Democratizing Programming with Blocks Languages

Franklyn Turbak
Wellesley College Computer Science Dept.

Distributed Multimedia Systems (DMS 2015)
Vancouver, August 31, 2015
Wellesley & MIT
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
Simple App Inventor Example
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
Papert on Computers & Constructionism

“In many schools today, the phrase “computer-aided instruction” means making the computer teach the child. One might say the computer is being used to program the child. In my vision, the child programs the computer, and in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intense contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building.” Mindstorms: Children, Computers, and Powerful Ideas (bolding mine)

"The word constructionism is a mnemonic for two aspects of the theory of science education underlying this project. From constructivist theories of psychology we take a view of learning as a reconstruction rather than as a transmission of knowledge. Then we extend the idea of manipulative materials to the idea that 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)
“But in this new world, you don’t have to go bankrupt if you fail because you can fail small. 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*

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.
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:
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.

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.

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
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.


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

run once

set corner type
SHARP

set thickness
15

times
5

do
repeat
forward
distance
200

left
degs
144

cardstock

acrylic
drawing boundary
TurtleBlocks Artifacts
PictureBlocks: Sketching & Engraving

user sketch  →  PictureBlocks program  →  resulting picture

print from engraving  ←  wood 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

Age Distribution of New Scratchers

Distribution of respondents by age

What is your age

Count

Number of users

Under 18  18 to 24  25 to 34  35 to 44  45 to 54  55 to 64  65 to 74  75 or older  None given
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)
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: examples
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
- Challenges in blocks programming
  - Usability
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
AI Syntax: Expressions
AI Syntax: Statements

- set Button1 . BackgroundColor to
- set Label1 . Text to
- set Canvas1 . PaintColor to
- if then
- while test do
- for each number from 1 to 5 by 1 do
- for each item in list do
- call Camera1 . TakePicture
- call TextToSpeech1 . Speak message
- call Canvas1 . DrawCircle x y r
- add items to list list item
- insert list item list index item
AI Syntax: Top Level Declarations

- Initialize global variable `name` to `initialize global (name to`
- To procedure
- Do
- To procedure2
- Result
- When `Camera1` .AfterPicture
  - Image
  - Do
- When `Canvas1` .TouchDown
  - X y
  - Do
- When `Canvas1` .Dragged
  - startX startY prevX prevY currentX currentY draggedSprite
  - Do
AI Syntax: Local Variable Declarations

```
initialize local name to in
initialize local neg_b to
initialize local disc to
initialize local two_a to
make a list
```
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: examples
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
- Challenges in blocks programming
  - Usability
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
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

```python
>>> def square (x):
...   x * x
...  
>>> square(5)
```
Connector Shapes in PictureBlocks

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

- **number**
  - 1
  - +
  - sqrt
  - atan
  - x
  - y

- **boolean**
  - true
  - not
  - and

- **string**
  - abc
  - num to string
  - join

- **color**
  - Red
  - wedge color

- **picture**
  - Load picname
  - clockwise pic
  - pic
  - fourPics
Polymorphism in PictureBlocks

polymorphic plug

polymorphic sockets

choose test then abs

choose test then 1

choose test then abc
pushRight: Complete Declaration and Call
Type Blocks

Marie Vasek ‘12
Wellesley

listof constructor
tuple constructor
function constructor

listof int  listof (listof string)  int * string  bool -> string
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: examples
- Lowering barriers with blocks
  - Syntax
  - Static semantics
  - Dynamic semantics
- Challenges in blocks programming
  - Usability
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
App Inventor: Dolt

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

Emery Gerndt Otopalk is currently working on a trace feature for watching all blocks after a breakpoint
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:
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
Usability: Current Work in AI

- Folders
- Searching for blocks on workspace
- Zooming
Usability: Droplet’s Isomorphic Blocks/Text Conversion

Used in PencilCode and Code.org’s AppLab JavaScript curriculum
AI: Conversion Between Blocks and Text

Karishma Chadha ‘14 Wellesley
Usability: Greenfoot’s Frame-based Editing

```
public void bubbleSort(int[] vals)
{
    var boolean swapped = true
    int n = vals.length
    while (swapped)
    {
        swapped = false
        for each (var int i : 1 .. n - 1)
        {
            if (vals[i] < vals[i - 1])
            {
                var int t = vals[i]
                vals[i] = vals[i - 1]
                vals[i - 1] = t
                swapped = true
            }
        }
    }
}
```

Sorts the given array, in place
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
Learnability in Blocks vs. Text

- **Lewis: Logo vs. Scratch study (SIGCSE 2010)**
  - Few significant differences between Logo-first and Scratch-first
  - Scratch-first did better on conditionals
  - Logo-first had more confidence as programmers.

- **Weintrop and Wilensky: Snap! vs Java (IDC 2015)**
  - Blocks easier to read and compose than text
  - Blocks perceived as less powerful, more verbose, inauthentic

- **Problem: Nonisomorphic languages**
  - Weintrop and Wilensky Commutative Assessment on blocks vs. text in isomorphic languages (ICER 2015) is promising approach
  - Matsuzaka taught Java with blocks environment isomorphic to text (SIGCSE 2015). Students perceived text as more “real”.
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
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

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

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:
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_
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
  - Learnability in blocks vs. text
  - Perception: blocks programming not “real”, maybe harmful
- Research questions
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  Trinity 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 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: