

The Design of Naming Features in App Inventor 2

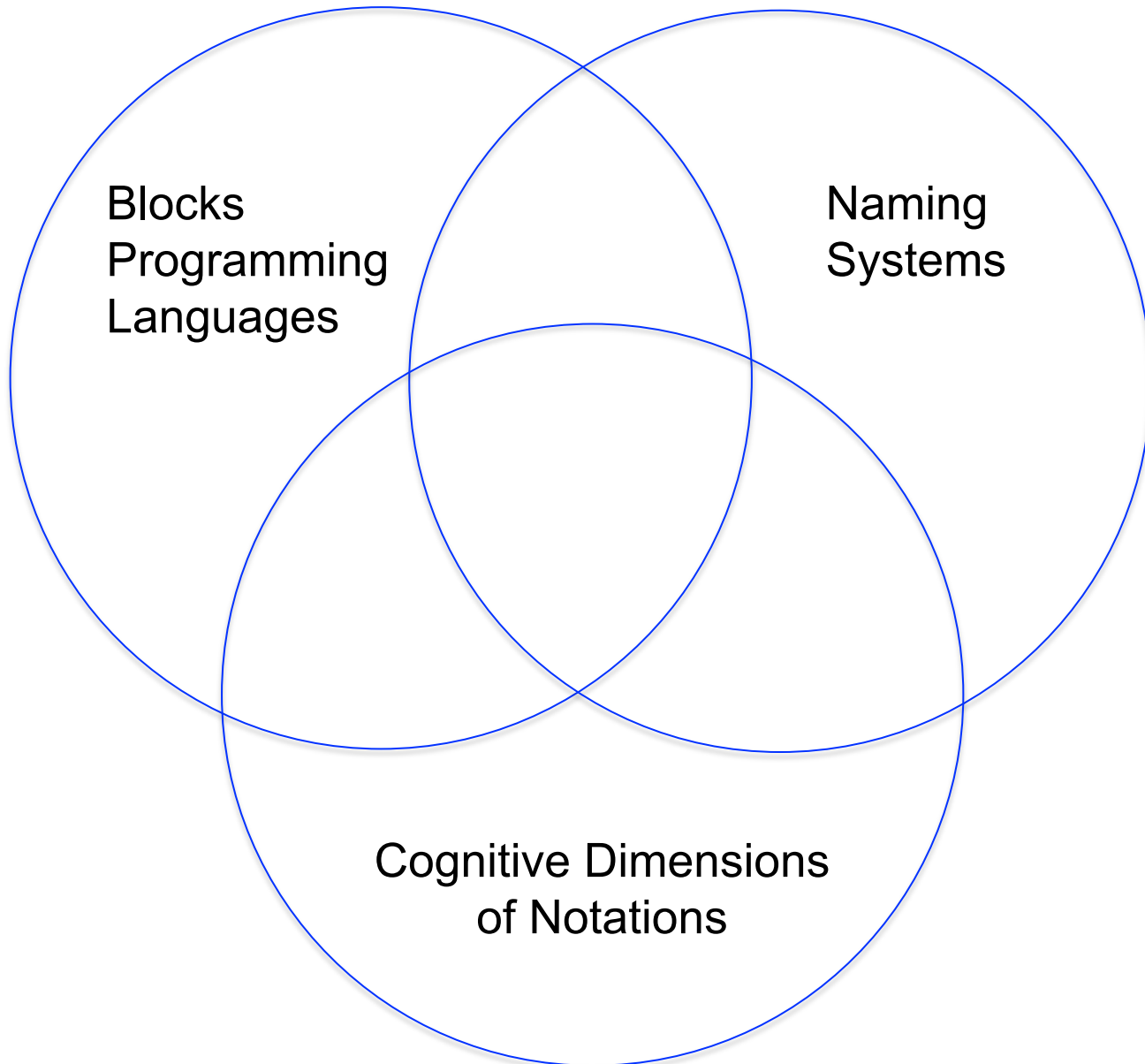


Lyn Turbak
Wellesley College

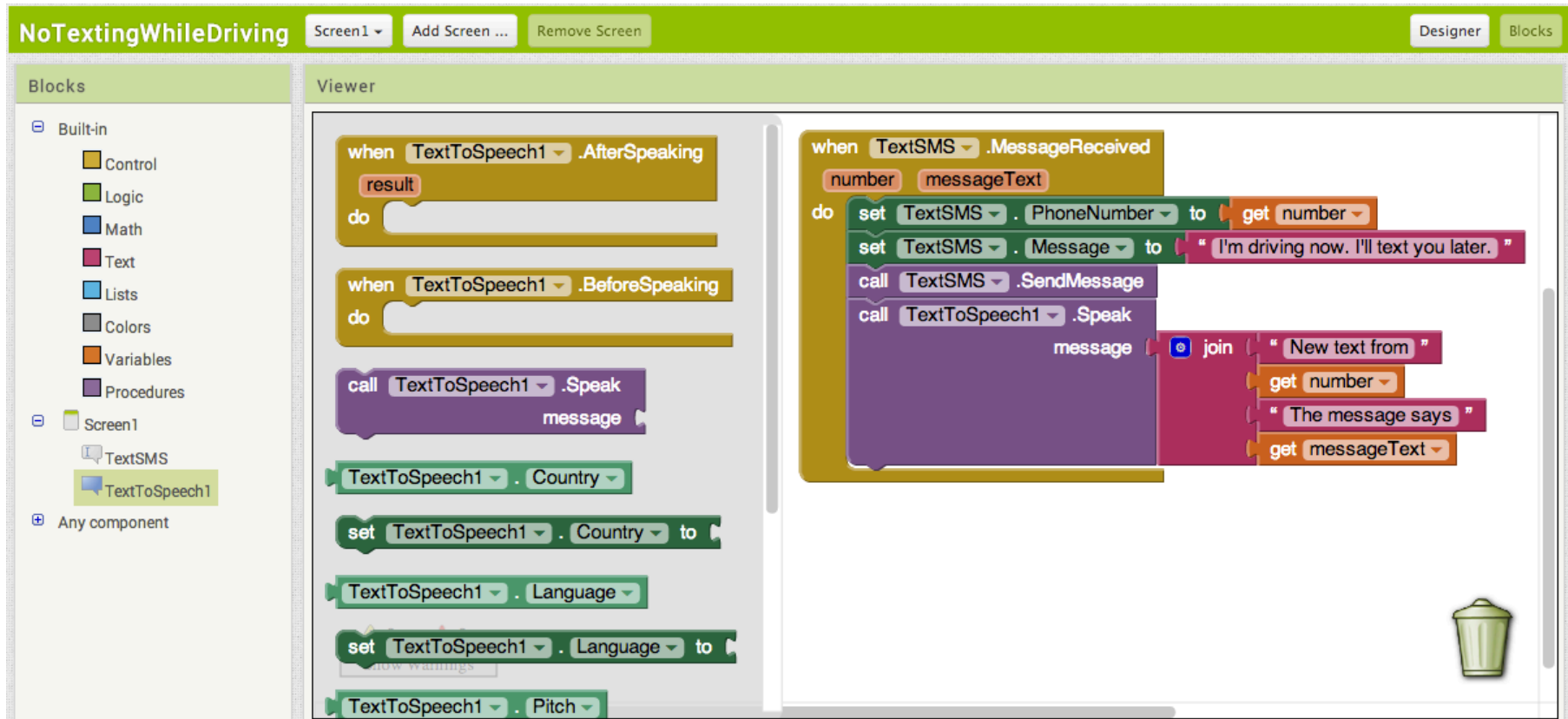
Dave Wolber
U. of San Francisco

Paul Medlock-Walton
MIT

Intersection of Three Areas



Blocks Programming Languages



Code.org's *Hour of Code* (launched Dec. 2013): > 26 million participants.

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**.

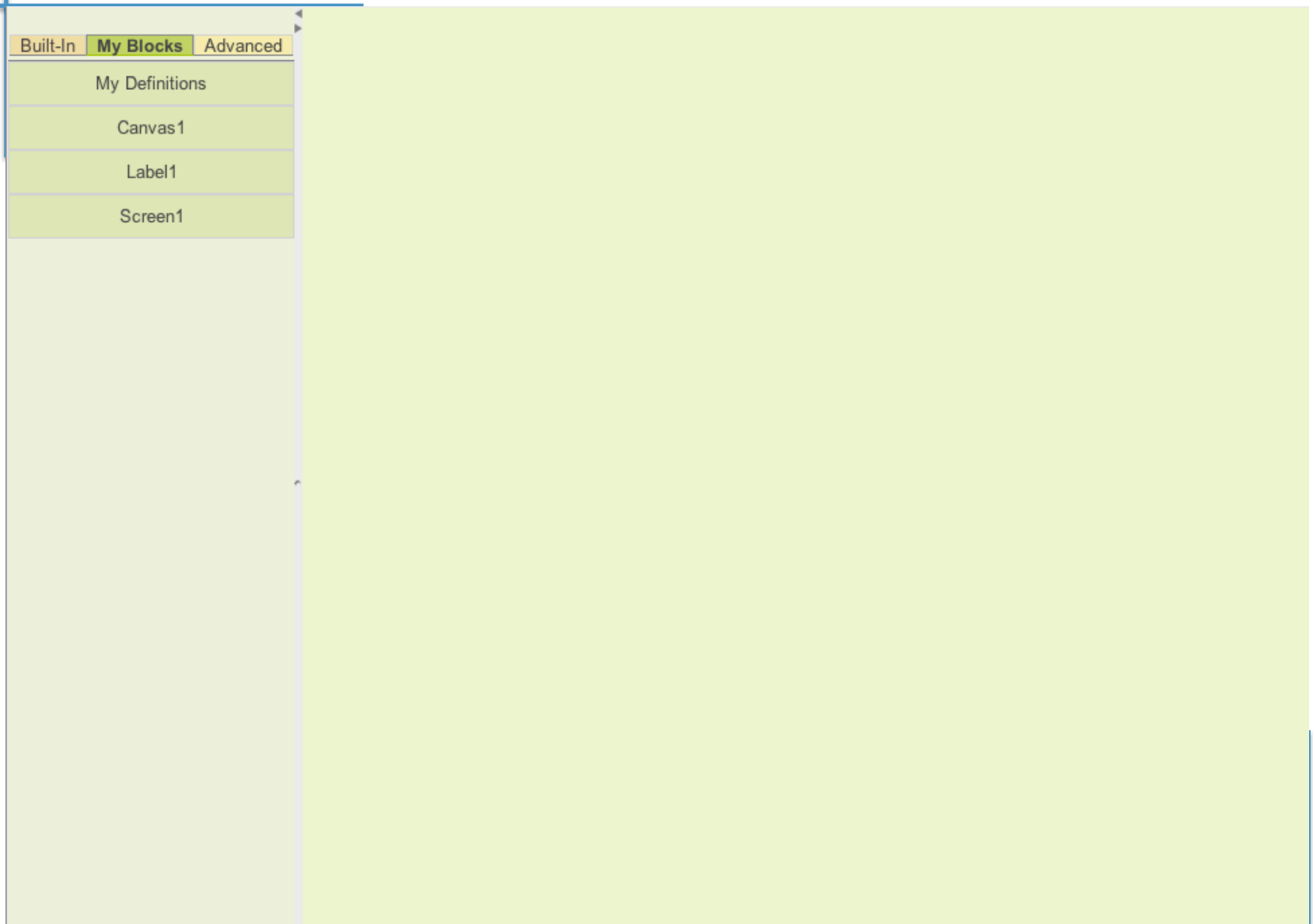
```
public class Example {  
  
    public static int test (int x) {  
        int sum = 0;  
        for (int i = 1; i < x; i++) {  
            int x = square(i);  
            sum = sum + x;  
        }  
        return sum;  
    }  
  
    public static int square (int n) {  
        return n * n;  
    }  
}
```

$$\int_a^b x \cdot \left(\int_c^x \square \, dx \right) dx \quad \prod_{x=1}^n \left(\sum_{x=1}^x \square \right)$$
$$\forall x. ((f(x) = g(x)) \wedge \exists x. \square)$$

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

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Blocks' panel is active, showing a list of components: Canvas1, Label1, and Screen1. The main workspace on the right contains four event-driven blocks for Canvas1:

- when Canvas1.Dragged**: This block has input fields for startX, startY, prevX, prevY, currentX, currentY, and draggedSprite. It includes a 'do' block for custom logic.
- when Canvas1.Flung**: This block has input fields for x, y, speed, heading, xvel, yvel, and flungSprite. It includes a 'do' block for custom logic.
- when Canvas1.TouchDown**: This block has input fields for x and y. It includes a 'do' block for custom logic.
- when Canvas1.TouchUp**: This block has input fields for x and y. It includes a 'do' block for custom logic.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) development environment. On the left, a vertical sidebar contains a 'My Blocks' palette with the following items: 'Built-In', 'My Blocks' (selected), and 'Advanced'. Below these are 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right is a light green canvas. A single block is placed on the canvas, representing an event listener. It starts with a 'when' trigger labeled 'Canvas1.TouchDown'. This trigger has two output ports: 'x' and 'y'. The 'x' port is connected to a 'name' property block, and the 'y' port is connected to another 'name' property block. The 'do' section of the block is currently empty.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Blocks' palette is visible, showing a list of components: 'Canvas1', 'Label1', and 'Screen1'. The 'Canvas1' component is selected. The main workspace on the right shows a custom block definition for 'Canvas1'. This definition includes three event blocks: 'when Canvas1.TouchDown', 'when Canvas1.TouchUp', and 'when Canvas1.Touched'. Each event block has a 'do' slot for custom code. The 'when Canvas1.TouchDown' block has two input fields for 'x' and 'y' coordinates, and a 'do' slot. The 'when Canvas1.TouchUp' block has two input fields for 'x' and 'y' coordinates, and a 'do' slot. The 'when Canvas1.Touched' block has two input fields for 'x' and 'y' coordinates, and a 'do' slot. Below these event blocks, there are three call blocks: 'Canvas1.Clear', 'Canvas1.DrawCircle', and 'Canvas1.DrawLine'. The 'Canvas1.DrawCircle' block has input fields for 'x', 'y', and 'r'. The 'Canvas1.DrawLine' block has input fields for 'x1', 'y1', 'x2', and 'y2'.

My Blocks | Built-In | Advanced

My Definitions

- Canvas1
- Label1
- Screen1

when Canvas1.TouchDown x y

do

when Canvas1.TouchUp x y

do

when Canvas1.Touched x y touchedSprite

do

call Canvas1.Clear

call Canvas1.DrawCircle x y r

call Canvas1.DrawLine x1 y1 x2 y2

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a tabbed menu containing 'Built-In', 'My Blocks', and 'Advanced'. The 'My Blocks' tab is selected, showing a list of components: 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right is a light green canvas where a Scratch-style block is assembled. The block starts with a 'when' trigger labeled 'Canvas1.TouchDown', which has two input fields for 'x' and 'y', each with a 'name' label. Below this is a 'do' block containing a 'call' block labeled 'Canvas1.DrawCircle', which also has input fields for 'x', 'y', and 'r'.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, there is a sidebar with three tabs: "Built-In", "My Blocks", and "Advanced". The "My Blocks" tab is selected. Below the tabs, there is a section titled "My Definitions" which lists three components: "Canvas1", "Label1", and "Screen1".

The main workspace on the right shows a Scratch-like block editor. A "Canvas1.TouchDown" block is being edited. It has two input fields labeled "x" and "y" with "value" labels. Below this block, there is a "do" block containing a "call" block. The "call" block is labeled "Canvas1.DrawCircle" and has three input fields labeled "x", "y", and "r".

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, a sidebar contains a 'My Blocks' tab, which is active. Below the tabs, a list of components is shown: 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right shows a block definition for 'Canvas1.TouchDown'. This block is a 'when' block with two input fields, 'x' and 'y', each labeled 'name'. Below the 'when' block is a 'do' block containing a 'call' block. The 'call' block is labeled 'Canvas1.DrawCircle' and has three input fields: 'x' (labeled 'value'), 'y', and 'r'.

Built-In **My Blocks** Advanced

My Definitions

Canvas1

Label1

Screen1

when **Canvas1.TouchDown**

x name x

y name y

do

call

Canvas1.DrawCircle

x value x

y

r

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, a sidebar contains a 'My Definitions' panel with a list of components: Canvas1, Label1, and Screen1. The 'Canvas1' component is selected. The main workspace shows a block definition for the 'Canvas1.TouchDown' event. This block is a green 'do' block with a 'call' sub-block. The 'call' sub-block is a purple block labeled 'Canvas1.DrawCircle'. The 'Canvas1.DrawCircle' block has three input fields: 'x' (labeled 'value x'), 'y' (labeled 'value y'), and 'r' (labeled 'value x'). The 'Canvas1.TouchDown' block has two input fields: 'x' (labeled 'value x') and 'y' (labeled 'value y').

My Definitions

- Canvas1
- Label1
- Screen1

Canvas1.TouchDown

- do
 - call
 - Canvas1.DrawCircle
 - x: value x
 - y: value y
 - r: value x

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a 'My Blocks' tab selected. Below the tabs are four categories: 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right contains a single block: a 'when Canvas1.TouchDown' event block. This block has two input fields, 'x' and 'y', each with a 'name' label. Below the event block is a 'do' block containing a 'call Canvas1.DrawCircle' block. The 'call' block has three input fields: 'x' (labeled 'value'), 'y' (labeled 'value'), and 'r'.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is the 'My Blocks' palette, which is organized into categories: Built-In, My Blocks, and Advanced. Under the 'My Blocks' category, there are several sub-categories: Definition, Text, Lists, Math, Logic, Control, and Colors. The 'Definition' sub-category is currently selected, showing a list of custom blocks. The main workspace on the right shows a custom block definition for a 'procedureWithResult' block. This block has an argument 'arg' and a 'do' block containing a 'Canvas1.TouchDown' block. The 'Canvas1.TouchDown' block has two arguments, 'x' and 'y', which are connected to 'name' and 'value' inputs respectively. The 'procedureWithResult' block has a 'return' block that outputs the 'name' and 'value' of the 'Canvas1.TouchDown' block. Below the 'Canvas1.TouchDown' block, there is a 'def variable' block with a 'name' input and a 'value' input. The 'Canvas1.DrawCircle' block is also visible in the workspace, with arguments for 'x', 'y', and 'r'.

My Blocks | Built-In | Advanced

Definition

- Text
- Lists
- Math
- Logic
- Control
- Colors

Custom Blocks:

- procedureWithResult** (arg)
 - do
 - Canvas1.TouchDown
 - x (name)
 - y (value)
 - return
 - name (y)
 - value (x)
- def variable** (as)
 - name (name)
 - value (value)
- Canvas1.DrawCircle** (x, y, r)

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a tabbed menu at the top containing 'Built-In', 'My Blocks', and 'Advanced'. Below the tabs are several category buttons: 'Definition' (blue), 'Text' (orange), 'Lists' (yellow), 'Math' (green), 'Logic' (orange), 'Control' (yellow), and 'Colors' (green). The main workspace on the right has a light green background and contains a Scratch-style block editor. The editor starts with a 'def variable as' block. Below it is a 'when Canvas1.TouchDown' block with two input fields labeled 'x' and 'y', each with a 'name' dropdown set to 'x' and 'y' respectively. A 'do' block is attached to the 'when' block, containing a 'call Canvas1.DrawCircle' block. The 'call' block has three input fields: 'x' (value 'x'), 'y' (value 'y'), and 'r' (empty).

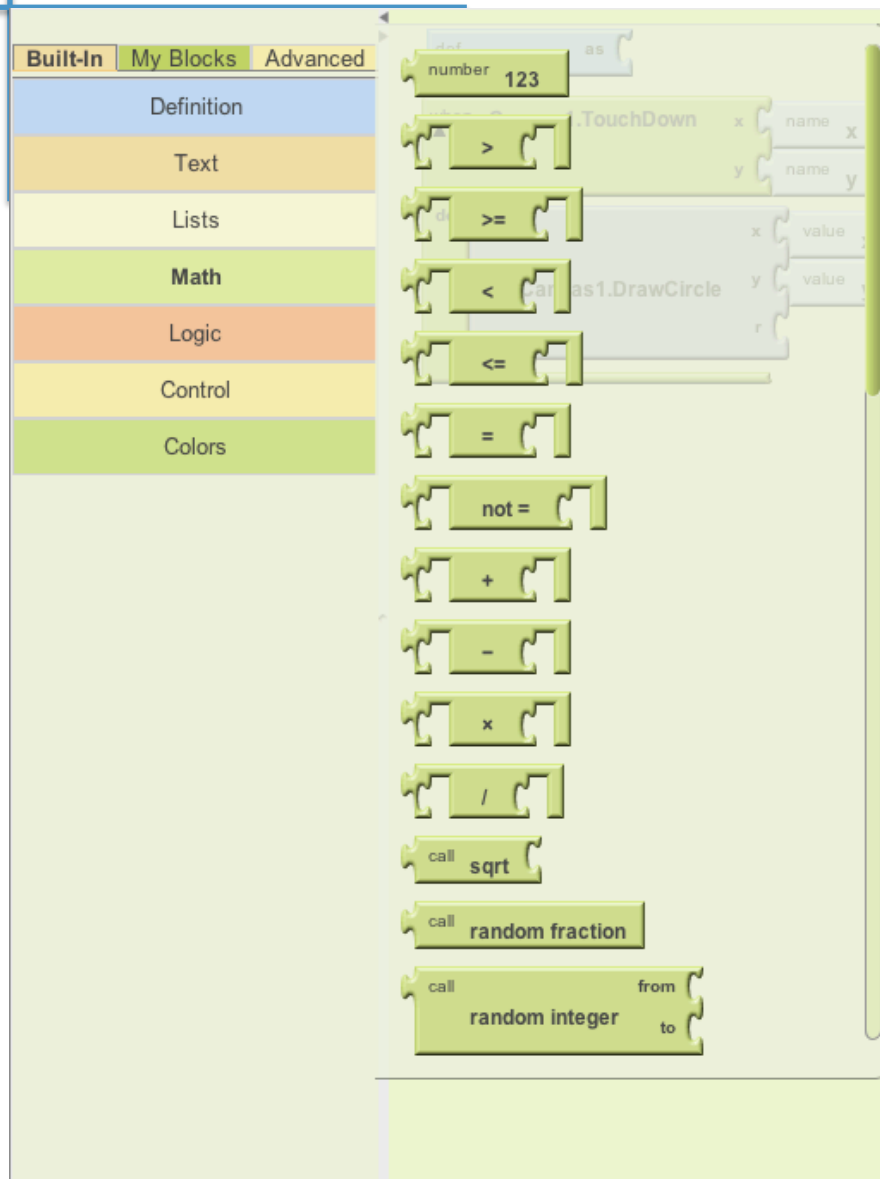
App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a tabbed menu at the top containing 'Built-In', 'My Blocks', and 'Advanced'. Below the tabs are several category buttons: 'Definition' (blue), 'Text' (orange), 'Lists' (yellow), 'Math' (green), 'Logic' (orange), 'Control' (yellow), and 'Colors' (green). The main workspace on the right has a light green background and contains a block definition for a function named 'radius'.

The 'radius' function is defined with the following structure:

- def radius as** (blue block)
- when Canvas1.TouchDown** (green block) with two input fields: 'x' (name x) and 'y' (name y).
- do** (green block) containing a **call** (purple block) to **Canvas1.DrawCircle**. The 'call' block has three input fields: 'x' (value x), 'y' (value y), and 'r'.

App Inventor Classic (AI1) Demo



App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a category menu. The menu has three tabs: "Built-In" (selected), "My Blocks", and "Advanced". Below the tabs are several category buttons: "Definition", "Text", "Lists", "Math", "Logic", "Control", and "Colors". The main workspace on the right is light green and contains a block definition for a variable named "radius".

The block definition consists of a blue "def" block with a triangle icon, labeled "radius", followed by an "as" block and a yellow "number" block with the value "123". Below this is a "when" block with a triangle icon, labeled "Canvas1.TouchDown". This "when" block has two input fields: "x" and "y", each with a "name" label. Below the "when" block is a "do" block with a "call" block. The "call" block is labeled "Canvas1.DrawCircle" and has three input fields: "x", "y", and "r", each with a "value" label. The "x" and "y" inputs are connected to the "x" and "y" inputs of the "when" block, and the "r" input is connected to the "radius" variable block.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a category menu. The menu has tabs for 'Built-In', 'My Blocks', and 'Advanced'. Below these tabs are several category buttons: 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The 'Logic' category is currently selected. The main workspace on the right is a light green area where code blocks are assembled. The code shown consists of a 'def' block with the name 'radius' and a value of '20'. Below this is a 'when Canvas1.TouchDown' event block. Inside the 'do' section of this event block is a 'call Canvas1.DrawCircle' block. The 'x' and 'y' inputs of the 'DrawCircle' block are connected to the 'x' and 'y' outputs of the 'TouchDown' event block. The 'r' input of the 'DrawCircle' block is connected to the 'radius' variable block.

Category Menu:

- Built-In
- My Blocks
- Advanced
- Definition
- Text
- Lists
- Math
- Logic
- Control
- Colors

Code Blocks:

```
def radius as number 20
```

```
when Canvas1.TouchDown
```

- x name x
- y name y

```
do
```

- call Canvas1.DrawCircle
 - x value x
 - y value y
 - r radius

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, a sidebar contains a 'My Definitions' panel with a list of components: 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right features a Scratch-style block editor. The blocks are as follows:

- A 'global radius as number 20' block.
- A 'set global radius to' block, which is currently empty.
- A 'value x' block.
- A 'value y' block.
- A 'Canvas1.DrawCircle' block, which has four input fields: 'x', 'y', 'value x', and 'value y'.

The 'Canvas1.DrawCircle' block is currently selected, and its 'x' and 'y' inputs are connected to the 'value x' and 'value y' blocks respectively. The 'value x' and 'value y' blocks are also connected to the 'x' and 'y' inputs of the 'Canvas1.DrawCircle' block.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a 'My Blocks' tab selected, showing a list of components: 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right contains a Scratch-style block editor with the following code:

```
def radius as number 20

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

The code defines a global variable `radius` with a value of 20. It then sets up a `when Canvas1.TouchDown` event listener. Inside the `do` block, it calls `Canvas1.DrawCircle` with three arguments: the `x` coordinate (labeled 'x' with 'value x'), the `y` coordinate (labeled 'y' with 'value y'), and the `radius` variable (labeled 'r' with 'global radius').

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Blocks' palette is visible, showing a custom block definition for 'Canvas1'. The main workspace on the right shows a sequence of blocks for handling touch events on 'Canvas1'.

My Blocks Palette:

- Built-In
- My Blocks**
- Advanced
- My Definitions
- Canvas1**
- Label1
- Screen1

Canvas1 Block Definition:

- when Canvas1.TouchUp** (x, y) → name x, name y
- do** (empty block)
- when Canvas1.Touched** (x, y) → value x, value y, touchedSprite (global radius)
- do** (empty block)
- call Canvas1.Clear**
- call Canvas1.DrawCircle** (x, y, r)
- call Canvas1.DrawLine** (x1, y1, x2, y2)
- call Canvas1.DrawPoint** (x, y)
- call** (text)

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a 'My Blocks' tab selected, showing a list of components: 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace on the right contains the following blocks:

- A **def radius** block with a type of 'number' and a value of '20'.
- A **when Canvas1.TouchDown** block with two input fields: 'x' (name 'x') and 'y' (name 'y'). It contains a **do** loop with a **call Canvas1.DrawCircle** block. The 'DrawCircle' block has three inputs: 'x' (value 'x'), 'y' (value 'y'), and 'r' (global 'radius').
- A **when Canvas1.TouchUp** block with two input fields: 'x' (name 'x1') and 'y' (name 'y1'). It contains an empty **do** loop.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is the **Blocks palette** with three tabs: **Built-In**, **My Blocks** (selected), and **Advanced**. The **My Blocks** tab shows a list of categories: **Definition**, **Text**, **Lists**, **Math**, **Logic**, **Control**, and **Colors**. The **Control** category is highlighted.

The main workspace shows a custom block assembly. It begins with a **test** block containing an **as** block with the value **number 20**. This is followed by an **if** block with a **then-do** section containing a **Canvas1.TouchDown** block. The **Canvas1.TouchDown** block has two input fields: **x** (with a dropdown menu set to **name** and value **x**) and **y** (with a dropdown menu set to **name** and value **y**). Below the **if** block is an **ifelse** block. The **ifelse** block has a **test** section and two branches: **then-do** and **else-do**. The **then-do** branch contains a **Canvas1.DrawCircle** block with inputs for **x** (value **x**), **y** (value **y**), and **r** (global variable **radius**). The **else-do** branch is currently empty.

Below the **ifelse** block is a **choose** block with a **test** section and two branches: **then-do** and **else-do**. The **then-do** branch contains a **then-return** block, and the **else-do** branch contains an **else-return** block. Below the **choose** block is a **foreach** block with a **variable** input field, a **do** section, and an **in list** input field. Below the **foreach** block is a **for range** block with a **variable** input field, **start**, **end**, and **step** input fields, and a **do** section.

On the right side of the workspace, there is a **Canvas1.TouchUp** block. It has two input fields: **x** (with a dropdown menu set to **name** and value **x1**) and **y** (with a dropdown menu set to **name** and value **y1**).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a category menu. The top of the sidebar has three tabs: "Built-In", "My Blocks", and "Advanced". Below these are six categories: "Definition", "Text", "Lists", "Math", "Logic", and "Colors". The "Logic" category is currently selected, and its sub-blocks are visible in the main workspace.

The main workspace contains the following code blocks:

- A **def** block with the name **radius** and a value of **20**.
- A **when Canvas1.TouchDown** event block. It has two input fields: **x** (name **x**) and **y** (name **y**). Below the event block is a **do** loop containing a **call** block. The **call** block has three arguments: **x** (value **x**), **y** (value **y**), and **r** (global **radius**). The block is labeled **Canvas1.DrawCircle**.
- A **when Canvas1.TouchUp** event block. It has two input fields: **x** (name **x1**) and **y** (name **y1**). Below the event block is a **do** loop containing a **for range** block. The **for range** block has three arguments: **variable** (name **i**), **start** (empty), **end** (empty), and **step** (number **1**). Below the **for range** block is an empty **do** loop.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu and a block palette. The category menu includes 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The block palette has tabs for 'Built-In', 'My Blocks', and 'Advanced'. The 'Built-In' tab is active, showing a list of blocks: 'def radius as number 20', 'when Canvas1.TouchDown', 'do', 'call', 'Canvas1.DrawCircle', 'when Canvas1.TouchUp', 'do', 'for range', 'variable', 'name', 'i', 'start', 'number', '1', 'end', 'value', 'y1', 'step', 'number', '2', 'x', and 'global radius'.

The main workspace contains two scripts. The first script, titled 'def radius as number 20', is a 'when Canvas1.TouchDown' event handler. It contains a 'do' block with a 'call' block that calls 'Canvas1.DrawCircle'. The 'call' block has three arguments: 'x' (name), 'y' (name), and 'r' (global radius). The second script, titled 'when Canvas1.TouchUp', is a 'when Canvas1.TouchUp' event handler. It contains a 'do' block with a 'for range' loop. The 'for range' block has four arguments: 'variable' (name), 'start' (number), 'end' (value), and 'step' (number). The 'do' block contains a 'global radius' block.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a category menu. The main workspace on the right contains two event-driven code blocks.

Category Menu (Left Sidebar):

- Built-In
- My Blocks
- Advanced
- Definition
- Text
- Lists
- Math
- Logic
- Control
- Colors

Canvas1.TouchDown Event Block:

- when Canvas1.TouchDown** (x: name x, y: name y)
- do** (call Canvas1.DrawCircle with x: value x, y: value y, r: global radius)

Canvas1.TouchUp Event Block:

- when Canvas1.TouchUp** (x: name x1, y: name y1)
- do** (for range loop: variable i, start number 1, end value y1, step number 2, x: global radius)
- do** (call Canvas1.DrawCircle with x: value x, y: value y, r: global radius)

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a vertical sidebar with a category menu. The main workspace on the right contains two event-driven code blocks.

Category Menu (Left Sidebar):

- Built-In
- My Blocks
- Advanced
- Definition
- Text
- Lists
- Math
- Logic
- Control
- Colors

Canvas1.TouchDown Event Block:

- when Canvas1.TouchDown** (x, y)
 - do**
 - call** Canvas1.DrawCircle (x, y, r)
 - x: value x
 - y: value y
 - r: global radius

Canvas1.TouchUp Event Block:

- when Canvas1.TouchUp** (x, y)
 - do**
 - for range**
 - variable: i
 - start: number 1
 - end: value y1
 - step: number 2
 - do**
 - call** Canvas1.DrawCircle (x, y, r)
 - x: global radius
 - y: global radius
 - r: global radius

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Definitions' panel lists 'Canvas1', 'Label1', and 'Screen1'. The main workspace shows two event-driven code blocks.

Left Code Block (Canvas1.TouchDown):

- Block: `Canvas1.TouchDown` (x, y)
- Block: `set global radius to` (value: `20`)
- Block: `Canvas1.DrawCircle` (x: `value x`, y: `value y`, r: `global radius`)
- Block: `value x1`
- Block: `value y`
- Block: `value y1`

Right Code Block (Canvas1.TouchUp):

- Block: `Canvas1.TouchUp` (x, y)
- Block: `for range` (variable: `i`, start: `number 1`, end: `value y1`, step: `number 2`)
- Block: `do` (call: `Canvas1.DrawCircle` (x: `x`, y: `y`, r: `global radius`)))

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Blocks' tab is active, showing a list of components: Canvas1, Label1, and Screen1. The main workspace contains two event-driven code blocks:

- Canvas1.TouchDown:** This block triggers a 'do' loop containing a 'call' block for 'Canvas1.DrawCircle'. The parameters for 'Canvas1.DrawCircle' are: x (name x), y (name y), and r (global radius).
- Canvas1.TouchUp:** This block triggers a 'do' loop containing a 'for range' loop. The 'for range' loop has the following parameters: variable (name i), start (number 1), end (value y1), and step (number 2). Inside the 'for range' loop, there is a 'do' loop containing a 'call' block for 'Canvas1.DrawCircle'. The parameters for 'Canvas1.DrawCircle' are: x (value x1), y (name y1), and r (global radius).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Definitions' panel lists 'Canvas1', 'Label1', and 'Screen1'. The main workspace shows two event-driven code blocks:

- Canvas1.TouchDown** event block:
 - Block: `set global radius to` with value `20`.
 - Block: `Canvas1.DrawCircle` with parameters:
 - `x`: `value x`
 - `y`: `value y`
 - `r`: `global radius`
- Canvas1.TouchUp** event block:
 - Block: `for range` with parameters:
 - `variable`: `i`
 - `start`: `number 1`
 - `end`: `value y1`
 - `step`: `number 2`
 - Block: `do` loop containing:
 - Block: `call` with parameters:
 - `Canvas1.DrawCircle` with parameters:
 - `x`: `value x1`
 - `y`: `global radius`

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, the 'My Blocks' tab is selected, showing a list of components: Canvas1, Label1, and Screen1. The main workspace contains two event-driven blocks:

- Canvas1.TouchDown:** This block triggers a 'do' loop containing a 'call' block for 'Canvas1.DrawCircle'. The parameters for 'Canvas1.DrawCircle' are: x (name x), y (name y), and r (global radius).
- Canvas1.TouchUp:** This block triggers a 'do' loop containing a 'for range' block. The 'for range' block has the following parameters: variable (name i), start (number 1), end (value y1), and step (number 2). Inside the 'do' loop, there is a 'call' block for 'Canvas1.DrawCircle' with parameters: x (value x1), y (value i), and r (global radius).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category list: Built-In, My Blocks, and Advanced. Under 'Built-In', the categories are Definition, Text, Lists, Math, Logic, Control, and Colors. The 'My Blocks' tab is selected, showing a list of custom blocks: a 'procedureWithResult' block with arguments 'x' (name) and 'y' (name), a 'procedure' block with arguments 'x' (value) and 'y' (value), a 'variable' block with argument 'name', and a 'radius' block with argument 'global radius'.

The main workspace shows two scripts. The first script, 'procedureWithResult', is a 'to procedureWithResult' block with arguments 'x' and 'y'. It contains a 'do' block with 'Canvas1.TouchDown' and a 'return' block with arguments 'x' and 'y'. The second script, 'Canvas1.TouchUp', is a 'when Canvas1.TouchUp' block with arguments 'x' and 'y'. It contains a 'do' block with a 'for range' loop. The 'for range' block has arguments 'variable' (name 'i'), 'start' (number '1'), 'end' (value 'y1'), and 'step' (number '2'). The 'do' block contains a 'call' block with arguments 'x' (value 'x1'), 'y' (value 'i'), and 'r' (global 'radius').

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu and a block palette. The category menu includes 'Built-In', 'My Blocks', and 'Advanced'. The block palette lists 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The main workspace contains the following blocks:

- def radius as number 20**: A block to define a global variable named 'radius' with a value of 20.
- when Canvas1.TouchDown**: A 'when clicked' block with two input fields for 'x' (name 'x') and 'y' (name 'y').
- do**: A 'do' block containing a **call Canvas1.DrawCircle** block. The 'x' input is 'value x', the 'y' input is 'value y', and the 'r' input is 'global radius'.
- to procedure arg**: A 'to procedure' block with an 'arg' input field and a 'do' block.
- when Canvas1.TouchUp**: A 'when clicked' block with two input fields for 'x' (name 'x1') and 'y' (name 'y1').
- do**: A 'do' block containing a **for range** block. The 'variable' input is 'i', the 'start' input is 'number 1', the 'end' input is 'value y1', and the 'step' input is 'number 2'. The 'x' input of the 'for range' block is 'global radius'.
- do**: A 'do' block containing a **call Canvas1.DrawCircle** block. The 'x' input is 'value x1', the 'y' input is 'value i', and the 'r' input is 'global radius'.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, a sidebar contains a category menu with 'Built-In', 'My Blocks', and 'Advanced' tabs. Below these are categories: 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The main workspace is divided into two sections. The left section contains three code blocks: a 'procedureWithResult' block with a 'do' loop containing 'Canvas1.TouchDown' and a 'return' block; a 'procedure' block with a 'do' loop containing 'Canvas1.DrawCircle'; and a 'def variable' block with a 'name' block and an 'arg' block. The right section contains a 'when Canvas1.TouchUp' block with a 'do' loop containing a 'for range' block. The 'for range' block has a 'variable' block with 'name i', a 'start' block with 'number 1', an 'end' block with 'value y1', and a 'step' block with 'number 2'. The 'do' loop contains a 'call' block with 'Canvas1.DrawCircle'. The 'Canvas1.DrawCircle' block has three inputs: 'x' with 'value x1', 'y' with 'value i', and 'r' with 'global radius'.

Left Section Code:

```
def procedureWithResult arg  
do  
  Canvas1.TouchDown x name x  
  return y name y  
end  
  
def procedure arg  
do  
  Canvas1.DrawCircle x value x y value y r global radius  
end  
  
def variable as  
name name arg  
do  
  I  
end
```

Right Section Code:

```
when Canvas1.TouchUp x name x1 y name y1  
do  
  for range variable name i start number 1 end value y1 step number 2 x global radius  
  do  
    call Canvas1.DrawCircle x value x1 y value i r global radius  
  end  
end
```

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu and a block palette. The category menu includes 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The block palette has tabs for 'Built-In', 'My Blocks', and 'Advanced'. The workspace on the right contains the following code blocks:

- def radius as** (blue block) with a value of **number 20**.
- when Canvas1.TouchDown** (green block) with inputs **x** (name **x**) and **y** (name **y**). It contains a **do** loop with a **call Canvas1.DrawCircle** block. The **DrawCircle** block has inputs **x** (value **x**), **y** (value **y**), and **r** (global **radius**).
- to drawDots** (purple block) with two **arg** inputs, one of which is named **name**. It contains a **do** loop.
- when Canvas1.TouchUp** (green block) with inputs **x** (name **x1**) and **y** (name **y1**). It contains a **do** loop with a **for range** block. The **for range** block has inputs **variable** (name **i**), **start** (number **1**), **end** (value **y1**), and **step** (number **2**). It also has an input **x** (global **radius**). Inside the **do** loop is a **call Canvas1.DrawCircle** block with inputs **x** (value **x1**), **y** (value **i**), and **r** (global **radius**).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu containing: Built-In, My Blocks, and Advanced. Below the menu are tabs for Definition, Text, Lists, Math, Logic, Control, and Colors. The main workspace on the right contains the following code blocks:

- def radius as** (blue block) with a value of **number 20**.
- when Canvas1.TouchDown** (green block) with inputs **x** (name **x**) and **y** (name **y**).
 - do** (green block) containing a **call Canvas1.DrawCircle** (purple block) with inputs **x** (value **x**), **y** (value **y**), and **r** (global **radius**).
- to drawDots** (purple block) with two arguments, the first being **name x2**.
- when Canvas1.TouchUp** (green block) with inputs **x** (name **x1**) and **y** (name **y1**).
 - do** (green block) containing a **for range** (yellow block) with inputs **variable** (name **i**), **start** (number **1**), **end** (value **y1**), and **step** (number **2**). The **for range** block has an output **x** (global **radius**).
 - do** (green block) containing a **call Canvas1.DrawCircle** (purple block) with inputs **x** (value **x1**), **y** (value **i**), and **r** (global **radius**).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left, a sidebar contains a category menu with 'Built-In', 'My Blocks', and 'Advanced' tabs. Below these are categories: 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The main workspace is divided into two panels. The left panel shows a 'procedureWithResult' block with a 'do' loop containing a 'Canvas1.TouchDown' block. The 'return' block has two arguments: 'name x' and 'name y'. Below this is a 'procedure' block with a 'do' loop containing a 'Canvas1.DrawCircle' block. The 'Canvas1.DrawCircle' block has three arguments: 'value x', 'value y', and 'global radius'. Below the 'Canvas1.DrawCircle' block is a 'def variable' block with 'name' and 'as' arguments. The right panel shows a 'when Canvas1.TouchUp' block with two arguments: 'x name x1' and 'y name y1'. Below this is a 'do' loop containing a 'for range' block. The 'for range' block has three arguments: 'variable name i', 'start number 1', and 'end value y1'. Below the 'for range' block is a 'do' loop containing a 'call' block. The 'call' block has three arguments: 'x value x1', 'y value i', and 'r global radius'. The 'Canvas1.DrawCircle' block is called within the 'do' loop.

Left Panel Code:

```
def procedureWithResult arg  
  do  
    Canvas1.TouchDown x name x  
    return name y  
  end do  
end procedure  
def variable as  
  name name arg name x2  
  arg  
end def
```

Right Panel Code:

```
when Canvas1.TouchUp x name x1  
  y name y1  
do  
  for range variable name i  
    start number 1  
    end value y1  
    step number 2 x global radius  
  do  
    call  
      Canvas1.DrawCircle x value x1  
      y value i  
      r global radius  
    end call  
  end do  
end do
```

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu and a block palette. The category menu includes: Definition, Text, Lists, Math, Logic, Control, and Colors. The block palette has tabs for Built-In, My Blocks, and Advanced. The workspace on the right contains the following code blocks:

- def radius** as number 20
- when Canvas1.TouchDown** block with inputs: x (name x), y (name y). The **do** section contains a **call Canvas1.DrawCircle** block with inputs: x (value x), y (value y), and r (global radius).
- to drawDots** block with inputs: arg (name x2), arg (name name), and an empty **do** section.
- when Canvas1.TouchUp** block with inputs: x (name x1), y (name y1). The **do** section contains a **for range** block with inputs: variable (name i), start (number 1), end (value y1), and step (number 2). The **do** section of the **for range** block contains a **call Canvas1.DrawCircle** block with inputs: x (value x1), y (value i), and r (global radius).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu and a block palette. The category menu includes 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The block palette has tabs for 'Built-In', 'My Blocks', and 'Advanced'. The 'Built-In' tab is active, showing various block categories. The workspace on the right contains the following code blocks:

- def radius as number 20**: A block to define a global variable named 'radius' with a value of 20.
- when Canvas1.TouchDown**: A block that triggers when the canvas is touched down. It has two input fields for 'x' (name x) and 'y' (name y). Inside a 'do' loop, it calls **Canvas1.DrawCircle** with inputs: 'x' (value x), 'y' (value y), and 'r' (global radius).
- to drawDots**: A block that defines a function named 'drawDots'. It has three arguments: 'x2' (name x2), 'y2' (name y2), and an empty 'arg' field. It contains a 'do' loop.
- when Canvas1.TouchUp**: A block that triggers when the canvas is touched up. It has two input fields for 'x' (name x1) and 'y' (name y1). Inside a 'do' loop, it contains a **for range** block with the following settings:
 - variable: i
 - start: number 1
 - end: value y1
 - step: number 2The 'for range' block has an 'x' input field set to 'global radius'. Inside the 'do' loop of the 'for range' block, it calls **Canvas1.DrawCircle** with inputs: 'x' (value x1), 'y' (value i), and 'r' (global radius).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a category menu and a block palette. The category menu includes 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The block palette has tabs for 'Built-In', 'My Blocks', and 'Advanced'. The 'Built-In' tab is active, showing a list of blocks: 'Definition', 'Text', 'Lists', 'Math', 'Logic', 'Control', and 'Colors'. The 'My Blocks' tab is also visible, showing a custom block named 'radius'.

The main workspace contains three block structures:

- Canvas1.TouchDown** block: A 'when' block that triggers a 'do' loop containing a 'call' block for 'Canvas1.DrawCircle'. The 'x' and 'y' inputs are connected to 'name x' and 'name y' blocks. The 'r' input is connected to a 'global radius' block.
- Canvas1.TouchUp** block: A 'when' block that triggers a 'do' loop. The 'x' and 'y' inputs are connected to 'name x1' and 'name y1' blocks.
- drawDots** block: A 'to' block with three arguments: 'name x2', 'name y2', and 'arg'. It contains a 'do' loop with a 'for range' block. The 'for range' block has 'variable i', 'start 1', 'end y1', and 'step 2'. It also has a 'x' input connected to a 'global radius' block. Inside the 'do' loop is a 'call' block for 'Canvas1.DrawCircle' with inputs 'x1', 'i', and 'radius'.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface, which is divided into two main sections: a left-hand sidebar and a central workspace.

Left Sidebar:

- Top Tabs:** "Built-In", "My Blocks", and "Advanced". The "My Blocks" tab is currently selected.
- My Definitions:** A list of components defined in the project, including "Canvas1", "Label1", and "Screen1".

Central Workspace:

The workspace contains several blocks of code:

- Global Variable Definition:** A "def" block named "radius" is defined with a value of "20".
- Canvas1.TouchDown Event:** A "when" block that triggers a "do" loop. Inside the loop, a "call" block is used to execute "Canvas1.DrawCircle". The arguments for this call are: "x" (value of "x"), "y" (value of "y"), and "r" (global variable "radius").
- Canvas1.TouchUp Event:** A "when" block that triggers a "do" loop, currently empty.
- drawDots Function:** A "to" block named "drawDots" that takes three arguments: "x2", "y2", and an unnamed argument. Inside the "do" loop, a "for range" block is used to iterate from "1" to "radius" (global variable) with a step of "2". Inside the loop, a "call" block is used to execute "Canvas1.DrawCircle" with arguments "x", "y" (value of "i"), and "r" (global variable "radius").

App Inventor Classic (AI1) Demo

The image shows a Scratch script for drawing a circle with dots. The script is organized into three sections: Built-In, My Blocks, and Advanced. The My Definitions section lists Canvas1, Label1, and Screen1. The script starts with a 'when Canvas1.TouchUp' event, followed by a 'do' block containing a 'call' block for 'Canvas1.DrawCircle'. The 'Canvas1.DrawCircle' block has arguments for 'x', 'y', and 'radius'. The 'radius' argument is connected to a 'global radius' block. The 'x' and 'y' arguments are connected to 'value' blocks for 'x' and 'y'. The script also includes a 'set global radius to' block and a 'for range' loop for 'i' from 1 to 2, with a step of 2. The loop contains a 'call' block for 'Canvas1.DrawCircle' with arguments for 'x', 'y', and 'radius'.

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface. On the left is a sidebar with a 'My Blocks' tab selected, showing a list of components: 'My Definitions', 'Canvas1', 'Label1', and 'Screen1'. The main workspace contains three block definitions:

- def radius as number 20**: A block defining a global variable named 'radius' with a value of 20.
- when Canvas1.TouchDown**: A block that triggers an event when the canvas is touched. It has two input fields for 'x' and 'y' (both labeled 'name'). Inside the 'do' loop, there is a **call Canvas1.DrawCircle** block with inputs for 'x' (value 'x'), 'y' (value 'y'), and 'r' (global 'radius').
- when Canvas1.TouchUp**: A block that triggers an event when the canvas is touched up. It has two input fields for 'x1' and 'y1' (both labeled 'name'). The 'do' loop is currently empty.

Below these, there is a block definition for **to drawDots**, which is a function with three arguments: 'x2' (name), 'y2' (name), and an unlabeled argument. Inside its 'do' loop, there is a **for range** block with 'variable' 'i', 'start' '1', and 'step' '2'. The 'step' block also has a multiplier 'x' and a global 'radius' input. Inside the 'for range' loop, there is a **call Canvas1.DrawCircle** block with inputs for 'x' (value 'x2'), 'y' (value 'i'), and 'r' (global 'radius').

App Inventor Classic (AI1) Demo

The image shows a Scratch code editor with a script for drawing a circle. The script is as follows:

```

when green flag clicked
  set radius to 20
  drawDots
  Canvas1.TouchUp
  do
  
```

The 'My Definitions' panel shows the following definitions:

- Canvas1
- Label1
- Screen1

The 'drawDots' function is defined in the 'My Definitions' panel, showing a loop that draws multiple circles:

```

drawDots
  for range
    variable i
    start number 1
    end
    step
      number 2
      x
      global radius
    do
      call
        Canvas1.DrawCircle
        x value x2
        y value i
        r global radius
      end
    end
  end

```

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface, which is divided into three main sections: a left sidebar, a top toolbar, and a central workspace.

Left Sidebar: This section contains a list of components and their definitions. At the top, there are three tabs: "Built-In", "My Blocks", and "Advanced". Below these tabs, the "My Definitions" section lists the following components:

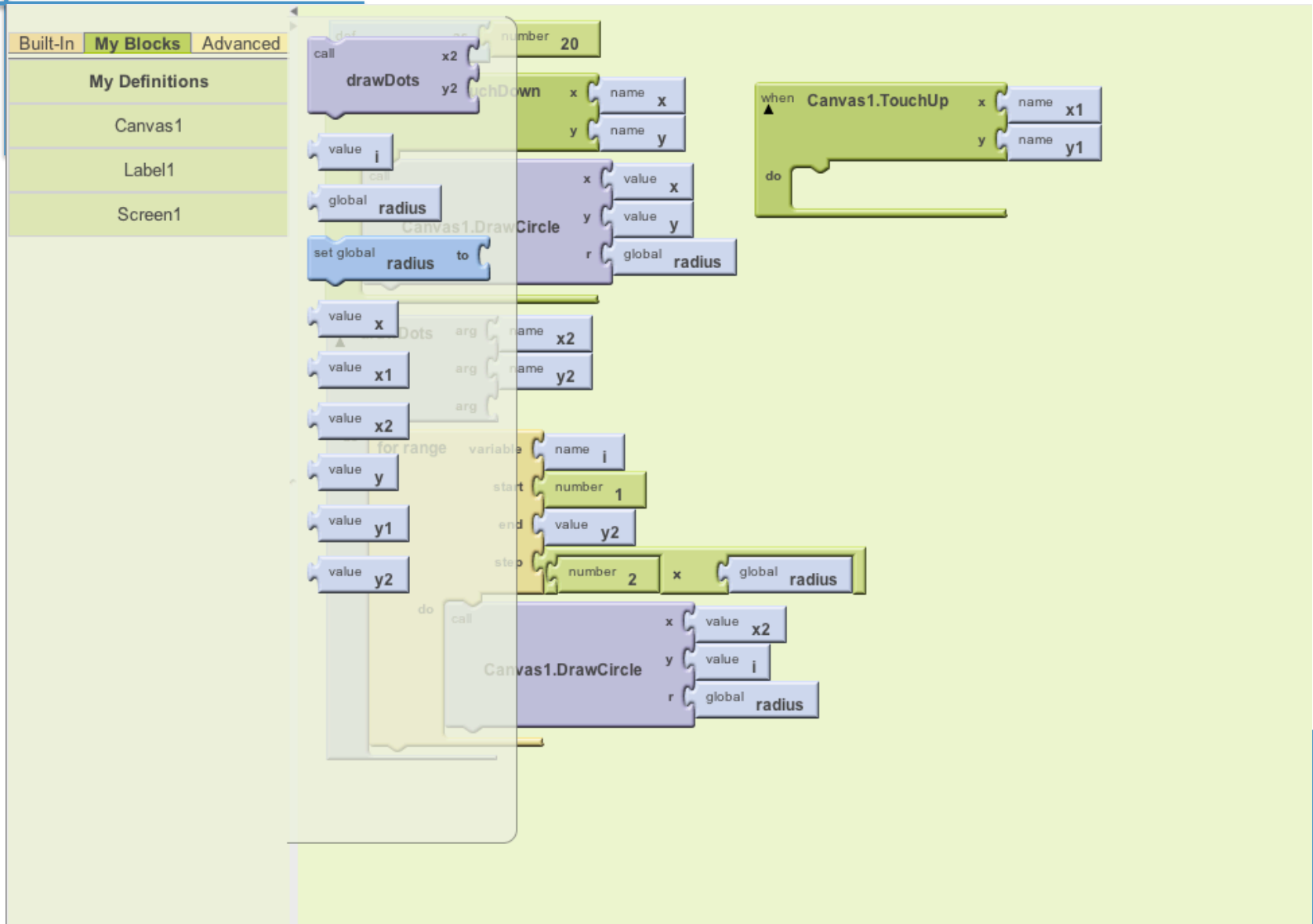
- Canvas1
- Label1
- Screen1

Top Toolbar: This section contains a set of blocks for defining and using variables. It includes a "def" block for defining a variable, a "radius" block for defining a variable, and a "number" block for defining a variable.

Central Workspace: This section contains the visual programming logic. It features several blocks and their connections:

- Canvas1.TouchDown** block: This block is connected to a "do" block. Inside the "do" block, there is a "call" block that calls the **Canvas1.DrawCircle** method. The arguments for this call are: "x" (value x), "y" (value y), and "r" (global radius).
- Canvas1.TouchUp** block: This block is connected to a "do" block. Inside the "do" block, there is a "call" block that calls the **Canvas1.DrawCircle** method. The arguments for this call are: "x" (value x1), "y" (value y1), and "r" (global radius).
- drawDots** block: This block is connected to a "do" block. Inside the "do" block, there is a "for range" block. The "for range" block has the following settings: "variable" (name i), "start" (number 1), "end" (value y2), and "step" (number 2). The "for range" block is connected to a "do" block. Inside this "do" block, there is a "call" block that calls the **Canvas1.DrawCircle** method. The arguments for this call are: "x" (value x2), "y" (value i), and "r" (global radius).

App Inventor Classic (AI1) Demo



App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface, which is divided into three main sections: a left sidebar, a top toolbar, and a central workspace.

Left Sidebar: This section contains a list of components and a definitions panel. The top row has tabs for "Built-In", "My Blocks", and "Advanced". Below these are the following components: "My Definitions", "Canvas1", "Label1", and "Screen1".

Top Toolbar: This section contains a row of three tabs: "Built-In", "My Blocks", and "Advanced".

Central Workspace: This section contains the visual programming blocks for the application. The blocks are organized as follows:

- Global Variable Definition:** A "def" block is used to define a global variable named "radius" with a value of 20.
- Canvas1.TouchDown Event:** A "when" block is used to trigger an action when the canvas is touched. It contains a "do" block with a "call" block that calls the "Canvas1.DrawCircle" method. The arguments for the "call" block are: "x" (value x), "y" (value y), and "r" (global radius).
- Canvas1.TouchUp Event:** A "when" block is used to trigger an action when the canvas is touched. It contains a "do" block with a "call" block that calls the "drawDots" method. The arguments for the "call" block are: "x2" (value x1) and "y2" (value y1).
- drawDots Function:** A "to" block is used to define a function named "drawDots". It has three arguments: "arg" (name x2), "arg" (name y2), and "arg" (name y2). The function contains a "do" block with a "for range" block. The "for range" block has the following arguments: "variable" (name i), "start" (number 1), "end" (value y2), and "step" (number 2). The "for range" block contains a "do" block with a "call" block that calls the "Canvas1.DrawCircle" method. The arguments for the "call" block are: "x" (value x2), "y" (value i), and "r" (global radius).

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface, which is divided into three main sections: a left sidebar, a top toolbar, and a central workspace.

Left Sidebar (My Definitions):

- Built-In:** Contains blocks for Canvas1, Label1, and Screen1.
- My Blocks:** Contains a custom block named **drawDots**.
- Advanced:** Contains a custom block named **radius**.

Top Toolbar:

- def radius as number 20:** A block that defines a global variable named **radius** with a value of 20.

Central Workspace:

- Canvas1.TouchDown:** A block that triggers an event when the canvas is touched. It has two input fields for **x** and **y**, each with a "name" label.
- Canvas1.DrawCircle:** A block that draws a circle on the canvas. It has three input fields: **x** (value **x**), **y** (value **y**), and **r** (global **radius**).
- Canvas1.TouchUp:** A block that triggers an event when the canvas is touched again. It has two input fields for **x** and **y**, each with a "name" label.
- drawDots:** A custom block that draws a series of dots. It has three input fields: **x2** (value **x2**), **y2** (value **y2**), and **radius** (global **radius**).
- for range:** A block that loops through a range of values. It has four input fields: **variable** (name **i**), **start** (number **1**), **end** (value **y2**), and **step** (number **2**).
- Canvas1.DrawCircle (inside for range):** A block that draws a circle on the canvas. It has three input fields: **x** (value **x2**), **y** (value **i**), and **r** (global **radius**).

App Inventor Classic (AI1) Demo

Built-In **My Blocks** **Advanced**

My Definitions

- Canvas1
- Label1
- Screen1

def radius as number 20

when Canvas1.TouchDown

- x name x
- y name y
- do call Canvas1.DrawCircle
 - x value x
 - y value y
 - r global radius

when Canvas1.TouchUp

- x name x1
- y name y1
- do call drawDots
 - x2 value x1
 - y2 value y1

to drawDots

- arg name x2
- arg name y2
- do
 - 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
 - x value x2
 - y value i
 - r global radius
 - set global numDots to number 1 + global numDots
- set Label1.Text to global numDots

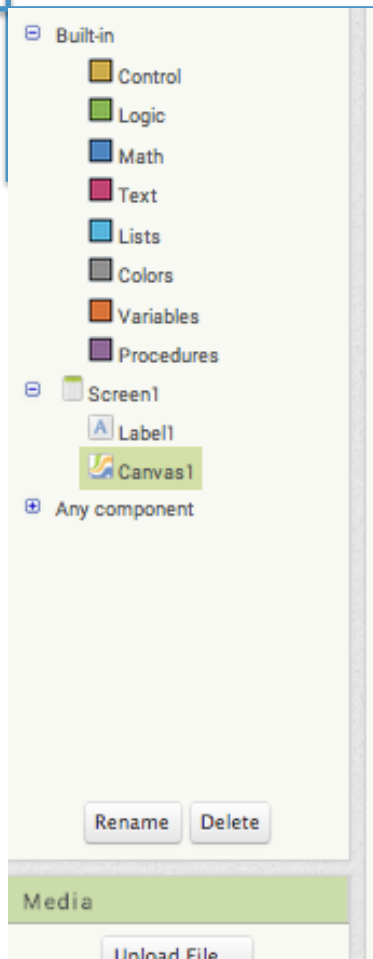
def numDots as number 0

App Inventor Classic (AI1) Demo

The screenshot displays the App Inventor Classic (AI1) interface, showing a project titled "My Definitions" with components: Canvas1, Label1, and Screen1. The main workspace contains several blocks:

- def radius as number 20**: A block defining a global variable named `radius` with a value of 20.
- when Canvas1.TouchDown**: A block that triggers an event when the canvas is touched down. It has two input fields: `x` (name) and `y` (name). The `do` block contains a **call Canvas1.DrawCircle** block with inputs: `x` (value), `y` (value), and `r` (global `radius`).
- when Canvas1.TouchUp**: A block that triggers an event when the canvas is touched up. It has two input fields: `x` (name) and `y` (name). The `do` block contains a **call drawDots** block with inputs: `xDots` (value `x1`) and `y2` (value `y1`).
- to drawDots**: A block that defines a function named `drawDots`. It has three arguments: `xDots` (name), `y2` (name), and an empty `arg` field. The `do` block contains:
 - set global numDots to number 0**: A block that sets the global variable `numDots` to 0.
 - for range**: A block that iterates over a range of values. It has inputs: `variable` (name `i`), `start` (number 1), `end` (value `y2`), and `step` (number 2). It also has a `x` input field with a global `radius` value.
 - do**: A block that contains a **call Canvas1.DrawCircle** block with inputs: `x` (value `xDots`), `y` (value `i`), and `r` (global `radius`).
 - set global numDots to number 1 + global numDots**: A block that increments the global variable `numDots` by 1.
 - set Label1.Text to global numDots**: A block that sets the text of `Label1` to the value of the global variable `numDots`.
- def numDots as number 0**: A block defining a global variable named `numDots` with a value of 0.

App Inventor 2 (AI2) Demo



App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the Components palette, which is organized into categories: Built-in, Screen1, and Any component. The Built-in category includes Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. The Screen1 category includes Label1 and Canvas1. The Any component category is also visible. Below the palette are buttons for 'Rename' and 'Delete'. At the bottom left, there is a 'Media' section with an 'Upload File ...' button.

The main area on the right shows the Events panel for Canvas1. It contains several event blocks:

- when Canvas1 .Dragged**: Includes input fields for startX, startY, prevX, prevY, currentX, currentY, and draggedSprite, followed by a 'do' block.
- when Canvas1 .Flung**: Includes input fields for x, y, speed, heading, xvel, yvel, and flungSprite, followed by a 'do' block.
- when Canvas1 .TouchDown**: Includes input fields for x and y, followed by a 'do' block.
- when Canvas1 .TouchUp**: Includes input fields for x and y, followed by a 'do' block.
- when Canvas1 .Touched**: Includes input fields for x, y, and touchedSprite, followed by a 'do' block.

Below these event blocks, there is a 'call Canvas1 .Clear' block.

App Inventor 2 (AI2) Demo

☰ Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

☰ Screen1

- Label1
- Canvas1

☰ Any component

Rename Delete

Media

Upload File ...

when Canvas1 .TouchDown

x y

do

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Screen1' section, 'Canvas1' is selected. Below the palette are 'Rename' and 'Delete' buttons. At the bottom left, there is a 'Media' section with an 'Upload File...' button. The main workspace on the right contains a sequence of visual programming blocks:

- A 'when Canvas1 .TouchUp' event block with a 'do' loop containing a 'y' variable block.
- A 'when Canvas1 .Touched' event block with a 'do' loop containing 'x', 'y', and 'touchedSprite' blocks.
- A 'call Canvas1 .Clear' block.
- A 'call Canvas1 .DrawCircle' block with inputs for 'x', 'y', and 'r'.
- A 'call Canvas1 .DrawLine' block with inputs for 'x1', 'y1', 'x2', and 'y2'.
- A 'call Canvas1 .DrawPoint' block with an input for 'x'.

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

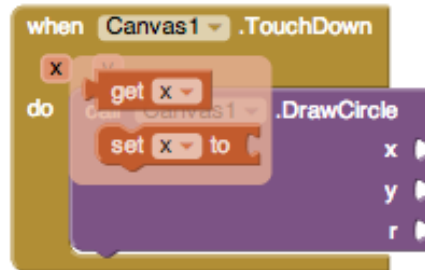
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

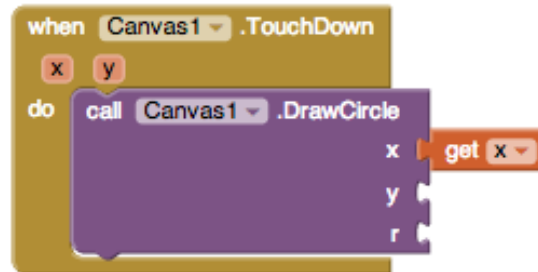
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

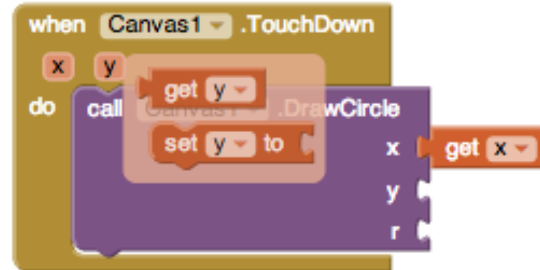
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

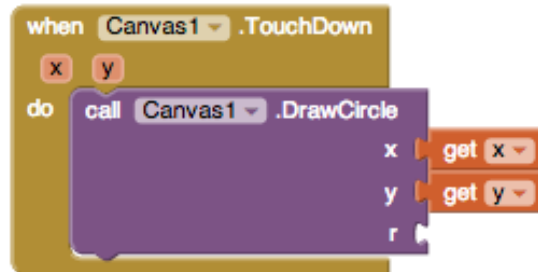
- Label1
- Canvas1

Any component

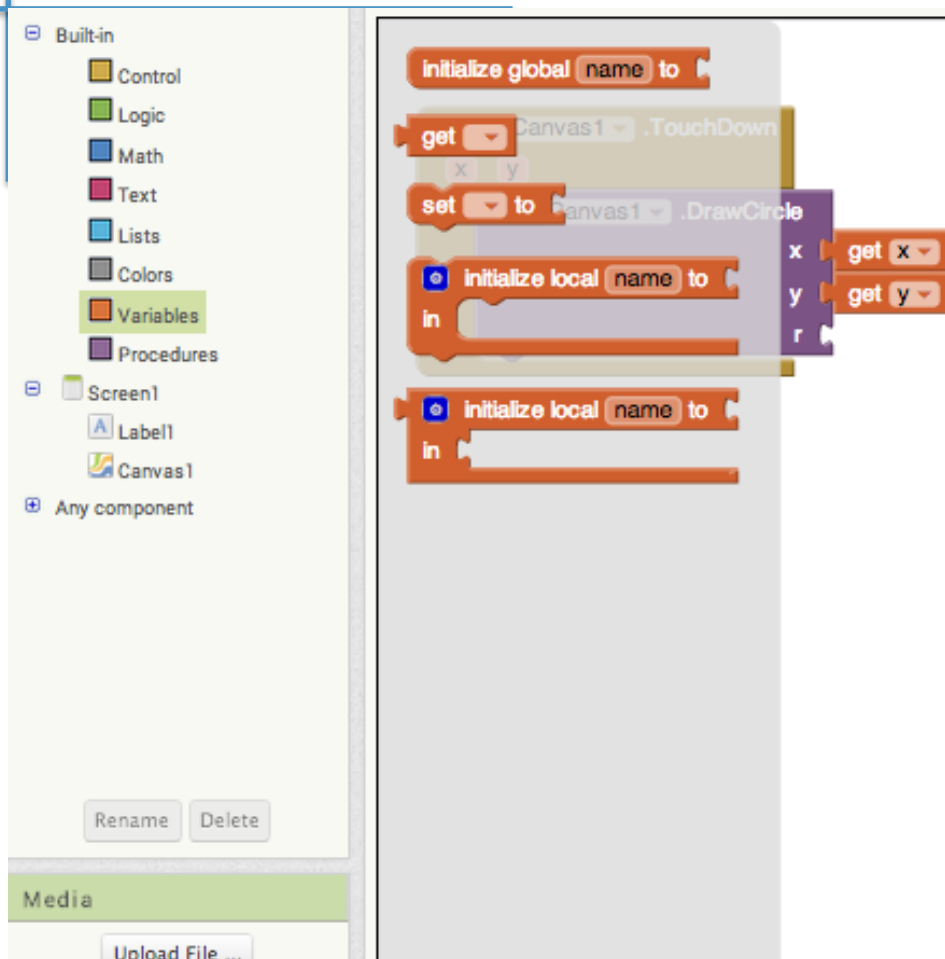
Rename Delete

Media

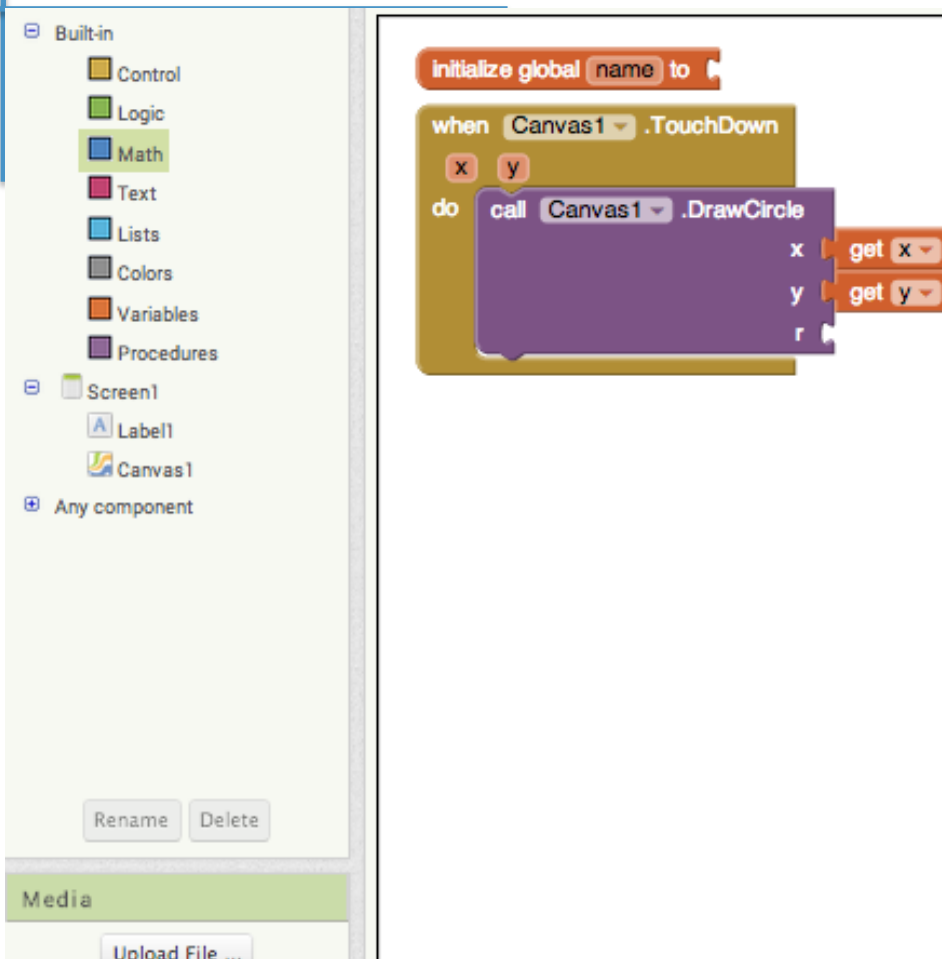
Upload File ...



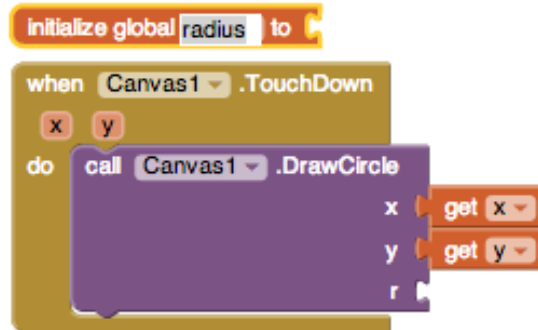
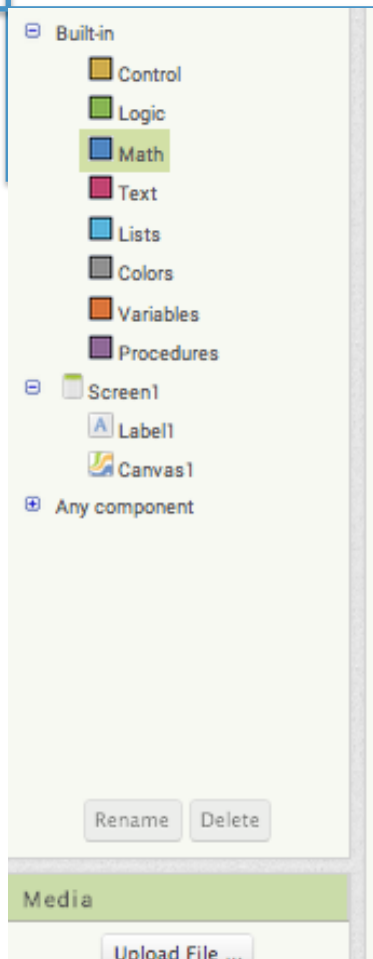
App Inventor 2 (AI2) Demo



App Inventor 2 (AI2) Demo



App Inventor 2 (AI2) Demo



App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the component palette, which is organized into sections: 'Built-in' (containing Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures), 'Screen1' (containing Label1 and Canvas1), and 'Any component'. Below the palette are 'Rename' and 'Delete' buttons. At the bottom of the interface is a 'Media' section with an 'Upload File ...' button.

The main workspace on the right shows a code block for the Canvas1 component. The code is as follows:

```
initialize global radius to  
when Canvas1.TouchDown  
  x y  
  do call Canvas1.DrawCircle  
    x get x  
    y get y  
    r
```

App Inventor 2 (AI2) Demo



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

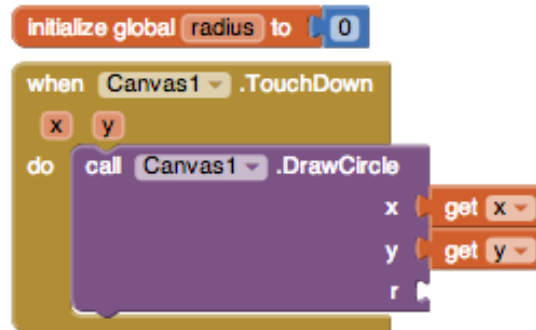
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

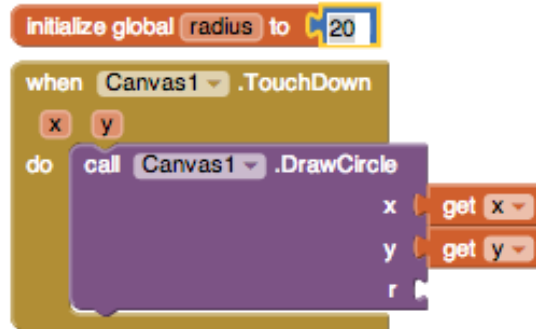
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Screen1' section, 'Label1' and 'Canvas1' are listed. At the bottom of this palette are 'Rename' and 'Delete' buttons. Below the palette is a 'Media' section with an 'Upload File ...' button. The main workspace on the right shows a code block for 'Canvas1'. It starts with an 'initialize global radius to 20' block. This is followed by a 'when Canvas1.TouchDown' event listener. Inside this listener, there is a 'do' block containing a 'call Canvas1.DrawCircle' block. The 'DrawCircle' block has three inputs: 'x' (connected to a 'get x' block), 'y' (connected to a 'get y' block), and 'r' (connected to the 'radius' variable).

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do
    call Canvas1.DrawCircle
      x get x
      y get y
      r radius
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the component palette, which is organized into sections: 'Built-in' (containing Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures), 'Screen1' (containing Label1 and Canvas1), and 'Any component'. At the bottom of the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button. The main workspace on the right shows a code block for the Canvas1 component. The code block is a 'when' block triggered by 'Canvas1' with a 'do' block containing a 'call Canvas1.DrawCircle' block. The 'DrawCircle' block has three inputs: 'x' (set to 'get x'), 'y' (set to 'get y'), and 'r' (set to 'get global radius'). Above the 'DrawCircle' block is an 'initialize global radius to 20' block. The 'when' block also has a 'set global radius to' block that is set to the value of the 'get global radius' block.

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...

```
initialize global radius to 20
when Canvas1
  get global radius
  set global radius to
  do call Canvas1.DrawCircle
    x get x
    y get y
    r
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Control' category, 'Screen1' is expanded, showing 'Label1' and 'Canvas1'. Below the palette are 'Rename' and 'Delete' buttons. At the bottom of the interface is a 'Media' section with an 'Upload File ...' button.

The main workspace on the right shows a code block for 'Canvas1'. The code is as follows:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius
```

App Inventor 2 (AI2) Demo



The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

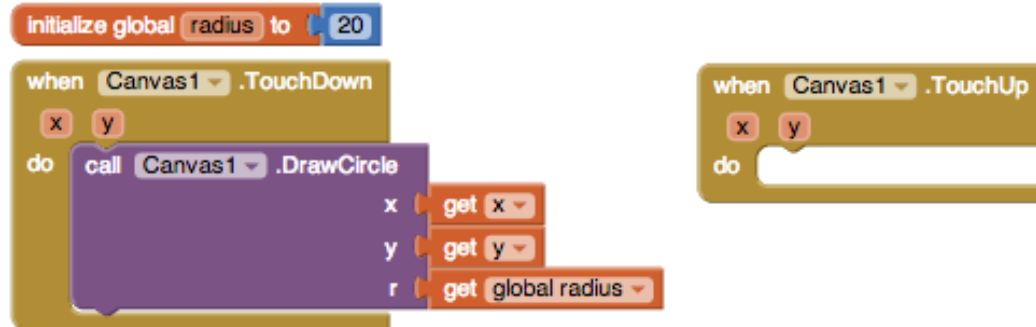
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is a sidebar with a component palette. The 'Built-in' category is expanded, showing sub-categories: Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under 'Screen1', 'Label1' and 'Canvas1' are listed. At the bottom of the sidebar are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows a visual programming script for Canvas1. The script begins with an 'if' block where the condition is 'global radius to 20'. The 'then' block contains a 'Canvas1.TouchDown' event block. This event block has two input fields for 'x' and 'y'. Below the event block is a 'for each' loop: 'for each number from 1 to 5 by 1'. The 'do' block of this loop contains three blocks: 'circle' (with 'x' and 'y' inputs), 'get x' (outputting to 'x'), 'get y' (outputting to 'y'), and 'get global radius' (outputting to 'r').

To the right of the main workspace, a separate block is shown: 'when Canvas1.TouchUp'. This block has two input fields for 'x' and 'y', and a 'do' block.

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Unload File...

initialize global radius to 20

when Canvas1.TouchDown

x y

do

call Canvas1.DrawCircle

x get x

y get y

r get global radius

when Canvas1.TouchUp

x y

do

for each number from 1 to 5 by 1

do

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

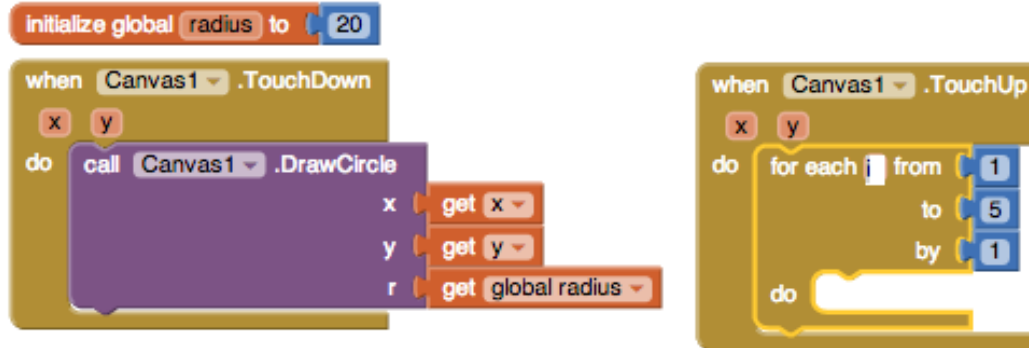
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

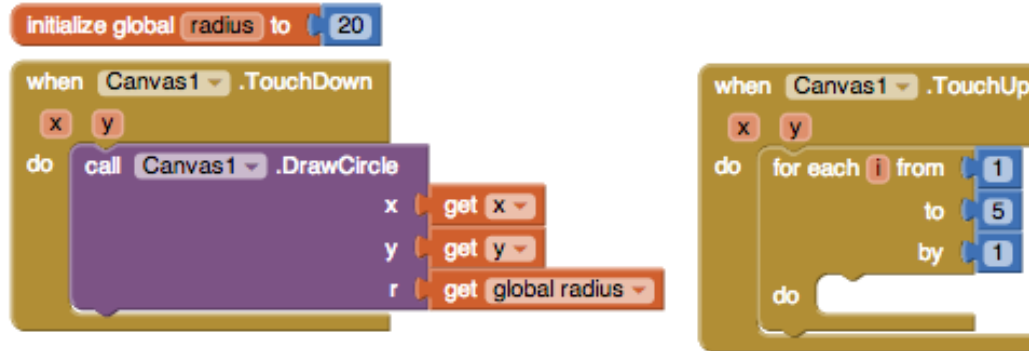
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

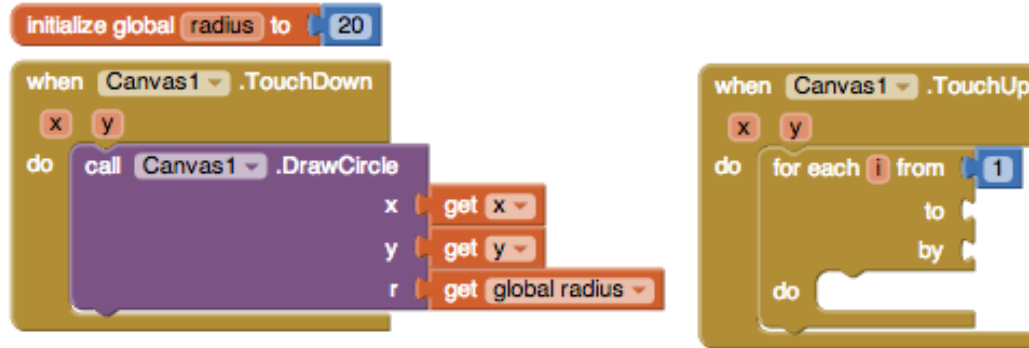
- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...



App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Screen1' section, 'Label1' and 'Canvas1' are listed. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows the code editor for 'Canvas1'. The logic is as follows:

- Initialize global radius to 20**: A block that sets a global variable named 'radius' to the value 20.
- when Canvas1.TouchDown**: A logic block that triggers when the canvas is touched down. It contains:
 - do**: A loop block that contains a **call Canvas1.DrawCircle** block. The arguments for this block are:
 - x**: `get x` (the x-coordinate of the touch)
 - y**: `get y` (the y-coordinate of the touch)
 - r**: `get global radius` (the value of the global radius variable)
- when Canvas1.TouchUp**: A logic block that triggers when the touch is lifted. It contains:
 - do**: A loop block that contains a **for each i from 1 to get y by 2 × get global radius** block. This block iterates from 1 to the value of 'get y' divided by twice the 'global radius'.

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the component palette, which includes a 'Built-in' section with categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Below this are 'Screen1' components (Label1, Canvas1) and an 'Any component' section. At the bottom left are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows the code editor for 'Canvas1'. It contains two event-driven blocks:

- when Canvas1.TouchDown**: This block has two input fields for 'x' and 'y'. The 'do' block contains a 'call Canvas1.DrawCircle' block with three arguments: 'x' (connected to 'get x'), 'y' (connected to 'get y'), and 'r' (connected to 'get global radius').
- when Canvas1.TouchUp**: This block has two input fields for 'x' and 'y'. The 'do' block contains a 'for each i from 1 to get y by 2 x get global radius' loop. Inside the loop is a 'call Canvas1.DrawCircle' block with three arguments: 'x' (connected to 'get x'), 'y' (connected to 'get y'), and 'r' (connected to 'get global radius').

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...

initialize global radius to 20

when Canvas1.TouchDown

x y

do

call Canvas1.DrawCircle

x get x

y get y

r get global radius

when Canvas1.TouchUp

x y

do

for each i from 1 to get y by 2

do

call Canvas1.DrawCircle

x get x

y get y

r global radius

global radius

i

x

✓ y

App Inventor 2 (AI2) Demo

Component Palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do
    call Canvas1.DrawCircle
      x: get x
      y: get y
      r: get global radius

when Canvas1.TouchUp
  x y
  do
    for each i from 1 to get y by 2 x get global radius
    do
      call Canvas1.DrawCircle
        x: get x
        y: get i
        r: get global radius
```

Media Section:

Upload File ...

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under 'Screen1', 'Canvas1' is selected. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows two event-driven code blocks:

- when Canvas1.TouchDown**: This block contains a 'do' loop with a 'call Canvas1.DrawCircle' block. The 'DrawCircle' block has three inputs: 'x' (connected to 'get x'), 'y' (connected to 'get y'), and 'r' (connected to 'get global radius').
- when Canvas1.TouchUp**: This block contains a 'do' loop with a 'for each i from 1 to get y by 1' loop. Inside this loop is a 'call Canvas1.DrawCircle' block. The 'DrawCircle' block has three inputs: 'x' (connected to 'get x'), 'y' (connected to 'get i'), and 'r' (connected to 'get global radius').

A tooltip is visible over the 'y' input of the 'DrawCircle' block in the 'when Canvas1.TouchUp' block, showing the value 'global radius'.

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' component palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under 'Screen1', components 'Label1' and 'Canvas1' are listed. At the bottom of the palette are 'Rename' and 'Delete' buttons. Below the palette is a 'Media' section with an 'Upload File ...' button.

The main workspace shows the code editor for 'Canvas1'. The logic is as follows:

- Initialize global radius to 20**: A block that sets a global variable named 'radius' to the value 20.
- when Canvas1 .TouchDown**: An event listener block that triggers when the canvas is touched down. It contains:
 - do** block with a **call Canvas1 .DrawCircle** block. The parameters for this block are:
 - x**: `get x`
 - y**: `get y`
 - r**: `get global radius`
- when Canvas1 .TouchUp**: An event listener block that triggers when the canvas is touched up. It contains:
 - do** block with a **for each i from 1 to get y by 2 × get global radius** loop. The loop body contains a **call Canvas1 .DrawCircle** block with parameters:
 - x**: `get x`
 - y**: `get i`
 - r**: `get global radius`

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...

to procedure Canvas1.TouchDown

do

Canvas1.DrawCircle

x get x

y get y

r get global radius

when Canvas1.TouchUp

x y

do

for each i from 1 to get y by 2 x get global radius

do

call Canvas1.DrawCircle

x get x

y get i

r get global radius

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' component palette with categories: Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under 'Screen1', 'Canvas1' is selected. Below the palette are 'Rename' and 'Delete' buttons. The 'Media' section at the bottom has an 'Upload File ...' button.

The main workspace shows the following code blocks:

- Initialize global radius to 20** (orange block)
- when Canvas1.TouchDown** (yellow block) with inputs **x** and **y**. Inside the 'do' loop:
 - call Canvas1.DrawCircle** (purple block) with arguments:
 - x**: **get x** (orange block)
 - y**: **get y** (orange block)
 - r**: **get global radius** (orange block)
- to procedure** (purple block) with a **do** loop (empty).
- when Canvas1.TouchUp** (yellow block) with inputs **x** and **y**. Inside the 'do' loop:
 - for each i from 1 to get y by 2 x get global radius** (blue block) loop:
 - do** loop:
 - call Canvas1.DrawCircle** (purple block) with arguments:
 - x**: **get x** (orange block)
 - y**: **get i** (orange block)
 - r**: **get global radius** (orange block)

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, showing the Scratch-like block-based programming environment. The left sidebar contains a 'Built-in' category with 'Procedures' selected, and a 'Screen1' category with 'Canvas1' selected. The main workspace shows a sequence of blocks:

- initialize global radius to 20**
- when Canvas1.TouchDown** block with inputs **x** and **y**.
- draw circle** block with inputs **x**, **y**, and **r**, all set to **get global radius**.
- to procedure** block.

A callout box shows the **inputs** for the **draw circle** block: **input: x**, **y**, and **r**.

The right workspace shows a **when Canvas1.TouchUp** block with a **do** loop containing a **for each i from 1 to 2** loop, which calls **Canvas1.DrawCircle** with inputs **x**, **y**, and **r**, all set to **get global radius**.

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, which is divided into three main sections: a left-hand palette, a central code editor, and a bottom media section.

Left-hand Palette:

- Built-in:** A category containing blocks for Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures.
- Screen1:** A category containing blocks for Label1 and Canvas1.
- Any component:** A category for additional components.
- Buttons:** 'Rename' and 'Delete' buttons are located below the 'Any component' category.
- Media:** A section at the bottom with an 'Upload File ...' button.

Central Code Editor:

The code editor contains the following blocks:

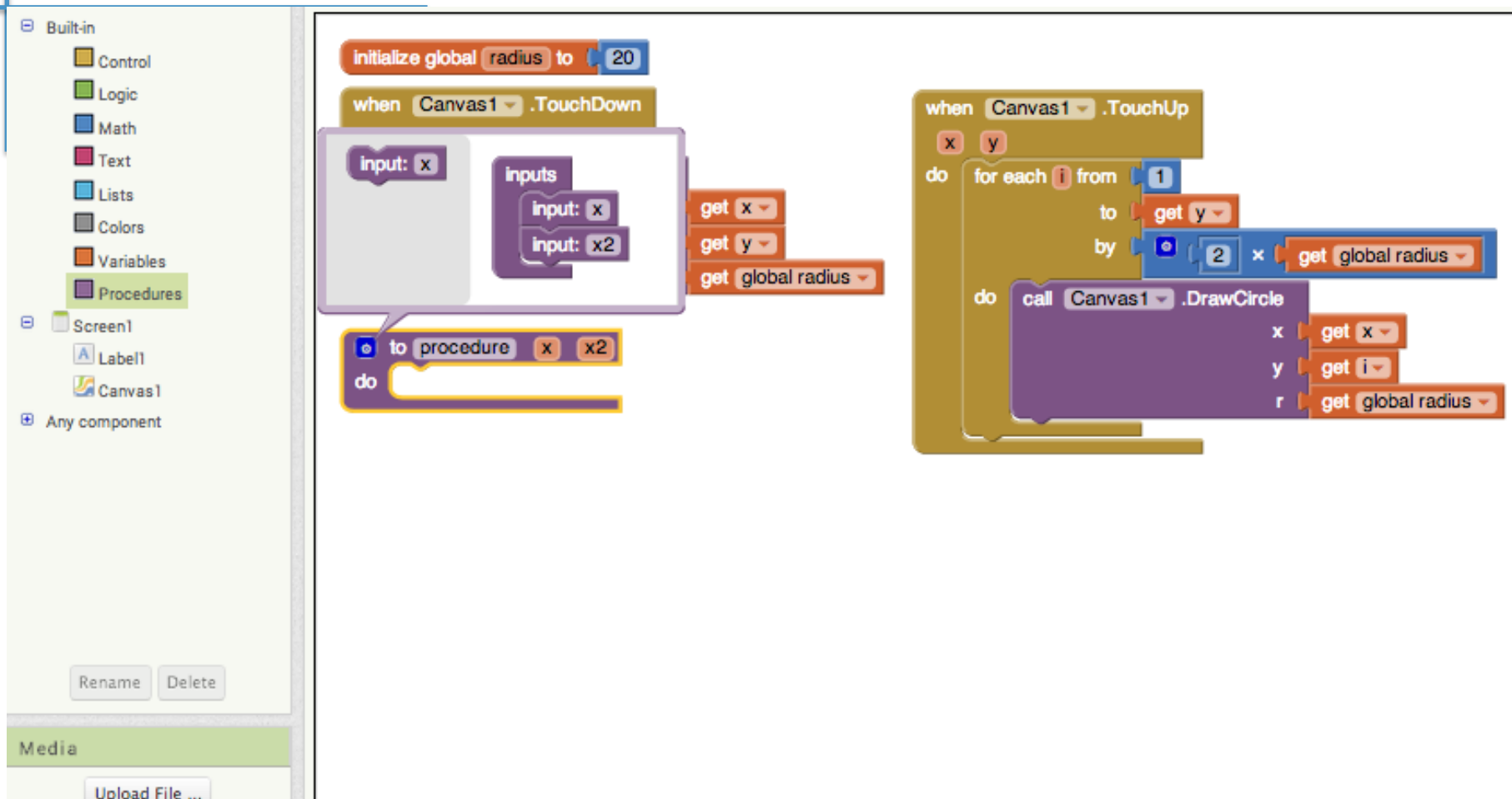
- Initialize global radius to 20:** A block from the 'Variables' category that sets a global variable named 'radius' to the value 20.
- when Canvas1.TouchDown:** A logic block that triggers the following code when the 'Canvas1' component is touched down.
 - Inputs:** A block that provides the 'x' and 'y' coordinates of the touch event.
 - to procedure x:** A block that calls a procedure named 'x'.
 - do:** A block that contains the following code:
 - get x:** A block that retrieves the 'x' coordinate.
 - get y:** A block that retrieves the 'y' coordinate.
 - get global radius:** A block that retrieves the value of the global variable 'radius'.

Right-hand Code Editor:

The code editor contains the following blocks:

- when Canvas1.TouchUp:** A logic block that triggers the following code when the 'Canvas1' component is touched up.
 - do:** A block that contains the following code:
 - for each i from 1 to get y by 2 x get global radius:** A loop block that iterates from 1 to the value of 'get y' by an increment of '2 x get global radius'. The loop variable is 'i'.
 - do:** A block that contains the following code:
 - call Canvas1.DrawCircle:** A block that calls the 'DrawCircle' method of the 'Canvas1' component. It has three inputs: 'x' (from 'get x'), 'y' (from 'get i'), and 'r' (from 'get global radius').

App Inventor 2 (AI2) Demo



Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...

Script:

```
initialize global radius to 20

when Canvas1.TouchDown
do
  to procedure x x2
    inputs
    input: x
    input: x2
    get x
    get y
    get global radius

when Canvas1.TouchUp
x y
do
  for each i from 1 to get y by 2 x get global radius
  do
    call Canvas1.DrawCircle
      x get x
      y get i
      r get global radius
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the component palette, and on the right is the code editor.

Component Palette (Left):

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Buttons: Rename, Delete

Code Editor (Right):

Global Variable: radius (initialized to 20)

when Canvas1.TouchDown

- do
 - call Canvas1.DrawCircle
 - x: get x
 - y: get y
 - r: get global radius

to procedure x x2

- do

when Canvas1.TouchUp

- do
 - for each i from 1 to get y by 2 x get global radius
 - do
 - call Canvas1.DrawCircle
 - x: get x
 - y: get i
 - r: get global radius

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is a sidebar with a component palette. The main area shows the code editor for Canvas1.

Component Palette (Left Sidebar):

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Code Editor (Canvas1):

- Initialize global radius to 20**
- when Canvas1.TouchDown**
 - do
 - call Canvas1.DrawCircle
 - x: get x
 - y: get y
 - r: get global radius
- to drawDots x x2**
 - do
- when Canvas1.TouchUp**
 - do
 - for each i from 1 to get y by 2 x get global radius
 - do
 - call Canvas1.DrawCircle
 - x: get x
 - y: get i
 - r: get global radius

Media Section (Bottom):

- Upload File ...

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Procedures' category, 'Screen1' is expanded, showing 'Label1' and 'Canvas1'. Below the palette are 'Rename' and 'Delete' buttons. At the bottom of the interface is a 'Media' section with an 'Upload File ...' button.

The main workspace shows the code blocks for the 'Canvas1' component:

- Initialize global radius to 20**: A block that sets a global variable named 'radius' to the value 20.
- when Canvas1 .TouchDown**: A logic block that triggers when the canvas is touched down. It contains:
 - do** block with:
 - call Canvas1 .DrawCircle**: A block that calls the 'DrawCircle' method of 'Canvas1'. It has three inputs: 'x' (connected to 'get x'), 'y' (connected to 'get y'), and 'r' (connected to 'get global radius').
- to drawDots xDots x2**: A procedure block with a 'do' block that is currently empty.
- when Canvas1 .TouchUp**: A logic block that triggers when the canvas is touched up. It contains:
 - do** block with:
 - for each i from 1 to get y by 2 x get global radius**: A loop block that iterates from 1 to the value of 'get y', with a step of '2 x get global radius'. The loop variable is 'i'.
 - do** block with:
 - call Canvas1 .DrawCircle**: A block that calls the 'DrawCircle' method of 'Canvas1'. It has three inputs: 'x' (connected to 'get x'), 'y' (connected to 'get i'), and 'r' (connected to 'get global radius').

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under 'Screen1', 'Canvas1' is selected. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows the code editor with the following logic blocks:

- Initialize global radius to 20** (orange block)
- when Canvas1.TouchDown** (yellow block) with a **do** loop containing:
 - call Canvas1.DrawCircle** (purple block) with arguments:
 - x**: **get x** (orange block)
 - y**: **get y** (orange block)
 - r**: **get global radius** (orange block)
- to drawDots** (purple block) with arguments **xDots** and **y**.

On the right, there is another code block:

- when Canvas1.TouchUp** (yellow block) with a **do** loop containing:
 - for each i from 1 to get y by 2 x get global radius** (blue block)
 - do** loop containing:
 - call Canvas1.DrawCircle** (purple block) with arguments:
 - x**: **get x** (orange block)
 - y**: **get i** (orange block)
 - r**: **get global radius** (orange block)

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Procedures' category, 'Screen1' is expanded, showing 'Label1' and 'Canvas1'. Below the palette are 'Rename' and 'Delete' buttons. At the bottom of the left sidebar is a 'Media' section with an 'Upload File ...' button.

The main workspace shows the code for 'Canvas1'. The code is as follows:

```
initialize global radius to 20

when Canvas1.TouchDown
  x
  y
  do
    call Canvas1.DrawCircle
      x: get x
      y: get y
      r: get global radius

  to drawDots xDots y
  do

when Canvas1.TouchUp
  x
  y
  do
    for each i from 1
      to get y
      by 2 * get global radius
    do
      call Canvas1.DrawCircle
        x: get x
        y: get i
        r: get global radius
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Procedures' category, 'Screen1' is expanded, showing 'Label1' and 'Canvas1'. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows a visual programming script for a canvas application:

- Initialize global radius to 20**: A block to set a global variable 'radius' to the value 20.
- when Canvas1 .TouchDown**: A logic block that triggers when the canvas is touched. It contains:
 - x** and **y** input fields to capture touch coordinates.
 - do** block containing a **call Canvas1 .DrawCircle** block with three arguments:
 - x**: **get x** (retrieves the touch x-coordinate).
 - y**: **get y** (retrieves the touch y-coordinate).
 - r**: **get global radius** (retrieves the global radius variable).
- when Canvas1 .TouchUp**: A logic block that triggers when the touch is lifted. It contains an empty **do** block.
- to drawDots xDots y**: A procedure block with two inputs, 'xDots' and 'y'. It contains:
 - do** block containing a **for each i from 1** loop.
 - to**: **get y** (retrieves the 'y' input).
 - by**: **(2 x get global radius)** (calculates the step size for the loop).
 - do** block containing a **call Canvas1 .DrawCircle** block with three arguments:
 - x**: **get x** (retrieves the 'xDots' input).
 - y**: **get i** (retrieves the current loop index).
 - r**: **get global radius** (retrieves the global radius variable).

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is a sidebar with a 'Built-in' category containing blocks for Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Below this is a 'Screen1' section with components 'Label1' and 'Canvas1'. At the bottom of the sidebar are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows a Scratch-like block editor for a canvas application. The blocks are as follows:

- Initialize global radius to 20** (orange block)
- when Canvas1 .TouchDown** (yellow block) with inputs **x** and **y**. Inside the 'do' loop:
 - call Canvas1 .DrawCircle** (purple block) with inputs **x** (from **get x**), **y** (from **get y**), and **r** (from **get global radius**).
- when Canvas1 .TouchUp** (yellow block) with inputs **x** and **y**. Inside the 'do' loop, there is an empty space.
- to drawDots xDots y** (blue block) with inputs **xDots** and **y**. Inside the 'do' loop:
 - for each i from 1 to get y by 2 x get global radius** (blue block). The 'by' input is **2 x get global radius**.
 - do call Canvas1 .DrawCircle** (purple block) with inputs **x** (from **get x**), **y** (from **get**), and **r** (from **get**). A dropdown menu is open for the **y** and **r** inputs, showing options: **global radius**, **i**, **xDots**, and **y**.

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the component palette, and on the right is the code editor.

Component Palette (Left):

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Buttons: Rename, Delete

Media

- Upload File ...

Code Editor (Right):

Initialize global radius to 20

when Canvas1.TouchDown

do

- call Canvas1.DrawCircle
 - x: get x
 - y: get y
 - r: get global radius

when Canvas1.TouchUp

do

-

to drawDots xDots y

do

- for each i from 1 to get y by 2 × get global radius
 - do call Canvas1.DrawCircle
 - x: get xDots
 - y: get i
 - r: get global radius

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Procedures' category, 'Screen1' is expanded, showing 'Label1' and 'Canvas1'. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows a Scratch-like block editor with the following code:

- A 'when Canvas1 TouchDown' event block with a 'do' block containing:
 - A 'to procedure' block with a 'do' block containing:
 - A 'Canvas1 TouchDown' block with inputs for 'x' and 'y'.
- A 'to procedure' block named '.DrawCircle' with a 'result' block and three inputs: 'x' (get x), 'y' (get y), and 'r' (get global radius).
- A 'call drawDots' block with inputs for 'xDots' and 'y'.
- A 'do' block containing:
 - A 'for each i from 1 to get y by 2 x get global radius' loop.
 - A 'do' block inside the loop containing a 'call Canvas1 DrawCircle' block with inputs for 'x' (get xDots), 'y' (get i), and 'r' (get global radius).

On the right side of the workspace, there is a 'when Canvas1 TouchUp' event block with a 'do' block containing a 'x y' block.

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...

Scripts:

- initialize global radius to 20**
- when Canvas1.TouchDown**
 - Inputs: x, y
 - do
 - call Canvas1.DrawCircle
 - x: get x
 - y: get y
 - r: get global radius
- when Canvas1.TouchUp**
 - Inputs: x, y
 - do
 - call drawDots
 - xDots
 - y
- to drawDots xDots y**
 - do
 - for each i from 1 to get y by 2 x get global radius
 - do
 - call Canvas1.DrawCircle
 - x: get xDots
 - y: get i
 - r: get global radius

App Inventor 2 (AI2) Demo

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Colors
- Variables
- Procedures

Screen1

- Label1
- Canvas1

Any component

Rename Delete

Media

Upload File ...

Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do
    call Canvas1.DrawCircle
      x get x
      y get y
      r get global radius

when Canvas1.TouchUp
  x
  do
    call Canvas1.drawDots
      xDots get x
      y set x to y

to drawDots xDots y
  do
    for each i from 1
      to get y
      by 2 x get global radius
    do
      call Canvas1.DrawCircle
        x get xDots
        y get i
        r get global radius
```

App Inventor 2 (AI2) Demo

Left Sidebar:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Main Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius

when Canvas1.TouchUp
  x y
  do call drawDots
    xDots get x
    y y

to drawDots xDots y
  do for each i from 1
    to get y
    by 2 * get global radius
    do call Canvas1.DrawCircle
      x get xDots
      y get i
      r get global radius
```

Buttons: Rename, Delete

Media: Upload File ...

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Screen1' section, 'Label1' and 'Canvas1' are listed. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main area shows the code editor for 'Canvas1'. The code is as follows:

```
Initialize global radius to 20

when Canvas1.TouchDown
  x y
  do
    call Canvas1.DrawCircle
      x get x
      y get y
      r get global radius

when Canvas1.TouchUp
  x y
  do
    call
      get y
      set y to get x
      y

to drawDots xDots y
  do
    for each i from 1
      to get y
      by 2 x get global radius
    do
      call Canvas1.DrawCircle
        x get xDots
        y get i
        r get global radius
```

App Inventor 2 (AI2) Demo

Component Palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Code Editor:

```
Initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius

when Canvas1.TouchUp
  x y
  do call drawDots
    xDots get x
    y get y

to drawDots xDots y
  do for each i from 1
    to get y
    by 2 * get global radius
    do call Canvas1.DrawCircle
      x get xDots
      y get i
      r get global radius
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Screen1' section, 'Label1' and 'Canvas1' are listed. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows the following code blocks:

- Initialize global radius to 20**: A block to set a global variable.
- when Canvas1.TouchDown**: A logic block with two inputs, `x` and `y`. The 'do' section contains a **call Canvas1.DrawCircle** block with arguments: `x` (from `get x`), `y` (from `get y`), and `r` (from `get global radius`).
- when Canvas1.TouchUp**: A logic block with two inputs, `x` and `y`. The 'do' section contains a **call drawDots** block with arguments: `xDots` (from `get x`) and `y` (from `get y`).
- to drawDots xDots y**: A procedure block with two inputs, `xDots` and `y`. The 'do' section contains a **for each i from 1 to get y by 2 x get global radius** loop. Inside the loop is a **call Canvas1.DrawCircle** block with arguments: `x` (from `get xDots`), `y` (from `get i`), and `r` (from `get global radius`).

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, featuring a left-hand sidebar with a component palette and a main workspace for code blocks.

Component Palette (Left Sidebar):

- Built-in:**
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1:**
 - Label1
 - Canvas1
- Any component**

Buttons for "Rename" and "Delete" are visible below the palette.

Media Section:

- Upload File ...

Code Blocks (Main Workspace):

- Global Variable Initialization:**
 - `initialize global name to 20`
- Canvas1 TouchDown Event:**
 - `get x`
 - `set to canvas1 .DrawCircle`
 - `initialize local name to` (in a loop)
 - `initialize local name to` (in a loop)
- Canvas1 TouchUp Event:**
 - `when Canvas1 .TouchUp` (with inputs `x` and `y`)
 - `do` block containing:
 - `call drawDots` (with inputs `xDots` and `y`)
 - `get x`
 - `get y`
- Loop:**
 - `for each i from 1`
 - `to get y`
 - `by 2 x get global radius`
 - `do` block containing:
 - `call Canvas1 .DrawCircle` (with inputs `x`, `y`, and `r`)
 - `get xDots`
 - `get i`
 - `get global radius`

App Inventor 2 (AI2) Demo

Component Palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Buttons: Rename, Delete

Media Section: Upload File ...

Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius

when Canvas1.TouchUp
  x y
  do call drawDots
    xDots get x
    y get y

to drawDots xDots y
  do
    initialize local name to
    in

for each i from 1
  to get y
  by 2 x get global radius
  do call Canvas1.DrawCircle
    x get xDots
    y get i
    r get global radius
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is a sidebar with a 'Built-in' category containing blocks for Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Below this is a 'Screen1' category with 'Label1' and 'Canvas1'. At the bottom of the sidebar are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace shows four code blocks:

- Initialize global radius to 20**: A block to set a global variable named 'radius' to the value 20.
- when Canvas1.TouchDown**: A logic block that triggers when the canvas is touched down. It has input fields for 'x' and 'y'. The 'do' section contains a 'call Canvas1.DrawCircle' block with arguments: 'x' (connected to 'get x'), 'y' (connected to 'get y'), and 'r' (connected to 'get global radius').
- when Canvas1.TouchUp**: A logic block that triggers when the canvas is touched up. It has input fields for 'x' and 'y'. The 'do' section contains a 'call drawDots' block with arguments: 'xDots' (connected to 'get x') and 'y' (connected to 'get y').
- to drawDots xDots y**: A procedure block with two arguments, 'xDots' and 'y'. The 'do' section contains an 'initialize local numDots to' block followed by an empty 'in' loop structure.

Below the 'when Canvas1.TouchUp' block is a 'for each i from 1 to get y by 2 x get global radius' loop. The 'do' section of this loop contains a 'call Canvas1.DrawCircle' block with arguments: 'x' (connected to 'get xDots'), 'y' (connected to 'get i'), and 'r' (connected to 'get global radius').

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, showing the code for a drawing application. The interface is divided into three main sections: a left sidebar, a top code area, and a bottom media area.

Left Sidebar:

- Built-in:** Control, Logic, Math, Text, Lists, Colors, Variables, Procedures.
- Screen1:** Label1, Canvas1.
- Any component:** (empty)

Top Code Area:

- Initialize global radius to 20**
- when Canvas1.TouchDown**
 - do
 - call Canvas1.DrawCircle
 - x: get x
 - y: get y
 - r: get global radius
- when Canvas1.TouchUp**
 - do
 - call drawDots
 - xDots: get x
 - y: get y

Bottom Code Area:

- to drawDots xDots y**
 - do
 - initialize local numDots to 0
 - in
- for each i from 1 to get y by 2 x get global radius**
 - do
 - call Canvas1.DrawCircle
 - x: get xDots
 - y: get i
 - r: get global radius

Media Area:

- Upload File ...

App Inventor 2 (AI2) Demo

Component Palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius

when Canvas1.TouchUp
  x y
  do call drawDots
    xDots get x
    y get y

to drawDots xDots y
  do
    initialize local numDots to 0
    in for each i from 1
      to get y
      by 2 x get global radius
      do call Canvas1.DrawCircle
        x get xDots
        y get i
        r get global radius
```

App Inventor 2 (AI2) Demo

Component Palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius

when Canvas1.TouchUp
  x y
  do call drawDots
    xDots get x
    y get y

to drawDots xDots y
  do
    initialize local numDots to 20
    in for each i from 1 to 20 by 2
      do call Canvas1.DrawCircle
        x get xDots
        y get i
        r get global radius
```

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is the 'Built-in' components palette, which includes categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Under the 'Screen1' section, 'Label1' and 'Canvas1' are listed. Below the palette are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File...' button.

The main workspace shows the following code blocks:

- Initialize global radius to 20**
- when Canvas1 .TouchDown**
 - do
 - call Canvas1 .DrawCircle
 - x: get x
 - y: get y
 - r: get global radius
- when Canvas1 .TouchUp**
 - do
 - call drawDots
 - xDots: get x
 - y: get y
- to drawDots xDots y**
 - do
 - initialize local numDots to 0
 - in for each i from 1 to get y by 2 x get global radius
 - do call Canvas1 .DrawCircle
 - x: get xDots
 - y: get i
 - r: get global radius
 - set numDots to

App Inventor 2 (AI2) Demo

Component Palette:

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Label1
 - Canvas1
- Any component

Code Editor:

```
initialize global radius to 20

when Canvas1.TouchDown
  x y
  do call Canvas1.DrawCircle
    x get x
    y get y
    r get global radius

when Canvas1.TouchUp
  x y
  do call drawDots
    xDots get x
    y get y

to drawDots xDots y
  do
    initialize local numDots to 0
    in for each i from 1 to get y by 2 x get global radius
      do call Canvas1.DrawCircle
        x get xDots
        y get i
        r get global radius
      set numDots to 1 + get numDots
```


App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface. On the left is a sidebar with a 'Built-in' components list (Control, Logic, Math, Text, Lists, Colors, Variables, Procedures) and a project-specific list for 'Screen1' (Label1, Canvas1). Below these are 'Rename' and 'Delete' buttons, and a 'Media' section with an 'Upload File ...' button.

The main workspace contains the following code blocks:

- Initialize global radius to 20**
- when Canvas1 .TouchDown**
 - do
 - call Canvas1 .DrawCircle
 - x: get x
 - y: get y
 - r: get global radius
- when Canvas1 .TouchUp**
 - do
 - call drawDots
 - xDots: get x
 - y: get y
- to drawDots xDots y**
 - do
 - initialize local numDots to 0
 - in for each i from 1 to get y by 2 x get global radius
 - do
 - call Canvas1 .DrawCircle
 - x: get xDots
 - y: get i
 - r: get global radius
 - set numDots to 1 + get numDots
 - set Label1 .Text to get numDots

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, which is divided into three main sections: a left-hand palette, a central block editor, and a bottom media section.

Left-hand Palette:

- Built-in:** A category containing blocks for Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures.
- Screen1:** A category containing blocks for Label1 and Canvas1.
- Any component:** A category for blocks that can be used on any component.

Central Block Editor:

- Initialize global radius to 20:** A block that sets a global variable named 'radius' to the value 20.
- when Canvas1 .TouchDown:** A logic block that triggers an event when the canvas is touched down. It contains a 'do' block with the following steps:
 - call Canvas1 .DrawCircle:** A block that calls the 'DrawCircle' method of the Canvas1 component. It has three inputs: 'x' (from a 'get x' block), 'y' (from a 'get y' block), and 'r' (from a 'get global radius' block).
- when Canvas1 .TouchUp:** A logic block that triggers an event when the canvas is touched up. It contains a 'do' block with the following steps:
 - call drawDots:** A block that calls a procedure named 'drawDots'. It has two inputs: 'xDots' (from a 'get x' block) and 'y' (from a 'get y' block).
- to drawDots xDots y:** A procedure block that defines the 'drawDots' procedure. It contains the following steps:
 - Initialize local numDots to 0:** A block that sets a local variable named 'numDots' to the value 0.
 - in for each i from 1 to get y by 2 x get global radius:** A loop block that iterates over a range of values. The range starts at 1, ends at 'get y', and increments by '2 x get global radius'.
 - do call Canvas1 .DrawCircle:** A block that calls the 'DrawCircle' method of the Canvas1 component. It has three inputs: 'x' (from a 'get xDots' block), 'y' (from a 'get i' block), and 'r' (from a 'get global radius' block).
 - set numDots to 1 + get numDots:** A block that increments the 'numDots' variable by 1.
 - set Label1 .Text to get numDots:** A block that sets the text of the Label1 component to the value of 'numDots'.

Bottom Media Section:

- Media:** A section for managing media files.
- Upload File ...:** A button to upload a file to the application.

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, showing the visual programming blocks for a drawing application. The interface is divided into three main sections: the left sidebar, the top workspace, and the bottom media section.

Left Sidebar:

- Built-in:** Control, Logic, Math, Text, Lists, Colors, Variables, Procedures.
- Screen1:** Label1, Canvas1.
- Any component:** (empty)

Top Workspace:

- Initialize global radius to 20** (Variables block).
- when Canvas1.TouchDown** (Logic block):
 - do **call Canvas1.DrawCircle** (Procedures block):
 - x: **get x** (Text block)
 - y: **get y** (Text block)
 - r: **get global radius** (Text block)
- when Canvas1.TouchUp** (Logic block):
 - do **call drawDots** (Procedures block):
 - xDots: **get x** (Text block)
 - y: **get y** (Text block)

Bottom Workspace:

- to drawDots xDots y** (Procedures block):
 - do **Initialize local numDots to 0** (Variables block).
 - in for each i from 1 to get y by 2 x get global radius** (Loop block):
 - do **call Canvas1.DrawCircle** (Procedures block):
 - x: **get xDots** (Text block)
 - y: **get i** (Text block)
 - r: **get global radius** (Text block)
 - set numDots to 1 + get numDots** (Variables block).
 - set Label1.Text to get numDots** (Text block).

Media Section:

- Upload File ...** (Text block)

App Inventor 2 (AI2) Demo

The screenshot displays the App Inventor 2 (AI2) interface, showing the visual programming blocks for a drawing application. The interface is divided into three main sections: a left sidebar, a central workspace, and a bottom media section.

Left Sidebar:

- Built-in:** Control, Logic, Math, Text, Lists, Colors, Variables, Procedures.
- Screen1:** Label1, Canvas1.
- Any component:** (empty)

Central Workspace:

- Initialize global radius to 20** (orange block).
- when Canvas1 .TouchDown** (yellow block):
 - do:
 - call Canvas1 .DrawCircle
 - x: get x (orange block)
 - y: get y (orange block)
 - r: get global radius (orange block)
- when Canvas1 .TouchUp** (yellow block):
 - do:
 - call drawDots
 - i2: get x (orange block)
 - y: get y (orange block)

drawDots Procedure:

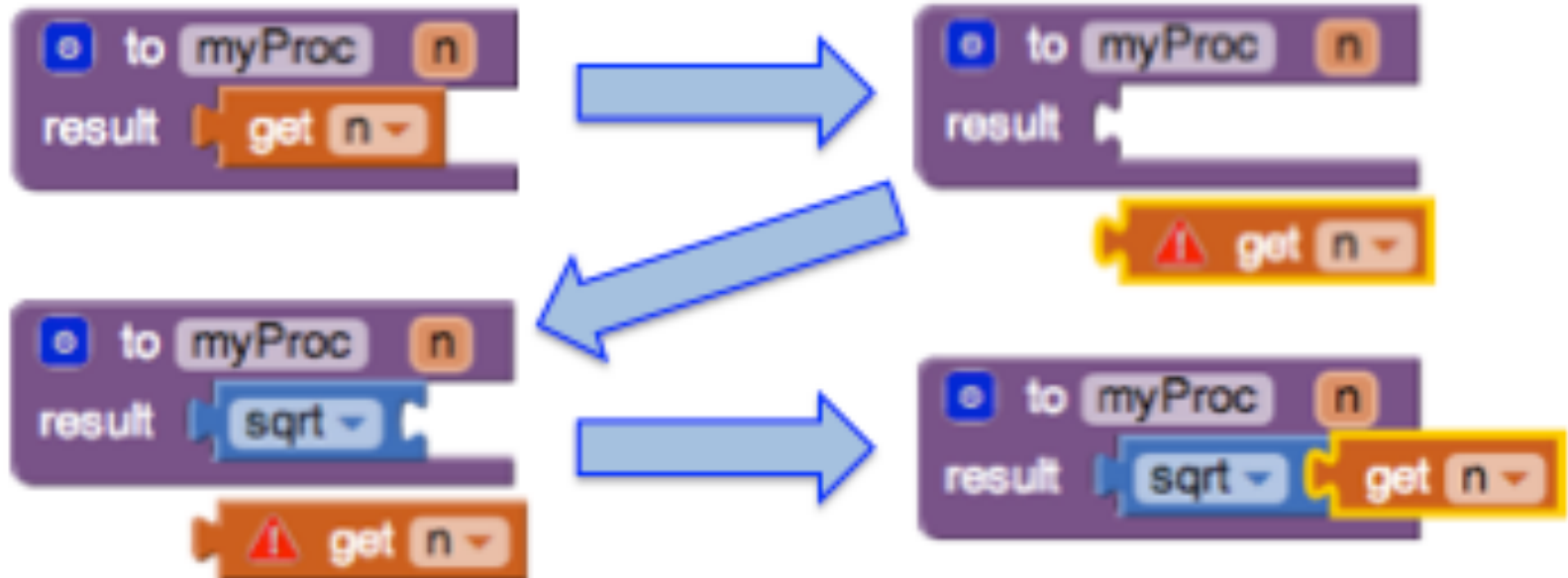
- to drawDots i2 y** (blue block):
 - do:
 - initialize local numDots to 0** (orange block)
 - in for each i from 1 to get y by 2 x get global radius** (blue block):
 - do:
 - call Canvas1 .DrawCircle
 - x: get i2 (orange block)
 - y: get i (orange block)
 - r: get global radius (orange block)
 - set numDots to 1 + get numDots (orange block)
 - set Label1 . Text to get numDots (orange block)

Bottom Media Section:

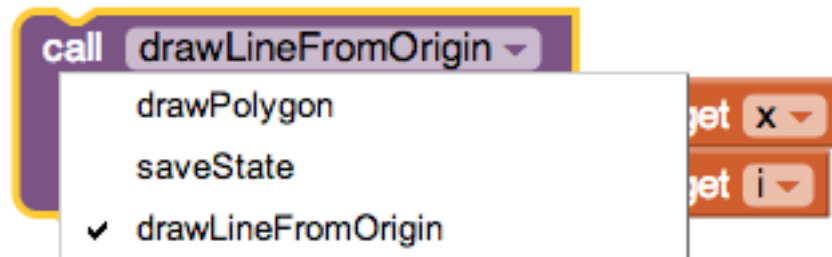
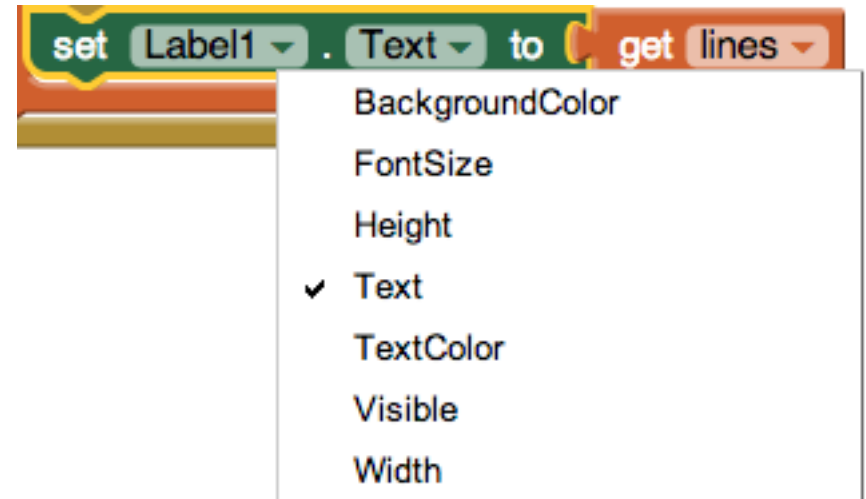
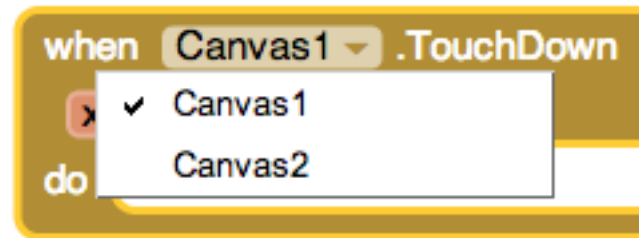
- Media
- Upload File ...

Flagging Unbound Variables

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:



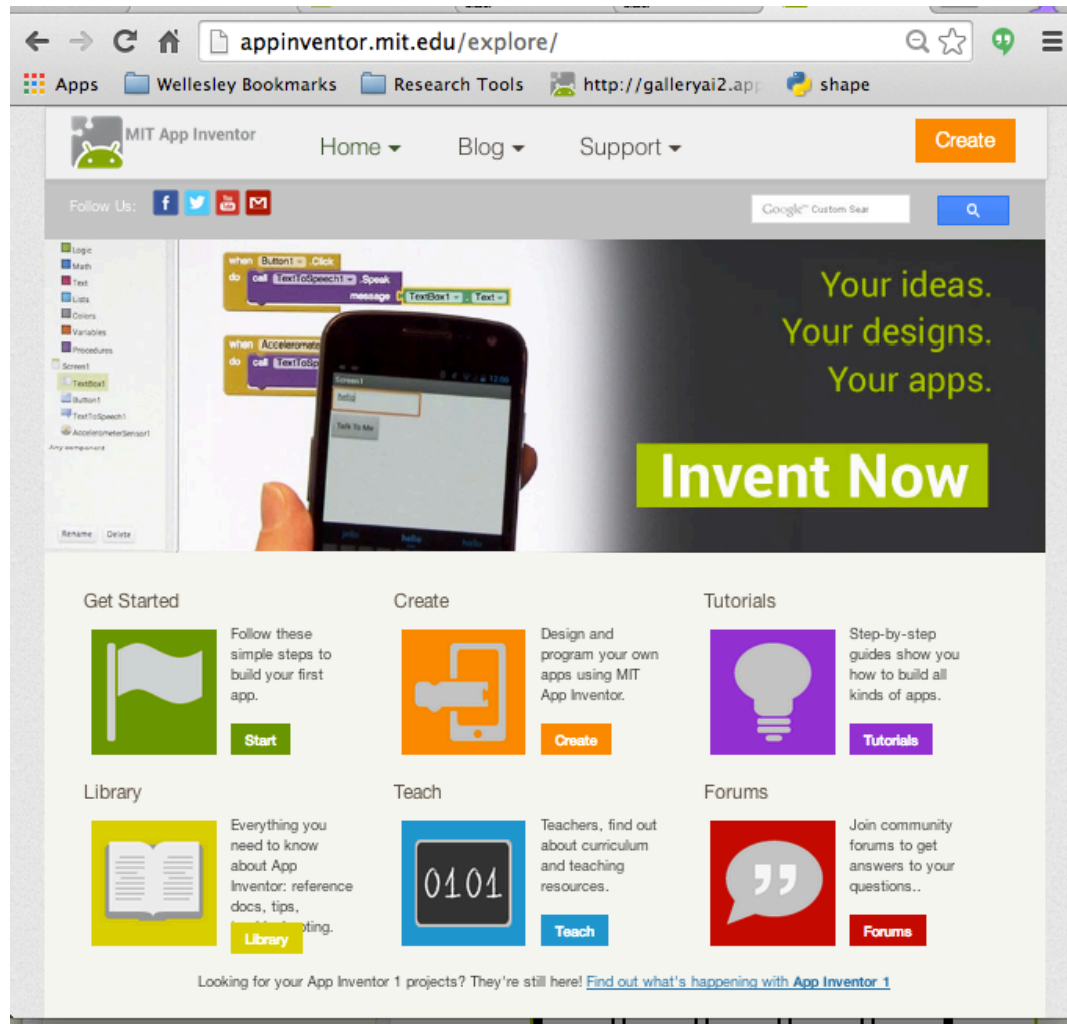
Reducing Viscosity with Drop-Downs



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.

Thank You! Questions?



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.