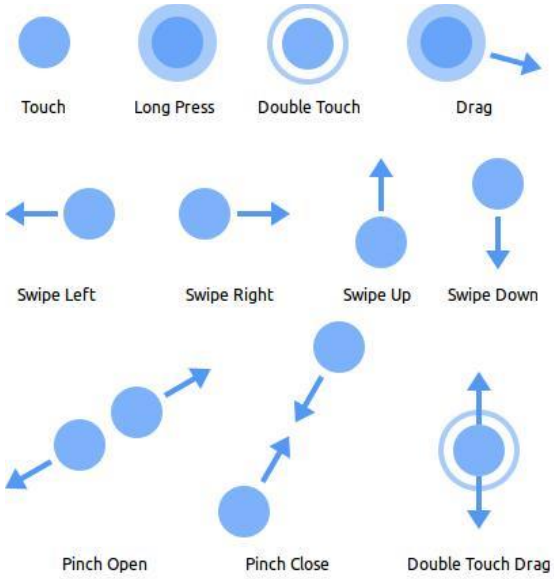
Introduction to Android Gesture Detection

A "touch gesture" occurs when a user places one or more fingers on the touch screen, and your application interprets that pattern of touches as a particular gesture. There are correspondingly two phases to gesture detection:

1. Gather data about touch events.
2. Interpret the data to see if it meets the criteria for any of the gestures your app supports.

These are the most common gestures:



When a user places one or more fingers on the screen, this triggers the callback `onTouchEvent()` on the View that received the touch events. For each sequence of touch events (position, pressure, size, addition of another finger, etc.) that is ultimately identified as a gesture, `onTouchEvent()` is fired several times.

The gesture starts when the user first touches the screen, continues as the system tracks the position of the user's finger(s), and ends by capturing the final event of the user's fingers leaving the screen.

Throughout this interaction, the `MotionEvent` delivered to `onTouchEvent()` provides the details of every interaction. Your app can use the data provided by the `MotionEvent` to determine if a gesture it cares about happened.

Android provides the `GestureDetector` class for detecting common gestures. Some of the gestures it supports include `onDown()`, `onLongPress()`, **`onFling()`,** and so on. You can use `GestureDetector` in conjunction with the `onTouchEvent()` method described above.

Capture event for a single view (an ImageView widget, for example)

You can attach an `View.OnTouchListener` object to any `View` object using the `setOnTouchListener()` method. For example:

```
findViewById<View>(R.id.my_view).setOnTouchListener { v, event ->
    // ... Respond to touch events
    true
}
```

Detect a particular gesture

If you only want to process a few gestures, you can extend `GestureDetector.SimpleOnGestureListener`

`GestureDetector.SimpleOnGestureListener` provides an implementation for all of the `on<TouchEvent>`methods by returning `false` for all of them. Thus you can override only the methods you care about.

Detect a swipe event

Swipe gesture activities vary based on context. The speed at which a gesture is performed is the primary distinction between Drag, Swipe, and Fling.

- Drag: Fine gesture, slower, more controlled, typically has an on-screen target
- Swipe: Gross gesture, faster, typically has no on-screen target
- Fling: Gross gesture, with no on-screen target

Gesture velocity impacts whether the action is immediately reversible.

- A swipe becomes a fling based on ending velocity and whether the affected element has crossed a threshold (or point past which an action can be undone).
- A drag maintains contact with an element, so reversing the direction of the gesture will drag the element back across the threshold.
- A fling moves at a faster speed and removes contact with the element while it crosses the threshold, preventing the action from being undone.

See *OnSwipeTouchListener.kt* file for the implementation.

Then, in your activity, you can attach the *OnTouchListener* to your desired view (for example, an ImageView widget) to respond to a swipe event. You can attach the listener like this:

```
widgetID.setOnTouchListener(object : OnSwipeTouchListener() {
    override fun onSwipeLeft() {
    //perform action
    }
   override fun onSwipeRight() {
   //perform action

  }
})
```