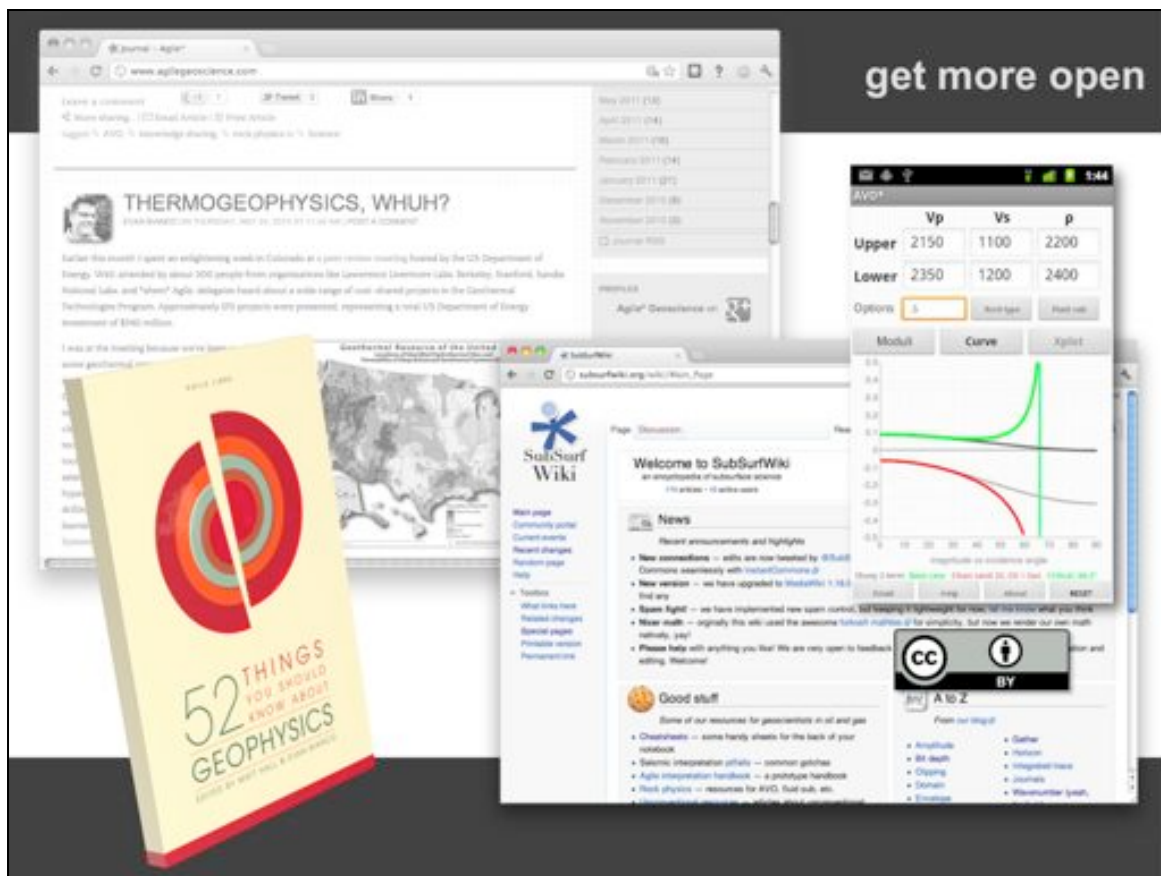




This is a version of a talk I gave at the European Association of Geoscientists and Engineers workshop **Open source E&P software — 6 years later**, organized by Karl Schleicher of the University of Texas at Austin and BEG. The talk was a progression from the IPIMS open source workshop in Houston in June 2011.

I have blogged a bit about this workshop at <http://www.agilegeoscience.com/journal/2012/6/12/two-decades-of-geophysics-freedom.html>.

I have removed the builds for this version of the talk. The slides and the notes are shared under the terms of Creative Commons Attribution 3.0 unported, or CC-BY — except where noted. Some of the images have different license terms. Please respect those terms.



For us at Agile, openness is part of what we do. Our blog and wiki are licensed under permissive Creative Commons terms. Our small software projects also use permissive licenses (BSD-style). We've just done a small book about Geophysics — all 39 authors agreed to license the work as CC-BY.

A lot of what we do is 'knowledge sharing'. Indeed all this — the blog, the wiki, the book, and even the apps — I consider to be 'knowledge sharing'.

So open mobile software is a natural extension for us. There's no deep thinking here about a business model or making money from any of this — we do everything for fun. At least at first...



So what? What's so great about mobile devices?

Well, here are some reasons I've heard. I'd subscribe to most of them myself. When it comes to the workplace, I think it's worth building these little apps because lots of people have these devices with them all the time, in meetings, at coffee, on the train, etc. I like the idea that we can put a little bit of science in the geoscientists pocket this way, and help people follow through on ideas or questions *right now*. Not later when you're back at your desk with Excel or whatever, but now the problem doesn't seem as pertinent or urgent...

And it's fun to be able to wonder about something like an AVO response, or volumetric scenario, and check it immediately and easily. The back-of-the-digital-envelope is what I called it in a blog post recently.

There are always doubters, of course. Sometimes you can see their point if view: these devices are light and cheap and fun...



... but they're light, cheap and fun. They're not serious, not for business, not for science. They're gimmicky toys. A fad.

"Put it away and stop fiddling at the table."

From a developer's point of view, there are issues too: lots of devices, display resolutions, a very rapid development cycle for the OS, tweaked operating environments.

And everything has to be free.

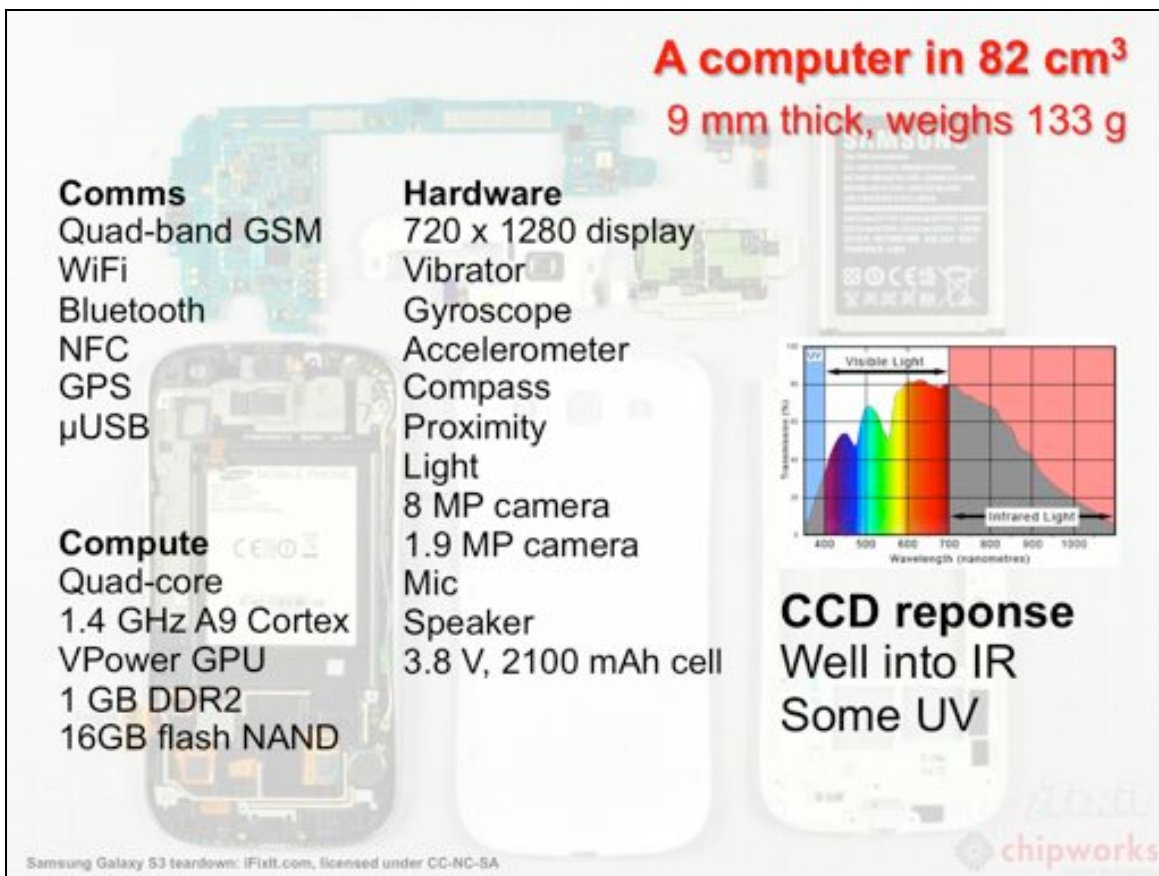


This mobile phone is 25 times smaller, in terms of volume, than my laptop.

So what's in these things? What are they made of?

You don't need me to tell you how incredible the technology is...





Around 8 or 10 sensors, depending on what you count. A GPS with compass and gyroscope. Two cameras. 16 GB storage. Four communications protocols. This thing can even make phone calls. To Australia!

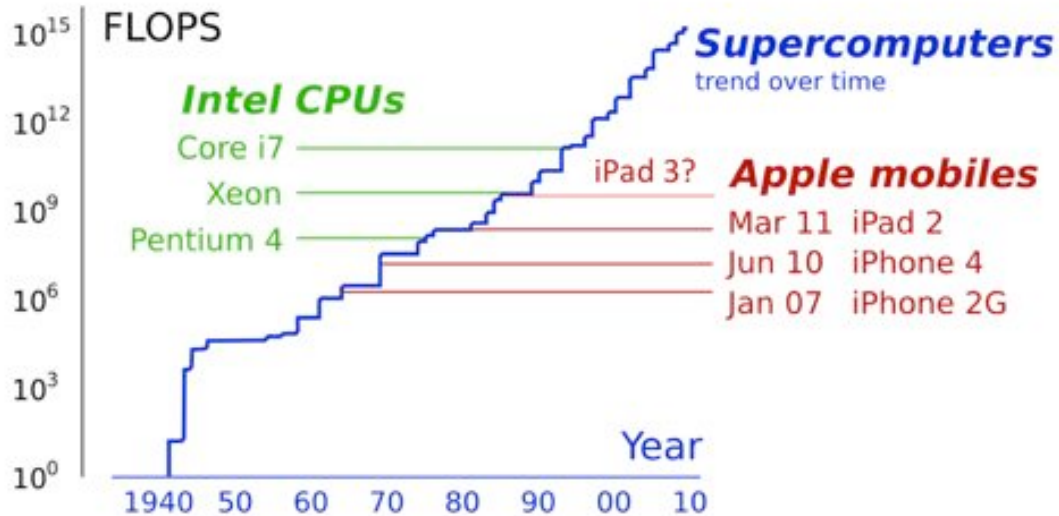
The cameras are pretty cool because their response spectra go well into the infrared. Try pointing one at a remote control or warm object. I have no idea what use this is, but it's awesome.

The really interesting parts are the compute power and the display.

The 1.4 GHz quad-core processor is a so-called 'system on a chip'. It's a CPU and graphics chip in one. Note that these things are often underclocked to save power. I don't think many mobile software take advantage of the multiple-cores — I'm sure they are under-used.

These CPUs are getting pretty powerful...

## Mobile supercomputing



Data from various Wikipedia articles on supercomputers, desktops, and mobile devices and <http://en.wikipedia.org/wiki/File:Supercomputing-rmax-graph.png>

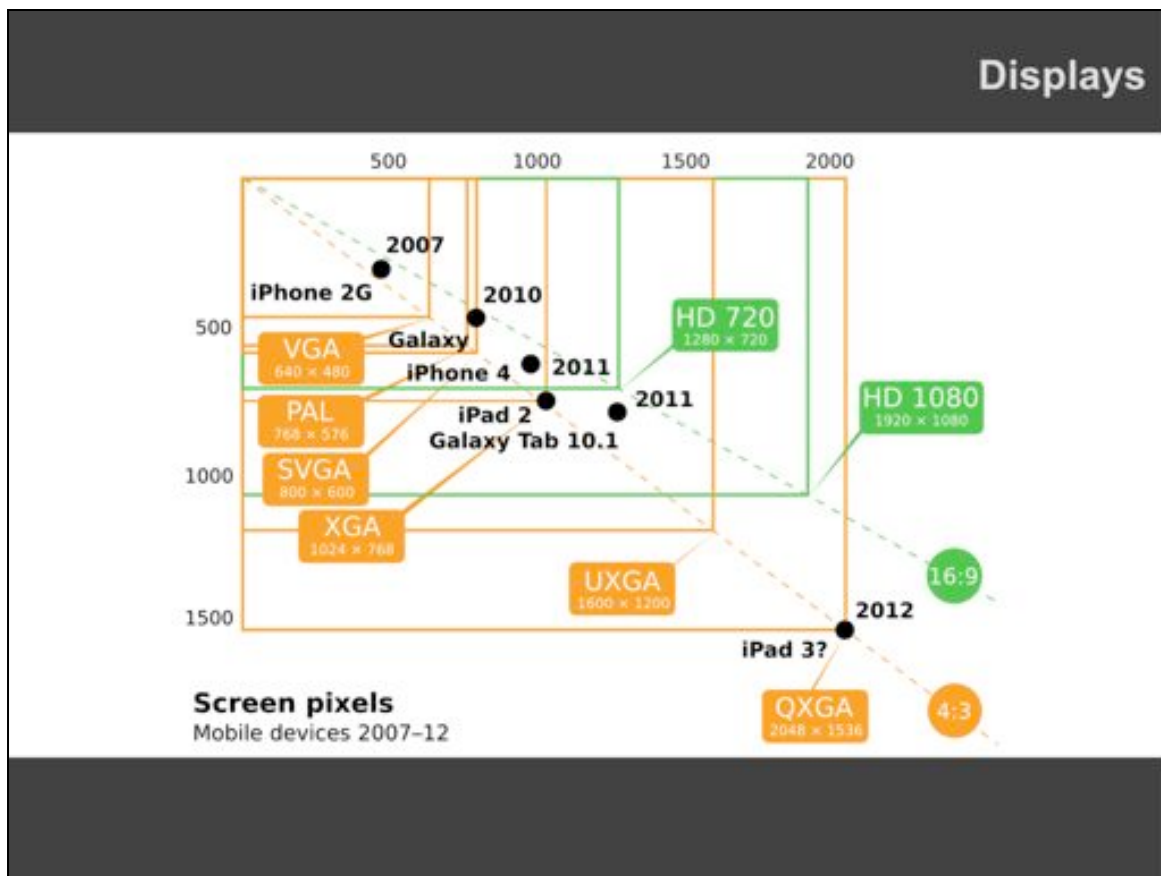
Now, these data are highly suspect and the whole idea of measuring compute performance in FLOPS is rejected by some people.

But if we can be at all consistent in how it's measured, which we can't, maybe it's vaguely interesting to look at a trend over time.

The chart shows supercomputer performance over time. It looks a bit like these mobile chips are progressing about four times more quickly than supercomputers did. I have projected the trend brazenly into the near future... I think we're at a GFLOPS already.

For reference, my iMac does about 40 GFLOPS or so. No doubt there will be limitations like power consumption, etc, but there is progress with batteries too...

Certainly serious work can be done on these devices.



The other big factor for serious work is the display, of course.

This is a noisy figure, but I've tried to show how these devices are mapping to standard display sizes. In about the last year, these screens have jumped from the very small end firmly into the middle range. The industry leader in terms of technology is the iPad, which has a very high pixel density and thus the sharpest image. I've yet to see a Galaxy Tab 10.1.

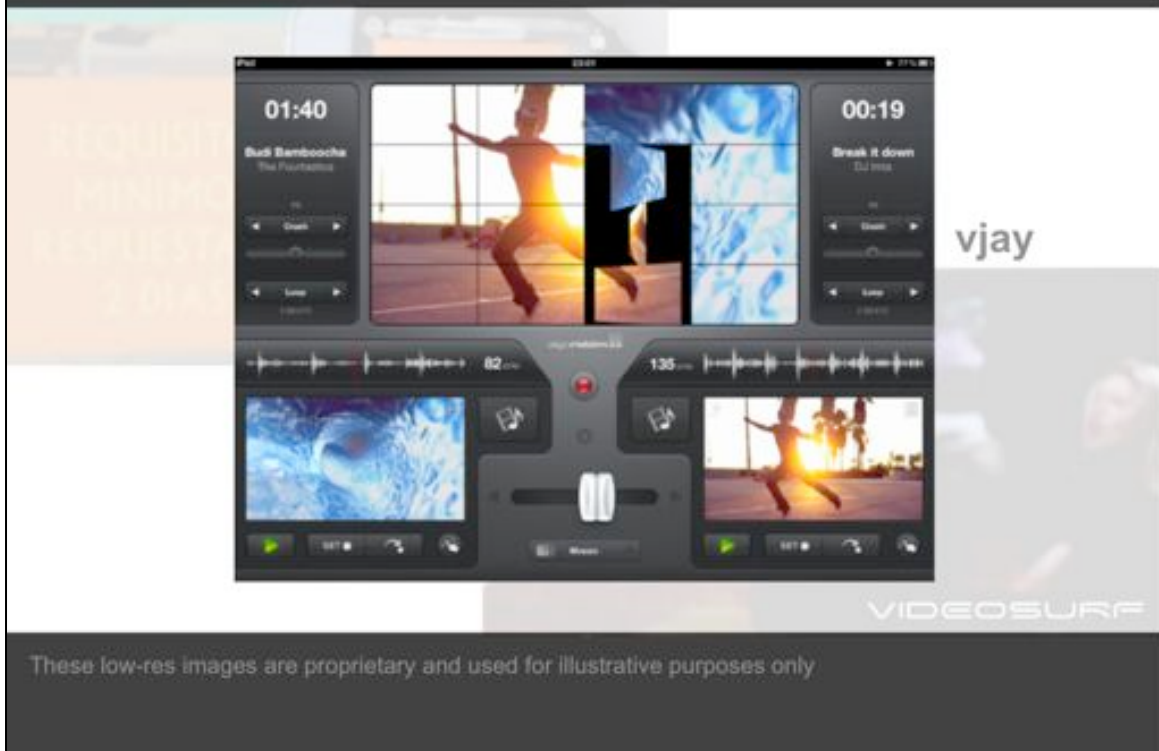
The iPad 3 — released since I made this figure — has double the pixels in each dimension, compared to its predecessor. So we have a hand-held desktop-sized display (in terms of pixels, not dimensions), running at a billion FLOPS.

Incredible.

But what can you actually do with this sort of technology?



## Highly creative apps

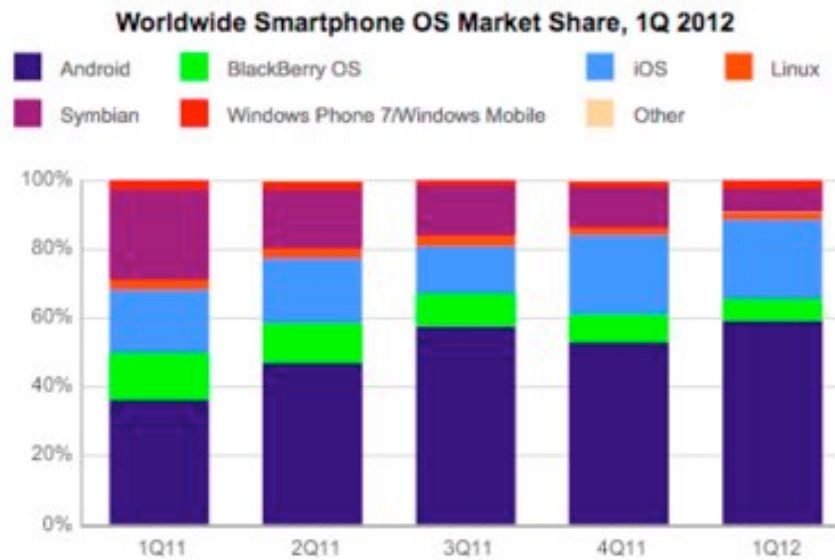


Most of the interaction with the device is through *apps*.

This app was recently released for iPad. It's worth watching some of the promotional material for it — it's a DJ mixing deck for video. It's perfect for a tablet — you want a dynamic, touchable display. It needs the power of these modern chips — processing and mixing audio and video in real time is taxing even for many laptops and desktops.

Imagine having software like this for seismic processing...

## Consumers like Android...



The two most important mobile operating systems are iOS, Apple's proprietary software for iPhone and iPad, and Android, Google's open source software for phones and tablets. All other platforms are dying out as manufacturers flock to Android.

As a result of which, people are buying Android devices in increasing numbers — today it's running on more than half of new devices.

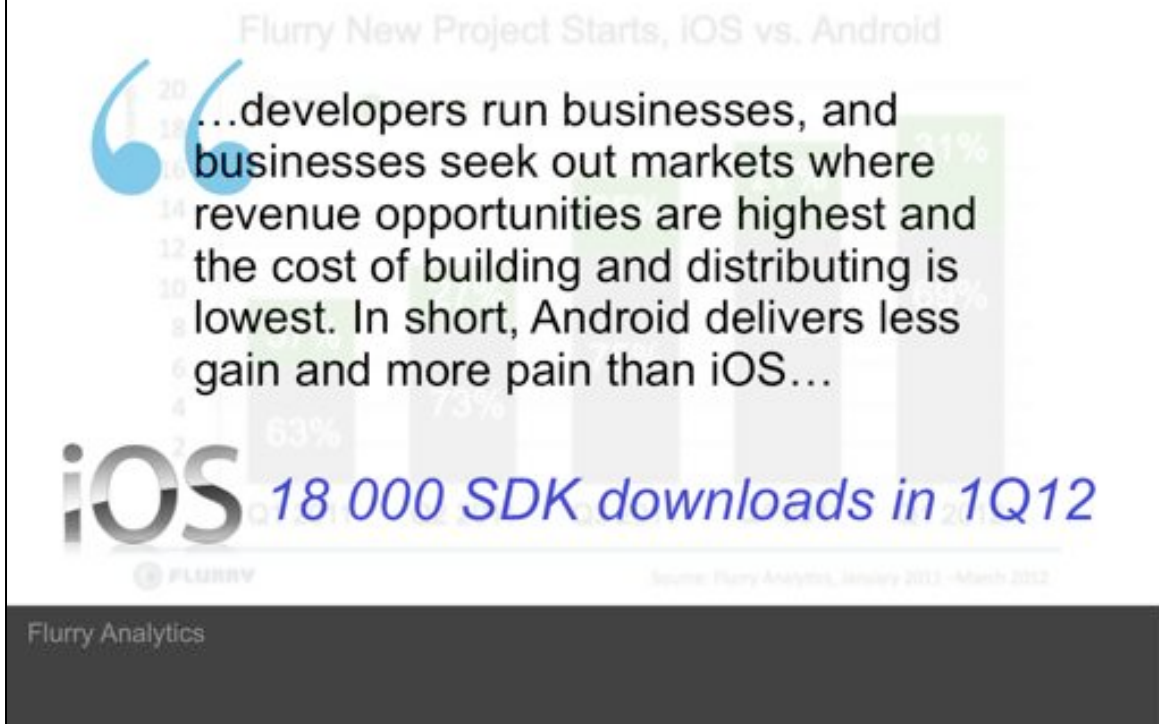
## ...but developers like iOS



But developers love iOS, mainly because of the lucrative App Store, which is much better developed and more profitable than Google's Play Store, or Amazon's app market. iPhone and iPad owners, it seems, don't mind paying for iOS apps. In contrast, developers for Android often grumble that people won't pay for anything.

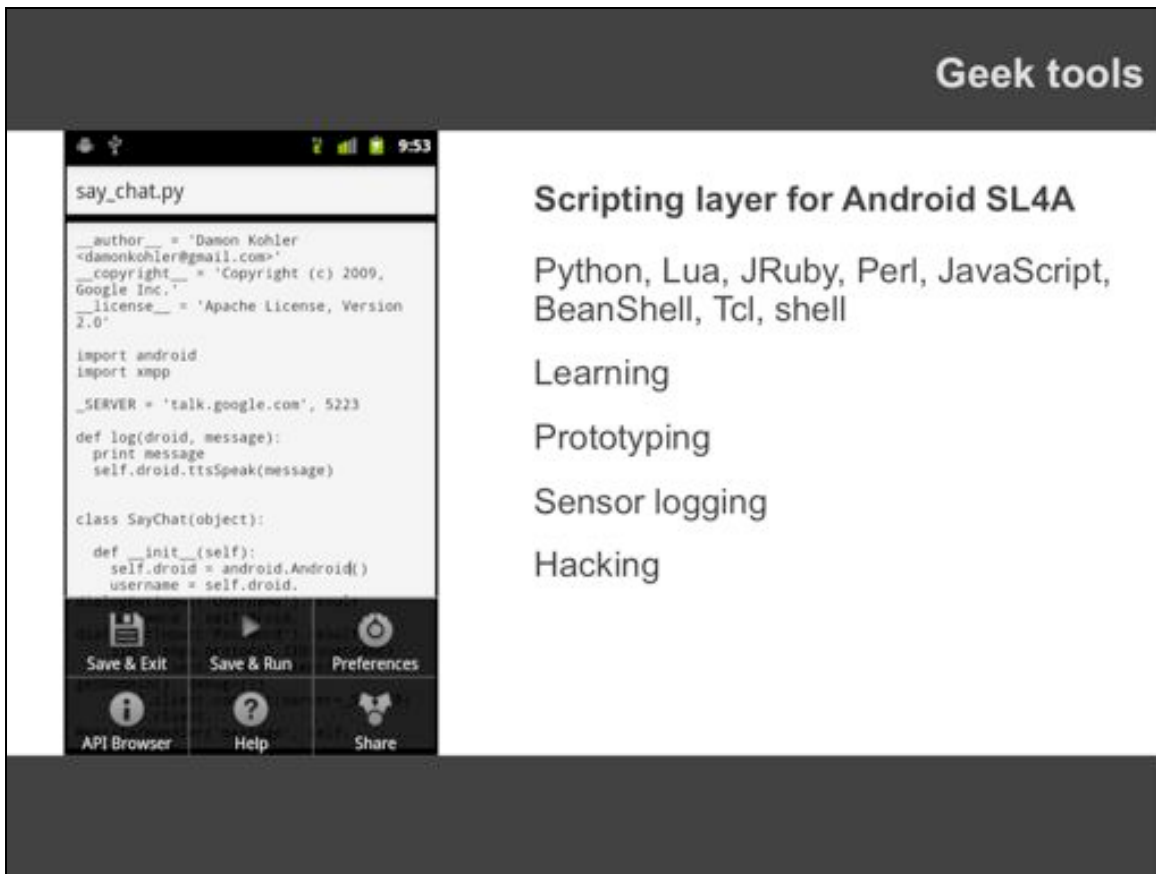
Indeed, this was my experience too >> <http://www.agilegeoscience.com/journal/2011/9/12/avo-is-free.html>

...but developers like iOS



The same report that provided those figures really rams it home: Android delivers less gain and more pain than iOS.

Look at this — 18 000 iOS SDK downloads in the first quarter of the year. Many of those are new developers. I don't have the figures for the Android SDK — maybe it's about the same. I wonder how many are programming scientific applications, and how many are just trying to make the next Angry Birds or Instagram...



This is one for this audience: Android Scripting Layer. Build and run scripts right on your phone. You can interface any of the phone's functions. No need to build apps!

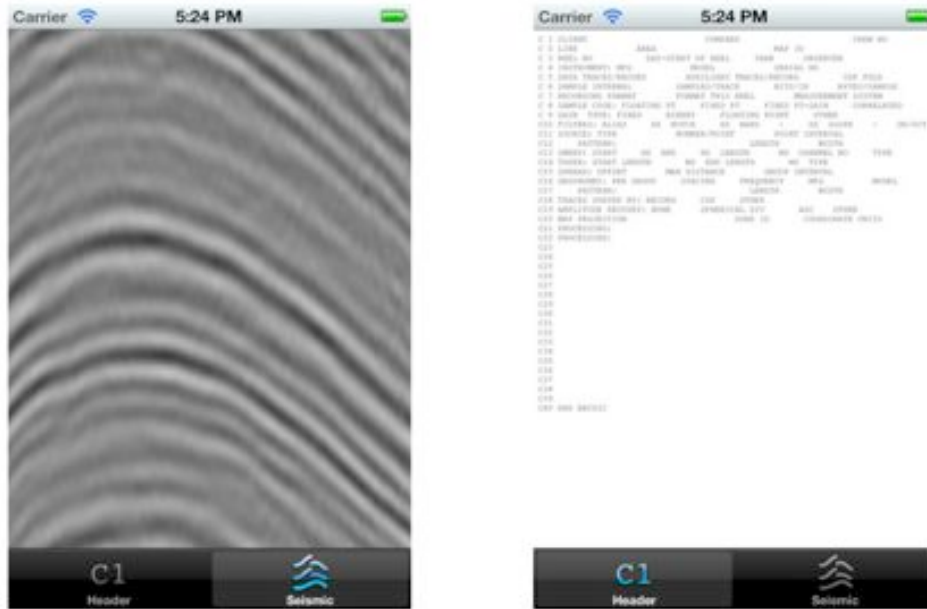
The tool already supports Python, JavaScript, Jruby, Perl, and shell... and other languages too.

This sort of thing reflects the beauty of having an open platform.

This is the sort of thing you'd need if you were going to, say, send your phone into space and monitor its systems to see how the GPS, comms, and other sensors were coping with the altitude and location.



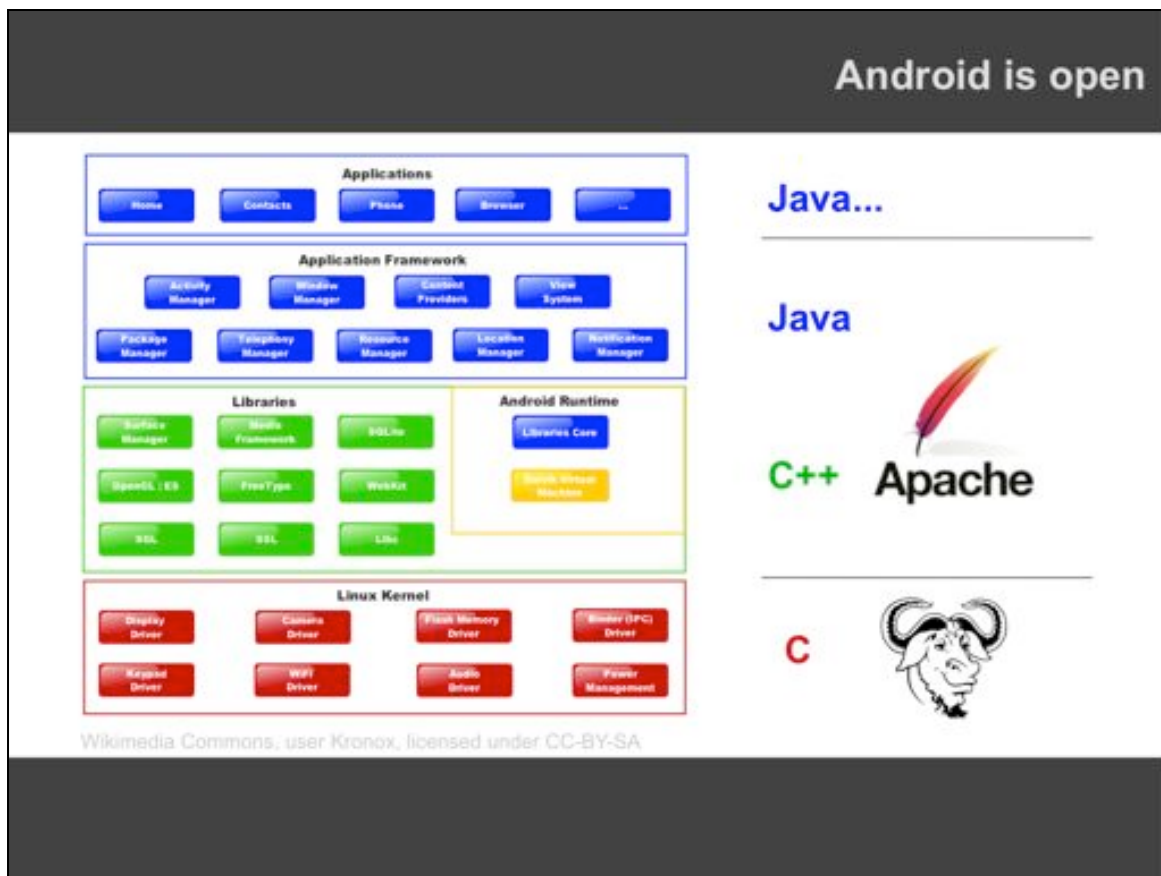
## Seismic data too



Pocket Seis by Unplugged Development, \$20

These devices aren't just used for fun, obviously. There are sciencey, geeky tools out there.

This is the first app, that I know of, that can display SEGY files on an iPhone. The developer is trying to sell it for \$20 — I doubt this will work out, because this is the only thing it does, but perhaps it's the first seismic app. Pocket Seis — I'm sure you can expense it.



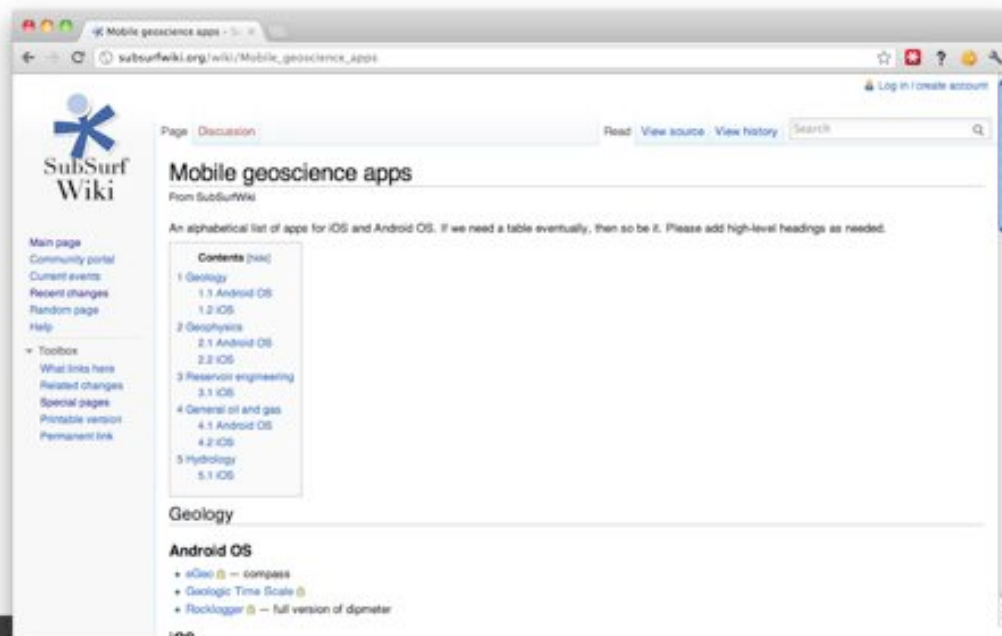
So Android is open... what does that mean exactly?

Well, most of the versions of the entire operating system are completely open source and licensed under the Apache license, a permissive license used for about 25% of the projects on Google Code.

And Android is built on top of the Linux kernel, licensed by the GPL of course. Software is usually written in Java, and runs on the Dalvik Virtual Machine, analogous to the Java Virtual Machine on PCs. (It's a register-based architecture rather than stack-based and no, I don't know what that means).

App Inventor apps, just like other apps, live up here in this top layer, with whatever license you like. While they are not written in Java exactly, they are converted to Java from the visual programming environment you build them in.

## List of mobile geoscience apps



[subsurfwiki.org/wiki/Mobile\\_geoscience\\_apps](http://subsurfwiki.org/wiki/Mobile_geoscience_apps)

For what it's worth, I have started a list of geoscience mobile apps in my wiki, AgileWiki... You are free to add to it.

[http://agileinterpretation.com/wiki/Mobile\\_geoscience\\_apps](http://agileinterpretation.com/wiki/Mobile_geoscience_apps)

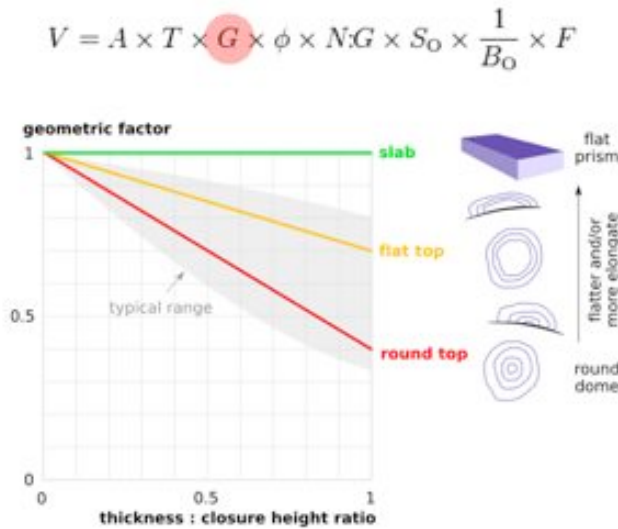
Volume+ app interface showing input fields and a calculated result:

- Area or Volume: 25
- Thickness or blank: 50
- Relief: Geometry
- Porosity  $\phi$ : 0.2
- Net : Gross N:G: 55
- Oil saturation  $S_o$ : 0.75
- Volume factor  $B_o$ : 1.2
- Fudge factor  $F$ : arbitrary

Calculated result: **541** million bbl

Additional values: 99.63, 53.9 billion USD, 85.9 million m<sup>3</sup>

Buttons: Imperial, Store, Fetch, Email, Help

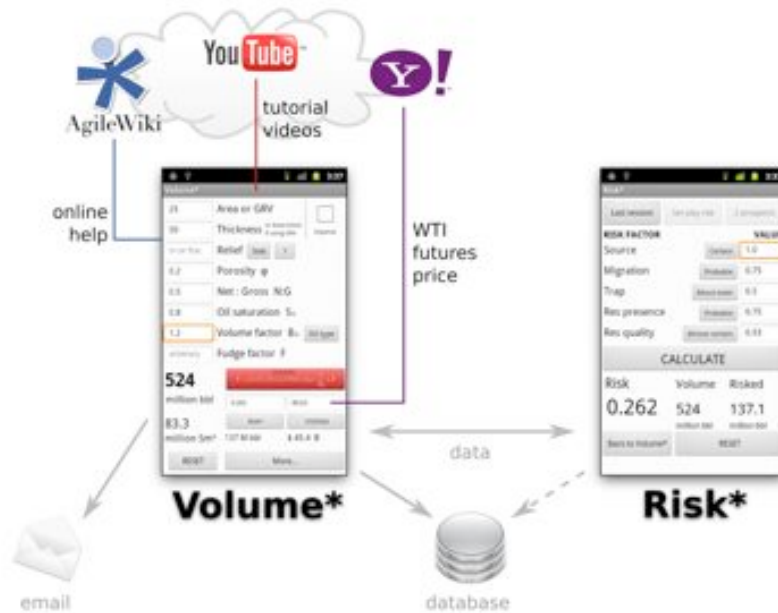


Here's our little volumetrics app, for Android. What's the point in such a simple app, basically just a calculator? You could almost do this math in your head (almost).

We like to think we've added a bit of value for people. For example, you can look up an appropriate value for the geometric factor — most people probably don't have this in their heads, and it's not a very intuitive number. You could ignore it, but your volumes will, in many cases, be too big. Another tricky parameter is  $B_o$  — do you know what to use for a heavy oil? Do you remember how to compute  $B_g$  for a gas reservoir? We can build in simple tables and tools to make this easy for everyone. The point is to bring others' experience and knowledge into the device, so you have it right there with you.

And notice here... it's very small, but this number is the oil price. You can type something in, but the app automatically looks up the current price for you — it even knows about different benchmarks, like Brent crude, or Canadian heavy oil...

## Open source petroleum geoscience



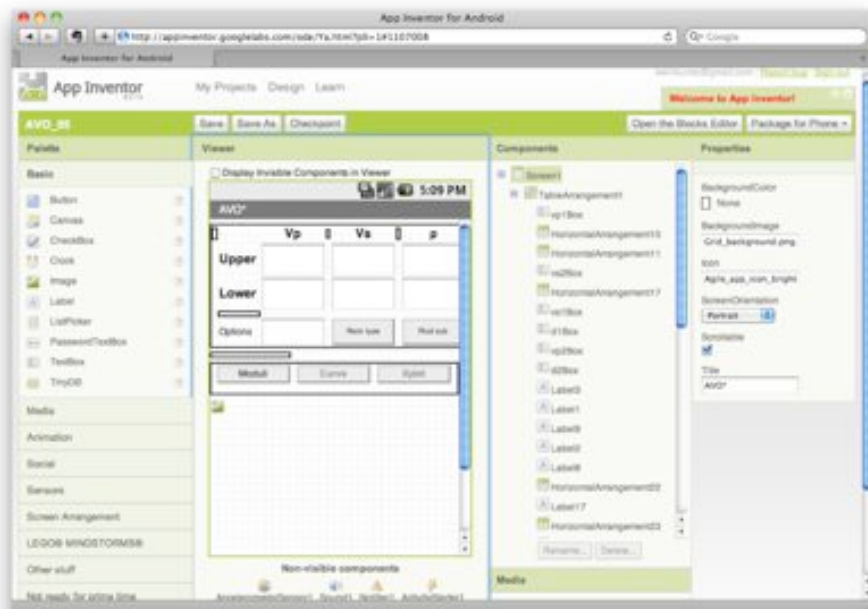
[github.com/agile-geoscience](https://github.com/agile-geoscience)

This ability to bring in data from the cloud — exploiting the fact that the device is probably online — is a powerful idea. As I said, we look up the futures price for oil (about one month hence), but we also have tutorial videos in YouTube, and documentation in SubSurfWiki. We could store data in the cloud too — but some users are more comfortable with data being kept on the device, so for now at least we keep it there. We can also email results to others, ... and the apps can talk to each other, passing data back and forth. We have a sister app, Risk, that computes probability of discovery — and passes its result back to Volume for a risked resource number.

I'll come back to the power of the cloud again in a minute.



# MIT App Inventor for Android



[code.google.com/p/app-inventor-releases](http://code.google.com/p/app-inventor-releases)  
released under Apache license

We developed all our apps on MIT App Inventor (previously Google's project, now open source). If you have a Google account, you can already just navigate to [appinventor.googlelabs.com](http://appinventor.googlelabs.com) and start using it. It runs in the browser and stores all your projects in the cloud.

There are three components:

- + The screen viewer — shown here — where you build the app and specify how it will look and the default parameters for the components
- + The blocks editor — which I'll show in a sec — where you build the logic for the program, specifying what it will *do*
- + The device itself, which you tether all the way through development, so that you can run the app on there. All changes are instantly live, so testing is very easy, with no compiling or running.

You can run an emulator instead of running on the device. So you can actually build these things without having an Android device of your own.



Here's how one of my apps looks on my phone (left) and in the emulator. It's just a simple calculator for amplitude-vs-offset modeling. The nifty thing here is the graph.

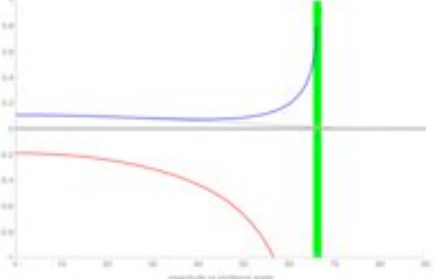
I was initially stumped by drawing a graph, and didn't really know where to turn. Did I need to draw one with the raster tools? Maybe use a package like agraphengine, but that's only for 'real' Android coding — you can't have arbitrary code blocks in App Inventor yet.

Google to the rescue with their Charts web API. I can simply add an image to my app, then instead of calling on an in-app file (a PNG or JPG, say), I can provide a URL.

It's not pretty...

## The programmable web

```
http://chart.googleapis.com/chart
?cht=lc
&chd=t:-1|-1|-1
&chds=-1,1
&chs=600x400
&chf=bg,s,FFFFFF
&chco=BBBBBB,FF0000,0000FF
&chxr=0,0,90,10|1,0,100|2,-1,1
&chxt=x,x,y
&chxl=1:magnitude%20vs%20incidence%20angle|
&chxp=1,50
&chxs=0N*f0*
&chm=R,00FF00,0,0.72545,0.74545|r,AAAAAA,0,0.495,0.505
&chfd=0,x,0,90,0.1,0.10827-0.11312*(sin(x*0.0174533))%5E2|1,x,
0,90,0.1,0.01786-0.02044*(sin(x*0.0174533))%5E2-0.2075/(cos
(0.5*x*0.0174533%2B0.5*asin(sin(x*0.0174533)*1.09302)))
%5E2-0.15206*(sin(x*0.0174533))%5E2|2,x,0,90,0.1,0.06383-0.07305*
(sin(x*0.0174533))%5E2%2B0.04444/(cos(0.5*x*0.0174533%2B0.5*asin
(sin(x*0.0174533)*1.09302)))%5E2-0.09951*(sin(x*0.0174533))%5E2
```



...but a URL like this one returns a graph, as here.

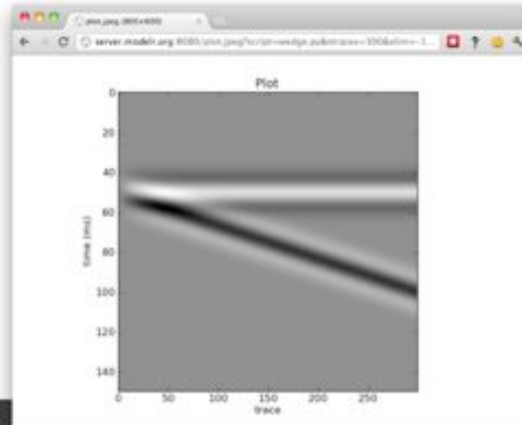
Everything after **chfd** is just the function definition: one for the two-term Shuey equation, and two for the Aki-Richards equation (one of them is in red text). So my charting problem is reduced to simply constructing this URL as a text object, which is pretty easy.

Of course, it means the user has to be online, but that's increasingly common, at least in cities and offices.

This maybe be trivial to you, but for me, this was an epiphany. Not only is there an ecosystem of code snippets and code APIs, but now this universe of web APIs, communicating through simple URLs, is mid-bogglingly inspiring to me. If you've heard of the mashable web, or the programmable web, this is what that is.

One of my new projects is to build a web API of my own to serve up synthetic gathers...

```
http://server.modelr.org:8080/plot.jpeg  
?script=wedge.py  
&ntraces=300  
&xlim=-1,1  
&title=Plot  
&f=25  
&Rpp0=2355,2200,1100  
&Rpp1=2400,2500,1200  
&max_thickness=50  
&theta=0  
&pad=50
```

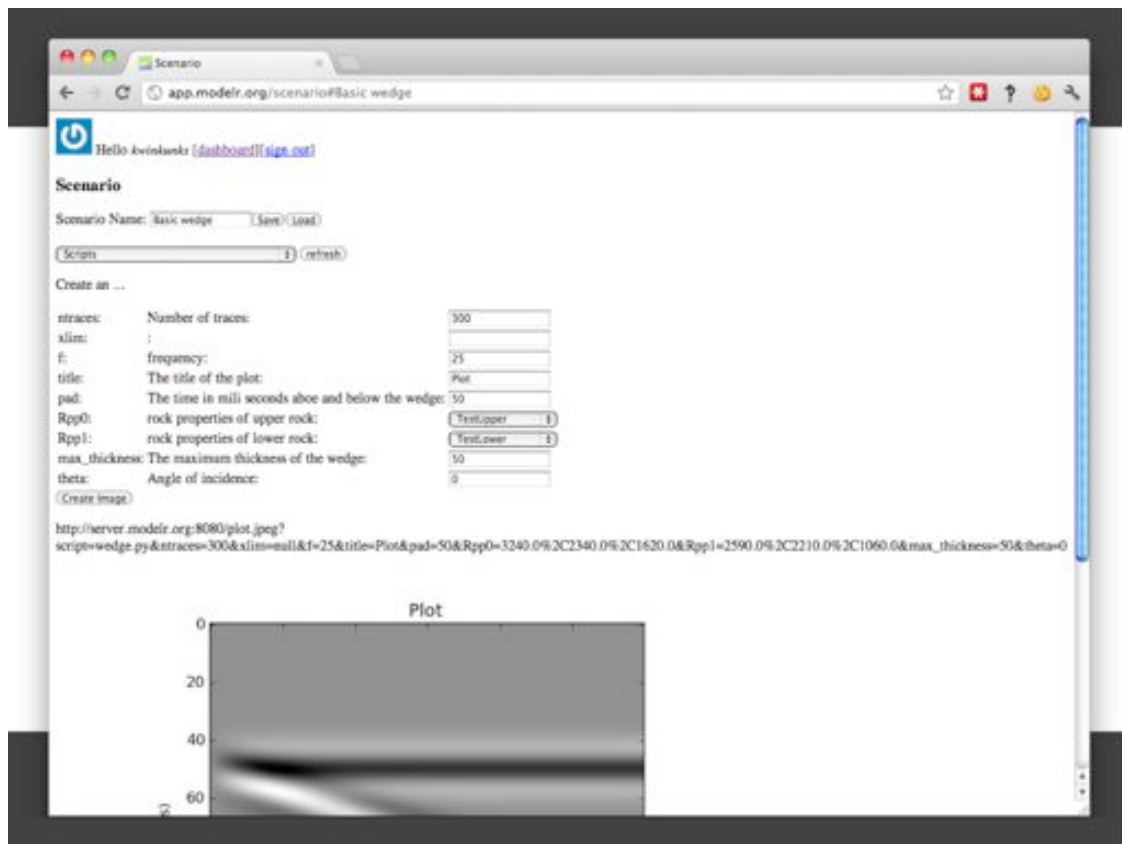


Here's the prototype, built in less than a week, with a lot of help from Sean Ross-Ross, of Enthought.

As you see, it eats URLs, just like Google's Charts API. You provide a wavelet type, scale, etc, and I can pass back an image of a gather... or In this case it's a wedge model.

So now maybe I can really do some quite hard, sciencey, things on the phone, because the compute and rendering is happening somewhere else.

server.modelr.org is a Python program, with its own little web server, sitting on an Amazon EC2 instance in West Virginia. It's very cheap, decently robust, and it works!

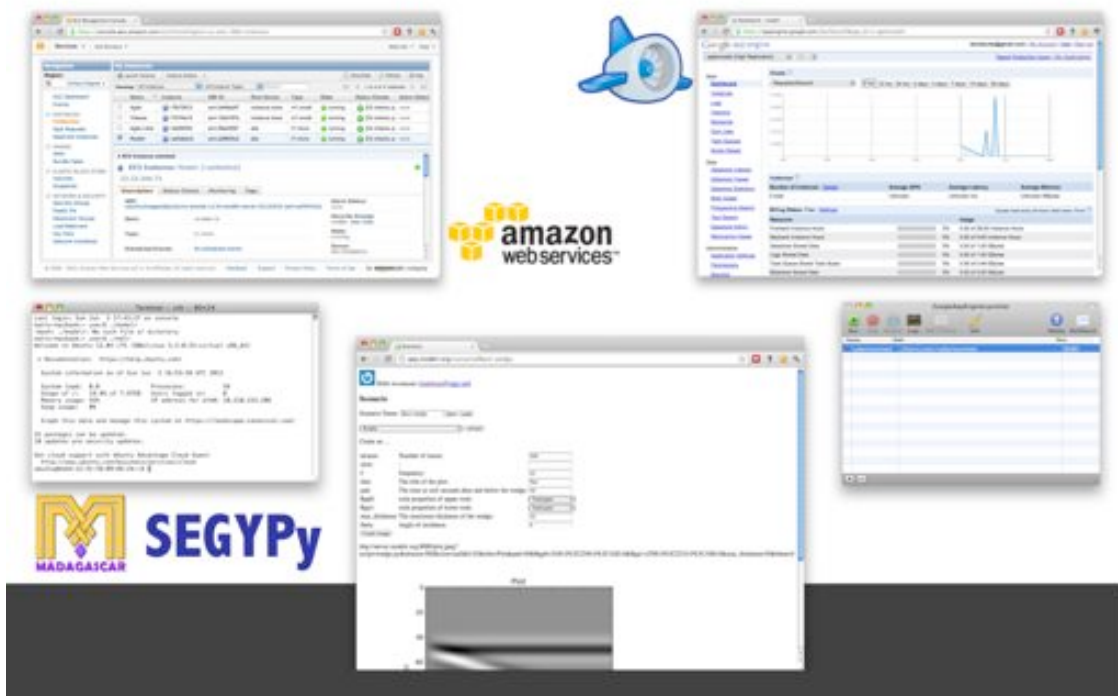


We did build a simple interface to the API too — so you don't have to go in via the API. The interface is running on Google App Engine, not Amazon EC2. App Engine is great because it has built-in support for user logins, it has a simple database, and it's fully scalable (not really an issue for me!). There's a bit of Javascript for these menus. It's very simple. Next step is to make it prettier, probably with Twitter's amazing Bootstrap framework.

This tool uses the web API, as you might be able to see here at the front, to fetch the plot. It's not using a back door or anything — it's just another way to get at the functionality.



## Behind the scenes

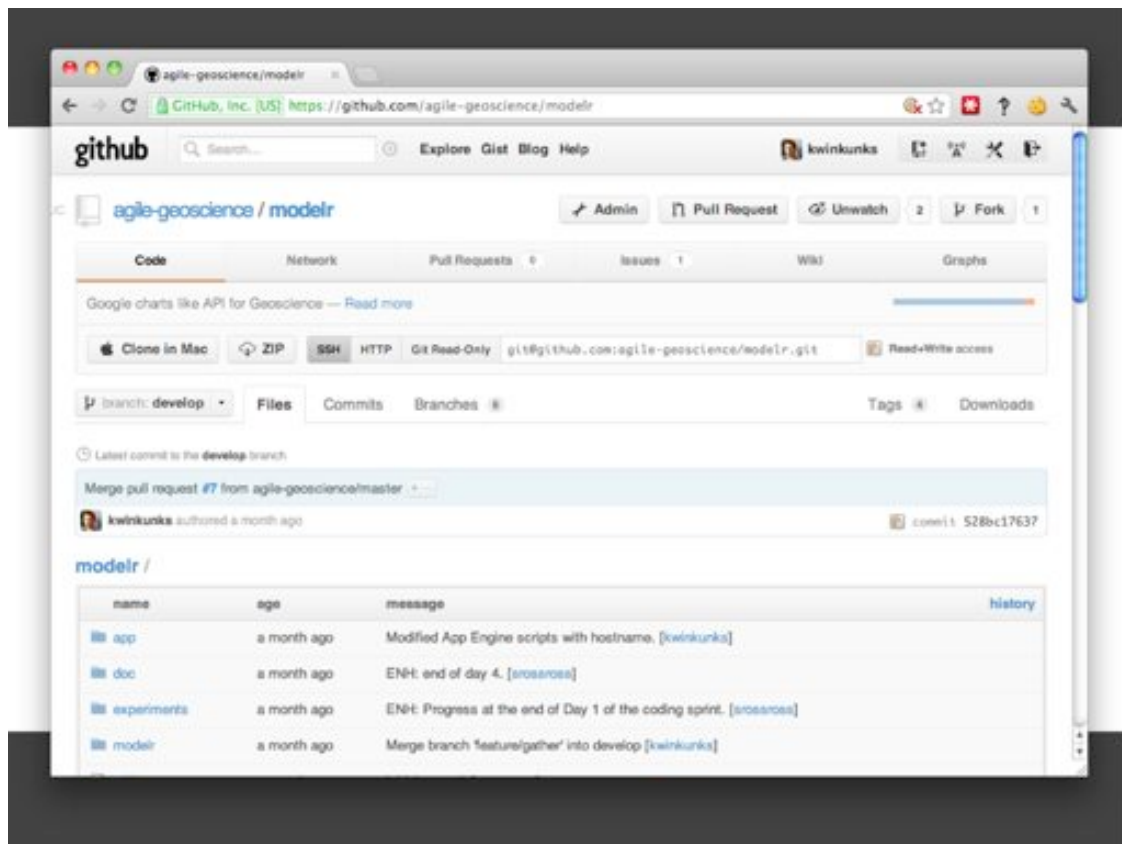


So here's the system, in case it didn't make sense. In the middle is the user's browser. Out here on the right is Google App Engine, providing the interface, login, etc. You access your App Engine instance via this little launcher tool.

On the left is Amazon Web Services, specifically an Elastic Compute 2 instance, which I access via SSL and command line. It's just a Linux (Ubuntu) machine, with Enthought's EPD Free running on it. It's very robust — these things rarely need rebooting.

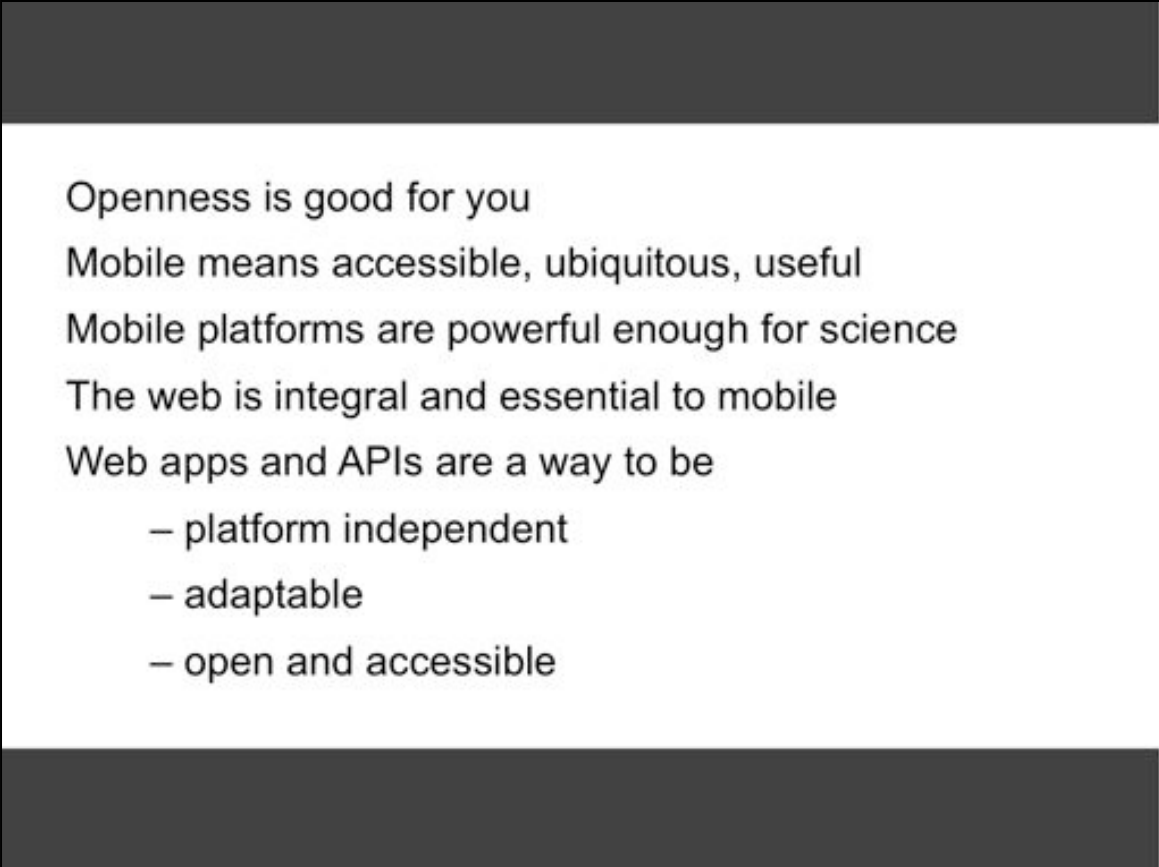
One day I'd love to add Madagascar and SEGPy, and other nice tools, to this Amazon EC2 stack. Then we can do finite difference modeling, for example, and not have to settle for convolution, as we do now.

By the way, there's a public image (Amazon Machine Image) with the whole stack on it — you're welcome to install it if you want to try this out.



All the code is on GitHub, if you're interested in poking around. Agile Geoscience is the name,... or you can look for my personal account, Kwinkunks.

You'll also find the code for some of our mobile apps there. It's not really code, though, because these are just zipped MIT App Inventor projects.



Openness is good for you  
Mobile means accessible, ubiquitous, useful  
Mobile platforms are powerful enough for science  
The web is integral and essential to mobile  
Web apps and APIs are a way to be

- platform independent
- adaptable
- open and accessible

So where am I going with all this... I don't really know. I started this exploration a little over a year ago, as a sort of professional hobbyist. I am very excited about this technology, especially Android and App Inventor, as a democratization of experiment-driven innovation.

Here's what I know — openness is good for you. It's good for us, as a community. And it's good for science. I know you all know this.

Mobile software is accessible, it's everywhere, and it is — or can be — useful. Even for science... these devices are now definitely powerful enough for all sorts of scientific applications.

I think it's important to exploit the fact that these devices are on the web, and I believe web APIs are one way we, as a community, can build a rich palette of open, mashable tools that could support rich apps on any platform.

My real message is this: don't underestimate the power of toy-like tools to energize and inspire people. The next time you're thinking about an interface for your awesome science, think about using a visual programming interface, so that 'ordinary geophysicists' can play with your algorithms.