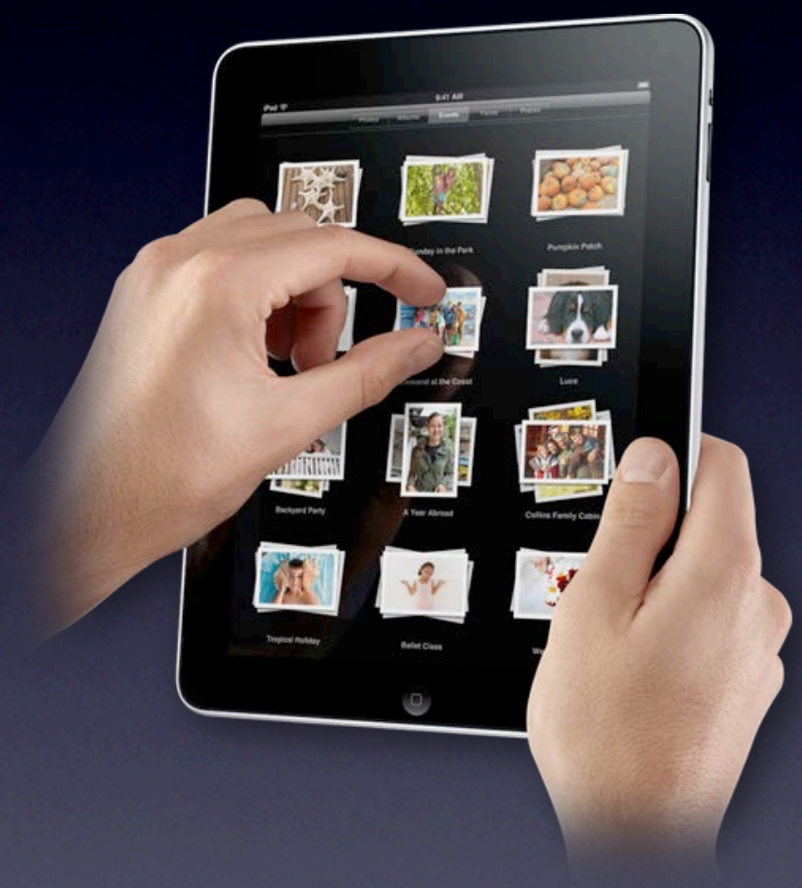
Touch

Steven R. Bagley

So far…

- Android UI stuff seen so far is similar to PC
- Different syntax, but similar concepts
- However, there's a huge difference between mobiles and PC



Discuss the difference in terms of input devices

Pointers

- Pointers provide accurate interaction
- At a distance
- Single-pixel accuracy

- Easy to see what you are clicking
- Single position
- Position always known

Touch-input

- Direct Contact
- Inaccurate — general area of touch known
- Interaction obscures the display
- Location known only when user touches
- Multitouch

Multitouch technology

- Resistive touch screens
- Capacitive touch screens
- Optical methods — Perceptive pixel

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

UI components

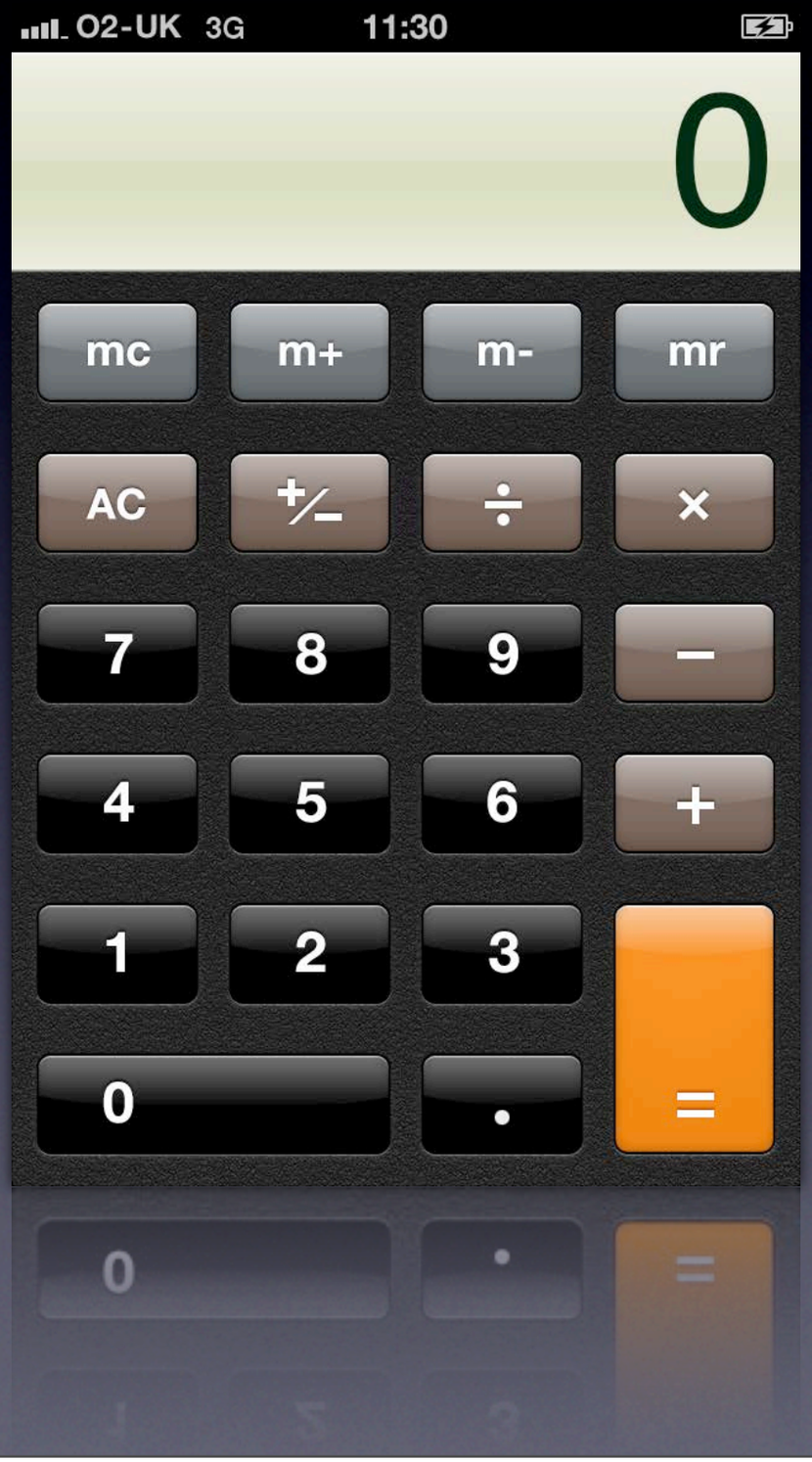
- The size of ‘buttons’ must be big enough that the user can touch them
- Ditto the spacing between them
- If they get too small, or too close together then it will be hard for the user to accurately use them
- Size is fixed relative to display

What size?

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

Even with the iPhone 4's retina display the 'resolution' is unchanged -- you can position accurately on half a point boundaries.

What size?



Even with the iPhone 4's retina display the 'resolution' is unchanged -- you can position accurately on half a point boundaries.

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

Hit-testing

- 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
- Decouple visual area from tested area…

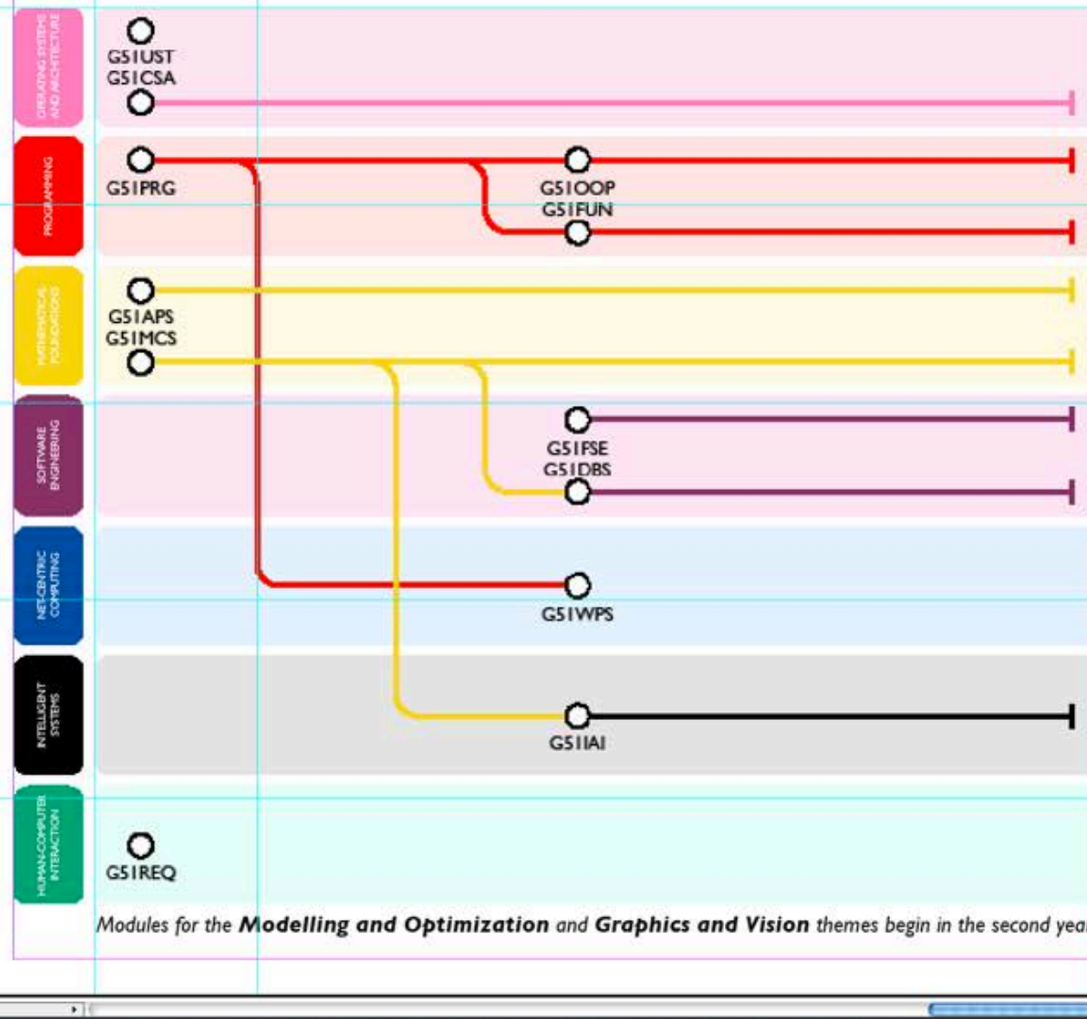
Hit-testing -- checking whether a click (or touch in this case) has hit a particular object

i.e look for hits in a much larger area than is visually shown

CourseStructure1st.indd @ 50%

Computer Science
First Year Course Structure

The University of Nottingham



*Modules for the **Modelling and Optimization** and **Graphics and Vision** themes begin in the second year*

Module Code	Module Title
Operating Systems and Architecture	
G51UST	Unix and Software Tools
G51CSA	Computer Systems Architecture
Programming	
G51PRG	Introduction to Programming (in C)
G51OOP	Object-Oriented Programming (in Java)
G51FUN	Functional Programming (in Haskell)
Mathematical Foundations	
G51APS	Algorithmic Problem Solving
G51MCS	Mathematics for Computer Scientists
Software Engineering	
G51FSE	Foundations of Software Engineering
G51DBS	Database Systems
Net-Centric Computing	
G51WPS	Web Programming and Scripting
Intelligent Systems	
G51IAI	Introduction to Artifical Intelligence
Human-Computer Interaction	
G51REQ	Requirements Capture

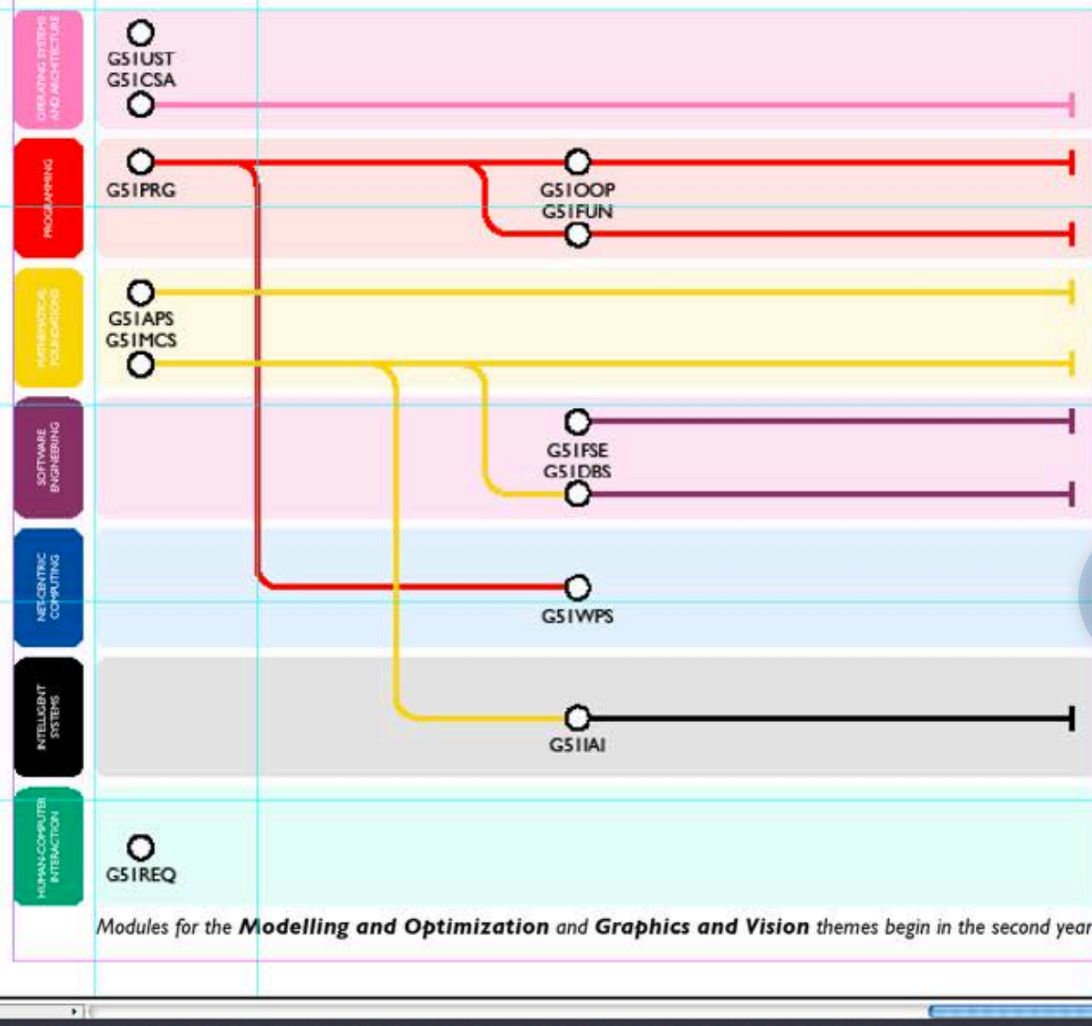
Compulsory Module Optional Module

Following Module builds on material covered in earlier module

CourseStructure1st.indd @ 50%

Computer Science
First Year Course Structure

The University of Nottingham



*Modules for the **Modelling and Optimization** and **Graphics and Vision** themes begin in the second year*

Module Code	Module Title
Operating Systems and Architecture	
G51UST	Unix and Software Tools
G51CSA	Computer Systems Architecture
Programming	
G51PRG	Introduction to Programming (in C)
G51OOP	Object-Oriented Programming (in Java)
G51FUN	Functional Programming (in Haskell)
Mathematical Foundations	
G51APS	Algorithmic Problem Solving
G51MCS	Mathematics for Computer Scientists
Software Engineering	
G51FSE	Foundations of Software Engineering
G51DBS	Database Systems
Net-Centric Computing	
G51WPS	Web Programming and Scripting
Intelligent Systems	
G51IAI	Introduction to Artifical Intelligence
Human-Computer Interaction	
G51REQ	Requirements Capture

Compulsory Module Optional Module

Following Module builds on material covered in earlier module

Device size

- On mobile devices, the UI is constrained by the ratio of the device size to finger size
- A UI that works on a 10” display won’t work on a 7”
- Hence, Steve Jobs comments about having to sand your fingers
- Is a 7” device a distinct class of device?

Or big old phone?

Visual Feedback

- Important that the user gets feedback to their interaction
- ‘Did I miss the button, or has the program hung?’
- On a computer, we can flash the button
- Easy to see since, pointer is very small

Visual Feedback

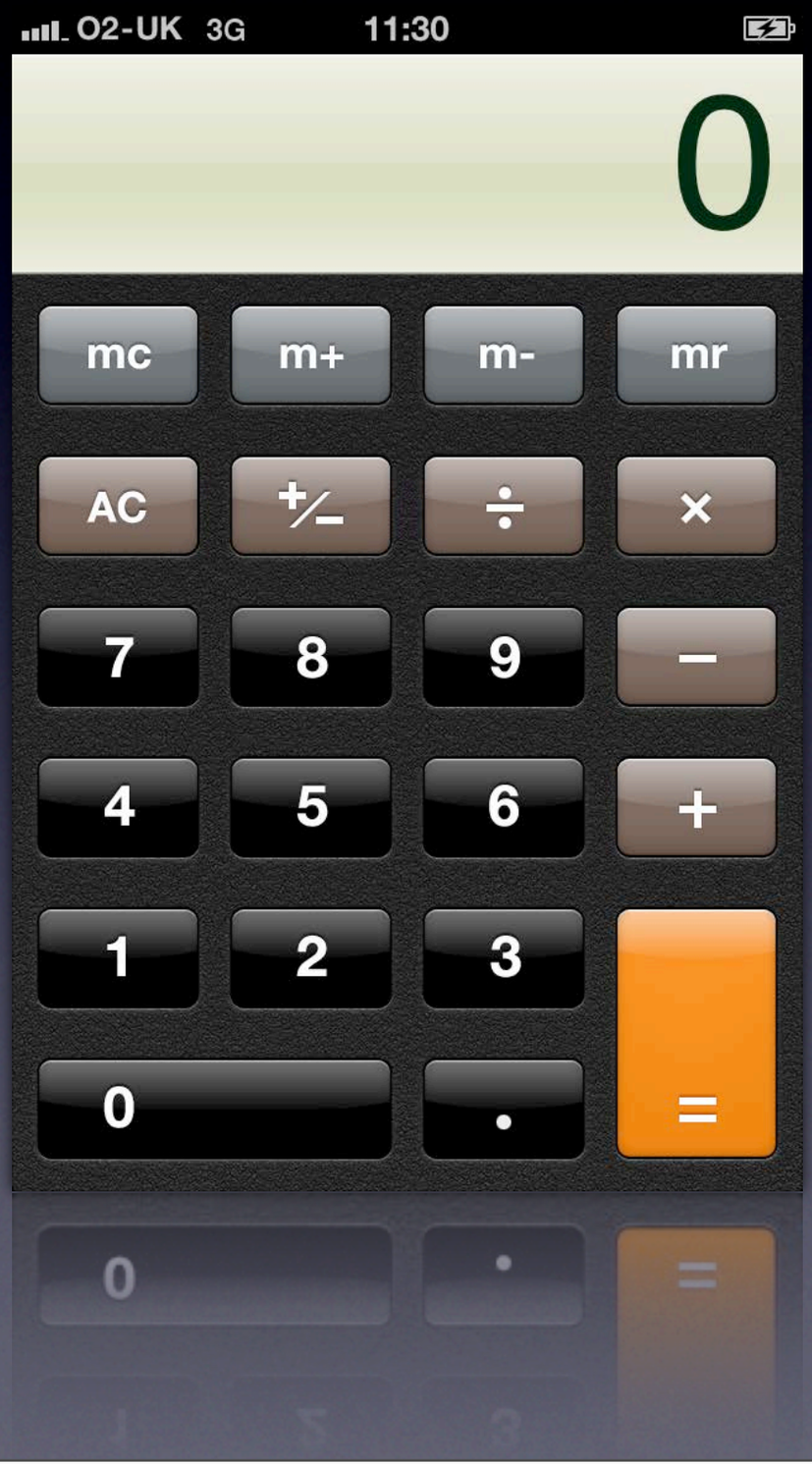
- On a phone, it is quite possible for a finger to obscure the button completely
- Flashing the button effect would effectively become invisible
- Need to find alternative approaches…

iPhone Calculator

- iPhone Calculator takes several approaches
- Some buttons have obvious effects (e.g. digit entry)
- For operators, it leaves the button highlighted

iPhone Calculator

- iPhone Calculator takes several approaches
- Some buttons have obvious effects (e.g. digit entry)
- For operators, it leaves the button highlighted



Visual Feedback

- Other options include making the buttons bigger
- Again, on the iPhone lots of buttons are the full width of the phone
- Need to think about how to give the user either implicit or explicit feedback that the touch was registered

Visual Feedback

- Other options include making the buttons bigger
- Again, on the iPhone lots of buttons are the full width of the phone
- Need to think about how to give the user either implicit or explicit feedback that the touch was registered

Visual Feedback

Visual Feedback

- Other options include making the buttons bigger
- Again, on the iPhone lots of buttons are the full width of the phone
- Need to think about how to give the user either implicit or explicit feedback that the touch was registered

In Contact

- Finger location only known when user touches
- UI paradigms, such as mouseovers, no long possible
- Instead need to create new interaction mechanisms — e.g. touch and hold without moving

New Paradigms

- Touch (and especially multitouch) provide opportunities for new UI paradigms
- Particular popular on mobiles are the use of gestures
- These are touch movements made on the device to signify an operation

Example Gestures

- Tap
- Hold
- Drag (single and multifinger)
- Pinch — to zoom
- Rotate
- Swipe

Programming for Touch

- Much like programming for mouse
- Get `TouchBegin`, `TouchMoved` and `TouchEnded` events
- There just happens to be more than one of them…
- Also, we need to parse a sequence of events into a gesture…

Look at this in detail tomorrow…

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, you'll get delivered a series of `MotionEvent`s

show how we can set up a touch listener

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 (see later)

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

pointer == touch in android speak (the first is pointer zero)
Print out getAction and getActionMasked…

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 `double`s)
- Note that Android may bundle up a series of touch events, so there is the ability to get ‘historic’ touches

historic touches are any touch events that happened between the latest and the last MotionEvent
getX()/getY() works for all events

Action Up

- A gesture ends in two 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

Single Touch events

- 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 (e.g. the position delta to move an object about)

Dragging

- 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 local to `View` origin

Make the text view move... (sort of :)

Swipes

- Can do similar for a swipe…
- Rather than moving the object, calculate the direction it is moving in
- This tells you the direction of the swipe
- Can also work out the speed

Multi Touch

- Very, very similar
- Same sequence of events as before with a few more events thrown in
- `ACTION_POINTER_DOWN` and `ACTION_POINTER_UP` tell you that a new pointer has been pressed
- Support for `256`, but some Androids only support `2`

Which pointer?

- Multitouch means `MotionEvent`s will contain multiple ‘pointers’
- `getPointerCount()` returns the number of pointers
- `getX()`/`getY()` can both take the index of a pointer in the event to return the x/y for

Will now demo this on the iPad — go show code

Which pointer?

- `getActionIndex()` tells us the index for the pointer caused this event for `ACTION_POINTER_DOWN`/`ACTION_POINTER_UP`
- However, number of pointers can change as fingers lifted or placed
- Each pointer given an `id` that won't change (the index will)

Will now demo this on the iPad — go show code

Which pointer?

- `getPointerId()` returns the `id` for a pointer from the index
- `findPointerIndex()` will get the index from an `id`
- Need to track both the `id` and past locations of pointers to move things about

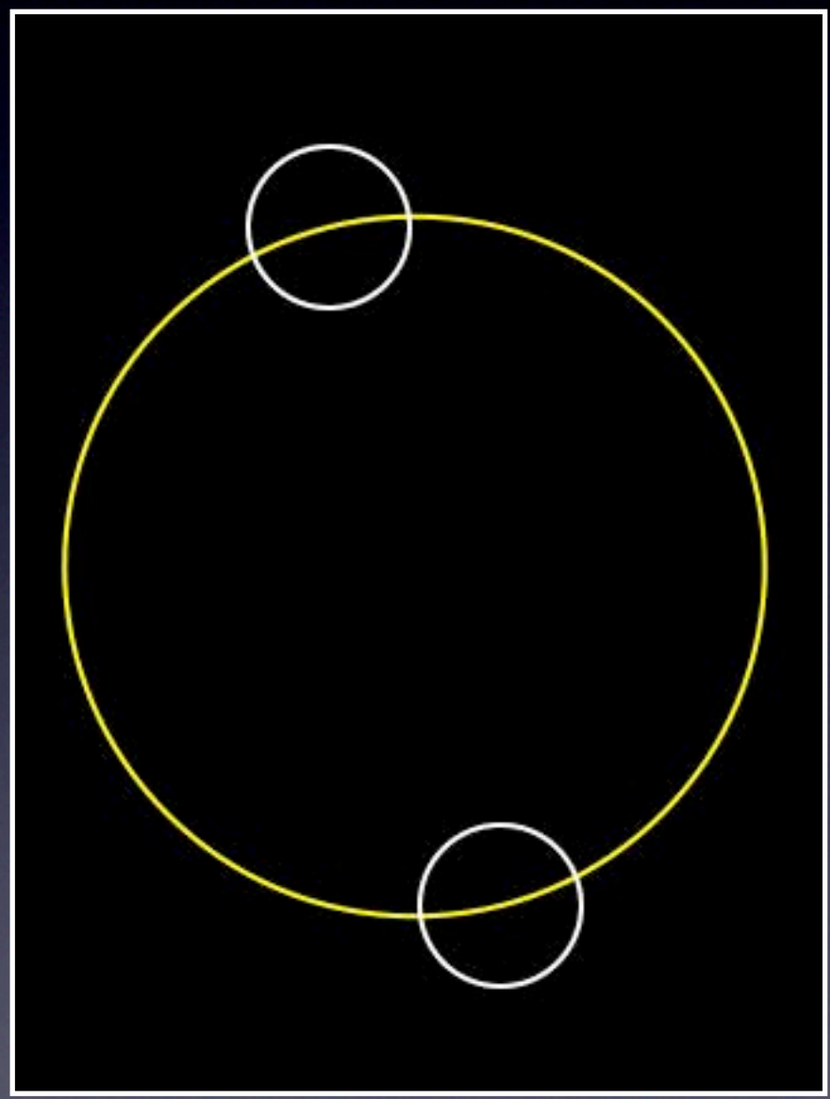
Will now demo this on the iPad — go show code

Multitouch gestures

- Ok, so can get the position of the individual touches
- How do we convert these into gestures?
- E.g. the typical pinch or rotate gestures
- Simple — just requires a little bit of maths…

Pinch to Zoom

- Imagine the two points lie on the circumference of a circle
- Can easily find the diameter of the circle — distance between the two points
- Use Pythagoras to calculate it



Pinch to Zoom

- As fingers move, diameter of the circle will change
- If you store the initial diameter of the circle
- Then the ratio of new diameter to old diameter will give the zoom ratio
- With this and the original size of the item, you can calculate the new item

Rotation

- Very similar approach for rotation
- The difference in position can (using basic trigonometry) give you the angle of the line bisecting the circle between the two points
- As the points move the angle will change
- Can use the difference between this and the original angle to work out the rotation

General Gestures

- Most gestures can be tracked in a similar fashion
- Store the initial position and see how the touch moves in relation to it

Automating Gestures

- This is a common thing you would want to do
- iOS provides a set of `GestureRecognizer` classes which can parse the touch events into a simple gesture for you
- Android has a similar base class `GestureDetector` but it doesn't seem to have many subclasses…

See Gesture Builder sample in SDK though