# The Current Phone Problem.

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>Mobile Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>Google, Yahoo, AOL, Windows Live, YouTube</td>
<td></td>
</tr>
<tr>
<td>Apps</td>
<td>Web browser. All kinds of vertical niche applications.</td>
<td>No Open Solution</td>
</tr>
<tr>
<td>GUI</td>
<td>Common “desktop” paradigm</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>Generic: usually keyboard, mouse, and monitor</td>
<td>Specialized: keypad, buttons, and inconsistent (and often limited) screen space</td>
</tr>
<tr>
<td>HW</td>
<td>x86 (Intel, AMD, VIA)</td>
<td>Lots of different platforms</td>
</tr>
</tbody>
</table>
But if we view this merely as an engineering problem to be solved...

• Then we WILL create a mobile phone that mimics a PC. We can do better than a 1960s vision.

• We would solve the problem, but we will fail to create new forms of computing.

• That’s winning a battle, but losing the war.

• So how do we create a new form of computing?

• Who is in charge? Or better yet, WHAT is in charge?
How to be God.

• The KEY to making complex, neoforms appear from simple systems is:
  • Access to the Building Blocks. The Amino acids of the systems. The Atoms of Molecules.
  • Freedom to WRITE new rules of combination.
  • Let’s take an example of starting small...
Big fleas have little fleas on their backs to bite them, and little fleas have lesser fleas, and so ad infinitum.

What happens when you start small and iterate?
It all starts small...

• Sometimes it’s the smallest, most simple concepts that work best.

• So take something small and begin...
Then iterate.

• The first iteration interpreted graphically looks like this:
And do it again.

- The next iteration interpreted graphically might look something like this:
Ok. Now what?

We have some basic building blocks in place...
Let’s Define Some Rules...

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Move forward by line length drawing a line</td>
</tr>
<tr>
<td>f</td>
<td>Move forward by line length without drawing a line</td>
</tr>
<tr>
<td>+</td>
<td>Turn left by turning angle</td>
</tr>
<tr>
<td>-</td>
<td>Turn right by turning angle</td>
</tr>
<tr>
<td>!</td>
<td>Reverse direction (i.e., turn by 180 degrees)</td>
</tr>
<tr>
<td>[</td>
<td>Push current drawing state onto stack</td>
</tr>
<tr>
<td>]</td>
<td>Pop current drawing state from the stack</td>
</tr>
<tr>
<td>*</td>
<td>Increment the line width by line width increment</td>
</tr>
<tr>
<td>-</td>
<td>Decrement the line width by line width increment</td>
</tr>
<tr>
<td>@</td>
<td>Draw a dot with line width radius</td>
</tr>
<tr>
<td>{</td>
<td>Open a polygon</td>
</tr>
<tr>
<td>}</td>
<td>Close a polygon and fill it with fill colour</td>
</tr>
<tr>
<td>&gt;</td>
<td>Multiply the line length by the line length scale factor</td>
</tr>
<tr>
<td>&lt;</td>
<td>Divide the line length by the line length scale factor</td>
</tr>
<tr>
<td>&amp;</td>
<td>Swap the meaning of + and -</td>
</tr>
<tr>
<td>(</td>
<td>Decrement turning angle by turning angle increment</td>
</tr>
<tr>
<td>)</td>
<td>Increment turning angle by turning angle increment</td>
</tr>
</tbody>
</table>
And You Can Do This:

Axiom: F+F+F+F
F --> FF+F-F+F+FF
\( \theta = 90 \)
Even this!

Axiom  $X$
$F \rightarrow FF$
$X \rightarrow F$
$(|X|+X)+F(|F|-X)$
$\varnothing = 22.5$
Does This Look Familiar?

<table>
<thead>
<tr>
<th>Turning angle = 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axiom (depth=0) =</td>
</tr>
<tr>
<td>F1F1F1</td>
</tr>
<tr>
<td>0 &lt; 0 &gt; 0 --&gt; 0</td>
</tr>
<tr>
<td>0 &lt; 0 &gt; 1 --&gt; 1[-F1F1]</td>
</tr>
<tr>
<td>0 &lt; 1 &gt; 0 --&gt; 1</td>
</tr>
<tr>
<td>0 &lt; 1 &gt; 1 --&gt; 1</td>
</tr>
<tr>
<td>1 &lt; 0 &gt; 0 --&gt; 0</td>
</tr>
<tr>
<td>1 &lt; 0 &gt; 1 --&gt; 1F1</td>
</tr>
<tr>
<td>1 &lt; 1 &gt; 0 --&gt; 1</td>
</tr>
<tr>
<td>1 &lt; 1 &gt; 1 --&gt; F0</td>
</tr>
<tr>
<td>* &lt; + &gt; * --&gt; -</td>
</tr>
<tr>
<td>* &lt; - &gt; * --&gt; +</td>
</tr>
</tbody>
</table>
Now We’re Done.

If we just recreate the PC on the Phone, it will be just another Flea. We need new species...
Standing on each other’s shoulders.

- Mere Access to atoms and rules is Necessary but not sufficient to creating new life forms.
- Lots of curves can fill this space, but only some will prove fruitful.
- The Combinatorial explosion within the design space requires a freedom for many to experiment.
- We need Collective wisdom and imagination.
OpenMoko 2007 Software Stack.

- **Mobile Handset Hardware (FIC Neo1973)**
- **Linux 2.6 Kernel & Device Drivers**
  - udev
  - blueZ
  - dbus
  - GSM
  - GPS
- **Linux Core Services**
  - core
  - net
  - UI
  - PIM
- **Linux User Interface**
  - matchbox
  - GTK+2
  - kdrive
  - libX11
- **OpenMoko Application Framework**
  - Dialer
  - (Finger Applications)
  - Main Menu
  - Search
  - Application Manager
  - Messages
  - Contacts
  - (Stylus Applications)
  - Clocks
  - Media Player
  - (others)
  - (others)
- **3rd Party Applications**
  - Web Browser
  - IM
  - Book Reader
  - Terminal
  - (others)
  - (others)

**Open Embedded x86 SDK (Target Board)**

**GPL**

**LGPL**

**GPL**
1) Atomic Access.

These are the building blocks of our system.
OpenMoko’s Application Framework.

- **libmokocore** – IPC, Device Control, Application State.
- **libmokoui** – Common look & feel.
- **libmokonet** – high-level connection queries.
- **libmokopim** – high-level PIM APIs.

(OpenMoko Application Framework)
libmokocore: At a Glance.

- OpenMoko IPC API
  - `run_contacts_application ("new_phone_number", "555-273-172");`

- Device Control API
  - `device_set_display_brightness(device, 100);`
  - `s = device_get_signal_strength(device, MC_PERIPHERAL_GSM);`
  - Uses `dbus(-glib), libgconf, libgconf-bridge`
libmokou: At a Glance.

- Full base GTK+ widgets
- Additional phone widget classes on top of GTK+
libmokonet: At a Glance.

- peers = get_file_sinks( BT | INTERNET );
- at_home = gps_within_region( "at_home" );
- gsmconn = gsm_connection_new ("555-728-1829");
libmokopim: At a Glance.

- Will probably never be written…
- Just use libebook, libecal, libcamel, and friends…
2) Freedom to Write Rules.

The ability to create your own combinations.
The OpenMoko User Interface.

- openmoko-panel
- openmoko-<application>
- openmoko-footer
openmoko-panel: At a Glance.

- Always visible and global for all applications.
- We just use matchbox-panel-2, lightweight gtk+-based panel
- Panel applet plugin host
  - Panel plugins are shared libraries
  - ${libdir}/matchbox-panel/*.so
- Read on startup of mb-panel-2
openmoko-<application>: At a Glance.

- Stylus applications
- Finger applications
- X11 legacy applications
openmoko-footer: At a Glance.

- Task Manager
- Status Bar
  - Temporary Notification area
- Application Toggling
3) Labs to Experiment.

Lots of people trying new stuff.
OpenMoko Application Development.

- Writing a Stylus Application
- Writing a Finger Application
- Using Other Widgets
Stylus Applications: Overview.

- **MokoPanedWindow** – base class for stylus windows
- **MokoMenuBox** – application menu, filter menu
- **<Navigation Widget>** – e.g. GtkTreeView
- **MokoToolBox** – search, action buttons
- **<Details Widget>** – e.g.GtkLabel
Finger Applications: Overview.

- **MokoFingerWindow** – Base class for finger windows
- **MokoFingerButton** – Large, finger-friendly button
- **MokoFingerWheel** – Scrolling, mode changing (icon indicates mode)
- **MokoFingerToolBox** – Three (or more) tools per page, multiple pages possible
Other Widgets.

- *MokoDialogWindow* – Full-screen modal dialog, can use any Gtk+ widget
- Field Widgets
- View mode
- Edit mode
- More...
4) Feedback.

Collective experimentation leads to new life forms.
2007 Finger Applications.

<table>
<thead>
<tr>
<th>PHASE 1</th>
<th>PHASE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dialer</strong></td>
<td><strong>Clocks</strong></td>
</tr>
<tr>
<td><strong>Main Menu</strong></td>
<td><strong>Screen Saver</strong></td>
</tr>
<tr>
<td><strong>Music Player</strong></td>
<td><strong>Calculator</strong></td>
</tr>
<tr>
<td><strong>History</strong></td>
<td><strong>Unit Converter</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Game</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Guitar Tuning</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Code Memo</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Your Applications...</strong></td>
</tr>
</tbody>
</table>
## 2007 Stylus Applications.

<table>
<thead>
<tr>
<th>PHASE 1</th>
<th>PHASE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contacts</td>
<td>Feed Reader</td>
</tr>
<tr>
<td>Dates</td>
<td>Messages</td>
</tr>
<tr>
<td>Application Manager</td>
<td>Preferences</td>
</tr>
<tr>
<td>Today</td>
<td>Media Player</td>
</tr>
<tr>
<td></td>
<td>Sketchbook</td>
</tr>
<tr>
<td></td>
<td>Terminal</td>
</tr>
<tr>
<td></td>
<td>IM</td>
</tr>
<tr>
<td></td>
<td>Web Browser Reader</td>
</tr>
<tr>
<td></td>
<td>System Info</td>
</tr>
</tbody>
</table>

*Your Applications...*
Community Resources.

{openmoko.org}
In 1973, Marty Cooper invented the mobile phone. This gave birth to an industry. We’re going to revolutionize it again. Only this time, you will write the rules.

Welcome to the New 1973. The future is open.
The Neo1973: Write Your Own Rules.
Your Mobile Lab for Experimentation.
Create New Building Blocks.
Cost Breakdown.

<table>
<thead>
<tr>
<th>Description</th>
<th>Retail</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Kit</strong></td>
<td>Neo1973 Battery Headset Compact Charger Carrying Case Stylus Lanyard MicroSD Card Micro USB Connectivity Cable Instruction Manual and Warranty</td>
</tr>
<tr>
<td><strong>Car Kit</strong></td>
<td>Windshield Mount and Device Holder Car Charger External Antenna</td>
</tr>
<tr>
<td><strong>Hacker’s Lunchbox</strong></td>
<td>Development Board Battery Compact Charger for Development Board FPC Shoulder Strap USB A-B</td>
</tr>
</tbody>
</table>
Our 2007 Roadmap.

- **Neo1973 Open R&D (Feb. 12)**
  - openmoko.org Opened (wiki, bugzilla, source, ...)

- **Neo1973 Phase 0 (Early Mar.)**
  - First Phones are Freed

- **Neo1973 Phase 1 (Late Mar.)**
  - Developer Sales Begin

- **Neo1973 Phase 1+ (Jun.)**
  - Hardware Refresh

- **Neo1973 Phase 2 (Sept.)**
  - Mass Market Stage
“Never send a human to do a machine’s job.”

Agent Smith, 1999.
Why in God’s name don’t we use phones and humans to do this...

• Schedule a call on your calendar
• Get your approval, check your time zone.
• Request to dial you at the appointed time...

“Neo... Call Mickey when he and I are both available.”
The PC is maladaptive. The Phone is maladaptive. Don’t follow the phone. Leapfrog it. The key is to achieve what the PC and the phone intended.

Computing everywhere. Intuitive computing. Computing that is as natural to us as finger painting.
The 21st Century’s Opportunity.

OpenMoko
Freedom to write new rules of combination.

Ubiquitous Computing
Our devices learn us rather than us learning our devices.

{Simple Systems} {Complex Forms}
How do simple systems evolve into complex forms?

• Open access to Essential building blocks
• Processor, input subsystems, output subsystems
• Open access to Rules for combining and controlling these subsystems
• Freedom by many to experiment
• A marketplace to reward Success
• OpenMoko provides this stuff
Our Business Model.

OpenMoko
(Open Mobile Communications Platform)

Business 1 Business 2 Business 3 Business n

New Rules New Rules New Rules

Base Rules

OpenMoko Development Kit

Neoforms

Manufacturing & Sales

Neoforms
Now, “Free Your Phone.”

Thanks for Your Time.
Mickey Lauer & Sean Moss-Pultz