TOUCH AND MULTITOUCH

CMPT 381

Outline

- Input basics: direct, indirect, absolute, relative
- How touch and multitouch work
- Touch and multitouch in Android
  - Historical points
  - Pointer ids
  - Touch area and orientation
- Touch and multitouch in JavaFX
  - Touch events and points
  - Gestures

Input basics

- Input Space
  - The space where pointing input occurs
  - Hardware to sense one or more 2D points
  - Not necessarily a 2D space (e.g., Trackpoint joystick)
- Display Space
  - The 2D space where the graphics of the UI appear
  - E.g., an LCD monitor
- How is input space related to display space?

Input basics

- Absolute vs Relative
  - Is input space mapped 1:1 to display space?
  - Absolute: yes – input space records *position*
    - each location in input space corresponds to only one point of display space
  - Relative: no – input space records *movement*
    - only relative movement (dX, dY) is sensed
    - allows clutching

Input basics

- Direct vs Indirect
  - Is input space overlaid on output space?
  - Indirect: no – input space is in a different physical region than display space
    - Cursor needed (where am I in display space?)
  - Direct: yes – input space and display space occupy the same physical region
    - No cursor needed (finger or stylus is the “cursor”)

Examples

	Absolute (1:1 mapping)	Relative (clutching)
Direct (input = display)		
Indirect (input ≠ display)		

Where do these go?

- Touchscreen
- Trackpad
- Mouse
- Wacom pen

Examples

	Absolute (1:1 mapping)	Relative (clutching)
Direct (input = display)	Touchscreen	?
Indirect (input ≠ display)	Wacom pen	Mouse Trackpad

Where do these go?

- Touchscreen
- Trackpad
- Mouse
- Wacom pen

Examples

	Absolute (1:1 mapping)	Relative (clutching)
Direct (input = display)	Touchscreen	Onscreen touchpad
Indirect (input ≠ display)	Wacom pen	Mouse Trackpad

Where do these go?

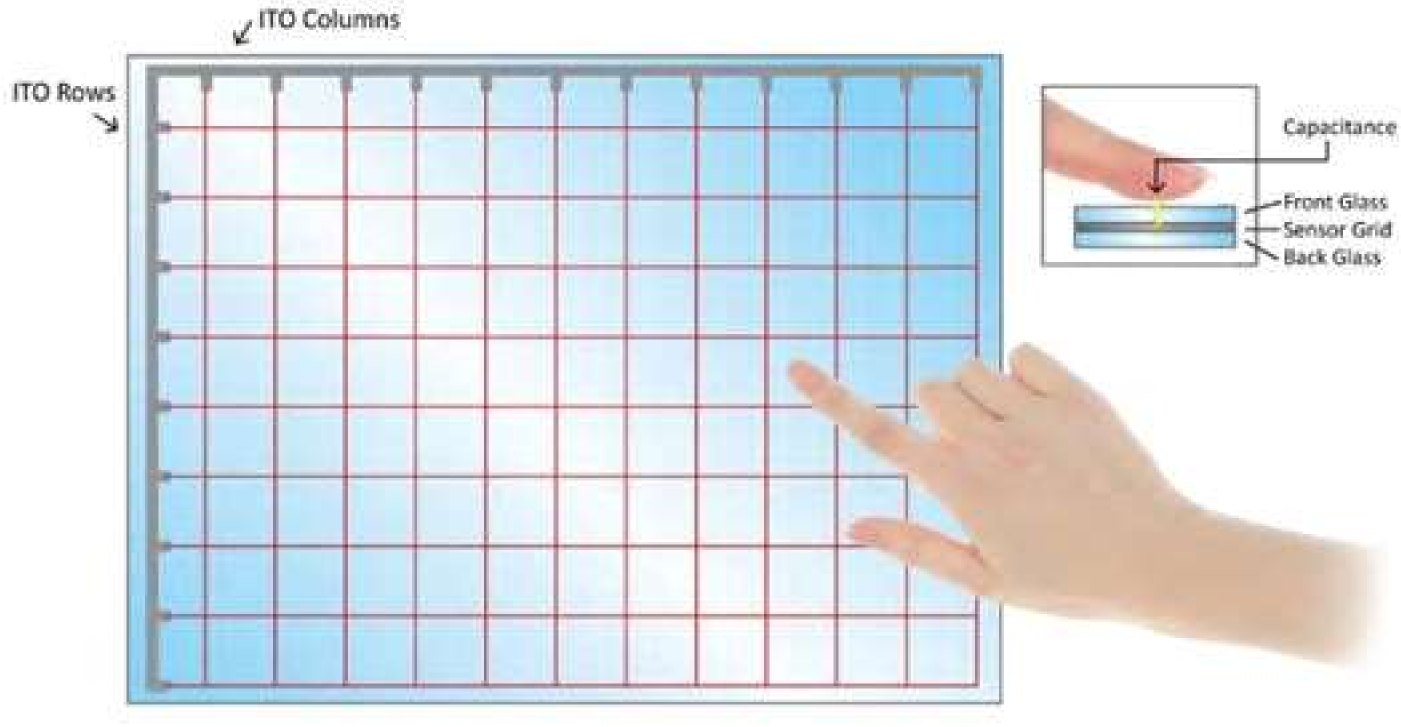
- Touchscreen
- Trackpad
- Mouse
- Wacom pen

Hardware for touch and multitouch

- Several technologies
  - Projected Capacitance
  - Resistive membrane
  - Optical beam
  - Others: diffuse illumination, Frustrated Total Internal Reflection (FTIR), depth camera, Vicon trackers…

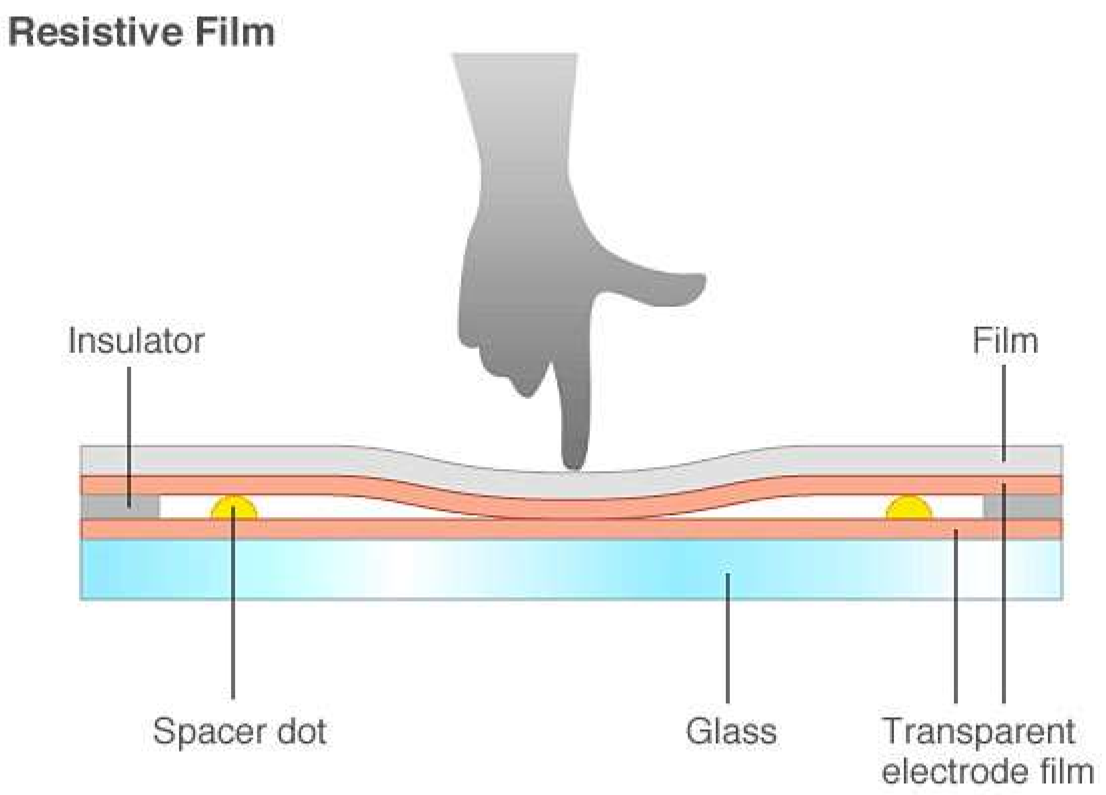
Projected Capacitive Touch

- A grid of electrodes detects anything conductive.
- Many points in the grid register, so must be filtered



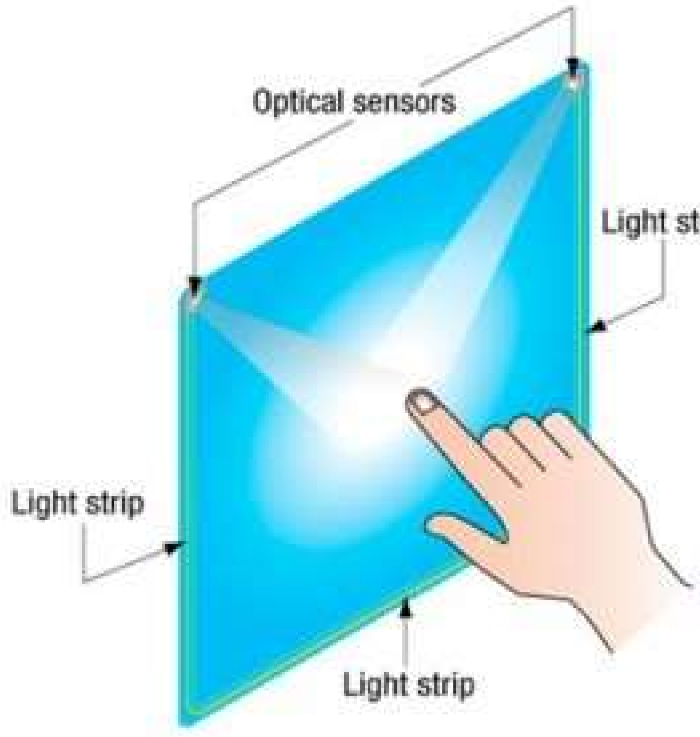
Resistive Touch

- Finger/stylus presses a membrane to touch a back layer, creating a contact

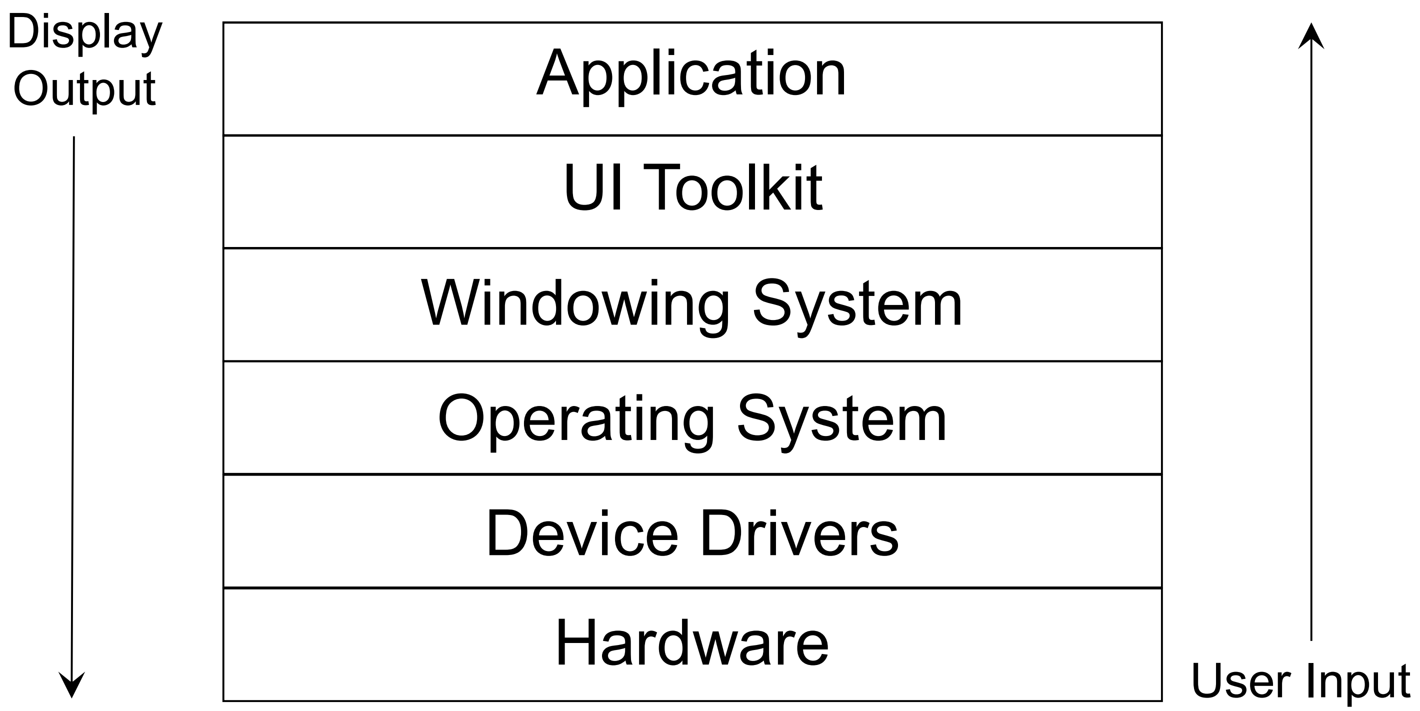


Optical Touch

- Infra-red light source around edges of screen
- IR light sensors detect breaks in the light



Touch in the layered model



Touch in the layered model

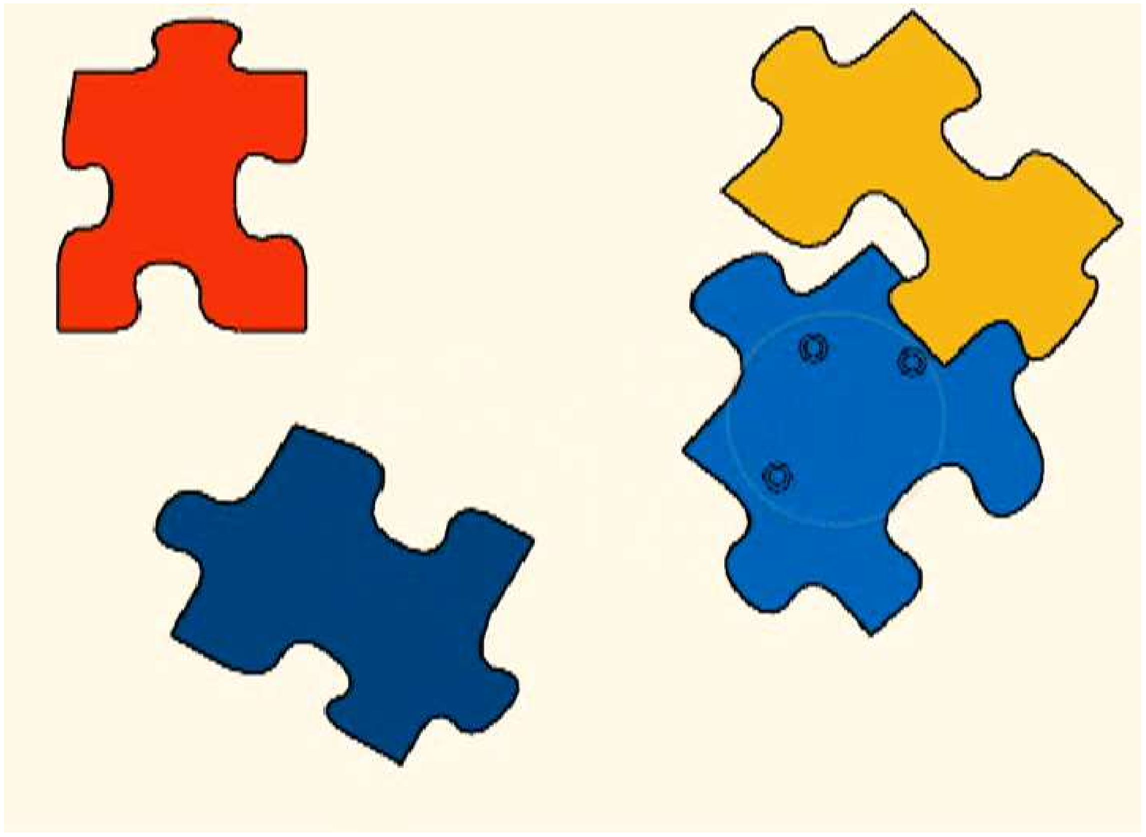
- Hardware: physical touchscreen
  - Capture raw capacitance field
- Hardware: touchscreen controller
  - Convert capacitance signal to X,Y coordinates (and possibly area and orientation)
- OS (device driver): filtered signal
  - Filter out palm touches, edge touches
  - Assign pointer IDs
  - Convert to display space coordinates

Touch in the layered model

- OS (device driver, continued):
  - Aggregate touches if sample rate > send rate
  - Convert to standard Linux event record (Android)
- OS (windowing system):
  - Which application gets the touch events?
- GUI Toolkit
  - Dispatch touch events to registered listeners

What can we do with multitouch?

- http://www.moscovich.net/tomer/papers/multi-finger-cursors.mpg

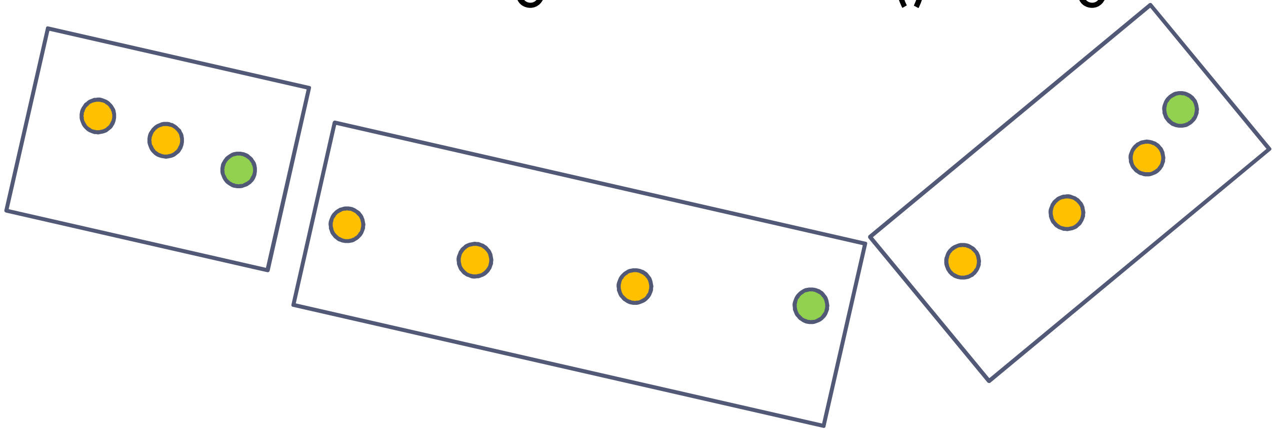


Touch and multitouch in Android

- Touch information is delivered as a MotionEvent
- We have already used this class:
  - MotionEvent.getX() and MotionEvent.getY()
- But there is much more to a MotionEvent
  - Historical points
  - Multiple touches
  - Touch area and orientation

Historical points

- Delivering touch events is expensive
- Touchscreen hardware can record touches more frequently than the device can afford to send them
  - "Sample rate is higher than send rate"
- However, easy to aggregate multiple touches
  - MotionEvent.getHistoricalX() and getHistoricalY()

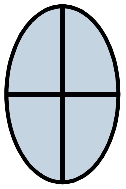


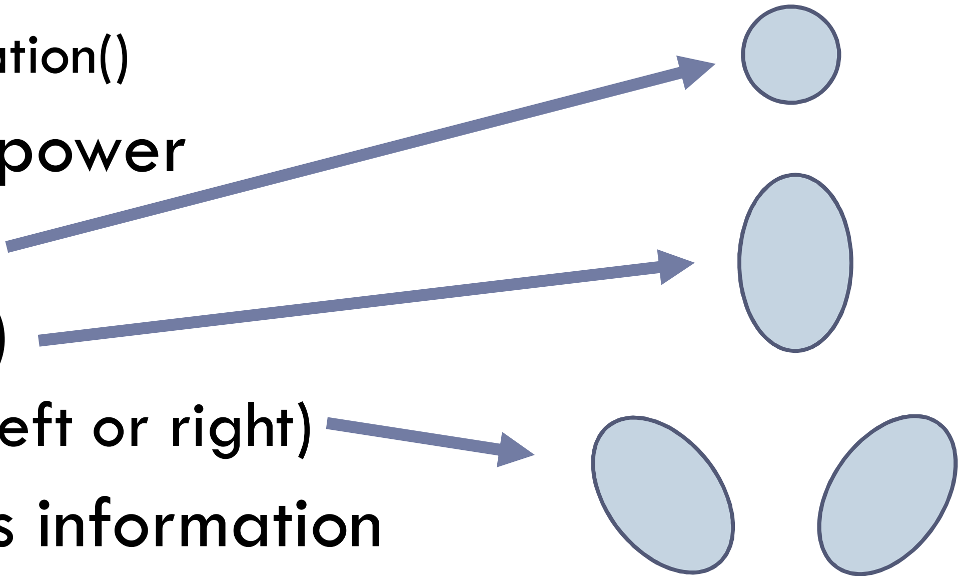
Multiple touch points

- MotionEvent.getX(int pointerIndex)
- Persistence of touch IDs
  - As long as a touch is down, maintains the same ID
  - Once action = MotionEvent.ACTION_UP, ID is discarded
- Android emulator does not support multitouch

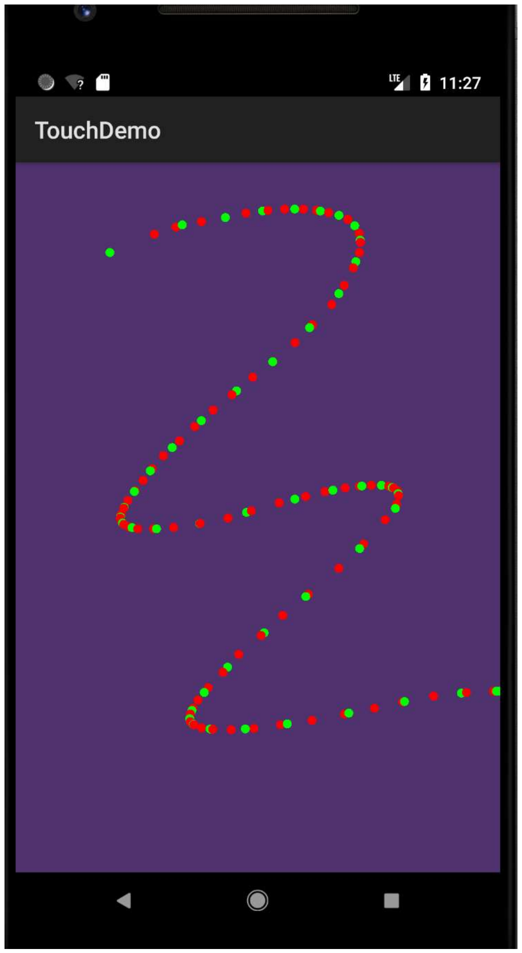
Touch area and orientation

- Characteristics of the touch, not just X,Y location:
  - Size of oval
    - MotionEvent.getTouchMajor(), .getTouchMinor()
  - Orientation of oval
    - MotionEvent.getOrientation()
- Additional expressive power
  - Fingertip (round area)
  - Finger pad (oval area)
  - Orientation (e.g., 45° left or right)
- Few devices report this information





Demo: historical points



Recognizing multitouch gestures

- Can interpret raw touch events to detect gestures like pinch-to-zoom, two-finger rotate, two-finger scroll
- However, GUI toolkits may include support for detecting typical gestures
- In android, for simple gestures (e.g., fling, long press):
  - GestureDetector class
    - developer.android.com/reference/android/view/GestureDetector.html
- For multitouch gestures (e.g., scale):
  - ScaleGestureDetector
    - developer.android.com/reference/android/view/ScaleGestureDetector.html

Touch and multitouch in JavaFX

- Touch action is delivered as a TouchEvent
  - Pressed, Moved, Released, and Stationary

- Each touch point is delivered as TouchPoint
  - TouchPoint.getX() and .getY
  - TouchPoint.getSceneX() and .getSceneY()

Multiple touch points

- To get all the touch points of an event:
  - TouchEvent.getTouchPoints() : List<TouchPoint>

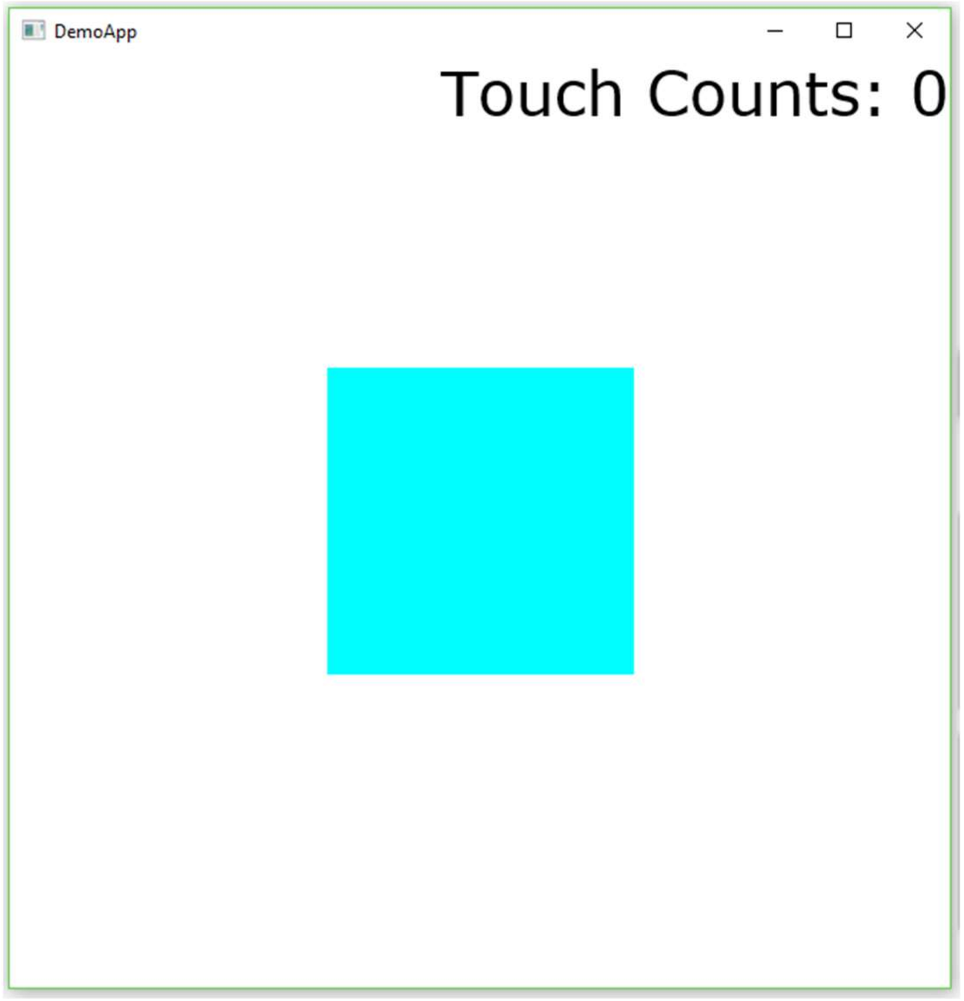
- Persistence of touch IDs
  - As long as a touch is down, maintains the same ID
  - Once action = TouchEvent.TOUCH_RELEASED, ID is discarded

Recognizing multitouch gestures

- Can detect typical gestures:
  - pinch-to-zoom, two-finger rotate, scroll, swipe
- Delivery of gestures is dependent on platform and input device

- In JavaFX, for simple gestures:
  - GestureEvent class
    - docs.oracle.com/javase/8/javafx/api/javafx/scene/input/GestureEvent.html
  - RotateEvent, ScrollEvent, SwipeEvent, ZoomEvent

Demo: JavaFX touch events & gestures



Touch Counts: 0