# G54MDP
# Mobile Device Programming

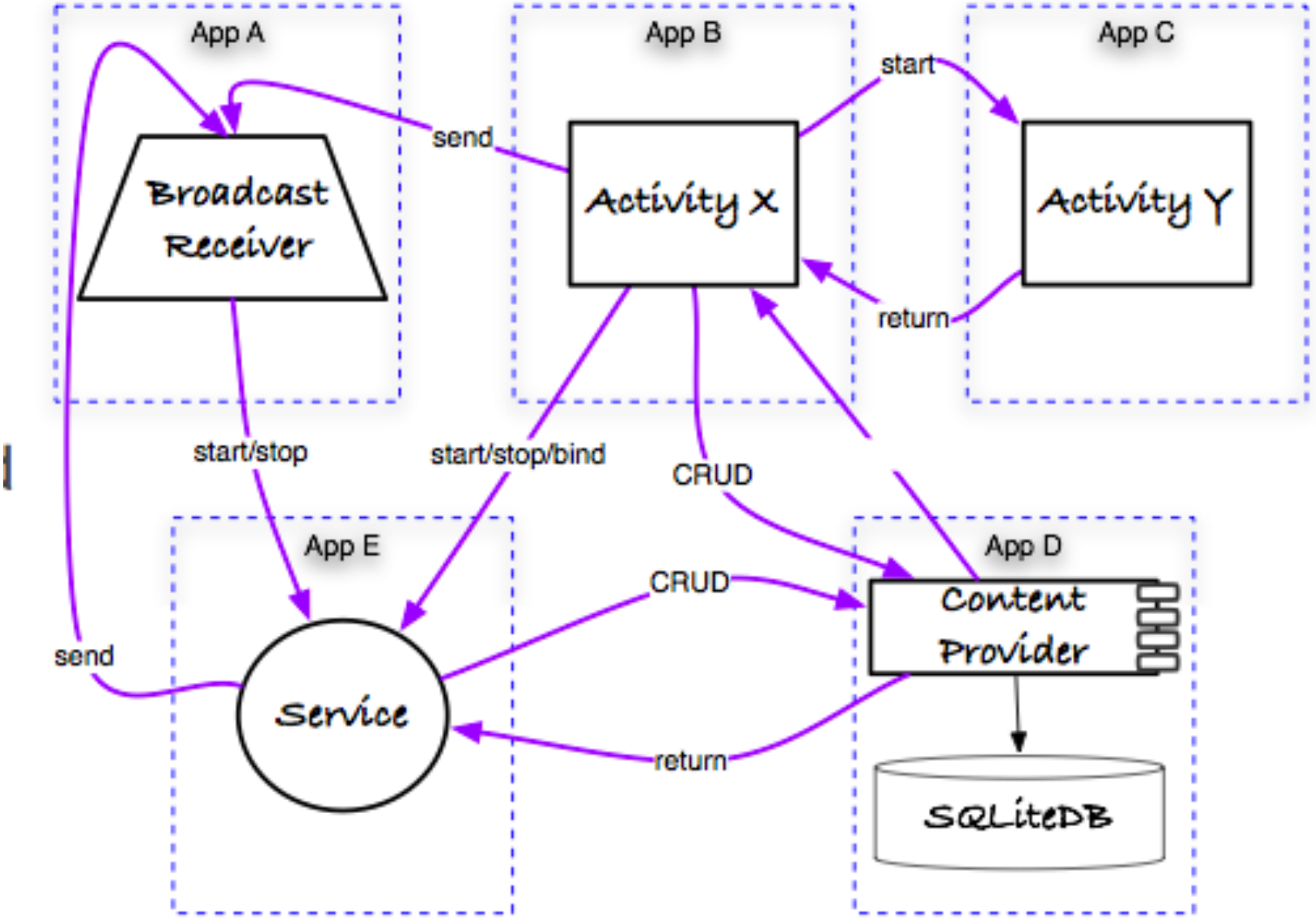
Lecture 14 – Broadcasts, Touch

# 4 Application Components

- Activity
  - A UI for part of a task
- Service
  - A long-running background task
- ContentProvider
  - Storage and provision of data
- …all driven by user activity within an application
  - Potentially a remote application
- What else?

# BroadcastReceiver

- Respond to system-wide broadcast announcements
  - The user has not necessarily done something, but the OS / phone / another application has
    - The screen has turned off, the battery is low, a new SMS has arrived, the phone has booted
  - Can be sent by an application, or received by an application from the OS
- Intents are sent to specific Activities / Services
  - Explicitly or implicitly via Intent filters, URI provision
- Broadcasts are sent to anything that cares to listen
  - System-wide Intent broadcasts


App A
App B
App C
Broadcast Receiver
Activity X
Activity Y
send
start
return
start/stop
start/stop/bind
CRUD
App E
App D
CRUD
Content Provider
send
Service
return
SQLiteDB

# Broadcasts

- Broadcasts are Intents
  - Send an Intent to start an Activity
  - Send an Intent to start a Service
  - Send an Intent to trigger Broadcast Receivers that subscribe to that particular class of Intent
    - Can define our own Broadcast Intents
    - Cannot send system Intents (battery, screen etc), as the security model prevents it
      - Why?
  - Again a messaging wrapper around Binder

# BroadcastReceiver

- Declare in AndroidManifest.xml
  - <activity>
  - <service>
  - <provider>
  - **<receiver>**
- Specify broadcast intents we are interested in
  - Listening to system intents may require certain permissions
    - i.e. phone state – why?
- Receiver registered at boot-time / install-time
  - Again, another **entry point** to our application

```
<receiver android:name="MyReceiver" >
    <intent-filter>
      <action android:name=”com.example.martinbroadcast" />
    </intent-filter>
</receiver>
```

# Implementing a BroadcastReceiver

- Subclass BroadcastReciever
  - Specify which Intents we are interested in receiving
    - Permissions permitting
- Implement the onReceive() method
- Then…
  - Send a Message to our application to change our behavior
    - Reduce the audio volume, play a sound
  - Start an Activity
    - “Never start an Activity in response to an incoming broadcast Intent.”
  - Start a Service
  - Show a Notification
    - Alert the user that there is something that they need to interact with

# Broadcast types

- Several broadcast methods available
- Normal or Ordered
  - Normal
    - Sent to all registered receivers
    - Undefined order
  - Ordered
    - Sequential processing in priority order
- Sticky or Non-Sticky
  - Sticky
    - Store the Intent after the initial broadcast
  - Non-Sticky
    - Discard the Intent after the broadcast
- Local or Global

# Ordered Broadcasts

- sendOrderedBroadcast(intent, receiverPermission)

```
<receiver android:name="MyReceiver" >
    <intent-filter android:priority="2" >
       <action android:name=”com.example.martinbroadcast" />
    </intent-filter>
</receiver>
```

- Preferred receivers given the chance to consume the broadcast before it is delivered to less preferred receivers
  - How many things should notify you that you have a new message?
  - In onReceive()
    - abortBroadcast() to stop the chain of delivery

# Sticky Broadcasts

- Sticky Intents are cached by Android
  - New Intents overwrite older Intents they match
- When BroadcastReceivers are dynamically registered
  - Cached sticky Intents matching the specified IntentFilter are broadcast to the BroadcastReceiver
  - One matching sticky Intent is returned to the caller
- isInitialStickyBroadcast()
  - How old is the broadcast?
  - ACTION_BATTERY_CHANGED

# Sticky Broadcasts

- *Sticky broadcasts are **global** to the system*
- *And because of this, performing a sticky broadcast is multiple orders of magnitude slower than just implementing direct calls within your own app*
  - *IPC for each receiver to register, IPC to the system to send it, IPC from the system back to your app to deliver it, marshalling and unmarshalling of all the data within the Intent over both IPCs*
- *More than that, there is NO protection on them, so any other application can watch your sticky broadcasts, or even send their own values back to you*
- *Now I am really regretting that I made that function public. :/*

# Local Broadcasts

- Like the normal BroadcastReceiver but only supports **local** broadcasts
  - Within the application
- Data broadcast will not leave the application
  - No data leakage
- Other applications cannot send broadcasts to our application
- More efficient than sending a global broadcast
  - No IPC
- Must register / sendBroadcasts using the LocalBroadcastManager
  - Programmatically rather than via the manifest

# Caveats

- Either the sender or receiver of a broadcast can implement permissions
  - Control who can send broadcast intents to my application
  - Control who can receive the intents that my application broadcasts
  - Limit broadcasts to my application components only
- Lifecycle
  - A receiver handling broadcasts is considered to be running in the foreground (albeit briefly)
  - Process is aggressively killed once onReceive() has returned
    - No binding, no startActivityForResult
  - Long-running code should start a Service instead
    - As with any other Activity
- The application receiving the broadcasts must have been explicitly started / not explicitly stopped
  - Applications cannot intercept broadcasts without the user having some awareness that they have given it permission
- Can register / unregister for broadcast events programmatically
  - Most things that the manifest specifies can be done programmatically

# Let’s have a look…

# Facebook App

- A thought experiment
- Which…
  - Activities
  - Services
  - ContentProviders
  - BroadcastReceivers?

# References

- http://developer.android.com/guide/topics/providers/content-providers.html
- http://developer.android.com/guide/components/fundamentals.html
- http://developer.android.com/reference/android/content/BroadcastReceiver.html

# Interfaces

- Android UI interaction metaphor
  - So far has been seen to be similar to PC
  - Different syntax, but similar concepts
    - Widgets, buttons, scrolling
    - onClick events
- Significant difference between mobiles and PC
  - What?


Finder

# Pointers

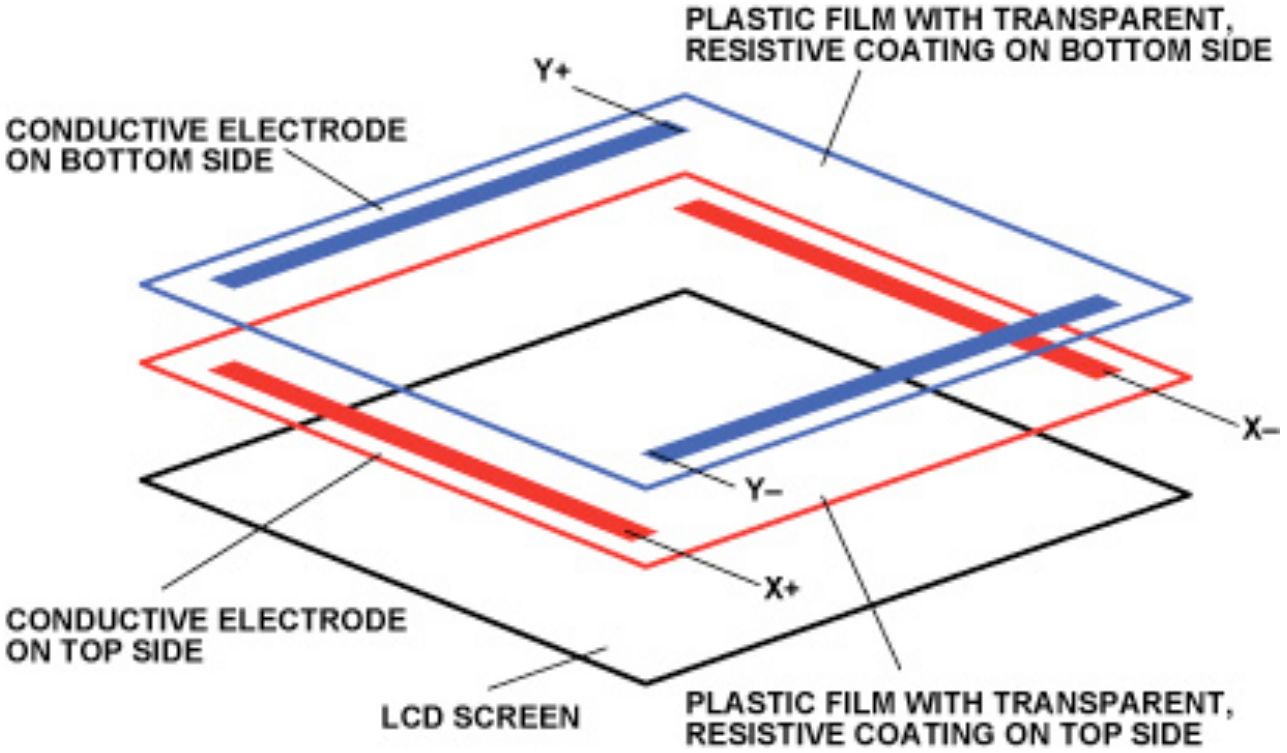
- Pointers provide accurate interaction
  - At a distance
  - Single-pixel accuracy
- Easy to see what you are clicking
- Single position
  - Position always known, even while not interacting
- C.f. HCI / Fitt's law

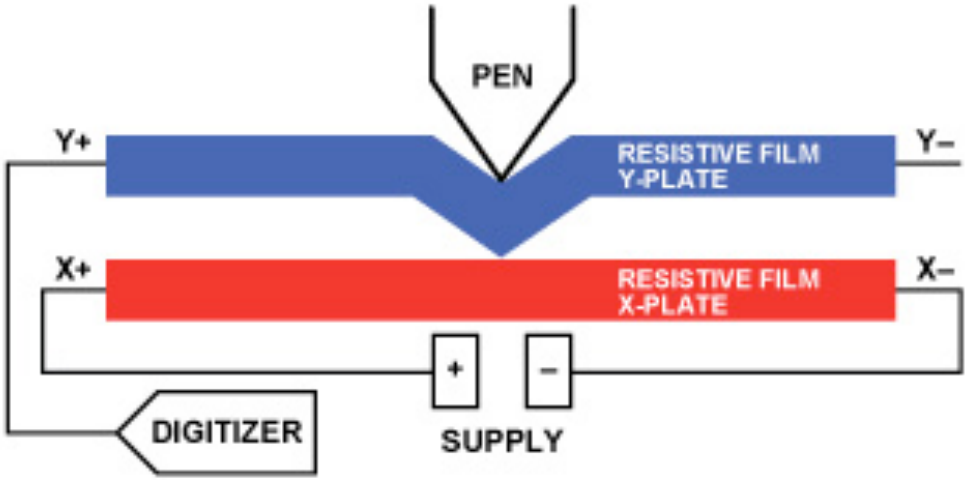
# Touch Input

- Direct Contact
  - Not mediated via a mouse
- Inaccurate
  - Only the general area of touch known
  - My finger is larger than a pixel
- Interaction obscures the display
  - I cannot see through my finger
- Location known only when user touches
- Multi-touch
  - Touching the screen with multiple fingers concurrently
  - Which finger?

# Touch Technologies

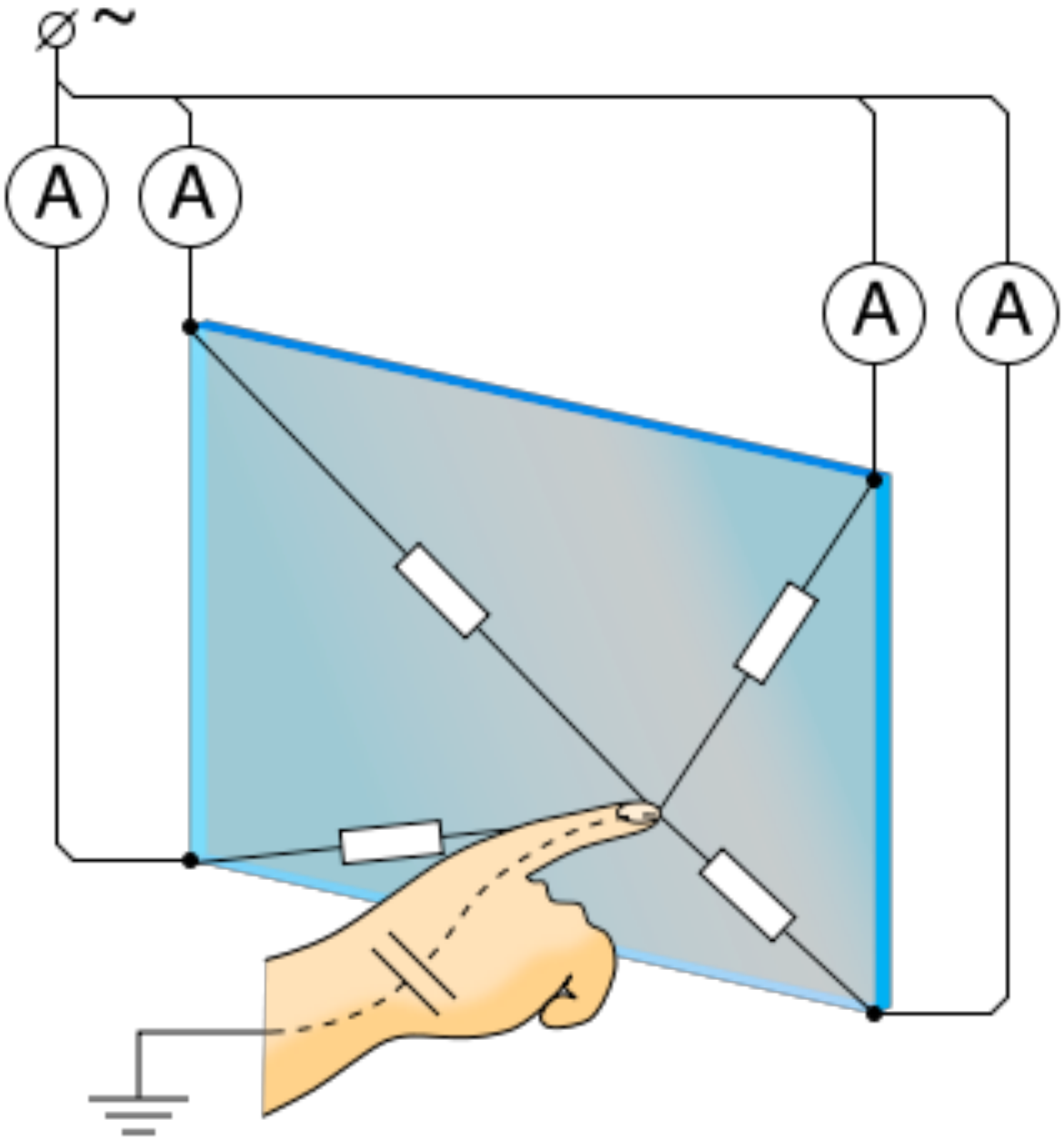
- (Optical / IR / Ultrasonic)
- Resistive
  - Cheap, but poor accuracy
  - Used with a stylus, finger
    - no special properties = “passive”
  - Two sheets of resistive material facing one another
    - Excite alternate axes with a voltage
  - Touch presses specific points together
    - Creates a connected circuit
    - Change in voltage allows determination of position along an axis


PLASTIC FILM WITH TRANSPARENT, RESISTIVE COATING ON BOTTOM SIDE
Y+
CONDUCTIVE ELECTRODE ON BOTTOM SIDE
X–
Y–
X+
CONDUCTIVE ELECTRODE ON TOP SIDE
LCD SCREEN
PLASTIC FILM WITH TRANSPARENT, RESISTIVE COATING ON TOP SIDE


PEN
Y+
RESISTIVE FILM Y-PLATE
Y–
X+
RESISTIVE FILM X-PLATE
X–
+
–
DIGITIZER
SUPPLY

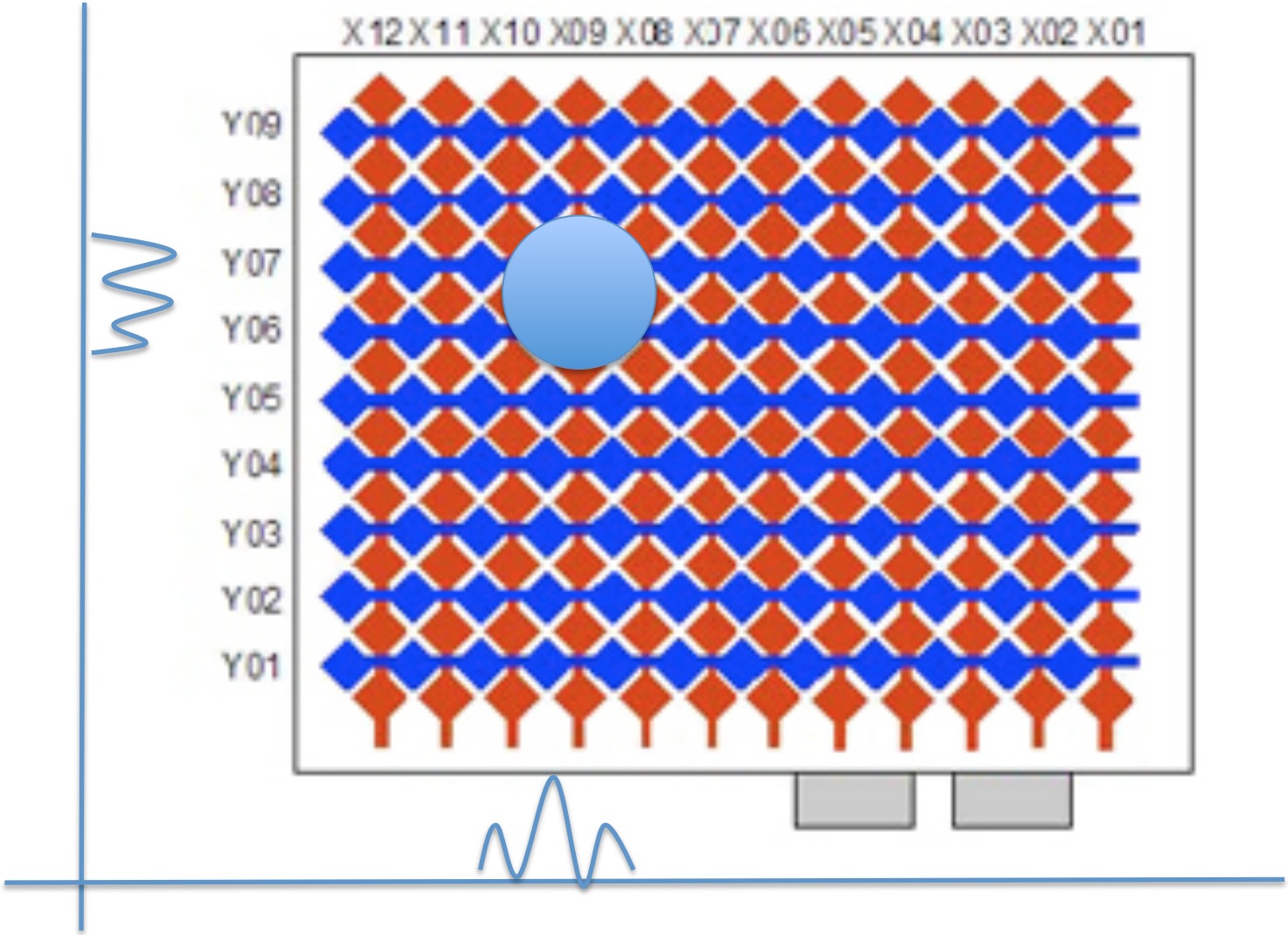
# Touch Technologies

- Capacitive
  - Screen is covered in a capacitive material
    - Indium-tin-oxide
      - Rare, heavy metal
      - Conductive, optically transparent
    - Capacitance = ability to store electric charge
    - Relies on “body capacitance”
      - Human beings act as small capacitors
      - Touching the screen modifies it’s electrostatic field
- Surface capacitance
  - Cover the screen with a uniform conductive material
  - Apply a small voltage to generate an electrostatic field
  - Measure effective capacitance at each corner of the screen
    - When the screen is touched, the capacitance changes
    - The larger the change, the closer to the corner the touch is
    - Combine measurements from all corners = location of the touch

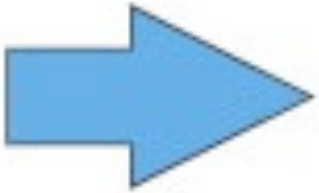

A
A
A
A

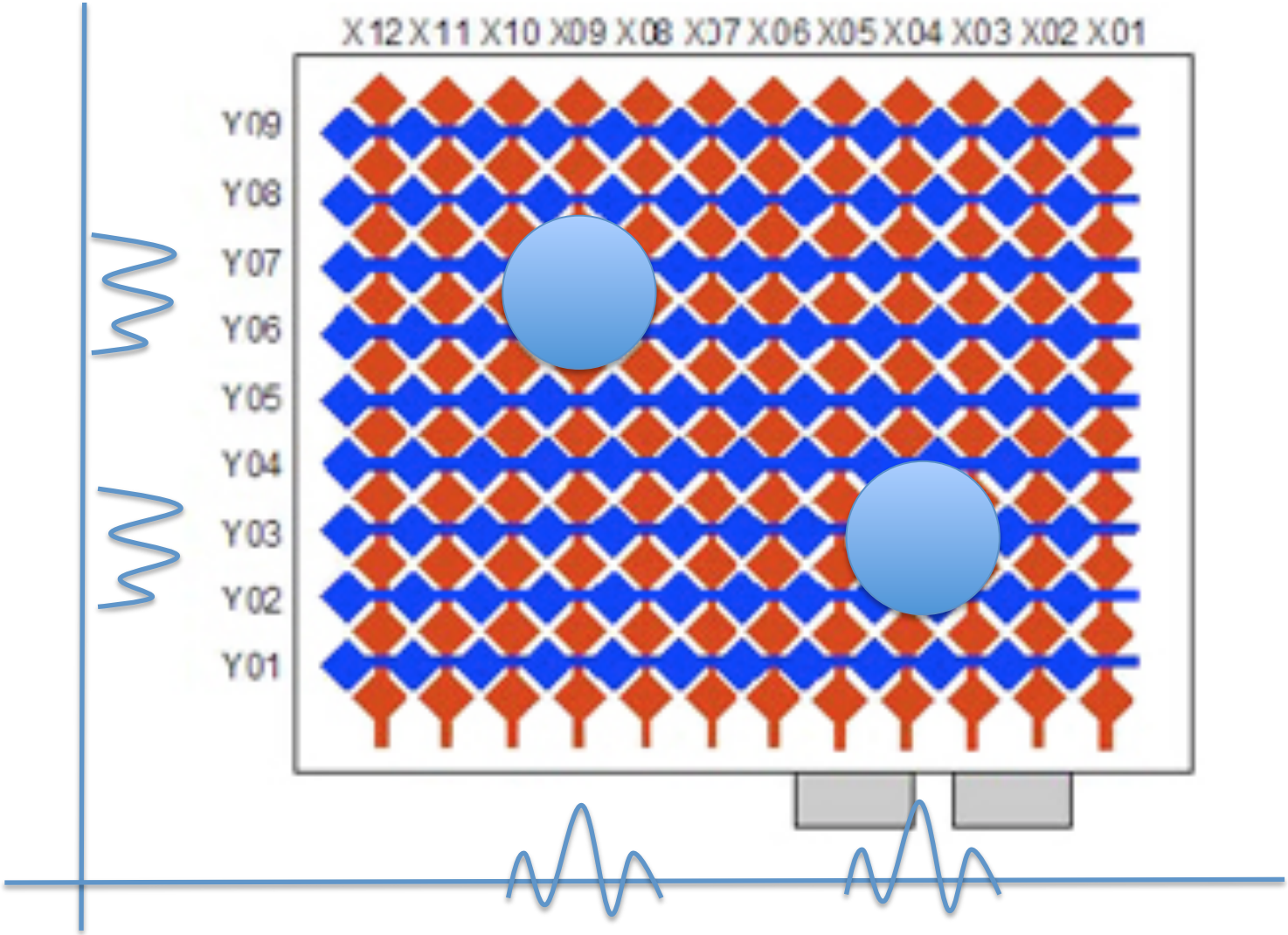
# Touch Technologies

- Projected / Mutual capacitance
  - Material is etched with rows / columns
  - Measure capacitance at each point – more accurate / multi-touch
    - How wide is a finger / what resolution of touch should we have?
    - Original iPhone 10 x 15 = 150 “points”
    - 5mm x 5mm diamond grid
- All capacitive sensing requires an “active” touch
  - Non-conductive materials will not change the electrostatic field
  - Fingers / capacitive glove / capacitive stylus


X12 X11 X10 X09 X08 X07 X06 X05 X04 X03 X02 X01
Y09
Y08
Y07
Y06
Y05
Y04
Y03
Y02
Y01

| | | | |
|---|---|---|---|
| 1 | 5 | 10 | 5 |
| 5 | 15 | 25 | 10 |
| 2 | 12 | 15 | 5 |
| 0 | 2 | 5 | 1 |



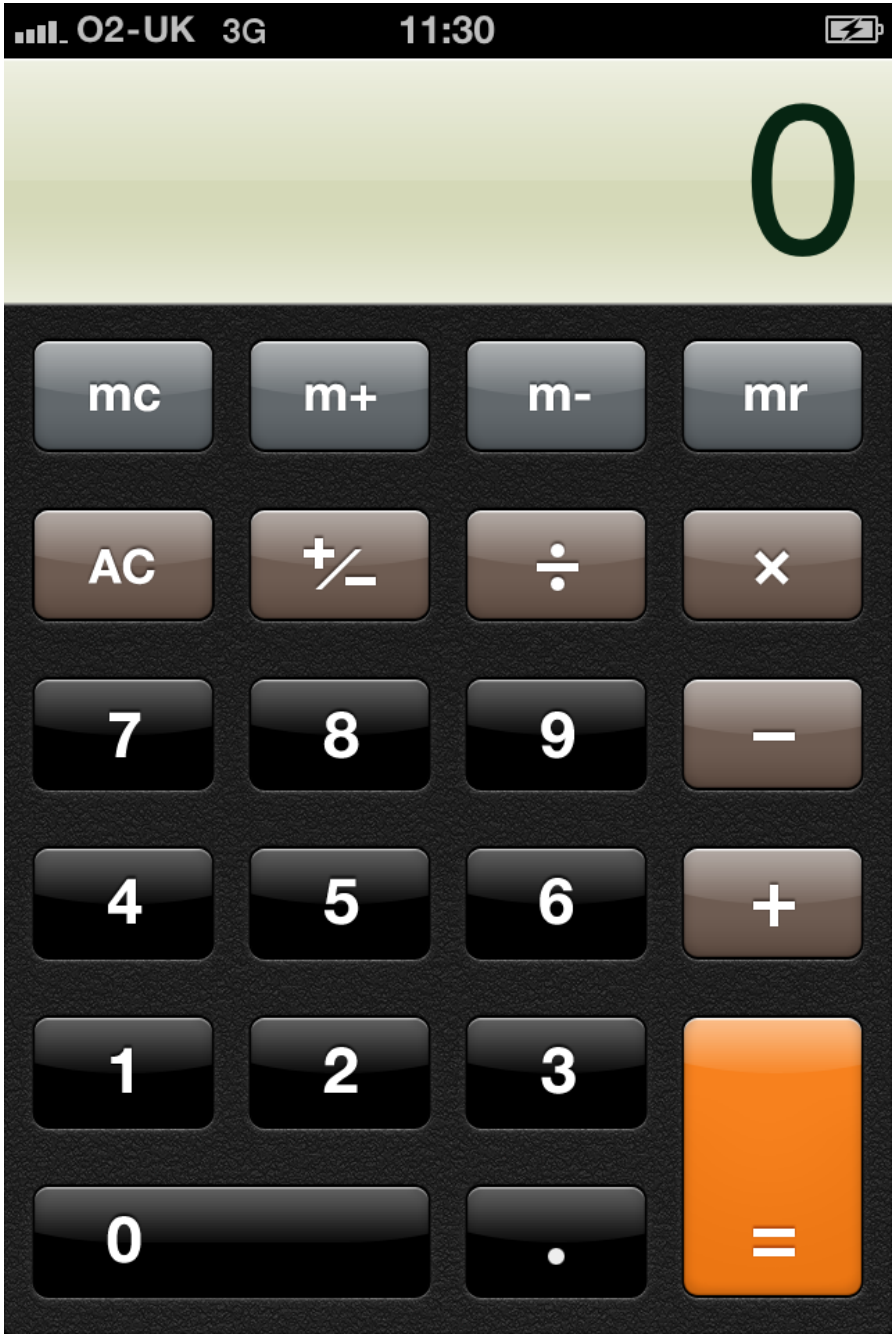

X12 X11 X10 X09 X08 X07 X06 X05 X04 X03 X02 X01
Y09
Y08
Y07
Y06
Y05
Y04
Y03
Y02
Y01

# Touch and the UI

- Touch relies on finger contact with the display
  - This has to alter the way we design our displays
- Size of the finger sets the properties of the UI, not the size of a display
  - 5” vs 1200 pixels
  - The size of ‘buttons’ must be big enough that the user can touch them
  - Ditto the spacing between them
  - If they get too small, or too close together then it will be hard for the user to accurately use them
- Size is fixed relative to display

# What size?

- Depends on size of finger relative to display
- Not number of pixels
- Apple recommend about 44x44 points for the iPhone (480x320 pixel screen)
- Equates to roughly the size of a finger

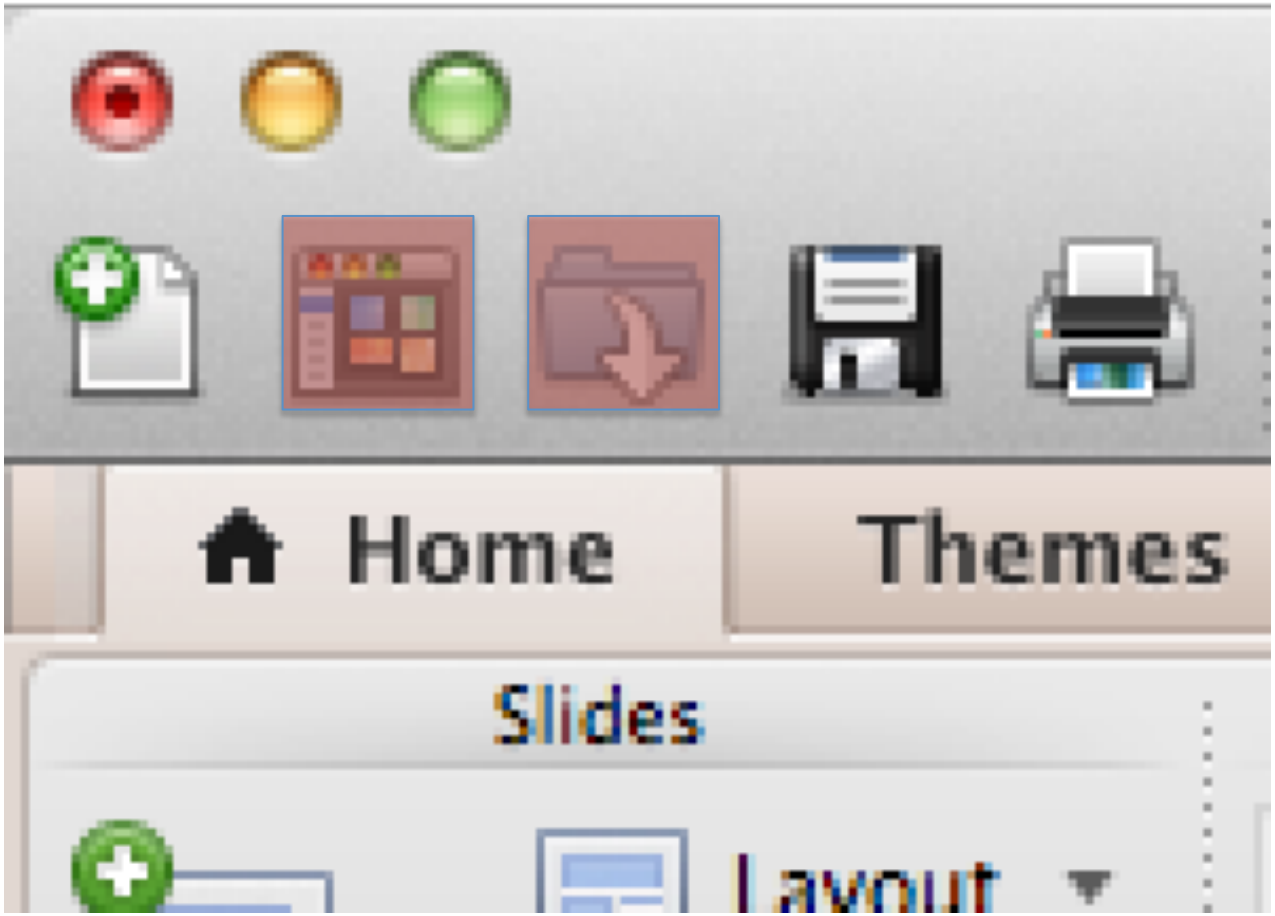

O2-UK 3G
11:30
0
mc
m+
m-
mr
AC
+/-
÷
×
7
8
9
−
4
5
6
+
1
2
3
0
.
=

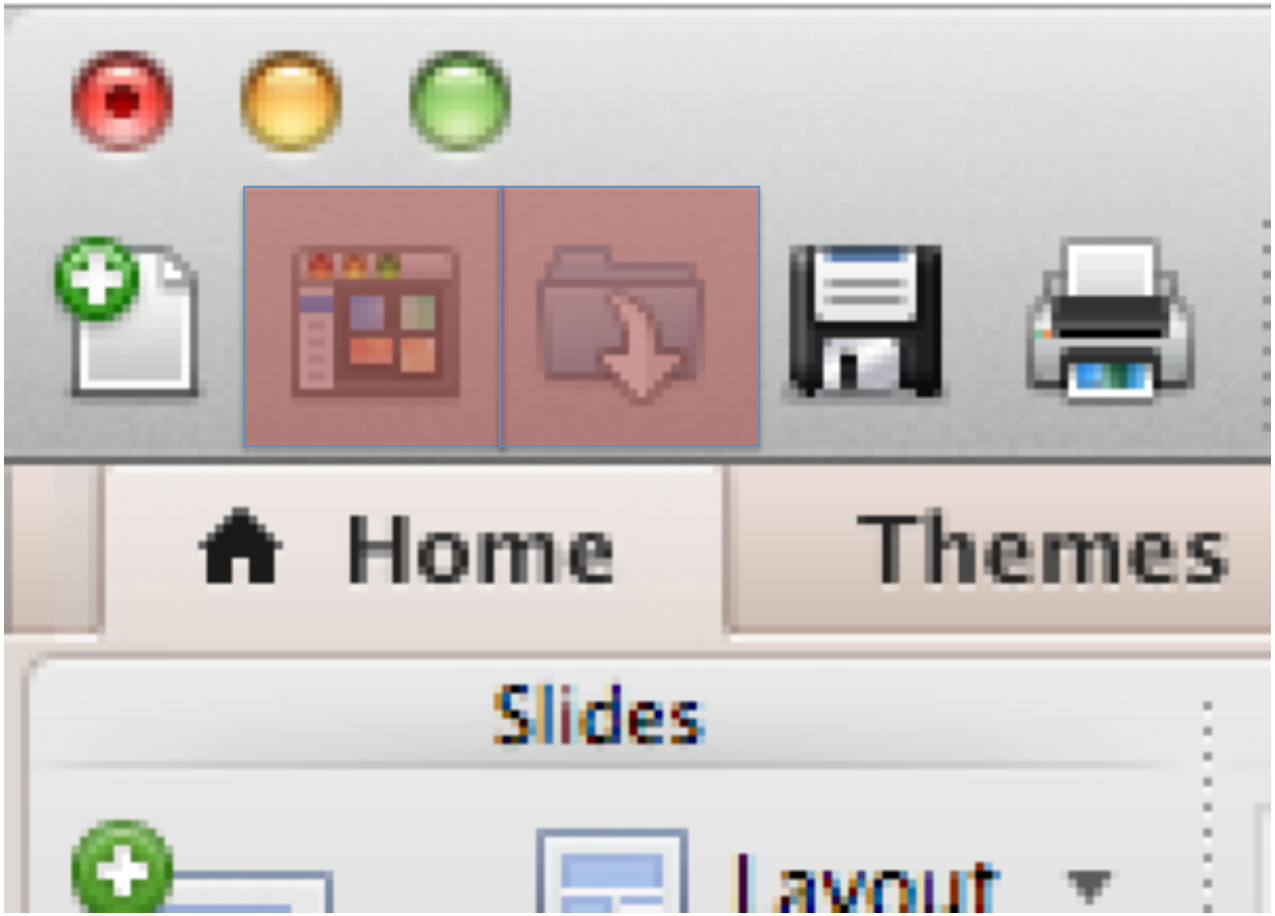
# What size?

- Things are slightly more complicated on Android
- Display size, shape and resolution varies considerably from device to device
- A button that is the right size on one device would be too small/large on another
- Hence, the use of relative layouts
  - Not just about filling white-space

# Hit-box size

- Has the user touched a widget or not
  - A touch has “hit” within a bounding box
- Think about the handles used to interact with a text frame in Word or something
- Need to be big enough that the user can accurately touch them
- Or rather the hit test area needs to big enough
- Potentially decouple visual area from tested area


Home
Themes
Slides
Layout


Home
Themes
Slides
Layout

# Device size

- On mobile devices, the UI is constrained by the ratio of the device size - finger size - resolution
  - Apple 44x44 pixels
  - Microsoft min 26 pixels, ideal 34
  - Nokia 1cmx1cm, 28x28 pixels
- A UI that works on a 10” display won’t necessarily “work” on a 7”
  - Steve Jobs comments about “not having to sand your fingers”
- Is a 7” device a distinct enough class of device?

Natural finger position,
completely covers visible target

Using finger tip shows target,
but have to reposition hand

# Fingers

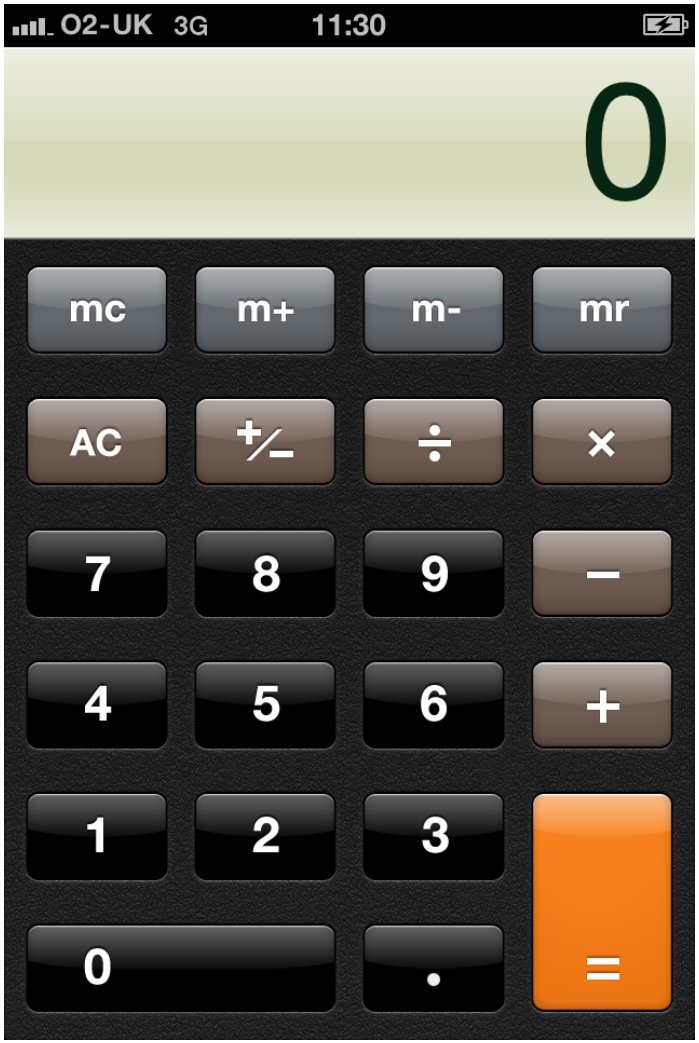
- Small touch targets
  - Touch errors
    - Finger overlaps on to neighbouring hit boxes
  - Fitts law
  - Thumbs are even bigger
- Average index finger width is 16-20mm
  - MIT study
  - 47-57 pixels on an average device
- Average thumb width is 25mm
  - 72 pixels
  - Edges of the control are visible

# Visual Feedback

- On a phone, it is quite possible for a finger to obscure the button completely
- Flashing button effect would effectively become invisible for a small button
  - Common in desktop interfaces
- Need to find alternative approaches

# iPhone calculator

- iPhone calculator takes several approaches
- Some buttons have obvious effects (e.g. digit entry)
- For operators, it leaves the button highlighted
  - Can be seen when the finger is removed



# Visual Feedback

- Other options include making the buttons bigger
  - Again, on the iPhone many buttons are the full width of the phone
  - Recall that the user is only actively engaging in one task using the phone at a time
    - Break the task into small, discrete Activities
    - Make full use of the available space
      - If not enough space, make more activities
- Need to think about how to give the user either implicit or explicit feedback that the touch was registered


AT&T
12/16 - 9:09 - 28MB
Trash
1 of 1
From:
Gmail Team
Details
Gmail is different. Here's what you need to know.
July 11, 2008 5:38 PM
Messages that are easy to find,
Reply
Reply All
Forward
Cancel