March 2013 ISSN: 2320-1363

# A Blocks Language for Mobile Phones: Android

**Ram Narayan Dash**

**ramnarayancbit@gmail.com**

**Abstract**

Point your phone at the sky, and Google's Skymap overlays information about the celestial objects in view. It is an example of how mobile technology is helping to bring science out of the classroom and lab and into the real world. The potential for science, and science-related apps, is unlimited. Google's Skymap was created by expert engineers; as are all apps, as the mobile app frontier is open only to a select group, open only to that .0001% of the world who can program in Objective-C (iPhone) or the Java SDK (Android). Like the web a decade ago, the mobile app world is read-only, a medium where most are consumers, not producers, of information. If you have an idea for an app you needed to be a programmer, or be able to pay one. It's unfortunate that scientists, and artists and domain-experts in all disciplines, cannot manifest their ideas directly. Fortunately, this closed world is in the midst of being opened to the masses. In July 2010, Google announced App Inventor for Android, an end-user tool for developing apps called it "Do-it-yourself App Creation Software" alon.com referred to it as a "Hypercard" for the smart-phone . Based on a blocks programming paradigm that had proven successful for kids, App Inventor dramatically lowers the barriers to becoming an app developer.

**Keywords:** Objective C, Block, Components

## 1. Introduction

### 1.1 Problem Definition

App Inventor is a visual, drag-and-drop language for building mobile apps on the Android platform. You design the user interface and components of an app using a web-based graphical user interface (GUI) builder, then specify the app's behavior by piecing together blocks as if you were working on a puzzle. The tool is free to use and runs in the "cloud" so is accessible from any browser. The blocks language provides a low-threshold entry to programming. The elimination of most typing dramatically reduces the frustrations beginners have with syntax, the blocks use color and visual cues to help avoid problems, and only some blocks lock into place, further eliminating the possibility of error. The blocks language paradigm has been proven successful for end-user

March 2013 ISSN: 2320-1363

programming, which is, opening up software development to a much larger pool than just computer scientists. It first gained popularity in the Lego Mindstorms robot programming environment. More recently, the Scratch language was introduced as a way for kids to create animated stories on the web. Because of its easy to use blocks language and burgeoning community site for sharing programs, developers have built and uploaded over a million programs on the Scratch site. With App Inventor, Google engineers have applied this same blocks language paradigm to general-purpose mobile app development.

### 1.2 Details about the Problem

The App Inventor language targets the full functionality of an Android device. App Inventor blocks do things like process incoming SMS texts, interface with the GPS location sensor of the phone, scan barcodes, and talk to web APIs. Of course a tool that lets anyone program a mobile phone or tablet has generated a lot of excitement. When Google announced its invite-only, beta version in July 2010, they received an overwhelming number of requests—the months following were spent preparing the server infrastructure for this huge user base that had emerged. The potential is there to bring a whole new class of "end-user" programmer into the mix, changing the mobile world from a read-only one where app development is limited to hardcore programmers, to a read-wrote mobile world where anyone can create new apps or customize existing ones.

Now consider the following App Inventor blocks, which I use as an initial sample in my App Inventor courses .This sample auto-responds to a text received by the phone, sending back the text, "I am driving right now, I will contact you later". The blocks are understandable, like with Scratch. They also have meaning in the real-world.

March 2013 ISSN: 2320-1363

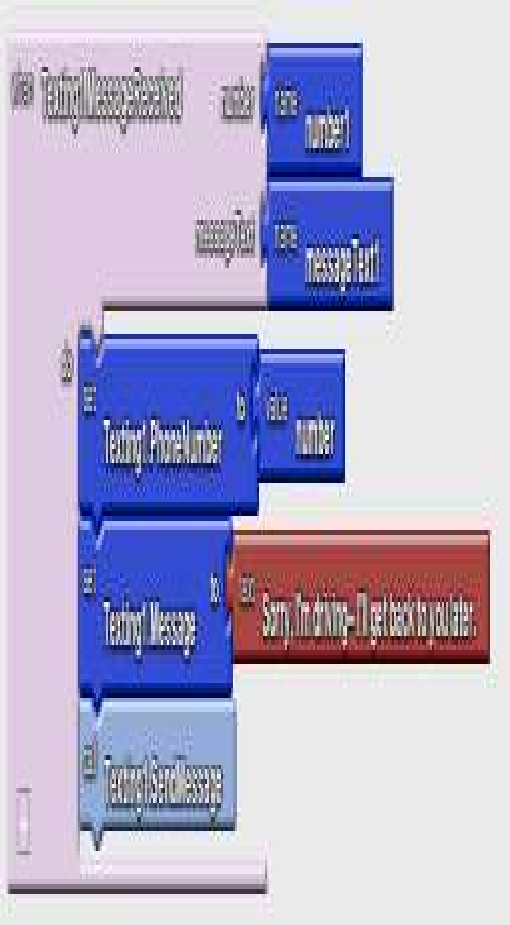

**Fig.1.2.1** Block in the perpetual Mode to couple with the tool.

The App Inventor programming environment has two windows: the component designer and the blocks editor. You can also plug a phone into the computer to view and test your app live during development. The Component Designer is a fairly typical WYSIWYG tool for designing user interfaces—you drag out buttons, labels, and textboxes from a palette into a view, modify component properties like background color and width, and in general specify how the app will look. You can also drag out the non-visual components you will need in your app, such as a Texting and Text-to-Speech components. App Inventor provides about thirty components separated into various palettes.

## 2. Related Work

**Basic**- the user interface components like button, label, textbox, image, and canvas as well as a component for persistent storage.

**Media**- components for playing sound files and video, as well as a component for interacting with the phone's camera.

**Animation**- a ball and general image sprite, both of which have high-level functions for movement and object-interaction.

**Social**- components for making phone calls, processing and sending SMS text messages, interacting with the phone's contact list, and a sample API component that interfaces with Twitter.

**Sensors**- components for sensing the phone's location, orientation, and acceleration.

**Screen Arrangement**—components for organizing user interface objects on the screen.

**Details about the set applied**

March 2013 ISSN: 2320-1363

Once the components of the app have been specified in the Component Designer, the behaviors are defined in the Blocks Editor. The Blocks Editor has two palettes, one for built-in functionality separate from the components, and another, “My Blocks”, which contains blocks created for each component added to the app. The built-in blocks allow you to do many of the things you do with a normal programming language, including defining variables and specifying conditional and iterative behaviors. The built-in blocks are separated into drawers.


**Fig2.1** Whirling Butterfly

**Definitions**-blocks for defining variables and procedures.

**Text**-text (string) functions, e.g. join, split

**Lists**-list functions, e.g., addItem, selectItem

**Math**-math functions

**Logic**-logical operators

March 2013 ISSN: 2320-1363

**Control-**if, ifelse, foreach, while

**Colors**-a palette of colors

The My Blocks section contains a drawer of functionality for each component that was added to the app. This adds to the concrete nature of the environment as the developer works with instances and not classes.

### 3. Methodology

Each component has event-handler blocks, functions, and property setters and getters. So, for instance, a (SMS) Texting component has a Message Received event-handler block containing a hole that can be filled in with the function blocks that should be executed in response. It also has a function Send Message, and blocks for changing or reading all of its properties.

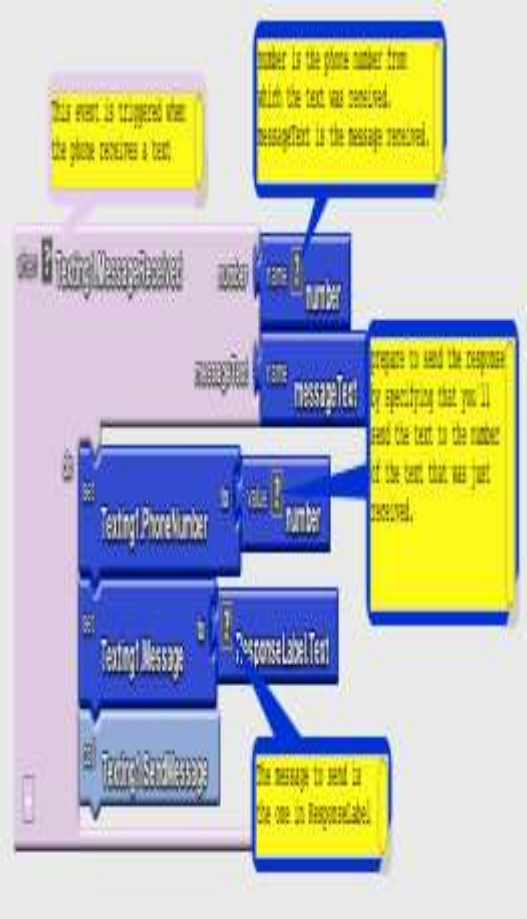


**Fig3.1** Building "No Text While Driving"

This section demonstrates how to build "No Text While Driving", including the text-to-speech and location sensing capabilities. We will build the app step-by-step, starting with the original auto-response. You can follow along and build the app with App Inventor, or just read along to get an idea for how App Inventor works. The reader is referred to for a tutorial version of this app with more detailed instructions.

March 2013 ISSN: 2320-1363

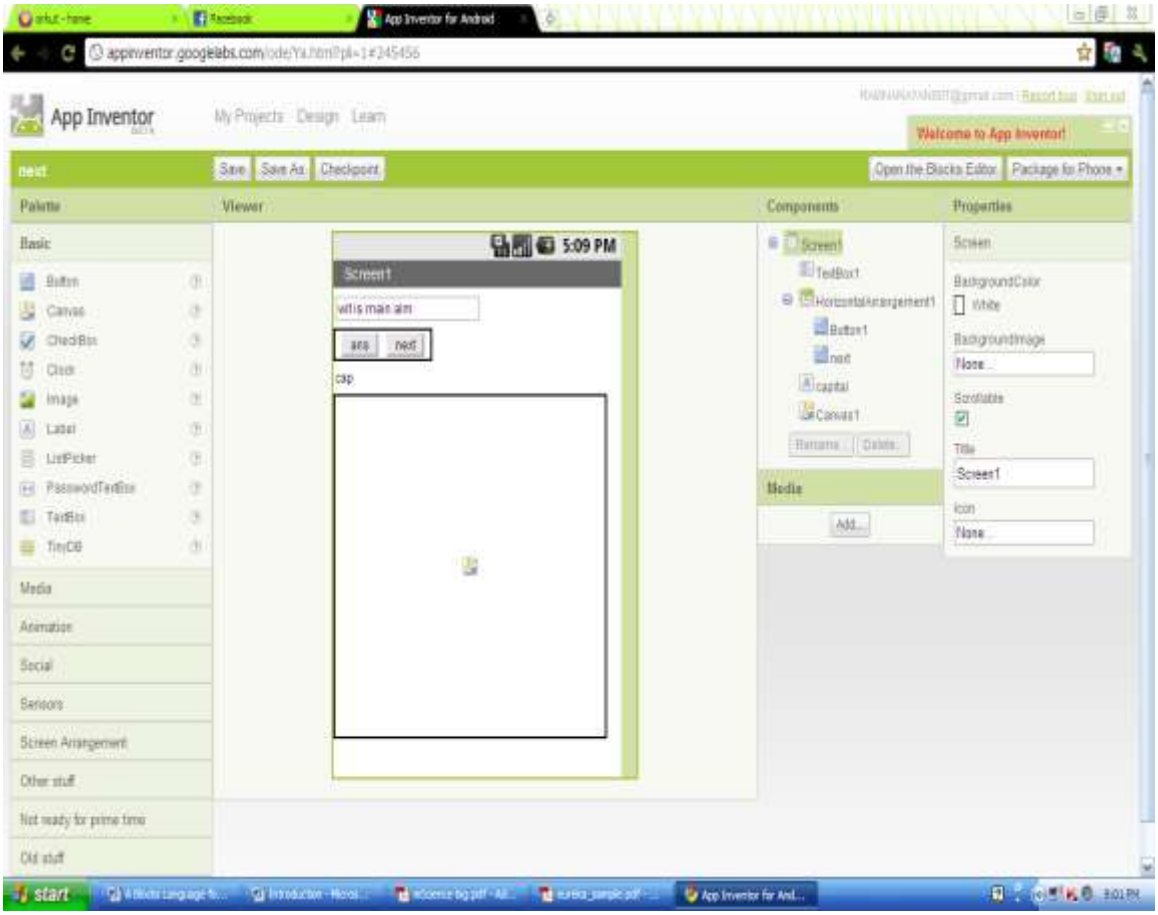


**Fig3.2** the Components of "No Text While Driving"

### 3.1 Results and discussion

This behavior works in consort with the LocationSensor1.LocationChanged event and the variable last Known Location. Instead of directly sending a message of the text in ResponseLabel.Text, the app first builds a message using make text. It combines the response text in ResponseLabel.Text with the text " My location is " and then the variable lastKnownLocation.The default value of the variable LastKnownLocation is "unknown", so if there has n' t yet been a reading by the location sensor, the second part of response message will contain the text "My last known location is: unknown".

March 2013 ISSN: 2320-1363

Adding Location Information to the ResponseSMS Text:

App's like Facebook;s Place and Google's Latitude use GPS information to help people track each other's location. There are major privacy concerns with such apps, one reason being that location tracking kindles people's fear of a "Big Brother" apparatus that a totalitarian government might set up to track its citizen'swhereabouts. Of course there are more personal concerns as well many people are happy keeping there where about private unless someone asks them where they are and they choose to answer. Of course location information can be quite useful as well—one thinks of a lost child or hikers lost in the

woods. Within the context of, "No Text While Driving", the "driver" location information can be used to convey a bit more information in response to incoming texts. Instead of just sending back, "I'm driving", the response can be something like "I'm driving and I;m at 3413 New Delhi". For someone awaiting the arrival of a friend or loved one, this extra information can allow them to plan accordingly.

App Inventor provides the LocationSensor component for interfacing with the GPS functionality of the phone. GPS stands for Geographical Positioning System. Besides latitude and longitude information, the LocationSensor also will provide the current street address; making use of Google Maps information to do so.The LocationSensor doesn't always have a reading. For this reason, care needs to be taken to use the component properly. LocationChanged.event occurs when the phone's location sensor first gets a reading, or when the phone is moved to produce a new reading. The scheme will use for this app is to respond to the LocationChanged event by placing the current address in a variable lastKnownLocation. Later, will change the response message to incorporate this address.Here's how the blocks should look:Figure Tracking the phone's location

March 2013 ISSN: 2320-1363

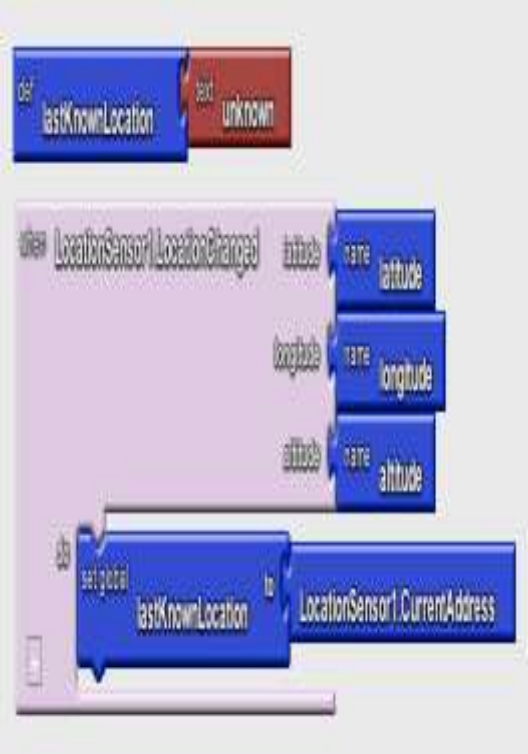


**Fig3.1.1** Tracking the phone's location

## 4. Conclusion

App Inventor is a right-tool and the right-time technology. Scratch has popularized, refined, and proven the viability of a blocks programming language for end-users, mobile apps are the current craze with Android specifically growing astronomically, and tablets have captivated out collective imaginations. Blogs, wikis, and the technical expertise of today's youth have changed the culture so that people now expect to produce and not just consume Information and why should the app world be any different. App Inventor's appearance at this time in history at the confluence of these trends could lead to the next evolutionary stage in people's ability to control and make use of technology. For scientists, App Inventor can help them expand horizons and integrate the lab, the outside world, and the virtual world.

## 5. References

[1] App Inventor Launch Announcement, http://googleblog.blogspot.com/2010/07/app-inventor-for-android.html, July 12, 2010

[2] Steve Lohr, Google's Do It Yourself googleblog.blogspot.com/2010/07/app-inventor-for-android.html, July 12, 2010.

[3] Dan Gillmor, Roll Your Own Apps, http://www.salon.com/technology/android/index.html?story=/tech/dan_gillmor/2010/07/12/ app_inventor_cloud change mobile programming, july 12, 2010.

[4] lego http://mindstorms.lego.com

[5] Scratch, http://scratch.mit.edu

[6] Mitchel Resnick , John Maloney , Andrés Monroy-Hernández , Natalie Rusk , Evelyn Eastmond , Karen Brennan , Amon Millner , Eric Rosenbaum , Jay Silver , Brian Silverman , Yasmin Kafai, Scratch: programming for all, Communications

March 2013 ISSN: 2320-1363

of the ACM, v.52 n.11, November 2009 [doi>10.1145/1592761.1592779]

[7] Scratch, Whirling Butterfly Example, http://info.scratch.mit.edu/node/

[8] App Inventor API, http://appinventorapi.com