Mobile Apps with App Inventor

written for 91.113

Michael Penta

Table of Contents

Mobile Apps 4
- Designing Apps in App Inventor 4
- Getting Started 5
- App Inventor Layout 5

Your First App 7
- Making a Plan (storyboarding) 7
  - Event Driven Programs 7
- Making your first app 8
  - Adding a Label 8
  - Adding a Button 9
  - Adding an image to a button 9
  - Adding a Sound 10
  - Starting The Blocks Editor 10
  - Programming the Button 11
  - Running the Code (Emulator) 12
- Programming Challenge 13

Sending a Tweet 14
- Making a Plan (storyboarding) 14
- Expanding the App 14
  - Adding a Button 14
  - Adding a Screen Arrangement 14
  - Adding a Twitter Component 15
- Programming the Twitter App 15
  - Login Button 15
  - Tweeting the Meow 16
  - Test the Twitter App 16

Dice Rolling App 17
- Storyboard 17
- Layout and Components 17
- Building The App 18
  - Adding a Screen Arrangement 18
  - Adding the labels 18
  - Adding a Button 18
  - Adding a Notifier 18
  - Adding an Accelerometer 18
- Programming the App 18
  - The Roll Button 18

Setup and Other Tutorials 21
- Setup App Inventor on any computer 21
- A Collection of Tutorials 21
- Where's My Car Location App 21
- HFOSS Tutorials 21
- AppInventorBlocks.com 21

Limitations of the Emulator 22

Cookbook **23**
  Switching Screens **23**
  Using Sensors **24**
    Using Location 24
    Using Orientation 25
    Using the Accelerometer 26
  Talking to the Web **27**
  Web Viewer **27**

Mobile Apps

A mobile application is a piece of software developed for mobile computing devices such as smart phones and tablet devices. There are many tools available for developing mobile apps, but you will use Google (now MIT) App Inventor. App Inventor was designed so that those with little or no programming experience could develop Mobile Applications for Android based phones.
A mobile application consists of the follow components:

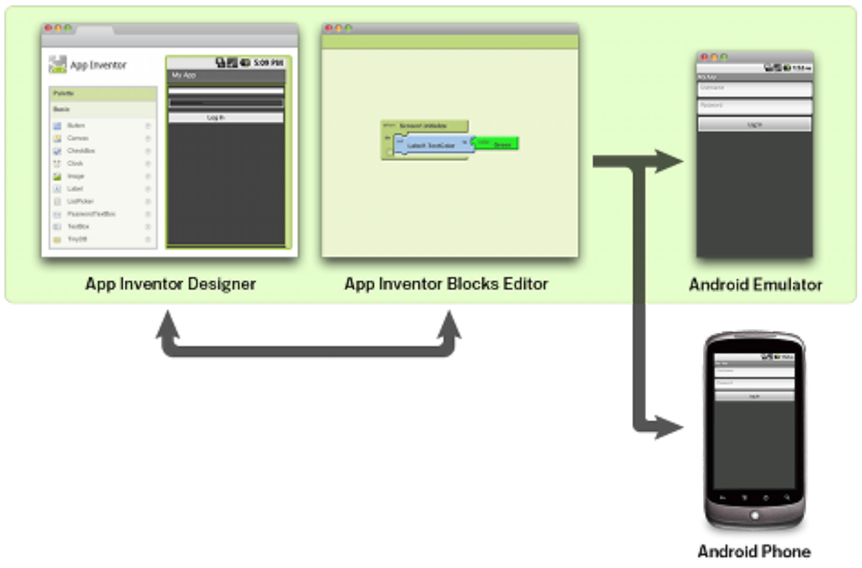
- **visual components** -- things the user can see on the phone's screen
  - buttons
  - labels
  - images
- **non-visual components** -- invisible things an application interfaces with
  - sound
  - twitter service
- **events** -- things that can happen, usually instigated by the user of the phone.
  - button being touched
  - user choosing something in a list
- **behaviors** -- actions executed by the application.
  - play a sound
  - move an image around a game board.
- **event-handlers** --- behaviors executed in response to some event.
- **variables**-- memory kept by the application
  - the score of a game

By connecting and combining these pieces, you can create a mobile application for Android phones.

Designing Apps in App Inventor

App Inventor has four elements that are used while designing apps.

1. **App Inventor Designer** – this is where you create the look of you app and decide which components your app will use.
2. **App Inventor Blocks Editor** – this is where you "write" code.
3. **Android Emulator** – you can test your app on a virtual phone
4. **Android Phone** – test and launch your app (optional)



Getting Started

To begin with App Inventor, you need a Google account. If you do not have one get one here – https://accounts.google.com

App Inventor is a web based programming environment. All you projects are stored on the web and can be accessed from any (JavaScript enabled) browser.

- Start by pointing your browser to **http://AppInventor.mit.edu/**
- Click the button in the top right corner that says **"Use MIT App Inventor"**
- Sign in with your account information (if you are not already)

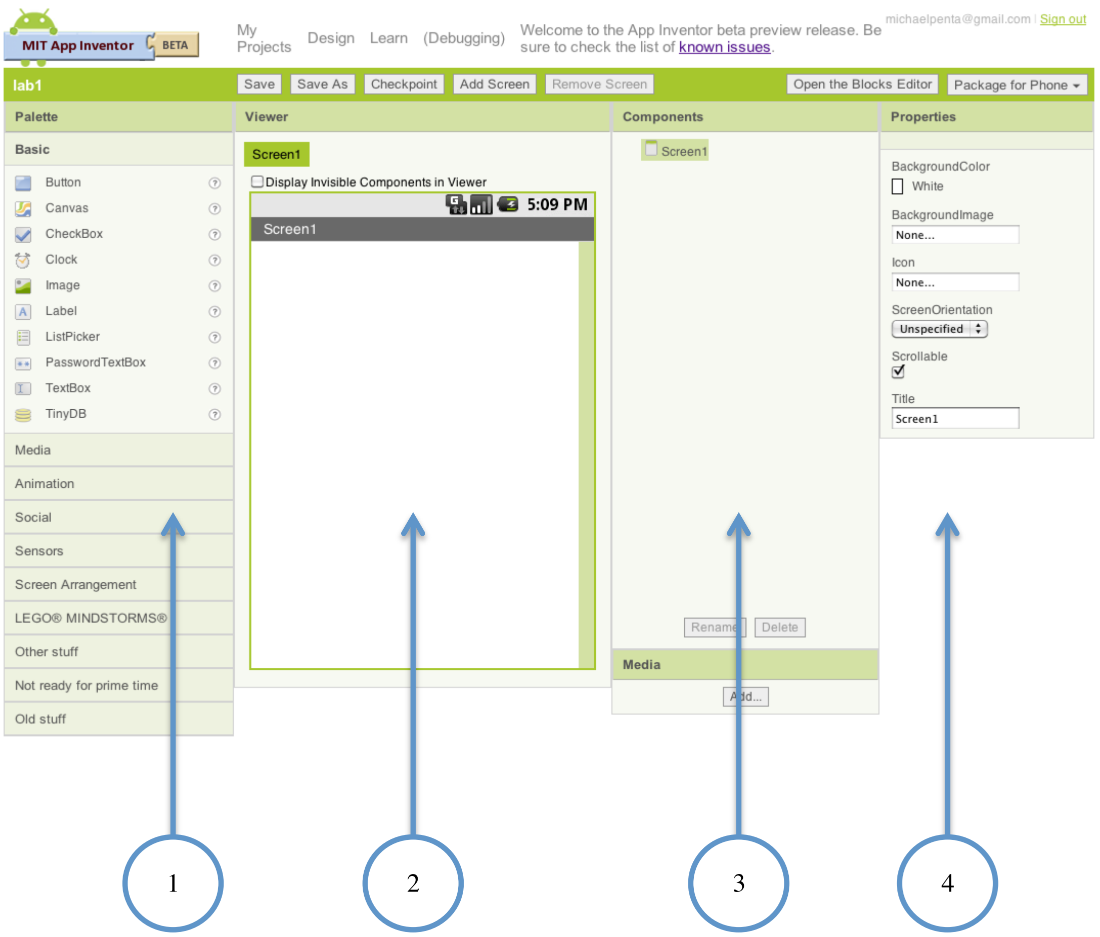
This will open up the **App Inventor Designer.** This is where you will design our apps and decide which components you will use.

Each app you create starts with a Project. The project will contain all of you components, code, and media

- Click **My Projects**.
- Click **New**
- Name Your App ***demo*** and press **OK**

App Inventor Layout

The designer consists of four parts, the Palette pane, the Viewer pane, the Components pane, and the Properties pan - each of these parts a labeled and described below.



Welcome to the App Inventor beta preview release. Be sure to check the list of known issues.

☐ Display Invisible Components in Viewer

BackgroundColor: ☐ White
BackgroundImage: None...
Icon: None...
ScreenOrientation: Unspecified
Scrollable: ✓
Title: Screen1

1. **Palette Pane**: collection of components. Think of these as the parts to your app. Some you can see, some you cannot. Drag these components to the viewer to use them.
2. **Viewer Pane**: this is the canvas for your app. You can arrange components to create the layout of your app. Components that are added to the viewer can be coded in the blocks editor
3. **Components Pane**: this is a list of components used in your app. Clicking on a name allows you to set its properties
4. **Properties Pane**: this is the section from which you edit the properties of the components picked from the palette. The component must be selected in the list in order to edit its properties

Your First App

You will need an image and a sound to complete this tutorial. They are available here (http://dl.dropbox.com/u/1394403/app1Media.zip). Download these files and continue.

Making a Plan (storyboarding)

Before building and programming your application, it is helpful to create a storyboard. A storyboard is a sequence of drawings and text that layout the users experience for the application. The storyboard will contain the major components and describe the user interaction (what happens when a user performs an action) - our first app is simple (only one screen and a few components), so our storyboard will not be very extensive.

Event Driven Programs

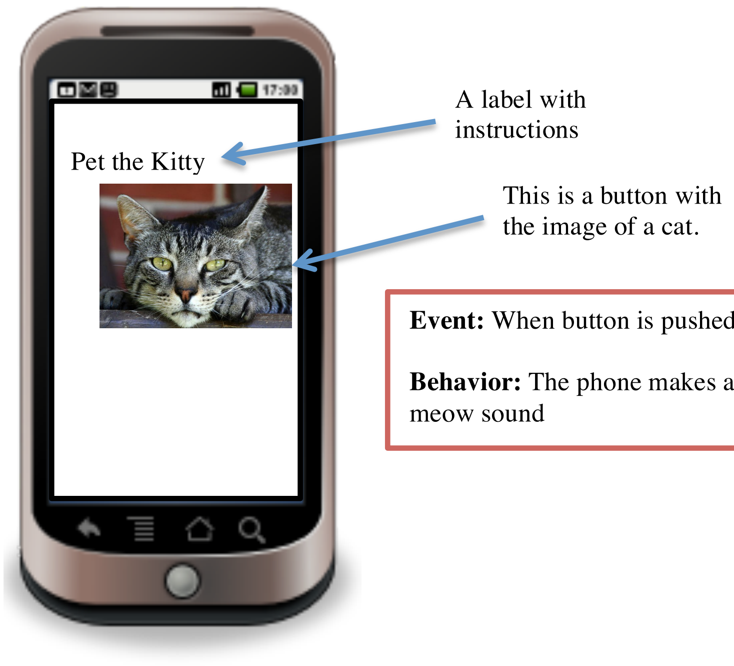
App Inventor is an EVENT driven environment. Events trigger actions/behaviors.

- Events are things like: clicking a Button, shaking the phone, changing your location, getting a text message, etc.

- Behaviors are things like: sending a text, tweeting, showing a map, opening an application or webpage, etc.

Your storyboard should identify all the events to which your app will react. Identifying events before hand makes programming easier.

App Description and Storyboard:
When the app is started, a picture of a cat will be shown. When the user touches the cat, a meow sound will be played.


Pet the Kitty
A label with instructions
This is a button with the image of a cat.

Event: When button is pushed

Behavior: The phone makes a meow sound

Making your first app

Now that you have an idea and a storyboard, you can start to build our app. Before you program the app, you need to add the components you want to program.

To make this app you will need the following components:

- A Button (Basic Component)
- A Label (Basic Component)
- A Sound (Media Component)

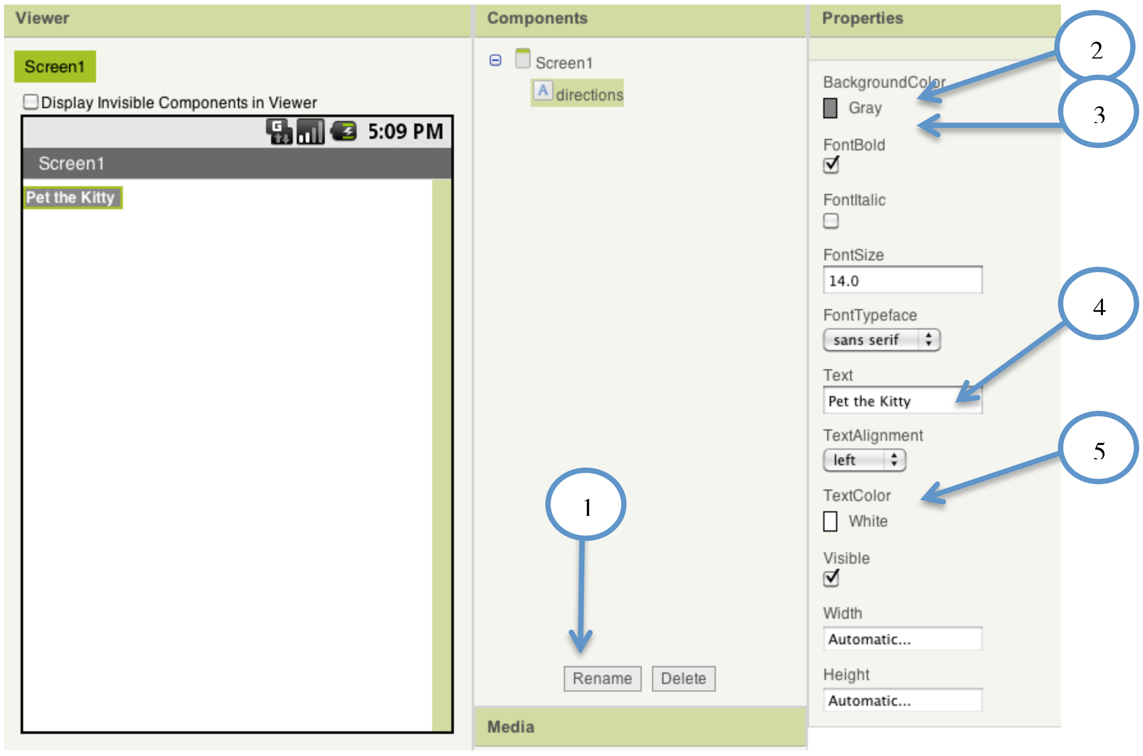
Adding a Label

Drag a Label component from the Basic section on the Component Pallet to the Viewer Pane. You will see the label listed in the Component Pane. By default, the newest component is selected. The selected component's properties can be set in the Properties Pane. It is a good idea to give every component a name (not label1, label2). The name you give it will be its name in the Blocks Editor; this will make things easier as you add more components. You also want to change the label text and some other properties.

Select the label component from the component pane, its properties will be shown in the content pane.

Change the properties as follows:

1. Rename the **label "directions**"
2. Set the **background color** to your choice of color
3. Set the **Font to bold**
4. Set the **text** property to "Pet the Kitty"
5. Set the **text color** to a color of your choosing

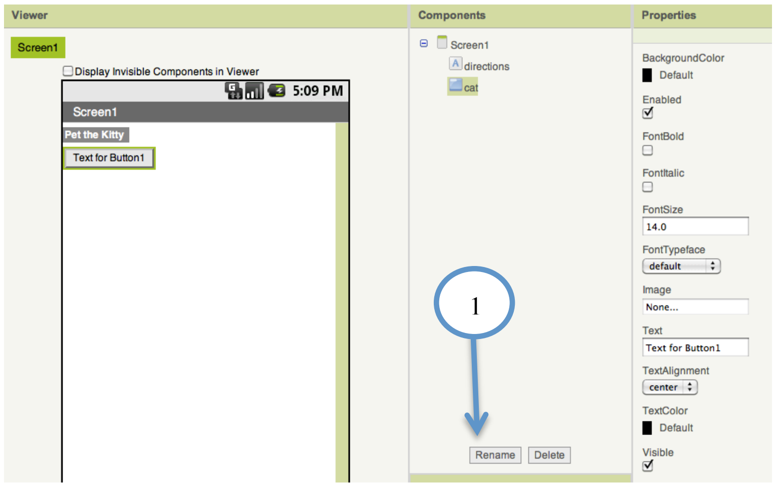


BackgroundColor
Gray
FontBold
✓
FontItalic
☐
FontSize
14.0
FontTypeface
sans serif
Text
Pet the Kitty
TextAlignment
left
TextColor
White
Visible
✓
Width
Automatic...
Height
Automatic...

Adding a Button

Drag a Button component (in the Basic section) to the Viewer Pane. You will see the button listed in the Component Pane.

1. Rename the button to "cat"



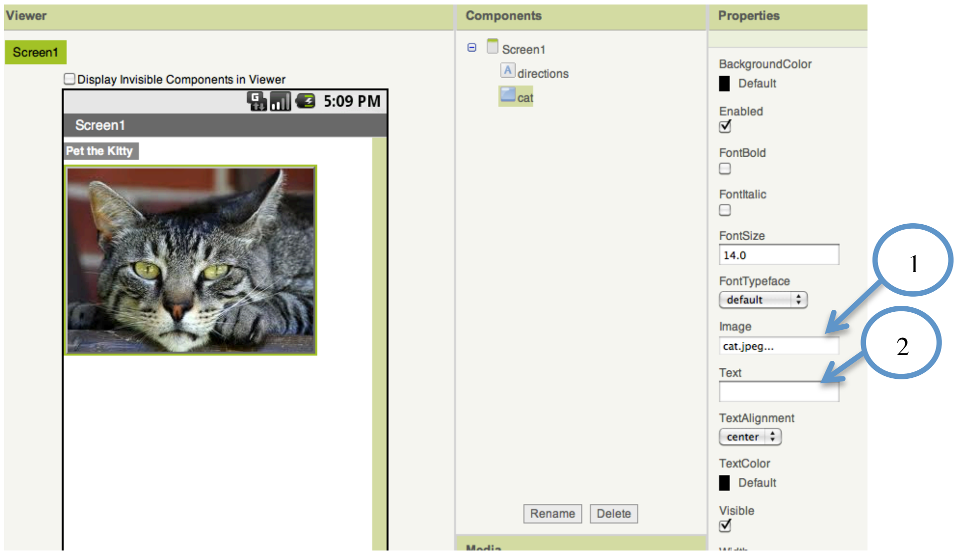
BackgroundColor: Default
Enabled: ✓
FontBold: ☐
FontItalic: ☐
FontSize: 14.0
FontTypeface: default
Image: None...
Text: Text for Button1
TextAlignment: center
TextColor: Default
Visible: ✓

You could change the text of the button or it "as is" and this tutorial will still work, but you can improve it by adding an image to the button.

Adding an image to a button

With the component selected in the Component Pane

1. Click the image property
   - Click **Add…**
   - Select an image from the file browser
   - Select the image from the list
   - Click **OK**
2. Delete the text from the **Text** property

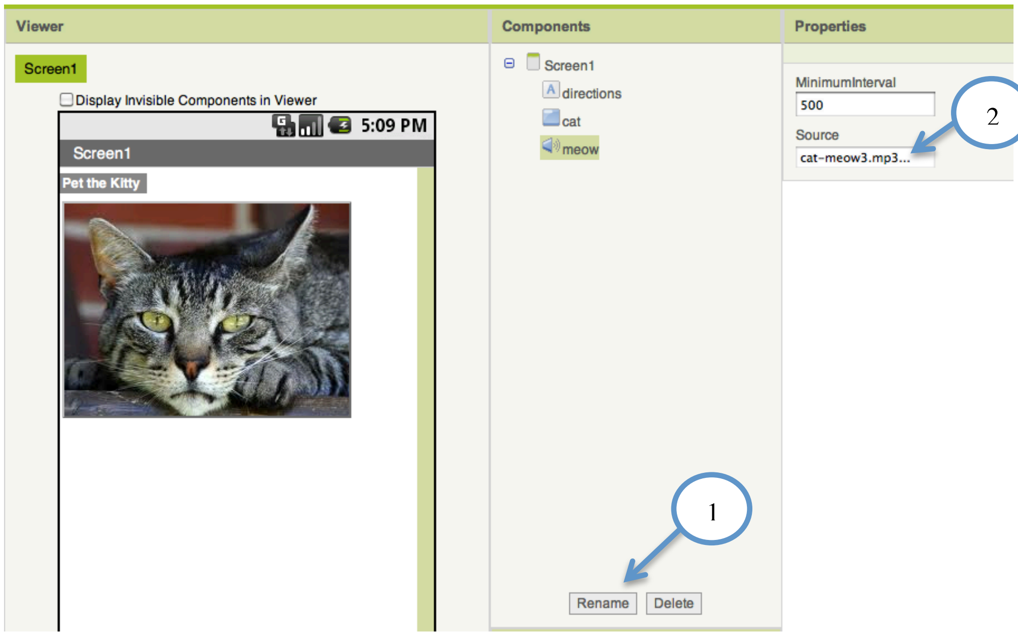


BackgroundColor: Default
Enabled: ✓
FontBold: ☐
FontItalic: ☐
FontSize: 14.0
FontTypeface: default
Image: cat.jpeg...
Text:
TextAlignment: center
TextColor: Default
Visible: ✓

Adding a Sound

Drag a Sound component (in the Media section) to the Viewer Pane. You will see the button listed in the Component Pane.

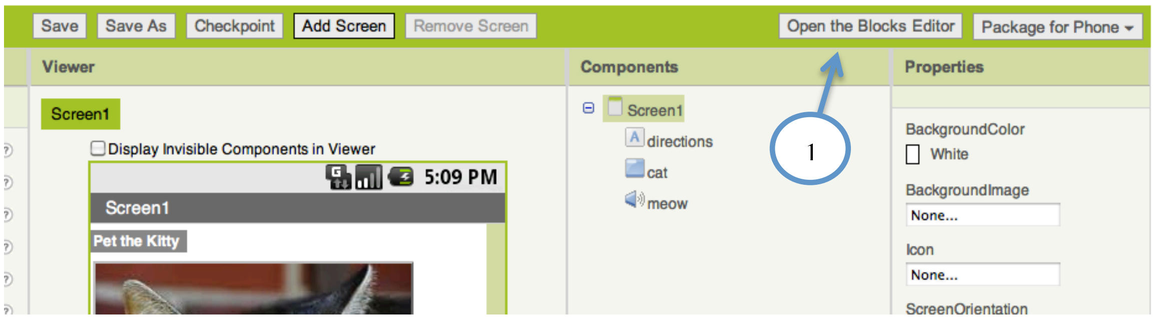
1. Rename the sound to meow
2. Click the source property
    - Click **Add…**
    - Select a sound from the file browser
    - Select the image from the list
    - Click **OK**



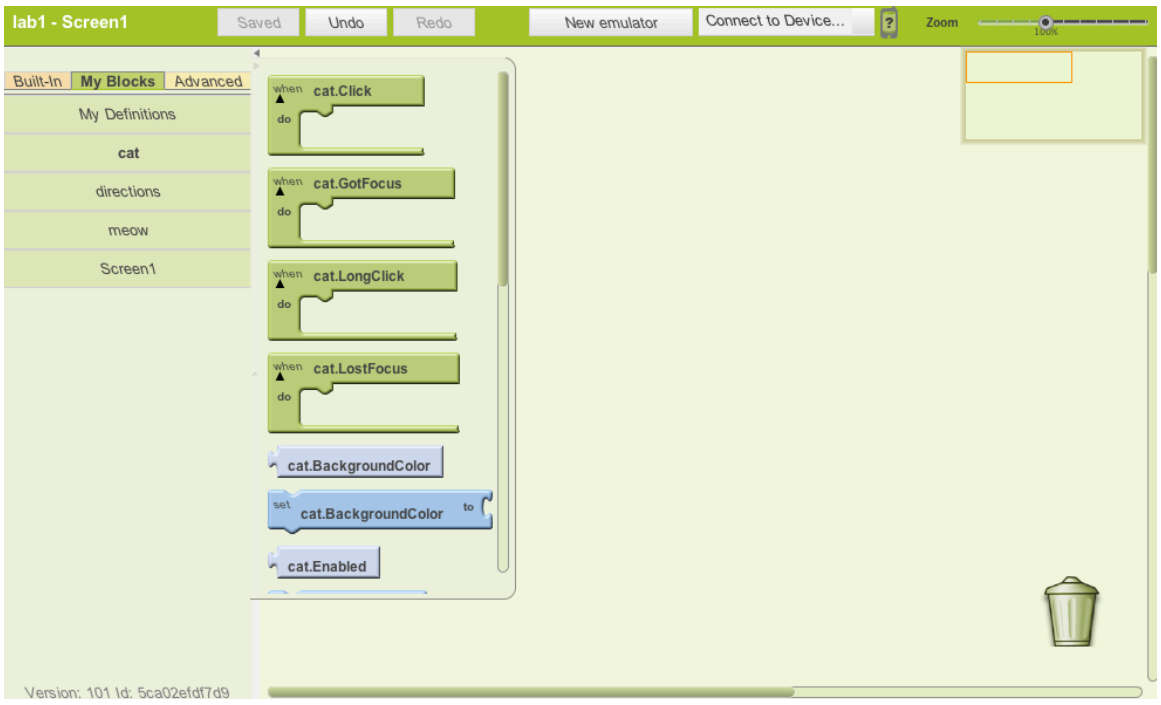
Starting The Blocks Editor

Dragging components to the Viewer Pane allows us to use those components in out program. To program components, you need to open the Blocks Editor.

1. Click the "**Open Blocks Editor**" Button
    a. The browser will download a file (*AppInventorForAndroidCodeblocks.jnlp*)
    b. If the file does not open automatically, double click it.



The Blocks Editor consists of the main programming stage (on the right), and the Blocks Tabs (on the left). Dragging blocks to the stage and connecting them, you will program the behaviors of your components.



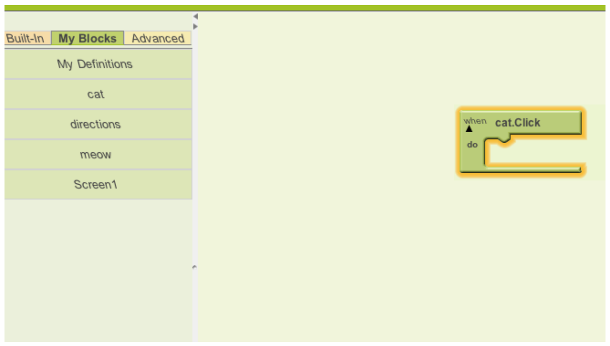
Programming the Button

Mobile Applications are EVENT DRIVEN. The user does something or a sensor value changes, and some behavior occurs as a result. The storyboard describes an event and resulting behavior – *"When a user presses a button, a meow sound is played."*

1. Click the **My Block**s tab
2. Click the **cat component**
   a. A list of blocks will be displayed

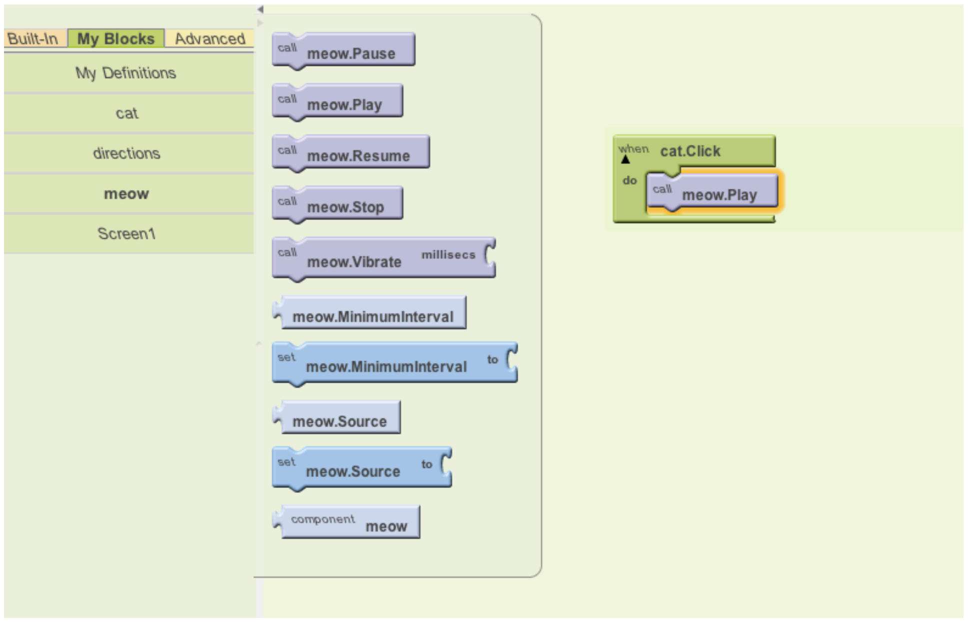
You will see the first block says "**cat.Click**" – this block's behavior will be triggered when the user clicks the cat button.

1. Drag the **cat.Click** block anywhere on the Stage.



This block will run when the cat Button is clicked. We can give this click event a behavior by adding blocks into its "mouth." The blocks inside the mouth run in sequential order.

- Click on the **meow component**
- Find the block that says “**meow.Play**”
- Drag it into the mouth of the **Click** block and connect them as shown



This code can be read as “ **When** the cat button is **Clicked**, **Play** the **meow** sound.”

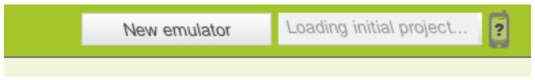
Running the Code (Emulator)

Now that we have given the button a behavior, we want to test the code. App Inventor has a phone emulator that we can use for testing code. The emulator has some limitations, but it will suffice in most cases.

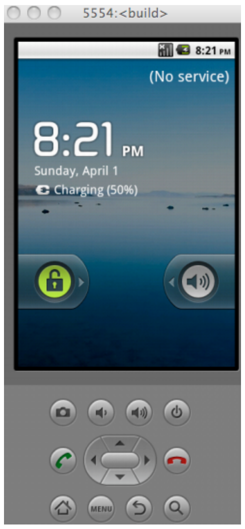
Starting the Emulator

You must first start the emulator.

- Click the “**New Emulator**” Button
  - A window will pop-up
  - **Click OK**



A window containing the phone emulator will open.



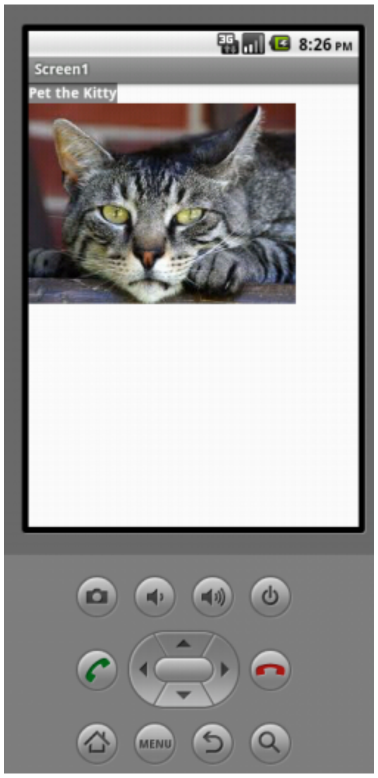
Connecting the Emulator

After the emulator loads, you need to connect the phone:

- Click the “**Connect to Device**” Button
- Select “**emulator-####**” from the list

Unlock and Run

Using your mouse, slide the unlock button on the phone emulator. Your code will run momentarily.



The code is running just as it would on a mobile phone. You can interact using the mouse – instead of your finger on a real phone.

Try interacting with the application by clicking the cat.

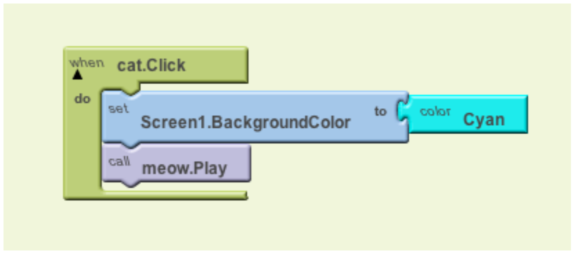
Adding More Behaviors

Any event can have many behaviors. In addition to making the cat meow, you can change the background color of the screen itself.

- Drag the "**Screen1.BackgroundColor**" block from the Screen component as shown below

Notice that to piece has a whole on the end – like a puzzle piece is missing. When a behavior requires additional information, we need to add more blocks. In this case, we need a color block.

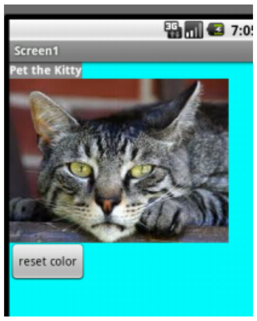
- Click the "Built-In" tab
- Click the "Color" component
- Drag a color and connect it as shown



SAVE YOUR WORK and try out your changes on the emulator.

Programming Challenge

You have seen some basic of the basic steps that are required ion making an App Inventor application. Let's take it a little farther…



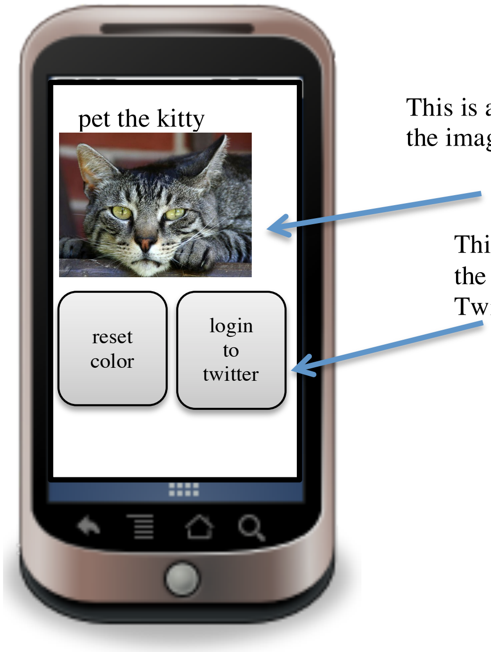
- Add an additional button to the screen.
- Rename the button to "reset"
- Change the button **text** to read "Reset Color"
- Program the button to set the **background color** of the screen to a different color
- Try your app on the emulator.

Sending a Tweet

Open the first app project. (You need to complete the first application before creating this one)

Making a Plan (storyboarding)

This application is the same as the first app, with an additional a login button for twitter. When the meow sound is played, the word meow is also tweeted to a demo twitter account.



This is a button with the image of a cat

This is a button with the text "Login to Twitter"

When: A user touches the Kitty

Behavior: Play a meow sound and tweet meow

When: A user touches the login button

Behavior: allow the user to login to twitter

Expanding the App

To expand this application we will need to add the following components:

- A (new) Button (Basic Component)
- A Twitter component (Social Component)
- A HorizontalArrangement (Screen Arrangement)

Adding a Button

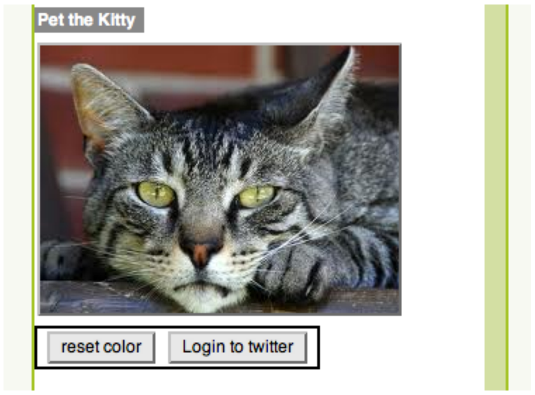
Drag a Button component (in the Basic section) to the Viewer Pane. You will see the button listed in the Component Pane.

1. Rename the button to "login"
2. Change the **Text Property** to "login to twitter"

Adding a Screen Arrangement

When you added the button, it is stacked on top of or beneath another component, like the rest of the components. We can change how components are arranged using a screen arrangement.

- Drag a **HorizontalArrangement** component (in the Screen Arrangement section) to the **Viewer Pane**.
- You will see the component listed in the Component Pane. Drag both of the buttons into the horizontal arrangement and arrange them as shown.



Adding a Twitter Component

Drag a Twitter component (in the Social section) to the Viewer Pane. You will see the component listed in the Component Pane.

A demo twitter account has been created for this app. The information for the account is as follows:

- **Username:** 91113App
- **Password:** App91113
- **Consumer Key:** FUbhwbx06g5n2kef2J0Bg
- **Consumer Secret:** ZbVZMzI6Z0TzghGAntD20QkRcuWMCsywla7xkPtmbI

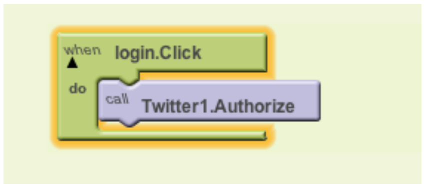
You can use this account, or create your own.

- Edit the Twitter Component Properties to match the account settings
  - Consumer Key: FUbhwbx06g5n2kef2J0Bg
  - Consumer Secret: ZbVZMzI6Z0TzghGAntD20QkRcuWMCsywla7xkPtmbI

Programming the Twitter App

Login Button

The storyboard identifies the behavior – "When the login button is clicked, login to Twitter"



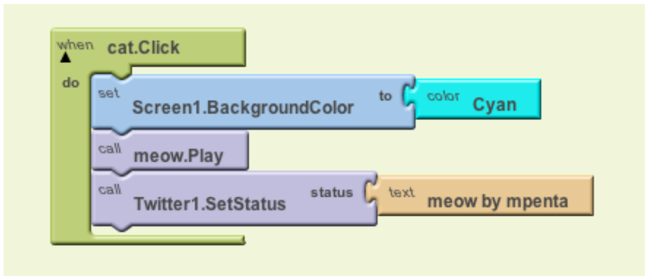
- Click the **My Blocks** tab
- Click the **login component** to open the drawer
- Drag a **Click event** block to the stage
- Click the **Twitter component** to open the drawer
- Connect it to the mouth of the **click event** as shown

You have used the click event before, but the Authorize behavior is new. This behavior checks if the user is logged into Twitter. If the user is not logged in, a Twitter login page is presented to the user.

Tweeting the Meow

Once the user is logged in, you can set the user status. The storyboard states that the app should tweet when the user touches the cat button.

- Click the **My Blocks** tab
- Click the **Twitter Component**
- Drag a **Twitter.SetStatus** behavior to the **cat.Click** event
- Click the **Built-In** tab
- Click the **Text** section
- Drag a **Text** block and connect it to the status as shown.
- Change the text to say "meow" include an identifier for your self


when cat.Click
do
set Screen1.BackgroundColor to
color Cyan
call meow.Play
call Twitter1.SetStatus status
text meow by mpenta

Test the Twitter App

If it is not already, start and connect the emulator.

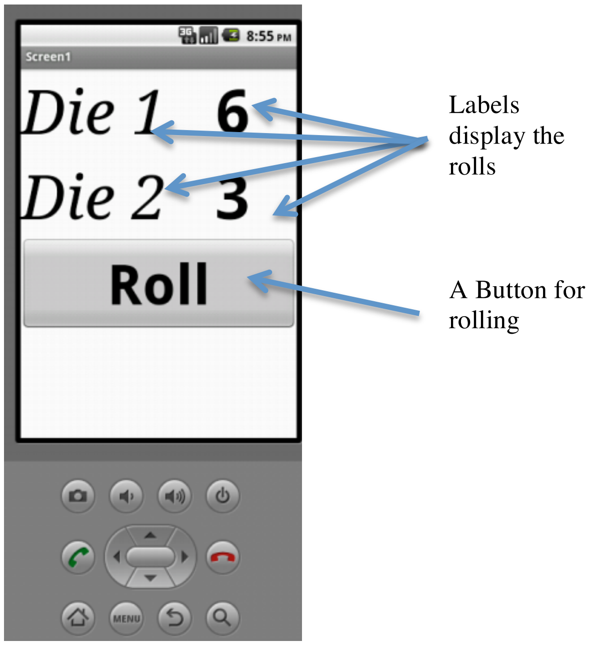
- Test they App by first clicking the login button
- Login with the information provided
  - **Username:** 91113App
  - **Password:** App91113
- Touch the cat
- Check the twitter stream
  - http://twitter.com/91113App

Dice Rolling App

The Dice Rolling application is exactly what it sounds like – it will roll two virtual dice. This application will give you more practice with App Inventor and provide you information on more of the tools available.

Storyboard

When the phone is shaken or a button is clicked, this application will display two randomly chosen numbers. When doubles are rolled a notification is presented to the user.



Event: Phone is shaken

Behavior: Choose two random numbers and display them

Event: Button is pressed

Behavior: Choose two random numbers and display them

Event: Doubles are rolled

Behavior: Display a notification

Layout and Components

This application uses the following components:

- Label (Basic - four of them)
- Table Arrangement (Screen Arrangement)
- Button (Basic)
- Accelerometer (Sensor)
- Notifier (Other Stuff)

The notifier and the accelerometer are non-visible components; you will not see them when you add them to the viewer pane. The other components are arranged as shown (The table below represents the table arrangement in the Viewer Pane).

Label1	Roll1
Label2	Roll2

Button

Building The App

Adding a Screen Arrangement

Drag a TableArrangement to the view pane

Adding the labels

Drag four Labels into the TableArrangement as shown in the layout section.

- Rename each label following the layout section
    - label1
    - label2
    - roll1
    - roll2
- Give the labels a large font (72)
- Make the roll labels bold
- Make the die labels italics

Adding a Button

Drag a Button to the Viewer Pane. You will see the button listed in the Component Pane.

- Rename the button to “roll”
- Change the **Text Property** to “Roll”

Adding a Notifier

Notifiers are small pop-up screens. This one will tell the user doubles were rolled.

Drag a Notifier (from Other Stuff) to the Viewer Pane.

- There are no properties to set.

Adding an Accelerometer

An accelerometer measures movement. In this app it will be used to sense the phone being shaken.

Drag an Accelerometer (from Sensors) to the Viewer Pane.

- Be sure the enabled property is checked.

Programming the App

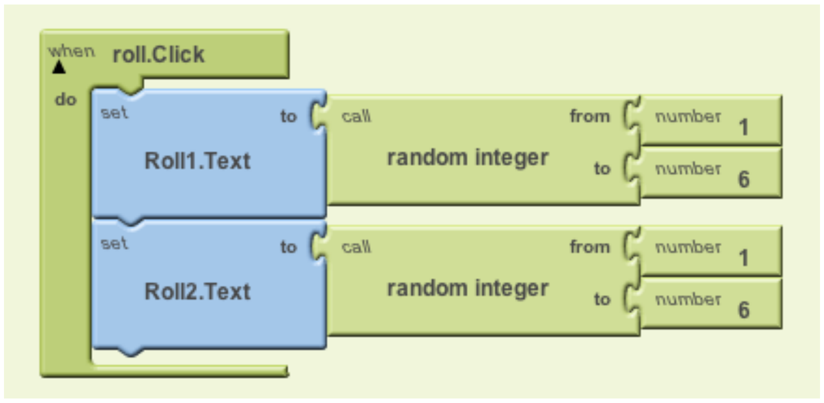
The Roll Button

When the roll button is clicked, the application is to display two random generated numbers in separate labels.

Random Number

App Inventor has a collection of Math blocks (**Built-in** tab, Math section). These blocks can do all kinds of math related actions. We want to use the blocks that generate a random integer from 1 to 6. Once the numbers are generated, the label text is changed to display the numbers.

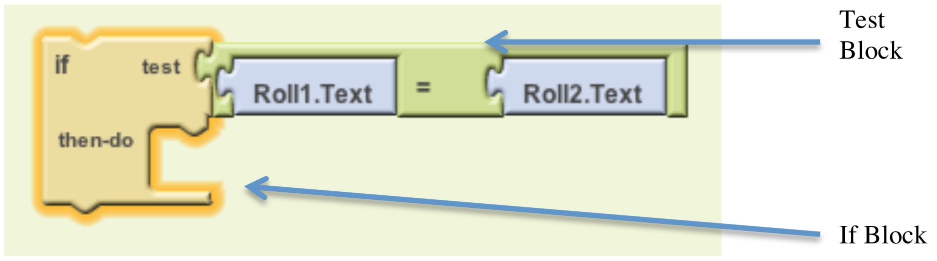
Assemble the blocks as shown.


when roll.Click
do
set Roll1.Text to
call random integer from number 1 to number 6
set Roll2.Text to
call random integer from number 1 to number 6

You can try the app in the emulator at this point. It will provide the basic functionality when the button is pressed.

If Block and the Notifier

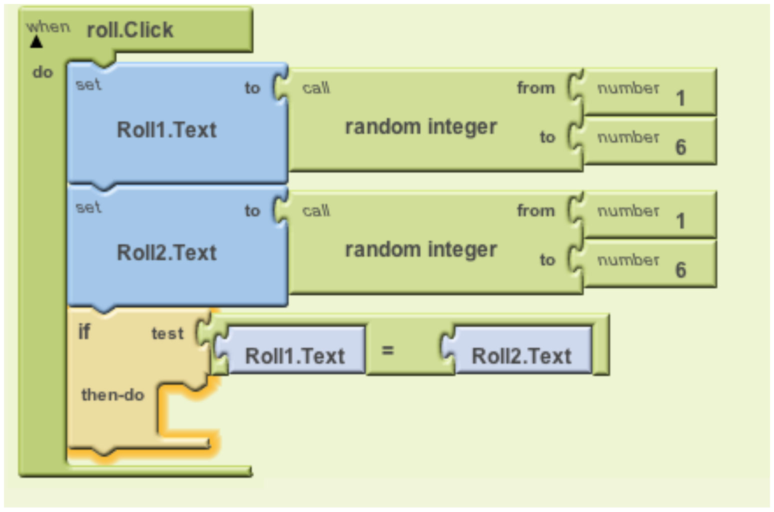
The application should display a notifier when doubles are rolled. This means we need to test the rolls and see it they are the same. Programing languages use IF statements to control the flow of the program. In this case, sometime the notifier is displayed and sometimes it is not. Control blocks can be found under the **Built-in** tab under the Control section of the blocks.



The decision can be stated as follows:

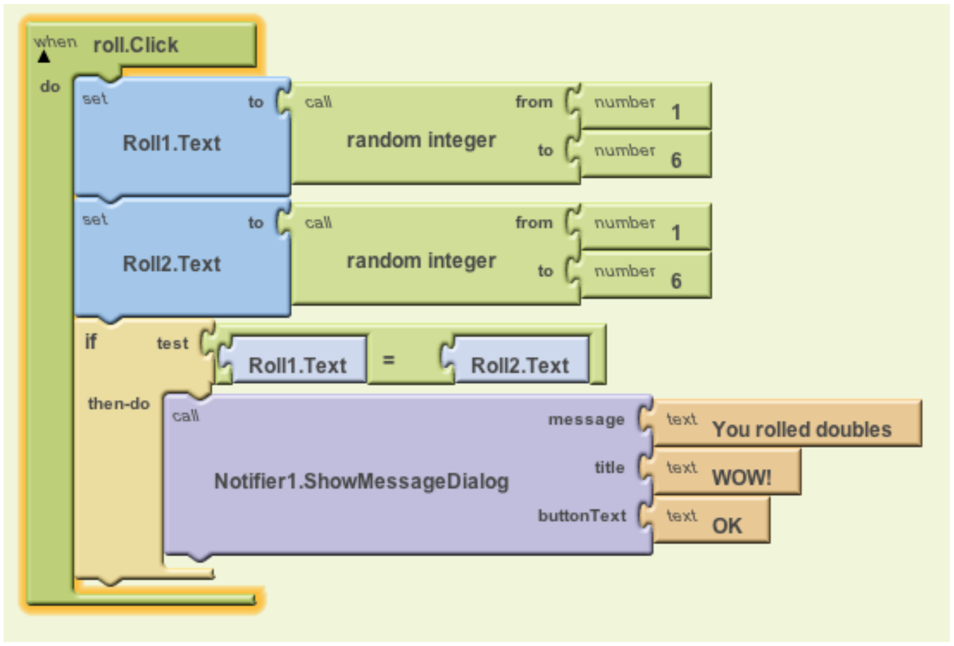
IF roll1 is equal to roll2, then display a notifier

Assemble the blocks as shown below:


when roll.Click
do
set Roll1.Text to
call random integer from number 1 to number 6
set Roll2.Text to
call random integer from number 1 to number 6
if test
Roll1.Text = Roll2.Text
then-do

The **if block** still needs a *then-do* behavior. In this case you will use **a notifier block**.

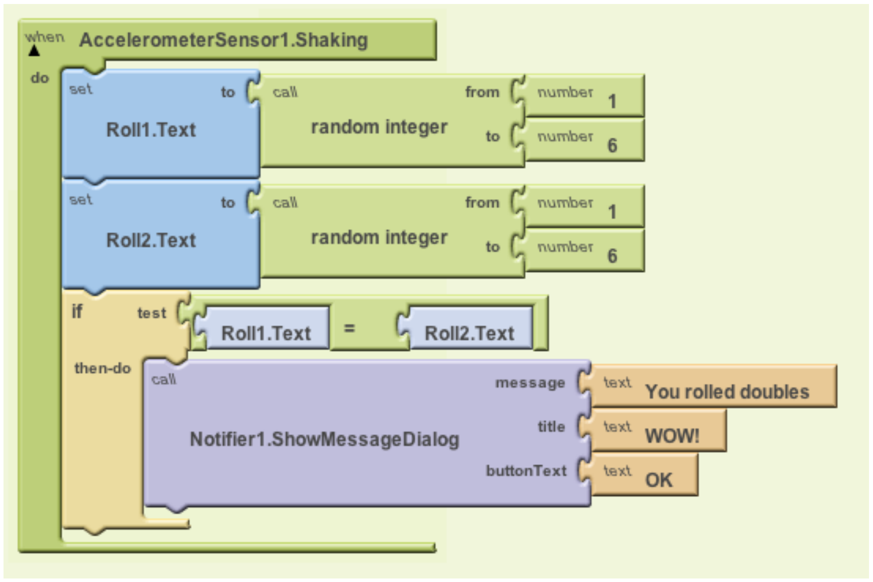
- Drag a **Notifier.ShowMessageDialog** Block to the do then behavior.
- Click on the missing input pieces
- Choose text
- Enter the text as shown.


when roll.Click
do
set Roll1.Text to
call random integer from number 1 to number 6
set Roll2.Text to
call random integer from number 1 to number 6
if test Roll1.Text = Roll2.Text
then-do
call Notifier1.ShowMessageDialog
message text You rolled doubles
title text WOW!
buttonText text OK

Test the app in the emulator.

Accelerometer

The accelerometer has a **shaking** event. You can use that even to trigger the same actions as the button. You cannot test this portion of the app unless you have an Android phone on which to test.


when AccelerometerSensor1.Shaking
do
set Roll1.Text to
call random integer from number 1 to number 6
set Roll2.Text to
call random integer from number 1 to number 6
if test Roll1.Text = Roll2.Text
then-do
call Notifier1.ShowMessageDialog
message text You rolled doubles
title text WOW!
buttonText text OK

Setup and Other Tutorials

Setup App Inventor on any computer
http://beta.appinventor.mit.edu/learn/setup/index.html

A Collection of Tutorials
http://beta.appinventor.mit.edu/learn/tutorials/index.html

Where's My Car Location App
https://joners-inventor.appspot.com/learn/tutorials/whereismycar/whereismycar.html

HFOSS Tutorials
http://notes.hfoss.org/index.php/AppInventor:Tutorials
http://notes.hfoss.org/index.php/AppInventor:Advanced_Tutorials

AppInventorBlocks.com
http://www.appinventorblocks.com/appinventor-sample-apps

Limitations of the Emulator

The emulator has some great features, but it is far from a fully functional phone. Some of the limitations of the emulator are listed here.

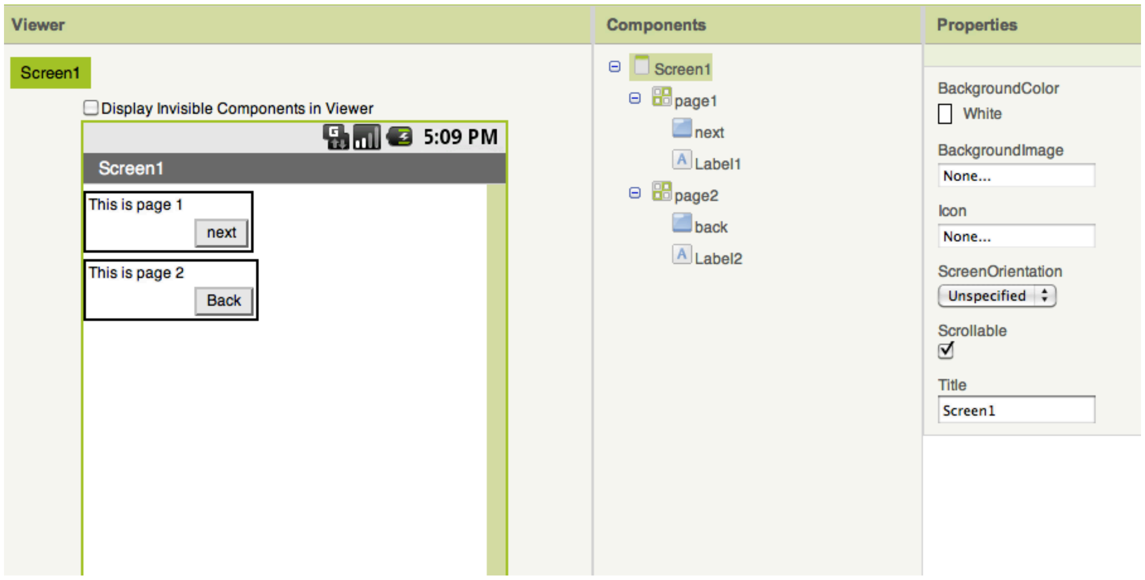
- Cannot send or receive text messages
- Cannot use orientation sensor
- Cannot use accelerometer
- Cannot use location settings
- Cannot open new screens

Cookbook

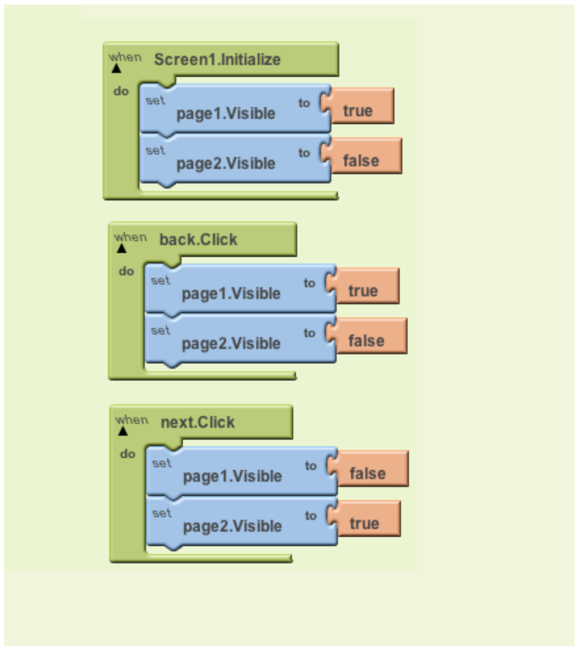
Example programs – all code available here (http://dl.dropbox.com/u/1394403/all-projects.zip)

Switching Screens

Multiple screens are not available in the current version of App Inventor, however this is easily faked using Screen Arrangement components and the visible property. Each arrangement contains a single page. When a button is clicked, all but one arrangement disappears. This is done using the visible property.



BackgroundColor: White
BackgroundImage: None...
Icon: None...
ScreenOrientation: Unspecified
Scrollable: ✓
Title: Screen1

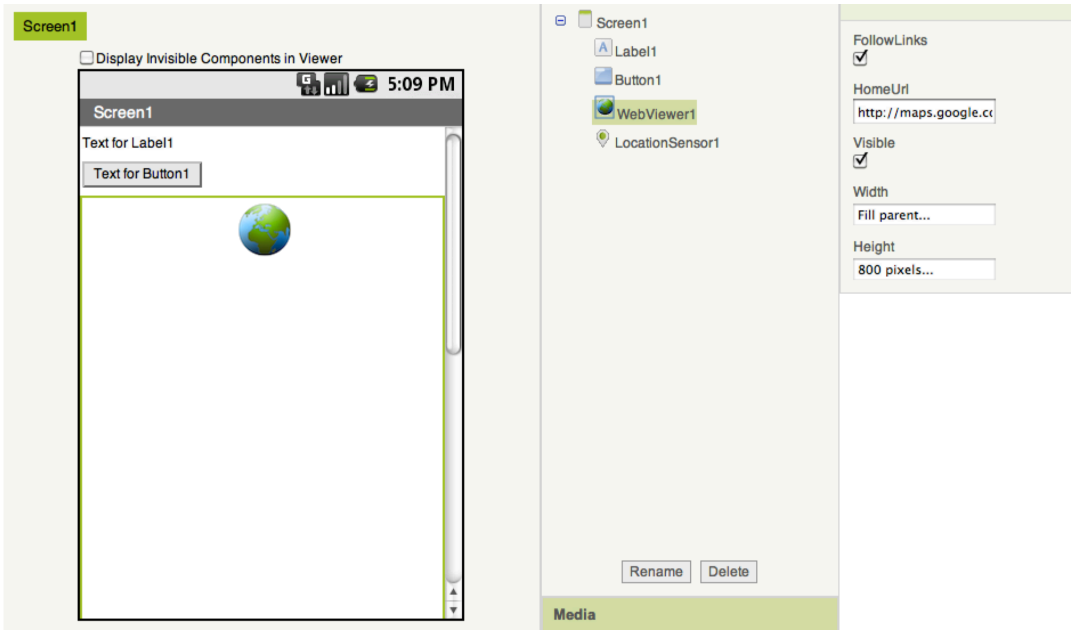


```
when Screen1.Initialize
do  set page1.Visible to true
    set page2.Visible to false

when back.Click
do  set page1.Visible to true
    set page2.Visible to false

when next.Click
do  set page1.Visible to false
    set page2.Visible to true
```

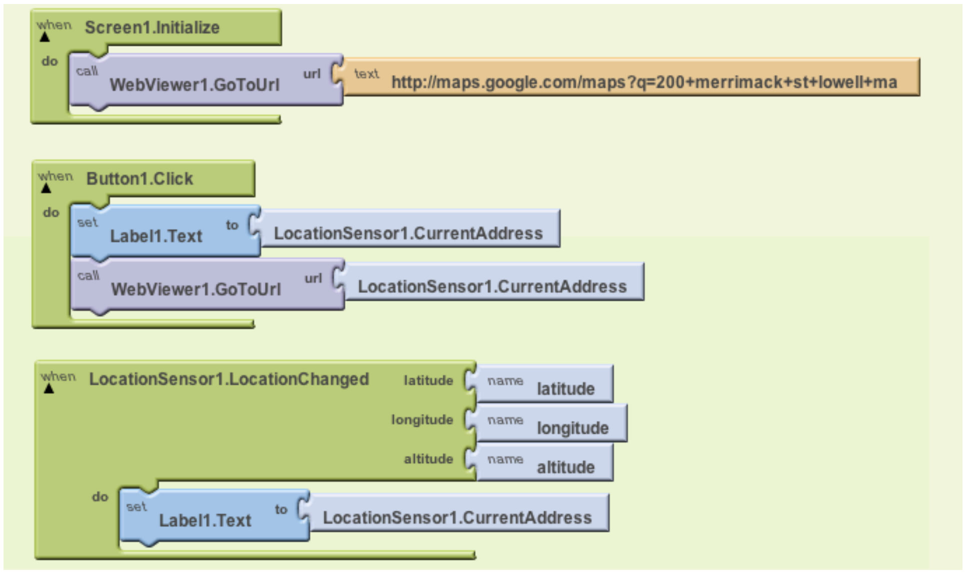
Using Sensors

Using Location

The location sensor must be enabled (set in the properties pane or using blocks) prior to use. In this example, when a button is pressed the current location is displayed on a label and plotted on a Google map in a web viewer component. The sensor also reacts to changes in location. When the location is changed, this example displays the current address. Other location information is also available such as altitude, longitude, and latitude.



FollowLinks ☑
HomeUrl http://maps.google.c
Visible ☑
Width Fill parent...
Height 800 pixels...



```
when Screen1.Initialize
do   call WebViewer1.GoToUrl url text http://maps.google.com/maps?q=200+merrimack+st+lowell+ma

when Button1.Click
do   set Label1.Text to LocationSensor1.CurrentAddress
     call WebViewer1.GoToUrl url LocationSensor1.CurrentAddress

when LocationSensor1.LocationChanged  latitude  name latitude
                                      longitude name longitude
                                      altitude  name altitude
do   set Label1.Text to LocationSensor1.CurrentAddress
```

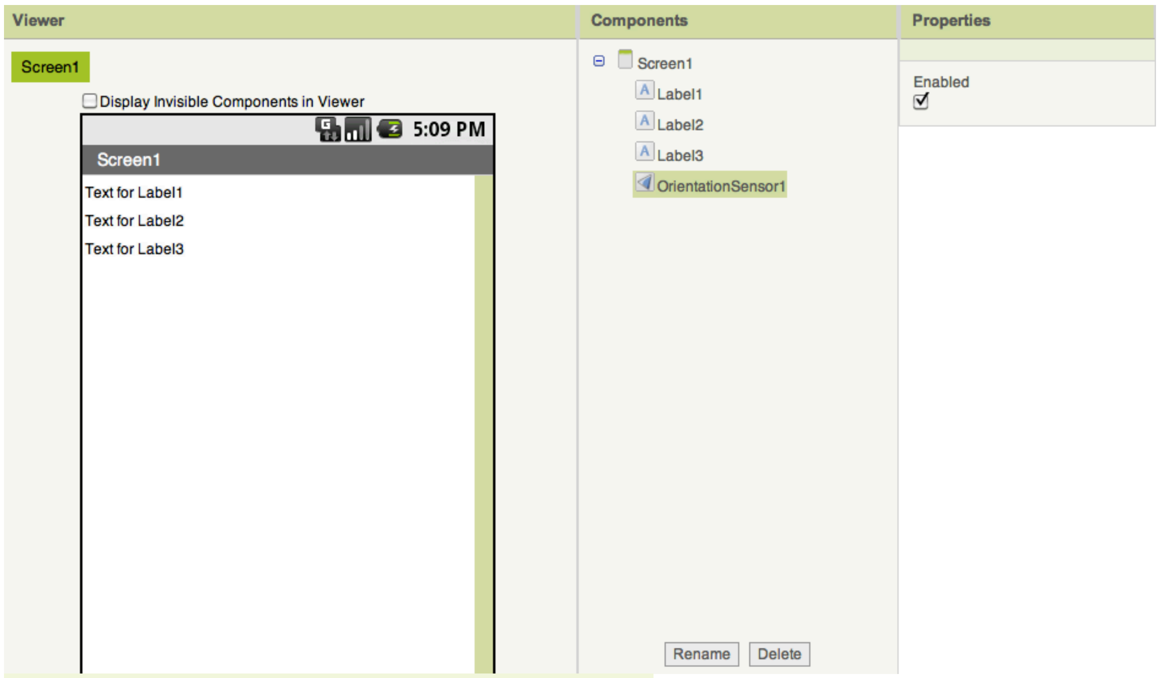
Faking It

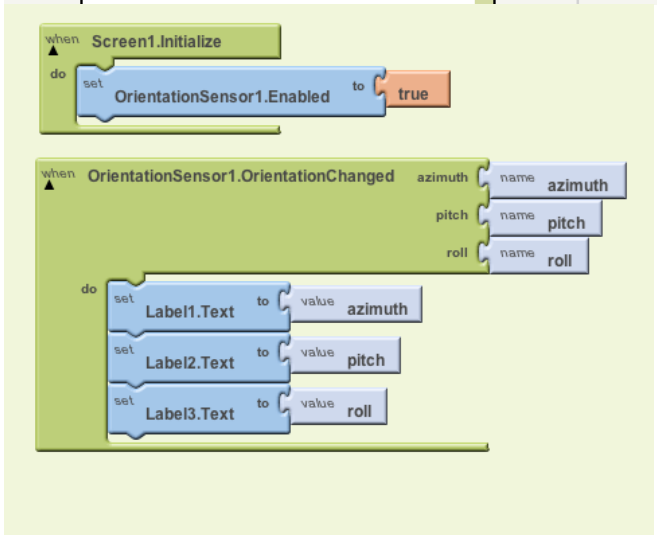
If you do not have access to an Android phone or tablet, you will need to fake it, as this feature is not supported in emulation.

Use a dummy address instead of getting the actual location.

Using Orientation

The orientation sensor must be enabled (set in the properties pane or using blocks) prior to use. In this example, when the orientation of the phone changes, the sensor data is displayed in a series of labels.





```
when Screen1.Initialize
do  set OrientationSensor1.Enabled to true

when OrientationSensor1.OrientationChanged  azimuth name azimuth
                                            pitch   name pitch
                                            roll    name roll
do  set Label1.Text to value azimuth
    set Label2.Text to value pitch
    set Label3.Text to value roll
```

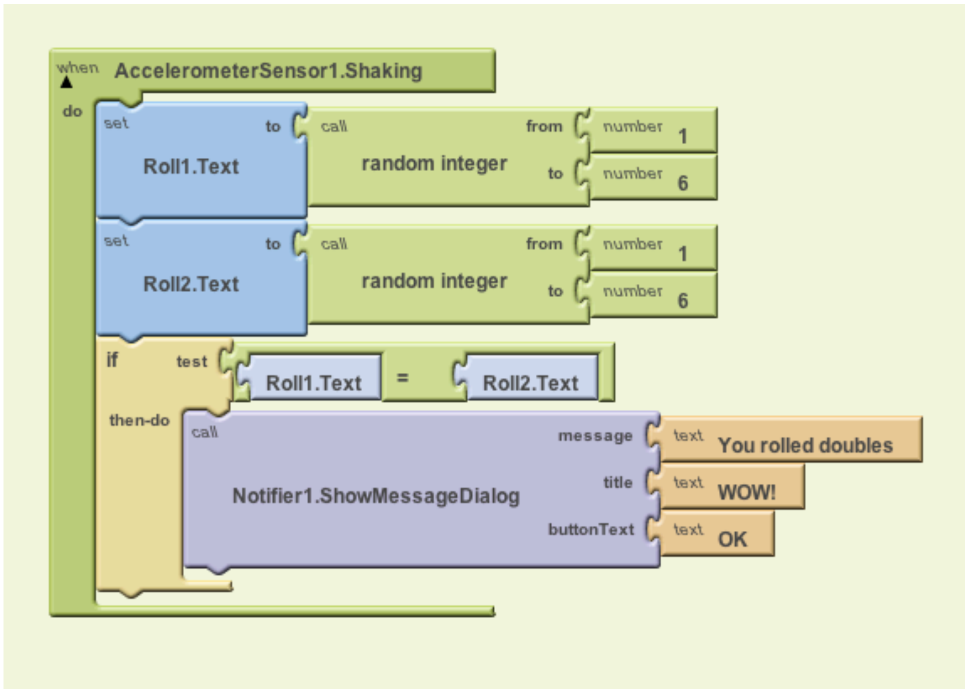
Faking It

If you do not have access to an Android phone or tablet, you will need to fake it, as this feature is not supported in emulation.

You can use a button to change the screen orientation.

Using the Accelerometer

The orientation sensor must be enabled (set in the properties pane or using blocks) prior to use. In this example, when the orientation of the phone changes, the sensor data is displayed in a series of labels.


when AccelerometerSensor1.Shaking
do
set Roll1.Text to call random integer from number 1 to number 6
set Roll2.Text to call random integer from number 1 to number 6
if test Roll1.Text = Roll2.Text
then-do call Notifier1.ShowMessageDialog
message text You rolled doubles
title text WOW!
buttonText text OK

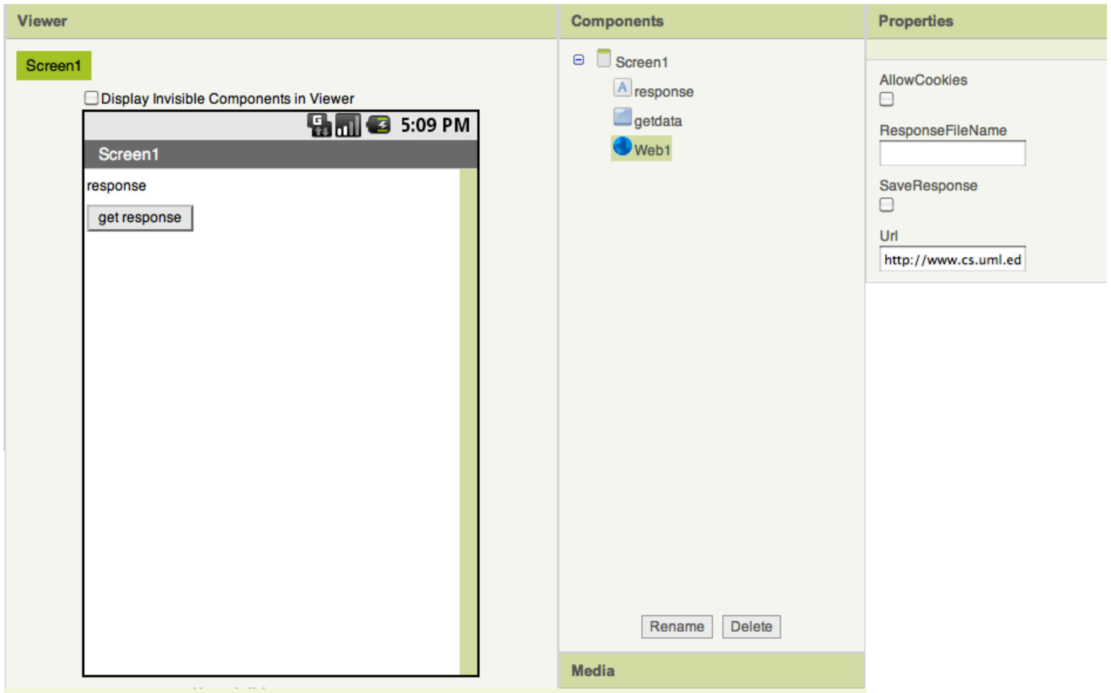
Faking It

If you do not have access to an Android phone or tablet, you will need to fake it, as this feature is not supported in emulation.

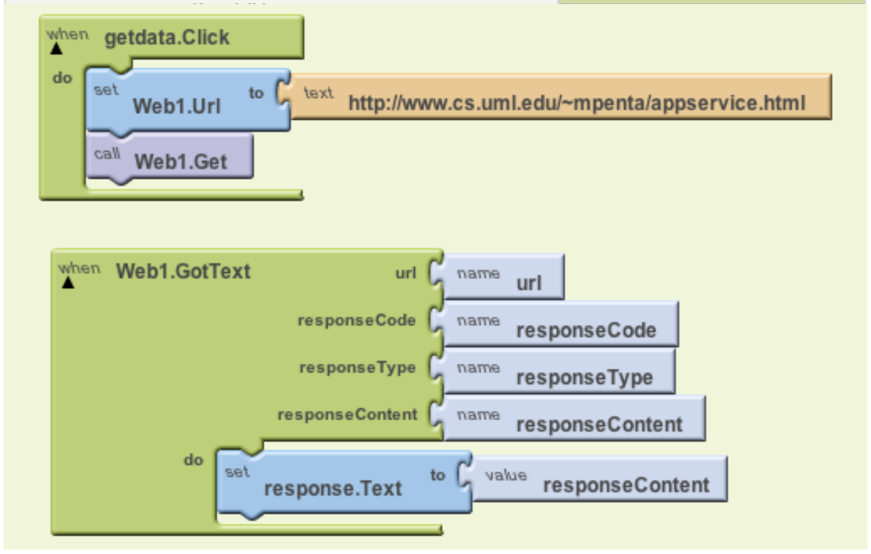
The Dice Roller app uses a button instead of a shake.

Talking to the Web

You can talk to many kinds of web services with AppInventor. This example makes a simple web call when a button is clicked. It receives a small amount of text and displays it in a label.



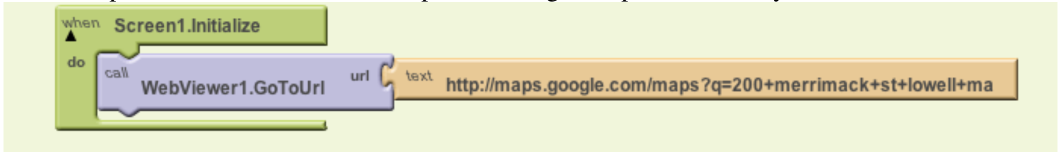
AllowCookies ☐
ResponseFileName
SaveResponse ☐
Url http://www.cs.uml.ed



```
when getdata.Click
do  set Web1.Url to  text http://www.cs.uml.edu/~mpenta/appservice.html
    call Web1.Get

when Web1.GotText  url              name url
                   responseCode     name responseCode
                   responseType     name responseType
                   responseContent  name responseContent
do  set response.Text to  value responseContent
```

Web Viewer

The example sets the web viewer to a specific Google Map location. Any URL can be used for the text.



```
when Screen1.Initialize
do  call WebViewer1.GoToUrl  url  text http://maps.google.com/maps?q=200+merrimack+st+lowell+ma
```

Web Viewer is also used in the location sensor example – See Using Sensors->Using Location