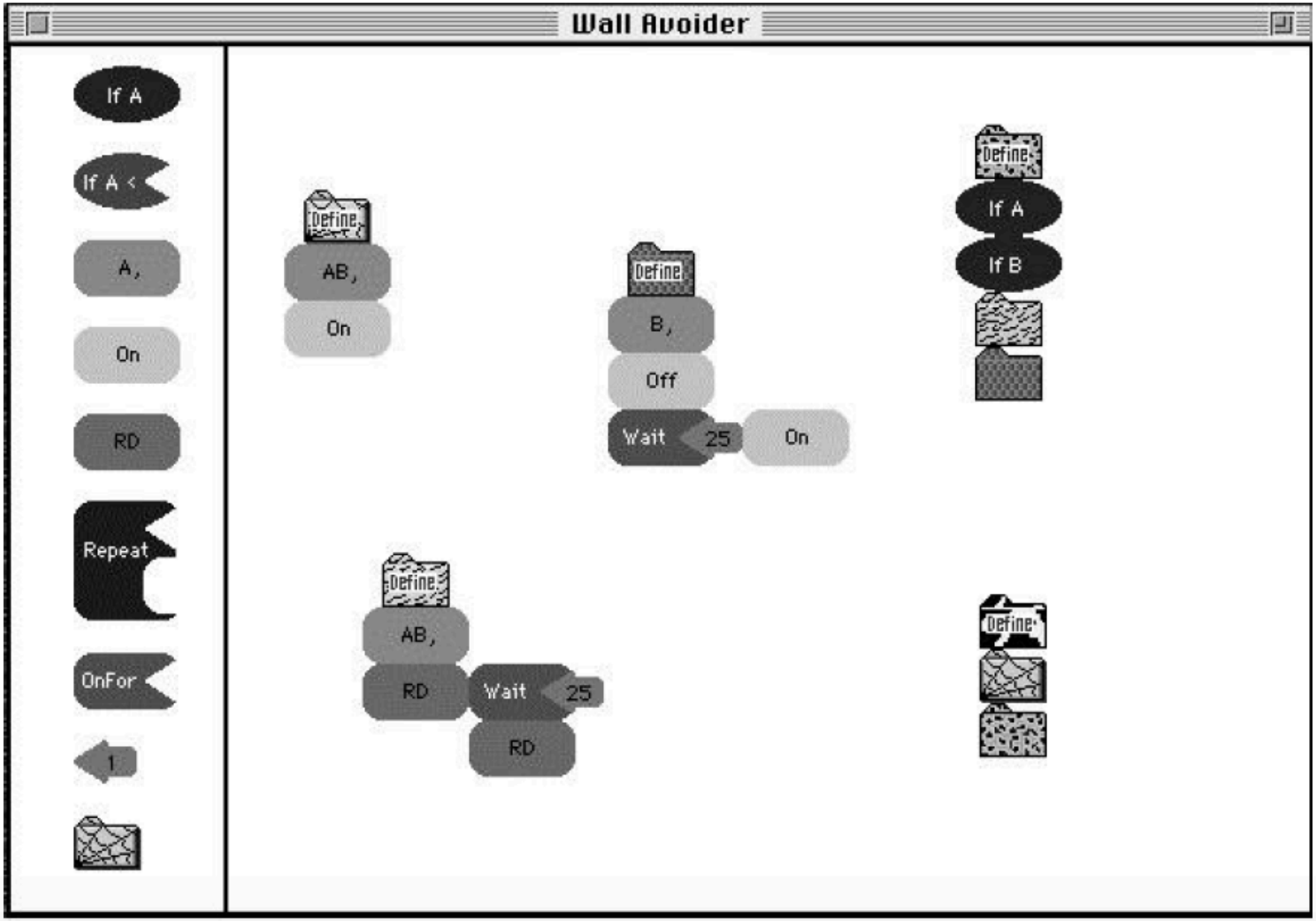
Blocks Languages in the Visual Languages Space



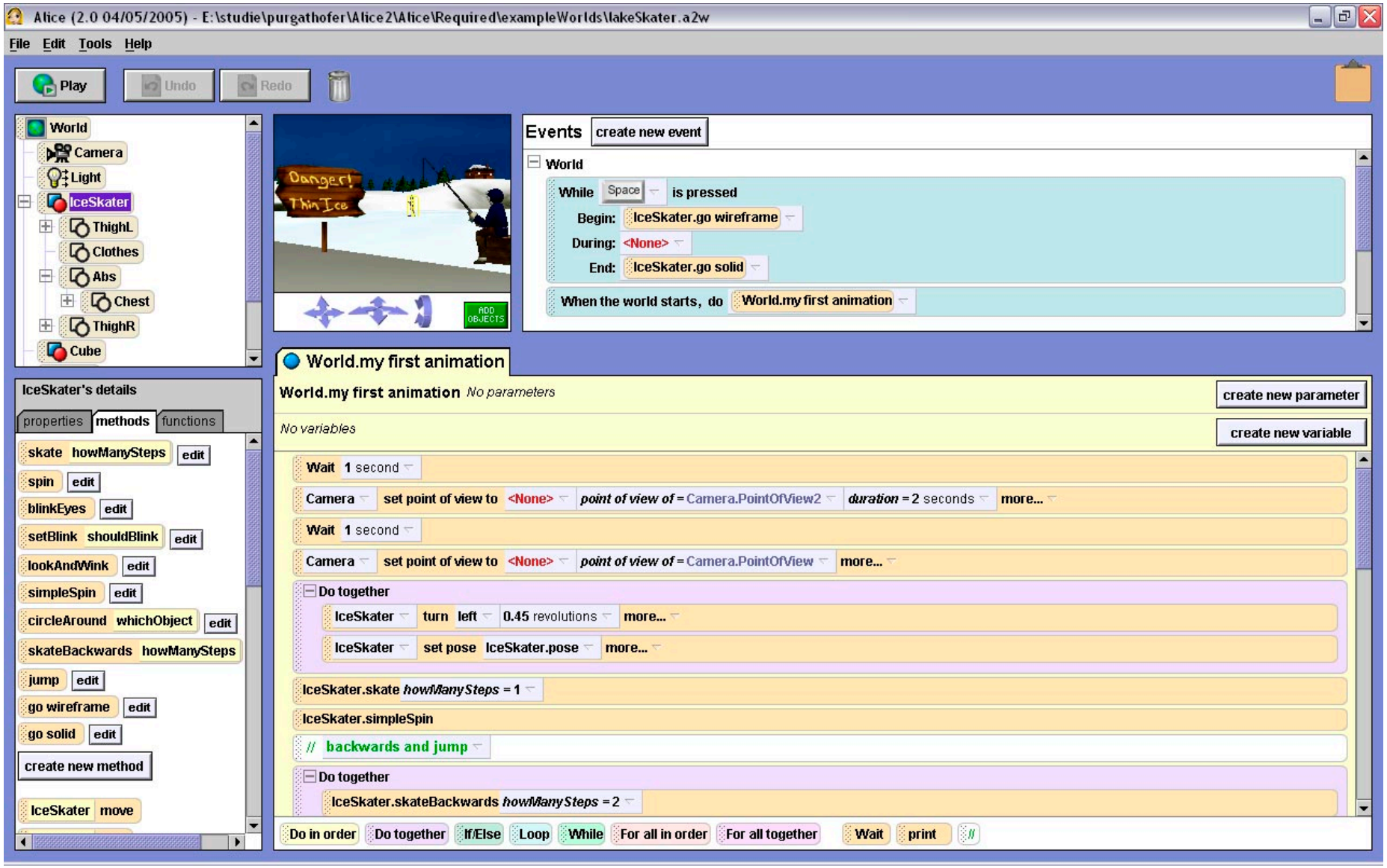
BLOX (Glinert, 1986)



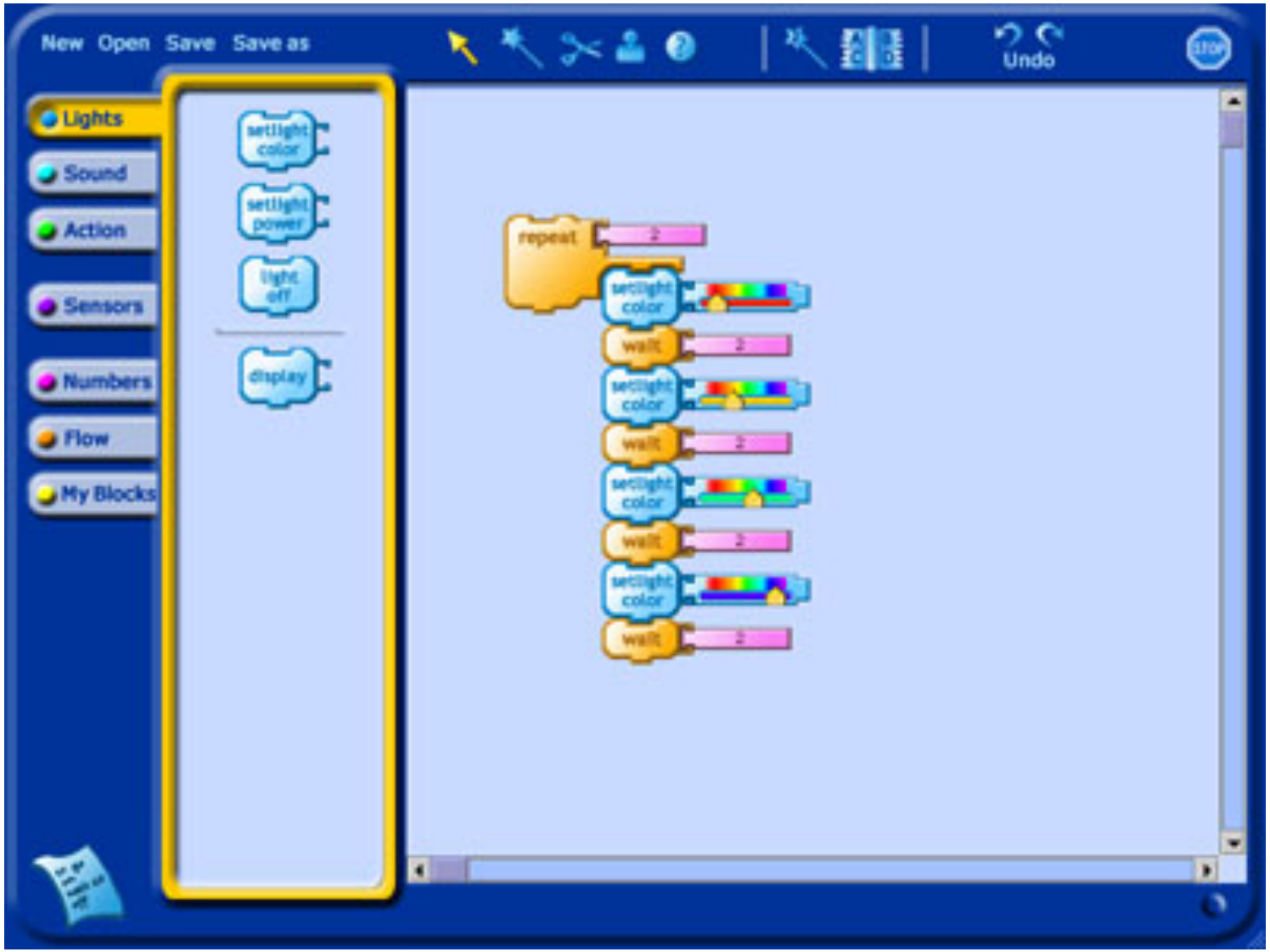
LogoBlocks (Begel, 1996)



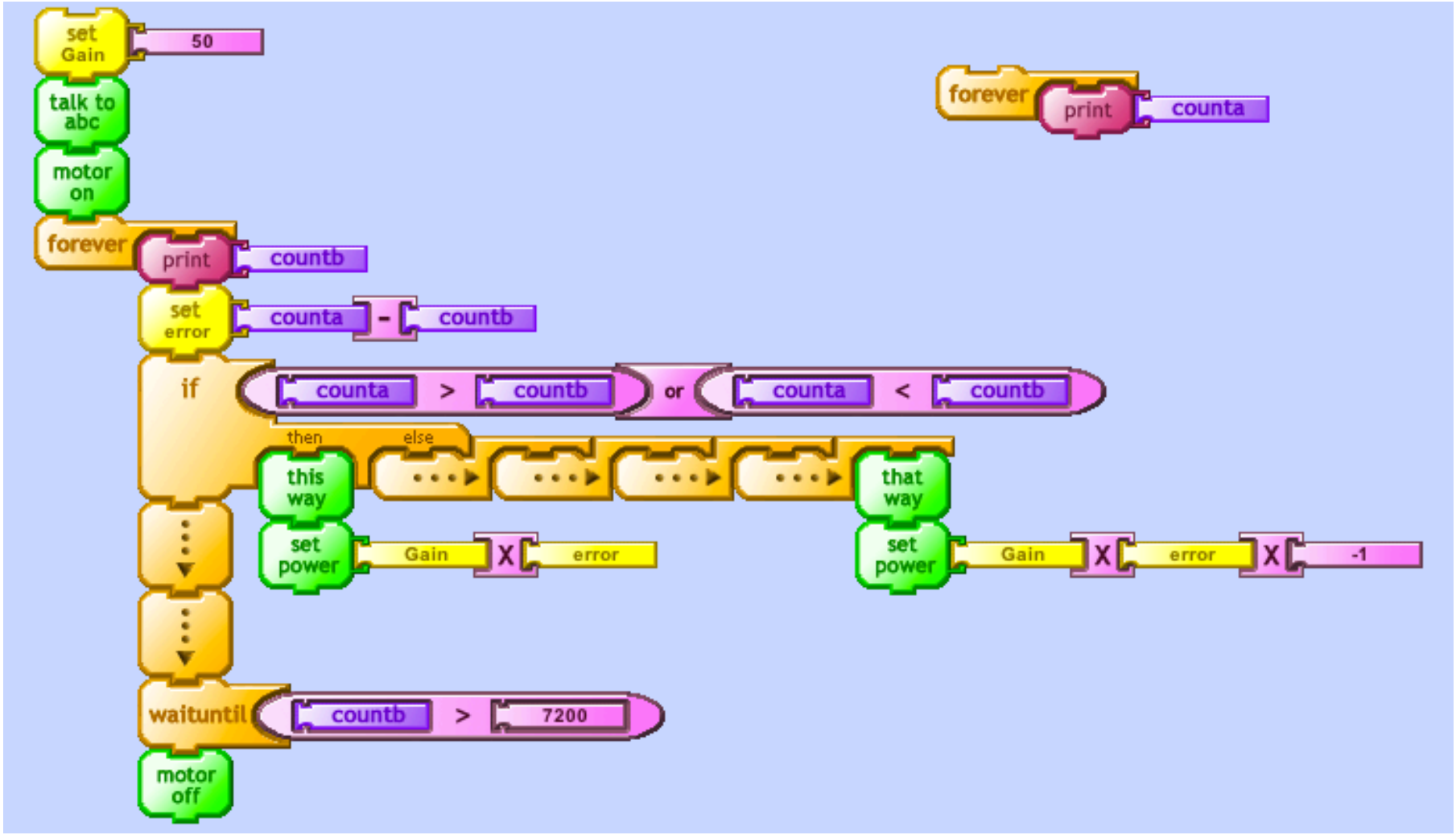
Alice (Pausch et al., 2001)



PicoBlocks (Bonta, Silverman, et al., 2006)

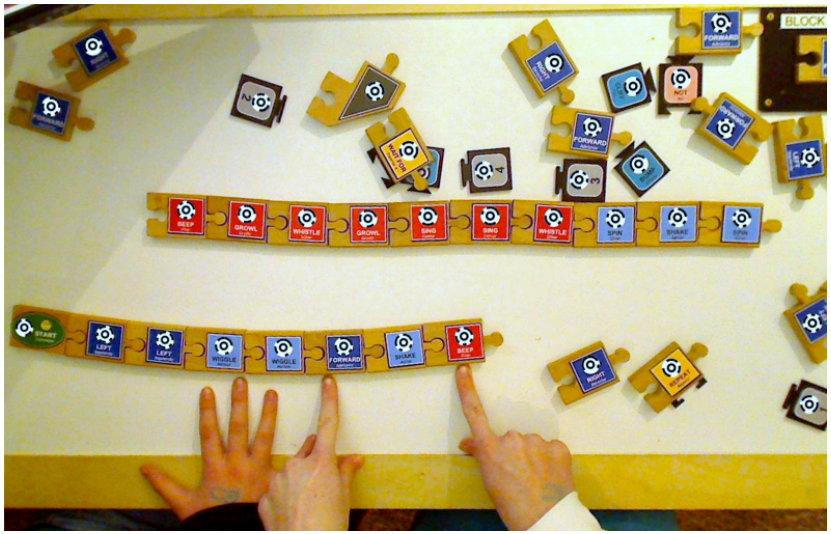


PicoBlocks Passes the “Lucite Test”



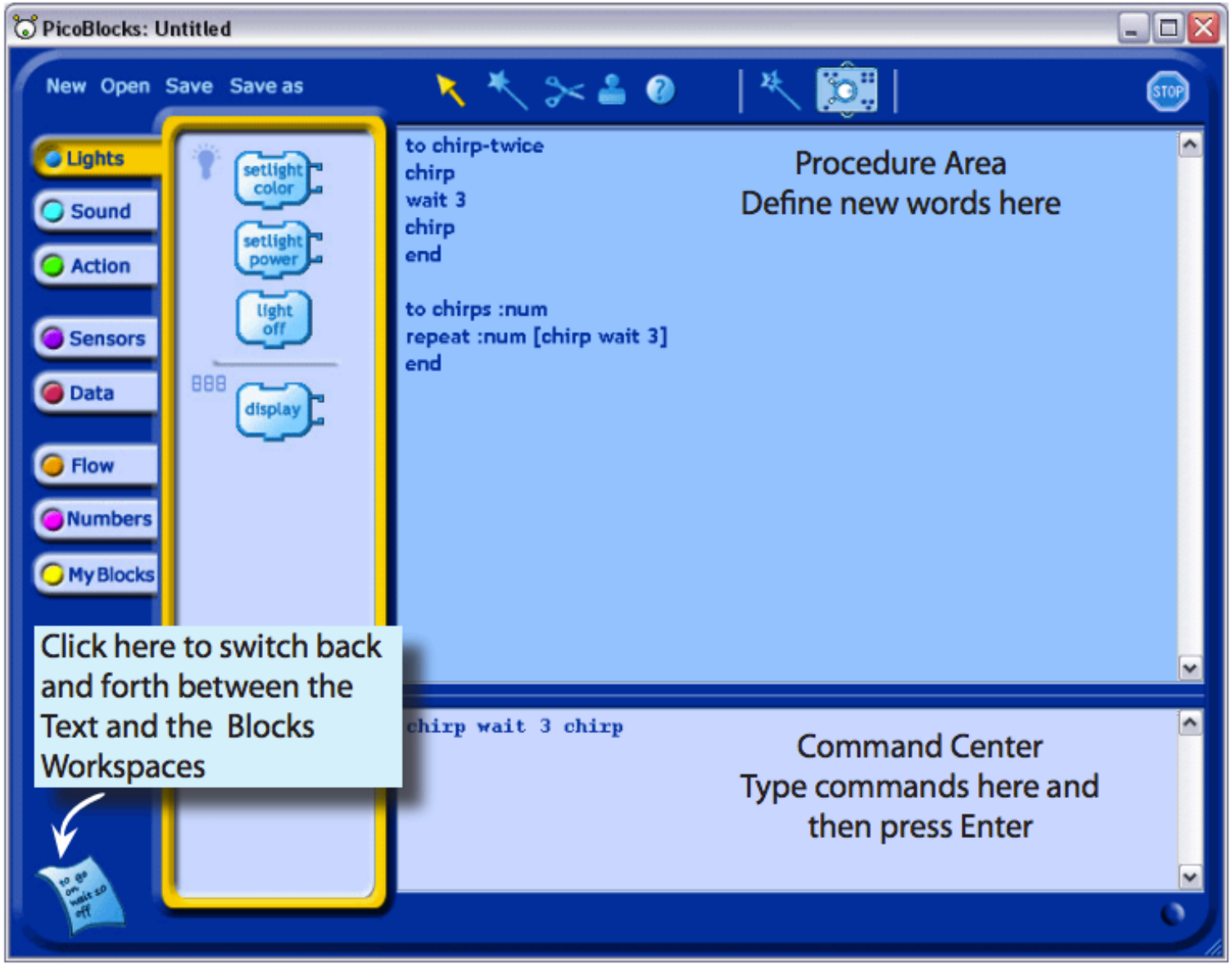
Languages with Physical Blocks

Robot Park (Horn, Solovey, & Jacob, 2007)



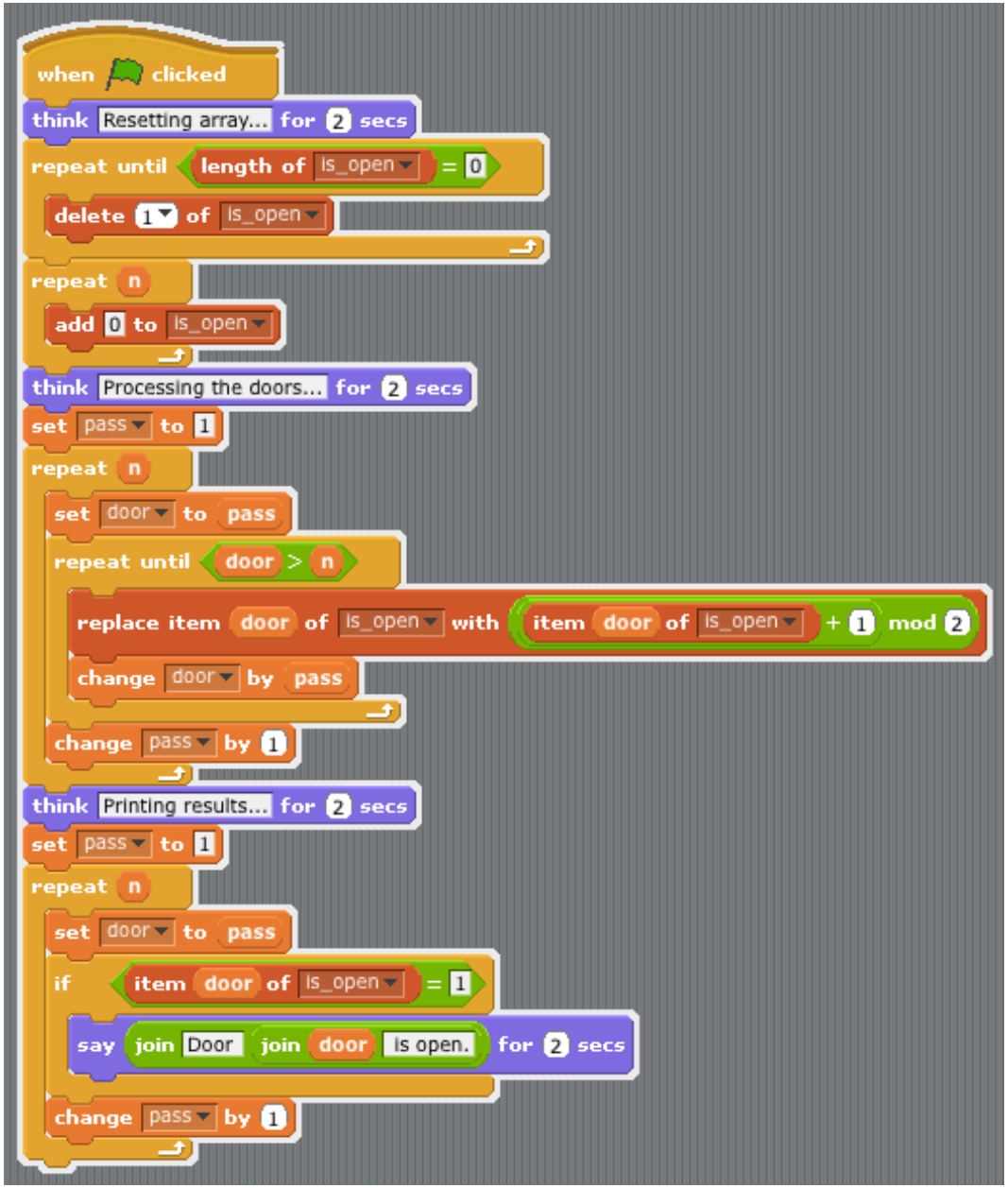
Tangible Kindergarten (Bers and Horn, 2009)

PicoBlocks Text/Extension Language

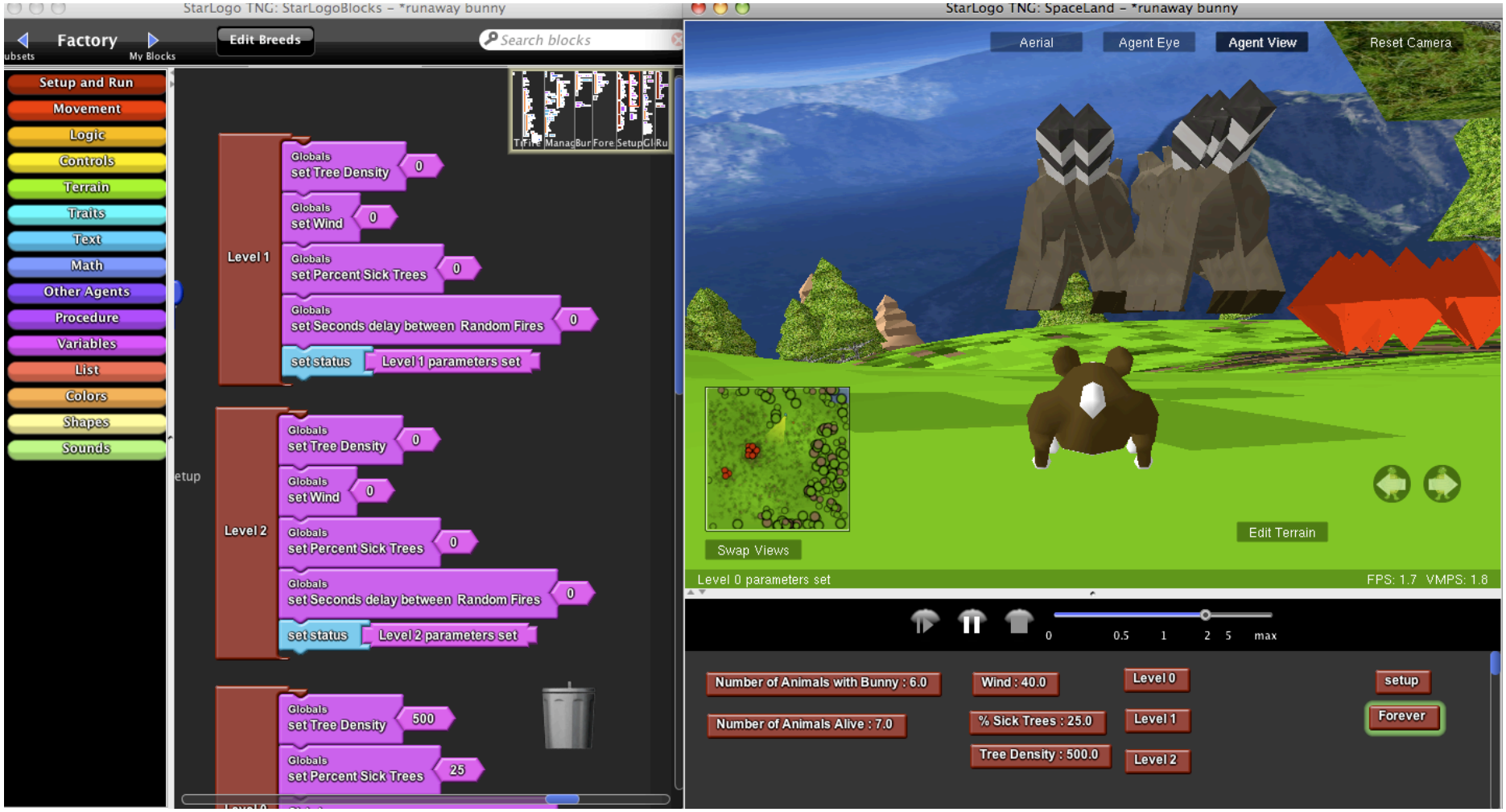


Scratch (Resnick et al., 2007)

Scratch (Resnick et al., 2007)

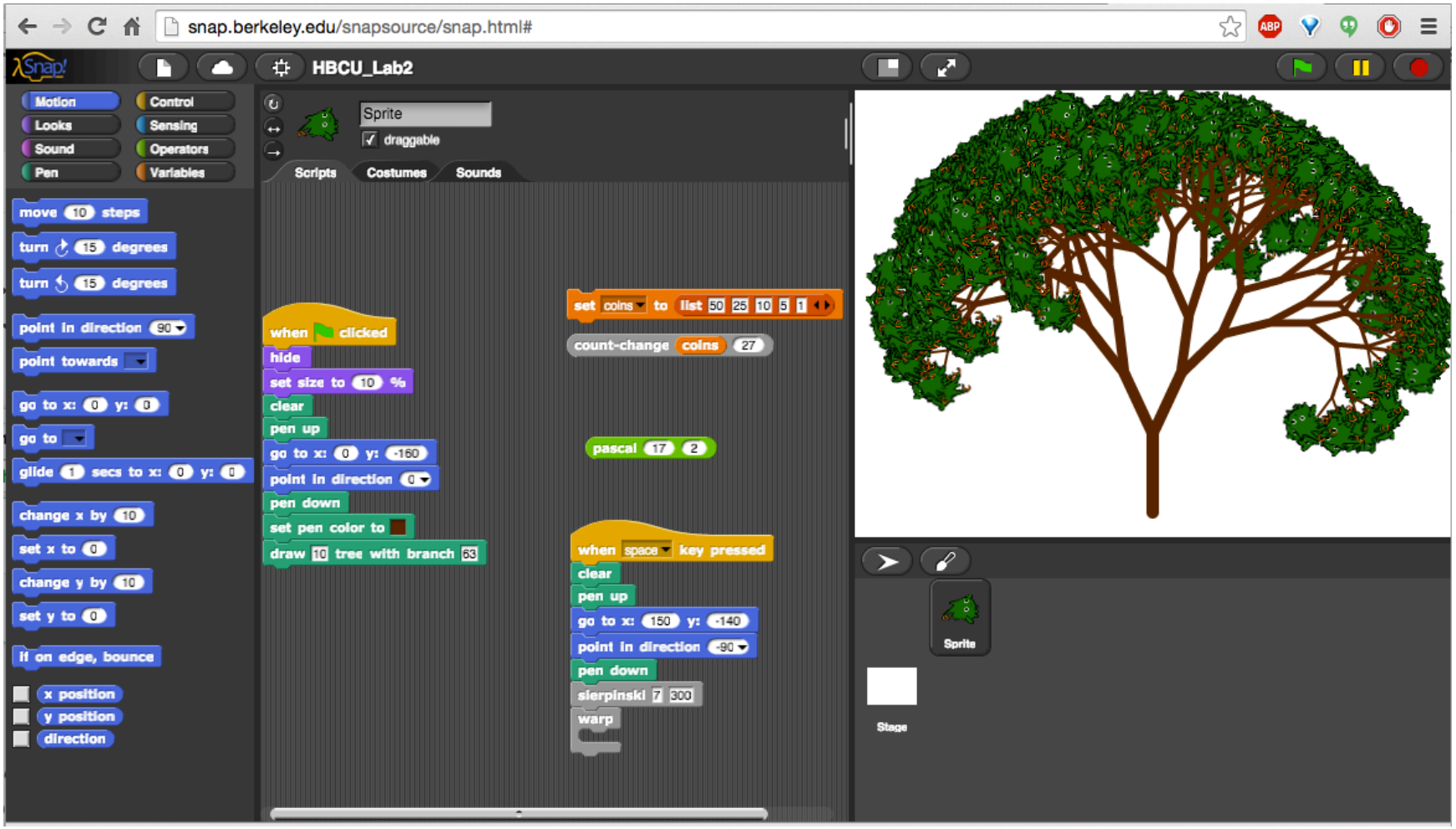


StarLogo TNG (Roque, Wendel, et al., 2007)



- Different plug shapes for different expression types: number, boolean, string, list
- Source of the OpenBlocks Java-based blocks framework

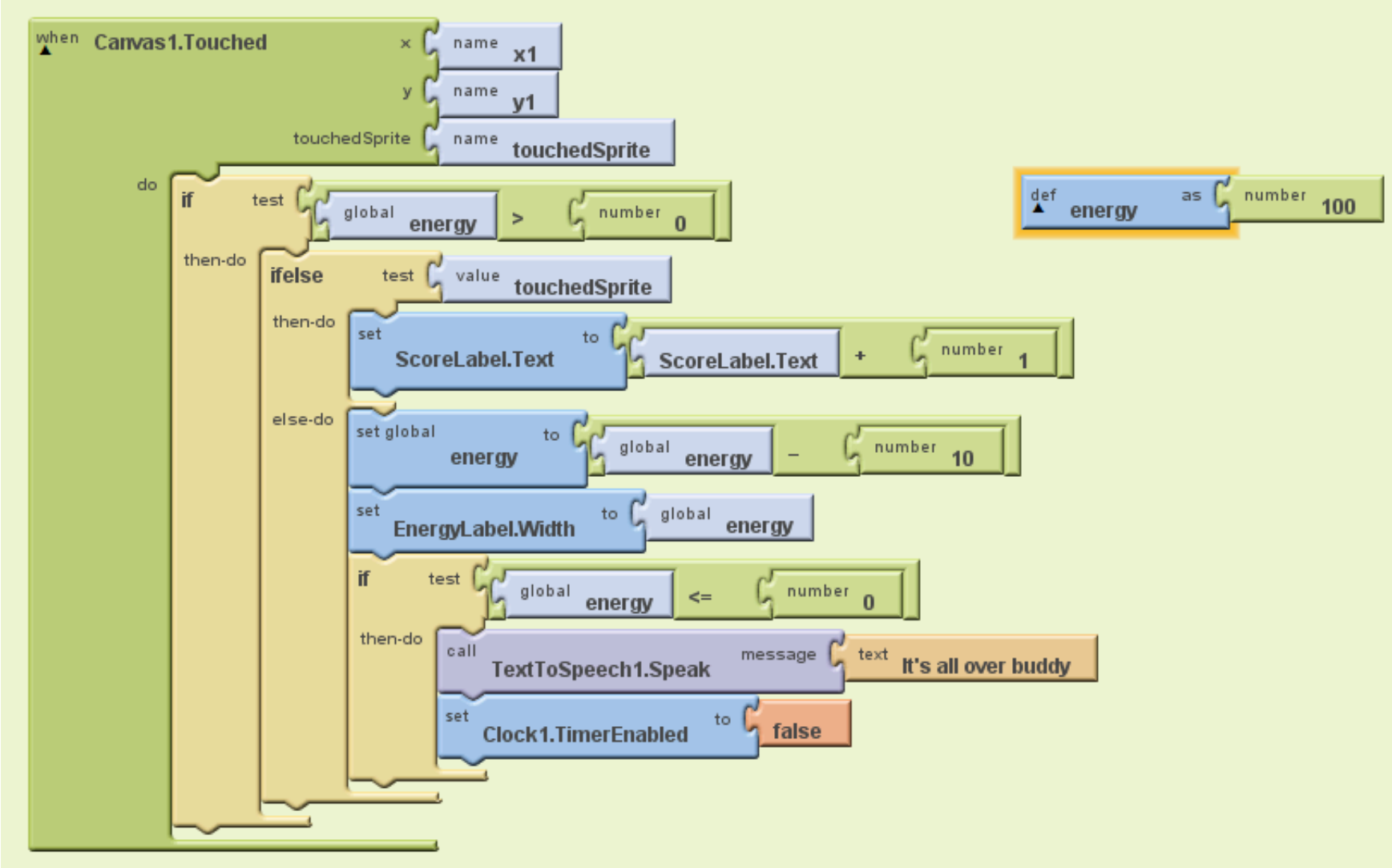
BYOB/Snap! (Harvey, Moenig, et al., starting 2008)



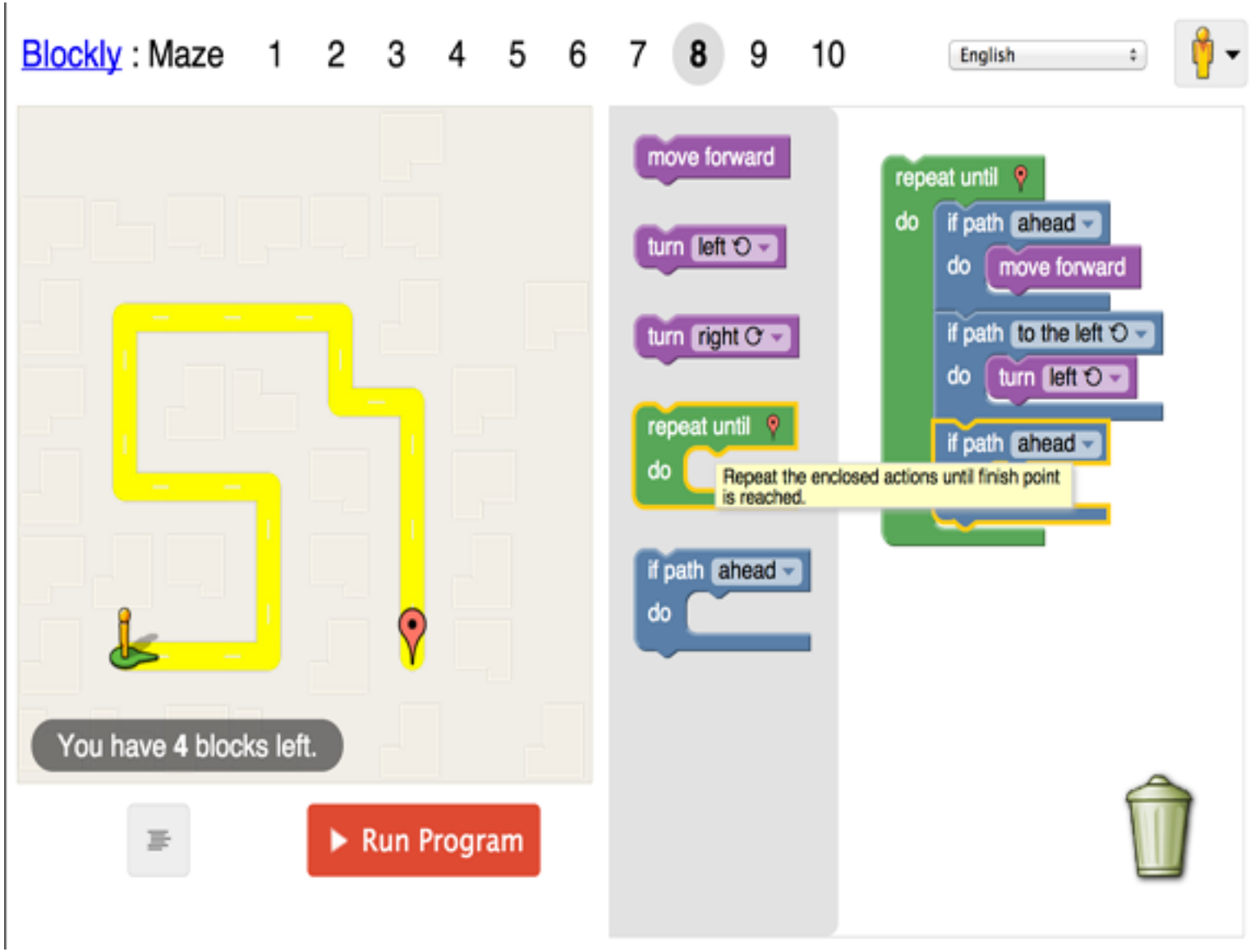
BYOB/Snap! Have First-class Functions

App Inventor Classic (Abelson et al., 2009)

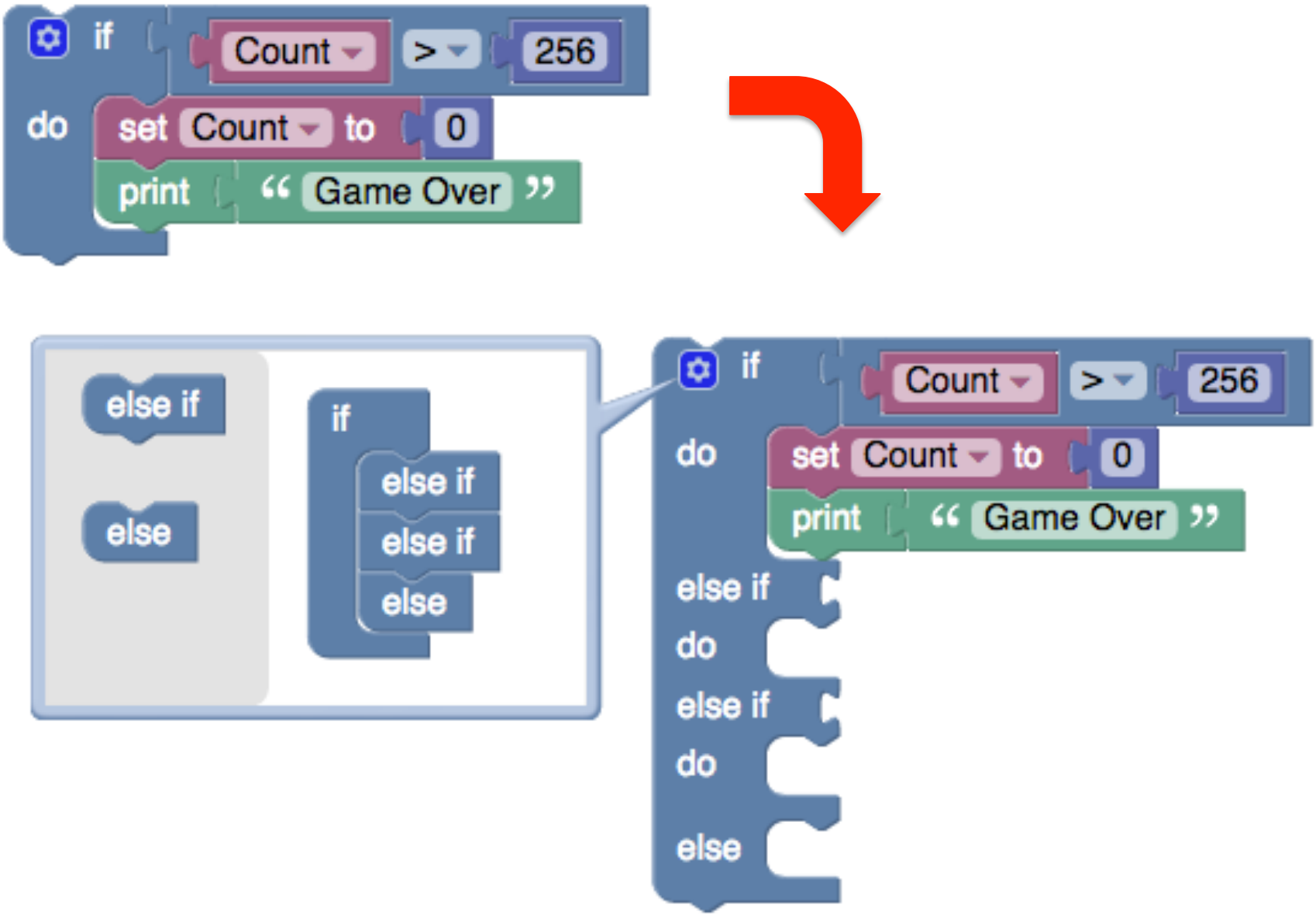
App Inventor Classic Blocks



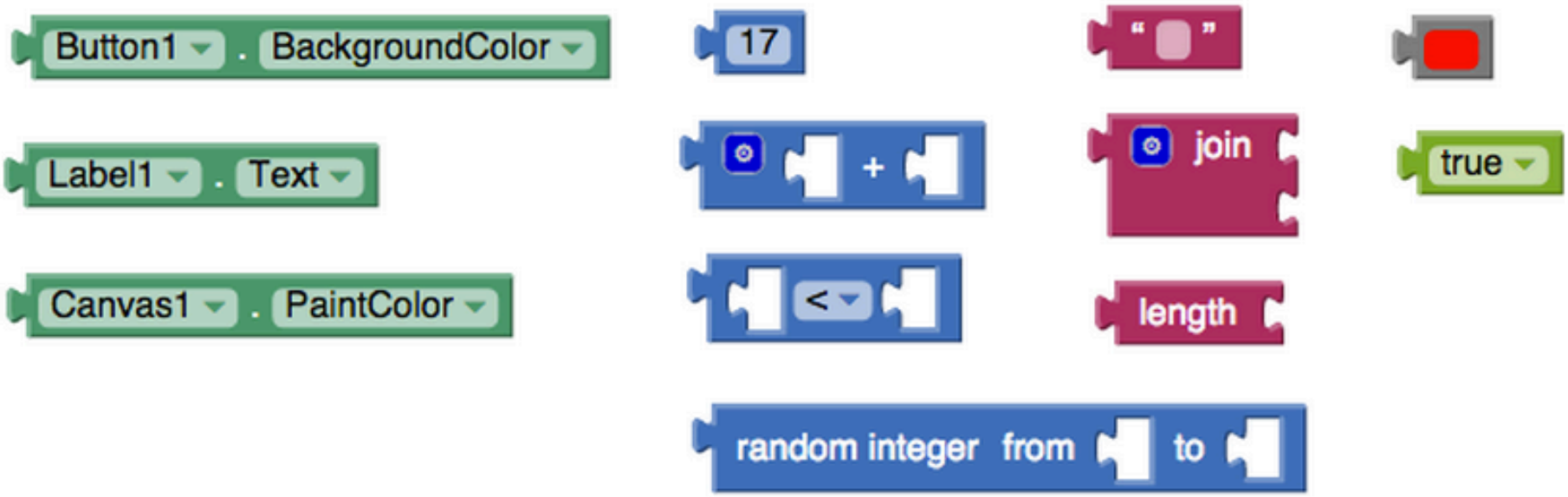
Blockly (Fraser, 2012)



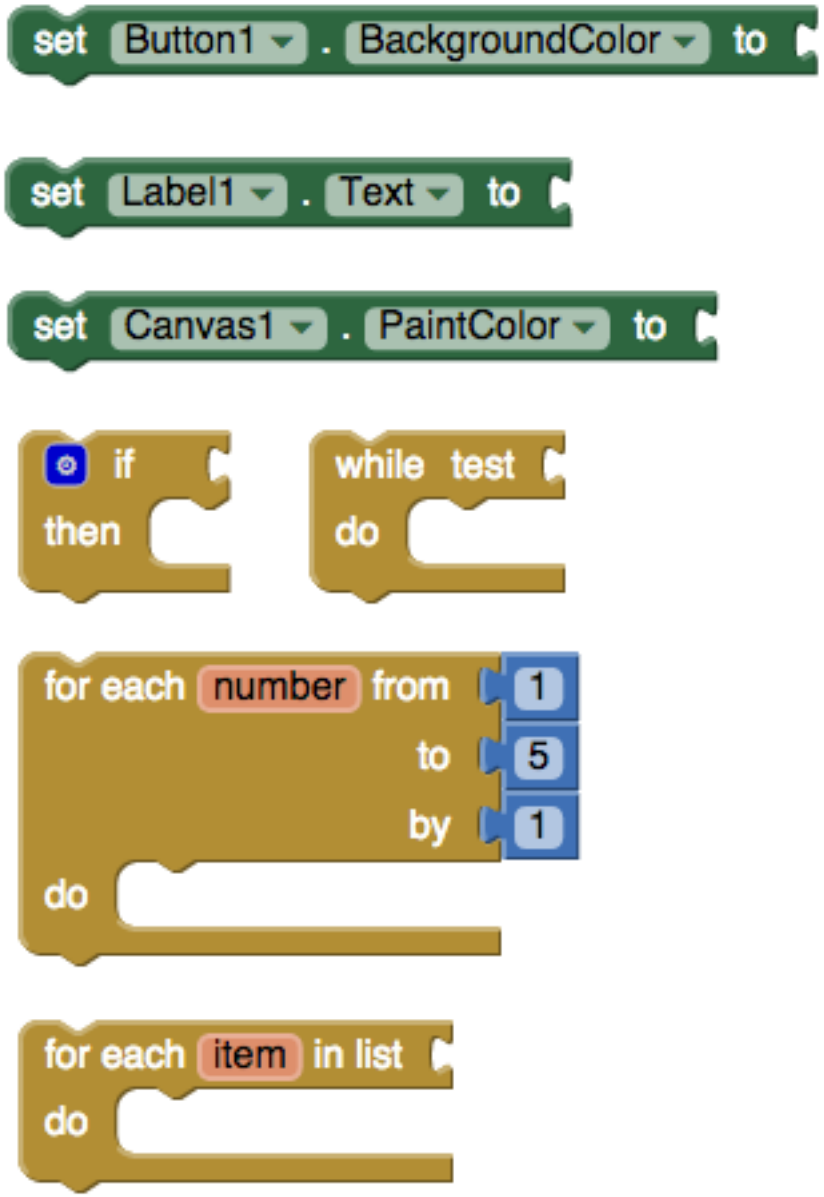
Blockly Mutators



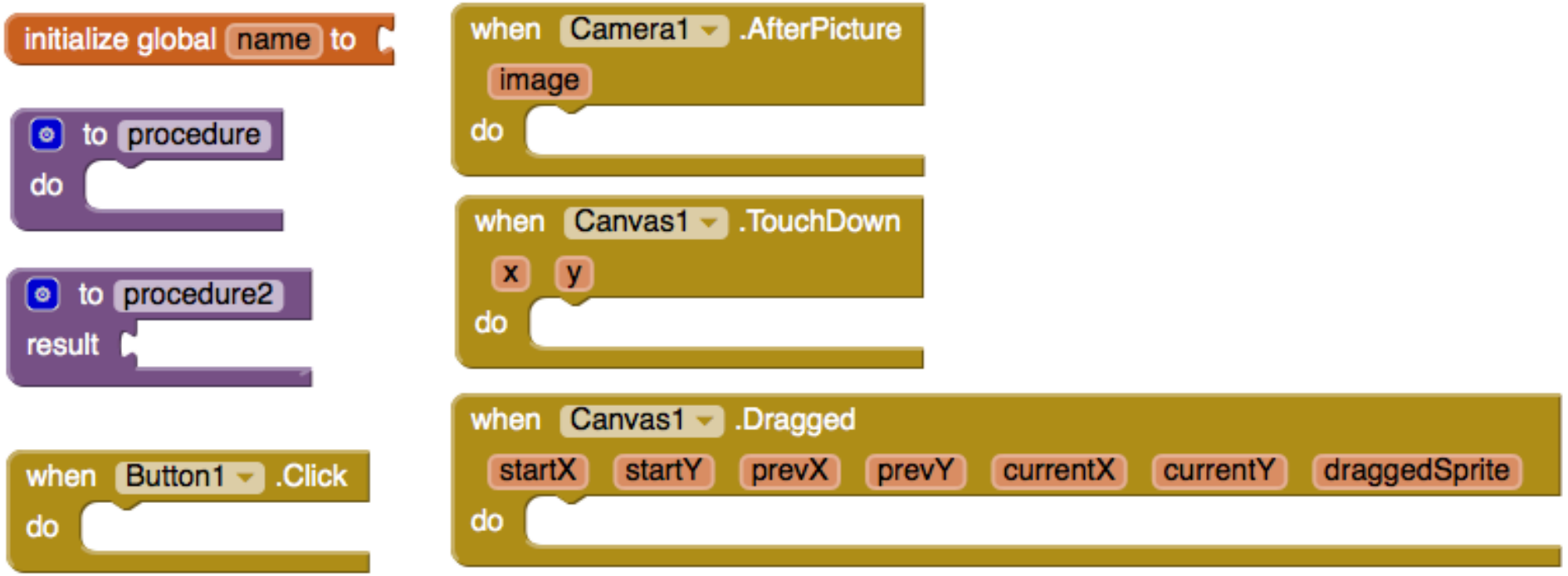
Back to AI: AI Syntax: Expressions



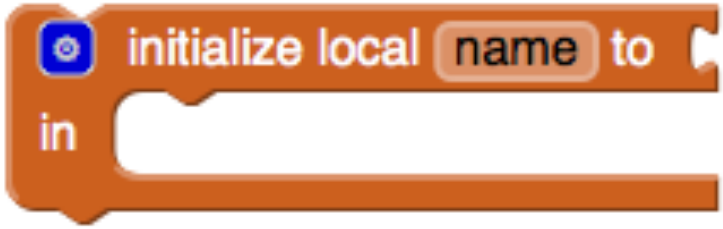
AI Syntax: Statements

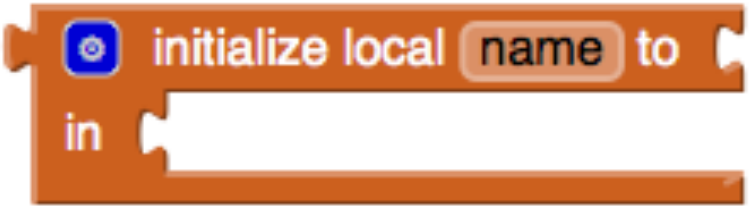


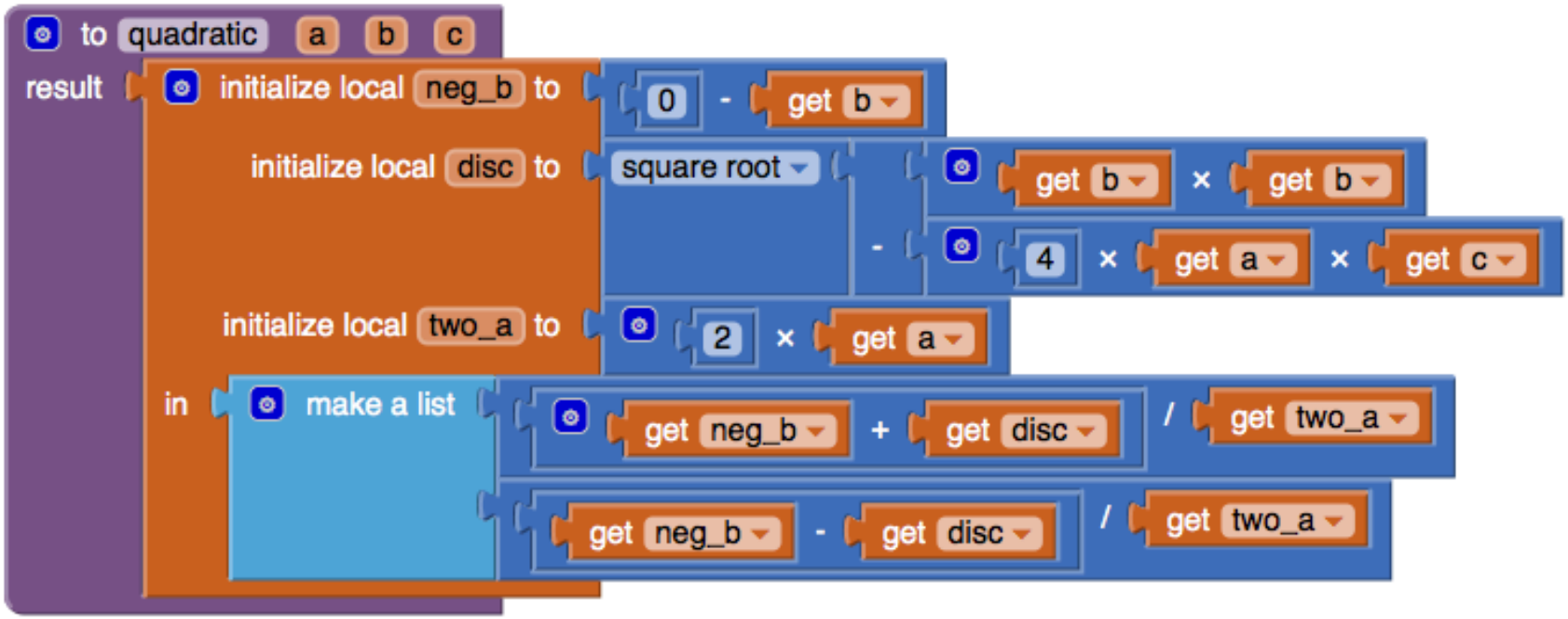
AI Syntax: Top Level Declarations



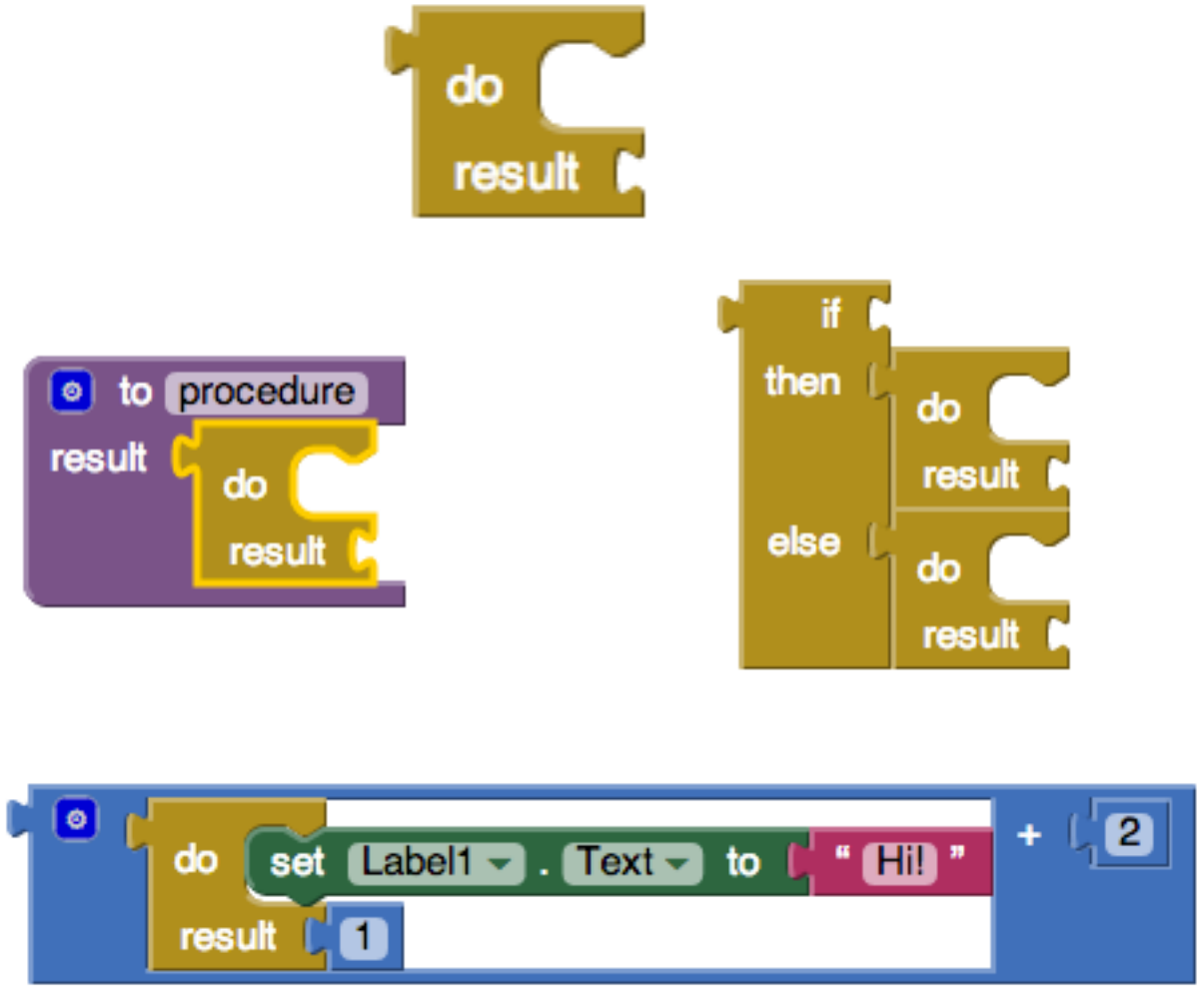
AI Syntax: Local Variable Declarations



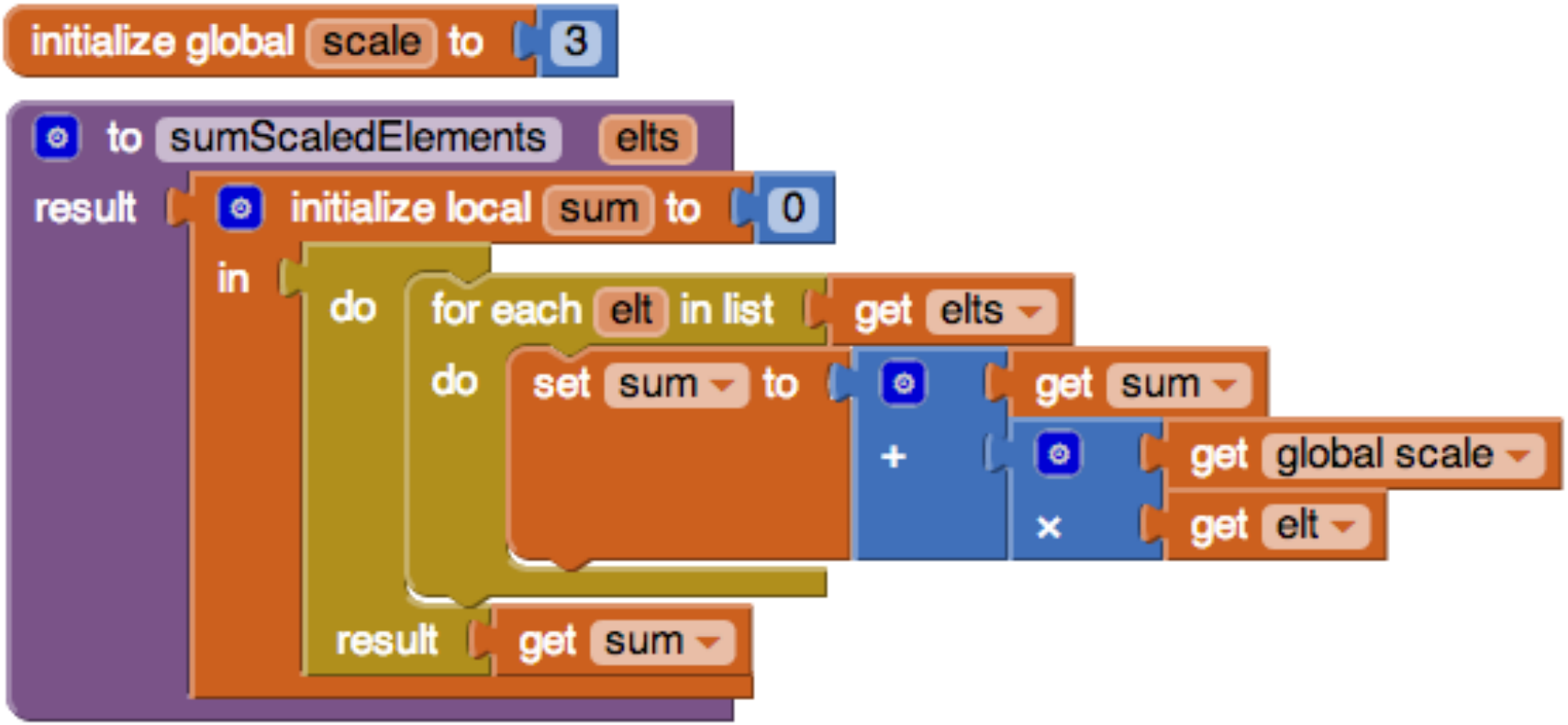




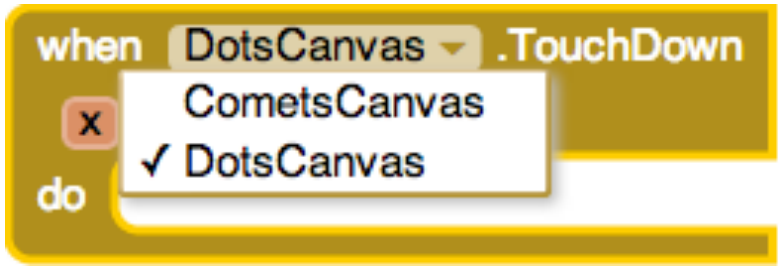
AI Syntax: Performing actions before returning value

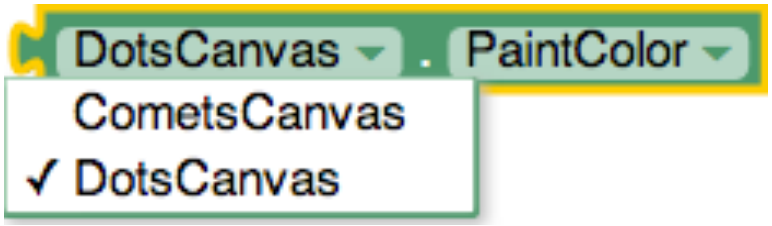


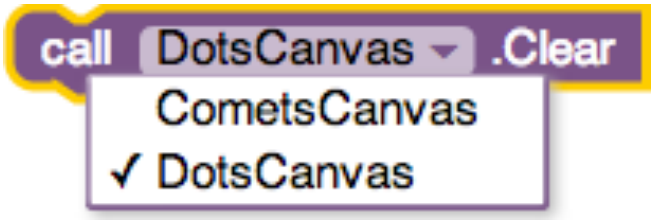
AI Syntax: All Together Now

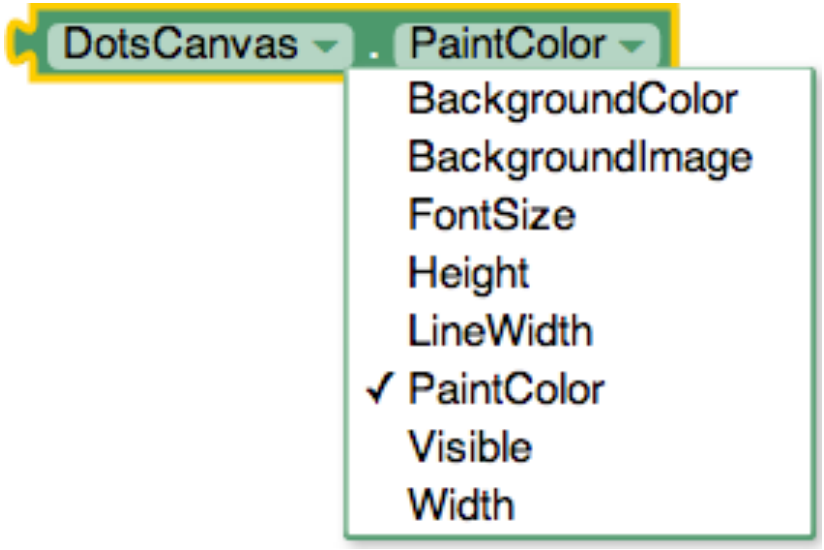


Drop-Downs Reduce Errors & Viscosity







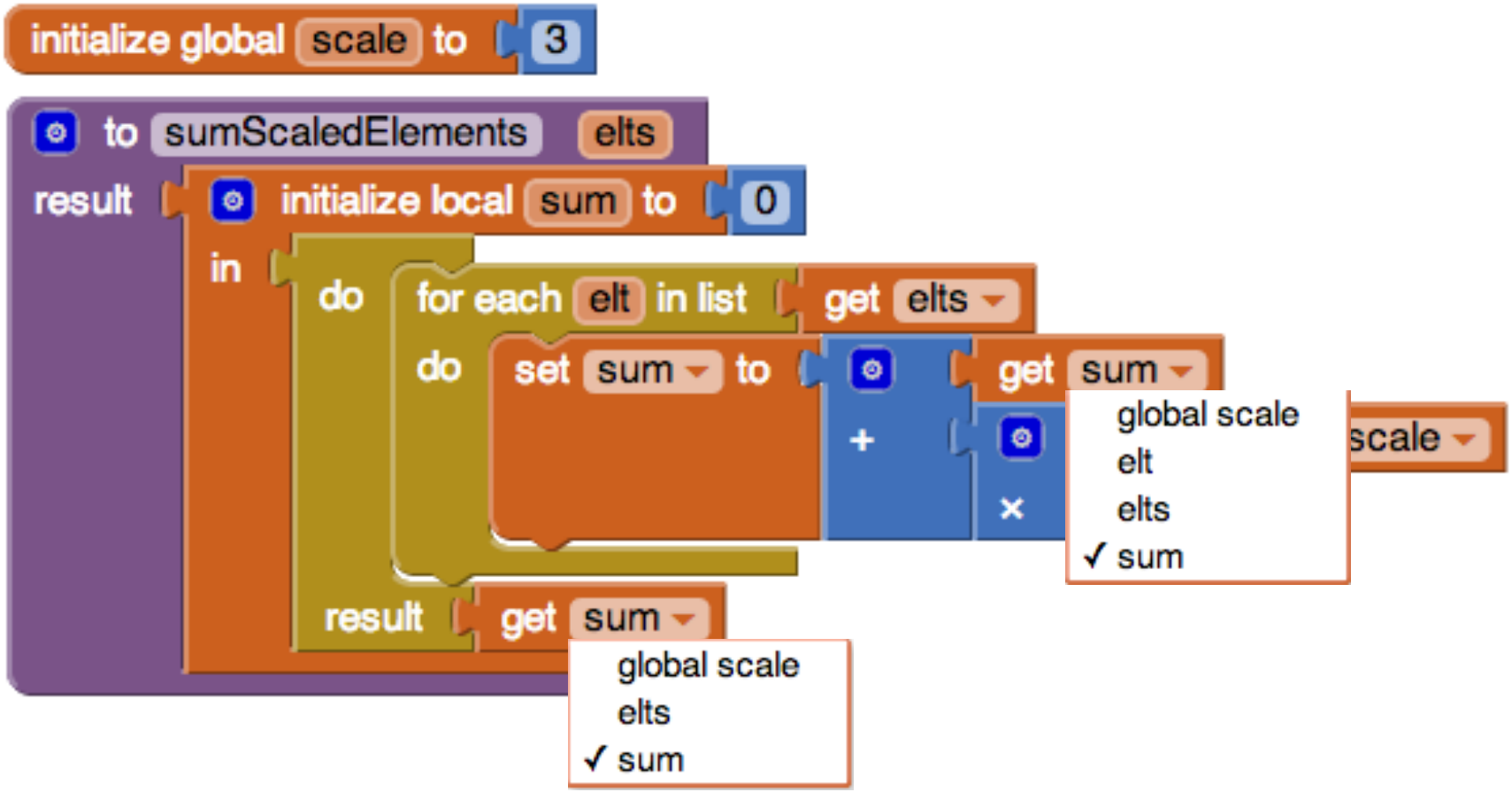


Talk Road Map

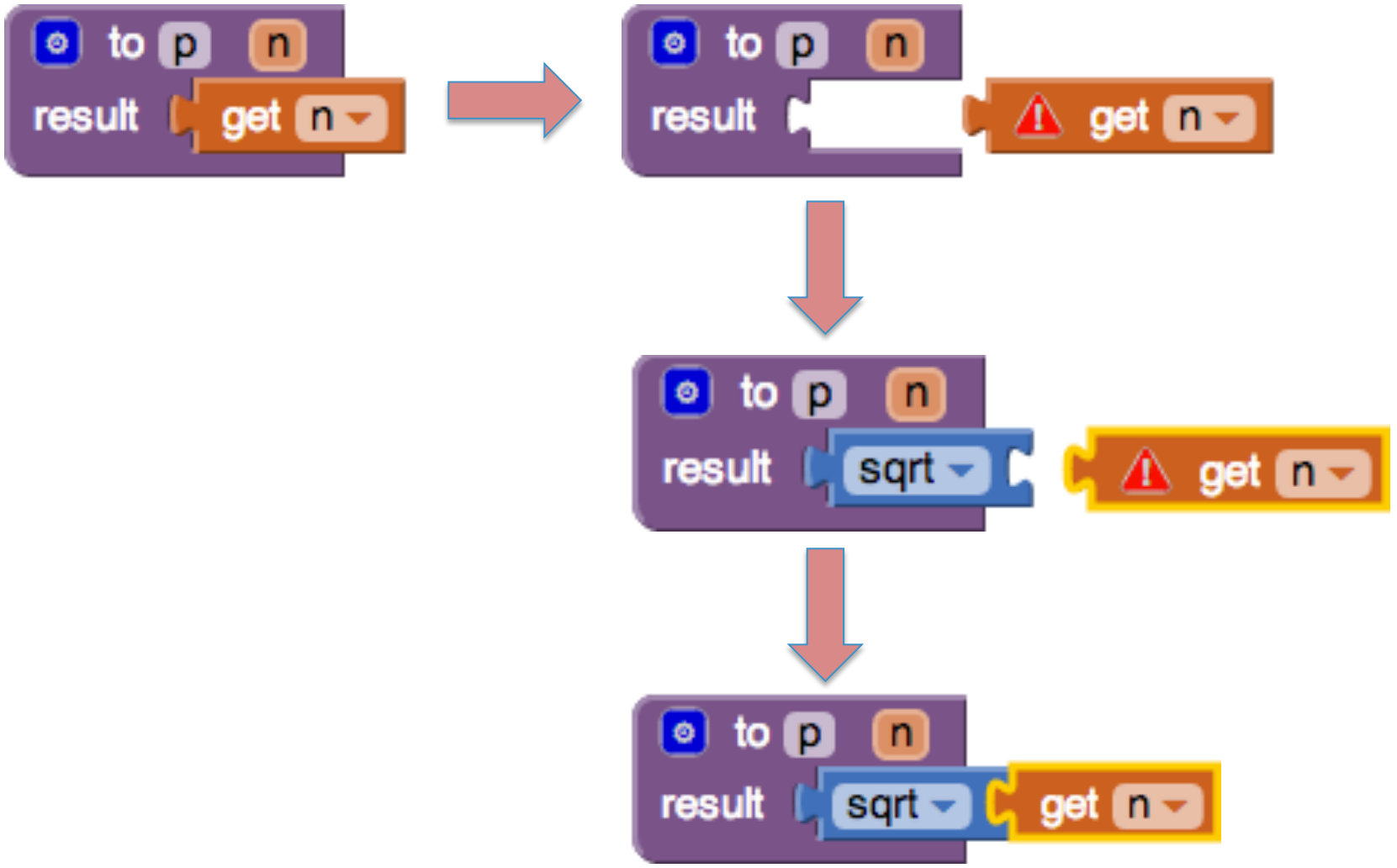
- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Name Scoping in AI

- Globals are in a separate namespace
- Indentation visually highlights area of name scope
- Drop-downs list only names in scope.
- Inner names can shadow outer ones
- Changing declared names automatically consistently changes all references

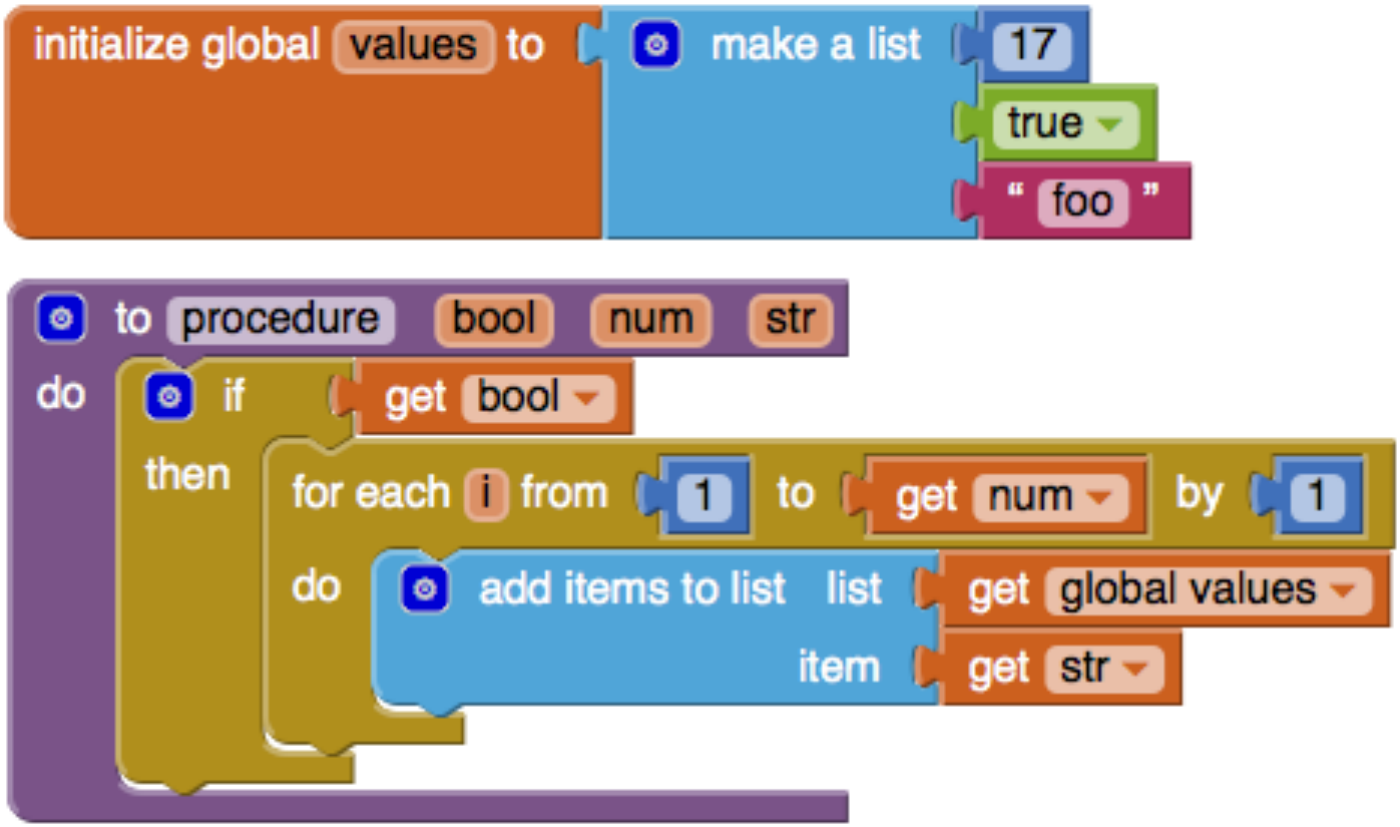


Handling Unbound Names



What About Types?

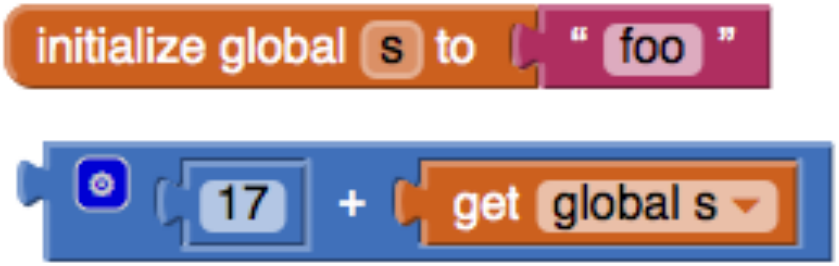
App Inventor is dynamically typed, so there's only one plug shape:



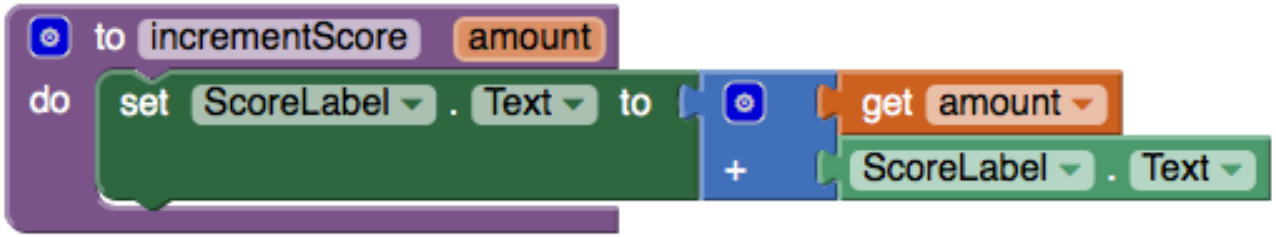
Simple “Soft” Static Type Checking

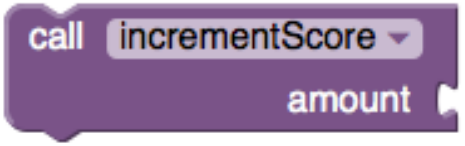
Type errors at block connection time are prohibited by “repulsion”

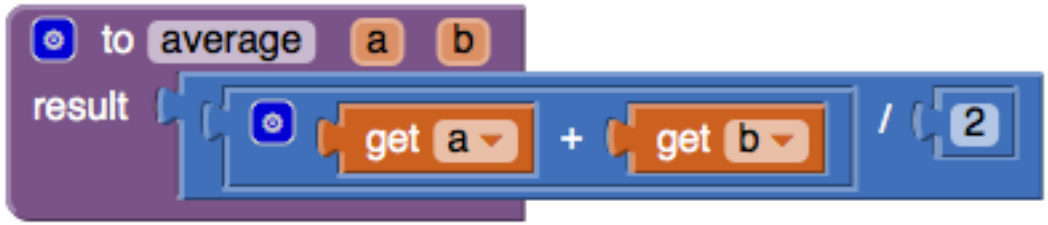
Dynamic type errors can be hidden by variables:

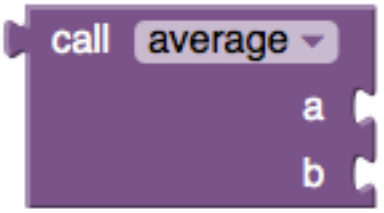


Distinguishing Void and Fruitful Procedures



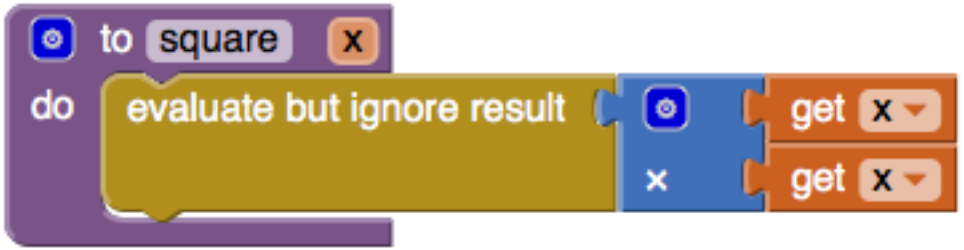


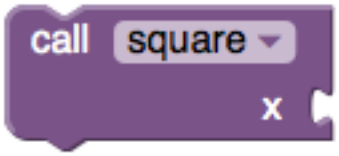




Python function gotcha

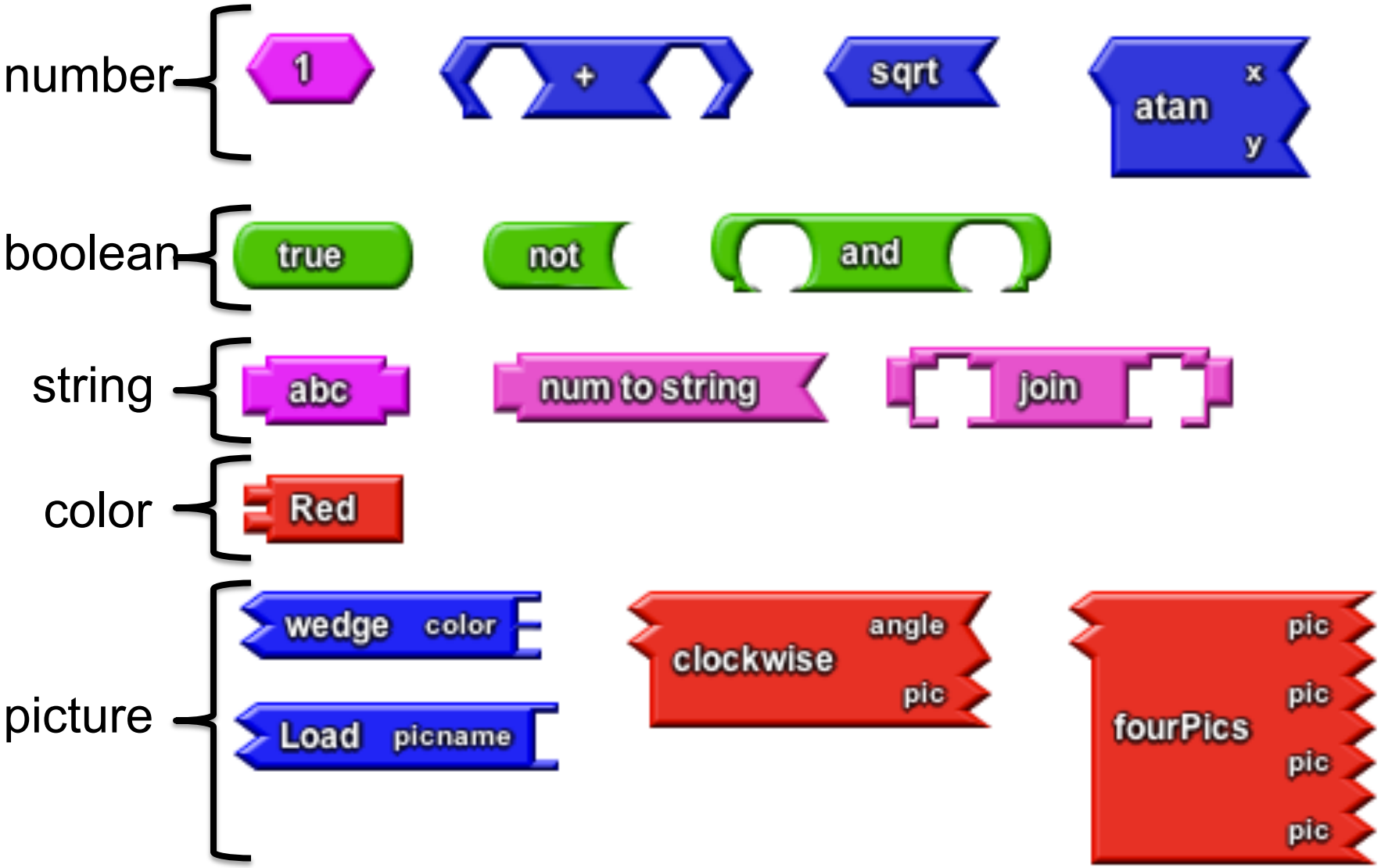
```
>>> def square (x):
...    x * x
...
>>> square(5)
>>>
```



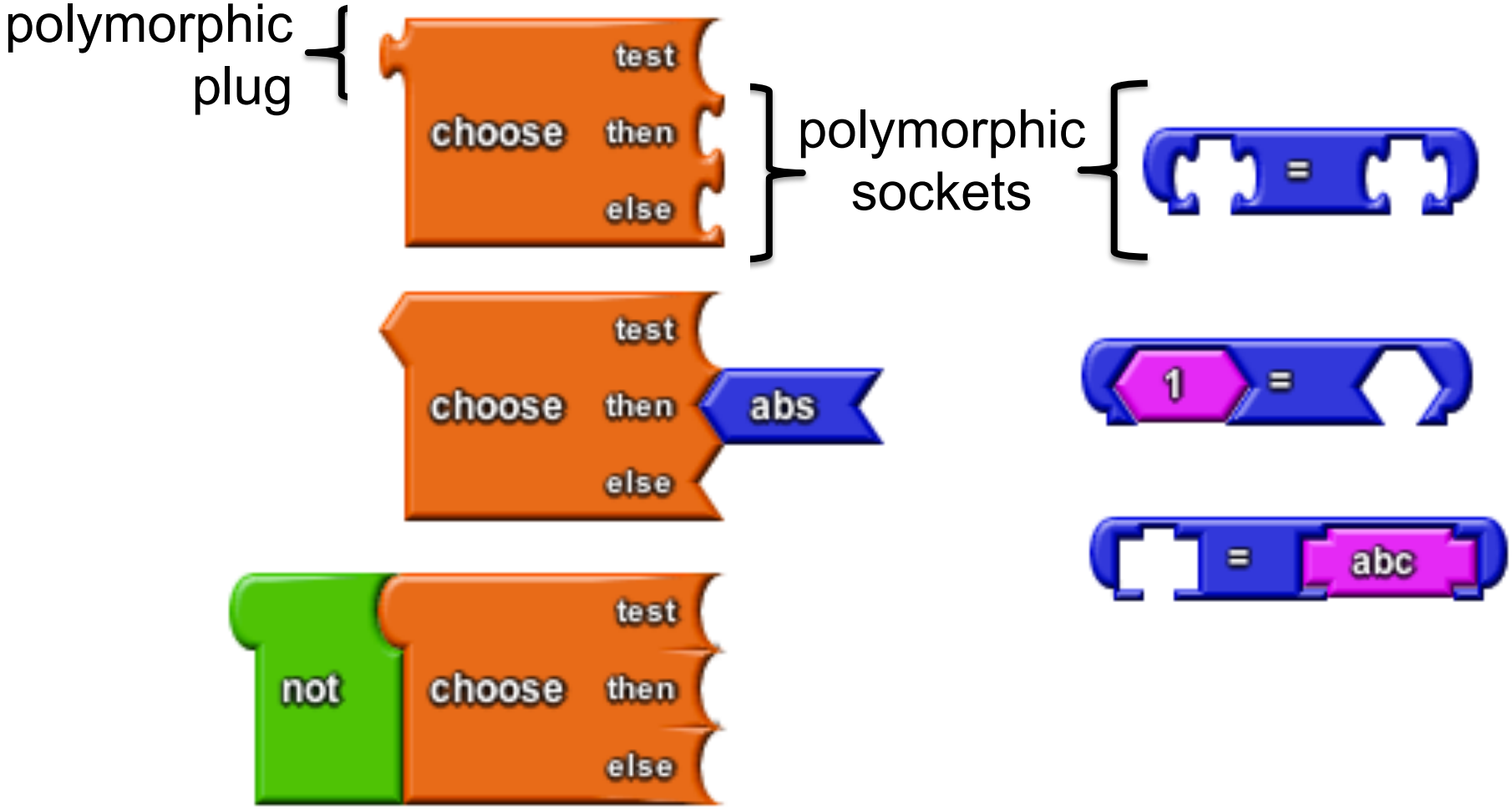


Connector Shapes in PictureBlocks

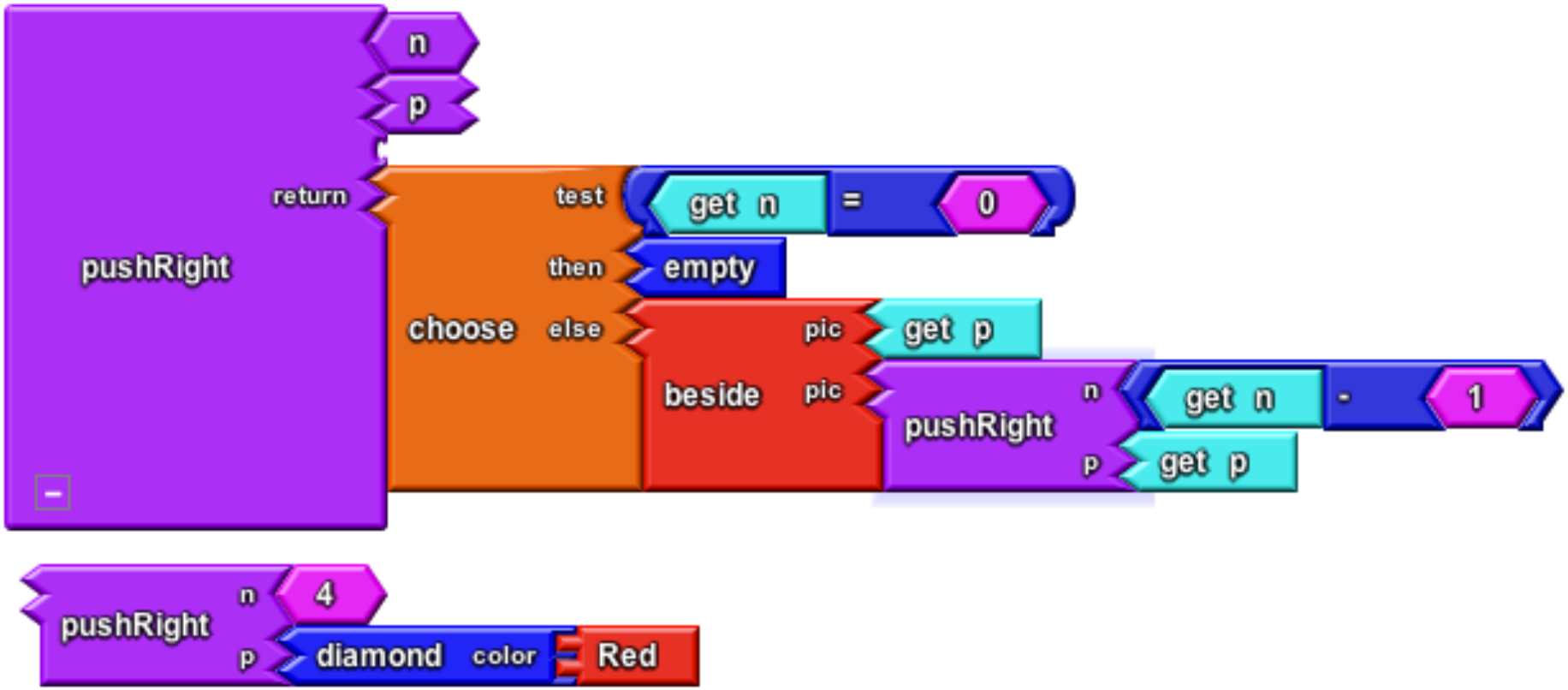
(Similar to types-as-shapes in StarLogo TNG)



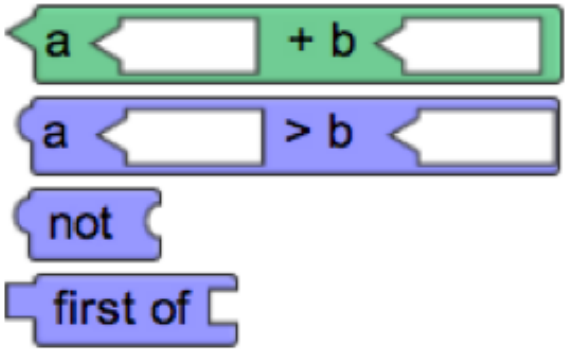
Polymorphism in PictureBlocks

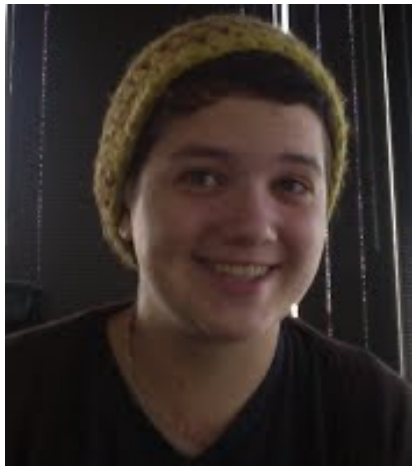


pushRight: Complete Declaration and Call

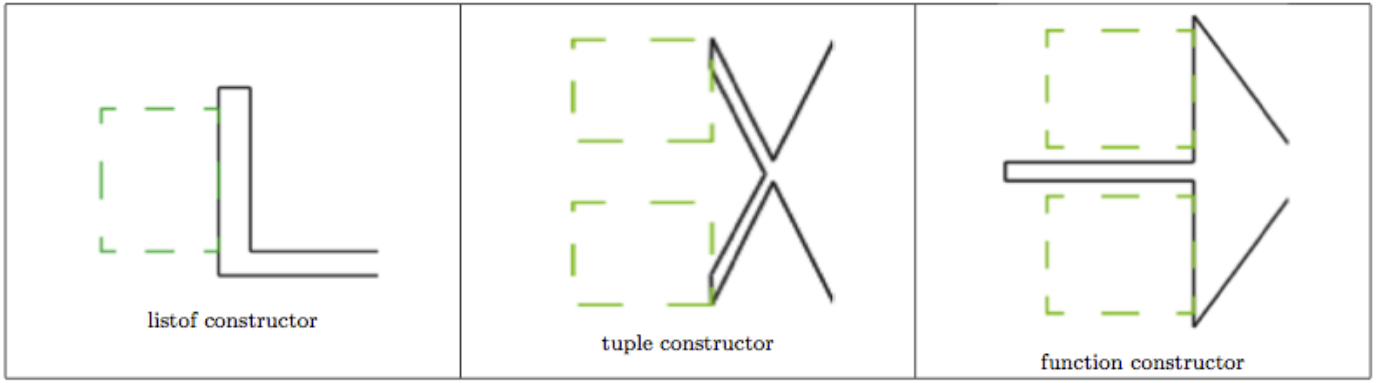


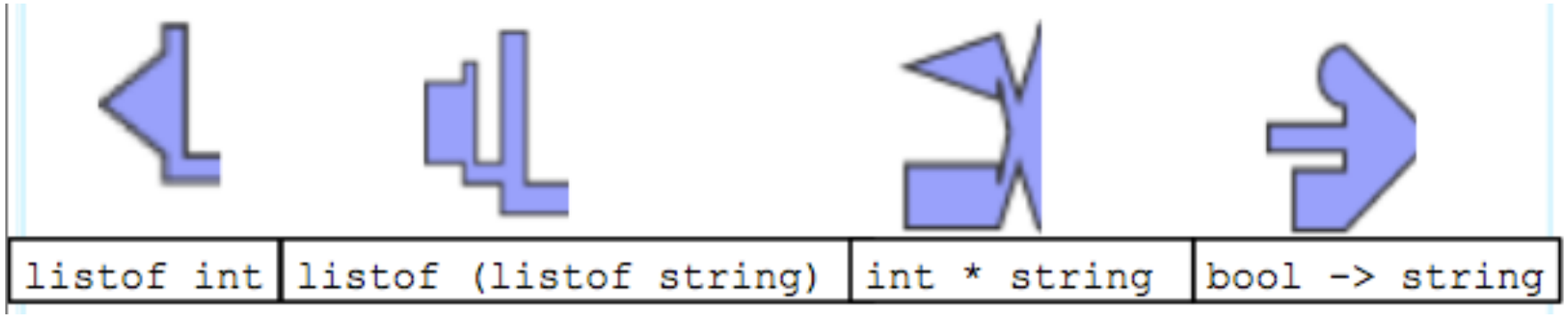
Type Blocks



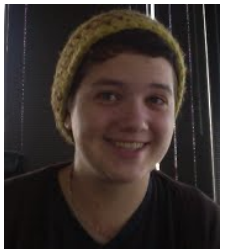


Marie Vasek '12
Wellesley





Type Blocks: More Examples



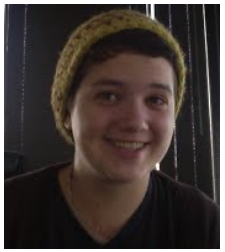
`listof (string * boolean)`

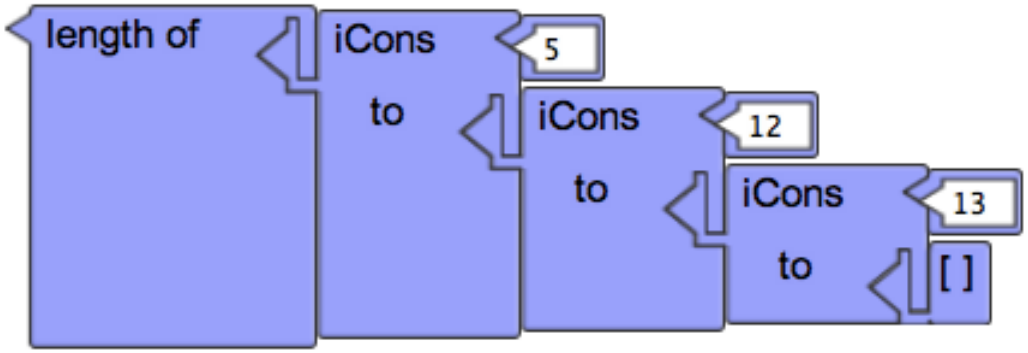
`(listof string) * boolean`

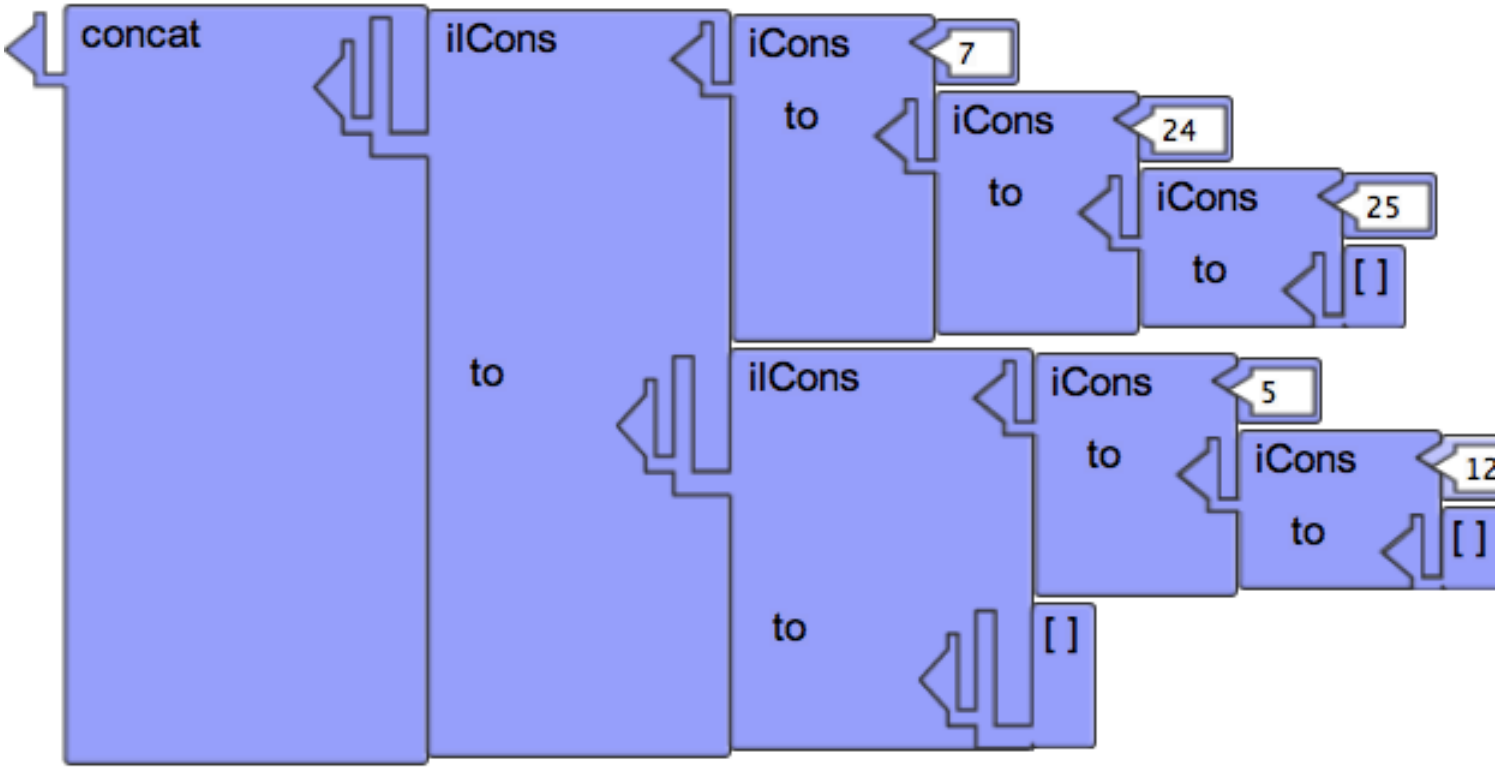
`boolean  * (string -> listof number)`

`(boolean  * string) -> (listof number)`

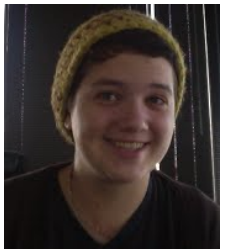
Type Blocks: Lists

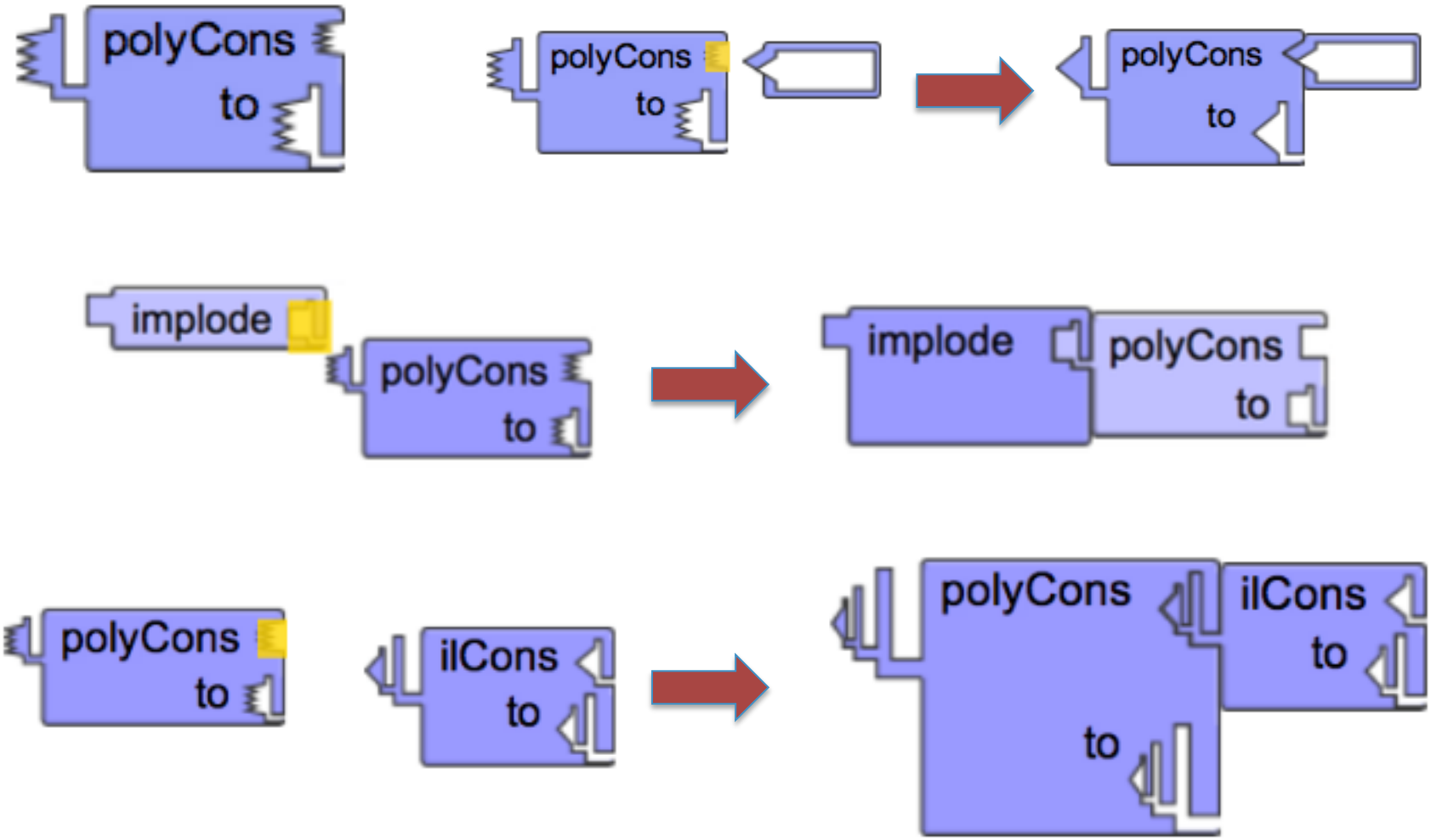






Type Blocks:
ML Style Universal Polymorphism





Talk Road Map

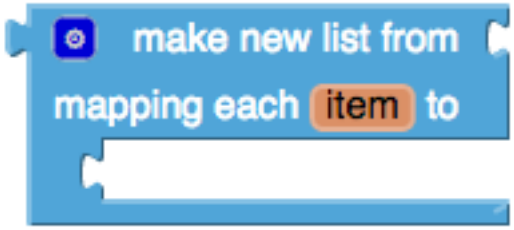
- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

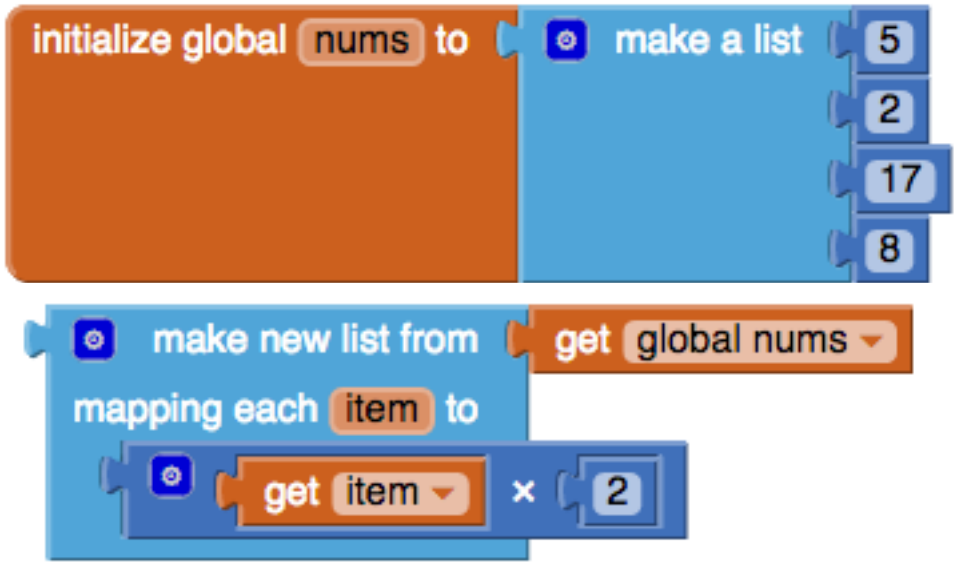
List Mapping

Python:

```
>>> nums = [5, 2, 17, 8]

>>> map(lambda x: x*2, nums)
[10, 4, 34, 16]
```

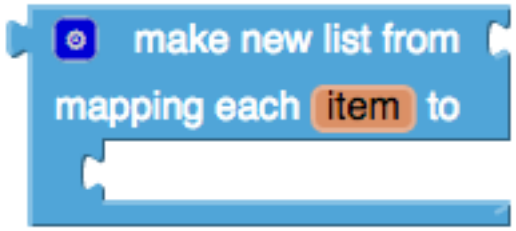
App Inventor doesn't have first-class functions, but can finesse mapping:

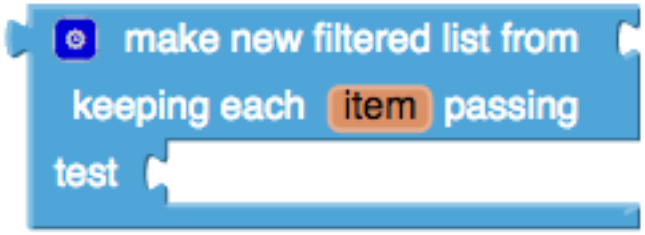


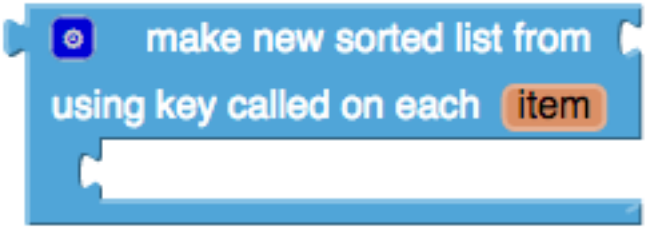


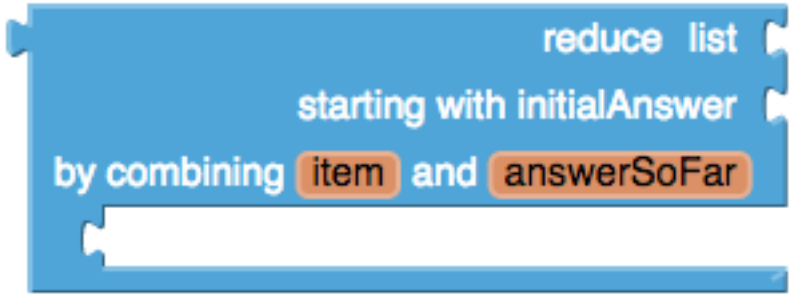
Experimental Higher-Order List Operators in AI

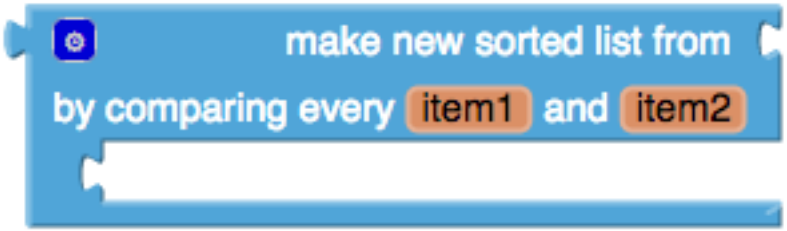
Soojin Kim '15
Wellesley



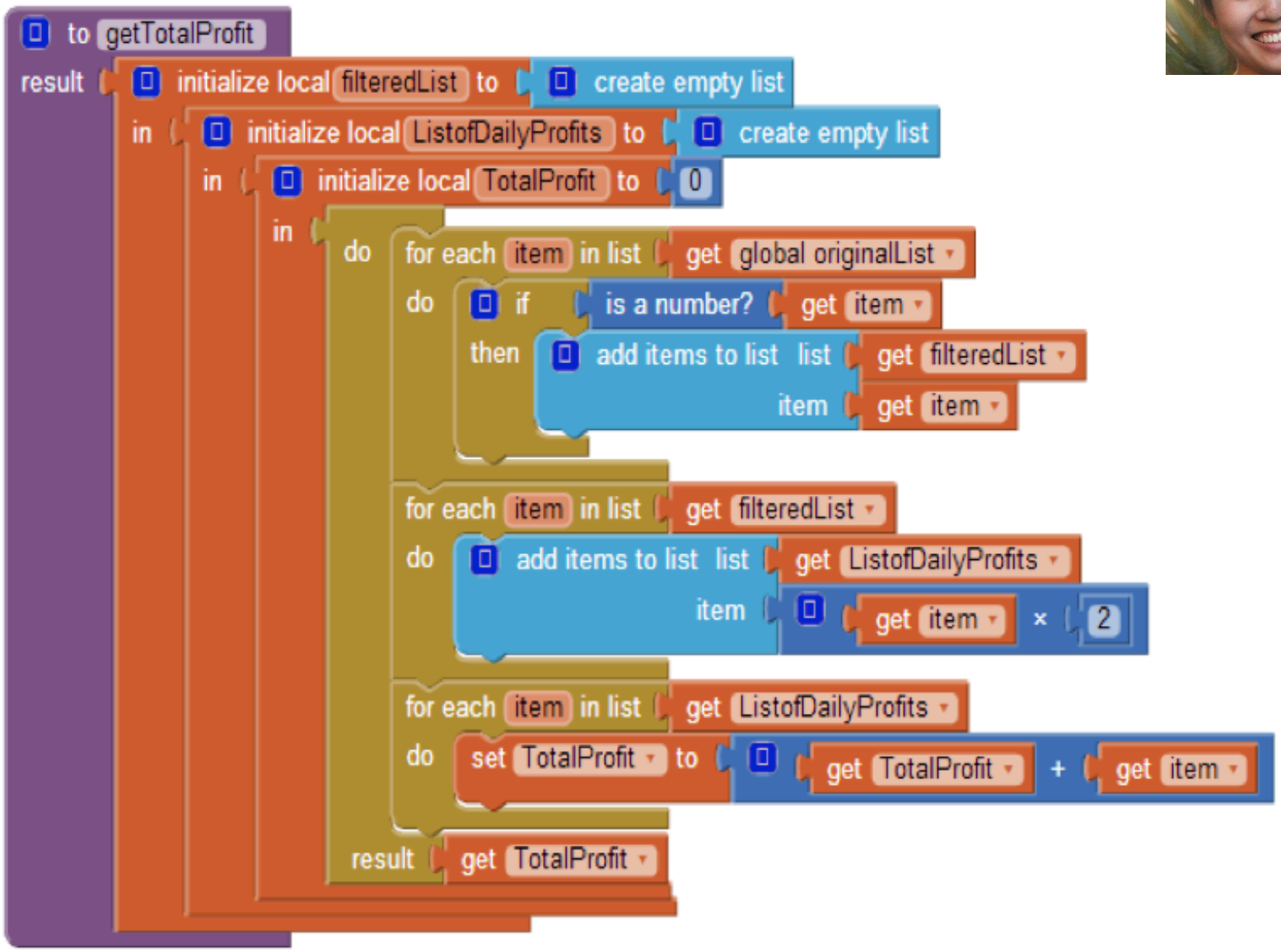




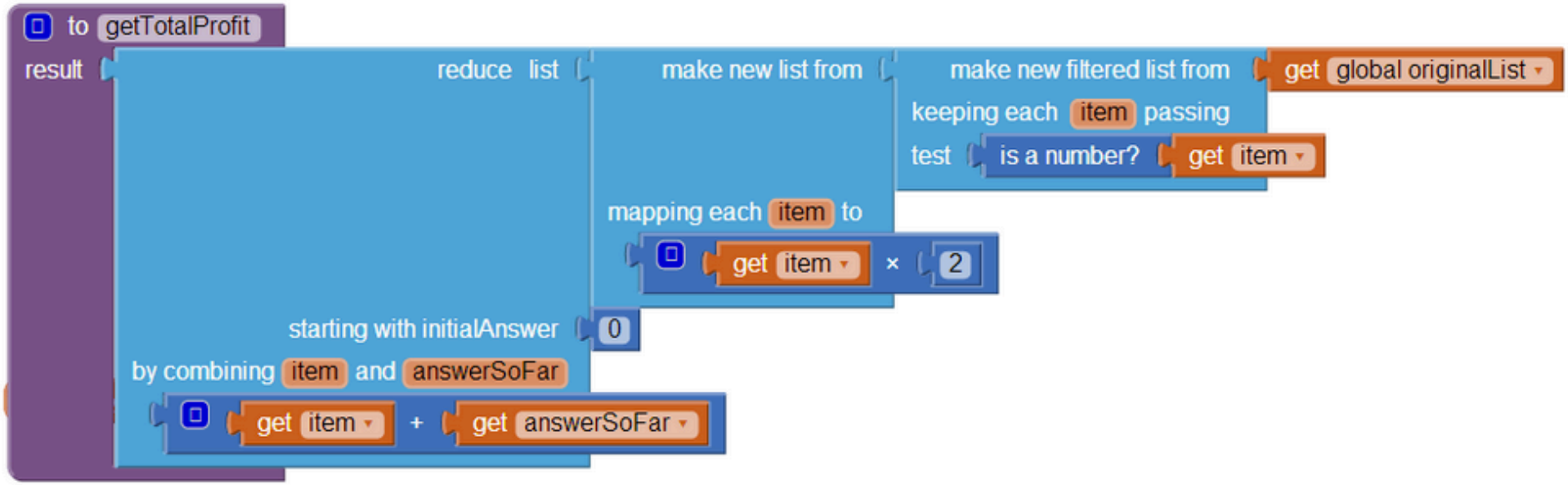




Loop-based List Processing



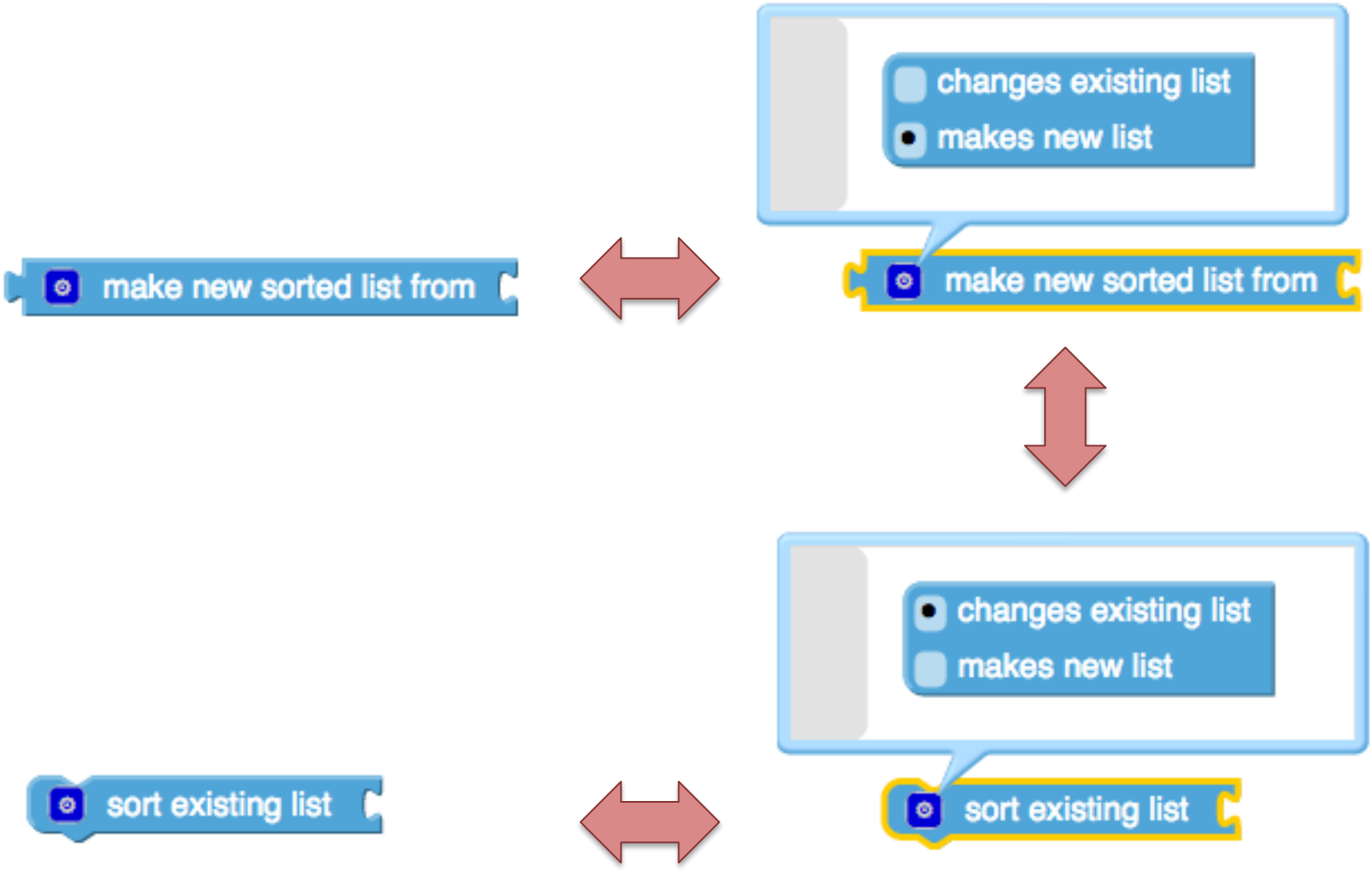
List Processing With Higher-Order Operators



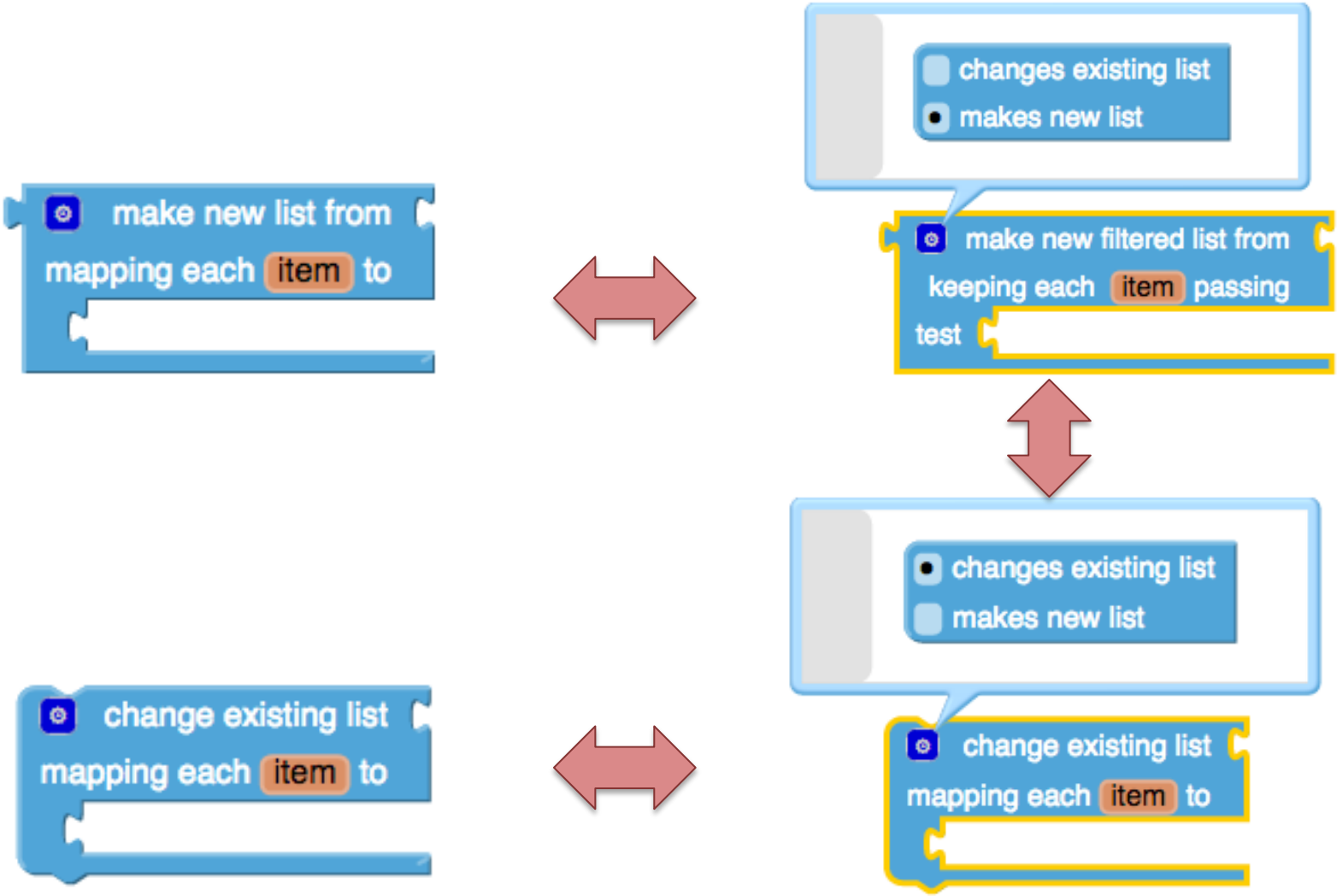
Nondestructive vs. Destructive List Ops In Python

```
>>> elts = [19, True, "foo", 23, "bar", 17, False]

>>> elts.sorted()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute
'sorted'

>>> sorted(elts)
[False, True, 17, 19, 23, 'bar', 'foo']

>>> elts
[19, True, 'foo', 23, 'bar', 17, False]

>>> elts.sort()

>>> elts
[False, True, 17, 19, 23, 'bar', 'foo']
```

Nondestructive vs. Destructive Sorting In AI



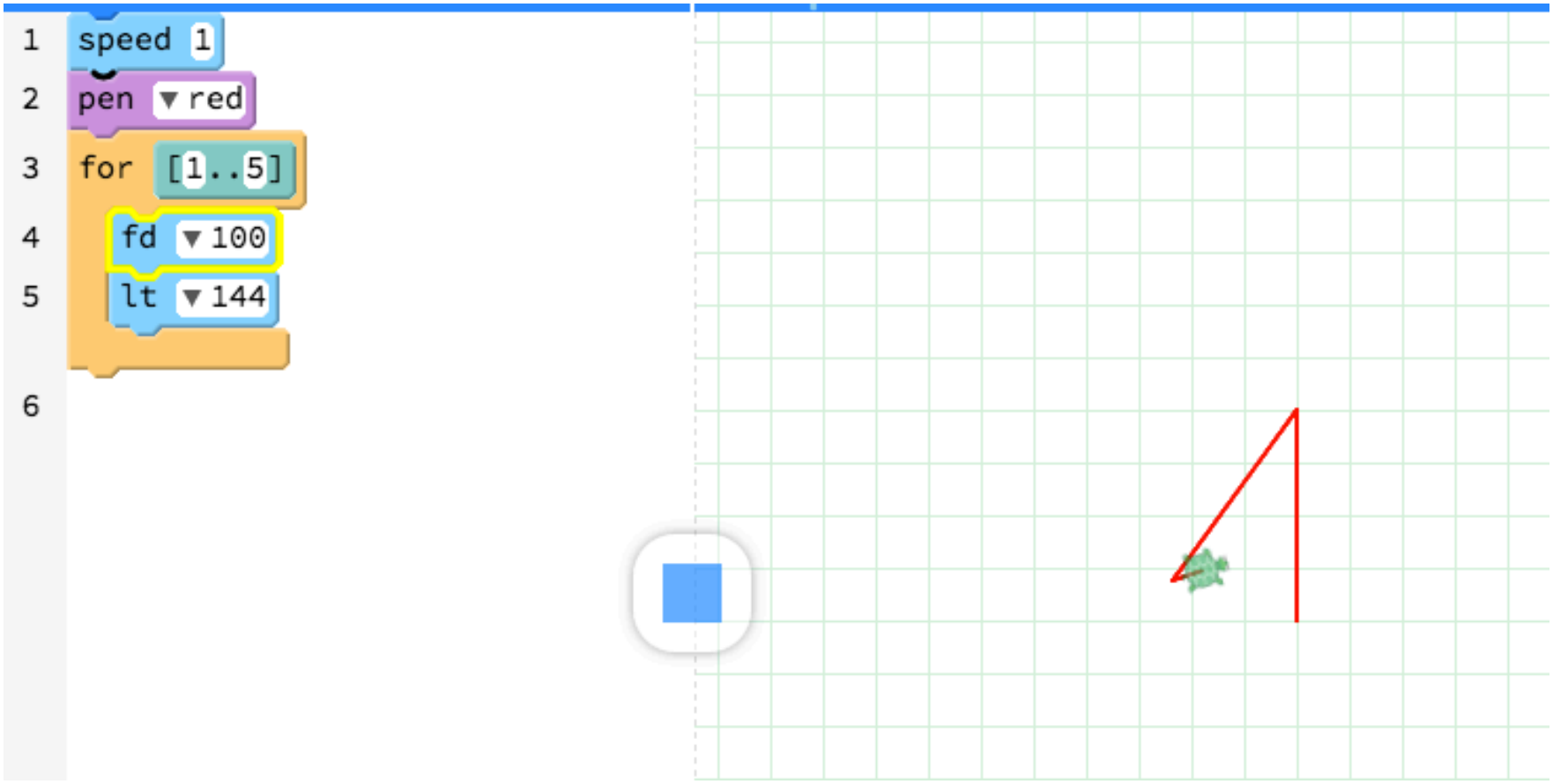
Other Nondestructive vs. Destructive List Ops In AI



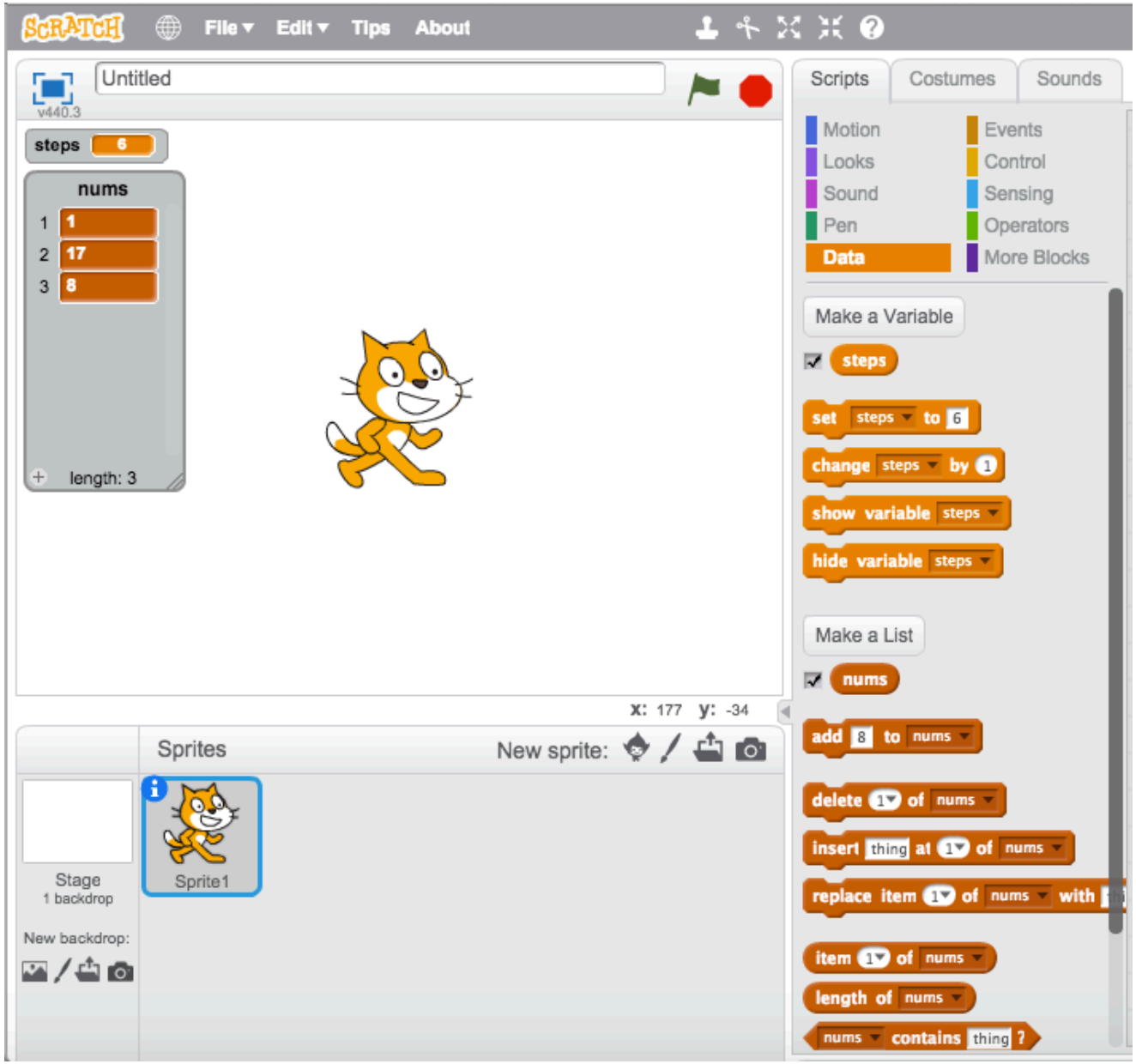
Talk Road Map

- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Stepping in PencilCode, early Scratch

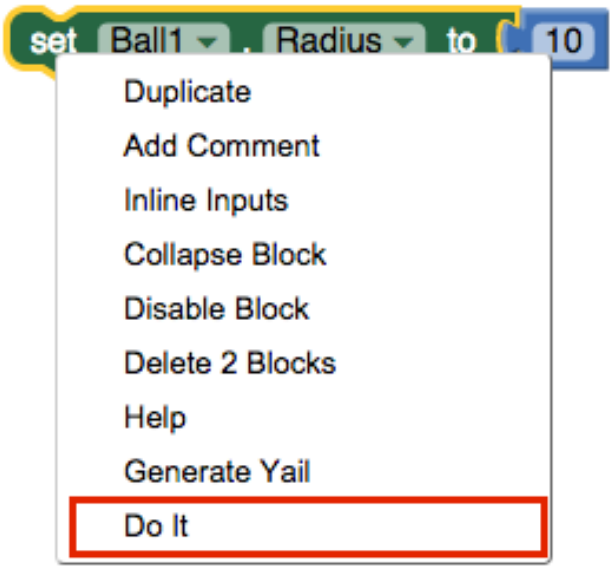


Variable Display in Scratch

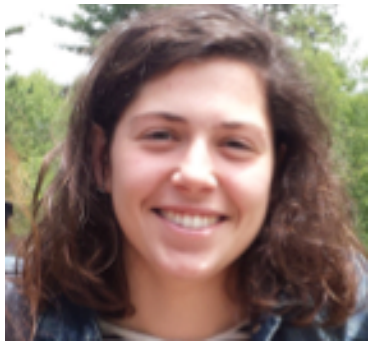


App Inventor: DoIt

Simple form of interactivity/liveness found in many blocks environments (as well as interpreter text-based languages).

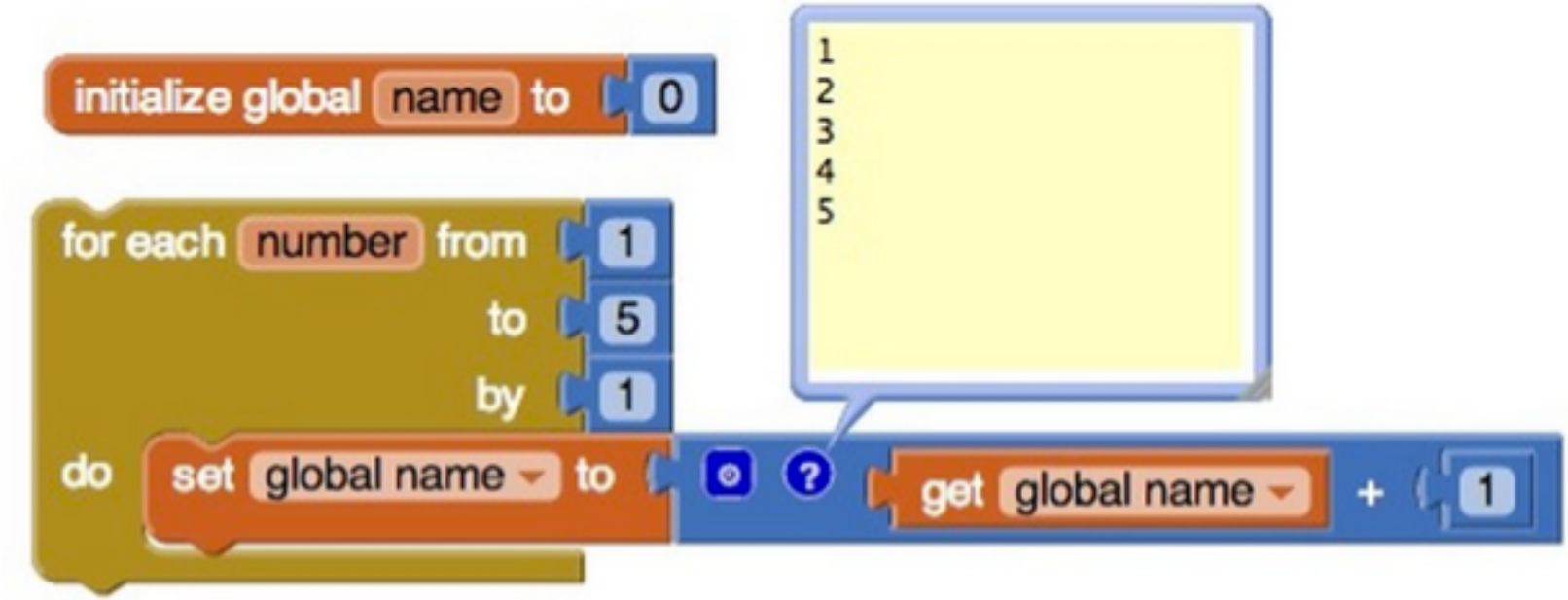


Better Debugging: Watch

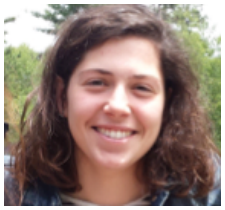


Johanna Okerlund ‘14 Wellesley

Emery Gerndt Otopalik ‘16 Wellesley



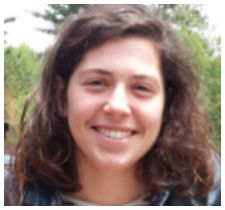
Better Error Handling

Currently, AI error window covers blocks and does not pinpoint block causing error:

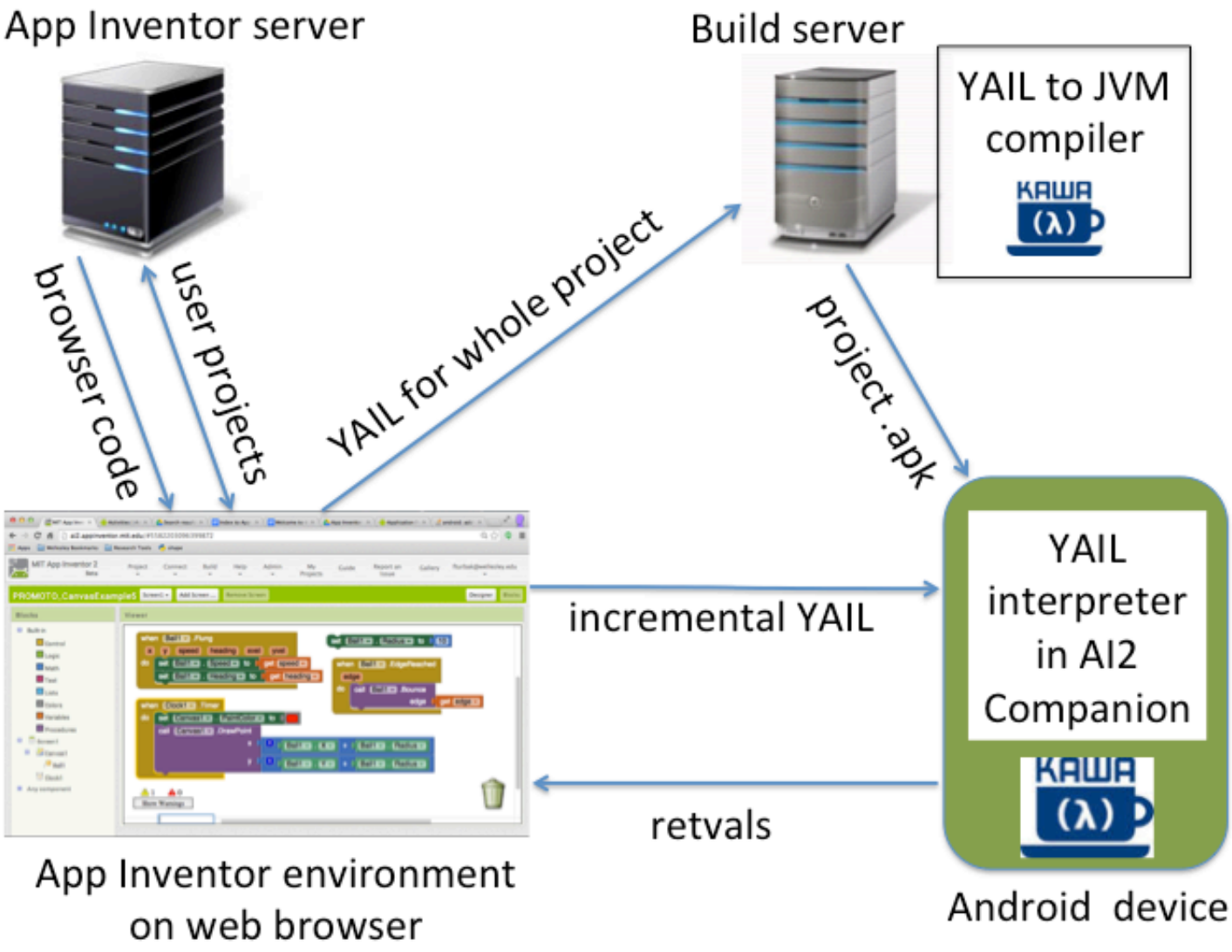
Soon, the error will appear on the block causing the error:

Better Error Handling

Error messages can appear on multiple blocks until the errors are fixed:

AI Live Development Architecture



YAIL Example

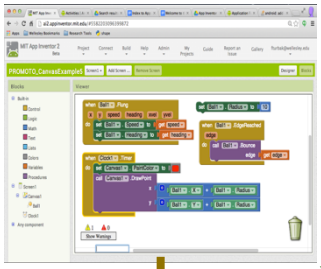
```
;; Screen1
(do-after-form-creation
  (set-and-coerce-property! 'Screen1 'Title
                            "Screen1" 'text))

;;; Canvas1
(add-component Screen1 Canvas Canvas1
  (set-and-coerce-property! 'Canvas1 'BackgroundColor
                            #xFF00FFFF 'number)
  (set-and-coerce-property! 'Canvas1 'Width 200  'number)
  (set-and-coerce-property! 'Canvas1 'Height 300 'number))
```

```
;;; Ball1
(add-component Canvas1 Ball Ball1
  (set-and-coerce-property! 'Ball1 'X 46 'number)
  (set-and-coerce-property! 'Ball1 'Y 27 'number))

(define-event Ball1 Flung  ($x $y $speed $heading
                            $xvel $yvel)
  (set-this-form)
  (set-and-coerce-property! 'Ball1 'Speed
                            (lexical-value $speed)
                            'number)
  (set-and-coerce-property! 'Ball1 'Heading
                            (lexical-value $heading)
                            'number))
```

Two-way WiFi communication via HTTP

App Inventor Browser

web server on
App Inventor Companion



YAIL1

any values?

Run YAIL1

OK

watchval1

YAIL2

[watchval1]

watchval2

Queue YAIL2

OK

screenchangeval

any values?

errorval

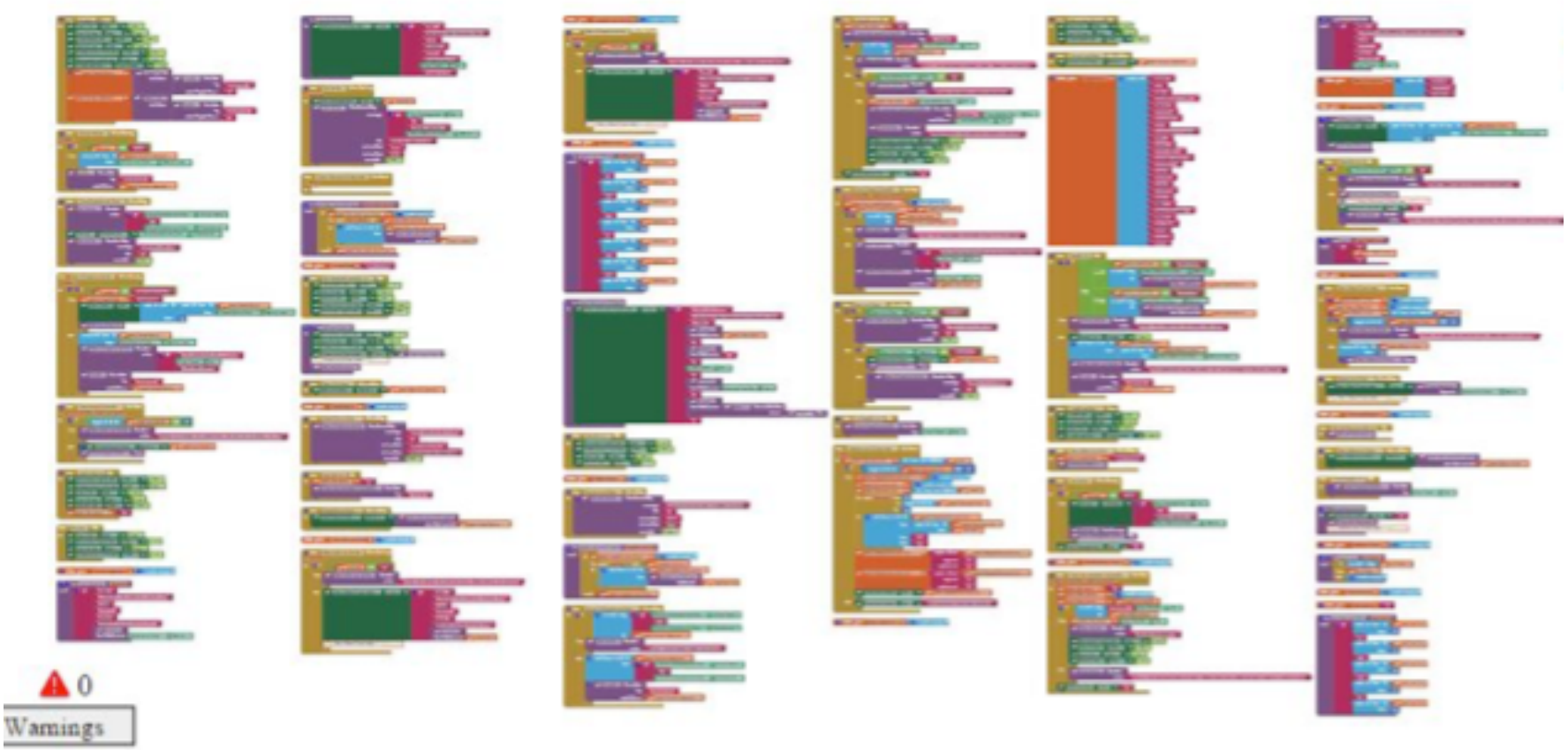
Run YAIL2

[watchval2, screenchangeval, errorval]

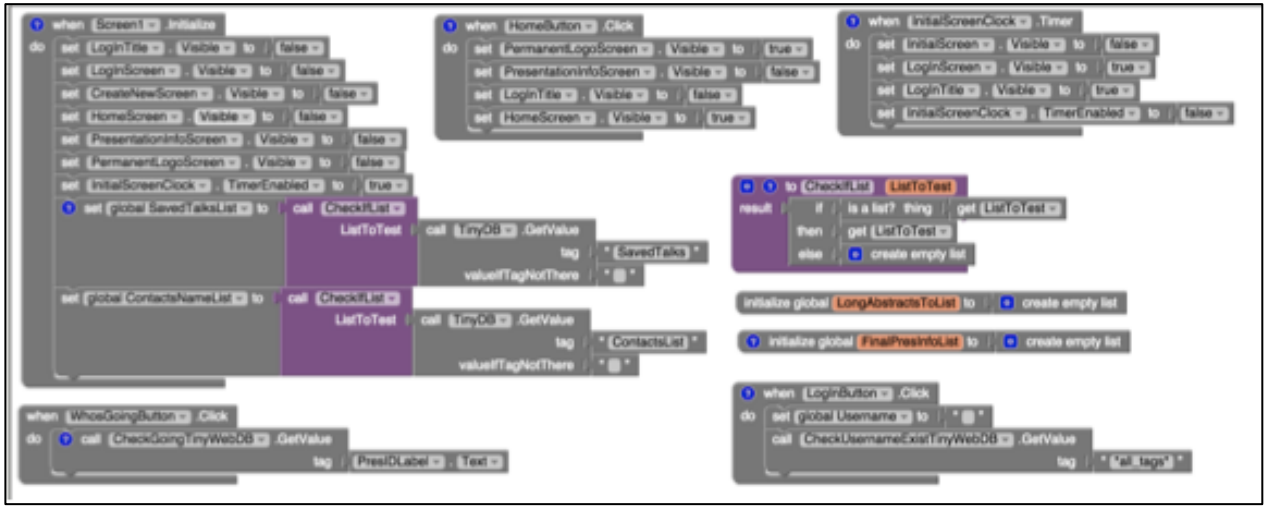
Talk Road Map

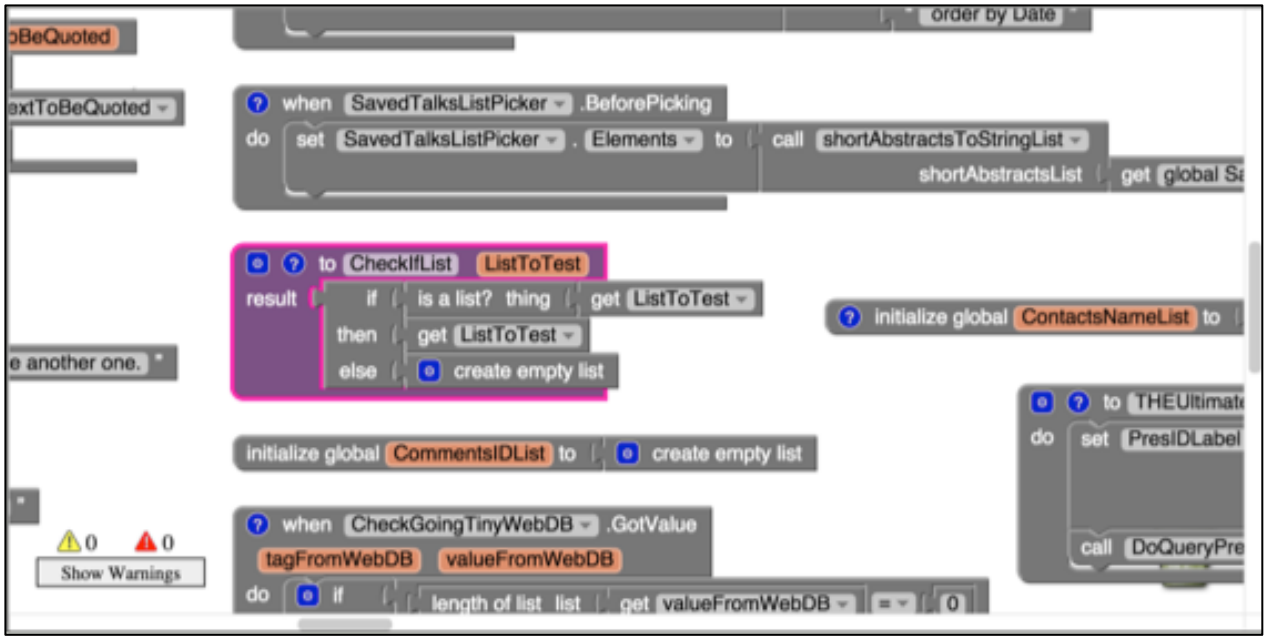
- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
  - Pragmatics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Usability: Big Programs are Hard to Understand



Usability: Searching 2D Blocks Workspaces





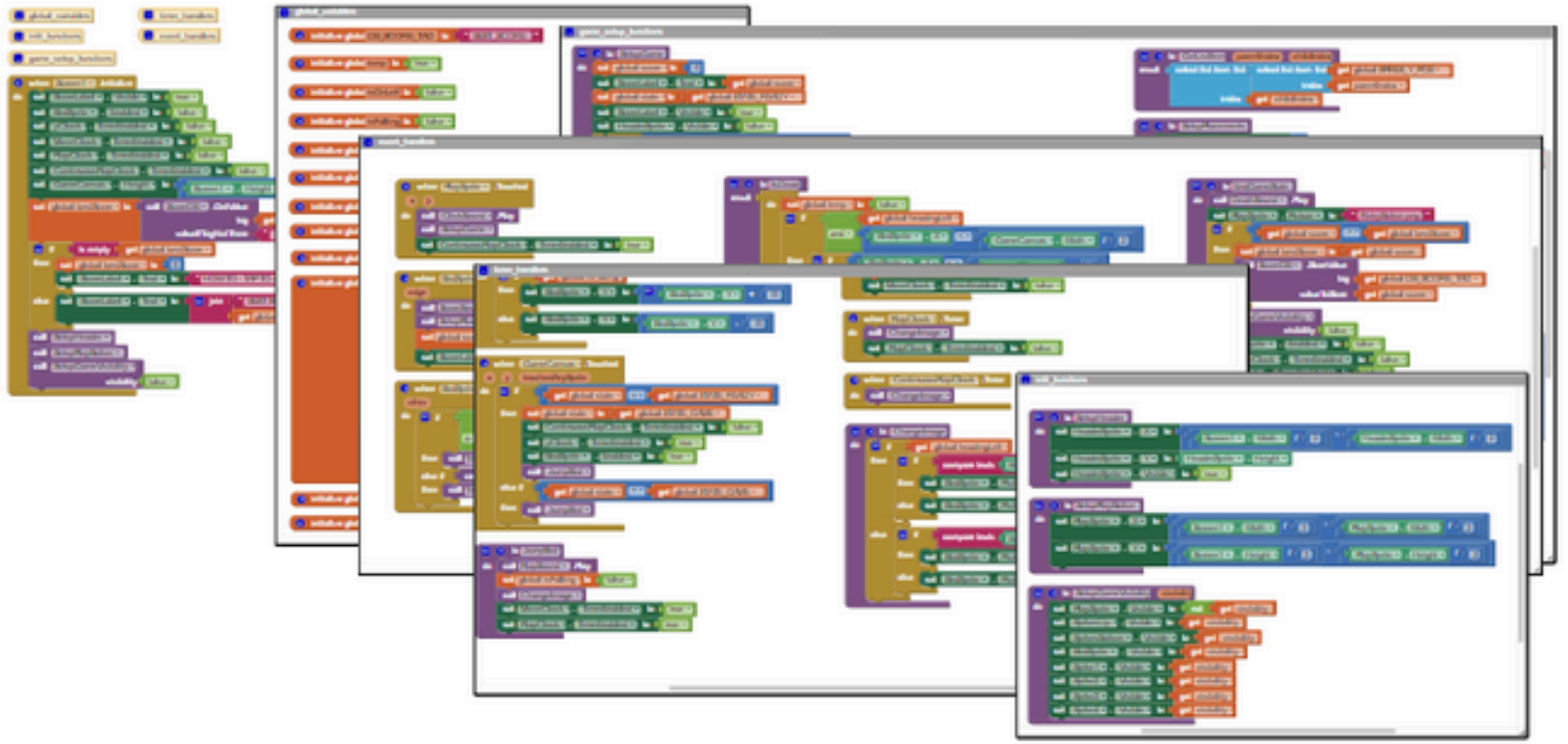
Cece Tsui '18
Wellesley

Usability: Organizing 2D Blocks Workspaces

Shirley X. Lu '15
Wellesley

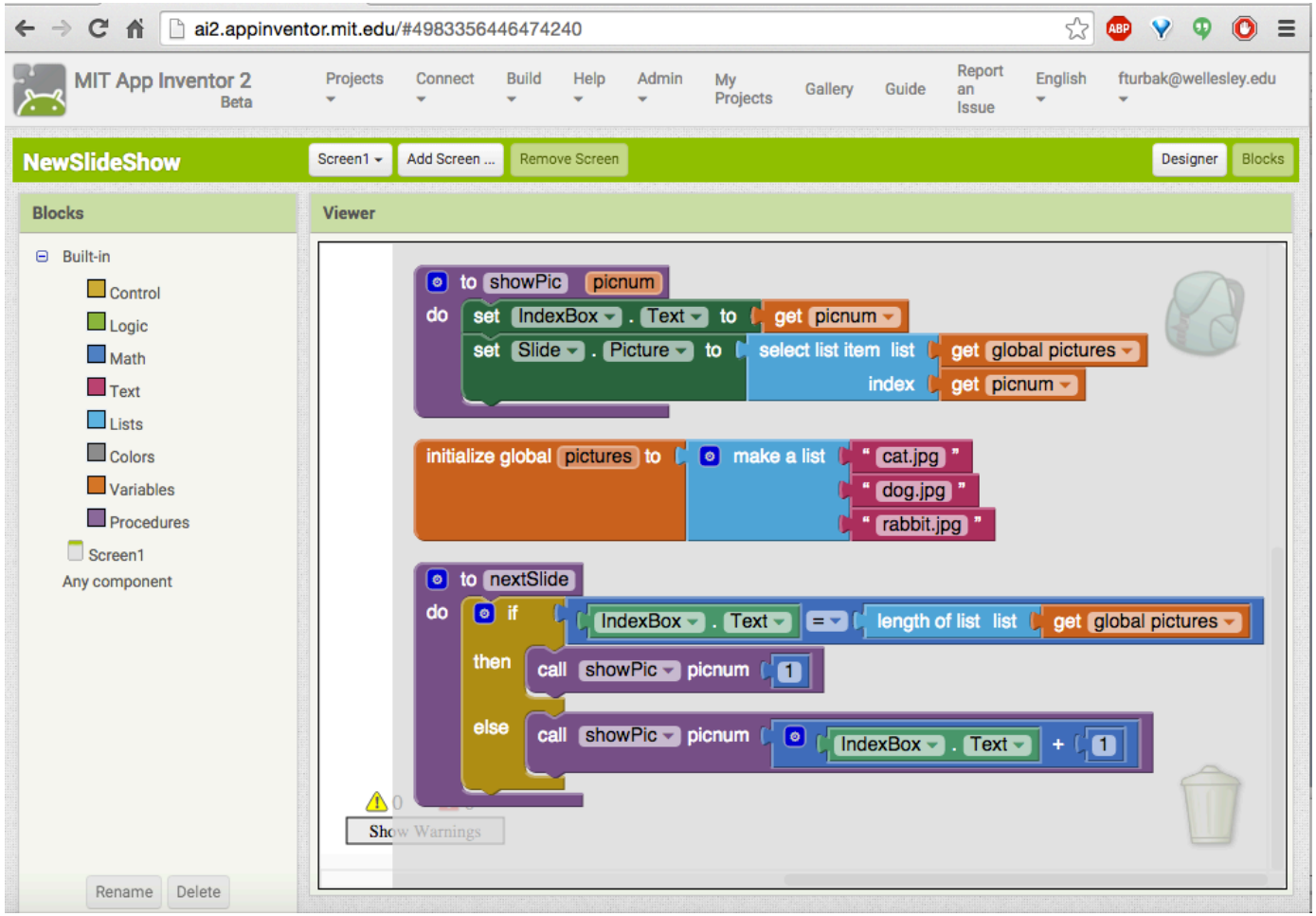
Devid Farinelli '16
U. of Bologna

Folders in App Inventor (under development)



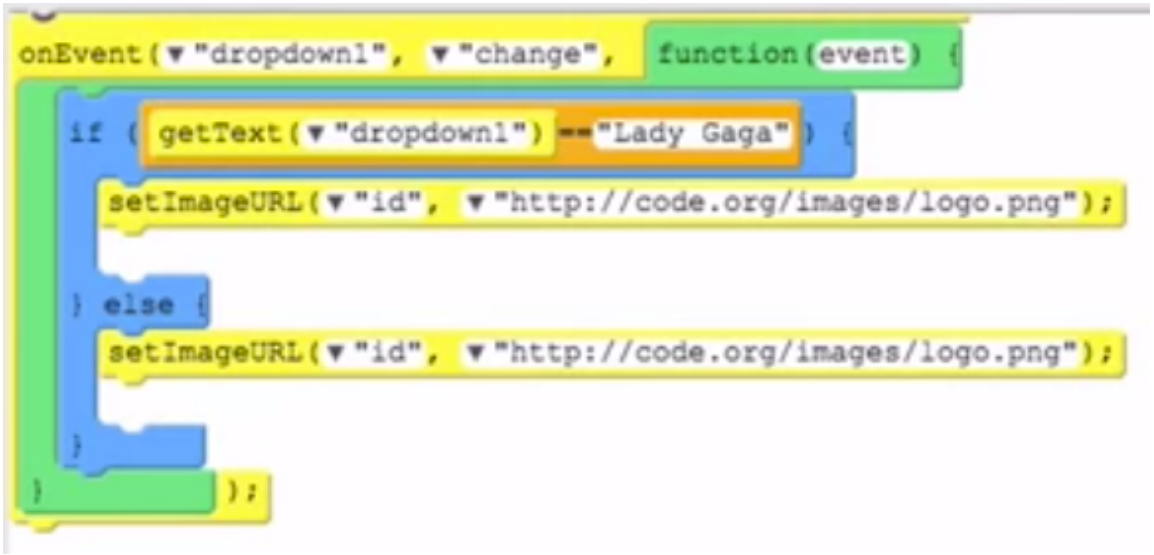
Usability: Reusing & Sharing Blocks Programs

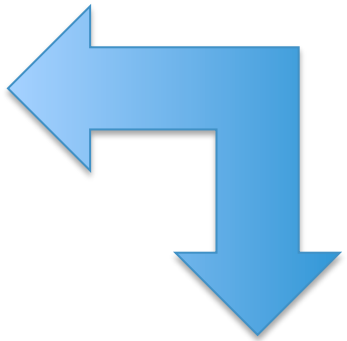
Backpack in Scratch and App Inventor



Usability: Droplet's Isomorphic Blocks/Text Conversion

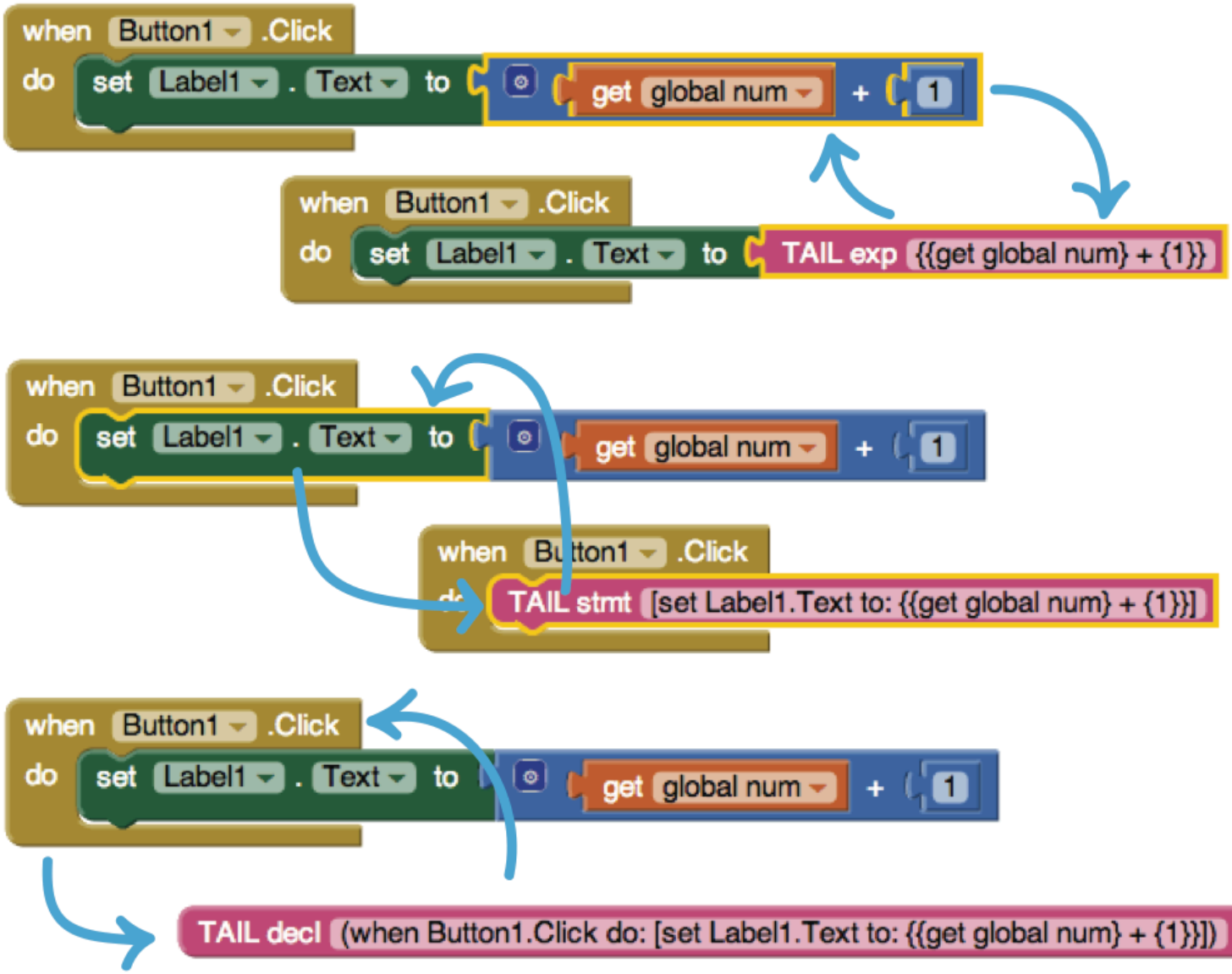
Used in PencilCode and Code.org's AppLab JavaScript curriculum





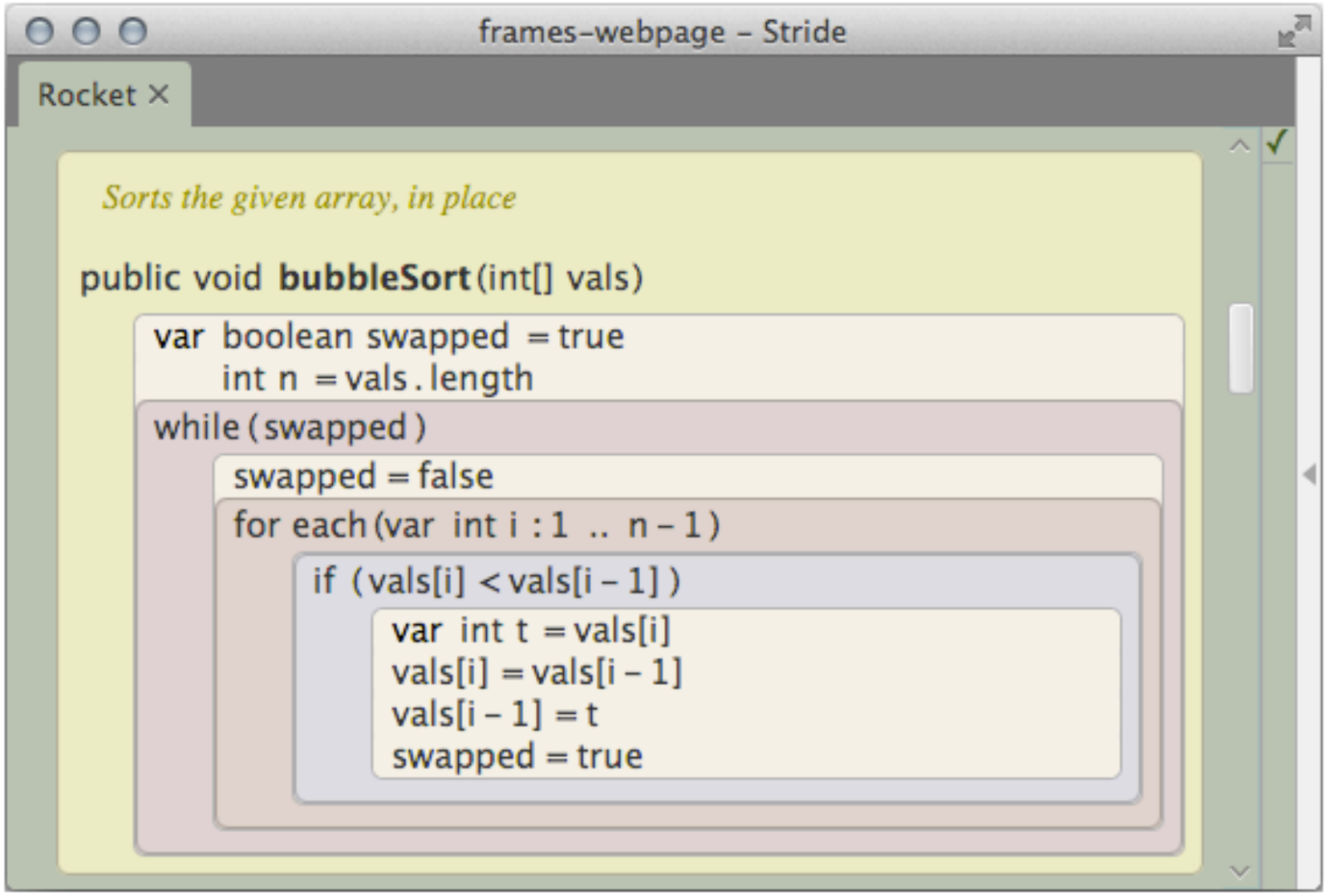
```
1  onEvent("dropdown1", "change", function(event) {
2    if (getText("dropdown1")=="Lady Gaga") {
3      setImageURL("id", "http://code.org/images/logo.png");
4
5    } else {
6      setImageURL("id", "http://code.org/images/logo.png");
7
8    }
9  });
10
```

Experimental Conversion Between Blocks and Text



Karishma Chadha '14 Wellesley

Usability: Greenfoot's Frame-based Editing

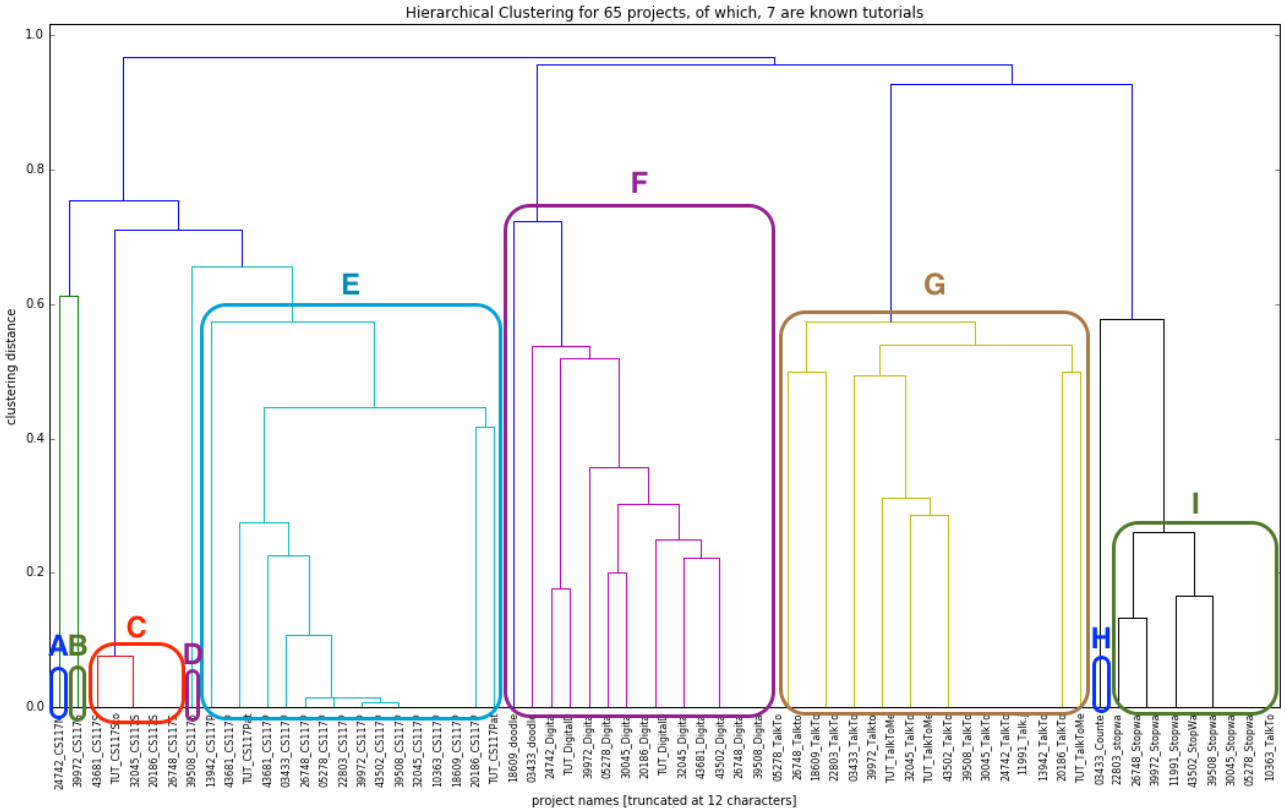


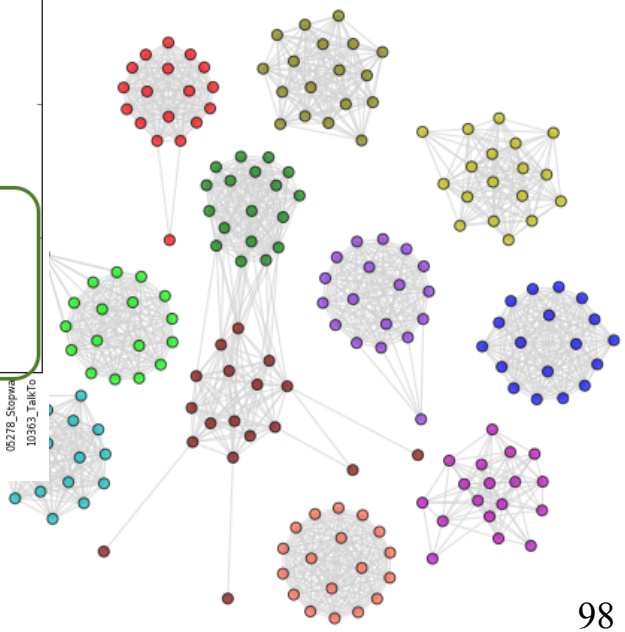
Analyzing App Inventor Programs

Eni Mustafaraj

Maja Svanberg ‘18

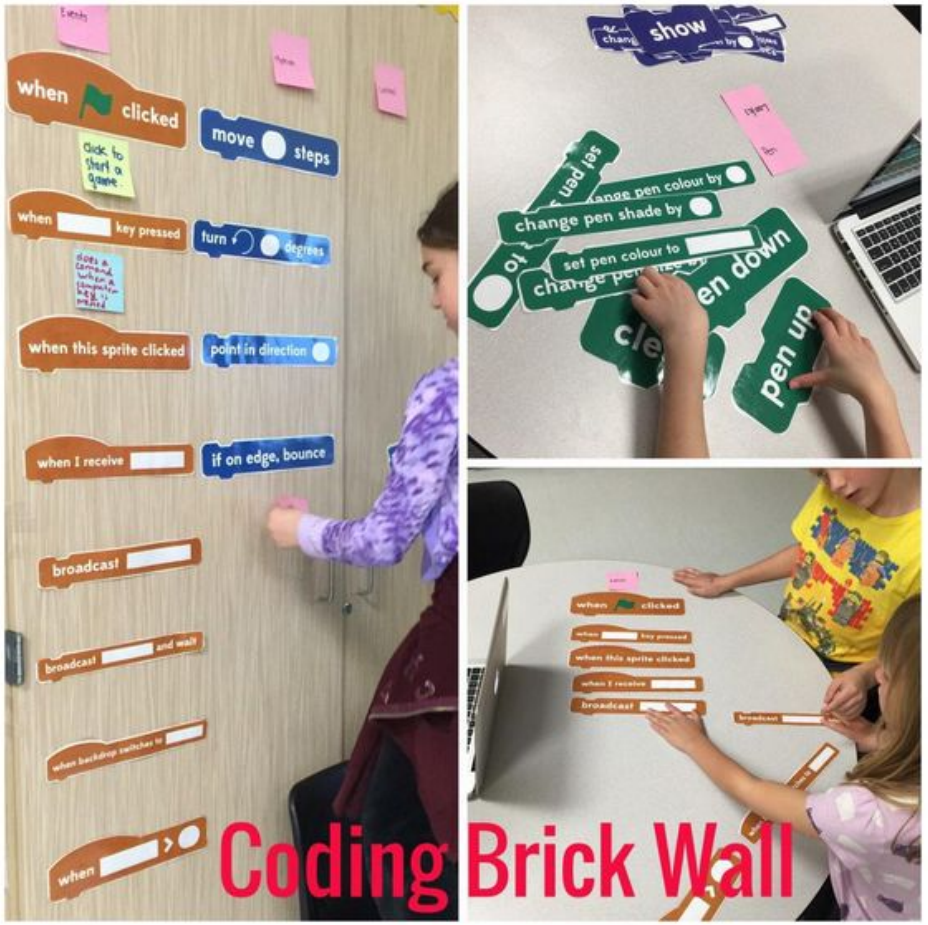
Shan Lu ‘20





New Project: Collaborative Blocks Programming

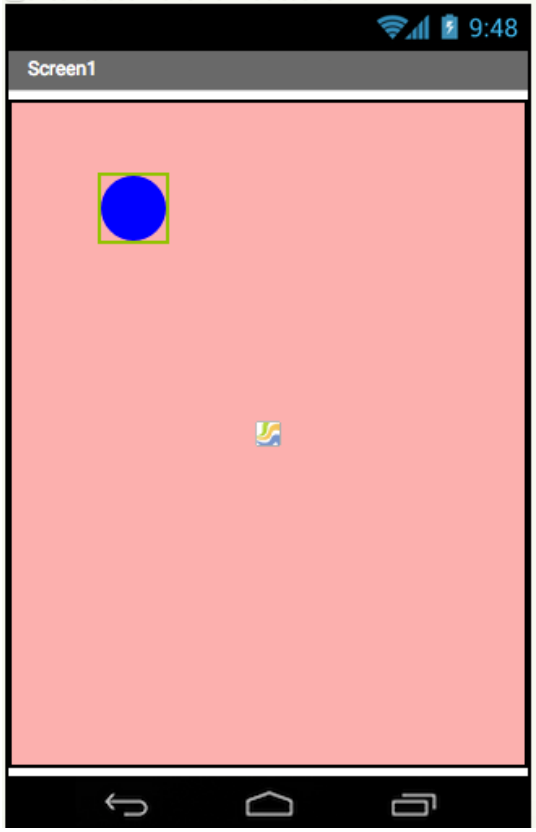
Summer Project:
Work with HCI Lab and MIT App Inventor group

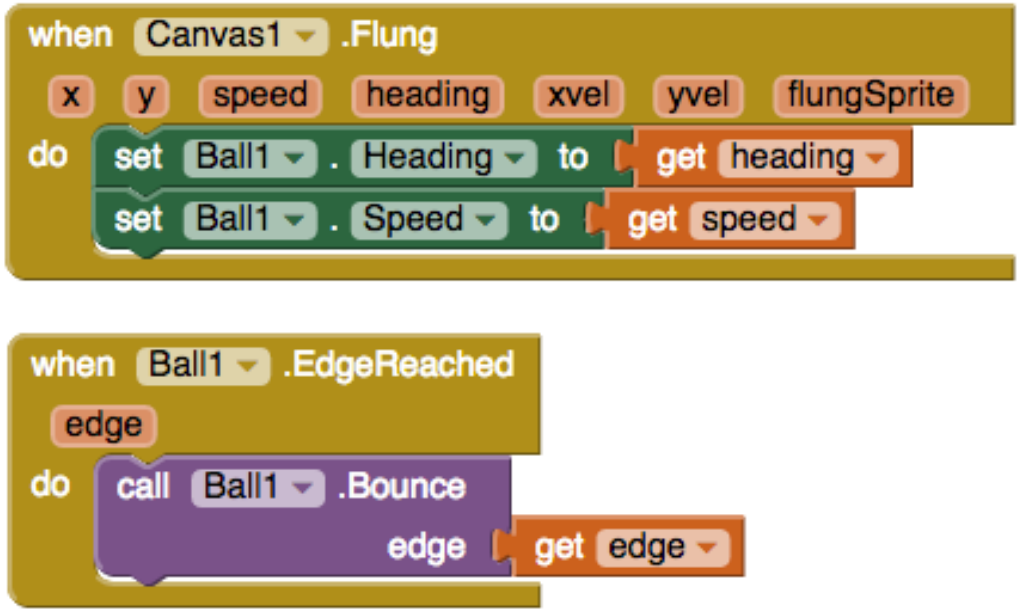


Talk Road Map

- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks: examples
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

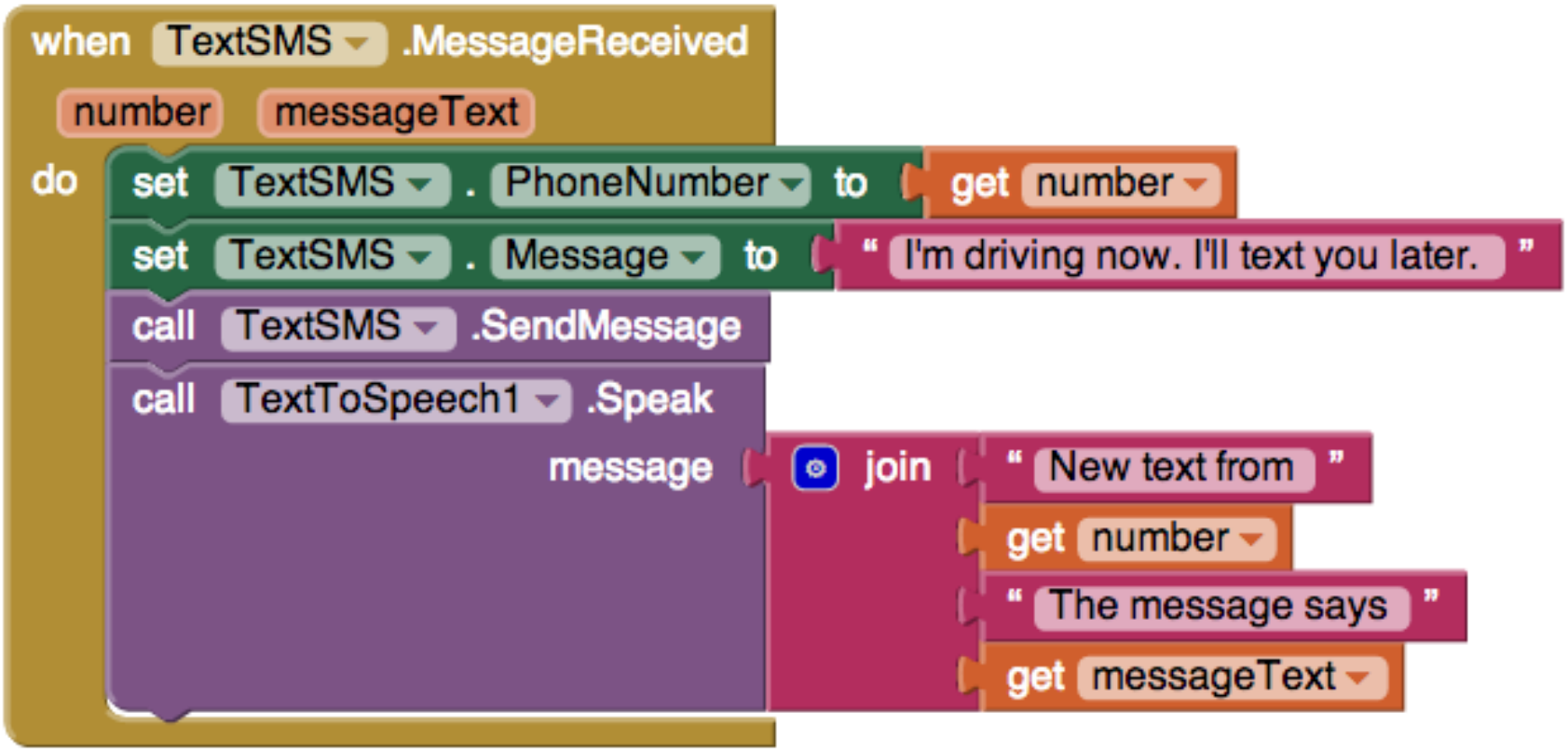
Thinking Outside the Blocks: Abstraction





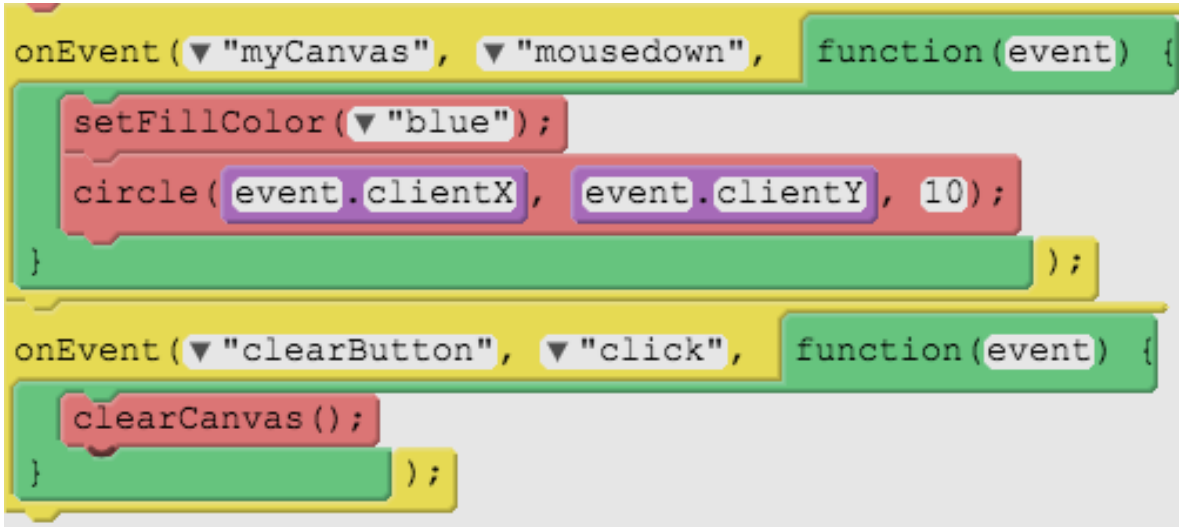
Thinking Outside the Blocks: Abstraction

What does this code do?

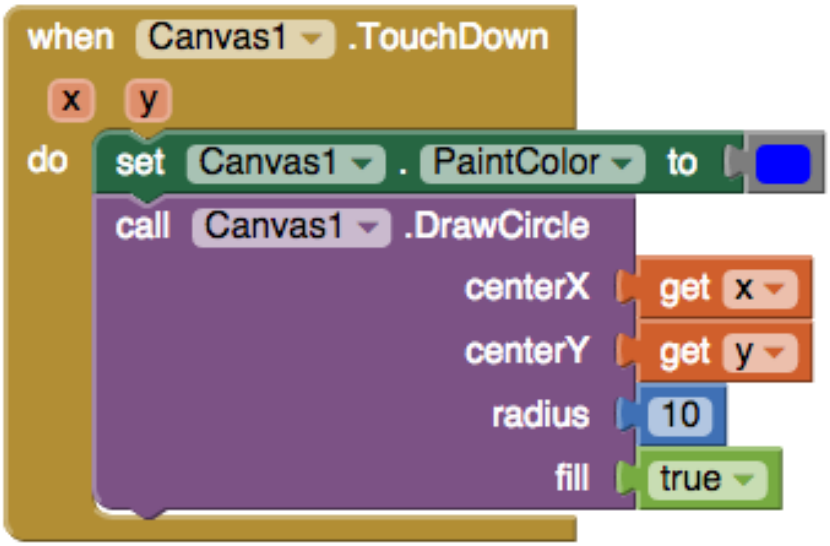


Thinking Outside the Blocks: Abstraction

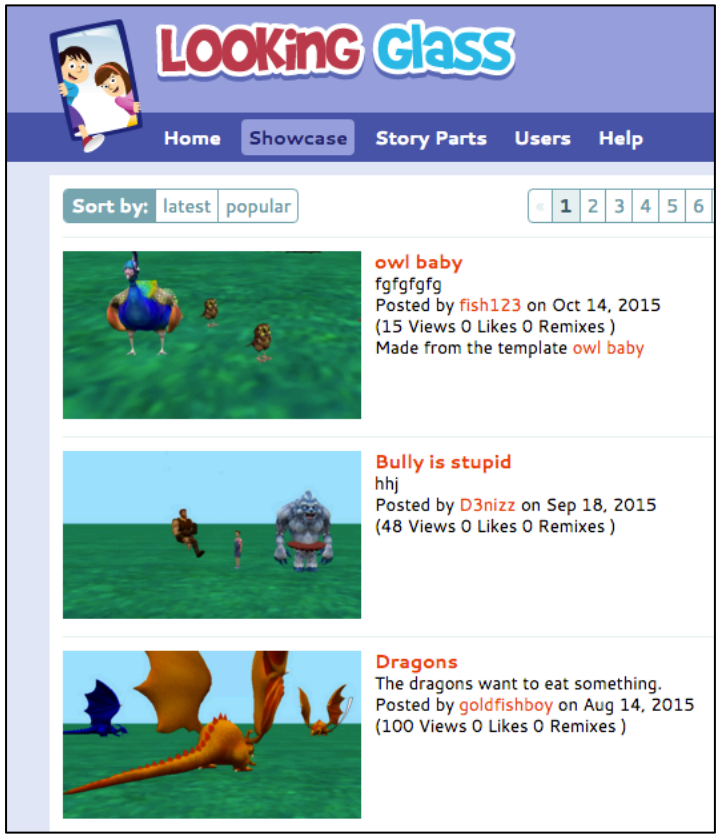
App Lab/
Droplet

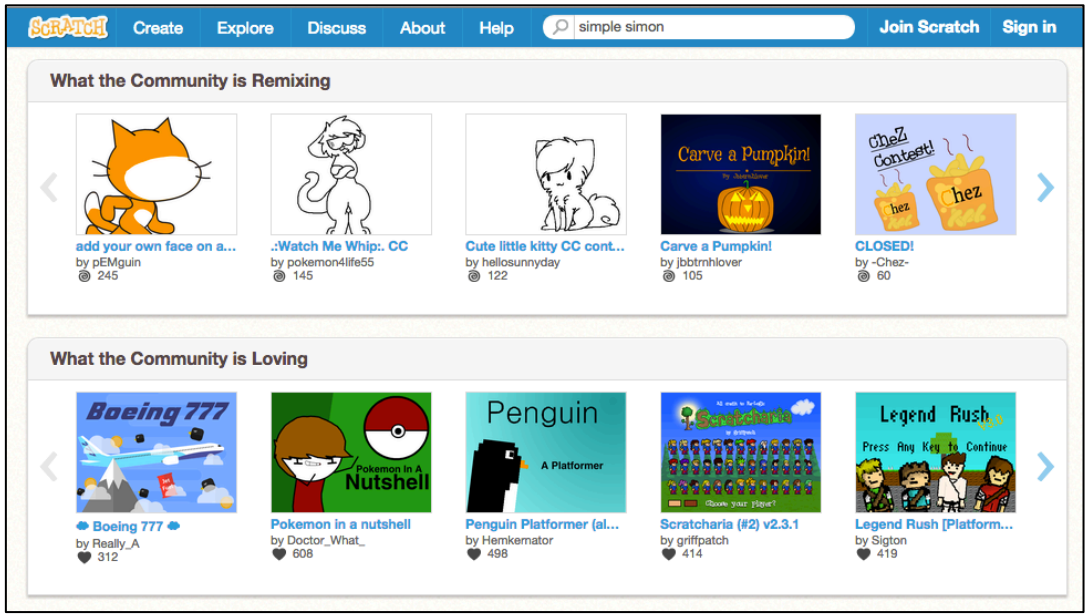


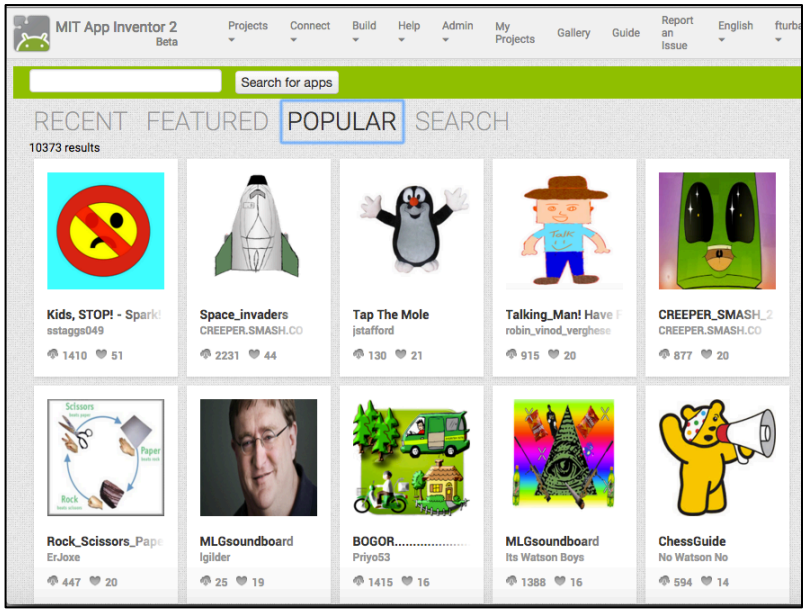
App
Inventor



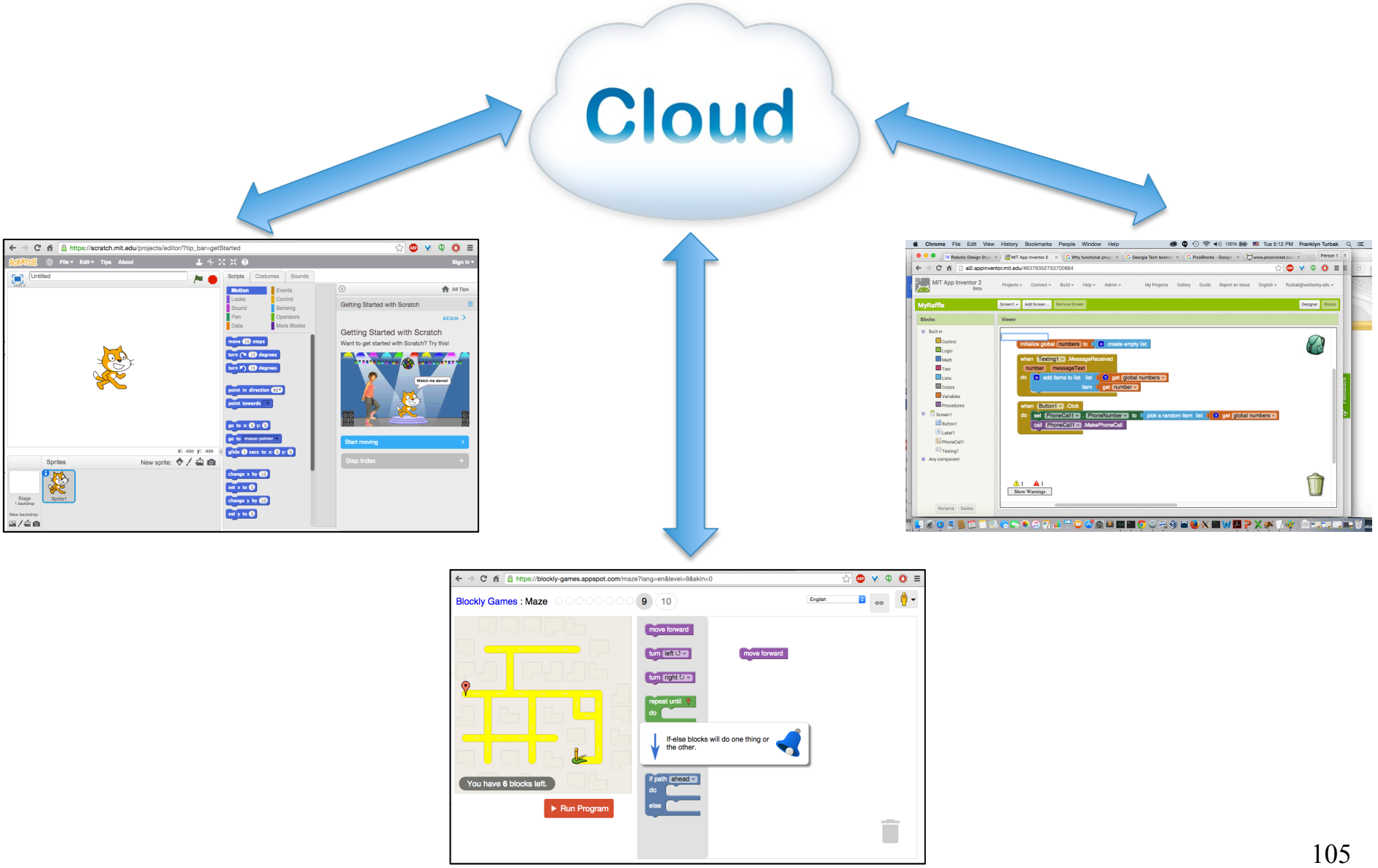
Thinking Outside the Blocks: Community







Thinking Outside the Blocks:
Browser-Based Environments & Cloud Program Storage



New Project: Collaborative Blocks Programming

Summer Project:
Work with HCI Lab and MIT App Inventor group

• Thinking outside the blocks

Talk Road Map

- Blocks demo: MIT App Inventor (AI)
- Democratizing programming with blocks: examples
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
- Challenges in blocks programming
  - Usability
  - Thinking outside the blocks
  - Perception: blocks programming not “real”

Negative Responses to Blocks Languages

I have never met a student who cut their teeth in any of these languages and did not come away profoundly damaged and unable to cope.

I mean this reads to me very similarly to teaching someone to be a carpenter by starting them off with plastic toy tools and telling them to go sculpt sand on the beach.

Not one thing they learn will bear any piece of resemblance to real work. All you're doing is teaching them misimpressions of what the job is, and tricking them out of having meaningful formative experiences.

http://blog.acthompson.net/2012/12/programming-with-blocks.html

These are not proper programming languages, anyone with half a brain knows that, but why deny those who can't or don't want to 'code' the opportunity of being creative with these tools and learning some logic skills along the way.

http://blog.acthompson.net/2012/12/programming-with-blocks.html

Working with actual code writing instead of a drag & drop interface prepares children better for the real world.

http://www.playcodemonkey.com/

Mark Sherman's Response

Mark Sherman
UMass Lowell

So they currently see this:

when it is really this:

Yes, it is colorful and newfangled, but it still gets jobs done. Not all of them, but a bunch of them.

Why do they see it this way? Because they grew up on this:

More Positive Feedback

I would like to express my utmost appreciation for your product. I'm teaching several pre-CS courses for gifted youth at Junior-high school level (7th-9th grades) as well as CS and software engineering at high school (10th – 12th grades) including Android development in Java. **It is really amazing that in AppInventor, 7th grade students (with about 50 hours prior experience in Scratch) can do in 6 hours what 12th grade students take about 200-300 hours to achieve in Java (and this is after studying CS and Android development for about 700 hours).** AppInventor goes way beyond the 80:20 principle (80% of the utility in 20% of the effort) – it is more like 60:5 (60% of the functionality, for less than 5% of the effort) which makes it much more fun, and opens up a lot of space for creativity.

Yossi Yaron, Israeli teacher

Some Research Questions

- 2D blocks workspaces:
  - What are good ways to search, navigate, and organize them?
  - Do they confer any advantages over linear text?
- How can debugging & visualization of dynamic execution for blocks environments be improved?
- What tools can improve collaborative development of blocks programs?
- How can we do programming on the devices themselves? (Existing examples: microApps, Pocket Code, Touch Develop.)
- Can any blocks affordances improve productivity in mainstream languages?
- What does big data analysis say about learnability/usability of blocks vs. text notations and transitioning from blocks to mainstream languages?
- What role do the following "nonblocks" aspects play in learnability and usability of blocks languages: web-based environments, cloud-based storage, high-level abstractions, sharing/remixing communities, liveness.

App Inventor Development Team

Hal Abelson MIT

Andrew McKinney MIT

Jeff Schiller MIT

Paul Medlock-Walton MIT

Jose Dominguez MIT

Mark Friedman Google

Sharon Perl Google

Liz Looney Google

Neil Fraser Google (Blockly)

Franklyn Turbak Wellesley College

Computational Thinking Through Mobile Computing
NSF Grant Team


Franklyn Turbak Eni Mustafaraj
Wellesley College

Ralph Morelli
Trinity College

Dave Wolber
U. of San Francisco

Larry Baldwin
BIRC



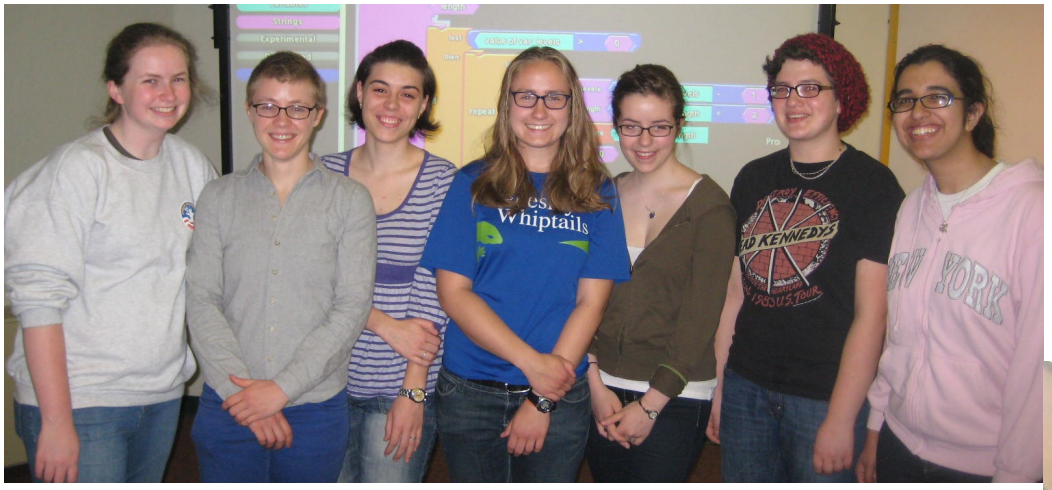
Hal Abelson Shay Pokress Josh Sheldon
MIT

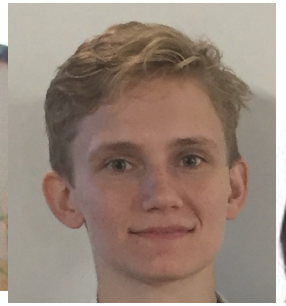


Fred Martin Mark Sherman Karen Roehr
University of Massachusetts Lowell

Acknowledgment: This work was supported by the National Science Foundation under Grants 1225680, 1225719, 1225745, 1225976, and 1226216.

Wellesley TinkerBlocks Students





Questions?

TINKERBLOCKS

Search this site

Home | App Inventor | Gallery | People | PictureBlocks | Publications | Talks/Posters | TurtleBlocks | TypeBlocks

Home

TinkerBlocks is a project for improving the expressiveness of blocks programming languages and the usability of blocks programming environments. So far we've created two blocks languages (TurtleBlocks and PictureBlocks) for creating tangible artifacts on laser cutters and vinyl cutters, and are working on a blocks language (TypeBlocks) in which the shape of a block connector encodes its type in a functional language. We're also working with members of the MIT App Inventor development team to Improve App Inventor.

The TinkerBlocks project is anchored at Wellesley College and is led by Lyn Turbak. Meet our team members!

Here are some images from our work: