# G54MDP
# Mobile Device Programming

Lecture 15 – Touch

# Facebook App

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

# 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

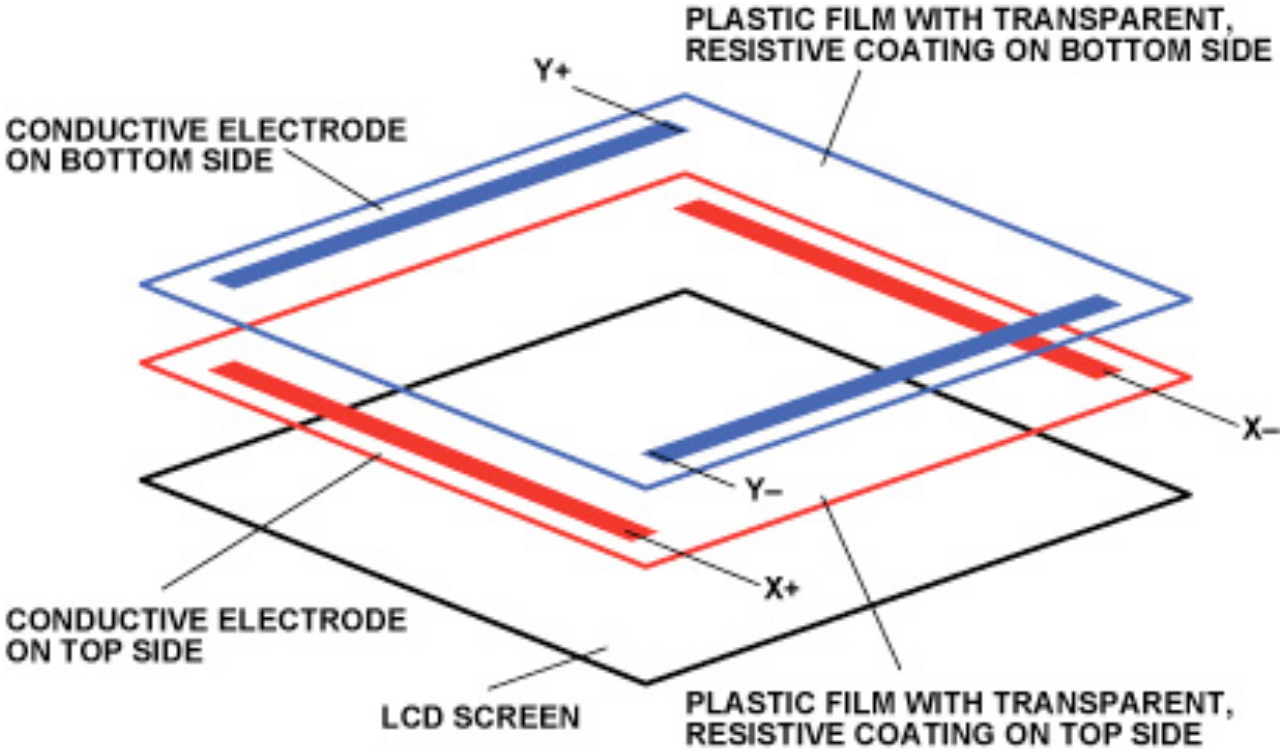
- Pointers provide accurate interaction
  - At a distance
  - Single-pixel accuracy
- Easy to see what you are clicking
- Single position
  - Position always known, even while not interacting
- C.f. HCI / Fitt's law

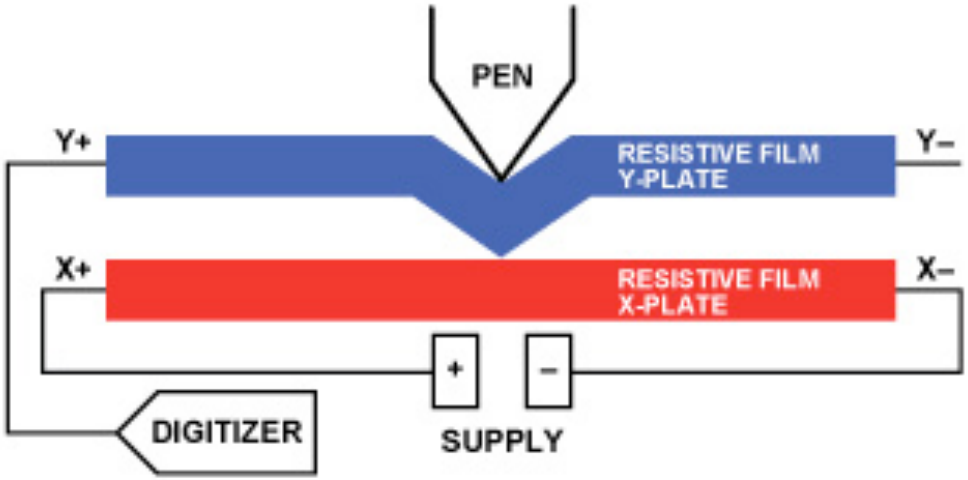
# Touch Input

- 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

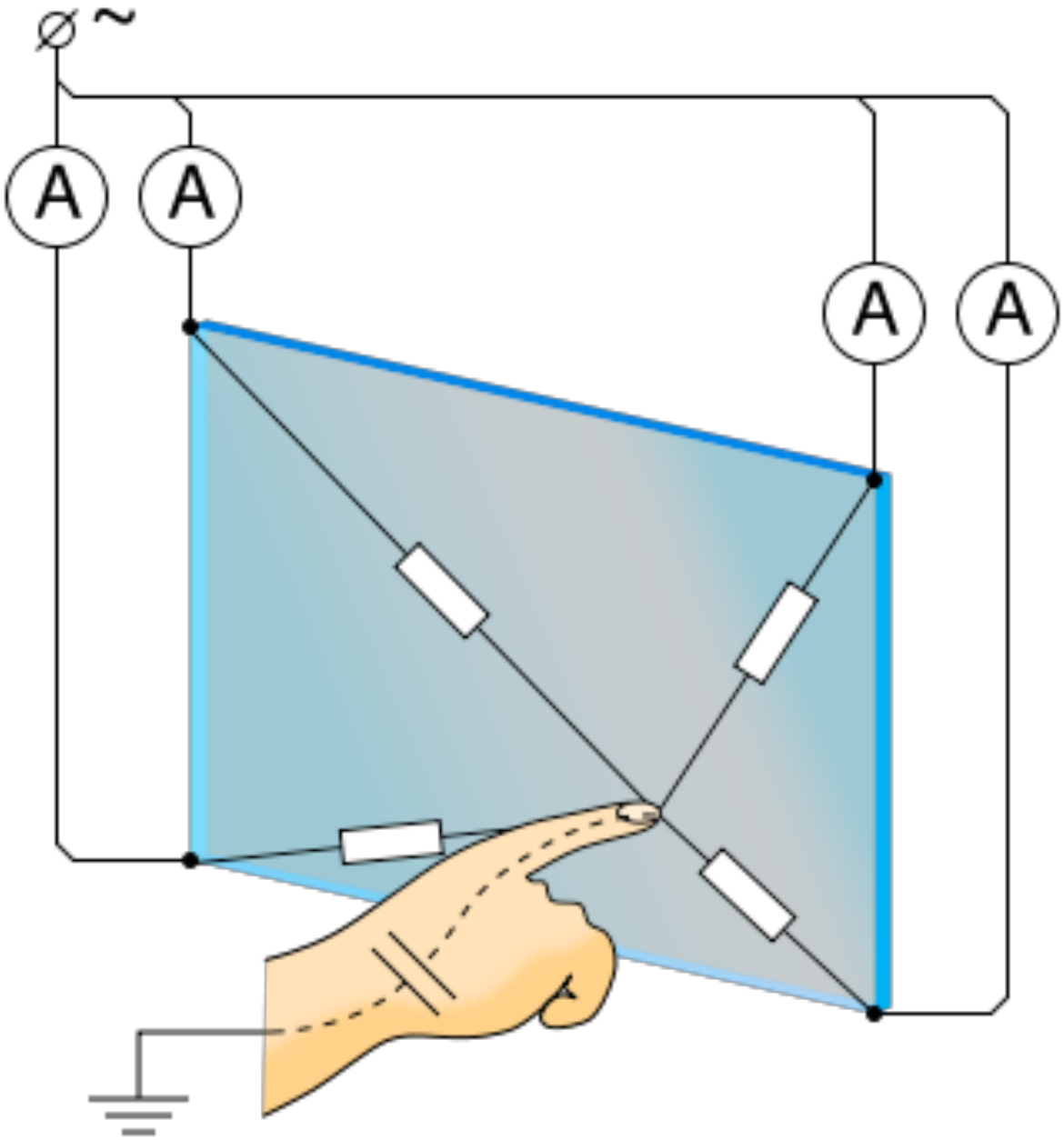
- (Optical / IR / Ultrasonic)
- Resistive
  - Cheap, but poor accuracy
  - Used with a stylus, finger
    - no special properties = “passive”
  - Two sheets of resistive material facing one another
    - Excite alternate axes with a voltage
  - Touch presses specific points together
    - Creates a connected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

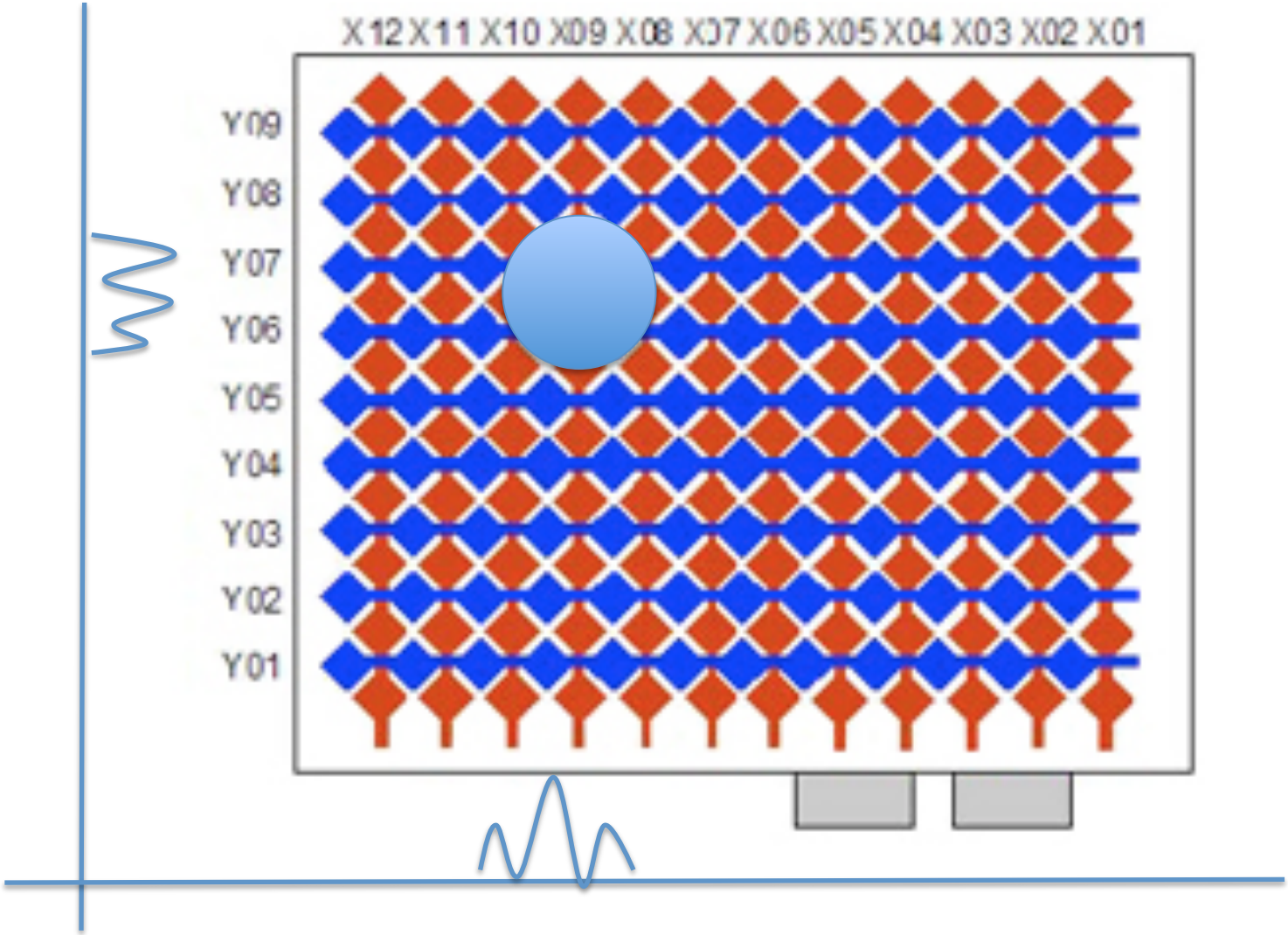
# Touch Technologies

- 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

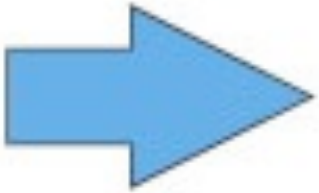

A
A
A
A

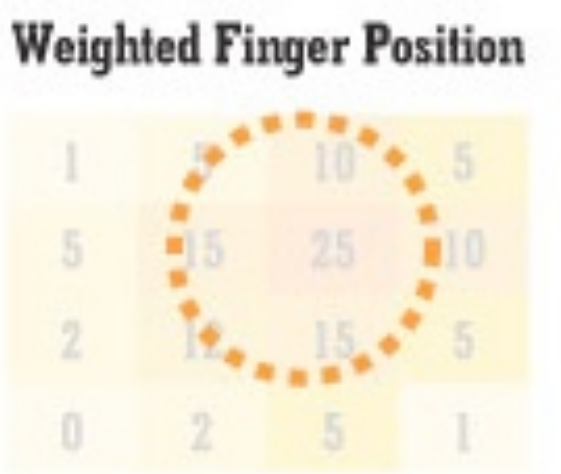
# Touch Technologies

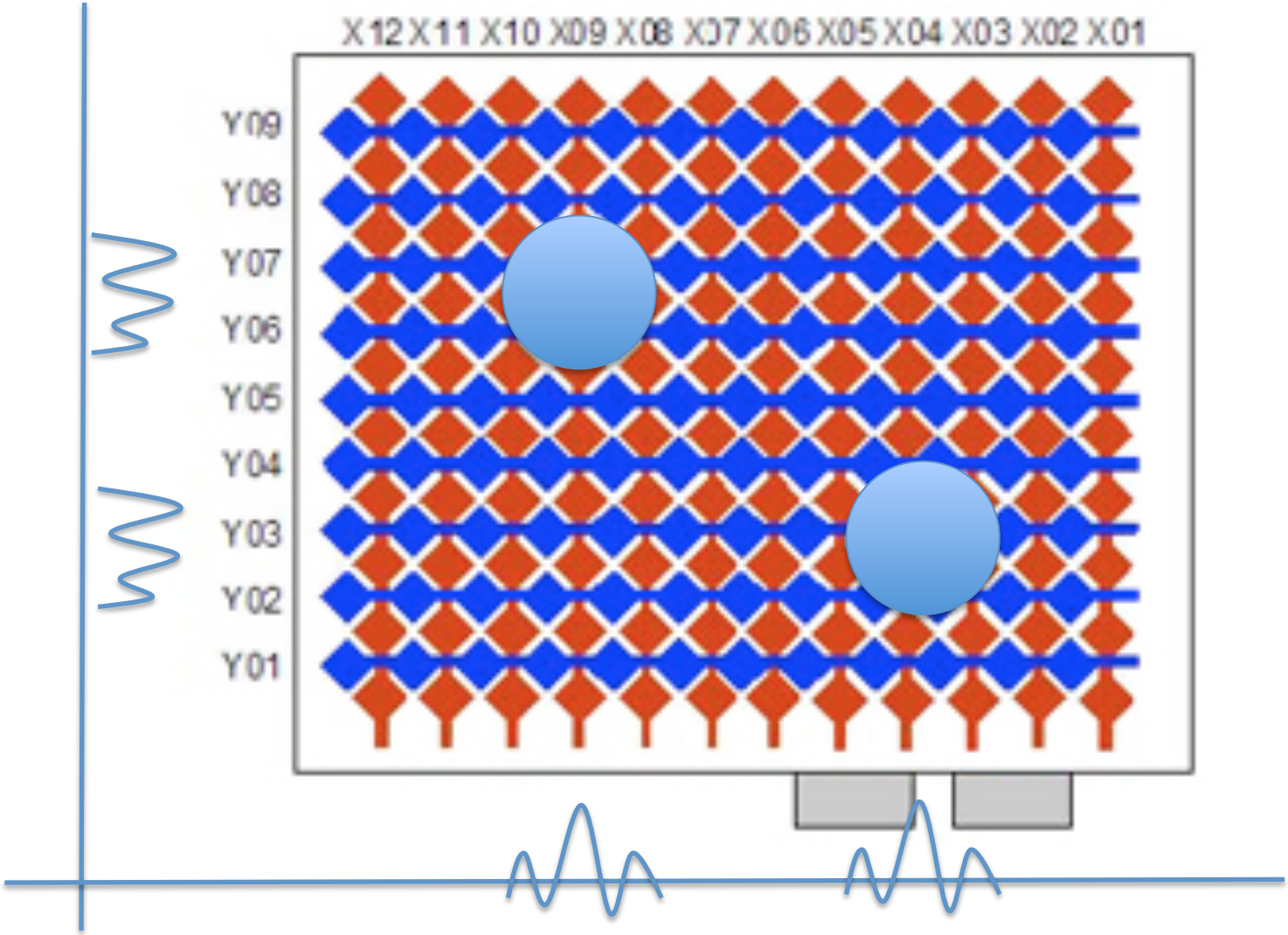
- Projected / Mutual capacitance
  - Material is etched with rows / columns
  - Measure capacitance at each point – more accurate / multi-touch
    - How wide is a finger / what resolution of touch should we have?
    - Original iPhone 10 x 15 = 150 “points”
    - 5mm x 5mm diamond grid
- All capacitive 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 |





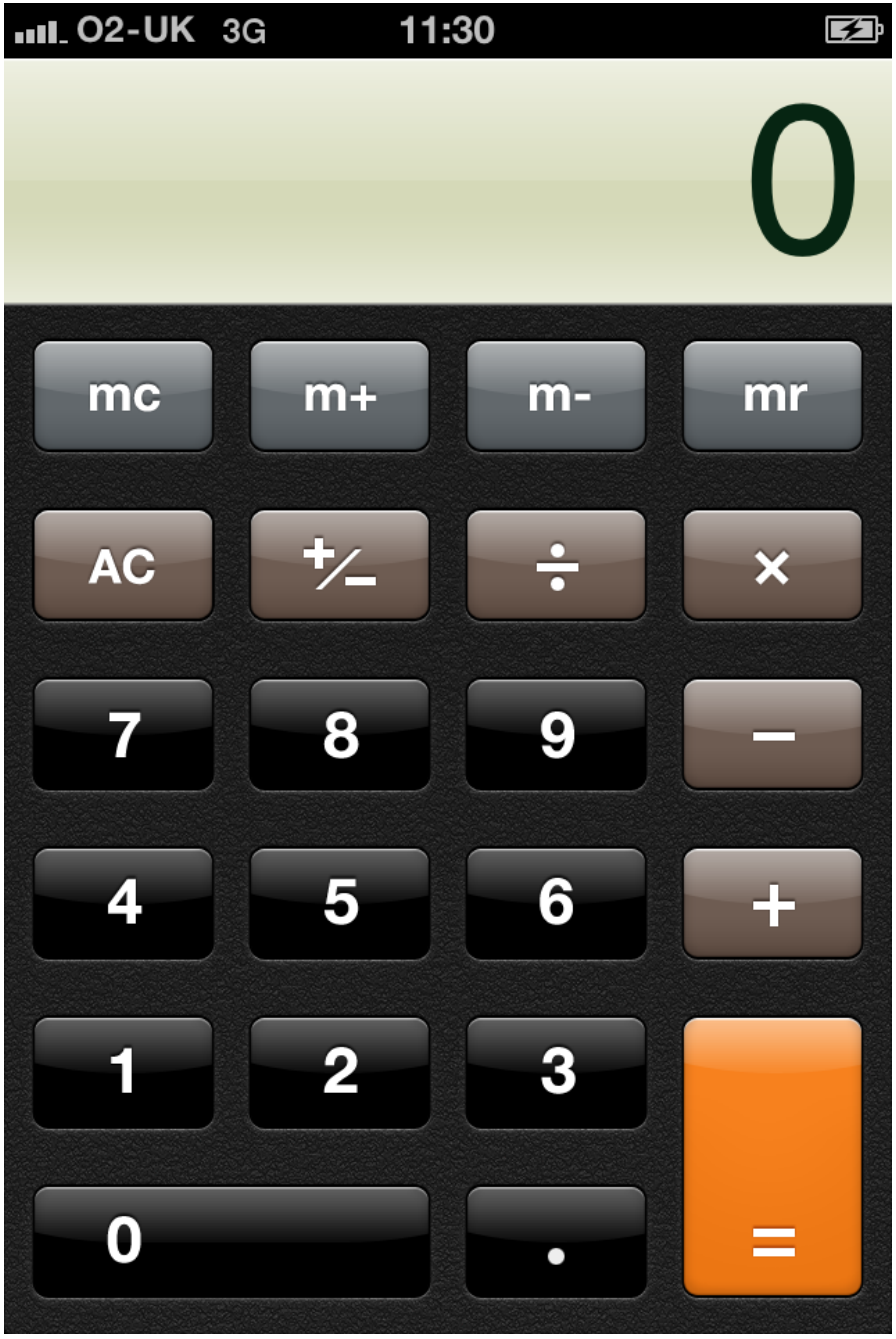

X12 X11 X10 X09 X08 X07 X06 X05 X04 X03 X02 X01
Y09
Y08
Y07
Y06
Y05
Y04
Y03
Y02
Y01

# 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

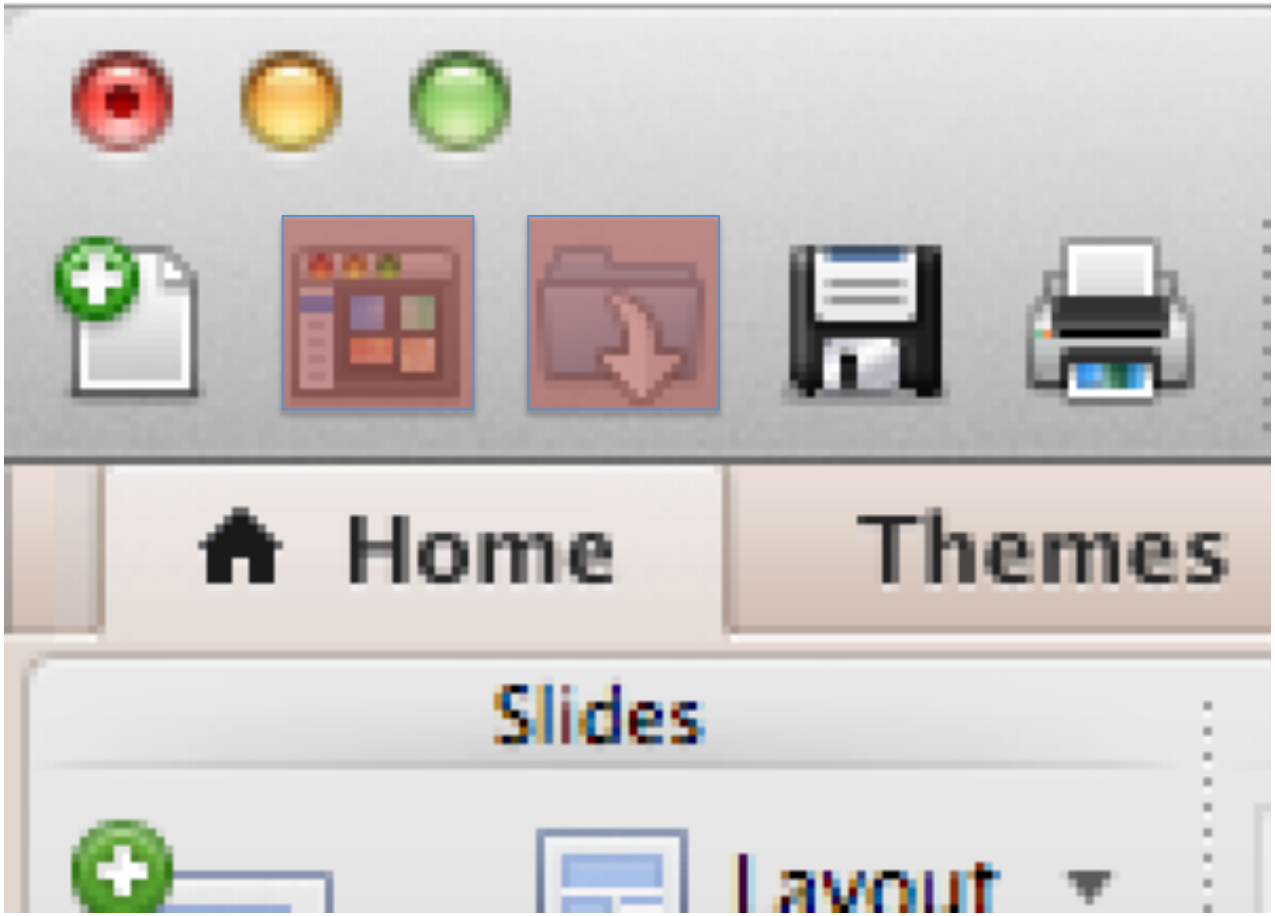

O2-UK 3G
11:30
0
mc
m+
m-
mr
AC
+/-
÷
×
7
8
9
−
4
5
6
+
1
2
3
0
.
=

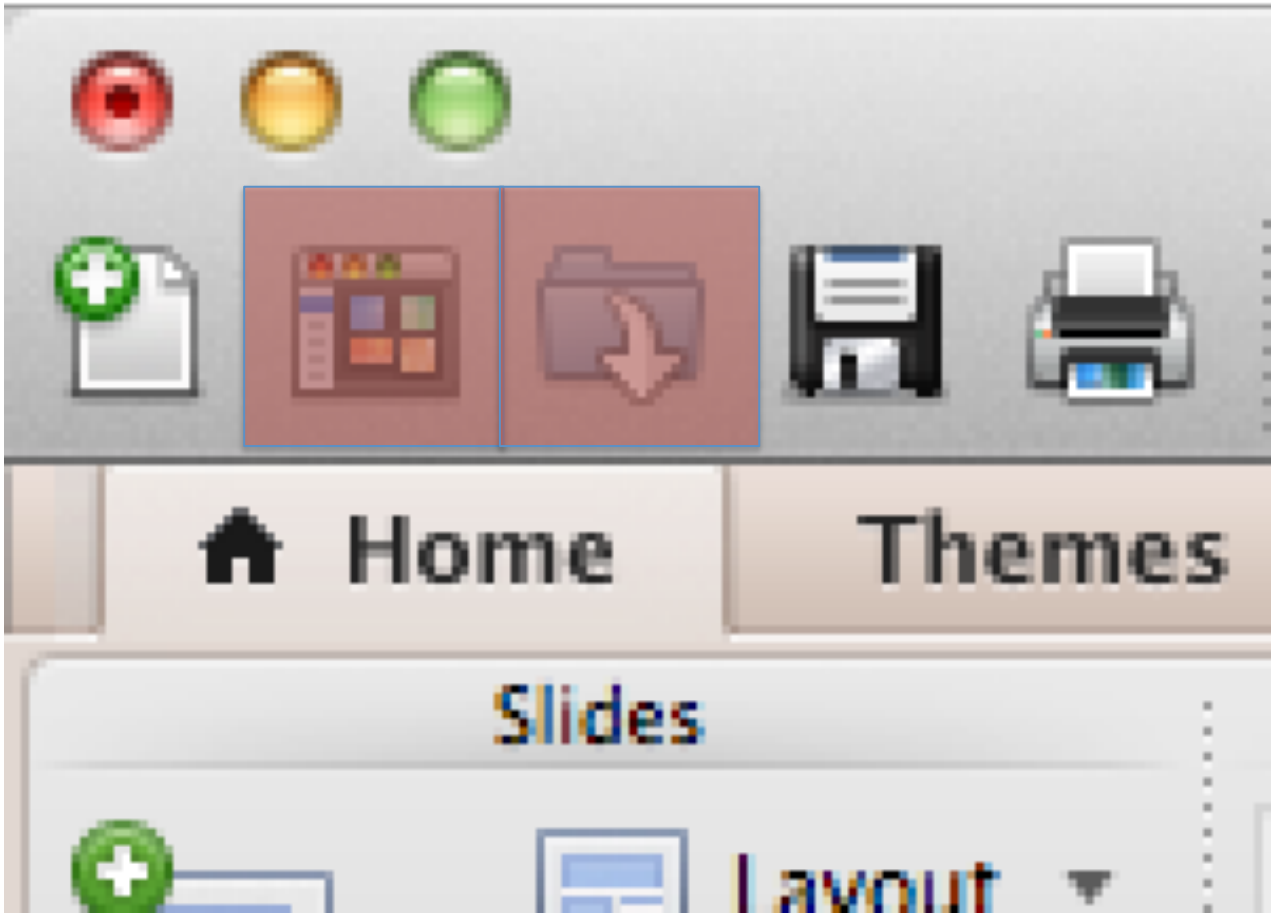
# What size?

- 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

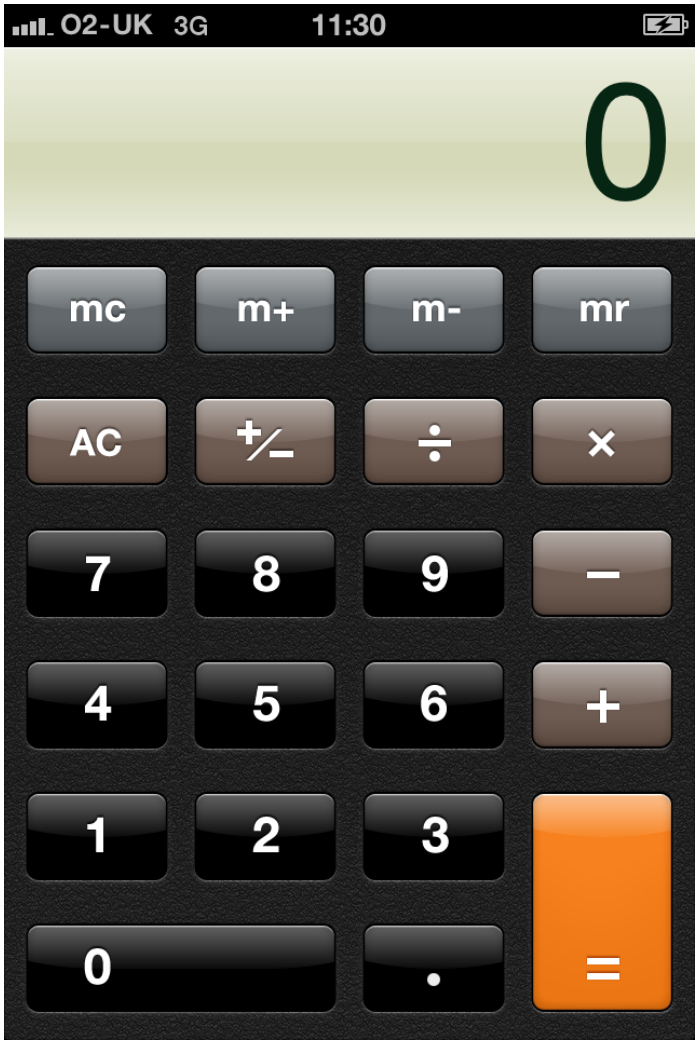
- Small touch targets
  - Touch errors
    - Finger overlaps on to neighbouring hit boxes
  - Fitts law
  - Thumbs are even bigger
- Average index finger width is 16-20mm
  - MIT study
  - 47-57 pixels on an average device
- Average thumb width is 25mm
  - 72 pixels
  - Edges of the control are visible

# 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

# Touch Metaphors

- Finger location only known when user touches
  - Desktop UI paradigms
    - mouseovers, hover, no longer possible
  - Instead need to create new interaction mechanisms
    - e.g. touch and hold without moving
      - Analogous to hover?
- Touch (and especially multitouch) provide opportunities for new UI paradigms
  - Particularly popular on mobiles are the use of **gestures**
  - These are complex touch movements made on the device to signify an operation

# Gestures

- Tap
- Hold
- Drag
- Pinch
- Rotate
- Swipe / fling
- ?

# Programming for Touch

- Much like programming for mouse
  - onClick, onMouseDown, onMouseMove
- Handle a sequence of events
  - TouchBegin
  - TouchMoved
  - TouchEnded
- There just happens to be more than one of them
  - Need to parse a sequence of events into a gesture
  - A gesture is a series of touch events that occur over a period of **time**
    - Change in Touch

# Touch Events

- Two ways to get touch events
- Either register a new OnTouchListener with a view, (with setOnTouchListener(…))
- Or implement onTouchEvent() in a custom View
- Either way, we are delivered a series of MotionEvents

# MotionEvent

- This object encapsulates information about Touch events
  - Sent went a touch begins (ACTION_DOWN)
  - When the finger moves (ACTION_MOVE)
  - And finally when the touch ends (ACTION_UP)
- Additional events also sent for multitouch
- Rely on the underlying OS to translate a “fat finger” into a discrete pointer

# Action Down

- A gesture starts when a finger is pressed
- A MotionEvent is generated for this ACTION_DOWN
- Can find the action by calling getAction()
- This can also have the identifier of the 'pointer' so use getActionMasked() instead
  - If we care about multi-touch

# Action Move

- As the finger moves, a series of ACTION_MOVE events will be sent
- Can find the new position using getX() and getY() (return floats)
  - Touch resolution is not necessarily the same as screen resolution
- Note that Android may bundle up a series of touch events
  - Ability to get 'historic' touches

# Action Up

- A gesture ends in three ways, the normal is for an ACTION_UP event
  - This signifies that the (last) finger has been taken off the display
- If the touch event has been cancelled for any reason
  - (e.g. phone rings), then an ACTION_CANCEL event is sent
  - ACTION_OUTSIDE if the finger moves outside the relevant view

# Single Touch

- So a touch gesture will be formed by
  - A single ACTION_DOWN
  - Zero or more ACTION_MOVE
  - An ACTION_UP to finish
- Can use the data from these to perform some interaction
  - Use position delta to move an object around the screen
    - Delta = change from original
  - Use movement velocity for a swipe / fling

# Dragging / Scrolling

- Store original location of thing to move
  - Store x,y-pair from ACTION_DOWN
- Calculate delta from stored value and value returned from ACTION_MOVE or ACTION_UP
  - Change the location of thing being moved by adding delta to original location
  - Note: need to adjust as position returned is relative to View origin

# Swipe / Fling

- Can do similar for a swipe / fling
- Rather than moving the object, calculate the velocity with which it is moving
  - Speed
  - Direction
- On ACTION_UP
  - Continue to move the object with that velocity
- Gives the user the impression of “flinging” UI elements across the screen
  - Obvious visual feedback