ECE 1778:
Creative Applications for Mobile Devices

Lecture 2

Today

1. Logistics/Organization of Course & Project
2. Introduction to Mobile Phone Environment
   - Android Development Toolkit
   - Basic Concepts
   - List and Files
3. Introduction to the App Inventor Environment
4. Introductions and Ideas, continued
   - Other half of class

Welcome Back: Some Logistics

If you missed the first lecture:

- Please see the first lecture on the course website:
  - http://www.eecg.utoronto.ca/~jayar/ece1778/
  - Look under content
- Please sign up on the sign up sheets
  - Can’t really do much in course if not taking for credit
  - **Apper** = non-programmer
  - **Programmer** = capable of learning new environment fast
  - can be both, which means you can program well and come from an application discipline

Have You Started on the Assignment?

- Programmers: **P1**
  - Any issues?
- Appers: **A1**
  - Any issues?

- This is a lot of work to begin,
  - Necessary so you can do a project!

Assignments Due Next Week

- Both assignments due next week, 10am, Tuesday January 25th
- Submit by email to course TA – braiden.brousseau@utoronto.ca

- If you're doing assignments on iphone, then you must send me a zip file of the full project directory, runnable under XCode 4.2.

Recall: The Goal

- The goal of this course is to bring together people from different disciplines and to build an interesting/creative mobile application
- First Priority is to create those inter-disciplinary groups
  - We have more programmers than Appers,
  - I'd like to encourage 2 Programmers & 1 Apper to Join forces in groups of **3**
  - Reserve the right to add 1 Apper to a group of 2.
- Groups of three programmers will not be allowed

Extra Meeting to Form Groups

- Wednesday January 19th
- 6:30pm-7:30pm
- Sandford Fleming, room B560
  - After today's finishing introductions
  - Will find a way to help make matches there.

Groups of One

- Had several requests to do projects in groups of one.
- Upon consideration, have decided **against** this, for these reasons:
  1. Want the more ambitious projects that are possible with 2 or 3 people
  2. Part of the learning of the course is project work in groups.
  3. Do not want the higher number of projects – the course is big enough already

- So: you will have to find a partner – come on Wednesday night!

Once You Have a Group

- Send email to:
  - me(jayar@eecg.utoronto.ca)
  - the course TA, Braiden Brouseau (braiden.brousseau@utoronto.ca)
- Provide:
  - names,
  - student numbers,
  - mobile platform you plan to do the project on
    - one of Android, iPhone (others require a special discussion)
    - If thinking about using Tablet
    - If you have your own device you can use

Note for iPhone/iPad Users

- Recall you have to have a mac to do this
- The University of Toronto has signed up under the University development program, see:
  - http://www.its.utoronto.ca/communication-and-collaboration/Apple_iOS_Developers_Centre.htm
  - Allows free download to device, which otherwise costs $US 99
  - Does not allow for app store distribution
    - (I assume, though, if you do pay $99 later, you could do this)

Initial Thoughts/Pointers on Project

- You should be thinking of ideas for projects, as precursor to finding and forming your group
  - So you can have something to talk about on Wednesday
- Once you have a group:
  - If **Apper** in group, Apper needs to give rough idea of discipline
  - All groups: start kicking around ideas
  - Send me an email when you think you have something concrete that you can describe
- Create a Plan; be sure to use **Spiral/Agile** approach
  - Begin by making some small version work, and grow, incrementally from there

Programmers:*
Mobile Phones and Android Development

Some Should still be of interest to **Appers**

Mobile Phones are Very Small Computers

Good:

- The most portable computers ever
  - With built in sensors
- Amazing portals to the internet
- Can also make phone calls!

Not so good:

- Very small screens
- No/small keyboard
- Inexact pointing compared to mouse
- Processor speed and memory are slower/tighter than desktop
- Must make sure don’t interfere with a phone*

An Android Application

- Is a series of windows (screens) presented to the user
  - Called ‘**Activities**’ in Android terminology

- Program responds to events
  - e.g. screen touches done by the user
  - e.g. shaking phone
  - event-driven programming vs. procedural

Mobile Programming is *Event-Driven*

- Who is familiar with Event-Driven Programming?
  - Prevalent in graphical user-interfaces
- Different from straight-line procedural programming
  - Executed path is more linear – processing data in -> out
- Event-Driven
  - Flow of program determined by a series of user events
  - Sets up a series of user views
  - Waits to respond to events, such as:
    - User actions: button push, finger move, phone shake
    - System notifications – time elapsed, phone call, notification from internet
- Can be more complex because must handle different interacting patterns of events
  - shake + notification

Other Android Terms

- **Services**
  - Longer running processes
  - e.g. continuing music play; monitoring web page
- **Intents**
  - Messages that notify applications of significant events, e.g.
  - SD card inserted into phone
  - User has arrived within 100 meters of geographic location
- **Content Providers**
  - Abstract data storage, made available to multiple applications
    - How applications communicate with each other
    - e.g. contacts or photos are content providers

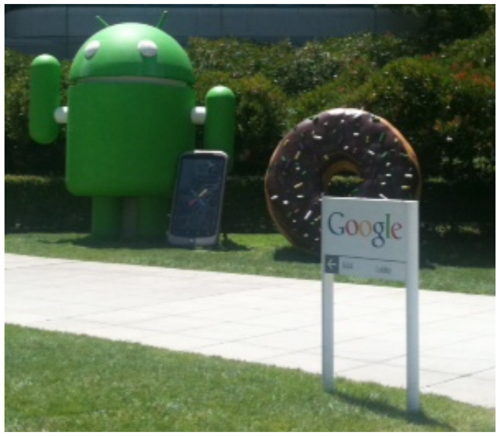
Projects and Targets

- To create an Android Application, must first create a **project**
  - Software directories that contain all of the files relating to the application

- Key element: The **manifest file**
  - AndroidManifest.xml
  - Describes what parts of the device you'll use
    - Some require user permission, e.g. GPS
  - Also which version of Android operating system/APIs

Android Versions

- Google rapidly evolves Android:
  - 1.5 May 2009 = 3
  - 1.6 October 2009 = 4
  - 2.0/2.1 January 2010 = 5/6/7
  - 2.2 May 2010 = 8
  - 2.3 December 2010 = 9
  - 3.0 later in 2011
- Each version has a name, too, usually has a name, in order: Cupcake, Donut, Éclair, Froyo and Gingerbread and Honeycomb



Project Structure

- A new Android project has the following structure:

- **AndroidManifest.xml**, an XML file describing the application being built and what components – activities, services, are being supplied by that application
- **build.xml**, an **Ant** script for compiling the application and installing it on the device
- **default.properties** and **local.properties**, property files used by the Ant build script

Project Structure, cont'd

- **assets/**, static files you wish packaged with the application for deployment onto the device
- **bin/**, holds the compiled application
  - bin/classes/ compiled Java classes
  - bin/classes.dex executable created from compiled Java classes
  - bin/yourapp.ap_ holds your application's resources, packaged as a ZIP file (where yourapp is the name of your application)
  - bin/yourapp-*.apk is the actual Android application (where * varies)
- **gen/**, **generated** source code (by compiler)
- **libs/**, third-party Java JARs
- **src/**, your Java source code

Resources in Project File

- **res/**, "resources" - icons, GUI layouts
  - res/drawable/ for images (PNG, JPEG, etc.)
  - res/layout/ for XML-based UI layout specifications
  - res/menu/ for XML-based menu specifications
  - res/raw/ for general-purpose files
  - res/values/ for strings, dimensions, and the like
  - res/xml/ for other general-purpose XML files you wish to ship

APK File

- The .apk file is the application
- It is a ZIP archive containing
  - the .dex file, the compiled edition of your resources (resources.arsc),
  - any un-compiled resources (such as what you put in res/raw/) and the
  - AndroidManifest.xml file.

Targets

- The 'Target' of your application is either an actual phone your want to run it on, or the emulator
  - The emulator is a software program running on the desktop that looks and acts like an Android phone
  - You'll all use it to initially test your programs/apps
- Emulator is called an **'Android Virtual Device'** or **AVD**
- There is some work in creating the device, as you have to specify various attributes of the fake phone, such as
  - Size of SD card memory
  - Which version of Android using
  - Size of screen

What Programmers Should Be Learning

- With Assignment 1:
  - After downloading the various elements of the programming environment
- Java basics if not already known
  - http://en.wikibooks.org/wiki/Java_Programming/Language_Fundamentals
  - Or some basic Java Text
  - I liked John Carter, ‘**Using Java**’
- Working within Eclipse
  - or, can choose to do everything in command/shell environment
  - lose some of Eclipse’ good features
- Running the basic environment
- Understanding File Types in the Android Project

Then, Closer to the Real Stuff

- Making a Simple XML Layouts
  - How to arrange
- Basic Widgets:
  - Labels, Buttons, Images,
  - checkbox, radio buttons
- Methods common to many of these, e.g.
  - setEnable(),
  - isEnabled();
  - Changing colour, text etc.
- Once handy with this, Assignment P1 is straightforward
- Eclipse & Emulator are somewhat buggy…

Things to Demonstrate

- Eclipse Startup
- New Project
- Creating new Android Virtual Device (AVD)
- Running a project
- Placing a single widget
  - XML description
  - Switching between graphic view and XML in Eclipse
  - Properties
- Connection to Java Code through findViewById (R.id.*XXX);*

Widgets

- Button, ImageButton
  - Button to press, with special image
- Textview
  - Basic text label, changeable
- Imageview
  - Basic picture
- EditText
  - for entering text fields
- CheckBox
  - Ticking off an entry
- Radio Buttons

Useful Methods

- toggle if a widget is enabled via setEnabled()
- see if it is enabled via isEnabled().
  - One common use pattern for this is to disable some widgets based on a CheckBox or RadioButton selection.

- give a widget focus via requestFocus()
- see if it is focused via isFocused().
  - You might use this in concert with disabling widgets as mentioned above, to ensure the proper widget has the focus once your disabling operation is complete.

Appers*: Google App Inventor

*Will still be of interest to **Programmers**

App Inventor

- Google App inventor is an attempt by Google to allow people without programming backgrounds to create apps for Android phones
- It works reasonably well
- We’re going to use it for the ‘Appers’ to give you a sense of how things work inside the phone
  - You may find it is something you can work with as well
  - It could help you with the layout and plans for the ultimate app your project group will build

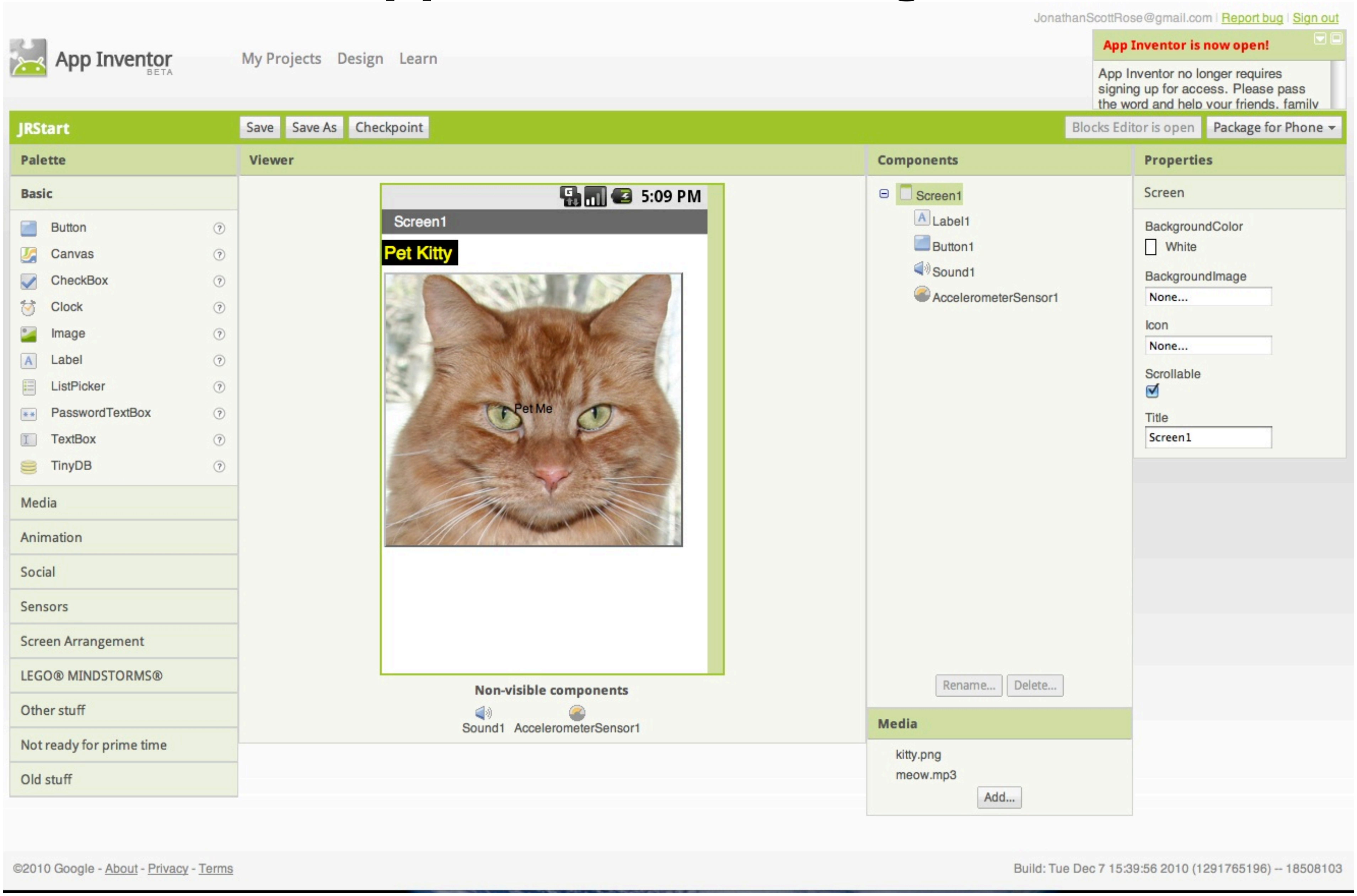
Two Screens

1. Designer
   - Where you show what each screen contains
   - Visual Components – buttons, pictures
   - Non-Visual: sounds, shaker detection
2. Blocks Editor
   - Write a ‘visual’ program
   - Blocks can be related to
     - The blocks put down in the designer – e.g.
       - ‘When button clicked’
       - Play sound
     - Built-in: math, logic, control,

Demonstration

- Hello Bark

Demo of App Inventor - Designer



App Inventor is now open!

App Inventor no longer requires signing up for access. Please pass the word and help your friends, family

BackgroundColor
☐ White

BackgroundImage
None...

Icon
None...

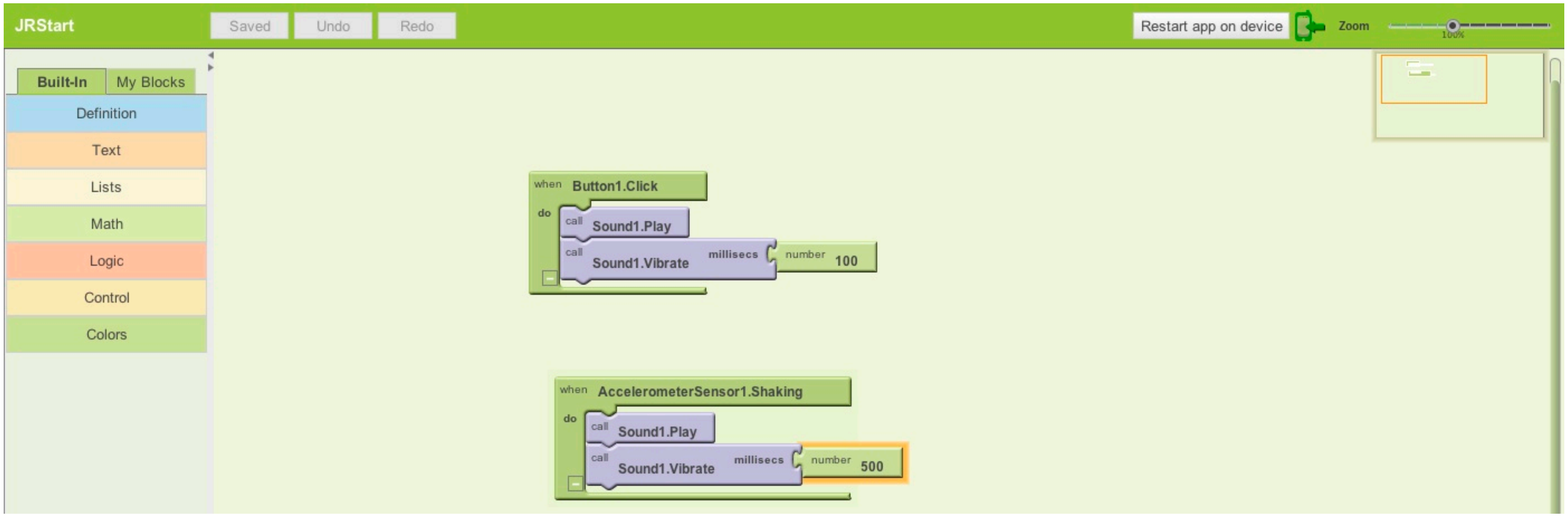
Scrollable
☑

Title
Screen1

©2010 Google - About - Privacy - Terms

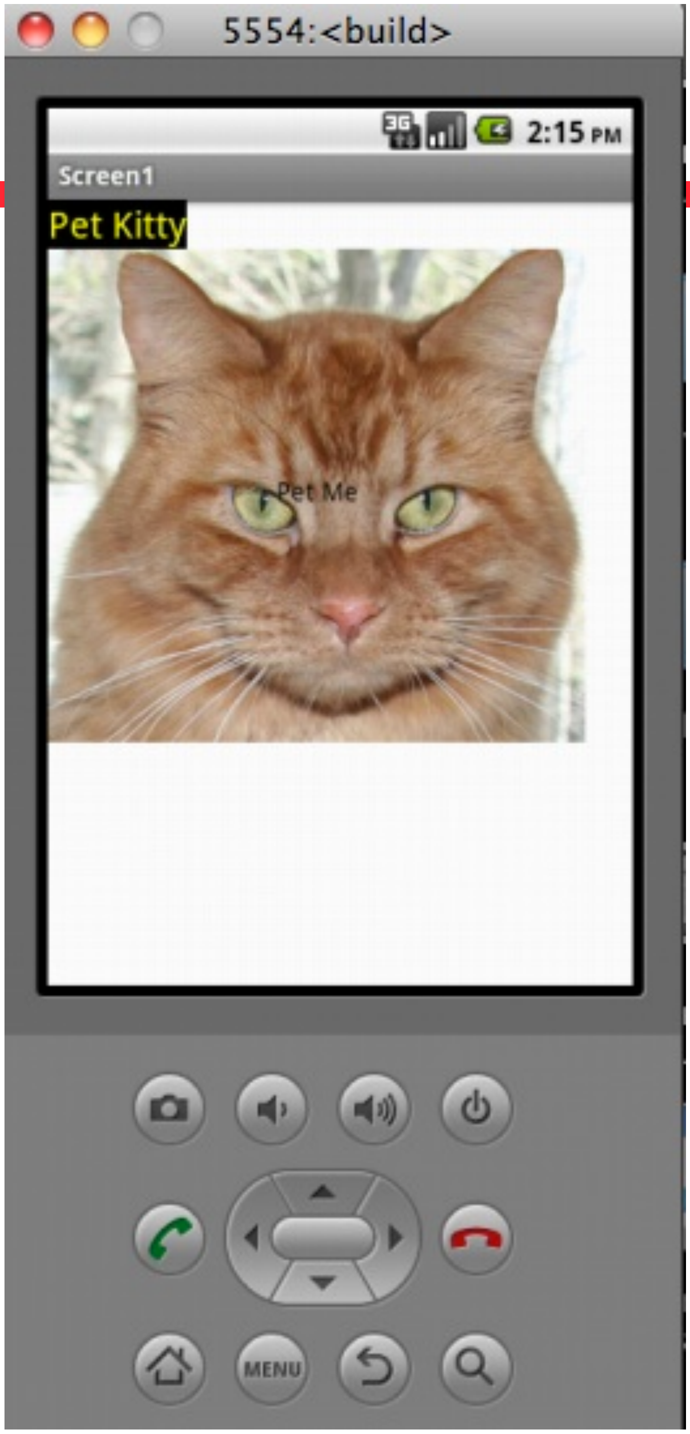
Build: Tue Dec 7 15:39:56 2010 (1291765196) -- 18508103

Blocks Editor



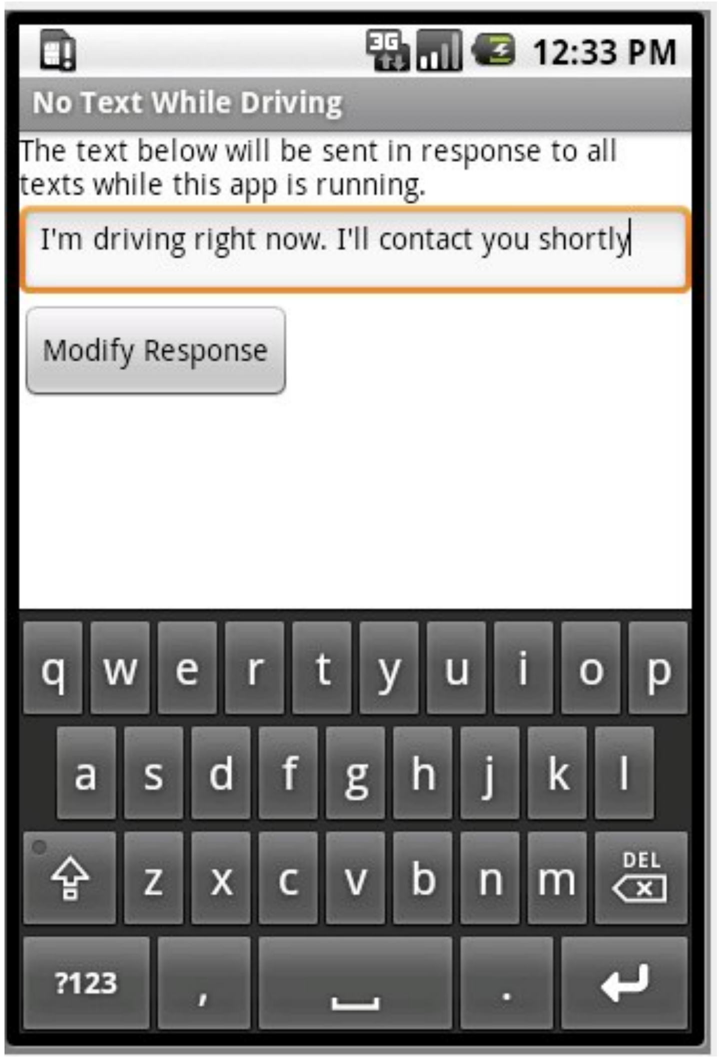
```
when Button1.Click
do  call Sound1.Play
    call Sound1.Vibrate millisecs  number 100

when AccelerometerSensor1.Shaking
do  call Sound1.Play
    call Sound1.Vibrate millisecs  number 500
```

App Inventor Emulator



No Text While Driving Application

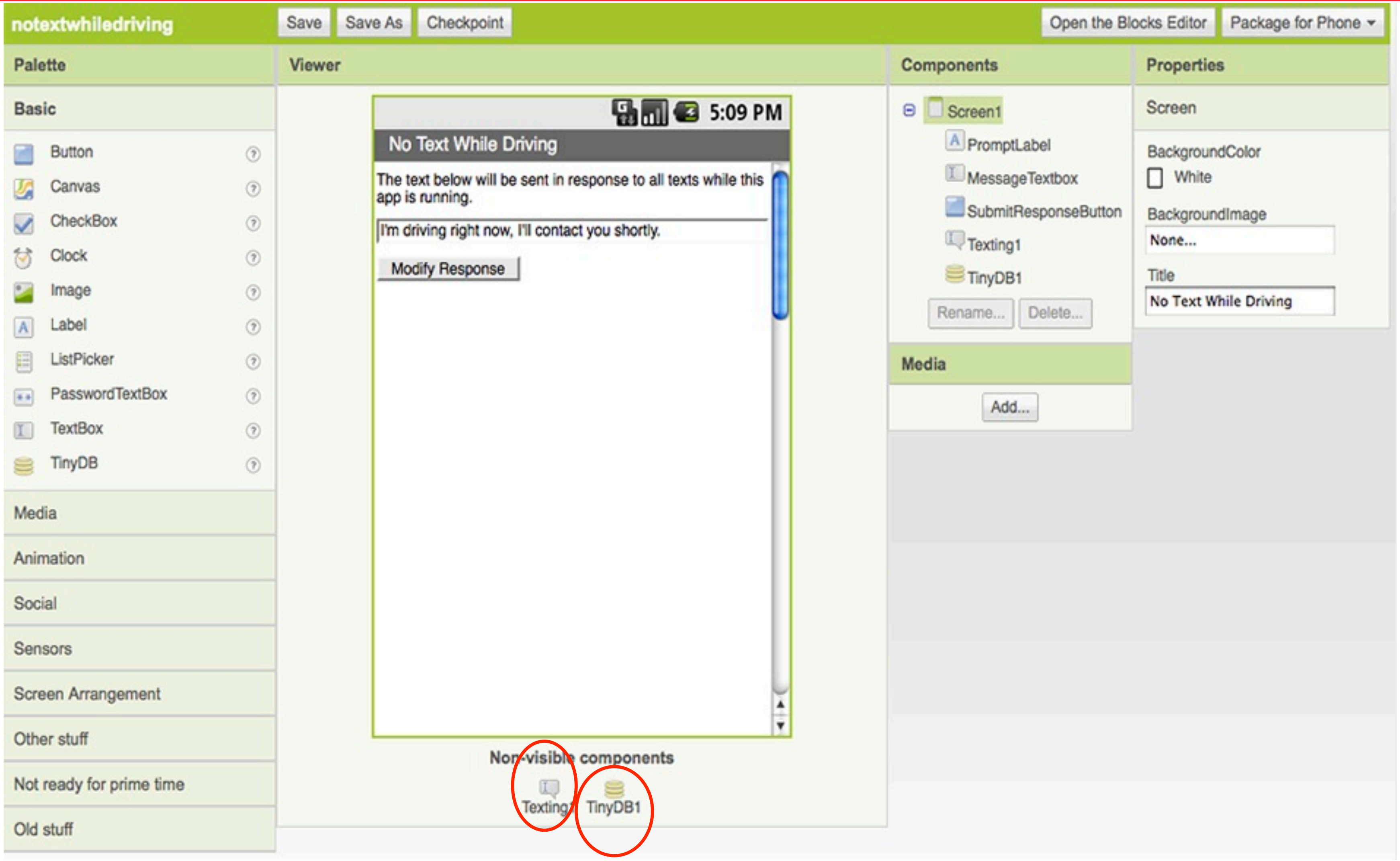
- Automatically responds to SMS Text message with a message.


No Text While Driving

The text below will be sent in response to all texts while this app is running.

I'm driving right now. I'll contact you shortly

Designer



notextwhiledriving

5:09 PM

No Text While Driving

The text below will be sent in response to all texts while this app is running.

I'm driving right now, I'll contact you shortly.

Modify Response

Non-visible components

Texting1 TinyDB1

Components:
- Screen1
  - PromptLabel
  - MessageTextbox
  - SubmitResponseButton
  - Texting1
  - TinyDB1

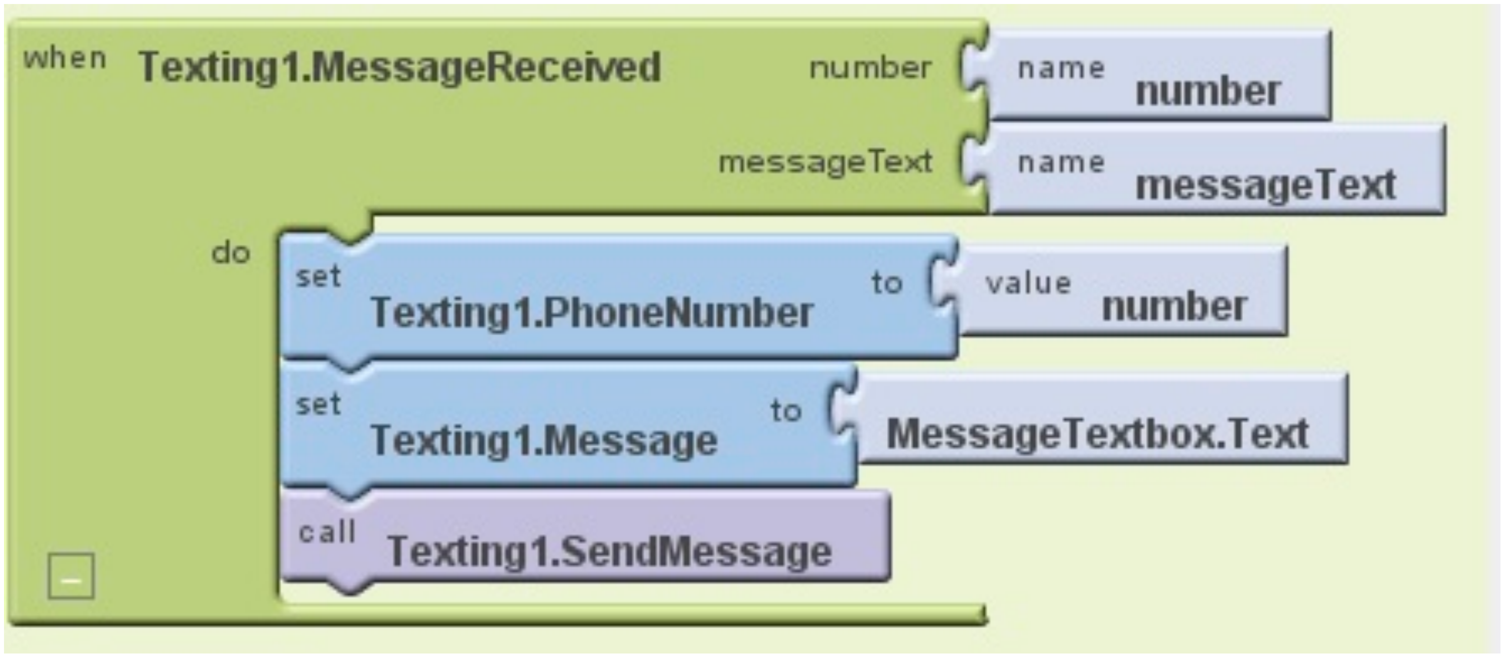
Properties: Screen

BackgroundColor: White

BackgroundImage: None...

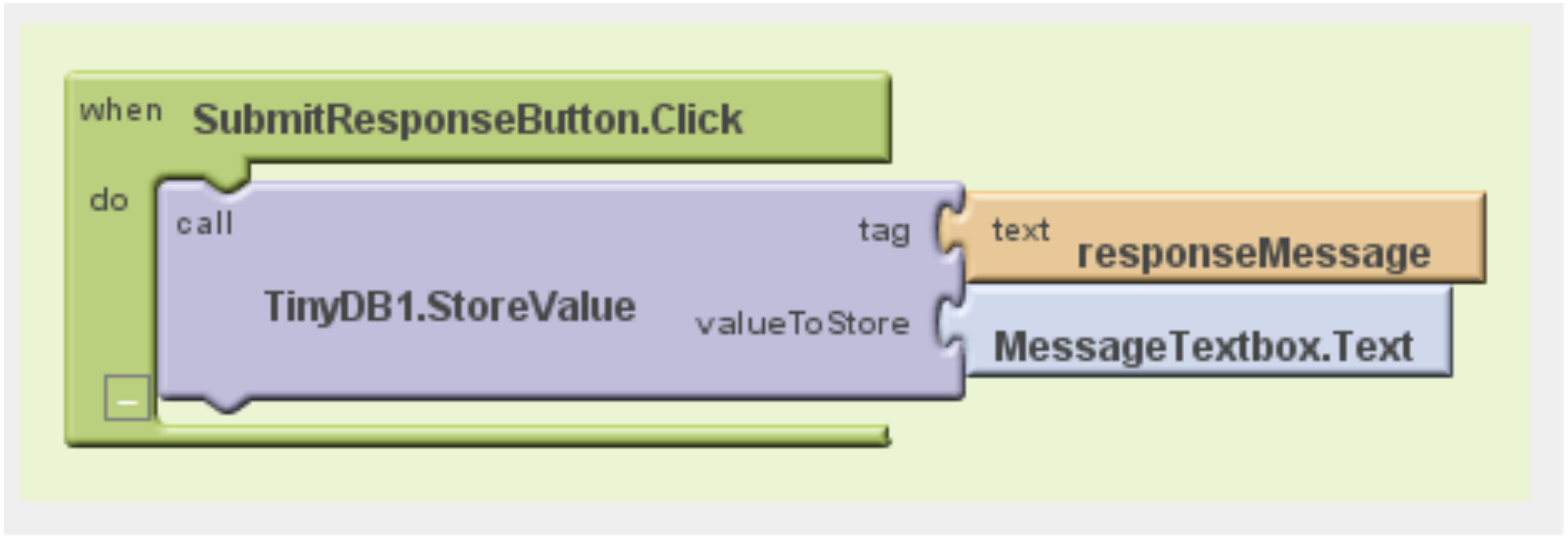
Title: No Text While Driving

Blocks: Texting Block



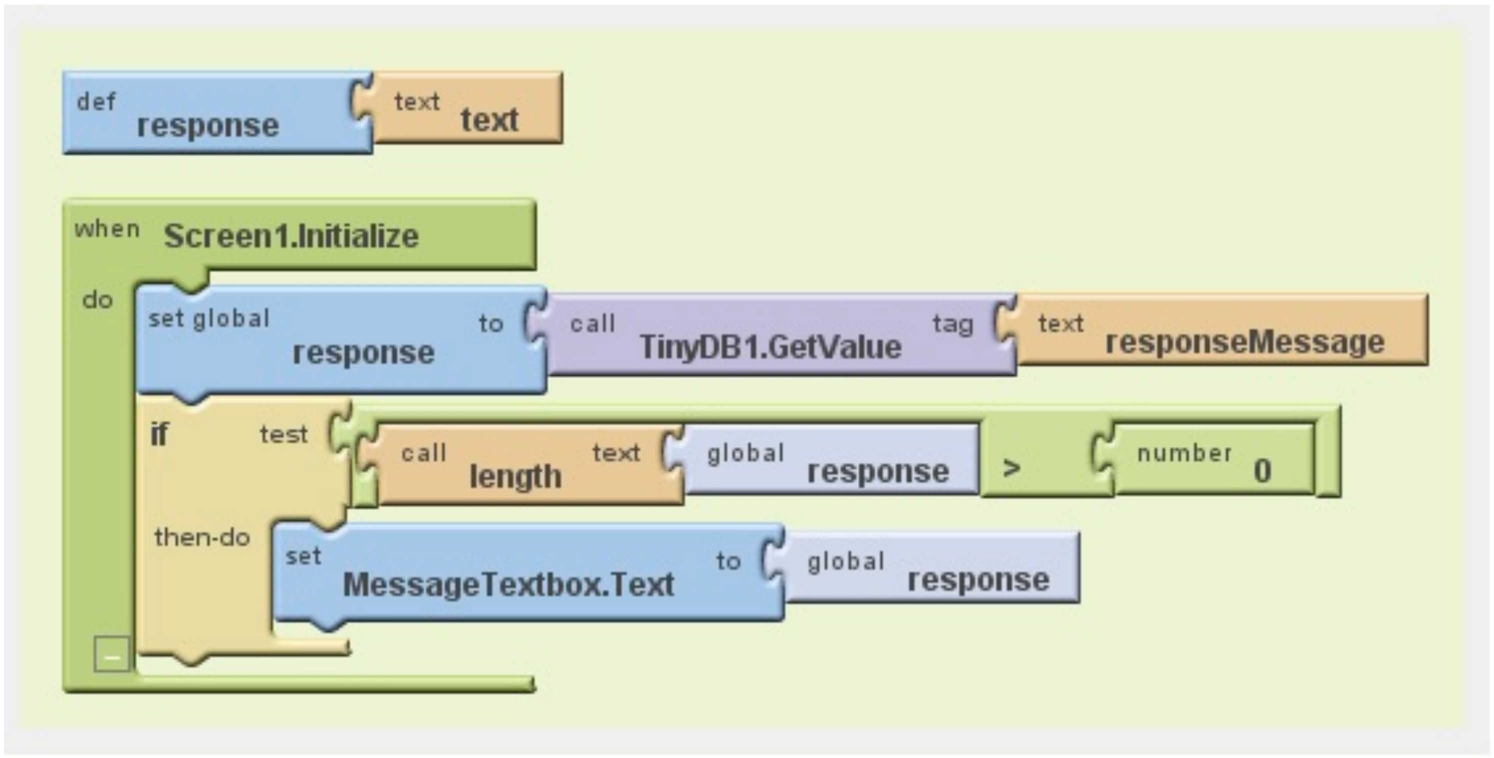
```
when Texting1.MessageReceived    number      name number
                                 messageText name messageText
do  set Texting1.PhoneNumber to value number
    set Texting1.Message to MessageTextbox.Text
    call Texting1.SendMessage
```

Store New Response in Data Base



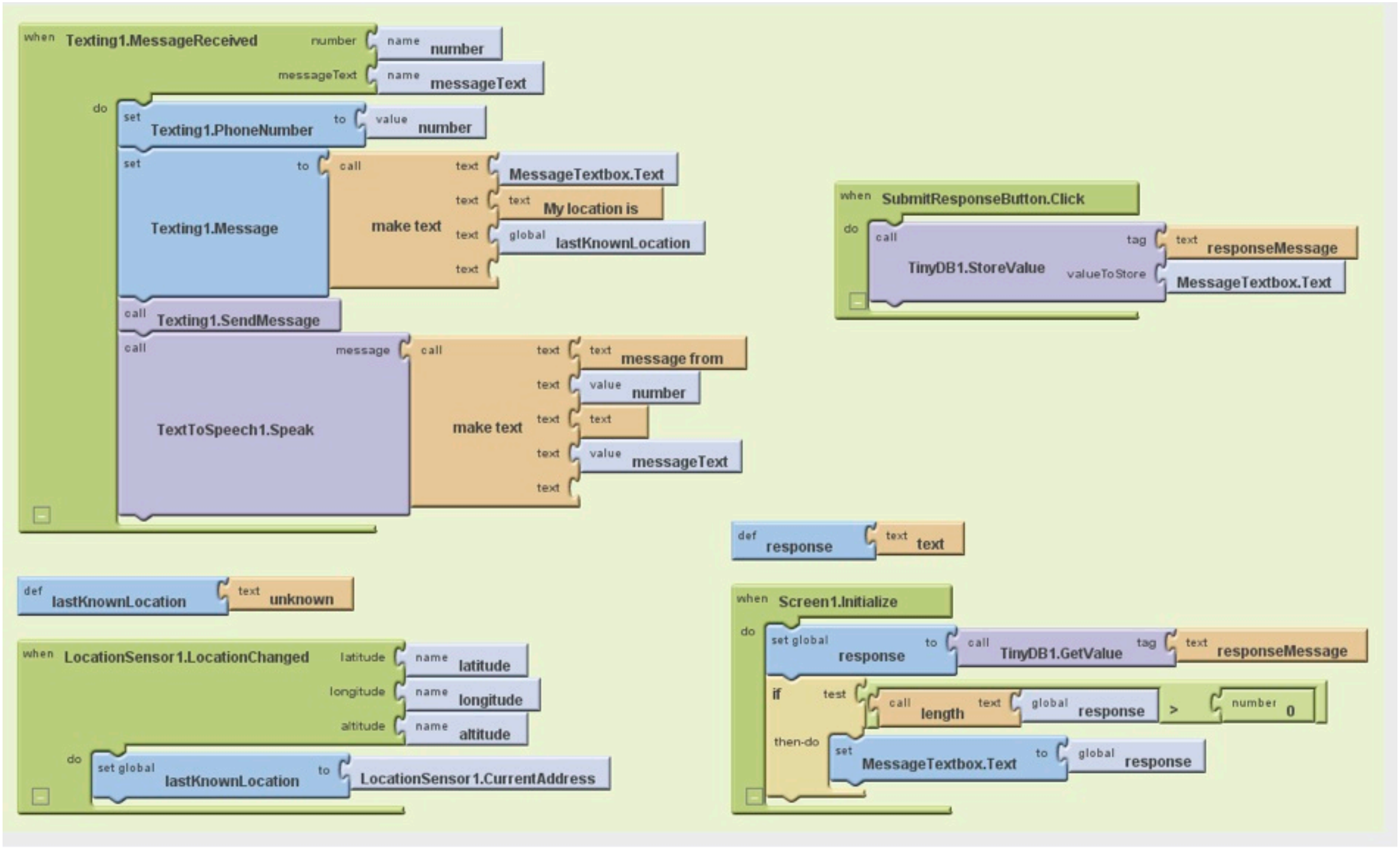
Initialize the Response on Startup

- When screen starts



```
def response  text text

when Screen1.Initialize
do  set global response to  call TinyDB1.GetValue  tag  text responseMessage
    if test  call length  text  global response  >  number 0
    then-do  set MessageTextbox.Text to  global response
```

Better: Speak Texts and Locate



```
when Texting1.MessageReceived
    number      name number
    messageText name messageText
  do
    set Texting1.PhoneNumber to value number
    set Texting1.Message to call make text
        text MessageTextbox.Text
        text text My location is
        text global lastKnownLocation
        text
    call Texting1.SendMessage
    call TextToSpeech1.Speak
        message call make text
            text text message from
            text value number
            text text
            text value messageText
            text

def lastKnownLocation text unknown

when LocationSensor1.LocationChanged
    latitude  name latitude
    longitude name longitude
    altitude  name altitude
  do
    set global lastKnownLocation to LocationSensor1.CurrentAddress

when SubmitResponseButton.Click
  do
    call TinyDB1.StoreValue
        tag text responseMessage
        valueToStore MessageTextbox.Text

def response text text

when Screen1.Initialize
  do
    set global response to call TinyDB1.GetValue tag text responseMessage
    if test call length text global response > number 0
    then-do
        set MessageTextbox.Text to global response
```

Introductions, continued

To help in Project Group-forming

Introductions, Continued

- Last Day, half of the class introduced themselves
- Let’s do the other half, hopefully sitting on the same side

- Please take notes for people who you think might be compatible partners
- On Wednesday night, we’ll try to put people in some categories to help you explore matches.

- Don’t forget, the priority has to be on matching to Appers

Introduce Yourself, Round 2

1. Name
2. Taking Course for Credit – yes, no, maybe
3. What discipline you work in & degree sought
4. What your thesis topic is (if doing thesis)
5. If you work, where.
6. Why you’re taking this course
7. What idea you have for an app.

Don’t Forget: Meeting to Form Groups

- Wednesday January 19th (Tomorrow)
- 6:30pm-7:30pm
- Sandford Fleming, room B560
  - After today’s finishing introductions
  - Will find a way to help make matches there

- Sandford Fleming is building south of Con Hall
- B560 is in basement, south side
  - In middle of Galbraith-Sandford Fleming block