# G54MDP
# Mobile Device Programming

Touch

# 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

# Multi Touch

- Very similar to single touch
- Same sequence of events as before with a few more events thrown in
  - ACTION_POINTER_DOWN and ACTION_POINTER_UP tell us that a **new** pointer has been pressed
- Support for 256, but some Android devices only support 2

# Which finger?

- 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 are lifted or placed
- Each pointer given an id that won't change
  - But its **index** within a bundle of movement events might
  - Need to track both the id and past locations of pointers to move things about

# Which finger?

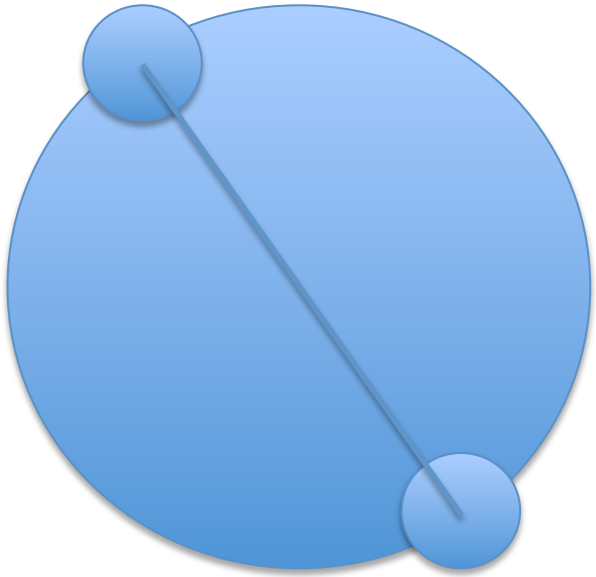
| | Action | ID |
|---|---|---|
| #1 touch → | ACTION_DOWN | 0 |
| #2 touch → | ACTION_POINTER_DOWN | 1 |
| #3 touch → | ACTION_POINTER_DOWN | 2 |
| | ACTION_MOVE | 0 |
| #2 lift → | ACTION_POINTER_UP | 1 |
| #1 lift → | ACTION_POINTER_UP | 0 |
| #3 lift → | ACTION_UP | 2 |

# Pointers to Gestures

- Maintain state of pointer IDs
  - Track movement of multiple fingers
- How do we convert these into gestures?
  - Pinch to zoom
  - Two-finger rotation
- Little SDK support for specific gestures
  - Implement ourselves with some simple trigonometry

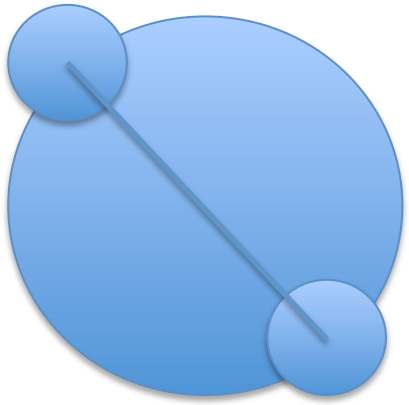
# Pinch to Zoom

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



# Pinch to Zoom

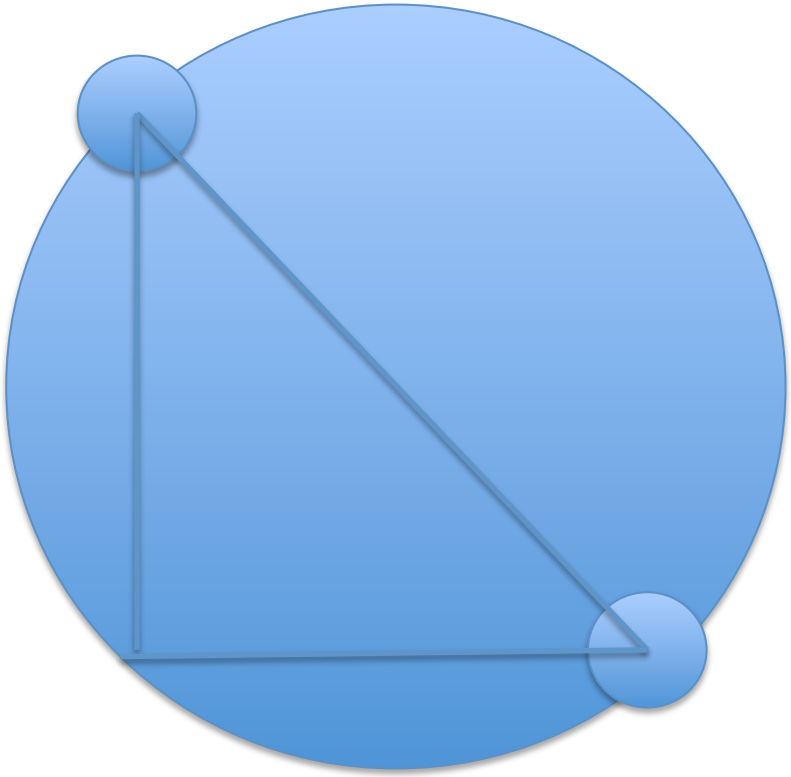
- As fingers move, diameter of the circle will change
  - store the initial diameter of the circle
- The ratio of new diameter to old diameter will give the zoom ratio
- With this and the original size of the item
  - calculate how to rescale the new item



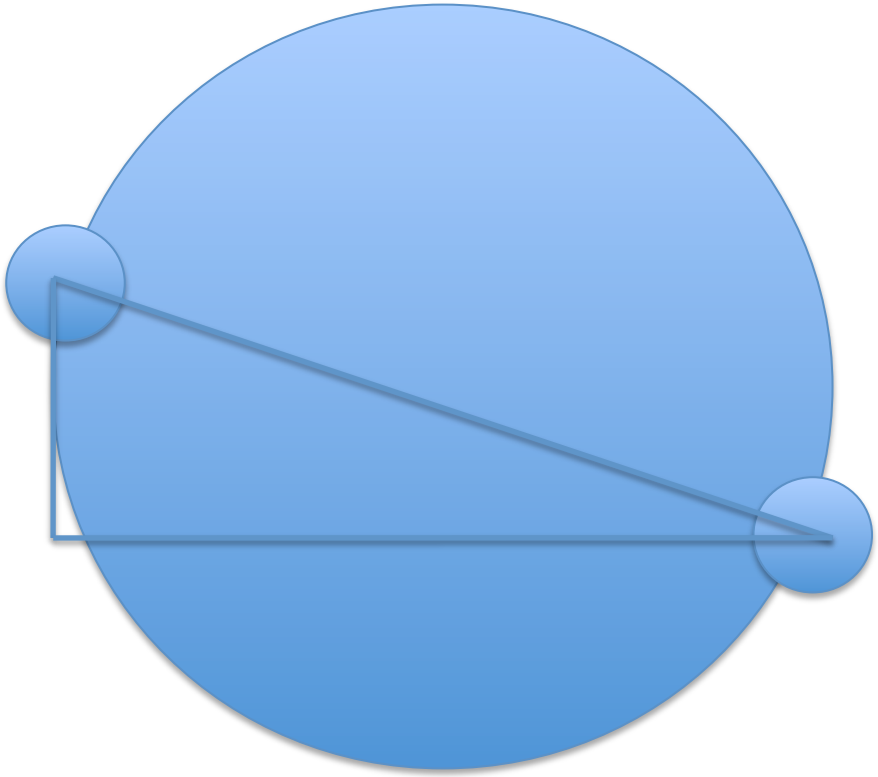
# Two-finger rotation

- Very similar approach for rotation
- The difference in position can (using basic trigonometry) give us 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
  - Magnitude may not be as important as direction
    - Rotate a photograph, document

# Two-finger rotation



# Two-finger rotation



# Android Support

- We could implement any number of complex gestures
- Android provides built-in callbacks for common gestures via GestureDetector.SimpleOnGestureListener class
  - onSingleTap
  - onDoubleTap
  - onShowPress
  - onLongPress
  - onScroll
  - onFling

# Custom Gestures

- Can create and save custom gestures as binary resources
  - A pattern of movements
- GestureLibrary attempts to recognise the gesture
  - Detects attempt at complex gesture
  - Returns predications as to which gesture the input may match
  - Inspect confidence of a match
    - Gestures are an imprecise science

# Let’s have a look…

# References

- http://developer.android.com/training/gestures/index.html
- http://developer.android.com/training/gestures/multi.html