The Design of Naming Features in App Inventor 2

Lyn Turbak  
Wellesley College

Dave Wolber  
U. of San Francisco

Paul Medlock-Walton  
MIT
Intersection of Three Areas

- Blocks Programming Languages
- Naming Systems
- Cognitive Dimensions of Notations
Blocks Programming Languages


74% used a blocks language: Blockly, Scratch, App Inventor 2, Tynker, Hopscotch.
Big Ideas of Naming

- A **declaration** introduces a name.

- A **reference** uses a name (in programming, have **getters** and **setters**).

- The **scope** of a declaration is the area in which the name can be referenced.

- A nested declaration of the same name **shadows/introduces a hole in the scope** of the outer name.

- **Name locality**: names in non-intersecting scopes can be chosen independently.

- A declaration and all its references can be **consistently renamed** by a substitution that avoids variable capture.

- Names can be organized into non-interacting **namespaces**.
Cognitive Dimensions of Notations

• A way to evaluate visual programming languages (Green, 1989)

• Three are particularly relevant for our work:
  1. Error-proneness (main error = unbound variable)
  2. Viscosity
  3. Consistency
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo
App Inventor Classic (AI1) Demo

```python
def radius as number 20

when Canvas1.TouchDown
    name x
    y
    call Canvas1.DrawCircle x y value x value y
    global radius

do
call Canvas1.DrawCircle x y value x value y

when Canvas1.TouchUp
    name x1
    y1

do
call drawDots x2 value x1 value y1

do
    def numDots as number 0
    arg name x2
    arg name y2
    set:global numDots to number 0

for range variable name i
    start number 1
    end value y2
    step number 2
    x global radius

do
call Canvas1.DrawCircle x2 value x2 value y2
    global radius

set:global numDots to number 1 + global numDots

do
call Label1.Text to global numDots
```
App Inventor Classic (AI1) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
App Inventor 2 (AI2) Demo
Unlike AI1 & other blocks languages, AI2 flags unbound variables (with a red error triangle). This makes variable errors more obvious and reduces viscosity when editing programs:
Reducing Viscosity with Drop-Downs

Unlike AI1 & other blocks languages, AI2 flags unbound variables (with a red error triangle). This makes variable errors more obvious and reduces viscosity when editing programs:
Summary

- AI2 naming features respect the big ideas of naming.
- AI2 reduces error-proneness via:
  - Drop-down menus of in-scope names for references.
  - Flagging unbound variables
- AI2 reduces viscosity via:
  - In-place edits with drop-down menus.
  - Remembering unbound variable names.
- AI2 increases consistency by:
  - Representing all variable declarations using the same nonblock notation.
  - Eliminating extensible sockets for procedure declarations.
- Unlike AI2, AI1 supports local variable declarations.
This work was supported by Wellesley College Faculty Grants, by sabbatical funding from Wellesley College and the University of San Francisco, and by the National Science Foundation under grants DUE-1226216 and DUE-1225745.