

Events-First Programming in App Inventor



Lyn Turbak
Wellesley College



Mark Sherman and Fred Martin
UMass Lowell



David Wolber
University of San Francisco



Shaileen Crawford Pokress
MIT Media Lab

CCSCNE, Providence College, April 25, 2014

Talk Overview

- Event-Based Programming in App Inventor 2
- Surprises in the App Inventor Event Model
- Event-Based Thinking with App Inventor 2

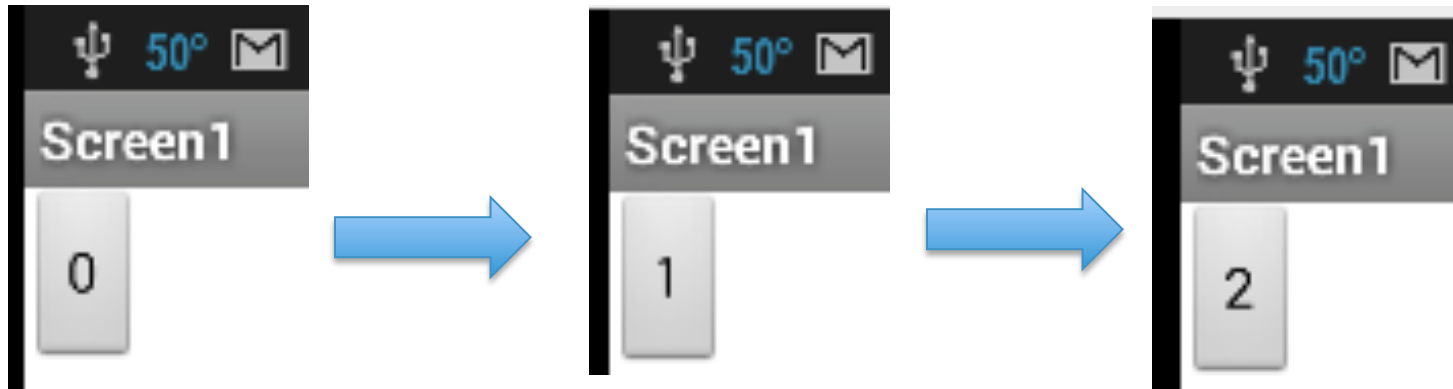
Talk Overview

- Event-Based Programming in App Inventor 2
- Surprises in the App Inventor Event Model
- Event-Based Thinking with App Inventor 2

Simple Events: CountButton

```
when Screen1.Initialize  
do set Button1.Text to 0
```

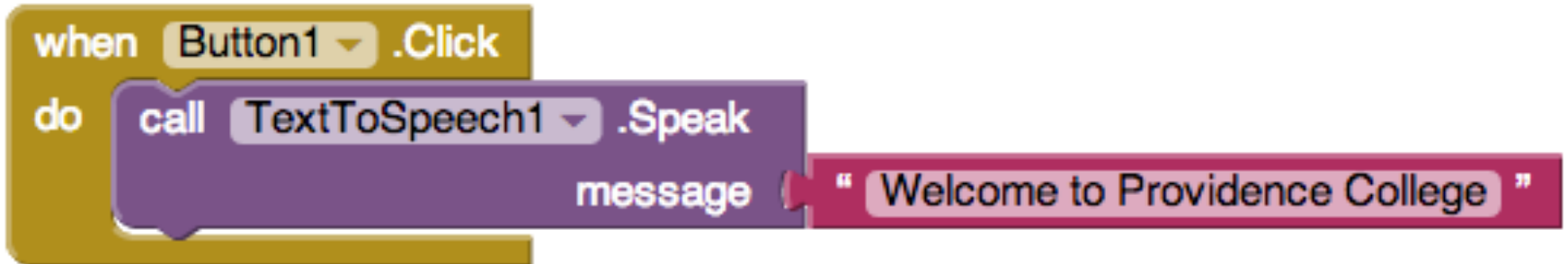
```
when Button1.Click  
do set Button1.Text to 1 + Button1.Text
```



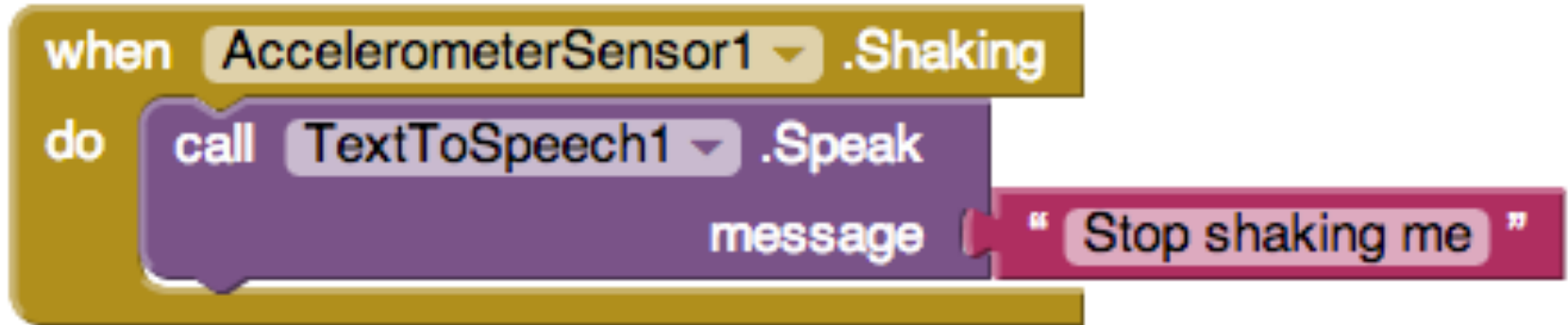
CountButton Example in Java

```
// Assume counter, counterLabel, and incrementButton
// are instance variables
counter = 0;
countButton = new JButton("Increment");
countButton.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            counter++;
            countButton.setText(Integer.toString(counter));
        }
    }
);
```

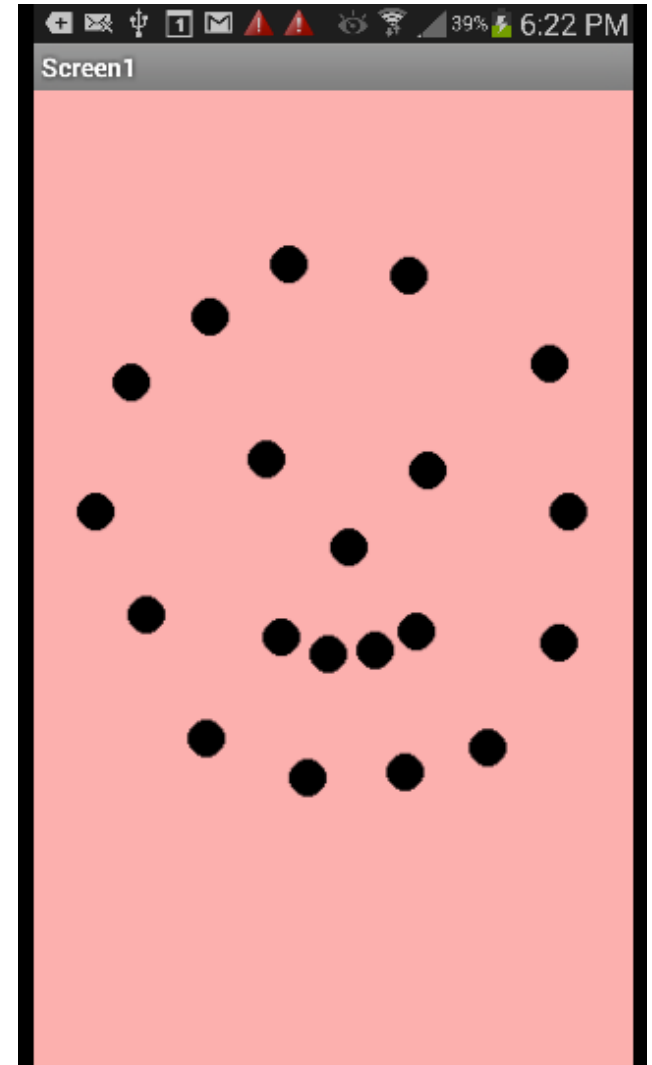
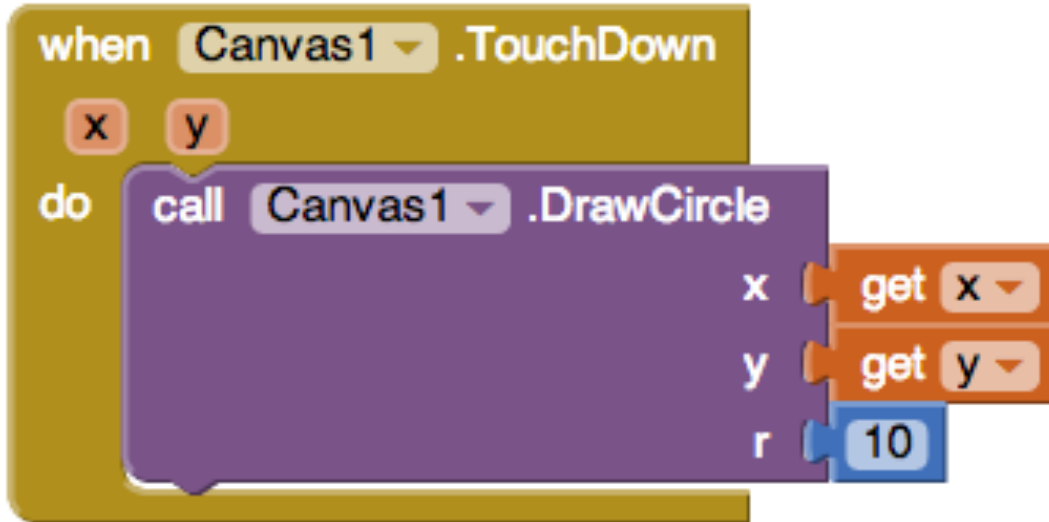
Simple Events: Talk To Me



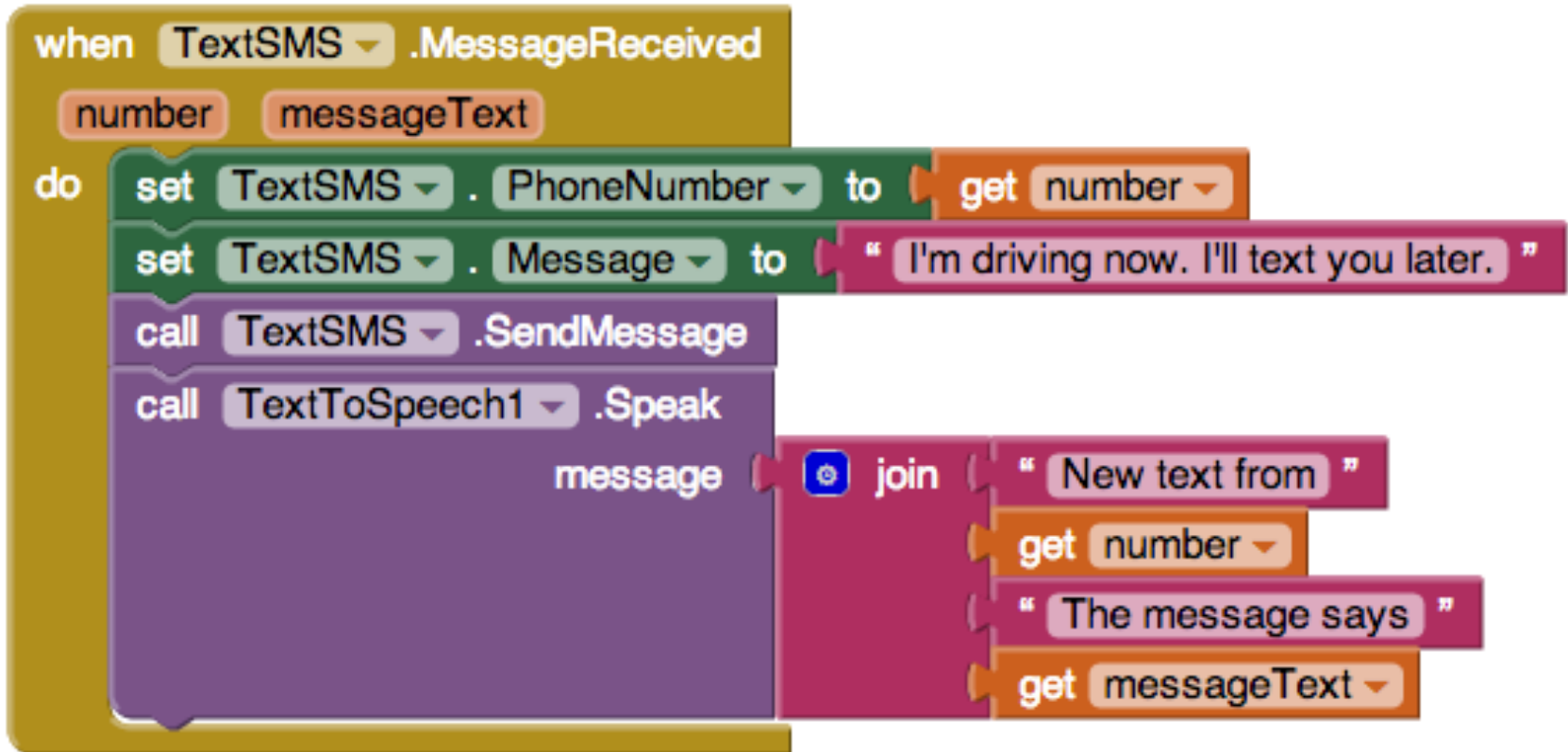
Simple Events: Stop Shaking Me



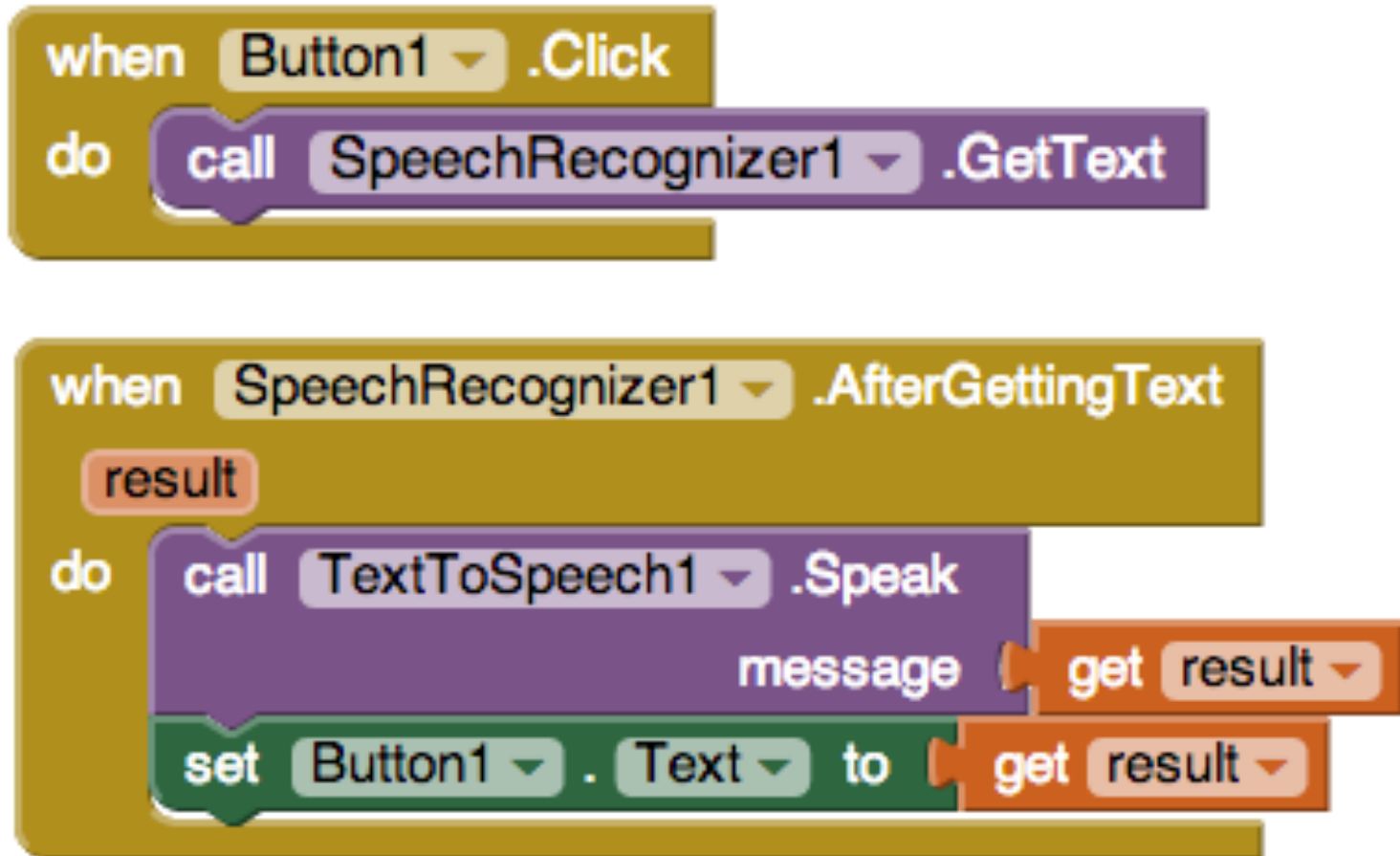
Event Parameters: Dots on Canvas



Event Parameters: No Texting While Driving



Callback Events: Speech to Text



Callback Events: Load Web Page

```
when LoadPageButton .Click
do
  set Web1 . Url to "http://appinventor.mit.edu"
  call Web1 .Get
```

```
when Web1 .GotText
  url responseCode responseType responseContent
do
  set TextBox1 . Text to get responseContent
```

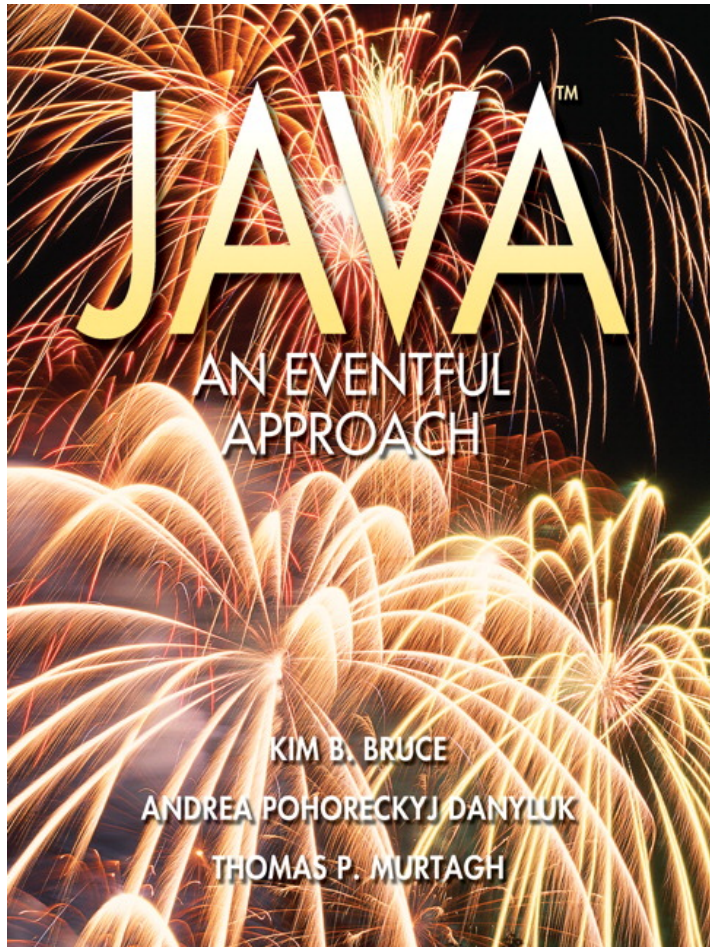
Web Example in JQuery JavaScript

```
$.get("http://appinventor.mit.edu",  
      function (page) { alert(page); }  
    );
```

Events First!

- Interactions with mobile apps are inherently event based.
- Want an event-based programming model that matches the event-based user model.
- Accessible to novices.
- Powerful paradigm for professionals.
- App Inventor programs are just a collection of event handlers.

Java: An Eventful Approach



- “We feel it is clearly more appropriate to adopt programming tools that reflect natural human models of the task undertaken than to adjust the thinking patterns of our student to fit the limitations of the programming techniques we present.”
- Java `objectdraw` library allowed event-based programming from week 1.
- Including event-based techniques enhanced (and did not displace) other topics.
- Students thrilled by real-world interactivity of EBP.

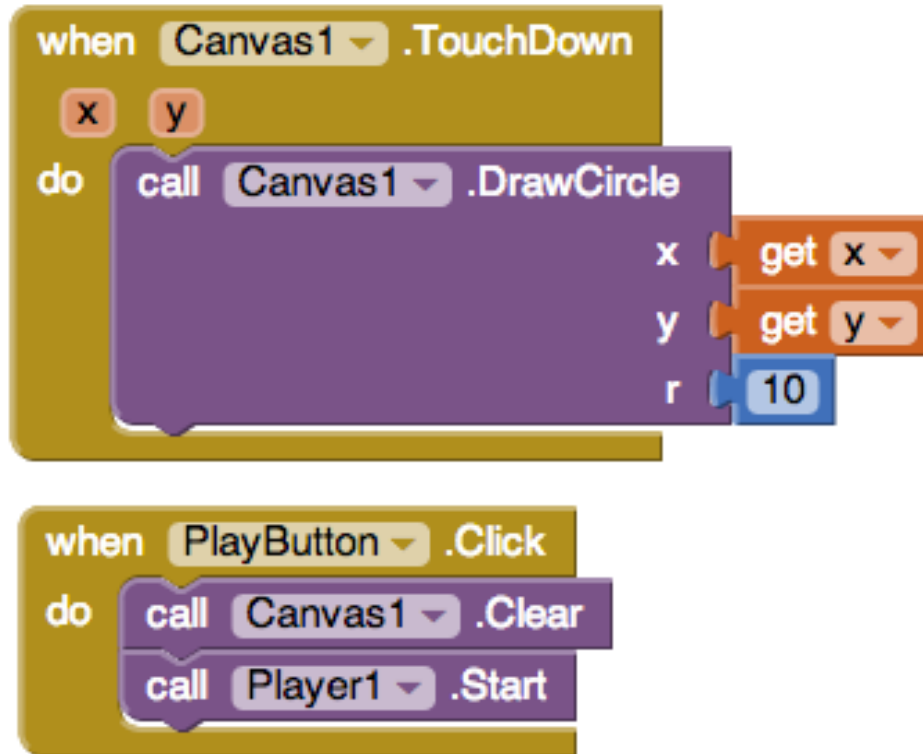
Talk Overview

- Event-Based Programming in App Inventor 2
- **Surprises in the App Inventor Event Model**
- Event-Based Thinking with App Inventor 2

Events in the App Inventor Notional Machine

- Only one event handler can be executing at any given time.
- Other events that occur while an event handler is executing are queued and handled later, in order.
- Any GUI changes during an event handler are not displayed until the event has completed.
- Certain system actions (playing a sound file, initiating a web request, etc.) are executed in a thread separate from the current event handler.
- Playing a sound on a Player component first terminates any sound currently playing *except* when the source file has not been reset, in which case the new play request is ignored if the sound is already playing.

Can dots be drawn while the sound plays?



Answer: **YES.**
The sound plays in a different thread, so the PlayButton.Click event completes quickly, allowing Canvas events to run.

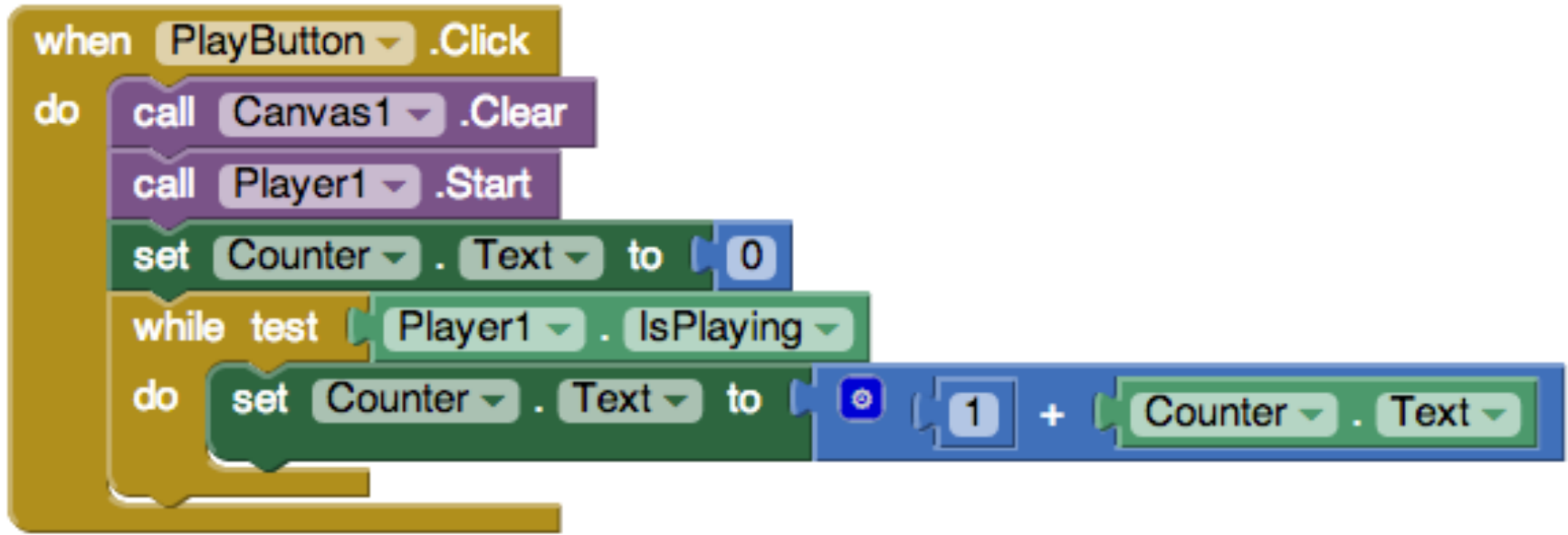
Can dots be drawn while the sound plays?

```
when Canvas1 .TouchDown
  x y
  do call Canvas1 .DrawCircle
    x get x
    y get y
    r 10
```

```
when PlayButton .Click
  do call Canvas1 .Clear
    call Player1 .Start
    set Counter . Text to 0
    while test Player1 . IsPlaying
      do set Counter . Text to 1 + Counter . Text
```

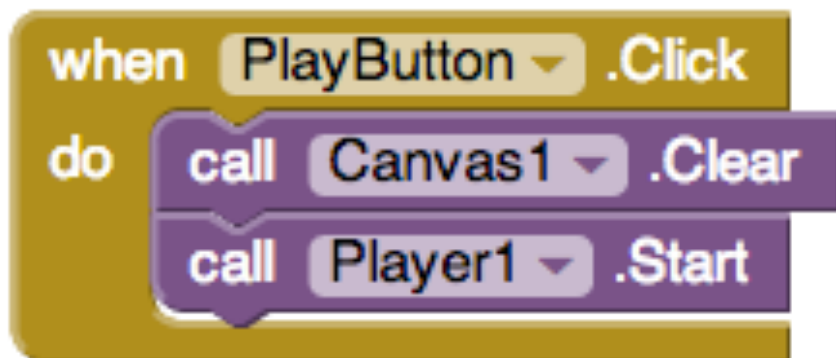
Answer: **NO.**
Although the sound plays in a different thread, the while loop holds control in the PlayButton.Click handler until the sound is done playing. Any other events are queued during this time.

Can we see the counter increment while the sound plays?



Answer: **NO**. The GUI is not updated until the event handler finishes executing, when the counter reaches its final value.

Suppose PlayButton is clicked to play a sound.
What happens if it's clicked again while
the first sound is playing?



Answer: **NOTHING.**
As before, the first PlayButton.Click event completes quickly. When the second event executes, the second attempt to play a sound is ignored because the same one is already playing.

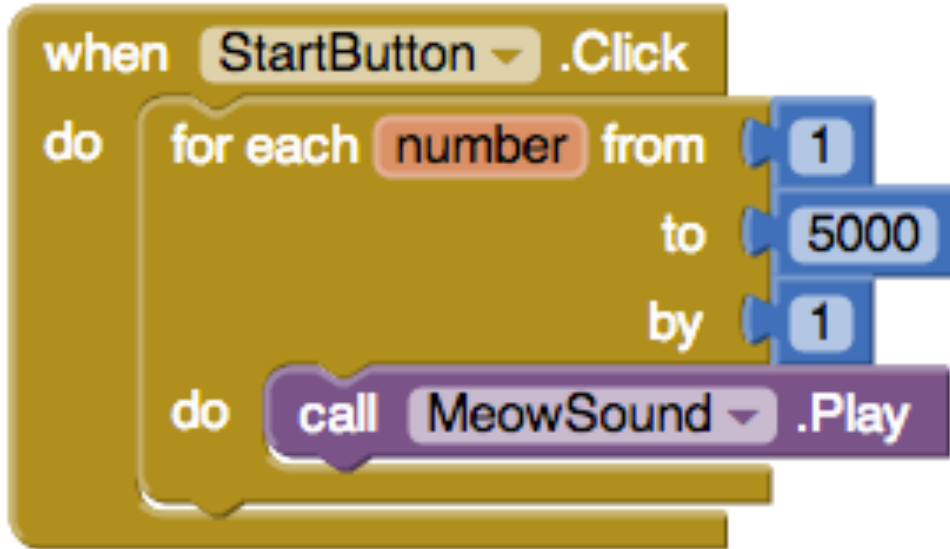
What happens when PlayAllButton is clicked?

```
when PlayAllButton .Click
do
  for each index from 1
    to length of list list
    by 1
  do
    set PlayAllButton . Text to get index
    set Player1 . Source to select list item list
    call Player1 .Start
  do
    set PlayAllButton . Text to " Play All "
```

The image shows a Scratch script for a button click event. The script starts with a 'when PlayAllButton .Click' block. Inside a 'do' block, there is a 'for each' loop. The loop iterates over 'index' from 1 to the 'length of list list' (where 'list' is 'get global recordings') by 1. Inside the loop, there are three blocks: 'set PlayAllButton . Text to get index', 'set Player1 . Source to select list item list' (where 'list' is 'get global recordings' and 'index' is 'get index'), and 'call Player1 .Start'. After the loop, there is a final 'set PlayAllButton . Text to " Play All "' block.

Answer: Only the last recording is played because each Player1.Start terminates the previous recording. Also, the button text does not display any of the intermediate indices.

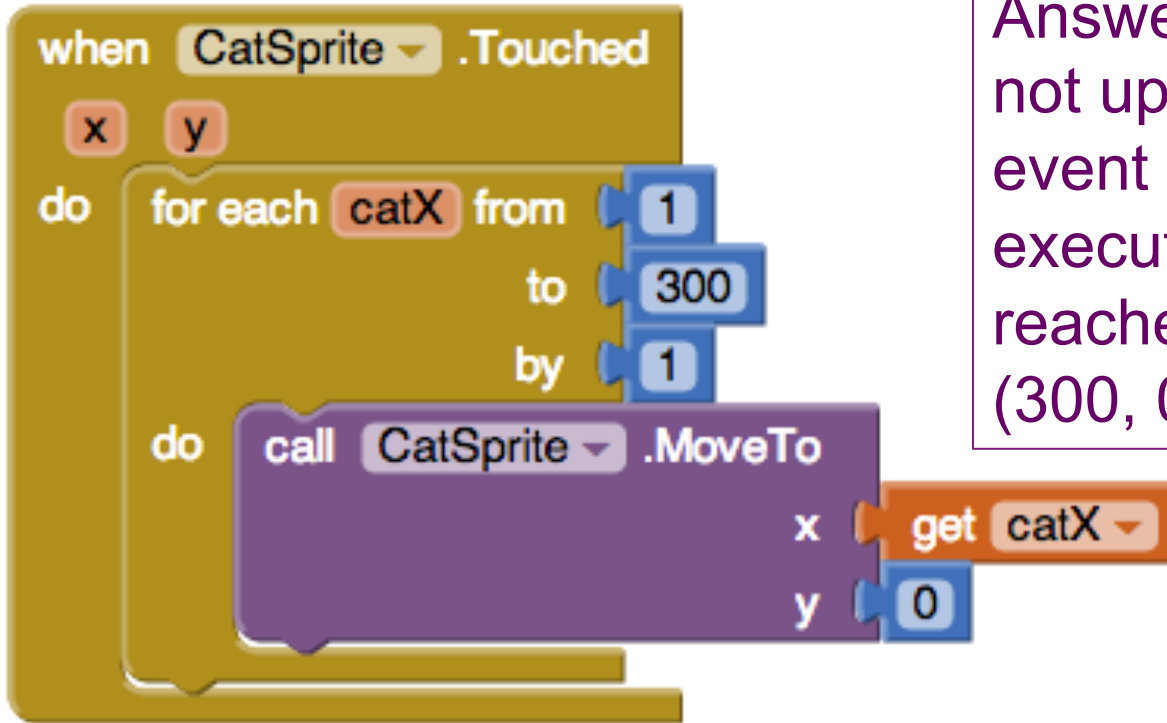
How many times is MeowSound played?



Answer: **Hard to predict, but way less than 5000 times.** The for loop executes quickly, and most requests to play MeowSound are ignored because the same sound is already playing.

In practice, the answer is about a dozen times.

Do we see the cat walk across the screen?



Answer: **NO**. The GUI is not updated until the event handler finishes executing, when the cat reaches its final position, (300, 0).

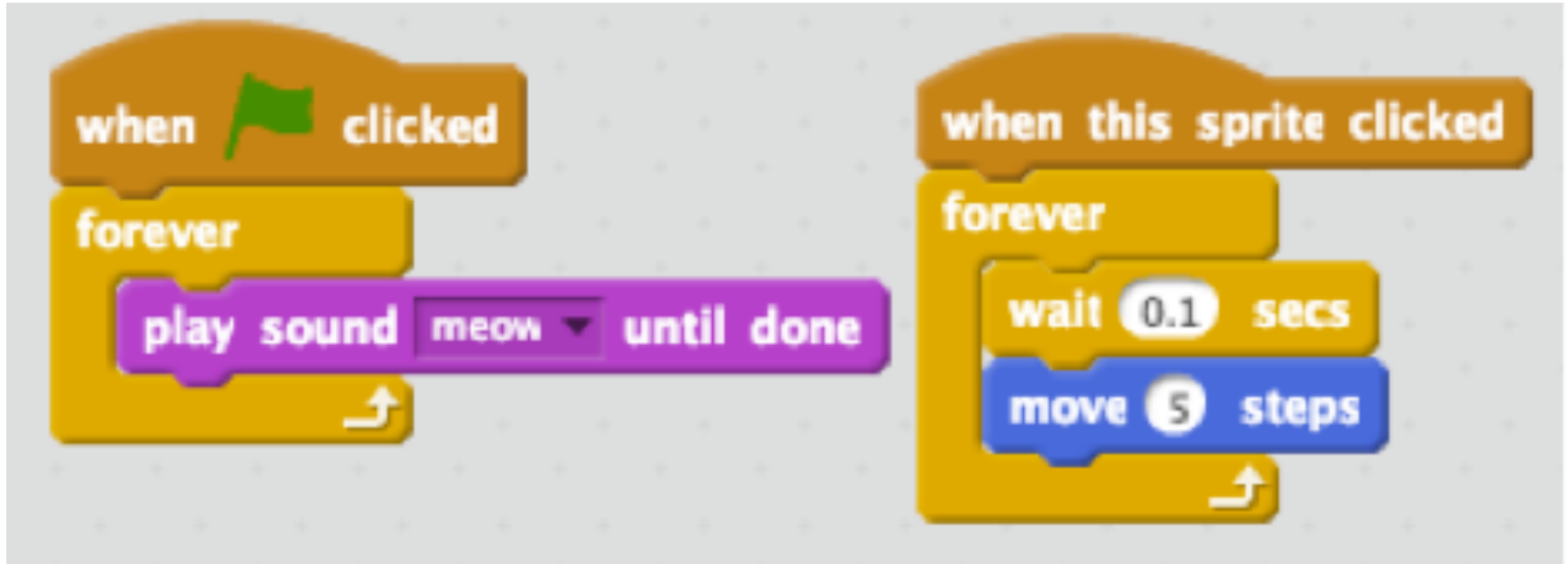
What happens if we click Start and touch the cat?

```
when StartButton .Click
do
  for each number from 1 to 5000 by 1
  do
    call MeowSound .Play
```

```
when CatSprite .Touched
  x y
do
  for each catX from 1 to 300 by 1
  do
    call CatSprite .MoveTo
      x get catX
      y 0
```

Answer: MeowSound is played about a dozen times, during which the cat touch event is queued. After the sound playing completes, the cat jumps to position (300, 0).

Event Handler Interleaving in Scratch



Talk Overview

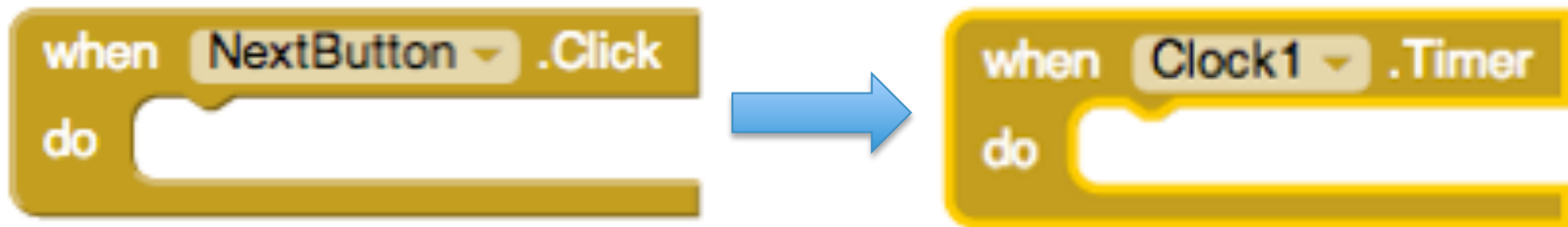
- Event-Based Programming in App Inventor 2
- Surprises in the App Inventor Event Model
- **Event-Based Thinking with App Inventor 2**

Global State Machines: Manually Advancing Slideshow

The image displays a sequence of Scratch code blocks for a manually advancing slideshow. The code is organized into three main sections:

- Initialization:** An orange block "initialize global index to 1" and another orange block "initialize global pictures to" followed by a blue "make a list" block containing three purple blocks: "cat.jpg", "dog.jpg", and "rabbit.jpg".
- Screen Initialization:** A yellow "when Screen1.Initialize" block containing a "do" loop with:
 - An orange "set global index to 1" block.
 - A green "set Slide . Picture to" block connected to a blue "select list item list" block. The "list" parameter is "get global pictures" and the "index" parameter is "1".
- Next Button Click:** A yellow "when NextButton.Click" block containing a "do" loop with:
 - An "if" block: "if" (checked) followed by "get global index" = "length of list list" (from "get global pictures").
 - A "then" block: "set global index to 1".
 - An "else" block: "set global index to" followed by "get global index" + "1".
 - A green "set Slide . Picture to" block connected to a blue "select list item list" block. The "list" parameter is "get global pictures" and the "index" parameter is "get global index".

Timers: Automatically Advancing Slideshow



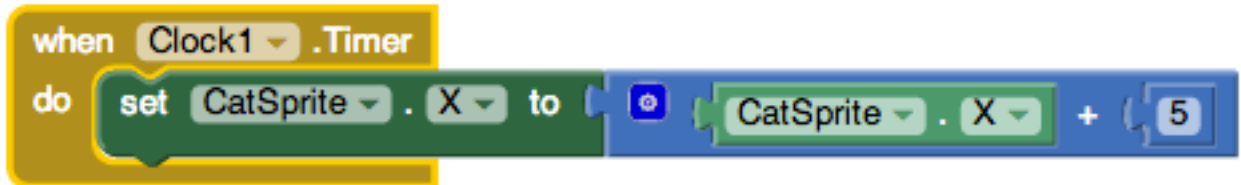
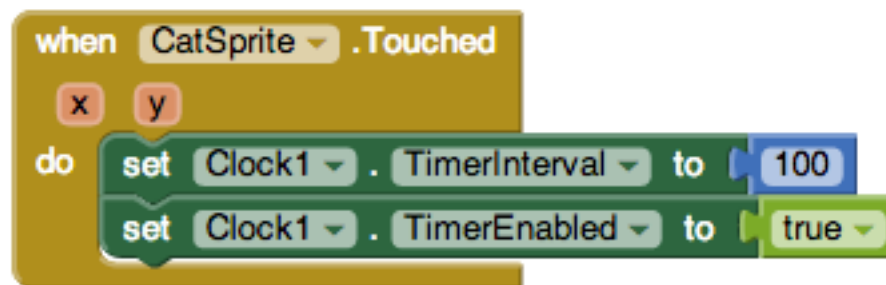
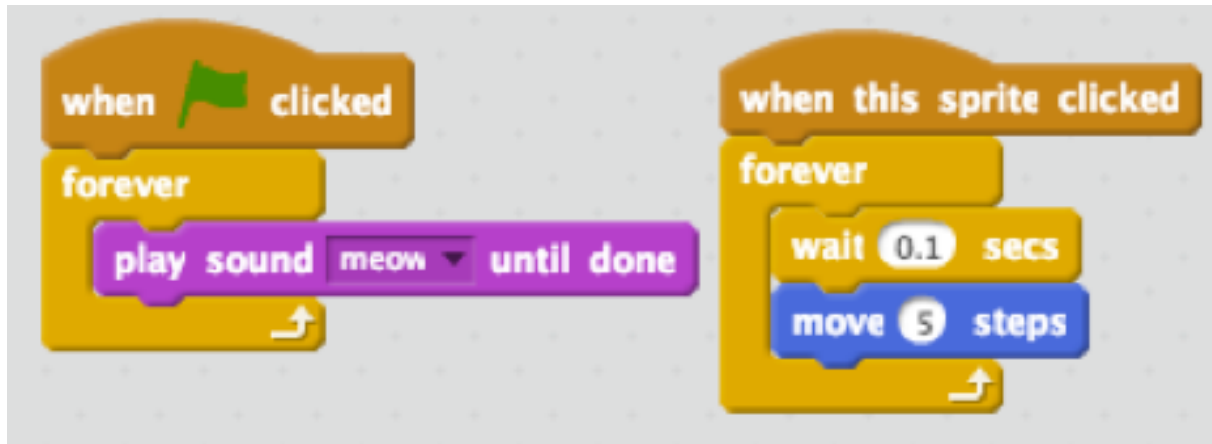
Working Player for Recording, Part 1



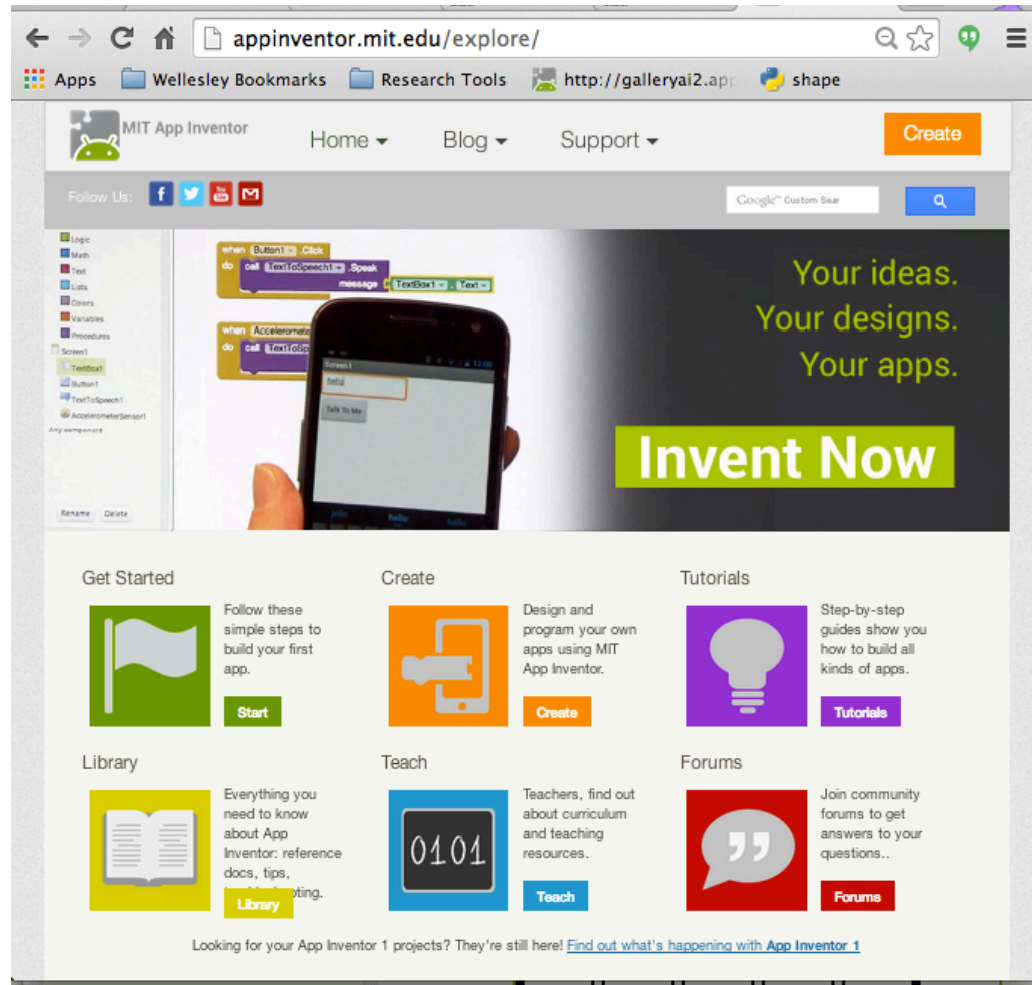
Working Player for Recording, Part 2

```
to playNextRecording
do
  if
    get global index > length of list list get global recordings
  then
    set PlayAllButton . Text to " Play All "
  else
    set PlayAllButton . Text to get global index
    set Player1 . Source to select list item list get global recordings
    index get global index
    call Player1 .Start
```

Scratch Example Revisited



Thank You! Questions?



The screenshot shows the MIT App Inventor website homepage. The browser address bar displays 'appinventor.mit.edu/explore/'. The page features a navigation menu with 'Home', 'Blog', and 'Support' links, and a prominent orange 'Create' button. Below the navigation is a 'Follow Us' section with social media icons for Facebook, Twitter, YouTube, and Email, alongside a Google Custom Search box. The main content area is a large banner with a smartphone displaying a 'Talk to Me' app, accompanied by the text 'Your ideas. Your designs. Your apps.' and a large green 'Invent Now' button. Below the banner are six categorized sections: 'Get Started' (with a 'Start' button), 'Create' (with a 'Create' button), 'Tutorials' (with a 'Tutorials' button), 'Library' (with a 'Library' button), 'Teach' (with a 'Teach' button), and 'Forums' (with a 'Forums' button). At the bottom of the page, a note states: 'Looking for your App Inventor 1 projects? They're still here! [Find out what's happening with App Inventor 1](#)'.



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

3-Day App Inventor Workshop for Undergraduate Faculty



Computational Thinking through Mobile Computing

Mon July 21 – Wed July 23, 2014
University of Massachusetts Lowell

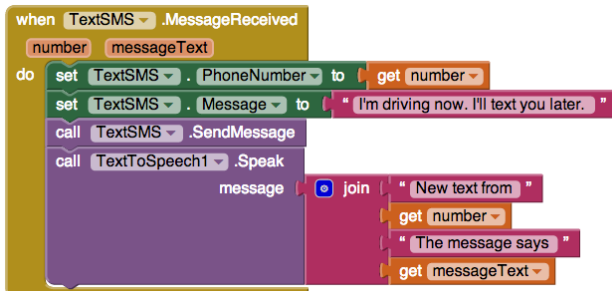


Workshop highlights:

- Hands-on sessions learning MIT App Inventor 2, with new browser-based blocks programming interface
- Teach computational thinking in CS0/CS1 while your students build Android phone & tablet apps
- Whole-course designs and modules for existing courses for majors and nonmajors
- Resources: videos, screencasts, tutorials, quiz-making environment, sample student projects
- Rubric for assessing mobile computational thinking
- Field trip to MIT Media Lab

Workshop Leaders: Learn from the creator of App Inventor and some of the early teaching pioneers.

- Hal Abelson (creator), Shay Pokress, Josh Sheldon (MIT)
- Ralph Morelli (Trinity College)
- Dave Wolber (University of San Francisco)
- Fred Martin, Karen Roehr, Mark Sherman (UMass Lowell)
- Eni Mustafaraj, Franklyn Turbak (Wellesley College)
- Larry Baldwin (BIRC, project evaluator)



App Inventor 2 blocks code for
No Texting While Driving App

More details and application:

<http://bit.ly/mobileCT-workshop-2014>

Preference given to those who plan to
use App Inventor in their courses

\$500 stipend. Apply now!

Application deadline: Apr 28, 2014.



<http://bit.ly/mobileCT-workshop-2014>