**Panoramica del tocco e dell'input**

Le pagine seguenti coprono tutto ciò che riguarda l'input dell'utente, dall'input e dai gesti tocco di base, alle tastiere e ai controller di gioco. Puoi aggiungere all'app funzionalità convenienti come copia/incolla e controllo ortografico e sviluppare servizi di testo personalizzati per offrire tastiere personalizzate (Editor metodi di input), dizionari e correttori ortografici che puoi distribuire agli utenti come applicazioni.

# Panoramica degli eventi di input

Su Android, c'è più di un modo per intercettare gli eventi dall'interazione di un utente con la tua applicazione. Quando si considerano eventi all'interno dell'interfaccia utente, l'approccio è quello di acquisire gli eventi dall'oggetto View specifico con cui l'utente interagisce. La classe View fornisce i mezzi per farlo.

All'interno delle varie classi View che userai per comporre il layout, potresti notare diversi metodi di callback pubblici che sembrano utili per gli eventi dell'interfaccia utente. Questi metodi vengono chiamati dal framework Android quando si verifica la rispettiva azione su quell'oggetto. Ad esempio, quando viene toccata una vista, ad esempio un pulsante, il metodo viene chiamato su quell'oggetto. Tuttavia, per intercettarlo, è necessario estendere la classe ed eseguire l'override del metodo. Tuttavia, estendere ogni oggetto View per gestire un evento di questo tipo non sarebbe pratico. Questo è il motivo per cui la classe View contiene anche un insieme di interfacce nidificate con callback che è possibile definire molto più facilmente. Queste interfacce, chiamate [listener di eventi](https://developer.android.com/guide/topics/ui/ui-events?hl=en#EventListeners), sono il ticket per acquisire l'interazione dell'utente con l'interfaccia utente.onTouchEvent()

Mentre si usano più comunemente i listener di eventi per ascoltare l'interazione dell'utente, potrebbe verificarsi un momento in cui si desidera estendere una classe View, per creare un componente personalizzato. Forse vuoi estendere la classe per rendere qualcosa di più elegante. In questo caso, sarà possibile definire i comportamenti di evento predefiniti per la classe utilizzando i gestori [eventi della classe](https://developer.android.com/guide/topics/ui/ui-events?hl=en#EventHandlers).[Button](https://developer.android.com/reference/android/widget/Button)

## Listener di eventi

Un listener di eventi è un'interfaccia nella classe che contiene un singolo metodo di callback. Questi metodi verranno chiamati dal framework Android quando la visualizzazione a cui è stato registrato il listener viene attivata dall'interazione dell'utente con l'elemento nell'interfaccia utente.[View](https://developer.android.com/reference/android/view/View)

Nelle interfacce listener di eventi sono inclusi i seguenti metodi di callback:

onClick()

Da. Questo viene chiamato quando l'utente tocca l'elemento (quando è in modalità touch), o si concentra sull'elemento con i tasti di navigazione o trackball e preme il tasto "invio" adatto o preme verso il basso sulla trackball.[View.OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener)

onLongClick()

Da. Questo viene chiamato quando l'utente tocca e tiene premuto l'elemento (quando è in modalità touch), o si concentra sull'elemento con i tasti di navigazione o trackball e preme e tiene premuto il tasto "invio" adatto o preme e tiene premuto il trackball (per un secondo).[View.OnLongClickListener](https://developer.android.com/reference/android/view/View.OnLongClickListener)

onFocusChange()

Da. Questo viene chiamato quando l'utente si sposta su o lontano dall'elemento, utilizzando i tasti di navigazione o trackball.[View.OnFocusChangeListener](https://developer.android.com/reference/android/view/View.OnFocusChangeListener)

onKey()

Da. Questo viene chiamato quando l'utente è concentrato sull'elemento e preme o rilascia un tasto hardware sul dispositivo.[View.OnKeyListener](https://developer.android.com/reference/android/view/View.OnKeyListener)

onTouch()

Da. Questo viene chiamato quando l'utente esegue un'azione qualificata come evento tocco, tra cui una pressione, un rilascio o qualsiasi gesto di movimento sullo schermo (entro i limiti dell'elemento).[View.OnTouchListener](https://developer.android.com/reference/android/view/View.OnTouchListener)

onCreateContextMenu()

Da. Questo viene chiamato quando viene creato un menu di scelta rapida (come risultato di un "clic lungo" sostenuto. Vedi la discussione sui menu di scelta rapida nella [guida per gli sviluppatori](https://developer.android.com/guide/topics/ui/menus#context-menu) menu.[View.OnCreateContextMenuListener](https://developer.android.com/reference/android/view/View.OnCreateContextMenuListener)

Questi metodi sono gli unici abitanti della loro rispettiva interfaccia. Per definire uno di questi metodi e gestire gli eventi, implementare l'interfaccia nidificata nell'attività o definirla come classe anonima. Quindi, passare un'istanza dell'implementazione al rispettivo metodo. (ad esempio, chiamare e passarlo l'implementazione del file .)View.set...Listener()[setOnClickListener()](https://developer.android.com/reference/android/view/View#setOnClickListener(android.view.View.OnClickListener))[OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener)

Nell'esempio seguente viene illustrato come registrare un listener on-click per un pulsante.

[KOTLIN](https://developer.android.com/guide/topics/ui/ui-events?hl=en#kotlin)

protected void onCreate(savedValues: Bundle) {  
    ...  
    val button: Button = findViewById(R.id.corky)  
    // Register the onClick listener with the implementation above  
    button.setOnClickListener { view ->  
        // do something when the button is clicked  
    }  
    ...  
}

È inoltre possibile implementare OnClickListener come parte dell'attività. In questo modo si eviterà il carico aggiuntivo della classe e l'allocazione degli oggetti. Per esempio:

[KOTLIN](https://developer.android.com/guide/topics/ui/ui-events?hl=en#kotlin)

class ExampleActivity : Activity(), OnClickListener {  
    
    protected fun onCreate(savedValues: Bundle) {  
        val button: Button = findViewById(R.id.corky)  
        button.setOnClickListener(this)  
    }  
  
    // Implement the OnClickListener callback  
    fun onClick(v: View) {  
        // do something when the button is clicked  
    }  
}

Si noti che il callback nell'esempio precedente non ha alcun valore restituito, ma alcuni altri metodi listener di eventi devono restituire un valore booleano. Il motivo dipende dall'evento. Per i pochi che lo fanno, ecco perché:onClick()

* [onLongClick()](https://developer.android.com/reference/android/view/View.OnLongClickListener#onLongClick(android.view.View)) - Restituisce un valore booleano per indicare se l'evento è stato utilizzato e non deve essere portato avanti. Cioè, restituisci true per indicare che hai gestito l'evento e dovrebbe fermarsi qui; restituisce false se non è stato gestito e/o l'evento deve continuare a qualsiasi altro listener con clic.
* [onKey()](https://developer.android.com/reference/android/view/View.OnKeyListener#onKey(android.view.View,%20int,%20android.view.KeyEvent)) - Restituisce un valore booleano per indicare se l'evento è stato utilizzato e non deve essere portato avanti. Cioè, restituisci true per indicare che hai gestito l'evento e dovrebbe fermarsi qui; restituisce false se non è stato gestito e/o l'evento deve continuare a qualsiasi altro listener on-key.
* [onTouch()](https://developer.android.com/reference/android/view/View.OnTouchListener#onTouch(android.view.View,%20android.view.MotionEvent)) - Restituisce un valore booleano per indicare se il listener utilizza questo evento. L'importante è che questo evento possa avere più azioni che si susguano. Pertanto, se si restituisce false quando viene ricevuto l'evento azione down, si indica che non è stato utilizzato l'evento e non si è interessati alle azioni successive di questo evento. Pertanto, non ti verranno chiamate altre azioni all'interno dell'evento, come un gesto con le dita o l'eventuale evento di azione up.

Tenere presente che gli eventi chiave hardware vengono sempre recapitati alla visualizzazione attualmente a fuoco. Vengono inviati a partire dall'inizio della gerarchia di visualizzazione e quindi verso il basso, fino a raggiungere la destinazione appropriata. Se la visualizzazione (o un figlio della visualizzazione) ha attualmente lo stato attivo, è possibile visualizzare il percorso dell'evento attraverso il metodo. In alternativa all'acquisizione di eventi chiave tramite la visualizzazione, è anche possibile ricevere tutti gli eventi all'interno dell'attività con e .[dispatchKeyEvent()](https://developer.android.com/reference/android/view/View#dispatchKeyEvent(android.view.KeyEvent))[onKeyDown()](https://developer.android.com/reference/android/app/Activity#onKeyDown(int,%20android.view.KeyEvent))[onKeyUp()](https://developer.android.com/reference/android/app/Activity#onKeyUp(int,%20android.view.KeyEvent))

Inoltre, quando si pensa all'input di testo per l'applicazione, tenere conto del fatto che molti dispositivi hanno solo metodi di input software. Tali metodi non devono essere basati su chiavi; alcuni possono usare l'input vocale, la grafia e così via. Anche se un metodo di input presenta un'interfaccia simile a una tastiera, in genere **non** attiverà la famiglia di eventi. Non creare mai un'interfaccia utente che richiede il controllo di pressioni specifiche dei tasti, a meno che non si desideri limitare l'applicazione ai dispositivi con una tastiera hardware. In particolare, non fare affidamento su questi metodi per convalidare l'input quando l'utente preme il tasto restituito; invece, usa azioni come segnalare al metodo di input come l'applicazione prevede di reagire, in modo che possa cambiare la sua interfaccia utente in modo significativo. Evita i presupposti su come dovrebbe funzionare un metodo di input software e fidati di esso per fornire testo già formattato all'applicazione.[onKeyDown()](https://developer.android.com/reference/android/app/Activity#onKeyDown(int,%20android.view.KeyEvent))[IME\_ACTION\_DONE](https://developer.android.com/reference/android/view/inputmethod/EditorInfo#IME_ACTION_DONE)

**Nota:** Android chiamerà prima i gestori eventi e quindi i gestori predefiniti appropriati dalla definizione della classe in secondo luogo. Di conseguenza, la restituzione di true da questi listener di eventi impedirrà la propagazione dell'evento ad altri listener di eventi e bloccherà anche il callback al gestore eventi predefinito nella visualizzazione. Quindi assicurati di voler terminare l'evento quando restituisci true.

## Gestori eventi

Se stai costruendo un componente personalizzato da View, sarai in grado di definire diversi metodi di callback usati come gestori eventi predefiniti. Nel documento sui componenti [di visualizzazione personalizzati](https://developer.android.com/guide/topics/ui/custom-components)vengono vengono utilizzati alcuni dei callback comuni utilizzati per la gestione degli eventi, tra cui:

* [onKeyDown(int, KeyEvent)](https://developer.android.com/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent)) - Chiamato quando si verifica un nuovo evento chiave.
* [onKeyUp(int, KeyEvent)](https://developer.android.com/reference/android/view/View#onKeyUp(int,%20android.view.KeyEvent)) - Chiamato quando si verifica un evento key up.
* [onTrackballEvent(MotionEvent)](https://developer.android.com/reference/android/view/View#onTrackballEvent(android.view.MotionEvent)) - Chiamato quando si verifica un evento di movimento trackball.
* [onTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent)) - Chiamato quando si verifica un evento di movimento touch screen.
* [onFocusChanged(boolean, int, Rect)](https://developer.android.com/reference/android/view/View#onFocusChanged(boolean,%20int,%20android.graphics.Rect)) - Chiamato quando la vista guadagna o perde lo stato attivo.

Esistono altri metodi di cui dovresti essere a conoscenza, che non fanno parte della classe View, ma possono influire direttamente sul modo in cui sei in grado di gestire gli eventi. Quindi, quando gestisci eventi più complessi all'interno di un layout, considera questi altri metodi:

* [Activity.dispatchTouchEvent(MotionEvent)](https://developer.android.com/reference/android/app/Activity#dispatchTouchEvent(android.view.MotionEvent)) - Ciò consente all'utente di intercettare tutti gli eventi di tocco prima che siano inviati alla finestra.[Activity](https://developer.android.com/reference/android/app/Activity)
* [ViewGroup.onInterceptTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent)) - Ciò consente a un oggetto di guardare gli eventi mentre vengono inviati alle visualizzazioni figlio.[ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)
* [ViewParent.requestDisallowInterceptTouchEvent(boolean)](https://developer.android.com/reference/android/view/ViewParent#requestDisallowInterceptTouchEvent(boolean)) - Chiamare questo su una vista padre per indicare che non deve intercettare gli eventi di tocco con .[onInterceptTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))

## Modalità tocco

Quando un utente naviga in un'interfaccia utente con tasti direzionali o trackball, è necessario mettere a fuoco elementi utilizzabili (come i pulsanti) in modo che l'utente possa vedere cosa accetterà l'input. Se il dispositivo ha funzionalità di tocco, tuttavia, e l'utente inizia a interagire con l'interfaccia toccandolo, non è più necessario evidenziare gli elementi o mettere a fuoco una particolare vista. Pertanto, esiste una modalità per l'interazione denominata "modalità tocco".

Per un dispositivo compatibile con il tocco, una volta che l'utente tocca lo schermo, il dispositivo entrerà in modalità touch. Da questo punto in poi, solo le viste per le quali è vero saranno messa a fuoco, come i widget di modifica del testo. Altre visualizzazioni toccabili, come i pulsanti, non prenderanno fuoco quando vengono toccate; semplicemente licenzieranno i loro ascoltatori on-click quando premuti. [isFocusableInTouchMode()](https://developer.android.com/reference/android/view/View#isFocusableInTouchMode())

Ogni volta che un utente colpisce un tasto direzionale o scorre con una trackball, il dispositivo uscirà dalla modalità touch e troverà una vista da mettere a fuoco. Ora, l'utente può riprendere a interagire con l'interfaccia utente senza toccare lo schermo.

Lo stato della modalità touch viene mantenuto in tutto il sistema (tutte le finestre e le attività). Per eseguire una query sullo stato corrente, è possibile chiamare per verificare se il dispositivo è attualmente in modalità tocco. [isInTouchMode()](https://developer.android.com/reference/android/view/View#isInTouchMode())

## Messa a fuoco sulla gestione

Il framework gestirà il movimento di messa a fuoco di routine in risposta all'input dell'utente. Ciò include la modifica dello stato attivo man mano che le viste vengono rimosse o nascoste o man mano che diventano disponibili nuove viste. Le visualizzazioni indicano la loro volontà di concentrarsi attraverso il metodo. Per modificare la possibilità che una visualizzazione prenda lo stato attivo, chiamare . Quando si è in modalità tocco, è possibile verificare se una vista consente lo stato attivo con . È possibile modificare questa situazione con . [isFocusable()](https://developer.android.com/reference/android/view/View#isFocusable())[setFocusable()](https://developer.android.com/reference/android/view/View#setFocusable(boolean))[isFocusableInTouchMode()](https://developer.android.com/reference/android/view/View#isFocusableInTouchMode())[setFocusableInTouchMode()](https://developer.android.com/reference/android/view/View#setFocusableInTouchMode(boolean))

Nei dispositivi che eseguono Android 9 (livello API 28) o superiore, le attività non assegnano uno stato attivo iniziale. Al contrario, è necessario richiedere esplicitamente lo stato attivo iniziale, se lo si desidera.

Il movimento di messa a fuoco si basa su un algoritmo che trova il vicino più vicino in una determinata direzione. In rari casi, l'algoritmo predefinito potrebbe non corrispondere al comportamento previsto dello sviluppatore. In queste situazioni, è possibile fornire override espliciti con i seguenti attributi XML nel file di layout: , , e . Aggiungere uno di questi attributi alla vista da cui viene lasciato lo stato attivo. Definire il valore dell'attributo come ID della vista a cui deve essere assegnato lo stato attivo. Per esempio:**nextFocusDownnextFocusLeftnextFocusRightnextFocusUp**

<LinearLayout  
    android:orientation="vertical"  
    ... >  
  <Button android:id="@+id/top"  
          android:nextFocusUp="@+id/bottom"  
          ... />  
  <Button android:id="@+id/bottom"  
          android:nextFocusDown="@+id/top"  
          ... />  
</LinearLayout>

Normalmente, in questo layout verticale, navigare verso l'alto dal primo pulsante non andrebbe da nessuna parte, né sarebbe navigando verso il basso dal secondo pulsante. Ora che il pulsante superiore ha definito quello inferiore come (e viceversa), lo stato attivo per la navigazione andrà in ciclo dall'alto verso il basso e dal basso verso l'alto.**nextFocusUp**

Se vuoi dichiarare una visualizzazione come attivo nell'interfaccia utente (quando tradizionalmente non lo è), aggiungi l'attributo XML alla visualizzazione nella dichiarazione di layout. Impostare il valore . È inoltre possibile dichiarare una vista come messa a fuoco mentre si è in modalità tocco con .android:focusable**true**android:focusableInTouchMode

Per richiedere a una vista specifica di prendere lo stato attivo, chiamare .[requestFocus()](https://developer.android.com/reference/android/view/View" \l "requestFocus())

Per ascoltare gli eventi dello stato attivo (ricevere una notifica quando una vista riceve o perde lo stato attivo), utilizzare , come discusso nella [sezione Listener di](https://developer.android.com/guide/topics/ui/ui-events?hl=en#EventListeners) eventi.[onFocusChange()](https://developer.android.com/reference/android/view/View.OnFocusChangeListener#onFocusChange(android.view.View,%20boolean))

# Usare i gesti tocco

Questa guida descrive come scrivere app che consentano agli utenti di interagire con un'app tramite gesti tocco. Android fornisce una varietà di API per aiutarti a creare e rilevare i gesti.

Anche se la tua app non deve dipendere dai gesti tocco per i comportamenti di base (poiché i gesti potrebbero non essere disponibili per tutti gli utenti in tutti i contesti), l'aggiunta di interazioni basate sul tocco all'app può aumentarne notevolmente l'utilità e l'appeal.

Per offrire agli utenti un'esperienza coerente e intuitiva, la tua app dovrebbe seguire le convenzioni Android accettate per i gesti tocco. La [guida alla progettazione dei](https://material.io/design/interaction/gestures.html) gesti mostra come usare i gesti comuni nelle app Android. Vedere anche la guida Material Design per il [movimento del materiale.](https://material.io/guidelines/motion/material-motion.html)

Per ulteriori informazioni su questo argomento, leggere le guide correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events)
* [sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la visualizzazione](https://developer.android.com/training/custom-views/making-interactive)

## Lezioni

[**Rilevare gesti comuni**](https://developer.android.com/training/gestures/detector)

Informazioni su come rilevare i movimenti tocco di base, ad esempio lo scorrimento, l'avventura e il doppio tocco, utilizzando . [GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)

# Rilevare gesti comuni

Un "gesto tocco" si verifica quando un utente posiziona una o più dita sul touchscreen e l'applicazione interpreta tale motivo di tocco come un particolare gesto. Di conseguenza, il rilevamento dei gesti è in due fasi:

1. Raccogliere dati sugli eventi di tocco.
2. Interpreta i dati per vedere se soddisfano i criteri per uno qualsiasi dei gesti supportati dall'app.

Fare riferimento alle risorse correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events.html) Guida api
* [Panoramica dei sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la vista](https://developer.android.com/training/custom-views/making-interactive)

#### Supportare le classi della libreria

Gli esempi di questa lezione utilizzano le classi e . Queste classi si trovano nella [Libreria di supporto](https://developer.android.com/tools/support-library). È necessario utilizzare le classi della libreria di supporto, ove possibile, per fornire compatibilità con i dispositivi che eseguono Android 1.6 e versioni precedenti. Si noti che non è una sostituzione per la classe. Piuttosto, fornisce metodi di utilità statici a cui passi l'oggetto per ricevere l'azione desiderata associata a quell'evento.[GestureDetectorCompat](https://developer.android.com/reference/androidx/core/view/GestureDetectorCompat)[MotionEventCompatMotionEventCompat](https://developer.android.com/reference/androidx/core/view/MotionEventCompat)[MotionEventMotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

## Raccogliere dati

Quando un utente posiziona una o più dita sullo schermo, attiva il callback sulla visualizzazione che ha ricevuto gli eventi tocco. Per ogni sequenza di eventi tocco (posizione, pressione, dimensioni, aggiunta di un altro dito, ecc.) che viene infine identificata come un gesto, viene sparato più volte.[onTouchEvent()onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Il gesto inizia quando l'utente tocca per la prima volta lo schermo, continua mentre il sistema tiene traccia della posizione delle dita dell'utente e termina catturando l'evento finale delle dita dell'utente che e lasciano lo schermo. Durante questa interazione, il consegnato fornisce i dettagli di ogni interazione. L'app può usare i dati forniti dall'oggetto per determinare se si è verificato un gesto a cui si preoccupa.[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

### Acquisire eventi tocco per un'attività o una visualizzazione

Per intercettare gli eventi di tocco in un'attività o in una visualizzazione, eseguire l'override del callback.[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Il frammento seguente viene utilizzato per estrarre l'azione eseguita dall'utente dal parametro . In questo modo vengono dati non elaborati necessari per determinare se si è verificato un gesto a cui si tiene:[getActionMasked()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#getActionMasked(android.view.MotionEvent))event

[KOTLIN](https://developer.android.com/training/gestures/detector#kotlin)

class MainActivity : Activity() {  
    ...  
    // This example shows an Activity, but you would use the same approach if  
    // you were subclassing a View.  
    override fun onTouchEvent(event: MotionEvent): Boolean {  
  
        val action: Int = MotionEventCompat.getActionMasked(event)  
  
        return when (action) {  
            MotionEvent.ACTION\_DOWN -> {  
                Log.d(DEBUG\_TAG, "Action was DOWN")  
                true  
            }  
            MotionEvent.ACTION\_MOVE -> {  
                Log.d(DEBUG\_TAG, "Action was MOVE")  
                true  
            }  
            MotionEvent.ACTION\_UP -> {  
                Log.d(DEBUG\_TAG, "Action was UP")  
                true  
            }  
            MotionEvent.ACTION\_CANCEL -> {  
                Log.d(DEBUG\_TAG, "Action was CANCEL")  
                true  
            }  
            MotionEvent.ACTION\_OUTSIDE -> {  
                Log.d(DEBUG\_TAG, "Movement occurred outside bounds of current screen element")  
                true  
            }  
            else -> super.onTouchEvent(event)  
        }  
    }  
}

È quindi possibile eseguire la propria elaborazione su questi eventi per determinare se si è verificato un gesto. Questo è il tipo di elaborazione che dovresti fare per un gesto personalizzato. Tuttavia, se l'app usa gesti comuni come doppio tocco, pressione lunga, lancio e così via, puoi sfruttare la classe. ti consente di rilevare facilmente i gesti comuni senza elaborare tu stesso i singoli eventi di tocco. Questo argomento è discusso di seguito in [Rileva gesti](https://developer.android.com/training/gestures/detector#detect).[GestureDetectorGestureDetector](https://developer.android.com/reference/android/view/GestureDetector)

### Acquisire eventi tocco per una singola visualizzazione

In alternativa a , è possibile associare un oggetto a qualsiasi oggetto utilizzando il metodo . In questo modo è possibile ascoltare eventi di tocco senza sottoclassare un oggetto esistente. Per esempio:[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))[View.OnTouchListener](https://developer.android.com/reference/android/view/View.OnTouchListener)[View](https://developer.android.com/reference/android/view/View)[setOnTouchListener()](https://developer.android.com/reference/android/view/View#setOnTouchListener(android.view.View.OnTouchListener))[View](https://developer.android.com/reference/android/view/View)

[KOTLIN](https://developer.android.com/training/gestures/detector#kotlin)

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

Attenzione alla creazione di un listener che restituisce l'evento. In questo caso, il listener non verrà chiamato per il successivo e la stringa di eventi. Questo perché è il punto di partenza per tutti gli eventi di tocco.false[ACTION\_DOWN](https://developer.android.com/reference/android/view/MotionEvent#ACTION_DOWN)[ACTION\_MOVE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)[ACTION\_UP](https://developer.android.com/reference/android/view/MotionEvent#ACTION_UP)[ACTION\_DOWN](https://developer.android.com/reference/android/view/MotionEvent#ACTION_DOWN)

Se si sta creando una visualizzazione personalizzata, è possibile eseguire l'override di , come descritto in precedenza.[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

## Rilevare i movimenti

Android fornisce la classe per rilevare gesti comuni. Alcuni dei gesti che supporta includono , , e così via. È possibile utilizzare insieme al metodo descritto sopra.[GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)[onDown()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onDown(android.view.MotionEvent))[onLongPress()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onLongPress(android.view.MotionEvent))[onFling()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onFling(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

### Rilevare tutti i gesti supportati

Quando si crea un'istanza di un oggetto, uno dei parametri necessari è una classe che implementa l'interfaccia . notifica agli utenti quando si è verificato un particolare evento di tocco. Per consentire all'oggetto di ricevere eventi, eseguire l'override del metodo View o Activity e passare tutti gli eventi osservati all'istanza del rilevatore.[GestureDetectorCompat](https://developer.android.com/reference/androidx/core/view/GestureDetectorCompat)[GestureDetector.OnGestureListenerGestureDetector.OnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener)[GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Nel frammento seguente, un valore restituito di dai singoli metodi indica che hai gestito l'evento tocco. Un valore restituito di passa gli eventi giù attraverso lo stack di visualizzazione fino a quando il tocco non è stato gestito correttamente.trueon<TouchEvent>false

Esegui lo snippet seguente per avere un'mente su come vengono attivate le azioni quando interagisci con il touchscreen e quali sono i contenuti di per ogni evento tocco. Ti renderai conto della quantità di dati generati anche per interazioni semplici.[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

[KOTLIN](https://developer.android.com/training/gestures/detector#kotlin)

private const val DEBUG\_TAG = "Gestures"  
  
class MainActivity :  
        Activity(),  
        GestureDetector.OnGestureListener,  
        GestureDetector.OnDoubleTapListener {  
  
    private lateinit var mDetector: GestureDetectorCompat  
  
    // Called when the activity is first created.  
    public override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity\_main)  
        // Instantiate the gesture detector with the  
        // application context and an implementation of  
        // GestureDetector.OnGestureListener  
        mDetector = GestureDetectorCompat(this, this)  
        // Set the gesture detector as the double tap  
        // listener.  
        mDetector.setOnDoubleTapListener(this)  
    }  
  
    override fun onTouchEvent(event: MotionEvent): Boolean {  
        return if (mDetector.onTouchEvent(event)) {  
            true  
        } else {  
            super.onTouchEvent(event)  
        }  
    }  
  
    override fun onDown(event: MotionEvent): Boolean {  
        Log.d(DEBUG\_TAG, "onDown: $event")  
        return true  
    }  
  
    override fun onFling(  
            event1: MotionEvent,  
            event2: MotionEvent,  
            velocityX: Float,  
            velocityY: Float  
    ): Boolean {  
        Log.d(DEBUG\_TAG, "onFling: $event1 $event2")  
        return true  
    }  
  
    override fun onLongPress(event: MotionEvent) {  
        Log.d(DEBUG\_TAG, "onLongPress: $event")  
    }  
  
    override fun onScroll(  
            event1: MotionEvent,  
            event2: MotionEvent,  
            distanceX: Float,  
            distanceY: Float  
    ): Boolean {  
        Log.d(DEBUG\_TAG, "onScroll: $event1 $event2")  
        return true  
    }  
  
    override fun onShowPress(event: MotionEvent) {  
        Log.d(DEBUG\_TAG, "onShowPress: $event")  
    }  
  
    override fun onSingleTapUp(event: MotionEvent): Boolean {  
        Log.d(DEBUG\_TAG, "onSingleTapUp: $event")  
        return true  
    }  
  
    override fun onDoubleTap(event: MotionEvent): Boolean {  
        Log.d(DEBUG\_TAG, "onDoubleTap: $event")  
        return true  
    }  
  
    override fun onDoubleTapEvent(event: MotionEvent): Boolean {  
        Log.d(DEBUG\_TAG, "onDoubleTapEvent: $event")  
        return true  
    }  
  
    override fun onSingleTapConfirmed(event: MotionEvent): Boolean {  
        Log.d(DEBUG\_TAG, "onSingleTapConfirmed: $event")  
        return true  
    }  
  
}

### Rilevare un sottoinsieme di gesti supportati

Se si desidera elaborare solo alcuni gesti, è possibile estendere anziché implementare l'interfaccia. [GestureDetector.SimpleOnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener)[GestureDetector.OnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener)

[GestureDetector.SimpleOnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener) fornisce un'implementazione per tutti i metodi restituendo per tutti loro. Quindi puoi ignorare solo i metodi a cui tenci. Ad esempio, il frammento riportato di seguito crea una classe che estende ed esegue l'override di e .on<TouchEvent>false[GestureDetector.SimpleOnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener)[onFling()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onFling(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[onDown()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onDown(android.view.MotionEvent))

Indipendentemente dal fatto che si utilizzi o meno , è possibile implementare un metodo che restituisca . Questo perché tutti i gesti iniziano con un messaggio. Se si ritorna da , come per impostazione predefinita, il sistema presuppone che si desideri ignorare il resto del movimento e gli altri metodi di non essere mai chiamati. Ciò può causare problemi imprevisti nell'app. L'unica volta da cui dovresti tornare è se vuoi davvero ignorare un intero gesto. [GestureDetector.OnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener)[onDown()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onDown(android.view.MotionEvent))true[onDown()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onDown(android.view.MotionEvent))false[onDown()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onDown(android.view.MotionEvent))[GestureDetector.SimpleOnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener)[GestureDetector.OnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener)false[onDown()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onDown(android.view.MotionEvent))

[KOTLIN](https://developer.android.com/training/gestures/detector#kotlin)

private const val DEBUG\_TAG = "Gestures"  
  
class MainActivity : Activity() {  
  
    private lateinit var mDetector: GestureDetectorCompat  
  
    public override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity\_main)  
        mDetector = GestureDetectorCompat(this, MyGestureListener())  
    }  
  
    override fun onTouchEvent(event: MotionEvent): Boolean {  
        mDetector.onTouchEvent(event)  
        return super.onTouchEvent(event)  
    }  
  
    private class MyGestureListener : GestureDetector.SimpleOnGestureListener() {  
  
        override fun onDown(event: MotionEvent): Boolean {  
            Log.d(DEBUG\_TAG, "onDown: $event")  
            return true  
        }  
  
        override fun onFling(  
                event1: MotionEvent,  
                event2: MotionEvent,  
                velocityX: Float,  
                velocityY: Float  
        ): Boolean {  
            Log.d(DEBUG\_TAG, "onFling: $event1 $event2")  
            return true  
        }  
    }  
}

# Tenere traccia dei movimenti del tocco e del puntatore

Questa lezione descrive come tenere traccia del movimento negli eventi di tocco.

Un nuovo evento viene attivato con un evento ogni volta che cambia la posizione, la pressione o le dimensioni correnti del contatto tocco. Come descritto in [Rilevamento di gesti comuni](https://developer.android.com/training/gestures/detector), tutti questi eventi vengono registrati nel parametro di .[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))[ACTION\_MOVE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Poiché il tocco basato sulle dita non è sempre la forma più precisa di interazione, il rilevamento degli eventi tocco si basa spesso più sul movimento che sul semplice contatto. Per aiutare le app a distinguere tra gesti basati sul movimento (ad esempio uno scorrimento rapido) e gesti non di movimento (ad esempio un solo tocco), Android include la nozione di "tocco slop". Lo slop tocco si riferisce alla distanza in pixel che il tocco di un utente può vagare prima che il gesto sia interpretato come un gesto basato sul movimento. Per ulteriori informazioni su questo argomento, vedere [Gestione degli eventi di tocco in un ViewGroup](https://developer.android.com/training/gestures/viewgroup#vc).

Esistono diversi modi per tenere traccia del movimento in un gesto, a seconda delle esigenze dell'applicazione. Per esempio:

* Posizione iniziale e finale di un puntatore (ad esempio, spostare un oggetto sullo schermo dal punto A al punto B).
* Direzione in cui viaggia il puntatore, come determinato dalle coordinate x e y.
* storia. È possibile trovare le dimensioni della cronologia di un gesto chiamando il metodo . È quindi possibile ottenere le posizioni, le dimensioni, il tempo e le pressioni di ciascuno degli eventi storici utilizzando i metodi dell'evento di movimento. La cronologia è utile quando si esegue il rendering di una traccia del dito dell'utente, ad esempio per il disegno tramite tocco. Vedi il riferimento per i dettagli.[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[getHistorySize()](https://developer.android.com/reference/android/view/MotionEvent#getHistorySize())getHistorical<Value>[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)
* Velocità del puntatore mentre si sposta sul touchscreen.

Fare riferimento alle risorse correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events) Guida api
* [Panoramica dei sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la vista](https://developer.android.com/training/custom-views/making-interactive)

## Velocità traccia

Potresti avere un gesto basato sul movimento che si basa semplicemente sulla distanza e / o sulla direzione percorsa dal puntatore. Ma la velocità spesso è un fattore determinante per tracciare le caratteristiche di un gesto o anche per decidere se il gesto si è verificato. Per semplificare il calcolo della velocità, Android fornisce la classe. ti aiuta a tenere traccia della velocità degli eventi di tocco. Ciò è utile per i gesti in cui la velocità fa parte dei criteri per il gesto, ad esempio un'avventura.[VelocityTrackerVelocityTracker](https://developer.android.com/reference/android/view/VelocityTracker)

Ecco un semplice esempio che illustra lo scopo dei metodi nell'API:[VelocityTracker](https://developer.android.com/reference/android/view/VelocityTracker)

[KOTLIN](https://developer.android.com/training/gestures/movement#kotlin)

private const val DEBUG\_TAG = "Velocity"  
  
class MainActivity : Activity() {  
    private var mVelocityTracker: VelocityTracker? = null  
  
    override fun onTouchEvent(event: MotionEvent): Boolean {  
  
        when (event.actionMasked) {  
            MotionEvent.ACTION\_DOWN -> {  
                // Reset the velocity tracker back to its initial state.  
                mVelocityTracker?.clear()  
                // If necessary retrieve a new VelocityTracker object to watch the  
                // velocity of a motion.  
                mVelocityTracker = mVelocityTracker ?: VelocityTracker.obtain()  
                // Add a user's movement to the tracker.  
                mVelocityTracker?.addMovement(event)  
            }  
            MotionEvent.ACTION\_MOVE -> {  
                mVelocityTracker?.apply {  
                    val pointerId: Int = event.getPointerId(event.actionIndex)  
                    addMovement(event)  
                    // When you want to determine the velocity, call  
                    // computeCurrentVelocity(). Then call getXVelocity()  
                    // and getYVelocity() to retrieve the velocity for each pointer ID.  
                    computeCurrentVelocity(1000)  
                    // Log velocity of pixels per second  
                    // Best practice to use VelocityTrackerCompat where possible.  
                    Log.d("", "X velocity: ${getXVelocity(pointerId)}")  
                    Log.d("", "Y velocity: ${getYVelocity(pointerId)}")  
                }  
            }  
            MotionEvent.ACTION\_UP, MotionEvent.ACTION\_CANCEL -> {  
                // Return a VelocityTracker object back to be re-used by others.  
                mVelocityTracker?.recycle()  
                mVelocityTracker = null  
            }  
        }  
        return true  
    }  
}

**Nota:** è necessario calcolare la velocità dopo un evento [**ACTION\_MOVE,**](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)non dopo [**ACTION\_UP**](https://developer.android.com/reference/android/view/MotionEvent#ACTION_UP). Dopo un [**ACTION\_UP,**](https://developer.android.com/reference/android/view/MotionEvent#ACTION_UP)le velocità X e Y saranno 0.

## Usare l'acquisizione del puntatore

Alcune app, ad esempio giochi, desktop remoto e client di virtualizzazione, traggono grande vantaggio dal controllo del puntatore del mouse. L'acquisizione del puntatore è una funzionalità disponibile in Android 8.0 (livello API 26) e versioni successive che fornisce tale controllo offrendo tutti gli eventi del mouse a una visualizzazione mirata nella tua app.

### Richiedi acquisizione puntatore

Una visualizzazione nell'app può richiedere l'acquisizione del puntatore solo quando la gerarchia di visualizzazione che la contiene ha lo stato attivo. Per questo motivo, è necessario richiedere l'acquisizione del puntatore quando è presente un'azione utente specifica nella visualizzazione, ad esempio durante un evento o nel gestore eventi dell'attività.[onClick()](https://developer.android.com/reference/android/view/View.OnClickListener#onClick(android.view.View))[onWindowFocusChanged()](https://developer.android.com/reference/android/app/Activity#onWindowFocusChanged(boolean))

Per richiedere l'acquisizione del puntatore, chiamare il metodo nella visualizzazione. Nell'esempio di codice seguente viene illustrato come richiedere l'acquisizione del puntatore quando l'utente fa clic su una visualizzazione:[requestPointerCapture()](https://developer.android.com/reference/android/view/View#requestPointerCapture())

[KOTLIN](https://developer.android.com/training/gestures/movement#kotlin)

fun onClick(view: View) {  
    view.requestPointerCapture()  
}

Una volta che la richiesta di acquisizione del puntatore ha esito positivo, Android chiama . Il sistema fornisce gli eventi del mouse alla visualizzazione mirata nell'app purché si trova nella stessa gerarchia di visualizzazione della visualizzazione che ha richiesto l'acquisizione. Altre app smettono di ricevere eventi del mouse fino al rilascio dell'acquisizione, inclusi gli eventi. Android fornisce eventi puntatore da fonti diverse dal mouse normalmente, ma il puntatore del mouse non è più visibile.[onPointerCaptureChange(true)](https://developer.android.com/reference/android/view/View#onPointerCaptureChange(boolean))[ACTION\_OUTSIDE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_OUTSIDE)

### Gestire gli eventi puntatore acquisiti

Una volta che una vista ha acquisito correttamente l'acquisizione del puntatore, Android inizia a fornire gli eventi del mouse. La visualizzazione mirata può gestire gli eventi eseguendo una delle attività seguenti:

1. Se si utilizza una visualizzazione personalizzata, eseguire l'override di .[onCapturedPointerEvent(MotionEvent)](https://developer.android.com/reference/android/view/View" \l "onCapturedPointerEvent(android.view.MotionEvent))
2. In caso contrario, registrare un file .[OnCapturedPointerListener](https://developer.android.com/reference/android/view/View.OnCapturedPointerListener)

Nell'esempio di codice riportato di seguito viene illustrato come implementare :[onCapturedPointerEvent(MotionEvent)](https://developer.android.com/reference/android/view/View" \l "onCapturedPointerEvent(android.view.MotionEvent))

[KOTLIN](https://developer.android.com/training/gestures/movement#kotlin)

override fun onCapturedPointerEvent(motionEvent: MotionEvent): Boolean {  
    // Get the coordinates required by your app  
    val verticalOffset: Float = motionEvent.y  
    // Use the coordinates to update your view and return true if the event was  
    // successfully processed  
    return true  
}

Nell'esempio di codice riportato di seguito viene illustrato come registrare un oggetto :[OnCapturedPointerListener](https://developer.android.com/reference/android/view/View.OnCapturedPointerListener)

[KOTLIN](https://developer.android.com/training/gestures/movement#kotlin)

myView.setOnCapturedPointerListener { view, motionEvent ->  
    // Get the coordinates required by your app  
    val horizontalOffset: Float = motionEvent.x  
    // Use the coordinates to update your view and return true if the event was  
    // successfully processed  
    true  
}

Sia che tu usi una vista personalizzata o registri un listener, la vista riceve un oggetto con coordinate del puntatore che specificano movimenti relativi come delta X/Y, simili alle coordinate fornite da un dispositivo trackball. È possibile recuperare le coordinate utilizzando e .[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[getX()](https://developer.android.com/reference/android/view/MotionEvent#getX())[getY()](https://developer.android.com/reference/android/view/MotionEvent#getY())

### Rilasciare l'acquisizione del puntatore

La visualizzazione nell'app può rilasciare l'acquisizione del puntatore chiamando , come illustrato nell'esempio di codice seguente:[releasePointerCapture()](https://developer.android.com/reference/android/view/View#releasePointerCapture())

[KOTLIN](https://developer.android.com/training/gestures/movement#kotlin)

override fun onClick(view: View) {  
    view.releasePointerCapture()  
}

Il sistema può togliere l'acquisizione dalla visualizzazione senza chiamare esplicitamente , più comunemente perché la gerarchia di visualizzazione che contiene la visualizzazione che ha richiesto l'acquisizione ha perso lo stato attivo.[releasePointerCapture()](https://developer.android.com/reference/android/view/View#releasePointerCapture())

[**Animare un gesto di scorrimento**](https://developer.android.com/training/gestures/scroll)

Informazioni su come utilizzare gli scroller ( o ) per produrre un'animazione a scorrimento in risposta a un evento tocco. [Scroller](https://developer.android.com/reference/android/widget/Scroller)[OverScroller](https://developer.android.com/reference/android/widget/OverScroller)

# Animare un gesto di scorrimento

In Android, lo scorrimento si ottiene in genere utilizzando la classe. Qualsiasi layout standard che potrebbe estendersi oltre i limiti del relativo contenitore deve essere annidato in un oggetto per fornire una visualizzazione scorrevole gestita dal framework. L'implementazione di uno scroller personalizzato dovrebbe essere necessaria solo per scenari speciali. In questa lezione viene descritto uno scenario di questo tipo: visualizzazione di un effetto di scorrimento in risposta ai movimenti tocco mediante scroller. [ScrollViewScrollView](https://developer.android.com/reference/android/widget/ScrollView)

È possibile utilizzare gli scroller ( o ) per raccogliere i dati necessari per produrre un'animazione a scorrimento in risposta a un evento tocco. Sono simili, ma includono metodi per indicare agli utenti di aver raggiunto i bordi del contenuto dopo una panoramica o un gesto di lancio. Nell'esempio viene utilizzata la classe (in realtà la classe) per visualizzare un effetto "bagliore" quando gli utenti raggiungono i bordi del contenuto.[Scroller](https://developer.android.com/reference/android/widget/Scroller)[OverScrollerOverScroller](https://developer.android.com/reference/android/widget/OverScroller)InteractiveChart[EdgeEffect](https://developer.android.com/reference/android/widget/EdgeEffect)[EdgeEffectCompat](https://developer.android.com/reference/androidx/core/widget/EdgeEffectCompat)

**Nota:** Ti consigliamo di usare piuttosto che per le animazioni a scorrimento. offre la migliore compatibilità con le versioni precedenti con i dispositivi meno recenti.  
Si noti inoltre che in genere è necessario utilizzare solo gli scroller quando si implementa lo scorrimento. e fai tutto questo per te se annidi il tuo layout al loro interno. [**OverScroller**](https://developer.android.com/reference/android/widget/OverScroller)[**Scroller**](https://developer.android.com/reference/android/widget/Scroller)[**OverScroller**](https://developer.android.com/reference/android/widget/OverScroller)[**ScrollView**](https://developer.android.com/reference/android/widget/ScrollView)[**HorizontalScrollView**](https://developer.android.com/reference/android/widget/HorizontalScrollView)

Uno scroller viene utilizzato per animare lo scorrimento nel tempo, usando la fisica dello scorrimento standard della piattaforma (attrito, velocità, ecc.). Lo scroller stesso in realtà non disegna nulla. Gli scroller tracciano gli offset di scorrimento nel tempo, ma non applicano automaticamente tali posizioni alla visualizzazione. È tua responsabilità ottenere e applicare nuove coordinate a una velocità che renderà l'animazione a scorrimento fluida.

Fare riferimento alle risorse correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events) Guida api
* [Panoramica dei sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la vista](https://developer.android.com/training/custom-views/making-interactive)

## Comprendere la terminologia a scorrimento

"Scorrimento" è una parola che può assumere significati diversi in Android, a seconda del contesto.

**Lo scorrimento** è il processo generale di spostamento del riquadro di visualizzazione,o cioè la "finestra" del contenuto che si sta guardando). Quando lo scorrimento è sia negli assi x che y, viene chiamato panoramica. Nell'applicazione di esempio fornita con questa classe , vengono illustrate due diversi tipi di scorrimento, trascinamento e lancio:InteractiveChart

* **Il trascinamento** è il tipo di scorrimento che si verifica quando un utente trascina il dito sul touchscreen. Il trascinamento semplice viene spesso implementato eseguendo l'override in . Per ulteriori informazioni sul trascinamento, vedere [Trascinamento e ridimensionamento](https://developer.android.com/training/gestures/scale).[onScroll()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onScroll(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[GestureDetector.OnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener)
* **L'avventura** è il tipo di scorrimento che si verifica quando un utente trascina e solleva rapidamente il dito. Dopo che l'utente ha alzato il dito, in genere si desidera continuare a scorrere (spostando il riquadro di visualizzazione), ma rallentare fino a quando il riquadro di visualizzazione non smette di muoversi. L'avventura può essere implementata eseguendo l'override in e utilizzando un oggetto scroller. Questo è il caso d'uso che è l'argomento di questa lezione.[onFling()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onFling(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[GestureDetector.OnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener)

È comune usare gli oggetti scroller insieme a un gesto di lancio, ma possono essere usati praticamente in qualsiasi contesto in cui si desidera che l'interfaccia utente venga visualizzata scorrendo in risposta a un evento tocco. Ad esempio, è possibile eseguire l'override per elaborare direttamente gli eventi tocco e produrre un effetto di scorrimento o un'animazione "aggancio alla pagina" in risposta a tali eventi tocco.[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

## Implementare lo scorrimento basato sul tocco

In questa sezione viene descritto come utilizzare uno scroller. Lo snippet mostrato di seguito proviene dall'esempio fornito con questa classe. Utilizza un oggetto e esegue l'override del metodo . Usa per tenere traccia del gesto di lancio. Se l'utente raggiunge i bordi del contenuto dopo il gesto di lancio, l'app visualizza un effetto "bagliore". InteractiveChart[GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)[GestureDetector.SimpleOnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener)[onFling()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onFling(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[OverScroller](https://developer.android.com/reference/android/widget/OverScroller)

**Nota:** L'app di esempio visualizza un grafico che puoi ingrandire, scorrere, scorrere e così via. Nel frammento seguente rappresenta le coordinate del rettangolo all'interno della visualizzazione in cui verrà disegnato il grafico. In un dato momento, un sottoinsieme del dominio e dell'intervallo totale del grafico viene disegnato in quest'area rettangolare. rappresenta la parte del grafico attualmente visibile sullo schermo. Poiché gli offset dei pixel vengono generalmente trattati come numeri interi, è di tipo . Poiché il dominio e l'intervallo del grafico sono valori decimali/float, è di tipo .**InteractiveChartmContentRectmCurrentViewportmContentRect**[**Rect**](https://developer.android.com/reference/android/graphics/Rect)**mCurrentViewport**[**RectF**](https://developer.android.com/reference/android/graphics/RectF)

La prima parte dello snippet mostra l'implementazione di :[onFling()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onFling(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))

[KOTLIN](https://developer.android.com/training/gestures/scroll#kotlin)

// The current viewport. This rectangle represents the currently visible  
// chart domain and range. The viewport is the part of the app that the  
// user manipulates via touch gestures.  
private val mCurrentViewport = RectF(AXIS\_X\_MIN, AXIS\_Y\_MIN, AXIS\_X\_MAX, AXIS\_Y\_MAX)  
  
// The current destination rectangle (in pixel coordinates) into which the  
// chart data should be drawn.  
private lateinit var mContentRect: Rect  
  
private lateinit var mScroller: OverScroller  
private lateinit var mScrollerStartViewport: RectF  
...  
private val mGestureListener = object : GestureDetector.SimpleOnGestureListener() {  
  
    override fun onDown(e: MotionEvent): Boolean {  
        // Initiates the decay phase of any active edge effects.  
        releaseEdgeEffects()  
        mScrollerStartViewport.set(mCurrentViewport)  
        // Aborts any active scroll animations and invalidates.  
        mScroller.forceFinished(true)  
        ViewCompat.postInvalidateOnAnimation(this@InteractiveLineGraphView)  
        return true  
    }  
    ...  
    override fun onFling(  
            e1: MotionEvent,  
            e2: MotionEvent,  
            velocityX: Float,  
            velocityY: Float  
    ): Boolean {  
        fling((-velocityX).toInt(), (-velocityY).toInt())  
        return true  
    }  
}  
  
private fun fling(velocityX: Int, velocityY: Int) {  
    // Initiates the decay phase of any active edge effects.  
    releaseEdgeEffects()  
    // Flings use math in pixels (as opposed to math based on the viewport).  
    val surfaceSize: Point = computeScrollSurfaceSize()  
    val (startX: Int, startY: Int) = mScrollerStartViewport.run {  
        set(mCurrentViewport)  
        (surfaceSize.x \* (left - AXIS\_X\_MIN) / (AXIS\_X\_MAX - AXIS\_X\_MIN)).toInt() to  
                (surfaceSize.y \* (AXIS\_Y\_MAX - bottom) / (AXIS\_Y\_MAX - AXIS\_Y\_MIN)).toInt()  
    }  
    // Before flinging, aborts the current animation.  
    mScroller.forceFinished(true)  
    // Begins the animation  
    mScroller.fling(  
            // Current scroll position  
            startX,  
            startY,  
            velocityX,  
            velocityY,  
            /\*  
             \* Minimum and maximum scroll positions. The minimum scroll  
             \* position is generally zero and the maximum scroll position  
             \* is generally the content size less the screen size. So if the  
             \* content width is 1000 pixels and the screen width is 200  
             \* pixels, the maximum scroll offset should be 800 pixels.  
             \*/  
            0, surfaceSize.x - mContentRect.width(),  
            0, surfaceSize.y - mContentRect.height(),  
            // The edges of the content. This comes into play when using  
            // the EdgeEffect class to draw "glow" overlays.  
            mContentRect.width() / 2,  
            mContentRect.height() / 2  
    )  
    // Invalidates to trigger computeScroll()  
    ViewCompat.postInvalidateOnAnimation(this)  
}

Quando chiama , viene attivato per aggiornare i valori per x e y. Questa situazione viene in genere eseguita quando un figlio della visualizzazione anima uno scorrimento utilizzando un oggetto scroller, come in questo esempio. [onFling()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onFling(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[postInvalidateOnAnimation()](https://developer.android.com/reference/androidx/core/view/ViewCompat#postInvalidateOnAnimation(android.view.View))[computeScroll()](https://developer.android.com/reference/android/view/View#computeScroll())

La maggior parte delle visualizzazioni passa la posizione x e y dell'oggetto scroller direttamente a . L'implementazione seguente di adotta un approccio diverso: chiama per ottenere la posizione corrente di x e y. Quando vengono soddisfatti i criteri per la visualizzazione di un effetto bordo "bagliore" overscroll (la visualizzazione viene ingrandita, x o y è fuori dai limiti e l'app non mostra già uno scorrimento esterno), il codice imposta l'effetto bagliore dello scorrimento e chiama per attivare un'invalidazione nella visualizzazione:[scrollTo()](https://developer.android.com/reference/android/view/View#scrollTo(int,%20int))[computeScroll()](https://developer.android.com/reference/android/view/View#computeScroll())[computeScrollOffset()](https://developer.android.com/reference/android/widget/OverScroller#computeScrollOffset())[postInvalidateOnAnimation()](https://developer.android.com/reference/androidx/core/view/ViewCompat#postInvalidateOnAnimation(android.view.View))

[KOTLIN](https://developer.android.com/training/gestures/scroll#kotlin)

// Edge effect / overscroll tracking objects.  
private lateinit var mEdgeEffectTop: EdgeEffect  
private lateinit var mEdgeEffectBottom: EdgeEffect  
private lateinit var mEdgeEffectLeft: EdgeEffect  
private lateinit var mEdgeEffectRight: EdgeEffect  
  
private var mEdgeEffectTopActive: Boolean = false  
private var mEdgeEffectBottomActive: Boolean = false  
private var mEdgeEffectLeftActive: Boolean = false  
private var mEdgeEffectRightActive: Boolean = false  
  
override fun computeScroll() {  
    super.computeScroll()  
  
    var needsInvalidate = false  
  
    // The scroller isn't finished, meaning a fling or programmatic pan  
    // operation is currently active.  
    if (mScroller.computeScrollOffset()) {  
        val surfaceSize: Point = computeScrollSurfaceSize()  
        val currX: Int = mScroller.currX  
        val currY: Int = mScroller.currY  
  
        val (canScrollX: Boolean, canScrollY: Boolean) = mCurrentViewport.run {  
            (left > AXIS\_X\_MIN || right < AXIS\_X\_MAX) to (top > AXIS\_Y\_MIN || bottom < AXIS\_Y\_MAX)  
        }  
  
        /\*  
         \* If you are zoomed in and currX or currY is  
         \* outside of bounds and you are not already  
         \* showing overscroll, then render the overscroll  
         \* glow edge effect.  
         \*/  
        if (canScrollX  
                && currX < 0  
                && mEdgeEffectLeft.isFinished  
                && !mEdgeEffectLeftActive) {  
            mEdgeEffectLeft.onAbsorb(mScroller.currVelocity.toInt())  
            mEdgeEffectLeftActive = true  
            needsInvalidate = true  
        } else if (canScrollX  
                && currX > surfaceSize.x - mContentRect.width()  
                && mEdgeEffectRight.isFinished  
                && !mEdgeEffectRightActive) {  
            mEdgeEffectRight.onAbsorb(mScroller.currVelocity.toInt())  
            mEdgeEffectRightActive = true  
            needsInvalidate = true  
        }  
  
        if (canScrollY  
                && currY < 0  
                && mEdgeEffectTop.isFinished  
                && !mEdgeEffectTopActive) {  
            mEdgeEffectTop.onAbsorb(mScroller.currVelocity.toInt())  
            mEdgeEffectTopActive = true  
            needsInvalidate = true  
        } else if (canScrollY  
                && currY > surfaceSize.y - mContentRect.height()  
                && mEdgeEffectBottom.isFinished  
                && !mEdgeEffectBottomActive) {  
            mEdgeEffectBottom.onAbsorb(mScroller.currVelocity.toInt())  
            mEdgeEffectBottomActive = true  
            needsInvalidate = true  
        }  
        ...  
    }  
}

Ecco la sezione del codice che esegue lo zoom effettivo:

[KOTLIN](https://developer.android.com/training/gestures/scroll#kotlin)

lateinit var mZoomer: Zoomer  
val mZoomFocalPoint = PointF()  
...  
  
// If a zoom is in progress (either programmatically or via double  
// touch), performs the zoom.  
if (mZoomer.computeZoom()) {  
    val newWidth: Float = (1f - mZoomer.currZoom) \* mScrollerStartViewport.width()  
    val newHeight: Float = (1f - mZoomer.currZoom) \* mScrollerStartViewport.height()  
    val pointWithinViewportX: Float =  
            (mZoomFocalPoint.x - mScrollerStartViewport.left) / mScrollerStartViewport.width()  
    val pointWithinViewportY: Float =  
            (mZoomFocalPoint.y - mScrollerStartViewport.top) / mScrollerStartViewport.height()  
    mCurrentViewport.set(  
            mZoomFocalPoint.x - newWidth \* pointWithinViewportX,  
            mZoomFocalPoint.y - newHeight \* pointWithinViewportY,  
            mZoomFocalPoint.x + newWidth \* (1 - pointWithinViewportX),  
            mZoomFocalPoint.y + newHeight \* (1 - pointWithinViewportY)  
    )  
    constrainViewport()  
    needsInvalidate = true  
}  
if (needsInvalidate) {  
    ViewCompat.postInvalidateOnAnimation(this)  
}

Questo è il metodo che viene chiamato nel frammento di cui sopra. Calcola le dimensioni correnti della superficie scorrevole, in pixel. Ad esempio, se l'intera area del grafico è visibile, questa è semplicemente la dimensione corrente di . Se il grafico viene ingrandito del 200% in entrambe le direzioni, le dimensioni restituite saranno due volte più grandi orizzontalmente e verticalmente.computeScrollSurfaceSize()mContentRect

[KOTLIN](https://developer.android.com/training/gestures/scroll#kotlin)

private fun computeScrollSurfaceSize(): Point {  
    return Point(  
            (mContentRect.width() \* (AXIS\_X\_MAX - AXIS\_X\_MIN) / mCurrentViewport.width()).toInt(),  
            (mContentRect.height() \* (AXIS\_Y\_MAX - AXIS\_Y\_MIN) / mCurrentViewport.height()).toInt()  
    )  
}

Per un altro esempio di utilizzo dello scroller, vedere [il codice sorgente](https://android.googlesource.com/platform/frameworks/support/+/5b614a46f6ffb3e9ca5ab6321c12412550a4e13a/viewpager/src/main/java/androidx/viewpager/widget/ViewPager.java) per la classe . Scorre in risposta alle fling e usa lo scorrimento per implementare l'animazione "aggancia alla pagina".[ViewPager](https://developer.android.com/reference/androidx/viewpager/widget/ViewPager)

[**Gestire i gesti multitoccare**](https://developer.android.com/training/gestures/multi)

Scopri come rilevare i gesti multipunto (dito).

# Gestire i gesti multitoccare

Un gesto multitoccare è quando più puntatori (dita) toccano lo schermo contemporaneamente. Questa lezione descrive come rilevare i gesti che coinvolgono più puntatori.

Fare riferimento alle risorse correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events) Guida api
* [Panoramica dei sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la vista](https://developer.android.com/training/custom-views/making-interactive)

## Tenere traccia di più puntatori

Quando più puntatori toccano lo schermo contemporaneamente, il sistema genera i seguenti eventi di tocco:

* [ACTION\_DOWN](https://developer.android.com/reference/android/view/MotionEvent#ACTION_DOWN)- Per il primo puntatore che tocca lo schermo. Questo inizia il gesto. I dati del puntatore per questo puntatore sono sempre in indice 0 in .[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)
* [ACTION\_POINTER\_DOWN](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#ACTION_POINTER_DOWN)- Per puntatori aggiuntivi che entrano nella schermata oltre il primo. I dati del puntatore per questo puntatore si trova in base all'indice restituito da .[getActionIndex()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat" \l "getActionIndex(android.view.MotionEvent))
* [ACTION\_MOVE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)- Durante un gesto della stampa si è verificato un cambiamento.
* [ACTION\_POINTER\_UP](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#ACTION_POINTER_UP)- Inviato quando un puntatore non primario sale.
* [ACTION\_UP](https://developer.android.com/reference/android/view/MotionEvent#ACTION_UP)- Inviato quando l'ultimo puntatore esce dallo schermo.

Tieni traccia dei singoli puntatori all'interno di un tramite l'indice e l'ID di ogni puntatore:[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

* **Indice:**un oggetto memorizza in modo efficace le informazioni su ogni puntatore in una matrice. L'indice di un puntatore è la sua posizione all'interno di questa matrice. La maggior parte dei metodi utilizzati per interagire con i puntatori prendono l'indice del puntatore come parametro, non l'ID puntatore. [MotionEventMotionEvent](https://developer.android.com/reference/android/view/MotionEvent)
* **ID:**ogni puntatore ha anche un mapping id che rimane persistente tra gli eventi di tocco per consentire il rilevamento di un singolo puntatore sull'intero gesto.

L'ordine in cui i singoli puntatori vengono visualizzati all'interno di un evento di movimento non è definito. Pertanto, l'indice di un puntatore può passare da un evento all'altro, ma è garantito che l'ID puntatore di un puntatore rimanga costante finché il puntatore rimane attivo. Usa il metodo per ottenere l'ID di un puntatore per tenere traccia del puntatore in tutti gli eventi di movimento successivi in un gesto. Quindi, per eventi di movimento successivi, utilizzare il metodo per ottenere l'indice del puntatore per un determinato ID puntatore in quell'evento di movimento. Per esempio:[getPointerId()](https://developer.android.com/reference/android/view/MotionEvent#getPointerId(int))[findPointerIndex()](https://developer.android.com/reference/android/view/MotionEvent#findPointerIndex(int))

[KOTLIN](https://developer.android.com/training/gestures/multi#kotlin)

private var mActivePointerId: Int = 0  
  
override fun onTouchEvent(event: MotionEvent): Boolean {  
    ...  
    // Get the pointer ID  
    mActivePointerId = event.getPointerId(0)  
  
    // ... Many touch events later...  
  
    // Use the pointer ID to find the index of the active pointer  
    // and fetch its position  
    val (x: Float, y: Float) = event.findPointerIndex(mActivePointerId).let { pointerIndex ->  
        // Get the pointer's current position  
        event.getX(pointerIndex) to event.getY(pointerIndex)  
    }  
    ...  
}

## Ottenere l'azione di un MotionEvent

È sempre necessario utilizzare il metodo (o, meglio ancora, la versione di compatibilità ) per recuperare l'azione di un oggetto . A differenza del metodo precedente, è progettato per funzionare con più puntatori. Restituisce l'azione mascherata eseguita, senza includere i bit di indice del puntatore. È quindi possibile utilizzare per restituire l'indice del puntatore associato all'azione. Questo è illustrato nello snippet qui sotto.[getActionMasked()](https://developer.android.com/reference/android/view/MotionEvent#getActionMasked())[MotionEventCompat.getActionMasked()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#getActionMasked(android.view.MotionEvent))[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[getAction()](https://developer.android.com/reference/android/view/MotionEvent#getAction())[getActionMasked()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#getActionMasked(android.view.MotionEvent))[getActionIndex()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#getActionIndex(android.view.MotionEvent))

**Nota:** In questo esempio viene utilizzata la classe . Questa classe si trova nella Libreria [di supporto](https://developer.android.com/tools/support-library). Dovresti usare per fornire il miglior supporto per una vasta gamma di piattaforme. Si noti che non è una sostituzione per la classe. Piuttosto, fornisce metodi di utilità statici a cui passi l'oggetto per ricevere l'azione desiderata associata a quell'evento.[**MotionEventCompatMotionEventCompatMotionEventCompat**](https://developer.android.com/reference/androidx/core/view/MotionEventCompat)[**MotionEventMotionEvent**](https://developer.android.com/reference/android/view/MotionEvent)

[KOTLIN](https://developer.android.com/training/gestures/multi#kotlin)

val (xPos: Int, yPos: Int) = MotionEventCompat.getActionMasked(event).let { action ->  
    Log.d(DEBUG\_TAG, "The action is ${actionToString(action)}")  
    // Get the index of the pointer associated with the action.  
    MotionEventCompat.getActionIndex(event).let { index ->  
        // The coordinates of the current screen contact, relative to  
        // the responding View or Activity.  
        MotionEventCompat.getX(event, index).toInt() to MotionEventCompat.getY(event, index).toInt()  
    }  
}  
  
if (event.pointerCount > 1) {  
    Log.d(DEBUG\_TAG, "Multitouch event")  
  
} else {  
    // Single touch event  
    Log.d(DEBUG\_TAG, "Single touch event")  
}  
  
...  
  
// Given an action int, returns a string description  
fun actionToString(action: Int): String {  
    return when (action) {  
        MotionEvent.ACTION\_DOWN -> "Down"  
        MotionEvent.ACTION\_MOVE -> "Move"  
        MotionEvent.ACTION\_POINTER\_DOWN -> "Pointer Down"  
        MotionEvent.ACTION\_UP -> "Up"  
        MotionEvent.ACTION\_POINTER\_UP -> "Pointer Up"  
        MotionEvent.ACTION\_OUTSIDE -> "Outside"  
        MotionEvent.ACTION\_CANCEL -> "Cancel"  
        else -> ""  
    }  
}

Per ulteriori informazioni sul multitoccare e su alcuni esempi, vedere la lezione [Drag and Scale](https://developer.android.com/training/gestures/scale).

[**Trascinare e ridimensionare**](https://developer.android.com/training/gestures/scale)

Informazioni su come implementare il trascinamento e il ridimensionamento basati sul tocco.

# Trascinare e ridimensionare

Questa lezione descrive come usare i gesti tocco per trascinare e ridimensionare gli oggetti sullo schermo, usando per intercettare gli eventi di tocco. [onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Fare riferimento alle risorse correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events) Guida api
* [Panoramica dei sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la vista](https://developer.android.com/training/custom-views/making-interactive)

## Trascinare un oggetto

Se si è destinati ad Android 3.0 o versione successiva, è possibile utilizzare i listener di eventi di trascinamento della selezione integrati con , come descritto in [Trascinamento della selezione](https://developer.android.com/guide/topics/ui/drag-drop). [**View.OnDragListener**](https://developer.android.com/reference/android/view/View.OnDragListener)

Un'operazione comune per un gesto tocco è usarla per trascinare un oggetto sullo schermo. Il frammento seguente consente all'utente di trascinare un'immagine sullo schermo. Si noti quanto segue:

* In un'operazione di trascinamento (o scorrimento), l'app deve tenere traccia del puntatore originale (dito), anche se sullo schermo vengono posizionate altre dita. Ad esempio, immagina che mentre trascina l'immagine in giro, l'utente posiziona un secondo dito sul touch screen e solleva il primo dito. Se la tua app sta solo monitorando singoli puntatori, considererà il secondo puntatore come predefinito e sposterà l'immagine in quella posizione.
* Per evitare che ciò accada, l'app deve distinguere tra il puntatore originale e gli eventuali puntatori di follow-on. A tale fine, tiene traccia degli eventi e descritti in [Gestione dei gesti multitoccare](https://developer.android.com/training/gestures/multi). e vengono passati al callback ogni volta che un puntatore secondario scende o sale. [ACTION\_POINTER\_DOWN](https://developer.android.com/reference/android/view/MotionEvent#ACTION_POINTER_DOWN)[ACTION\_POINTER\_UP](https://developer.android.com/reference/android/view/MotionEvent#ACTION_POINTER_UP)[ACTION\_POINTER\_DOWN](https://developer.android.com/reference/android/view/MotionEvent#ACTION_POINTER_DOWN)[ACTION\_POINTER\_UP](https://developer.android.com/reference/android/view/MotionEvent#ACTION_POINTER_UP)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))
* Nel caso, l'esempio estrae questo indice e garantisce che l'ID puntatore attivo non si riferisca a un puntatore che non tocca più lo schermo. In caso di parere, l'app seleziona un puntatore diverso per essere attiva e salva la posizione X e Y corrente. Poiché questa posizione salvata viene utilizzata nel caso in cui calcoli la distanza per spostare l'oggetto sullo schermo, l'app calcolerà sempre la distanza da spostare utilizzando i dati dal puntatore corretto.[ACTION\_POINTER\_UP](https://developer.android.com/reference/android/view/MotionEvent#ACTION_POINTER_UP)[ACTION\_MOVE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)

Il frammento seguente consente a un utente di trascinare un oggetto sullo schermo. Registra la posizione iniziale del puntatore attivo, calcola la distanza percorsa dal puntatore e sposta l'oggetto nella nuova posizione. Gestisce correttamente la possibilità di puntatori aggiuntivi, come descritto sopra.

Si noti che il frammento utilizza il metodo . È sempre necessario utilizzare questo metodo (o, meglio ancora, la versione di compatibilità ) per recuperare l'azione di un oggetto . A differenza del metodo precedente, è progettato per funzionare con più puntatori. Restituisce l'azione mascherata eseguita, senza includere i bit di indice del puntatore.[getActionMasked()](https://developer.android.com/reference/android/view/MotionEvent#getActionMasked())[MotionEventCompat.getActionMasked()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#getActionMasked(android.view.MotionEvent))[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[getAction()](https://developer.android.com/reference/android/view/MotionEvent#getAction())[getActionMasked()](https://developer.android.com/reference/androidx/core/view/MotionEventCompat#getActionMasked(android.view.MotionEvent))

[KOTLIN](https://developer.android.com/training/gestures/scale#kotlin)

// The ‘active pointer’ is the one currently moving our object.  
private var mActivePointerId = INVALID\_POINTER\_ID  
  
override fun onTouchEvent(ev: MotionEvent): Boolean {  
    // Let the ScaleGestureDetector inspect all events.  
    mScaleDetector.onTouchEvent(ev)  
  
    val action = MotionEventCompat.getActionMasked(ev)  
  
    when (action) {  
        MotionEvent.ACTION\_DOWN -> {  
            MotionEventCompat.getActionIndex(ev).also { pointerIndex ->  
                // Remember where we started (for dragging)  
                mLastTouchX = MotionEventCompat.getX(ev, pointerIndex)  
                mLastTouchY = MotionEventCompat.getY(ev, pointerIndex)  
            }  
  
            // Save the ID of this pointer (for dragging)  
            mActivePointerId = MotionEventCompat.getPointerId(ev, 0)  
        }  
  
        MotionEvent.ACTION\_MOVE -> {  
            // Find the index of the active pointer and fetch its position  
            val (x: Float, y: Float) =  
                    MotionEventCompat.findPointerIndex(ev, mActivePointerId).let { pointerIndex ->  
                        // Calculate the distance moved  
                        MotionEventCompat.getX(ev, pointerIndex) to  
                                MotionEventCompat.getY(ev, pointerIndex)  
                    }  
  
            mPosX += x - mLastTouchX  
            mPosY += y - mLastTouchY  
  
            invalidate()  
  
            // Remember this touch position for the next move event  
            mLastTouchX = x  
            mLastTouchY = y  
        }  
        MotionEvent.ACTION\_UP, MotionEvent.ACTION\_CANCEL -> {  
            mActivePointerId = INVALID\_POINTER\_ID  
        }  
        MotionEvent.ACTION\_POINTER\_UP -> {  
  
            MotionEventCompat.getActionIndex(ev).also { pointerIndex ->  
                MotionEventCompat.getPointerId(ev, pointerIndex)  
                        .takeIf { it == mActivePointerId }  
                        ?.run {  
                            // This was our active pointer going up. Choose a new  
                            // active pointer and adjust accordingly.  
                            val newPointerIndex = if (pointerIndex == 0) 1 else 0  
                            mLastTouchX = MotionEventCompat.getX(ev, newPointerIndex)  
                            mLastTouchY = MotionEventCompat.getY(ev, newPointerIndex)  
                            mActivePointerId = MotionEventCompat.getPointerId(ev, newPointerIndex)  
                        }  
            }  
        }  
    }  
    return true  
}

## Trascinare nella panoramica

Nella sezione precedente è stato mostrato un esempio di trascinamento di un oggetto sullo schermo. Un altro scenario comune è la panoramica, o quando il movimento di trascinamento di un utente causa lo scorrimento sia negli assi x che y. Il frammento precedente intercettò direttamente le azioni per implementare il trascinamento. Lo snippet in questa sezione sfrutta il supporto integrato della piattaforma per i gesti comuni. Esegue l'override in .[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[onScroll()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onScroll(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))[GestureDetector.SimpleOnGestureListener](https://developer.android.com/reference/android/view/GestureDetector.SimpleOnGestureListener)

Per fornire un po 'più di contesto, viene chiamato quando un utente trascina un dito per eseguire la panoramica del contenuto. viene chiamato solo quando un dito è premuto; non appena il dito viene sollevato dallo schermo, il gesto termina o viene avviato un gesto di lancio (se il dito si muoveva con una certa velocità poco prima che fosse sollevato). Per ulteriori informazioni su scorrimento e lancio, vedere [Animazione di un gesto di scorrimento](https://developer.android.com/training/gestures/scroll).[onScroll()onScroll()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onScroll(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))

Ecco lo snippet per : [onScroll()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onScroll(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))

[KOTLIN](https://developer.android.com/training/gestures/scale#kotlin)

// The current viewport. This rectangle represents the currently visible  
// chart domain and range.  
private val mCurrentViewport = RectF(AXIS\_X\_MIN, AXIS\_Y\_MIN, AXIS\_X\_MAX, AXIS\_Y\_MAX)  
  
// The current destination rectangle (in pixel coordinates) into which the  
// chart data should be drawn.  
private val mContentRect: Rect? = null  
  
private val mGestureListener = object : GestureDetector.SimpleOnGestureListener() {  
    ...  
    override fun onScroll(  
            e1: MotionEvent,  
            e2: MotionEvent,  
            distanceX: Float,  
            distanceY: Float  
    ): Boolean {  
        // Scrolling uses math based on the viewport (as opposed to math using pixels).  
  
        mContentRect?.apply {  
            // Pixel offset is the offset in screen pixels, while viewport offset is the  
            // offset within the current viewport.  
            val viewportOffsetX = distanceX \* mCurrentViewport.width() / width()  
            val viewportOffsetY = -distanceY \* mCurrentViewport.height() / height()  
  
  
            // Updates the viewport, refreshes the display.  
            setViewportBottomLeft(  
                    mCurrentViewport.left + viewportOffsetX,  
                    mCurrentViewport.bottom + viewportOffsetY  
            )  
        }  
  
        return true  
    }  
}

L'implementazione di scorre il riquadro di visualizzazione in risposta al gesto tocco:[onScroll()](https://developer.android.com/reference/android/view/GestureDetector.OnGestureListener#onScroll(android.view.MotionEvent,%20android.view.MotionEvent,%20float,%20float))

[KOTLIN](https://developer.android.com/training/gestures/scale#kotlin)

/\*\*  
 \* Sets the current viewport (defined by mCurrentViewport) to the given  
 \* X and Y positions. Note that the Y value represents the topmost pixel position,  
 \* and thus the bottom of the mCurrentViewport rectangle.  
 \*/  
private fun setViewportBottomLeft(x: Float, y: Float) {  
    /\*  
     \* Constrains within the scroll range. The scroll range is simply the viewport  
     \* extremes (AXIS\_X\_MAX, etc.) minus the viewport size. For example, if the  
     \* extremes were 0 and 10, and the viewport size was 2, the scroll range would  
     \* be 0 to 8.  
     \*/  
  
    val curWidth: Float = mCurrentViewport.width()  
    val curHeight: Float = mCurrentViewport.height()  
    val newX: Float = Math.max(AXIS\_X\_MIN, Math.min(x, AXIS\_X\_MAX - curWidth))  
    val newY: Float = Math.max(AXIS\_Y\_MIN + curHeight, Math.min(y, AXIS\_Y\_MAX))  
  
    mCurrentViewport.set(newX, newY - curHeight, newX + curWidth, newY)  
  
    // Invalidates the View to update the display.  
    ViewCompat.postInvalidateOnAnimation(this)  
}

## Usare il tocco per eseguire il ridimensionamento

Come discusso in [Rilevamento di gesti comuni](https://developer.android.com/training/gestures/detector), consente di rilevare i gesti comuni utilizzati da Android, ad esempio scorrimento, lancio e pressione lunga. Per il ridimensionamento, Android fornisce . e può essere usato insieme quando si desidera che una vista riconosca gesti aggiuntivi.[GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)[ScaleGestureDetector](https://developer.android.com/reference/android/view/ScaleGestureDetector)[GestureDetector](https://developer.android.com/reference/android/view/GestureDetector)[ScaleGestureDetector](https://developer.android.com/reference/android/view/ScaleGestureDetector)

Per segnalare gli eventi gesto rilevati, i rilevatori di gesti utilizzano oggetti listener passati ai costruttori. utilizzazioni. Android fornisce come classe di supporto che puoi estendere se non ti interessa tutti gli eventi segnalati.[ScaleGestureDetector](https://developer.android.com/reference/android/view/ScaleGestureDetector)[ScaleGestureDetector.OnScaleGestureListener](https://developer.android.com/reference/android/view/ScaleGestureDetector.OnScaleGestureListener)[ScaleGestureDetector.SimpleOnScaleGestureListener](https://developer.android.com/reference/android/view/ScaleGestureDetector.SimpleOnScaleGestureListener)

### Esempio di ridimensionamento di base

Ecco uno snippet che illustra gli ingredienti di base coinvolti nel ridimensionamento.

[KOTLIN](https://developer.android.com/training/gestures/scale#kotlin)

private var mScaleFactor = 1f  
  
private val scaleListener = object : ScaleGestureDetector.SimpleOnScaleGestureListener() {  
  
    override fun onScale(detector: ScaleGestureDetector): Boolean {  
        mScaleFactor \*= detector.scaleFactor  
  
        // Don't let the object get too small or too large.  
        mScaleFactor = Math.max(0.1f, Math.min(mScaleFactor, 5.0f))  
  
        invalidate()  
        return true  
    }  
}  
  
private val mScaleDetector = ScaleGestureDetector(context, scaleListener)  
  
override fun onTouchEvent(ev: MotionEvent): Boolean {  
    // Let the ScaleGestureDetector inspect all events.  
    mScaleDetector.onTouchEvent(ev)  
    return true  
}  
  
override fun onDraw(canvas: Canvas?) {  
    super.onDraw(canvas)  
  
    canvas?.apply {  
        save()  
        scale(mScaleFactor, mScaleFactor)  
        // onDraw() code goes here  
        restore()  
    }  
}

### Esempio di ridimensionamento più complesso

Ecco un esempio più complesso dell'esempio fornito con questa classe. L'esempio supporta sia lo scorrimento (panoramica) che il ridimensionamento con più dita, utilizzando le funzionalità "span" () e "focus" ():InteractiveChartInteractiveChart[ScaleGestureDetector](https://developer.android.com/reference/android/view/ScaleGestureDetector)[getCurrentSpanX/Y](https://developer.android.com/reference/android/view/ScaleGestureDetector#getCurrentSpanX())[getFocusX/Y](https://developer.android.com/reference/android/view/ScaleGestureDetector#getFocusX())

[KOTLIN](https://developer.android.com/training/gestures/scale#kotlin)

private val mCurrentViewport = RectF(AXIS\_X\_MIN, AXIS\_Y\_MIN, AXIS\_X\_MAX, AXIS\_Y\_MAX)  
private val mContentRect: Rect? = null  
...  
override fun onTouchEvent(event: MotionEvent): Boolean {  
    return mScaleGestureDetector.onTouchEvent(event)  
            || mGestureDetector.onTouchEvent(event)  
            || super.onTouchEvent(event)  
}  
  
/\*\*  
 \* The scale listener, used for handling multi-finger scale gestures.  
 \*/  
private val mScaleGestureListener = object : ScaleGestureDetector.SimpleOnScaleGestureListener() {  
  
    /\*\*  
     \* This is the active focal point in terms of the viewport. Could be a local  
     \* variable but kept here to minimize per-frame allocations.  
     \*/  
    private val viewportFocus = PointF()  
    private var lastSpanX: Float = 0f  
    private var lastSpanY: Float = 0f  
  
    // Detects that new pointers are going down.  
    override fun onScaleBegin(scaleGestureDetector: ScaleGestureDetector): Boolean {  
        lastSpanX = scaleGestureDetector.currentSpanX  
        lastSpanY = scaleGestureDetector.currentSpanY  
        return true  
    }  
  
    override fun onScale(scaleGestureDetector: ScaleGestureDetector): Boolean {  
        val spanX: Float = scaleGestureDetector.currentSpanX  
        val spanY: Float = scaleGestureDetector.currentSpanY  
  
        val newWidth: Float = lastSpanX / spanX \* mCurrentViewport.width()  
        val newHeight: Float = lastSpanY / spanY \* mCurrentViewport.height()  
  
        val focusX: Float = scaleGestureDetector.focusX  
        val focusY: Float = scaleGestureDetector.focusY  
        // Makes sure that the chart point is within the chart region.  
        // See the sample for the implementation of hitTest().  
        hitTest(focusX, focusY, viewportFocus)  
  
        mContentRect?.apply {  
            mCurrentViewport.set(  
                    viewportFocus.x - newWidth \* (focusX - left) / width(),  
                    viewportFocus.y - newHeight \* (bottom - focusY) / height(),  
                    0f,  
                    0f  
            )  
        }  
        mCurrentViewport.right = mCurrentViewport.left + newWidth  
        mCurrentViewport.bottom = mCurrentViewport.top + newHeight  
        // Invalidates the View to update the display.  
        ViewCompat.postInvalidateOnAnimation(this@InteractiveLineGraphView)  
  
        lastSpanX = spanX  
        lastSpanY = spanY  
        return true  
    }  
}

[**Gestire gli eventi di tocco in un ViewGroup**](https://developer.android.com/training/gestures/viewgroup)

Informazioni su come gestire gli eventi di tocco in un oggetto per garantire che gli eventi di tocco vengano inviati correttamente alle visualizzazioni di destinazione.[ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)

# Gestire gli eventi di tocco in un ViewGroup

Gestire gli eventi touch in un oggetto richiede particolare attenzione, perché è comune che un oggetto abbia figli che sono obiettivi per eventi di tocco diversi da quelli stessi. Per assicurarsi che ogni visualizzazione riceva correttamente gli eventi di tocco previsti, eseguire l'override del metodo .[ViewGroupViewGroupViewGroup](https://developer.android.com/reference/android/view/ViewGroup)[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))

Fare riferimento alle risorse correlate seguenti:

* [Eventi di input](https://developer.android.com/guide/topics/ui/ui-events) Guida api
* [Panoramica dei sensori](https://developer.android.com/guide/topics/sensors/sensors_overview)
* [Rendere interattiva la vista](https://developer.android.com/training/custom-views/making-interactive)

## Intercettare gli eventi di tocco in un ViewGroup

Il metodo viene chiamato ogni volta che viene rilevato un evento touch sulla superficie di un oggetto , anche sulla superficie dei relativi figli. Se restituisce , l'oggetto viene intercettato, il che significa che non viene passato al figlio, ma piuttosto al metodo dell'elemento padre.[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))[ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))true[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Il metodo offre a un genitore la possibilità di vedere qualsiasi evento di tocco prima che lo facciano i suoi figli. Se si ritorna da , la visualizzazione figlio che in precedenza gestiva gli eventi di tocco riceve un oggetto e gli eventi da quel punto in avanti vengono inviati al metodo dell'elemento padre per la gestione abituale. possono anche tornare e semplicemente spiare gli eventi mentre viaggiano lungo la gerarchia di visualizzazione fino ai loro obiettivi abituali, che gestiranno gli eventi con i propri . [onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))true[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))[ACTION\_CANCEL](https://developer.android.com/reference/android/view/MotionEvent#ACTION_CANCEL)[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))false[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

Nel frammento seguente, la classe estende . contiene più visualizzazioni figlio. Se si trascina il dito su una visualizzazione figlio orizzontalmente, la visualizzazione figlio non deve più ottenere eventi di tocco e deve gestire gli eventi tocco scorrendone il contenuto. Tuttavia, se si preme un pulsante nella visualizzazione figlio o si scorre la visualizzazione figlio verticalmente, il padre non deve intercettare tali eventi di tocco, perché il figlio è la destinazione prevista. In questi casi, deve restituire , e 's non verrà chiamato.MyViewGroup[ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)MyViewGroupMyViewGroup[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))falseMyViewGroup[onTouchEvent()](https://developer.android.com/reference/android/view/View#onTouchEvent(android.view.MotionEvent))

[KOTLIN](https://developer.android.com/training/gestures/viewgroup#kotlin)

class MyViewGroup @JvmOverloads constructor(  
        context: Context,  
        private val mTouchSlop: Int = ViewConfiguration.get(context).scaledTouchSlop  
) : ViewGroup(context) {  
  
    ...  
  
    override fun onInterceptTouchEvent(ev: MotionEvent): Boolean {  
        /\*  
         \* This method JUST determines whether we want to intercept the motion.  
         \* If we return true, onTouchEvent will be called and we do the actual  
         \* scrolling there.  
         \*/  
        return when (ev.actionMasked) {  
            // Always handle the case of the touch gesture being complete.  
            MotionEvent.ACTION\_CANCEL, MotionEvent.ACTION\_UP -> {  
                // Release the scroll.  
                mIsScrolling = false  
                false // Do not intercept touch event, let the child handle it  
            }  
            MotionEvent.ACTION\_MOVE -> {  
                if (mIsScrolling) {  
                    // We're currently scrolling, so yes, intercept the  
                    // touch event!  
                    true  
                } else {  
  
                    // If the user has dragged her finger horizontally more than  
                    // the touch slop, start the scroll  
  
                    // left as an exercise for the reader  
                    val xDiff: Int = calculateDistanceX(ev)  
  
                    // Touch slop should be calculated using ViewConfiguration  
                    // constants.  
                    if (xDiff > mTouchSlop) {  
                        // Start scrolling!  
                        mIsScrolling = true  
                        true  
                    } else {  
                        false  
                    }  
                }  
            }  
            ...  
            else -> {  
                // In general, we don't want to intercept touch events. They should be  
                // handled by the child view.  
                false  
            }  
        }  
    }  
  
    override fun onTouchEvent(event: MotionEvent): Boolean {  
        // Here we actually handle the touch event (e.g. if the action is ACTION\_MOVE,  
        // scroll this container).  
        // This method will only be called if the touch event was intercepted in  
        // onInterceptTouchEvent  
        ...  
    }  
}

Si noti che fornisce anche un metodo. Chiama questo metodo quando un figlio non desidera che l'elemento padre e i relativi predecessori intercettno gli eventi di tocco con . [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)[requestDisallowInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#requestDisallowInterceptTouchEvent(boolean))[ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)[onInterceptTouchEvent()](https://developer.android.com/reference/android/view/ViewGroup#onInterceptTouchEvent(android.view.MotionEvent))

### Elaborare ACTION\_OUTSIDE eventi

Se un oggetto riceve un oggetto con un oggetto , l'evento non verrà inviato ai relativi figli per impostazione predefinita. Per elaborare un oggetto con , eseguire l'override per eseguire l'invio all'oggetto appropriato o gestirlo nell'oggetto pertinente, ad esempio . [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup)[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[ACTION\_OUTSIDE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_OUTSIDE)[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[ACTION\_OUTSIDE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_OUTSIDE)[dispatchTouchEvent(MotionEvent event)](https://developer.android.com/reference/android/view/ViewGroup#dispatchTouchEvent(android.view.MotionEvent))[View](https://developer.android.com/reference/android/view/View)[Window.Callback](https://developer.android.com/reference/android/view/Window.Callback)[Activity](https://developer.android.com/reference/android/app/Activity)

## Utilizzare le costanti ViewConfiguration

Il frammento precedente utilizza la corrente per inizializzare una variabile denominata . Puoi utilizzare la classe per accedere a distanze, velocità e tempi comuni utilizzati dal sistema Android.[ViewConfiguration](https://developer.android.com/reference/android/view/ViewConfiguration)mTouchSlop[ViewConfiguration](https://developer.android.com/reference/android/view/ViewConfiguration)

"Tocco slop" si riferisce alla distanza in pixel che il tocco di un utente può vagare prima che il gesto sia interpretato come scorrimento. Lo slop touch viene in genere utilizzato per impedire lo scorrimento accidentale quando l'utente esegue altre operazioni di tocco, ad esempio il tocco di elementi sullo schermo.

Altri due metodi comunemente utilizzati sono e . Questi metodi restituiscono la velocità minima e massima (rispettivamente) per avviare un'avventura, misurata in pixel al secondo. Per esempio:[ViewConfiguration](https://developer.android.com/reference/android/view/ViewConfiguration)[getScaledMinimumFlingVelocity()](https://developer.android.com/reference/android/view/ViewConfiguration#getScaledMinimumFlingVelocity())[getScaledMaximumFlingVelocity()](https://developer.android.com/reference/android/view/ViewConfiguration#getScaledMaximumFlingVelocity())

[KOTLIN](https://developer.android.com/training/gestures/viewgroup#kotlin)

private val vc: ViewConfiguration = ViewConfiguration.get(context)  
private val mSlop: Int = vc.scaledTouchSlop  
private val mMinFlingVelocity: Int = vc.scaledMinimumFlingVelocity  
private val mMaxFlingVelocity: Int = vc.scaledMaximumFlingVelocity  
  
...  
  
MotionEvent.ACTION\_MOVE -> {  
    ...  
    val deltaX: Float = motionEvent.rawX - mDownX  
    if (Math.abs(deltaX) > mSlop) {  
        // A swipe occurred, do something  
    }  
    return false  
}  
  
...  
  
MotionEvent.ACTION\_UP -> {  
    ...  
    if (velocityX in mMinFlingVelocity..mMaxFlingVelocity && velocityY < velocityX) {  
        // The criteria have been satisfied, do something  
    }  
}

## Estendere l'area toccabile di una visualizzazione figlio

Android fornisce la classe per consentire a un genitore di estendere l'area toccabile di una vista figlio oltre i limiti del bambino. Questo è utile quando il bambino deve essere piccolo, ma dovrebbe avere una regione di tocco più grande. È inoltre possibile utilizzare questo approccio per ridurre l'area di tocco del bambino, se necessario.[TouchDelegate](https://developer.android.com/reference/android/view/TouchDelegate)

Nell'esempio seguente, un oggetto è la "visualizzazione del delegato", o o cioè il figlio di cui si estenderà l'area di tocco dell'elemento padre. Ecco il file di layout:[ImageButton](https://developer.android.com/reference/android/widget/ImageButton)

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
     android:id="@+id/parent\_layout"  
     android:layout\_width="match\_parent"  
     android:layout\_height="match\_parent"  
     tools:context=".MainActivity" >  
  
     <ImageButton android:id="@+id/button"  
          android:layout\_width="wrap\_content"  
          android:layout\_height="wrap\_content"  
          android:background="@null"  
          android:src="@drawable/icon" />  
</RelativeLayout>

Lo snippet riportato di seguito fa quanto segue:

* Ottiene la visualizzazione padre e invia un oggetto nel thread dell'interfaccia utente. Ciò garantisce che il genitore dispone i propri figli prima di chiamare il metodo . Il metodo ottiene il rettangolo di hit del bambino (area toccabile) nelle coordinate dell'elemento padre.[Runnable](https://developer.android.com/reference/java/lang/Runnable)[getHitRect()getHitRect()](https://developer.android.com/reference/android/view/View#getHitRect(android.graphics.Rect))
* Trova la visualizzazione figlio e chiama per ottenere i limiti dell'area toccabile del bambino.[ImageButton](https://developer.android.com/reference/android/widget/ImageButton)[getHitRect()](https://developer.android.com/reference/android/view/View#getHitRect(android.graphics.Rect))
* Estende i limiti del rettangolo di hit dell'oggetto .[ImageButton](https://developer.android.com/reference/android/widget/ImageButton)
* Crea un'istanza di un oggetto passando il rettangolo di hit espanso e la visualizzazione figlio come parametri.[TouchDelegate](https://developer.android.com/reference/android/view/TouchDelegate)[ImageButton](https://developer.android.com/reference/android/widget/ImageButton)
* Imposta l'oggetto nella visualizzazione padre, in modo che i tocchi all'interno dei limiti del delegato tocco siano instradati al figlio.[TouchDelegate](https://developer.android.com/reference/android/view/TouchDelegate)

In qualità di delegato tocco per la visualizzazione figlio, la visualizzazione padre riceverà tutti gli eventi di tocco. Se l'evento di tocco si è verificato all'interno del rettangolo di hit del bambino, l'elemento padre passerà l'evento tocco al figlio per la gestione.[ImageButton](https://developer.android.com/reference/android/widget/ImageButton)

[KOTLIN](https://developer.android.com/training/gestures/viewgroup#kotlin)

public class MainActivity : Activity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity\_main)  
  
        // Post in the parent's message queue to make sure the parent  
        // lays out its children before you call getHitRect()  
        findViewById<View>(R.id.parent\_layout).post {  
            // The bounds for the delegate view (an ImageButton  
            // in this example)  
            val delegateArea = Rect()  
            val myButton = findViewById<ImageButton>(R.id.button).apply {  
                isEnabled = true  
                setOnClickListener {  
                    Toast.makeText(  
                            this@MainActivity,  
                            "Touch occurred within ImageButton touch region.",  
                            Toast.LENGTH\_SHORT  
                    ).show()  
                }  
  
                // The hit rectangle for the ImageButton  
                getHitRect(delegateArea)  
            }  
  
            // Extend the touch area of the ImageButton beyond its bounds  
            // on the right and bottom.  
            delegateArea.right += 100  
            delegateArea.bottom += 100  
  
            // Sets the TouchDelegate on the parent view, such that touches  
            // within the touch delegate bounds are routed to the child.  
            (myButton.parent as? View)?.apply {  
                // Instantiate a TouchDelegate.  
                // "delegateArea" is the bounds in local coordinates of  
                // the containing view to be mapped to the delegate view.  
                // "myButton" is the child view that should receive motion  
                // events.  
                touchDelegate = TouchDelegate(delegateArea, myButton)  
            }  
        }  
    }  
}

# Gestire l'input da tastiera

Il sistema Android mostra una tastiera su schermo, nota come metodo di input soft, quandoun campo di testo nell'interfaccia utente riceve lo stato attivo. Per offrire la migliore esperienza utente, è possibile specificare le caratteristiche sul tipo di input previsto (ad esempio se si tratta di un numero di telefono o di un indirizzo di posta elettronica) e sul comportamento del metodo di input, ad esempio se esegue la correzione automatica per gli errori di ortografia.

Oltre ai metodi di input su schermo, Android supporta anche le tastiere hardware, quindi è importante che l'app ottimizzi la propria esperienza utente per l'interazione che potrebbe verificarsi tramite una tastiera collegata.

Questi argomenti e altro sono discussi nelle seguenti lezioni.

## Lezioni

[**Specificare il tipo di metodo di input**](https://developer.android.com/training/keyboard-input/style)

Informazioni su come visualizzare determinati metodi di input soft, ad esempio quelli progettati per numeri di telefono, indirizzi Web o altri formati. Viene inoltre illustrato come specificare caratteristiche quali il comportamento dei suggerimenti ortografici e i pulsanti di azione, ad **esempio Fatto** o **Avanti**.

# Specificare il tipo di metodo di input

Ogni campo di testo prevede un determinato tipo di input di testo, ad esempio un indirizzo e-mail, un numero di telefono o solo testo normale. Pertanto è importante specificare il tipo di input per ogni campo di testo nell'app in modo che il sistema 2000 il metodo di input soft appropriato (ad esempio una tastiera su schermo).

Oltre al tipo di pulsanti disponibili con un metodo di input, è necessario specificare comportamenti quali se il metodo di input fornisce suggerimenti ortografici, capitalizza nuove frasi e sostituisce il pulsante di ritorno a capo con un pulsante di azione, ad esempio **Fatto** o **Successivo**. Questa lezione mostra come specificare queste caratteristiche.

## Specificare il tipo di tastiera

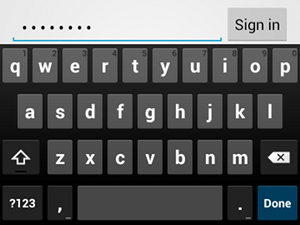
È sempre necessario dichiarare il metodo di input per i campi di testo aggiungendo [l'attributo android:inputType](https://developer.android.com/reference/android/widget/TextView#attr_android:inputType)all'elemento.[<EditText>](https://developer.android.com/reference/android/widget/EditText)



**Figura 1.** Tipo di input.phone

Ad esempio, se si desidera un metodo di input per l'immissione di un numero di telefono, utilizzare il valore:"phone"

<EditText  
    android:id="@+id/phone"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="wrap\_content"  
    android:hint="@string/phone\_hint"  
    android:inputType="phone" />



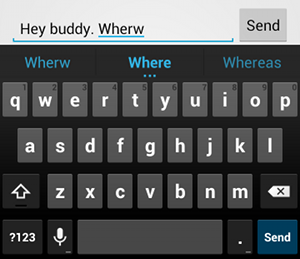
**Figura 2.** Tipo di input.textPassword

Oppure, se il campo di testo è per una password, usa il valore in modo che il campo di testo nasconda l'input dell'utente:"textPassword"

<EditText  
    android:id="@+id/password"  
    android:hint="@string/password\_hint"  
    android:inputType="textPassword"  
    ... />

Esistono diversi valori possibili documentati con [l'attributo android:inputType](https://developer.android.com/reference/android/widget/TextView#attr_android:inputType)e alcuni dei valori possono essere combinati per specificare l'aspetto del metodo di input e comportamenti aggiuntivi.

## Abilitare suggerimenti ortografici e altri comportamenti



**Figura 3.** L'aggiunta fornisce la correzione automatica per gli errori di ortografia.textAutoCorrect

[L'attributo android:inputType](https://developer.android.com/reference/android/widget/TextView#attr_android:inputType)consente di specificare vari comportamenti per il metodo di input. Ancora più importante, se il campo di testo è destinato all'input di testo di base (ad esempio per un messaggio di testo), è necessario abilitare la correzione ortografica automatica con il valore."textAutoCorrect"

È possibile combinare comportamenti e stili di metodo di input diversi con [l'attributo android:inputType.](https://developer.android.com/reference/android/widget/TextView#attr_android:inputType)Ad esempio, ecco come creare un campo di testo che capitalizzi la prima parola di una frase e correggi automaticamente gli errori di ortografia:

<EditText  
    android:id="@+id/message"  
    android:layout\_width="wrap\_content"  
    android:layout\_height="wrap\_content"  
    android:inputType=  
        "textCapSentences|textAutoCorrect"  
    ... />

## Specificare l'azione del metodo di input

La maggior parte dei metodi di input soft fornisce un pulsante di azione utente nell'angolo inferiore appropriato per il campo di testo corrente. Per impostazione predefinita, il sistema utilizza questo pulsante per **un'azione Avanti** o **Fatto,** a meno che il campo di testo non consenta il testo su più linee, ad esempio con , nel qual caso il pulsante di azione è un ritorno a capo. Tuttavia, è possibile specificare azioni aggiuntive che potrebbero essere più appropriate per il campo di testo, ad esempio **Invia** o **Vai**.android:inputType="textMultiLine"

Per specificare il pulsante di azione della tastiera, utilizzare [l'attributo android:imeOptions](https://developer.android.com/reference/android/widget/TextView#attr_android:imeOptions)con un valore di azione quale o . Per esempio:"actionSend""actionSearch"



**Figura 4.** Il pulsante Invia viene visualizzato quando si dichiara .android:imeOptions="actionSend"

<EditText  
    android:id="@+id/search"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="wrap\_content"  
    android:hint="@string/search\_hint"  
    android:inputType="text"  
    android:imeOptions="actionSend" />

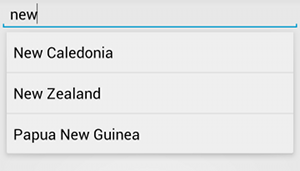
È quindi possibile ascoltare le pressioni sul pulsante di azione definendo un oggetto per l'elemento. Nel listener rispondere all'ID azione IME appropriato definito nella classe , ad esempio . Per esempio:[TextView.OnEditorActionListener](https://developer.android.com/reference/android/widget/TextView.OnEditorActionListener)[EditText](https://developer.android.com/reference/android/widget/EditText)[EditorInfo](https://developer.android.com/reference/android/view/inputmethod/EditorInfo)[IME\_ACTION\_SEND](https://developer.android.com/reference/android/view/inputmethod/EditorInfo#IME_ACTION_SEND)

[KOTLIN](https://developer.android.com/training/keyboard-input/style#kotlin)

findViewById<EditText>(R.id.search).setOnEditorActionListener { v, actionId, event ->  
    return@setOnEditorActionListener when (actionId) {  
        EditorInfo.IME\_ACTION\_SEND -> {  
            sendMessage()  
            true  
        }  
        else -> false  
    }  
}

## Fornire suggerimenti per il completamento automatico

Se si desidera fornire suggerimenti agli utenti mentre digitano, è possibile utilizzare una sottoclasse denominata . Per implementare il completamento automatico, è necessario specificare un oggetto che fornisca i suggerimenti di testo. Sono disponibili diversi tipi di adattatori, a seconda della posizione da cui provengono i dati, ad esempio da un database o da una matrice. [EditText](https://developer.android.com/reference/android/widget/EditText)[AutoCompleteTextView](https://developer.android.com/reference/android/widget/AutoCompleteTextView)[Adapter](https://developer.android.com/reference/android/widget/Adapter)



**Figura 5.** Esempio di con suggerimenti di testo. [AutoCompleteTextView](https://developer.android.com/reference/android/widget/AutoCompleteTextView)

Nella procedura seguente viene descritto come impostare un oggetto che fornisce suggerimenti da una matrice, utilizzando : [AutoCompleteTextView](https://developer.android.com/reference/android/widget/AutoCompleteTextView)[ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter)

1. Aggiungere l'al layout. Ecco un layout con solo il campo di testo: [AutoCompleteTextView](https://developer.android.com/reference/android/widget/AutoCompleteTextView)

<?xml version="1.0" encoding="utf-8"?>  
<AutoCompleteTextView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/autocomplete\_country"  
    android:layout\_width="fill\_parent"  
    android:layout\_height="wrap\_content" />

1. Definire la matrice che contiene tutti i suggerimenti di testo. Ad esempio, ecco una matrice di nomi di paese definiti in un file di risorse XML (): res/values/strings.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string-array name="countries\_array">  
        <item>Afghanistan</item>  
        <item>Albania</item>  
        <item>Algeria</item>  
        <item>American Samoa</item>  
        <item>Andorra</item>  
        <item>Angola</item>  
        <item>Anguilla</item>  
        <item>Antarctica</item>  
        ...  
    </string-array>  
</resources>

1. In o utilizzare il codice seguente per specificare la scheda che fornisce i suggerimenti: [Activity](https://developer.android.com/reference/android/app/Activity)[Fragment](https://developer.android.com/reference/android/app/Fragment)

[KOTLIN](https://developer.android.com/training/keyboard-input/style#kotlin)

// Get a reference to the AutoCompleteTextView in the layout  
val textView = findViewById(R.id.autocomplete\_country) as AutoCompleteTextView  
// Get the string array  
val countries: Array<out String> = resources.getStringArray(R.array.countries\_array)  
// Create the adapter and set it to the AutoCompleteTextView  
ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, countries).also { adapter ->  
    textView.setAdapter(adapter)  
}

Qui viene inizializzato un nuovo elemento per associare ogni elemento della matrice di stringhe a un oggetto presente nel layout (si tratta di un layout fornito da Android che fornisce un aspetto standard per il testo in un elenco). [ArrayAdapter](https://developer.android.com/reference/android/widget/ArrayAdapter)countries\_array[TextView](https://developer.android.com/reference/android/widget/TextView)simple\_list\_item\_1

1. Assegnare l'adattatore a chiamando . [AutoCompleteTextView](https://developer.android.com/reference/android/widget/AutoCompleteTextView)[setAdapter()](https://developer.android.com/reference/android/widget/AutoCompleteTextView#setAdapter(T))

[**Gestire la visibilità del metodo di input**](https://developer.android.com/training/keyboard-input/visibility)

Informazioni su come specificare quando visualizzare il metodo di input soft e come il layout deve adattarsi allo spazio ridotto dello schermo.

# Gestire la visibilità del metodo di input

Quando lo stato attivo dell'input si sposta all'interno o all'uscita da un campo di testo modificabile, Android mostra o nasconde il metodo di input (ad esempio la tastiera su schermo) a seconda dei casi. Il sistema prende anche decisioni su come appaiono l'interfaccia utente e il campo di testo sopra il metodo di input. Ad esempio, quando lo spazio verticale sullo schermo è vincolato, il campo di testo potrebbe riempire tutto lo spazio sopra il metodo di input. Per la maggior parte delle app, questi comportamenti predefiniti sono tutto ciò che serve.

In alcuni casi, tuttavia, potresti voler controllare più direttamente la visibilità del metodo di input e specificare come vuoi che il tuo layout venga visualizzato quando il metodo di input è visibile. Questa lezione spiega come controllare e rispondere alla visibilità del metodo di input.

## Mostrare il metodo di input all'avvio dell'attività

Sebbene Android dia fuoco al primo campo di testo nel layout all'avvio dell'attività, non mostra il metodo di input. Questo comportamento è appropriato perché l'immissione di testo potrebbe non essere l'attività principale nell'attività. Tuttavia, se l'immissione di testo è effettivamente l'attività principale (ad esempio in una schermata di accesso), probabilmente si desidera che il metodo di input venga visualizzato per impostazione predefinita.

Per visualizzare il metodo di input all'avvio dell'attività, aggiungere [l'attributo android:windowSoftInputMode](https://developer.android.com/guide/topics/manifest/activity-element#wsoft)all'elemento con il valore . Per esempio:<activity>"stateVisible"

<application ... >  
    <activity  
        android:windowSoftInputMode="stateVisible" ... >  
        ...  
    </activity>  
    ...  
</application>

**Nota:** Se il dispositivo dell'utente dispone di una tastiera hardware collegata, il metodo di input soft non viene visualizzato.

## Visualizzare il metodo di input su richiesta

Se nel ciclo di vita dell'attività è presente un metodo in cui si desidera assicurarsi che il metodo di input sia visibile, è possibile utilizzare per mostrarlo.[InputMethodManager](https://developer.android.com/reference/android/view/inputmethod/InputMethodManager)

Ad esempio, il metodo seguente accetta un oggetto in cui l'utente deve digitare qualcosa, chiama per dargli lo stato attivo, quindi per aprire il metodo di input:[View](https://developer.android.com/reference/android/view/View)[requestFocus()](https://developer.android.com/reference/android/view/View#requestFocus())[showSoftInput()](https://developer.android.com/reference/android/view/inputmethod/InputMethodManager#showSoftInput(android.view.View,%20int))

[KOTLIN](https://developer.android.com/training/keyboard-input/visibility#kotlin)

fun showSoftKeyboard(view: View) {  
    if (view.requestFocus()) {  
        val imm = getSystemService(Context.INPUT\_METHOD\_SERVICE) as InputMethodManager  
        imm.showSoftInput(view, InputMethodManager.SHOW\_IMPLICIT)  
    }  
}

**Nota:** Una volta che il metodo di input è visibile, non dovresti nasconderlo a livello di programmazione. Il sistema nasconde il metodo di input quando l'utente termina l'attività nel campo di testo o l'utente può nasconderla con un controllo di sistema, ad esempio con il pulsante Indietro.

## Specificare la modalità di risposta dell'interfaccia utente

Quando il metodo di input viene visualizzato sullo schermo, riduce la quantità di spazio disponibile per l'interfaccia utente dell'app. Il sistema prende una decisione su come regolare la parte visibile dell'interfaccia utente, ma potrebbe non farlo bene. Per garantire il comportamento migliore per la tua app, devi specificare come vuoi che il sistema mostrerà l'interfaccia utente nello spazio rimanente.

Per dichiarare il trattamento preferito in un'attività, utilizzare [l'attributo android:windowSoftInputMode](https://developer.android.com/guide/topics/manifest/activity-element#wsoft)nell'elemento del manifesto con uno dei valori "adjust".<activity>

Ad esempio, per assicurarsi che il sistema ridimensiona il layout nello spazio disponibile, in modo da garantire che tutto il contenuto del layout sia accessibile (anche se probabilmente richiede lo scorrimento), utilizzare :"adjustResize"

<application ... >  
    <activity  
        android:windowSoftInputMode="adjustResize" ... >  
        ...  
    </activity>  
    ...  
</application>

È possibile combinare la specifica di regolazione con la specifica [di visibilità iniziale del metodo di](https://developer.android.com/training/keyboard-input/visibility#ShowOnStart) input dall'alto:

    <activity  
        android:windowSoftInputMode="stateVisible|adjustResize" ... >  
        ...  
    </activity>

Specificare è importante se l'interfaccia utente include controlli a cui l'utente potrebbe dover accedere immediatamente dopo o durante l'esecuzione dell'input di testo. Ad esempio, se usi un layout relativo per posizionare una barra dei pulsanti nella parte inferiore dello schermo, usando il ridimensionamento del layout in modo che la barra dei pulsanti venga visualizzata sopra il metodo di input."adjustResize""adjustResize"

[**Supportare l'esplorazione della tastiera**](https://developer.android.com/training/keyboard-input/navigation)

Scopri come verificare che gli utenti possano esplorare l'app usando una tastiera e come apportare le modifiche necessarie all'ordine di navigazione.

# Supportare l'esplorazione della tastiera

Oltre ai metodi di input soft (come le tastiere su schermo), Android supporta le tastiere fisiche collegate al dispositivo. Una tastiera offre non solo una comoda modalità per l'input di testo, ma offre anche un modo per gli utenti di navigare e interagire con la tua app. Sebbene la maggior parte dei dispositivi portatili come i telefoni utilizzi il tocco come modalità principale di interazione, i tablet e dispositivi simili stanno crescendo in popolarità e a molti utenti piace collegare accessori per tastiera.

Poiché più dispositivi Android offrono questo tipo di esperienza, è importante ottimizzare l'app per supportare l'interazione tramite tastiera. Questa lezione descrive come supportare meglio la navigazione con una tastiera.

**Nota:** Il supporto della navigazione direzionale nell'applicazione è anche importante per garantire che l'applicazione [sia accessibile](https://developer.android.com/guide/topics/ui/accessibility/apps) agli utenti che non navigano utilizzando segnali visivi. Il supporto completo della navigazione direzionale nell'applicazione può anche aiutarti ad automatizzare [i test dell'interfaccia](https://developer.android.com/tools/testing/testing_ui) utente con strumenti [come uiautomator](https://developer.android.com/tools/help/uiautomator).

Fare riferimento alle risorse correlate seguenti:

* [Implementare l'accessibilità](https://developer.android.com/training/accessibility)

## Testare l'app

È possibile che gli utenti possano già navigare nella tua app usando una tastiera, perché il sistema Android abilita la maggior parte dei comportamenti necessari per impostazione predefinita.

Tutti i widget interattivi forniti dal framework Android (ad esempio e ) sono focalizzabili. Ciò significa che gli utenti possono navigare con dispositivi di controllo come un D-pad o una tastiera e ogni widget si illumina o cambia in altro modo il suo aspetto quando ottiene lo stato attivo per l'input.[Button](https://developer.android.com/reference/android/widget/Button)[EditText](https://developer.android.com/reference/android/widget/EditText)

Per testare l'app:

1. Installa la tua app in un dispositivo che offre una tastiera hardware.

Se non si dispone di un dispositivo hardware con una tastiera, collegare una tastiera Bluetooth o una tastiera USB (anche se non tutti i dispositivi supportano accessori USB).

Puoi anche usare l'emulatore Android:

* 1. In AVD Manager fare clic su **Nuovo dispositivo oppure selezionare** un profilo esistente e fare clic su **Clona**.
  2. Nella finestra visualizzata verificare che tastiera **e** **DPad siano** abilitati.

1. Per testare l'app, usa solo tab per spostarsi nell'interfaccia utente, assicurandoti che ogni controllo dell'interfaccia utente venga messa a fuoco come previsto.

Cerca tutti i casi in cui lo stato attivo si muove in un modo che non ti aspetti.

1. Inizia dall'inizio dell'app e usa invece i controlli di direzione (tasti di direzione sulla tastiera) per esplorare l'app.

Da ogni elemento attivo dell'interfaccia utente, premi Su, Giù, Sinistra e Destra.

Cerca tutti i casi in cui lo stato attivo si muove in un modo che non ti aspetti.

Se si verificano casi in cui lo spostamento con tab o controlli direzione non fa quello previsto, specificare dove deve andare lo stato attivo nel layout, come discusso nelle sezioni seguenti.

## Gestire l'esplorazione delle schede

Quando un utente naviga nell'app usando il tasto TAB della tastiera, il sistema passa lo stato attivo di input tra gli elementi in base all'ordine in cui vengono visualizzati nel layout. Se si utilizza un layout relativo, ad esempio, e l'ordine degli elementi sullo schermo è diverso dall'ordine nel file, potrebbe essere necessario specificare manualmente l'ordine di messa a fuoco.

Ad esempio, nel layout seguente, due pulsanti sono allineati sul lato destro e un campo di testo è allineato a sinistra del secondo pulsante. Per passare lo stato attivo dal primo pulsante al campo di testo, quindi al secondo pulsante, il layout deve definire in modo esplicito l'ordine di messa a fuoco per ciascuno degli elementi messa a fuoco con [l'attributo android:nextFocusForward:](https://developer.android.com/reference/android/view/View#attr_android:nextFocusForward)

<RelativeLayout ...>  
    <Button  
        android:id="@+id/button1"  
        android:layout\_alignParentTop="true"  
        android:layout\_alignParentRight="true"  
        android:nextFocusForward="@+id/editText1"  
        ... />  
    <Button  
        android:id="@+id/button2"  
        android:layout\_below="@id/button1"  
        android:nextFocusForward="@+id/button1"  
        ... />  
    <EditText  
        android:id="@id/editText1"  
        android:layout\_alignBottom="@+id/button2"  
        android:layout\_toLeftOf="@id/button2"  
        android:nextFocusForward="@+id/button2"  
        ...  />  
    ...  
</RelativeLayout>

Ora invece di inviare lo stato attivo da a allora , lo stato attivo si sposta in modo appropriato in base all'aspetto sullo schermo: da a allora .button1button2editText1button1editText1button2

## Gestire la navigazione direzionale

Gli utenti possono anche esplorare l'app usando i tasti di direzione su una tastiera (il comportamento è lo stesso di quando si naviga con un D-pad o un trackball). Il sistema fornisce una migliore ipotesi su quale vista dovrebbe essere messa a fuoco in una determinata direzione in base al layout delle viste sullo schermo. A volte, tuttavia, il sistema potrebbe indovinare sbagliato.

Se il sistema non passa lo stato attivo alla vista appropriata quando si naviga in una determinata direzione, specificare quale vista deve ricevere lo stato attivo con i seguenti attributi:

* [android:nextFocusUp](https://developer.android.com/reference/android/view/View#attr_android:nextFocusUp)
* [android:nextFocusDown](https://developer.android.com/reference/android/view/View#attr_android:nextFocusDown)
* [android:nextFocusLeft](https://developer.android.com/reference/android/view/View#attr_android:nextFocusLeft)
* [android:nextFocusRight](https://developer.android.com/reference/android/view/View#attr_android:nextFocusRight)

Ogni attributo designa la visualizzazione successiva per ricevere lo stato attivo quando l'utente si sposta in quella direzione, come specificato dall'ID visualizzazione. Per esempio:

<Button  
    android:id="@+id/button1"  
    android:nextFocusRight="@+id/button2"  
    android:nextFocusDown="@+id/editText1"  
    ... />  
<Button  
    android:id="@id/button2"  
    android:nextFocusLeft="@id/button1"  
    android:nextFocusDown="@id/editText1"  
    ... />  
<EditText  
    android:id="@id/editText1"  
    android:nextFocusUp="@id/button1"  
    ...  />

[**Gestire le azioni della tastiera**](https://developer.android.com/training/keyboard-input/commands)

Informazioni su come rispondere direttamente all'input della tastiera per le azioni dell'utente.

# Gestire le azioni della tastiera

Quando l'utente dà fuoco a una visualizzazione di testo modificabile, ad esempio un elemento, e l'utente ha una tastiera hardware collegata, tutto l'input viene gestito dal sistema. Se, tuttavia, si desidera intercettare o gestire direttamente l'input della tastiera da soli, è possibile farlo implementando metodi di callback dall'interfaccia, ad esempio e .[EditText](https://developer.android.com/reference/android/widget/EditText)[KeyEvent.Callback](https://developer.android.com/reference/android/view/KeyEvent.Callback)[onKeyDown()](https://developer.android.com/reference/android/view/KeyEvent.Callback#onKeyDown(int,%20android.view.KeyEvent))[onKeyMultiple()](https://developer.android.com/reference/android/view/KeyEvent.Callback#onKeyMultiple(int,%20int,%20android.view.KeyEvent))

Sia la classe e implementa l'interfaccia, pertanto in genere è necessario eseguire l'override dei metodi di callback nell'estensione di queste classi in base alle esigenze.[Activity](https://developer.android.com/reference/android/app/Activity)[View](https://developer.android.com/reference/android/view/View)[KeyEvent.Callback](https://developer.android.com/reference/android/view/KeyEvent.Callback)

**Nota:** Quando si gestiscono gli eventi della tastiera con la classe e le API correlate, è necessario aspettarsi che tali eventi della tastiera provenano solo da una tastiera hardware. Non dovresti mai fare affidamento sulla ricezione di eventi chiave per qualsiasi tasto su un metodo di input soft (una tastiera su schermo).[**KeyEvent**](https://developer.android.com/reference/android/view/KeyEvent)

## Gestire eventi a chiave singola

Per gestire un singolo tasto premere, implementare o, a caso, . Di solito, dovresti usare se vuoi essere sicuro di ricevere un solo evento. Se l'utente preme e tiene premuto il pulsante, viene chiamato più volte.[onKeyDown()](https://developer.android.com/reference/android/app/Activity#onKeyDown(int,%20android.view.KeyEvent))[onKeyUp()onKeyUp()](https://developer.android.com/reference/android/app/Activity#onKeyUp(int,%20android.view.KeyEvent))[onKeyDown()](https://developer.android.com/reference/android/app/Activity#onKeyDown(int,%20android.view.KeyEvent))

Ad esempio, questa implementazione risponde ad alcuni tasti della tastiera per controllare un gioco:

[KOTLIN](https://developer.android.com/training/keyboard-input/commands#kotlin)

override fun onKeyUp(keyCode: Int, event: KeyEvent): Boolean {  
    return when (keyCode) {  
        KeyEvent.KEYCODE\_D -> {  
            moveShip(MOVE\_LEFT)  
            true  
        }  
        KeyEvent.KEYCODE\_F -> {  
            moveShip(MOVE\_RIGHT)  
            true  
        }  
        KeyEvent.KEYCODE\_J -> {  
            fireMachineGun()  
            true  
        }  
        KeyEvent.KEYCODE\_K -> {  
            fireMissile()  
            true  
        }  
        else -> super.onKeyUp(keyCode, event)  
    }  
}

## Tasti di modifica handle

Per rispondere agli eventi del tasto di modifica, ad esempio quando un tasto viene combinato con MAIUSC o CTRL, è possibile eseguire una query sull'oggetto passato al metodo di callback. Diversi metodi forniscono informazioni sui tasti di modifica quali e . Tuttavia, la soluzione più semplice è verificare se il tasto di modifica esatto a cui si interessa viene premuto con metodi quali e . [KeyEvent](https://developer.android.com/reference/android/view/KeyEvent)[getModifiers()](https://developer.android.com/reference/android/view/KeyEvent#getModifiers())[getMetaState()](https://developer.android.com/reference/android/view/KeyEvent#getMetaState())[isShiftPressed()](https://developer.android.com/reference/android/view/KeyEvent#isShiftPressed())[isCtrlPressed()](https://developer.android.com/reference/android/view/KeyEvent#isCtrlPressed())

Ad esempio, ecco di nuovo l'implementazione, con una gestione aggiuntiva per quando il tasto MAIUSC viene tenuto premuto con una delle chiavi:[onKeyUp()](https://developer.android.com/reference/android/app/Activity#onKeyUp(int,%20android.view.KeyEvent))

[KOTLIN](https://developer.android.com/training/keyboard-input/commands#kotlin)

override fun onKeyUp(keyCode: Int, event: KeyEvent): Boolean {  
    return when (keyCode) {  
        ...  
        KeyEvent.KEYCODE\_J -> {  
            if (event.isShiftPressed) {  
                fireLaser()  
            } else {  
                fireMachineGun()  
            }  
            true  
        }  
        KeyEvent.KEYCODE\_K -> {  
            if (event.isShiftPressed) {  
                fireSeekingMissle()  
            } else {  
                fireMissile()  
            }  
            true  
        }  
        else -> super.onKeyUp(keyCode, event)  
    }  
}

# Supportare i controller di gioco

Puoi migliorare notevolmente l'esperienza utente nel tuo gioco consentendo ai giocatori di utilizzare i loro controller di gioco preferiti. Il framework Android fornisce API per rilevare ed elaborare l'input dell'utente dai controller di gioco.

Questa classe mostra come far funzionare il tuo gioco in modo coerente con i controller di gioco su diversi livelli di API Android (livello API 9 e versioni precedenti) e come migliorare l'esperienza di gioco per i giocatori supportando più controller contemporaneamente nella tua app.

## Lezioni

[**Gestire le azioni del controller**](https://developer.android.com/training/game-controllers/controller-input)

Scopri come gestire l'input dell'utente da elementi di input comuni sui controller di gioco, inclusi pulsanti del pad direzionale (D-pad), pulsanti del gamepad e joystick.

# Gestire le azioni del controller

A livello di sistema, Android segnala i codici evento di input dai controller di gioco come codici chiave Android e valori dell'asse. Nel tuo gioco, puoi ricevere questi codici e valori e convertirli in specifiche azioni di gioco.

Quando i giocatori si connettono fisicamente o accoppiano in modalità wireless un controller di gioco ai propri dispositivi basati su Android, il sistema rileva automaticamente il controller come dispositivo di input e inizia a segnalare i suoi eventi di input. Il gioco può ricevere questi eventi di input implementando i seguenti metodi di callback nell'attivo o nello stato attivo (è necessario implementare i callback per o , ma non per entrambi): [Activity](https://developer.android.com/reference/android/app/Activity)[View](https://developer.android.com/reference/android/view/View)[Activity](https://developer.android.com/reference/android/app/Activity)[View](https://developer.android.com/reference/android/view/View)

* Da: [Activity](https://developer.android.com/reference/android/app/Activity)
  + [dispatchGenericMotionEvent(android.view. MotionEvent)](https://developer.android.com/reference/android/app/Activity#dispatchGenericMotionEvent(android.view.MotionEvent))

Chiamato per elaborare eventi di movimento generici come i movimenti del joystick.

* + [dispatchKeyEvent(android.view.KeyEvent)](https://developer.android.com/reference/android/app/Activity#dispatchKeyEvent(android.view.KeyEvent))

Chiamato per elaborare eventi chiave come una pressione o un rilascio di un gamepad o di un pulsante D-pad.

* Da: [View](https://developer.android.com/reference/android/view/View)
  + [onGenericMotionEvent(android.view.MotionEvent)](https://developer.android.com/reference/android/view/View#onGenericMotionEvent(android.view.MotionEvent))

Chiamato per elaborare eventi di movimento generici come i movimenti del joystick.

* + [onKeyDown(int, android.view.KeyEvent)](https://developer.android.com/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent))

Chiamato per elaborare una pressione di un tasto fisico come un gamepad o un pulsante D-pad.

* + [onKeyUp(int, android.view.KeyEvent)](https://developer.android.com/reference/android/view/View#onKeyUp(int,%20android.view.KeyEvent))

Chiamato per elaborare il rilascio di una chiave fisica come un gamepad o un pulsante D-pad.

L'approccio consigliato è quello di acquisire gli eventi dall'oggetto specifico con cui l'utente interagisce. Esaminare gli oggetti seguenti forniti dai callback per ottenere informazioni sul tipo di evento di input ricevuto:[View](https://developer.android.com/reference/android/view/View)

[KeyEvent](https://developer.android.com/reference/android/view/KeyEvent)

Oggetto che descrive gli eventi del pulsante del pad direzionale (D-pad) e del gamepad. Gli eventi chiave sono accompagnati da un codice chiave che indica il pulsante specifico attivato, ad esempio o . È possibile ottenere il codice chiave chiamando o da callback di eventi chiave quali . [DPAD\_DOWN](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_DOWN)[BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)[getKeyCode()](https://developer.android.com/reference/android/view/KeyEvent#getKeyCode())[onKeyDown()](https://developer.android.com/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent))

[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

Oggetto che descrive l'input dei movimenti del joystick e del grilletto della spalla. Gli eventi di movimento sono accompagnati da un codice di azione e da un insieme di valori dell'asse. Il codice azione specifica la modifica dello stato che si è verificata, ad esempio un joystick da spostare. I valori dell'asse descrivono la posizione e altre proprietà di movimento per un controllo fisico specifico, ad esempio o . È possibile ottenere il codice di azione chiamando e il valore dell'asse chiamando . [AXIS\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_X)[AXIS\_RTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RTRIGGER)[getAction()](https://developer.android.com/reference/android/view/MotionEvent#getAction())[getAxisValue()](https://developer.android.com/reference/android/view/MotionEvent#getAxisValue(int))

Questa lezione si concentra su come gestire l'input dai tipi più comuni di controlli fisici (pulsanti del gamepad, pad direzionali e joystick) in una schermata di gioco implementando i metodi di callback e l'elaborazione e gli oggetti sopra menzionati.[View](https://developer.android.com/reference/android/view/View)[KeyEvent](https://developer.android.com/reference/android/view/KeyEvent)[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

## Verificare che un controller di gioco sia connesso

Quando si segnalano eventi di input, Android non distingue tra eventi provenienti da un dispositivo controller non di gioco ed eventi provenienti da un controller di gioco. Ad esempio, un'azione touch screen genera un evento che rappresenta la coordinata X della superficie di tocco, ma un joystick genera un evento che rappresenta la posizione X del joystick. Se il gioco si preoccupa di gestire l'input del controller di gioco, devi prima verificare che l'evento di input provenga da un tipo di origine pertinente.[AXIS\_XAXIS\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_X)

Per verificare che un dispositivo di input connesso sia un controller di gioco, chiamare per ottenere un campo di bit combinato di tipi di origine di input supportati su tale dispositivo. È quindi possibile verificare se sono impostati i seguenti campi:[getSources()](https://developer.android.com/reference/android/view/InputDevice#getSources())

* Un tipo di origine di indica che il dispositivo di input dispone di pulsanti gamepad, ad esempio . Si noti che questo tipo di origine non indica rigorosamente se il controller di gioco ha pulsanti D-pad, anche se la maggior parte dei gamepad in genere ha controlli direzionali.[SOURCE\_GAMEPAD](https://developer.android.com/reference/android/view/InputDevice#SOURCE_GAMEPAD)[BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)
* Un tipo di origine di indica che il dispositivo di input dispone di pulsanti D-pad, ad esempio .[SOURCE\_DPAD](https://developer.android.com/reference/android/view/InputDevice#SOURCE_DPAD)[DPAD\_UP](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_UP)
* Un tipo di origine di indica che il dispositivo di input dispone di bastoncini di controllo analogici, ad esempio un joystick che registra i movimenti lungo e .[SOURCE\_JOYSTICK](https://developer.android.com/reference/android/view/InputDevice#SOURCE_JOYSTICK)[AXIS\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_X)[AXIS\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Y)

Il frammento di codice seguente mostra un metodo di helper che consente di verificare se i dispositivi di input connessi sono controller di gioco. In tal caso, il metodo recupera gli ID dispositivo per i controller di gioco. Puoi quindi associare ogni ID dispositivo a un giocatore nel tuo gioco ed elaborare separatamente le azioni di gioco per ogni giocatore connesso. Per ulteriori informazioni sul supporto di più controller di gioco connessi contemporaneamente sullo stesso dispositivo Android, vedere [Supportare più controller di gioco](https://developer.android.com/training/game-controllers/multiple-controllers).

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

fun getGameControllerIds(): List<Int> {  
    val gameControllerDeviceIds = mutableListOf<Int>()  
    val deviceIds = InputDevice.getDeviceIds()  
    deviceIds.forEach { deviceId ->  
        InputDevice.getDevice(deviceId).apply {  
  
            // Verify that the device has gamepad buttons, control sticks, or both.  
            if (sources and InputDevice.SOURCE\_GAMEPAD == InputDevice.SOURCE\_GAMEPAD  
                    || sources and InputDevice.SOURCE\_JOYSTICK == InputDevice.SOURCE\_JOYSTICK) {  
                // This device is a game controller. Store its device ID.  
                gameControllerDeviceIds  
                        .takeIf { !it.contains(deviceId) }  
                        ?.add(deviceId)  
            }  
        }  
    }  
    return gameControllerDeviceIds  
}

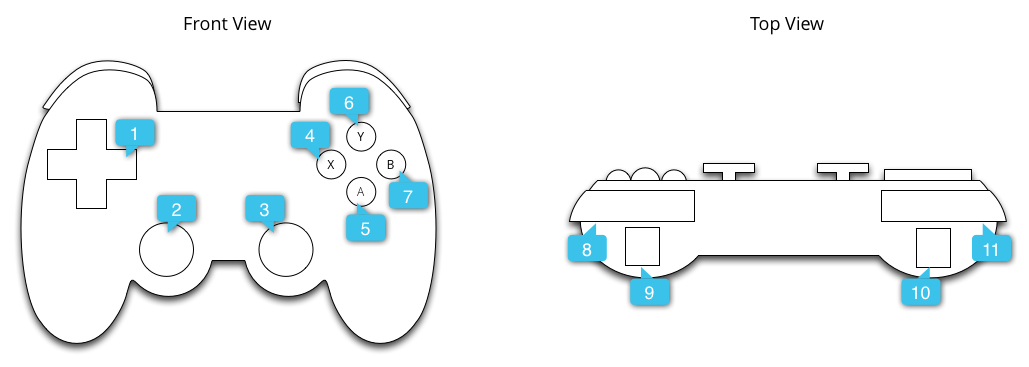
Inoltre, potresti voler verificare la presenza di singole funzionalità di input supportate da un controller di gioco connesso. Questo potrebbe essere utile, ad esempio, se vuoi che il tuo gioco usi solo l'input dal set di controlli fisici che comprende.

Per rilevare se un codice chiave o un codice asse specifico è supportato da un controller di gioco connesso, utilizzare queste tecniche:

* In Android 4.4 (livello API 19) o superiore, è possibile determinare se un codice chiave è supportato su un controller di gioco connesso chiamando .[hasKeys(int...)](https://developer.android.com/reference/android/view/InputDevice" \l "hasKeys(int...))
* In Android 3.1 (livello API 12) o superiore, puoi trovare tutti gli assi disponibili supportati su un controller di gioco connesso chiamando prima . Quindi, su ogni oggetto restituito, chiamare per ottenere l'ID asse.[getMotionRanges()](https://developer.android.com/reference/android/view/InputDevice#getMotionRanges())[InputDevice.MotionRange](https://developer.android.com/reference/android/view/InputDevice.MotionRange)[getAxis()](https://developer.android.com/reference/android/view/InputDevice.MotionRange#getAxis())

## Pulsante Elabora gamepad preme

La figura 1 mostra come Android mappa i codici chiave e i valori dell'asse ai controlli fisici sulla maggior parte dei controller di gioco.



**Figura 1.** Profilo per un controller di gioco generico.

I callout nella figura fanno riferimento a quanto segue:

1. [AXIS\_HAT\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_X), , , , ,[AXIS\_HAT\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_Y)[DPAD\_UP](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_UP)[DPAD\_DOWN](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_DOWN)[DPAD\_LEFT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_LEFT)[DPAD\_RIGHT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_RIGHT)
2. [AXIS\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_X), , [AXIS\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Y)[BUTTON\_THUMBL](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_THUMBL)
3. [AXIS\_Z](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Z), , [AXIS\_RZ](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RZ)[BUTTON\_THUMBR](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_THUMBR)
4. [BUTTON\_X](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_X)
5. [BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)
6. [BUTTON\_Y](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_Y)
7. [BUTTON\_B](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_B)
8. [BUTTON\_R1](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_R1)
9. [AXIS\_RTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RTRIGGER), [AXIS\_THROTTLE](https://developer.android.com/reference/android/view/MotionEvent#AXIS_THROTTLE)
10. [AXIS\_LTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_LTRIGGER), [AXIS\_BRAKE](https://developer.android.com/reference/android/view/MotionEvent#AXIS_BRAKE)
11. [BUTTON\_L1](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_L1)

I codici tasto comuni generati dalle pressioni dei pulsanti del gamepad includono , , e . Alcuni controller di gioco attivano anche il codice tasto quando viene premuto il centro della traversa del D-pad. Il gioco può esaminare il codice chiave chiamando o da callback di eventi chiave come e, se rappresenta un evento rilevante per il gioco, elaborarlo come azione di gioco. La tabella 1 elenca le azioni di gioco consigliate per i pulsanti più comuni del gamepad. [BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)[BUTTON\_B](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_B)[BUTTON\_SELECT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_SELECT)[BUTTON\_START](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_START)[DPAD\_CENTER](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER)[getKeyCode()](https://developer.android.com/reference/android/view/KeyEvent#getKeyCode())[onKeyDown()](https://developer.android.com/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent))

**La tabella 1.** Azioni di gioco consigliate per i pulsanti del gamepad.

|  |  |
| --- | --- |
| **Azione di gioco** | **Codice tasto pulsante** |
| Inizia il gioco nel menu principale o metti in pausa /annulla la pausa durante il gioco | [BUTTON\_START](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_START)\* |
| Menu Visualizza | [BUTTON\_SELECT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_SELECT)\* e [KEYCODE\_MENU](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_MENU)\* |
| Come il comportamento di navigazione indietro di Android descritto nella guida [alla progettazione](https://developer.android.com/design/patterns/navigation) della navigazione. | [KEYCODE\_BACK](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BACK) |
| Tornare a una voce precedente in un menu | [BUTTON\_B](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_B) |
| Confermare la selezione o eseguire l'azione di gioco principale | [BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A) e [DPAD\_CENTER](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER) |

\* Il tuo gioco non deve fare affidamento sulla presenza dei pulsanti Start, Select o Menu.

**Suggerimento:** Prendi in considerazione la possibilità di fornire una schermata di configurazione nel tuo gioco per consentire agli utenti di personalizzare i propri mapping dei controller di gioco per le azioni di gioco.

Il frammento seguente mostra come è possibile eseguire l'override per associare le pressioni dei pulsanti e a un'azione di gioco. [onKeyDown()](https://developer.android.com/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent))[BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)[DPAD\_CENTER](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER)

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

class GameView(...) : View(...) {  
    ...  
  
    override fun onKeyDown(keyCode: Int, event: KeyEvent): Boolean {  
        var handled = false  
        if (event.source and InputDevice.SOURCE\_GAMEPAD == InputDevice.SOURCE\_GAMEPAD) {  
            if (event.repeatCount == 0) {  
                when (keyCode) {  
                    // Handle gamepad and D-pad button presses to navigate the ship  
                    ...  
  
                    else -> {  
                        keyCode.takeIf { isFireKey(it) }?.run {  
                            // Update the ship object to fire lasers  
                            ...  
                            handled = true  
                        }  
                    }  
                }  
            }  
            if (handled) {  
                return true  
            }  
        }  
        return super.onKeyDown(keyCode, event)  
    }  
  
    // Here we treat Button\_A and DPAD\_CENTER as the primary action  
    // keys for the game.  
    private fun isFireKey(keyCode: Int): Boolean =  
            keyCode == KeyEvent.KEYCODE\_DPAD\_CENTER || keyCode == KeyEvent.KEYCODE\_BUTTON\_A  
}

**Nota:** Su Android 4.2 (livello API 17) e versioni inferiori, il sistema tratta come la chiave Posteriore Android per impostazione predefinita. Se la tua app supporta queste versioni di Android, assicurati di trattarla come l'azione di gioco principale. Per determinare la versione corrente di Android SDK nel dispositivo, fare riferimento al valore.[**BUTTON\_ABUTTON\_A**](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)[**Build.VERSION.SDK\_INT**](https://developer.android.com/reference/android/os/Build.VERSION#SDK_INT)

## Ingresso del pad direzionale di processo

Il pad direzionale a 4 direzioni (D-pad) è un controllo fisico comune in molti controller di gioco. Android segnala che D-pad UP e DOWN preme come eventi con un intervallo da -1.0 (fino a) a 1.0 (in basso) e D-pad LEFT o RIGHT preme come eventi con un intervallo da -1.0 (a sinistra) a 1.0 (a destra).[AXIS\_HAT\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_Y)[AXIS\_HAT\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_X)

Alcuni controller segnalano invece le pressioni del D-pad con un codice tasto. Se il gioco si preoccupa delle presse D-pad, è necessario trattare gli eventi dell'asse del cappello e i codici tasto D-pad come gli stessi eventi di input, come raccomandato nella tabella 2.

**La tabella 2.** Azioni di gioco predefinite consigliate per i codici tasto D-pad e i valori dell'asse del cappello.

|  |  |  |
| --- | --- | --- |
| **Azione di gioco** | **Codice tasto D-pad** | **Codice asse hat** |
| promuovere | [KEYCODE\_DPAD\_UP](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_UP) | [AXIS\_HAT\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_Y) (per i valori da 0 a -1,0) |
| retrocedere | [KEYCODE\_DPAD\_DOWN](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_DOWN) | [AXIS\_HAT\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_Y) (per i valori da 0 a 1,0) |
| Spostarsi a sinistra | [KEYCODE\_DPAD\_LEFT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_LEFT) | [AXIS\_HAT\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_X) (per i valori da 0 a -1,0) |
| Spostarsi a destra | [KEYCODE\_DPAD\_RIGHT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_RIGHT) | [AXIS\_HAT\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_X) (per i valori da 0 a 1,0) |

Il frammento di codice seguente mostra una classe helper che consente di controllare l'asse del cappello e i valori del codice chiave da un evento di input per determinare la direzione del D-pad.

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

class Dpad {  
  
    private var directionPressed = -1 // initialized to -1  
  
    fun getDirectionPressed(event: InputEvent): Int {  
        if (!isDpadDevice(event)) {  
            return -1  
        }  
  
        // If the input event is a MotionEvent, check its hat axis values.  
        (event as? MotionEvent)?.apply {  
  
            // Use the hat axis value to find the D-pad direction  
            val xaxis: Float = event.getAxisValue(MotionEvent.AXIS\_HAT\_X)  
            val yaxis: Float = event.getAxisValue(MotionEvent.AXIS\_HAT\_Y)  
  
            directionPressed = when {  
                // Check if the AXIS\_HAT\_X value is -1 or 1, and set the D-pad  
                // LEFT and RIGHT direction accordingly.  
                xaxis.compareTo(-1.0f) == 0 -> Dpad.LEFT  
                xaxis.compareTo(1.0f) == 0 -> Dpad.RIGHT  
                // Check if the AXIS\_HAT\_Y value is -1 or 1, and set the D-pad  
                // UP and DOWN direction accordingly.  
                yaxis.compareTo(-1.0f) == 0 -> Dpad.UP  
                yaxis.compareTo(1.0f) == 0 -> Dpad.DOWN  
                else -> directionPressed  
            }  
        }  
        // If the input event is a KeyEvent, check its key code.  
        (event as? KeyEvent)?.apply {  
  
            // Use the key code to find the D-pad direction.  
            directionPressed = when(event.keyCode) {  
                KeyEvent.KEYCODE\_DPAD\_LEFT -> Dpad.LEFT  
                KeyEvent.KEYCODE\_DPAD\_RIGHT -> Dpad.RIGHT  
                KeyEvent.KEYCODE\_DPAD\_UP -> Dpad.UP  
                KeyEvent.KEYCODE\_DPAD\_DOWN -> Dpad.DOWN  
                KeyEvent.KEYCODE\_DPAD\_CENTER ->  Dpad.CENTER  
                else -> directionPressed  
            }  
        }  
        return directionPressed  
    }  
  
    companion object {  
        internal const val UP = 0  
        internal const val LEFT = 1  
        internal const val RIGHT = 2  
        internal const val DOWN = 3  
        internal const val CENTER = 4  
  
        fun isDpadDevice(event: InputEvent): Boolean =  
            // Check that input comes from a device with directional pads.  
            event.source and InputDevice.SOURCE\_DPAD != InputDevice.SOURCE\_DPAD  
    }  
}

Puoi usare questa classe di supporto nel tuo gioco ovunque tu voglia elaborare l'input del D-pad (ad esempio, nei callback o).[onGenericMotionEvent()](https://developer.android.com/reference/android/view/View#onGenericMotionEvent(android.view.MotionEvent))[onKeyDown()](https://developer.android.com/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent))

Per esempio:

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

private val dpad = Dpad()  
...  
override fun onGenericMotionEvent(event: MotionEvent): Boolean {  
    if (Dpad.isDpadDevice(event)) {  
        when (dpad.getDirectionPressed(event)) {  
            Dpad.LEFT -> {  
                // Do something for LEFT direction press  
                ...  
                return true  
            }  
            Dpad.RIGHT -> {  
                // Do something for RIGHT direction press  
                ...  
                return true  
            }  
            Dpad.UP -> {  
                // Do something for UP direction press  
                ...  
                return true  
            }  
            ...  
        }  
    }  
  
    // Check if this event is from a joystick movement and process accordingly.  
    ...  
}

## Elaborare i movimenti del joystick

Quando i giocatori spostano un joystick sui loro controller di gioco, Android segnala un oggetto che contiene il codice di azione e le posizioni aggiornate degli assi del joystick. Il tuo gioco può utilizzare i dati forniti da per determinare se un movimento del joystick a cui importa è accaduto. [MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[ACTION\_MOVE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

Si noti che gli eventi di movimento del joystick possono mettere insieme più campioni di movimento all'interno di un singolo oggetto. L'oggetto contiene la posizione corrente per ogni asse joystick e più posizioni cronologiche per ogni asse. Quando si segnalano eventi di movimento con codice di azione (ad esempio movimenti del joystick), Android fa in batch i valori dell'asse per l'efficienza. I valori cronologici per un asse sono costituiti dall'insieme di valori distinti precedenti al valore dell'asse corrente e più recenti dei valori riportati in qualsiasi evento di movimento precedente. Vedi il riferimento per i dettagli.[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)[ACTION\_MOVE](https://developer.android.com/reference/android/view/MotionEvent#ACTION_MOVE)[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

Puoi usare le informazioni storiche per eseguire il rendering più accurato del movimento di un oggetto di gioco in base all'input del joystick. Per recuperare i valori correnti e cronologici, chiamare o . È inoltre possibile trovare il numero di punti storici nell'evento joystick chiamando .[getAxisValue()](https://developer.android.com/reference/android/view/MotionEvent#getAxisValue(int))[getHistoricalAxisValue()](https://developer.android.com/reference/android/view/MotionEvent#getHistoricalAxisValue(int,%20int))[getHistorySize()](https://developer.android.com/reference/android/view/MotionEvent#getHistorySize())

Nel frammento seguente viene illustrato come eseguire l'override del callback per elaborare l'input del joystick. È innanzitutto necessario elaborare i valori cronologici per un asse, quindi elaborarne la posizione corrente. [onGenericMotionEvent()](https://developer.android.com/reference/android/view/View#onGenericMotionEvent(android.view.MotionEvent))

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

class GameView(...) : View(...) {  
  
    override fun onGenericMotionEvent(event: MotionEvent): Boolean {  
  
        // Check that the event came from a game controller  
        return if (event.source and InputDevice.SOURCE\_JOYSTICK == InputDevice.SOURCE\_JOYSTICK  
                && event.action == MotionEvent.ACTION\_MOVE) {  
  
            // Process the movements starting from the  
            // earliest historical position in the batch  
            (0 until event.historySize).forEach { i ->  
                // Process the event at historical position i  
                processJoystickInput(event, i)  
            }  
  
            // Process the current movement sample in the batch (position -1)  
            processJoystickInput(event, -1)  
            true  
        } else {  
            super.onGenericMotionEvent(event)  
        }  
    }  
}

Prima di utilizzare l'input del joystick, è necessario determinare se il joystick è centrato, quindi calcolare i movimenti dell'asse di conseguenza. I joystick hanno tipicamente un'area piatta, cioè un intervallo di valori vicino alla coordinata (0,0) in corrispondenza della quale l'asse è considerato centrato. Se il valore dell'asse riportato da Android rientra nell'area piatta, è necessario trattare il controller come a riposo (cioè immobile lungo entrambi gli assi).

Lo snippet riportato di seguito mostra un metodo di aiuto che calcola il movimento lungo ogni asse. Questo helper viene vocato nel metodo descritto di seguito. processJoystickInput()

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

private fun getCenteredAxis(  
        event: MotionEvent,  
        device: InputDevice,  
        axis: Int,  
        historyPos: Int  
): Float {  
    val range: InputDevice.MotionRange? = device.getMotionRange(axis, event.source)  
  
    // A joystick at rest does not always report an absolute position of  
    // (0,0). Use the getFlat() method to determine the range of values  
    // bounding the joystick axis center.  
    range?.apply {  
        val value: Float = if (historyPos < 0) {  
            event.getAxisValue(axis)  
        } else {  
            event.getHistoricalAxisValue(axis, historyPos)  
        }  
  
        // Ignore axis values that are within the 'flat' region of the  
        // joystick axis center.  
        if (Math.abs(value) > flat) {  
            return value  
        }  
    }  
    return 0f  
}

Mettendo tutto insieme, ecco come potresti elaborare i movimenti del joystick nel tuo gioco:

[KOTLIN](https://developer.android.com/training/game-controllers/controller-input#kotlin)

private fun processJoystickInput(event: MotionEvent, historyPos: Int) {  
  
    val inputDevice = event.device  
  
    // Calculate the horizontal distance to move by  
    // using the input value from one of these physical controls:  
    // the left control stick, hat axis, or the right control stick.  
    var x: Float = getCenteredAxis(event, inputDevice, MotionEvent.AXIS\_X, historyPos)  
    if (x == 0f) {  
        x = getCenteredAxis(event, inputDevice, MotionEvent.AXIS\_HAT\_X, historyPos)  
    }  
    if (x == 0f) {  
        x = getCenteredAxis(event, inputDevice, MotionEvent.AXIS\_Z, historyPos)  
    }  
  
    // Calculate the vertical distance to move by  
    // using the input value from one of these physical controls:  
    // the left control stick, hat switch, or the right control stick.  
    var y: Float = getCenteredAxis(event, inputDevice, MotionEvent.AXIS\_Y, historyPos)  
    if (y == 0f) {  
        y = getCenteredAxis(event, inputDevice, MotionEvent.AXIS\_HAT\_Y, historyPos)  
    }  
    if (y == 0f) {  
        y = getCenteredAxis(event, inputDevice, MotionEvent.AXIS\_RZ, historyPos)  
    }  
  
    // Update the ship object based on the new x and y values  
}

Per supportare i controller di gioco che hanno funzionalità più sofisticate oltre a un singolo joystick, segui queste procedure consigliate:

* **Maneggiare bastoncini a doppio controller.** Molti controller di gioco hanno un joystick sinistro e destro. Per il bastone sinistro, Android segnala i movimenti orizzontali come eventi e movimenti verticali come eventi. Per il bastone destro, Android segnala i movimenti orizzontali come eventi e movimenti verticali come eventi. Assicurati di gestire entrambi i bastoncini del controller nel codice.[AXIS\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_X)[AXIS\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Y)[AXIS\_Z](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Z)[AXIS\_RZ](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RZ)
* **Gestire le presse del grilletto a spalla (ma fornire metodi di input alternativi).** Alcuni controller hanno grilletti della spalla sinistra e destra. Se questi trigger sono presenti, Android segnala una pressione del trigger sinistro come evento e una pressione del trigger destro come evento. In Android 4.3 (API level 18), un controller che produce un oggetto segnala anche un valore identico per l'asse. Lo stesso vale per e . Android segnala tutte le presse analogiche con un valore normalizzato da 0,0 (rilasciato) a 1,0 (completamente premuto). Non tutti i controller hanno trigger, quindi considera di consentire ai giocatori di eseguire quelle azioni di gioco con altri pulsanti. [AXIS\_LTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_LTRIGGER)[AXIS\_RTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RTRIGGER)[AXIS\_LTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_LTRIGGER)[AXIS\_BRAKE](https://developer.android.com/reference/android/view/MotionEvent#AXIS_BRAKE)[AXIS\_RTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RTRIGGER)[AXIS\_GAS](https://developer.android.com/reference/android/view/MotionEvent#AXIS_GAS)

[**Supportare i controller in tutte le versioni di Android**](https://developer.android.com/training/game-controllers/compatibility)

Scopri come fare in modo che i controller di gioco si comportino allo stesso modo su dispositivi che eseguono versioni diverse di Android.

# Supportare i controller in tutte le versioni di Android

Se stai supportando i controller di gioco nel tuo gioco, è tua responsabilità assicurarti che il tuo gioco risponda in modo coerente ai controller su dispositivi in esecuzione su diverse versioni di Android. Ciò consente al tuo gioco di raggiungere un pubblico più ampio e i tuoi giocatori possono godere di un'esperienza di gioco senza soluzione di continuità con i loro controller anche quando cambiano o aggiornano i loro dispositivi Android.

Questa lezione illustra come usare le API disponibili in Android 4.1 e versioni precedenti in modo compatibile con le versioni precedenti, consentendo al gioco di supportare le funzionalità seguenti sui dispositivi che eseguono Android 3.1 e versioni precedenti:

* Il gioco può rilevare se un nuovo controller di gioco viene aggiunto, modificato o rimosso.
* Il gioco può interrogare le funzionalità di un controller di gioco.
* Il gioco può riconoscere gli eventi di movimento in arrivo da un controller di gioco.

Gli esempi di questa lezione si basano sull'implementazione di riferimento fornita dall'esempio disponibile per il download precedente. Questo esempio mostra come implementare l'interfaccia per supportare diverse versioni di Android. Per compilare l'esempio, devi usare Android 4.1 (livello API 16) o versione successiva. Una volta compilata, l'app di esempio viene eseguita su qualsiasi dispositivo che esegue Android 3.1 (livello API 12) o superiore come destinazione di compilazione. ControllerSample.zipInputManagerCompat

## Prepararsi alle API astratte per il supporto dei controller di gioco

Si supponga di voler essere in grado di determinare se lo stato di connessione di un controller di gioco è cambiato nei dispositivi in esecuzione su Android 3.1 (livello API 12). Tuttavia, le API sono disponibili solo in Android 4.1 (livello API 16) e versioni superiori, quindi è necessario fornire un'implementazione che supporti Android 4.1 e versioni superiori fornendo al contempo un meccanismo di fallback che supporti Android 3.1 fino ad Android 4.0.

Per aiutarti a determinare quali funzionalità richiedono un tale meccanismo di fallback per le versioni precedenti, la tabella 1 elenca le differenze nel supporto del controller di gioco tra Android 3.1 (livello API 12) e 4.1 (livello API 16).

**La tabella 1.** API per il supporto del controller di gioco in diverse versioni di Android.

|  |  |  |  |
| --- | --- | --- | --- |
| **Informazioni sul controller** | **Controller API** | **API livello 12** | **API livello 16** |
| Identificazione del dispositivo | [getInputDeviceIds()](https://developer.android.com/reference/android/hardware/input/InputManager#getInputDeviceIds()) |  | • |
| [getInputDevice()](https://developer.android.com/reference/android/hardware/input/InputManager#getInputDevice(int)) |  | • |
| [getVibrator()](https://developer.android.com/reference/android/view/InputDevice#getVibrator()) |  | • |
| [SOURCE\_JOYSTICK](https://developer.android.com/reference/android/view/InputDevice#SOURCE_JOYSTICK) | • | • |
| [SOURCE\_GAMEPAD](https://developer.android.com/reference/android/view/InputDevice#SOURCE_GAMEPAD) | • | • |
| Stato connessione | [onInputDeviceAdded()](https://developer.android.com/reference/android/hardware/input/InputManager.InputDeviceListener#onInputDeviceAdded(int)) |  | • |
| [onInputDeviceChanged()](https://developer.android.com/reference/android/hardware/input/InputManager.InputDeviceListener#onInputDeviceChanged(int)) |  | • |
| [onInputDeviceRemoved()](https://developer.android.com/reference/android/hardware/input/InputManager.InputDeviceListener#onInputDeviceRemoved(int)) |  | • |
| Identificazione evento di input | Pressa D-pad ( , , , , [KEYCODE\_DPAD\_UP](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_UP)[KEYCODE\_DPAD\_DOWN](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_DOWN)[KEYCODE\_DPAD\_LEFT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_LEFT)[KEYCODE\_DPAD\_RIGHT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_RIGHT)[KEYCODE\_DPAD\_CENTER](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER)) | • | • |
| Pulsante Gamepad ( , , , , , , , , , [BUTTON\_A](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_A)[BUTTON\_B](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_B)[BUTTON\_THUMBL](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_THUMBL)[BUTTON\_THUMBR](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_THUMBR)[BUTTON\_SELECT](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_SELECT)[BUTTON\_START](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_START)[BUTTON\_R1](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_R1)[BUTTON\_L1](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_L1)[BUTTON\_R2](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_R2)[BUTTON\_L2](https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_BUTTON_L2)) | • | • |
| Movimento joystick e interruttore cappello ( , , , , [AXIS\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_X)[AXIS\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Y)[AXIS\_Z](https://developer.android.com/reference/android/view/MotionEvent#AXIS_Z)[AXIS\_RZ](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RZ)[AXIS\_HAT\_X](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_X)[AXIS\_HAT\_Y](https://developer.android.com/reference/android/view/MotionEvent#AXIS_HAT_Y)) | • | • |
| Pressione del grilletto analogico ( , [AXIS\_LTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_LTRIGGER)[AXIS\_RTRIGGER](https://developer.android.com/reference/android/view/MotionEvent#AXIS_RTRIGGER)) | • | • |

Puoi usare l'astrazione per creare il supporto del controller di gioco in grado di riconoscere la versione che funziona su più piattaforme. Questo approccio prevede i seguenti passaggi:

1. Definisci un'interfaccia Java intermedia che astrae l'implementazione delle funzionalità del controller di gioco richieste dal tuo gioco.
2. Crea un'implementazione proxy della tua interfaccia che usa API in Android 4.1 e versioni superiori.
3. Crea un'implementazione personalizzata della tua interfaccia che usa API disponibili tra Android 3.1 fino ad Android 4.0.
4. Crea la logica per passare da un'implementazione all'altro in fase di esecuzione e inizia a usare l'interfaccia nel gioco.

Per una panoramica di come l'astrazione può essere utilizzata per garantire che le applicazioni possano funzionare in modo compatibile con le versioni precedenti di Android, vedere [Creazione di URI compatibili con le versioni precedenti](https://developer.android.com/training/backward-compatible-ui).

## Aggiungere un'interfaccia per la compatibilità con le versioni precedenti

Per garantire la compatibilità con le versioni precedenti, è possibile creare un'interfaccia personalizzata e quindi aggiungere implementazioni specifiche della versione. Un vantaggio di questo approccio è che ti consente di rispecchiare le interfacce pubbliche su Android 4.1 (livello API 16) che supportano i controller di gioco.

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

// The InputManagerCompat interface is a reference example.  
// The full code is provided in the ControllerSample.zip sample.  
interface InputManagerCompat {  
    val inputDeviceIds: IntArray  
    fun getInputDevice(id: Int): InputDevice  
  
    fun registerInputDeviceListener(  
            listener: InputManager.InputDeviceListener,  
            handler: Handler?  
    )  
  
    fun unregisterInputDeviceListener(listener:InputManager.InputDeviceListener)  
  
    fun onGenericMotionEvent(event: MotionEvent)  
  
    fun onPause()  
    fun onResume()  
  
    interface InputDeviceListener {  
        fun onInputDeviceAdded(deviceId: Int)  
        fun onInputDeviceChanged(deviceId: Int)  
        fun onInputDeviceRemoved(deviceId: Int)  
    }  
}

L'interfaccia fornisce i metodi seguenti:InputManagerCompat

getInputDevice()

Specchi. Ottiene l'oggetto che rappresenta le funzionalità di un controller di gioco.[getInputDevice()](https://developer.android.com/reference/android/hardware/input/InputManager#getInputDevice(int))[InputDevice](https://developer.android.com/reference/android/view/InputDevice)

getInputDeviceIds()

Specchi. Restituisce una matrice di numeri interi, ognuno dei quali è un ID per un dispositivo di input diverso. Questo è utile se stai costruendo un gioco che supporta più giocatori e vuoi rilevare quanti controller sono connessi.[getInputDeviceIds()](https://developer.android.com/reference/android/hardware/input/InputManager#getInputDeviceIds())

registerInputDeviceListener()

Specchi. Consente di registrarsi per essere informati quando un nuovo dispositivo viene aggiunto, modificato o rimosso.[registerInputDeviceListener()](https://developer.android.com/reference/android/hardware/input/InputManager#registerInputDeviceListener(android.hardware.input.InputManager.InputDeviceListener,%20android.os.Handler))

unregisterInputDeviceListener()

Specchi. Annulla la registrazione di un listener di dispositivi di input.[unregisterInputDeviceListener()](https://developer.android.com/reference/android/hardware/input/InputManager#unregisterInputDeviceListener(android.hardware.input.InputManager.InputDeviceListener))

onGenericMotionEvent()

Specchi. Consente al gioco di intercettare e gestire oggetti e valori dell'asse che rappresentano eventi come movimenti del joystick e presse analogiche.[onGenericMotionEvent()](https://developer.android.com/reference/android/view/View#onGenericMotionEvent(android.view.MotionEvent))[MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

onPause()

Interrompe il polling per gli eventi del controller di gioco quando l'attività principale è in pausa o quando il gioco non ha più lo stato attivo.

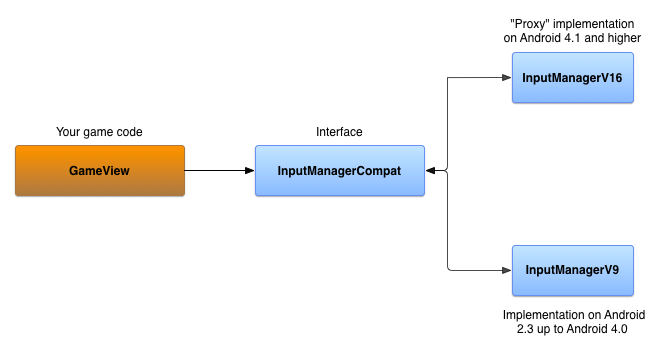
onResume()

Avvia il polling degli eventi del controller di gioco quando l'attività principale viene ripresa o quando il gioco viene avviato ed eseguito in primo piano.

InputDeviceListener

Rispecchia l'interfaccia. Consente al tuo gioco di sapere quando un controller di gioco è stato aggiunto, modificato o rimosso.[InputManager.InputDeviceListener](https://developer.android.com/reference/android/hardware/input/InputManager.InputDeviceListener)

Quindi, crea implementazioni per questo lavoro in diverse versioni della piattaforma. Se il gioco è in esecuzione su Android 4.1 o versione successiva e chiama un metodo, l'implementazione proxy chiama il metodo equivalente in . Tuttavia, se il tuo gioco è in esecuzione su Android 3.1 fino ad Android 4.0, l'implementazione personalizzata elabora le chiamate ai metodi utilizzando solo API introdotte non oltre Android 3.1. Indipendentemente dall'implementazione specifica della versione utilizzata in fase di esecuzione, l'implementazione restituisce i risultati della chiamata in modo trasparente al gioco.InputManagerCompatInputManagerCompat[InputManager](https://developer.android.com/reference/android/hardware/input/InputManager)InputManagerCompat



**Figura 1.** Diagramma di classe delle implementazioni specifiche dell'interfaccia e della versione.

## Implementare l'interfaccia su Android 4.1 e versioni successiva

InputManagerCompatV16 è un'implementazione dell'interfaccia che proxya le chiamate al metodo a un oggetto effettivo e . L'oggetto è ottenuto dal sistema .InputManagerCompat[InputManager](https://developer.android.com/reference/android/hardware/input/InputManager)[InputManager.InputDeviceListener](https://developer.android.com/reference/android/hardware/input/InputManager.InputDeviceListener)[InputManager](https://developer.android.com/reference/android/hardware/input/InputManager)[Context](https://developer.android.com/reference/android/content/Context)

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

// The InputManagerCompatV16 class is a reference implementation.  
// The full code is provided in the ControllerSample.zip sample.  
public class InputManagerV16(  
        context: Context,  
        private val inputManager: InputManager =  
            context.getSystemService(Context.INPUT\_SERVICE) as InputManager,  
        private val listeners:  
            MutableMap<InputManager.InputDeviceListener, V16InputDeviceListener> = mutableMapOf()  
) : InputManagerCompat {  
    override val inputDeviceIds: IntArray = inputManager.inputDeviceIds  
  
    override fun getInputDevice(id: Int): InputDevice = inputManager.getInputDevice(id)  
  
    override fun registerInputDeviceListener(  
            listener: InputManager.InputDeviceListener,  
            handler: Handler?  
    ) {  
        V16InputDeviceListener(listener).also { v16listener ->  
            inputManager.registerInputDeviceListener(v16listener, handler)  
            listeners += listener to v16listener  
        }  
    }  
  
    // Do the same for unregistering an input device listener  
    ...  
  
    override fun onGenericMotionEvent(event: MotionEvent) {  
        // unused in V16  
    }  
  
    override fun onPause() {  
        // unused in V16  
    }  
  
    override fun onResume() {  
        // unused in V16  
    }  
  
}  
  
class V16InputDeviceListener(  
        private val idl: InputManager.InputDeviceListener  
) : InputManager.InputDeviceListener {  
  
    override fun onInputDeviceAdded(deviceId: Int) {  
        idl.onInputDeviceAdded(deviceId)  
    }  
    // Do the same for device change and removal  
    ...  
}

## Implementa l'interfaccia su Android 3.1 fino ad Android 4.0

Per creare un'implementazione che supporti Android 3.1 fino ad Android 4.0, puoi usare i seguenti oggetti: InputManagerCompat

* Uno degli ID dispositivo per tenere traccia dei controller di gioco collegati al dispositivo.[SparseArray](https://developer.android.com/reference/android/util/SparseArray)
* Oggetto per elaborare gli eventi del dispositivo. Quando un'app viene avviata o ripresa, riceve un messaggio per avviare il polling per la disconnessione del controller di gioco. L'oggetto avvia un ciclo per controllare ogni controller di gioco connesso noto e vedere se viene restituito un ID dispositivo. Un valore restituito indica che il controller di gioco è disconnesso. Interrompe il polling quando l'app è in pausa.[HandlerHandlerHandler](https://developer.android.com/reference/android/os/Handler)null[Handler](https://developer.android.com/reference/android/os/Handler)
* Oggetto di oggetti. Utilizzerai i listener per aggiornare lo stato di connessione dei controller di gioco tracciati.[Map](https://developer.android.com/reference/java/util/Map)InputManagerCompat.InputDeviceListener

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

// The InputManagerCompatV9 class is a reference implementation.  
// The full code is provided in the ControllerSample.zip sample.  
class InputManagerV9(  
        val devices: SparseArray<Array<Long>> = SparseArray(),  
        private val listeners:  
        MutableMap<InputManager.InputDeviceListener, Handler> = mutableMapOf()  
) : InputManagerCompat {  
    private val defaultHandler: Handler = PollingMessageHandler(this)  
    …  
}

Implementare un oggetto che estenda ed eseguono l'override del metodo . Questo metodo verifica se un controller di gioco collegato è stato disconnesso e notifica ai listener registrati.PollingMessageHandler[Handler](https://developer.android.com/reference/android/os/Handler)[handleMessage()](https://developer.android.com/reference/android/os/Handler#handleMessage(android.os.Message))

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

private class PollingMessageHandler(  
        inputManager: InputManagerV9,  
        private val mInputManager: WeakReference<InputManagerV9> = WeakReference(inputManager)  
) : Handler() {  
  
    override fun handleMessage(msg: Message) {  
        super.handleMessage(msg)  
        when (msg.what) {  
            MESSAGE\_TEST\_FOR\_DISCONNECT -> {  
                mInputManager.get()?.also { imv ->  
                    val time = SystemClock.elapsedRealtime()  
                    val size = imv.devices.size()  
                    for (i in 0 until size) {  
                        imv.devices.valueAt(i)?.also { lastContact ->  
                            if (time - lastContact[0] > CHECK\_ELAPSED\_TIME) {  
                                // check to see if the device has been  
                                // disconnected  
                                val id = imv.devices.keyAt(i)  
                                if (null == InputDevice.getDevice(id)) {  
                                    // Notify the registered listeners  
                                    // that the game controller is disconnected  
                                    imv.devices.remove(id)  
                                } else {  
                                    lastContact[0] = time  
                                }  
                            }  
                        }  
                    }  
                    sendEmptyMessageDelayed(MESSAGE\_TEST\_FOR\_DISCONNECT, CHECK\_ELAPSED\_TIME)  
                }  
            }  
        }  
    }  
}

Per avviare e interrompere il polling per la disconnessione del controller di gioco, eseguire l'override dei metodi seguenti:

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

private const val MESSAGE\_TEST\_FOR\_DISCONNECT = 101  
private const val CHECK\_ELAPSED\_TIME = 3000L  
  
class InputManagerV9(  
        val devices: SparseArray<Array<Long>> = SparseArray(),  
        private val listeners:  
        MutableMap<InputManager.InputDeviceListener, Handler> = mutableMapOf()  
) : InputManagerCompat {  
    ...  
    override fun onPause() {  
        defaultHandler.removeMessages(MESSAGE\_TEST\_FOR\_DISCONNECT)  
    }  
  
    override fun onResume() {  
        defaultHandler.sendEmptyMessageDelayed(MESSAGE\_TEST\_FOR\_DISCONNECT, CHECK\_ELAPSED\_TIME)  
    }  
    ...  
}

Per rilevare che è stato aggiunto un dispositivo di input, eseguire l'override del metodo . Quando il sistema segnala un evento di movimento, verificare se l'evento proviene da un ID dispositivo già monitorato o da un nuovo ID dispositivo. Se l'ID del dispositivo è nuovo, avvisare i listener registrati.onGenericMotionEvent()

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

override fun onGenericMotionEvent(event: MotionEvent) {  
    // detect new devices  
    val id = event.deviceId  
    val timeArray: Array<Long> = mDevices.get(id) ?: run {  
        // Notify the registered listeners that a game controller is added  
        ...  
        arrayOf<Long>().also {  
            mDevices.put(id, it)  
        }  
    }  
    timeArray[0] = SystemClock.elapsedRealtime()  
}

La notifica dei listener viene implementata utilizzando l'oggetto per inviare un oggetto alla coda messaggi. Contiene un riferimento a un oggetto . Quando l'oggetto viene eseguito, il metodo di callback appropriato del listener viene chiamato per segnalare se il controller di gioco è stato aggiunto, modificato o rimosso. [Handler](https://developer.android.com/reference/android/os/Handler)DeviceEvent[Runnable](https://developer.android.com/reference/java/lang/Runnable)DeviceEventInputManagerCompat.InputDeviceListenerDeviceEvent

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

class InputManagerV9(  
        val devices: SparseArray<Array<Long>> = SparseArray(),  
        private val listeners:  
        MutableMap<InputManager.InputDeviceListener, Handler> = mutableMapOf()  
) : InputManagerCompat {  
    ...  
    override fun registerInputDeviceListener(  
            listener: InputManager.InputDeviceListener,  
            handler: Handler?  
    ) {  
        listeners[listener] = handler ?: defaultHandler  
    }  
  
    override fun unregisterInputDeviceListener(listener: InputManager.InputDeviceListener) {  
        listeners.remove(listener)  
    }  
  
    private fun notifyListeners(why: Int, deviceId: Int) {  
        // the state of some device has changed  
        listeners.forEach { listener, handler ->  
            DeviceEvent.getDeviceEvent(why, deviceId, listener).also {  
                handler?.post(it)  
            }  
        }  
    }  
    ...  
}  
  
private val sObjectQueue: Queue<DeviceEvent> = ArrayDeque<DeviceEvent>()  
  
private class DeviceEvent(  
        private var mMessageType: Int,  
        private var mId: Int,  
        private var mListener: InputManager.InputDeviceListener  
) : Runnable {  
  
    companion object {  
        fun getDeviceEvent(messageType: Int, id: Int, listener: InputManager.InputDeviceListener) =  
                sObjectQueue.poll()?.apply {  
                    mMessageType = messageType  
                    mId = id  
                    mListener = listener  
                } ?: DeviceEvent(messageType, id, listener)  
  
    }  
  
    override fun run() {  
        when(mMessageType) {  
            ON\_DEVICE\_ADDED -> mListener.onInputDeviceAdded(mId)  
            ON\_DEVICE\_CHANGED -> mListener.onInputDeviceChanged(mId)  
            ON\_DEVICE\_REMOVED -> mListener.onInputDeviceChanged(mId)  
            else -> {  
                // Handle unknown message type  
            }  
        }  
    }  
  
}

Ora hai due implementazioni di: una che funziona su dispositivi che eseguono Android 4.1 e versioni superiori e un'altra che funziona su dispositivi che eseguono Android 3.1 fino ad Android 4.0.InputManagerCompat

## Utilizzare l'implementazione specifica della versione

La logica di commutazione specifica della versione viene implementata in una classe che funge da [factory](https://en.wikipedia.org/wiki/Factory_(software_concept)).

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

object Factory {  
    fun getInputManager(context: Context): InputManagerCompat =  
            if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.JELLY\_BEAN) {  
                InputManagerV16(context)  
            } else {  
                InputManagerV9()  
            }  
}

Ora puoi semplicemente creare un'istanza di un oggetto e registrare un oggetto nel tuo principale . A causa della logica di cambio versione configurata, il gioco utilizza automaticamente l'implementazione appropriata per la versione di Android in esecuzione del dispositivo.InputManagerCompatInputManagerCompat.InputDeviceListener[View](https://developer.android.com/reference/android/view/View)

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

class GameView(context: Context) : View(context), InputManager.InputDeviceListener {  
    private val inputManager: InputManagerCompat = Factory.getInputManager(context).apply {  
        registerInputDeviceListener(this@GameView, null)  
        ...  
    }  
    ...  
}

Eseguire quindi l'override del metodo nella visualizzazione principale, come descritto in [Handle a MotionEvent from a Game Controller](https://developer.android.com/training/game-controllers/controller-input#analog). Il tuo gioco dovrebbe ora essere in grado di elaborare gli eventi del controller di gioco in modo coerente sui dispositivi che eseguono Android 3.1 (livello API 12) e versioni superiori. [onGenericMotionEvent()](https://developer.android.com/reference/android/view/View#onGenericMotionEvent(android.view.MotionEvent))

[KOTLIN](https://developer.android.com/training/game-controllers/compatibility#kotlin)

override fun onGenericMotionEvent(event: MotionEvent): Boolean {  
    inputManager.onGenericMotionEvent(event)  
  
    // Handle analog input from the controller as normal  
    ...  
    return super.onGenericMotionEvent(event)  
}

È possibile trovare un'implementazione completa di questo codice di compatibilità nella classe fornita nell'esempio disponibile per il download precedente.GameViewControllerSample.zip

[**Supportare più controller di gioco**](https://developer.android.com/training/game-controllers/multiple-controllers)

Scopri come rilevare e utilizzare più controller di gioco connessi contemporaneamente.

# Supportare più controller di gioco

Mentre la maggior parte dei giochi sono progettati per supportare un singolo utente per dispositivo Android, è anche possibile supportare più utenti con controller di gioco connessi contemporaneamente sullo stesso dispositivo Android.

Questa lezione illustra alcune tecniche di base per gestire l'input nel gioco multiplayer a dispositivo singolo da più controller connessi. Ciò include il mantenimento di una mappatura tra gli avatar del giocatore e ogni dispositivo controller e l'elaborazione appropriata degli eventi di input del controller.

## Mappare i lettori agli ID dispositivo controller

Quando un controller di gioco è collegato a un dispositivo Android, il sistema gli assegna un ID dispositivo intero. È possibile ottenere gli ID dispositivo per i controller di gioco connessi chiamando , come illustrato in [Verificare che un controller di gioco sia connesso](https://developer.android.com/training/game-controllers/controller-input#input). Puoi quindi associare ogni ID dispositivo a un giocatore nel tuo gioco ed elaborare le azioni di gioco per ogni giocatore separatamente. [InputDevice.getDeviceIds()](https://developer.android.com/reference/android/view/InputDevice#getDeviceIds())

**Nota:** Nei dispositivi che eseguono Android 4.1 (livello API 16) e versioni superiori, è possibile ottenere il descrittore di un dispositivo di input utilizzando , che restituisce un valore stringa persistente univoco per il dispositivo di input. A differenza di un ID dispositivo, il valore del descrittore non cambierà anche se il dispositivo di input viene disconnesso, riconnesso o riconfigurato. [**getDescriptor()**](https://developer.android.com/reference/android/view/InputDevice#getDescriptor())

Il frammento di codice riportato di seguito mostra come usare un oggetto per associare l'avatar di un giocatore a un controller specifico. In questo esempio, la variabile memorizza un insieme di oggetti. Un nuovo avatar giocatore viene creato nel gioco quando un nuovo controller viene collegato da un utente e rimosso quando il controller associato viene rimosso. [SparseArray](https://developer.android.com/reference/android/util/SparseArray)mShipsShip

I metodi di callback e fanno parte del livello di astrazione introdotto in [Support Controllers Across Android Versions](https://developer.android.com/training/game-controllers/compatibility#status_callbacks}). Implementando questi callback listener, il gioco può identificare l'ID del dispositivo del controller di gioco quando un controller viene aggiunto o rimosso. Questo rilevamento è compatibile con Android 2.3 (livello API 9) e versioni precedenti. onInputDeviceAdded()onInputDeviceRemoved()

[KOTLIN](https://developer.android.com/training/game-controllers/multiple-controllers#kotlin)

private val ships = SparseArray<Ship>()  
  
override fun onInputDeviceAdded(deviceId: Int) {  
    getShipForID(deviceId)  
}  
  
override fun onInputDeviceRemoved(deviceId: Int) {  
    removeShipForID(deviceId)  
}  
  
private fun getShipForID(shipID: Int): Ship {  
    return ships.get(shipID) ?: Ship().also {  
        ships.append(shipID, it)  
    }  
}  
  
private fun removeShipForID(shipID: Int) {  
    ships.remove(shipID)  
}

## Elaborare l'input di più controller

Il gioco deve eseguire il seguente ciclo per elaborare l'input da più controller:

1. Rilevare se si è verificato un evento di input.
2. Identificare l'origine di input e il relativo ID dispositivo.
3. In base all'azione indicata dal codice della chiave dell'evento di input o dal valore dell'asse, aggiornare l'avatar del giocatore associato a tale ID dispositivo.
4. Eseguire il rendering e l'aggiornamento dell'interfaccia utente.

[KeyEvent](https://developer.android.com/reference/android/view/KeyEvent) e agli eventi di input sono associati ID dispositivo. Il tuo gioco può approfittarne per determinare da quale controller proviene l'evento di input e aggiornare l'avatar del giocatore associato a quel controller. [MotionEvent](https://developer.android.com/reference/android/view/MotionEvent)

Il seguente frammento di codice mostra come è possibile ottenere un riferimento avatar del giocatore corrispondente a un ID dispositivo controller di gioco e aggiornare il gioco in base alla pressione del pulsante dell'utente su quel controller.

[KOTLIN](https://developer.android.com/training/game-controllers/multiple-controllers#kotlin)

override fun onKeyDown(keyCode: Int, event: KeyEvent): Boolean {  
    if (event.source and InputDevice.SOURCE\_GAMEPAD == InputDevice.SOURCE\_GAMEPAD) {  
        event.deviceId.takeIf { it != -1 }?.also { deviceId ->  
            val currentShip: Ship = getShipForID(deviceId)  
            // Based on which key was pressed, update the player avatar  
            // (e.g. set the ship headings or fire lasers)  
            return true  
        }  
    }  
    return super.onKeyDown(keyCode, event)  
}

**Nota:** Come best practice, quando il controller di gioco di un utente si disconnette, è necessario sospendere il gioco e chiedere se l'utente desidera riconnettersi.

# Creare un metodo di input

Un IME (Input Method Editor) è un controllo utente che consente agli utenti di immettere testo. Android fornisce un framework di input-method estensibile che consente alle applicazioni di fornire agli utenti metodi di input alternativi, come tastiere su schermo o persino input vocale. Dopo aver installato le IMI desiderate, un utente può selezionare quale utilizzare dalle impostazioni di sistema e utilizzarlo nell'intero sistema; è possibile abilitare un solo IME alla volta.

Per aggiungere un IME al sistema Android, è necessario creare un'applicazione Android contenente una classe che estende . Inoltre, in genere si crea un'attività "impostazioni" che passa le opzioni al servizio IME. Puoi anche definire un'interfaccia utente delle impostazioni visualizzata come parte delle impostazioni di sistema. [InputMethodService](https://developer.android.com/reference/android/inputmethodservice/InputMethodService)

La presente guida tratta quanto segue:

* Ciclo di vita dell'IME
* Dichiarazione di componenti IME nel manifesto dell'applicazione
* The IME API
* Progettazione di un'interfaccia utente IME
* Invio di testo da un IME a un'applicazione
* Utilizzo dei sottotipi IME

Se non hai mai lavorato con gli IME prima, dovresti prima leggere l'articolo introduttivo [Metodi di input sullo](http://android-developers.blogspot.com/2009/04/updating-applications-for-on-screen.html) schermo.

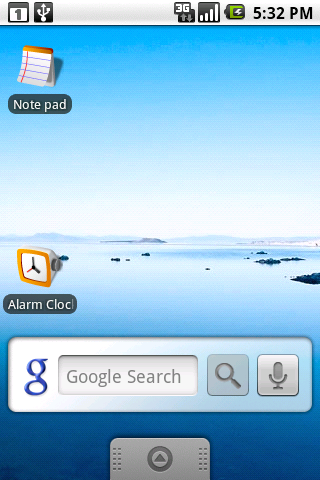
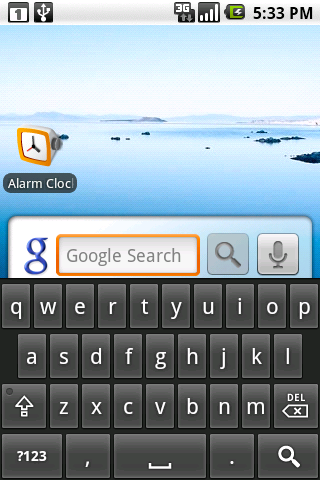
## [**Metodi di input su schermo**](https://android-developers.googleblog.com/2009/04/updating-applications-for-on-screen.html)

A partire da Android 1.5, la piattaforma Android offre un Framework del metodo di input (IMF) che consente di creare metodi di input sullo schermo come le tastiere software. Questo articolo fornisce una panoramica di cosa sono gli editor di metodi di input Android (IMI) e cosa deve fare un'applicazione per funzionare bene con loro. Il FMI è progettato per supportare nuove classi di dispositivi Android, come quelli senza tastiere hardware, quindi è importante che la tua applicazione funzioni bene con il FMI e offra una grande esperienza per gli utenti.

### Che cos'è un metodo di input?

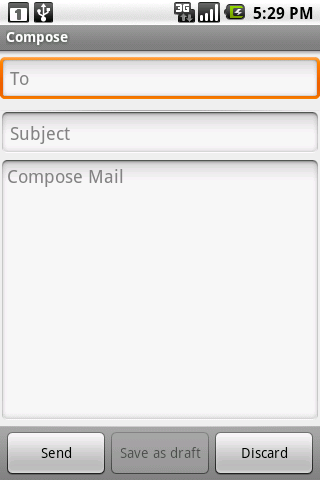
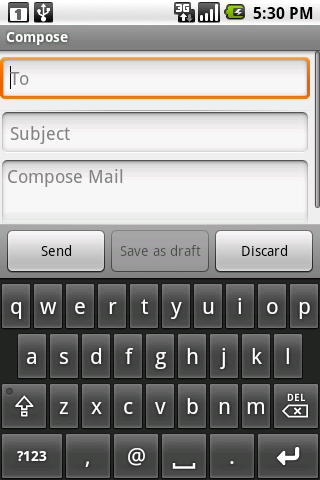
Android IMF è progettato per supportare una varietà di IMI, tra cui tastiera morbida, riconoscimento della scrittura a mano e traduttori hard keyboard. Il nostro focus, tuttavia, sarà sulle tastiere morbide, poiché questo è il tipo di metodo di input che attualmente fa parte della piattaforma.

Un utente accede in genere all'IME corrente toccando una visualizzazione di testo da modificare, come mostrato qui nella schermata iniziale:

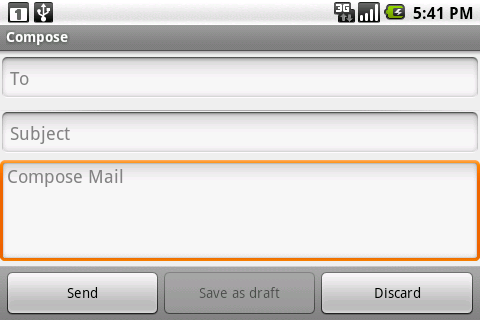
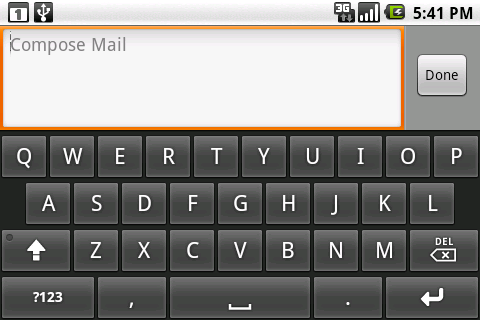
La tastiera morbida è posizionata nella parte inferiore dello schermo sopra la finestra dell'applicazione. Per organizzare lo spazio disponibile tra l'applicazione e l'IME, utilizziamo alcuni approcci; quello mostrato qui è chiamato pan and scane comporta semplicemente lo scorrimento della finestra dell'applicazione in modo che la visualizzazione attualmente focalizzata sia visibile. Questa è la modalità predefinita, poiché è la più sicura per le applicazioni esistenti.

Molto spesso il layout dello schermo preferito è un ridimensionamento , in cui la finestra dell'applicazione viene ridimensionata per essere completamentevisibile. Un esempio viene mostrato qui, quando si compone un messaggio di posta elettronica:

Le dimensioni della finestra dell'applicazione vengono modificate in modo che nessuna di essa sia nascosta dall'IME, consentendo l'accesso completo sia all'applicazione che all'IME. Questo naturalmente funziona solo per le applicazioni che hanno un'area ridimensionabile che può essere ridotta per fare abbastanza spazio, ma lo spazio verticale in questa modalità è in realtà non inferiore a quello disponibile nell'orientamento orizzontale, quindi molto spesso un'applicazione può già ospitarlo.

L'ultima modalità principale è la modalità a schermo intero o di estrazione. Questo viene utilizzato quando l'IME è troppo grande per condividere ragionevolmente lo spazio con l'applicazione sottostante. Con gli IE standard, incontrerai questa situazione solo quando lo schermo ha un orientamento orizzontale, anche se altri IMI sono liberi di usarlo ogni volta che lo desiderano. In questo caso la finestra dell'applicazione viene lasciata così com'è e l'IME visualizza semplicemente lo schermo intero sopra di esso, come mostrato qui:

Poiché l'IME copre l'applicazione, ha una propria area di modifica, che mostra il testo effettivamente contenuto nell'applicazione. Ci sono anche alcune opportunità limitate che l'applicazione ha per personalizzare parti dell'IME (il pulsante "fatto" in alto e inserire l'etichetta chiave in basso) per migliorare l'esperienza dell'utente.

### Attributi XML di base per il controllo delle IMI

Ci sono una serie di cose che il sistema fa per cercare di aiutare le applicazioni esistenti a funzionare con gli IMI nel miglior modo possibile, come ad esempio:

* Usa la modalità di panoramica e scansione per impostazione predefinita, a meno che non possa ragionevolmente indovinare che la modalità di ridimensionamento funzionerà in base all'esistenza di elenchi, visualizzazioni di scorrimento e così via.
* Analizza i vari attributi TextView esistenti per indovinare il tipo di contenuto (numeri, testo normale e così via) per aiutare la tastiera soft a visualizzare un layout di tasti appropriato.
* Assegnare alcune azioni predefinite all'IME a schermo intero, ad esempio "campo successivo" e "fatto".

Ci sono anche alcune semplici cose che puoi fare nella tua applicazione che spesso miglioreranno notevolmente la sua esperienza utente. Tranne dove esplicitamente menzionato, questi funzioneranno in qualsiasi versione della piattaforma Android, anche quelli precedenti ad Android 1.5 (poiché ignoreranno semplicemente queste nuove opzioni).

#### Specifica del tipo di input di ogni controllo EditText

La cosa più importante da eseguire per un'applicazione è utilizzare il nuovo attributo in ogni . L'attributo fornisce informazioni molto più dettagliate sul contenuto del testo. Questo attributo sostituisce in realtà molti attributi esistenti (, , , , , e ). Se si specificano gli attributi precedenti e il nuovo attributo, il sistema utilizza e ignora gli altri. android:inputTypeEditTextandroid:passwordandroid:singleLineandroid:numericandroid:phoneNumberandroid:capitalizeandroid:autoTextandroid:editableandroid:inputTypeandroid:inputType

L'attributo ha tre pezzi:android:inputType

* La classe è l'interpretazione complessiva dei caratteri. Le classi attualmente supportate sono (testo normale), (numero decimale), (numero di telefono) e (data o ora).textnumberphonedatetime
* La variazione è un ulteriore perfezionamento della classe. Nell'attributo si specificano normalmente la classe e la variante insieme, con la classe come prefisso. Ad esempio, è un campo di testo in cui l'utente immetterà qualcosa che è un indirizzo di posta elettronica (foo@bar.com) in modo che il layout della chiave abbia un carattere '@' in facile accesso ed è un campo numerico con un segno. Se viene specificata solo la classe, si ottiene la variante predefinita/generica.textEmailAddressnumberSigned
* È possibile specificare flag aggiuntivi che forniscono ulteriore perfezionamento. Questi flag sono specifici di una classe. Ad esempio, alcuni flag per la classe sono , e .texttextCapSentencestextAutoCorrecttextMultiline

Ad esempio, ecco il nuovo EditText per la visualizzazione testo del messaggio dell'applicazione di messaggistica istantanea:

<EditText android:id="@+id/edtInput"

android:layout\_width="0dip"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:inputType="textShortMessage|textAutoCorrect|textCapSentences|textMultiLine"

android:imeOptions="actionSend|flagNoEnterAction"

android:maxLines="4"

android:maxLength="2000"

android:hint="@string/compose\_hint"/>

Una descrizione completa di tutti i tipi di input è disponibile nella documentazione. È importante utilizzare i tipi di input corretti disponibili, in modo che la tastiera soft possa utilizzare il layout di tastiera ottimale per il testo che l'utente immetterà.

#### Abilitazione della modalità di ridimensionamento e di altre funzionalità della finestra

La seconda cosa più importante da fare per la tua app è specificare il comportamento complessivo della finestra in relazione al metodo di input. L'aspetto più visibile di questo è il controllo del ridimensionamento rispetto alla modalità di panoramica e scansione, ma ci sono anche altre cose che puoi fare per migliorare la tua esperienza utente.

In genere si controlerà questo comportamento tramite l'attributo in ogni definizione dell'. Come il tipo di input, ci sono un paio di diversi dati che possono essere specificati qui combinandoli insieme:android:windowSoftInputMode<activity>AndroidManifest.xml

* La modalità di regolazione della finestra viene specificata con o . Si consiglia vivamente di specificare sempre l'uno o l'altro.adjustResizeadjustPan
* È possibile controllare ulteriormente se l'IME verrà visualizzato automaticamente quando viene visualizzata l'attività e altre situazioni in cui l'utente si sposta su di essa. Il sistema non mostrerà automaticamente un IME per impostazione predefinita, ma in alcuni casi può essere conveniente per l'utente se un'applicazione abilita questo comportamento. È possibile richiederlo con . Ci sono anche una serie di altre opzioni di stato per un controllo più fine che puoi trovare nella documentazione.stateVisible

Un esempio tipico di questo campo può essere visualizzato nell'attività di modifica del contatto, che garantisce che venga ridimensionato e visualizza automaticamente l'IME per l'utente:

<activity name="EditContactActivity"

android:windowSoftInputMode="stateVisible|adjustResize">

...

</activity>

**Nota:** A partire da Android 1.5 (API Level 3), la piattaforma offre un nuovo metodo, {@link android.view.Window#setSoftInputMode(modalità int)}, che le finestre non attività possono utilizzare per controllarne il comportamento. Chiamare questo metodo nel tuo renderà la tua applicazione incompatibile con le versioni precedenti della piattaforma Android.

#### Controllo dei pulsanti di azione

La personalizzazione finale che remo a guardare sono i pulsanti "azione" nell'IME. Attualmente esistono due tipi di azioni:

* Il tasto INVIO su una tastiera soft è in genere associato a un'azione quando non si opera su un testo di modifica a riga mult. Ad esempio, sul G1 premendo il tasto hard enter si sposterà in genere al campo successivo o l'applicazione lo intercetterà per eseguire un'azione; con una tastiera morbida, questo overload del tasto INVIO rimane, poiché il pulsante invio invia solo un evento tasto di ingresso.
* In modalità a schermo intero, un IME può anche mettere un pulsante di azione aggiuntivo a destra del testo in fase di modifica, dando all'utente un rapido accesso a un'operazione comune dell'applicazione.

Queste opzioni vengono controllate con l'attributo su . Il valore che fornisi qui può essere qualsiasi combinazione di:android:imeOptionsTextView

* Una delle costanti di azione predefinita (, , , , ). Se nessuno di questi è specificato, il sistema dedurrà o a seconda che ci sia un campo messa a fuoco dopo questo; è possibile forzare esplicitamente nessuna azione con .actionGoactionSearchactionSendactionNextactionDoneactionNextactionDoneactionNone
* L'opzione indica all'IME che l'azione non deve essere disponibile sul tasto INVIO, anche se il testo stesso non è multi-riga. In questo modo si evita di avere azioni irrecuperabili come (inviare) che possono essere toccate accidentalmente dall'utente durante la digitazione.flagNoEnterAction
* Rimuove il pulsante di azione dall'area di testo, lasciando più spazio per il testo.flagNoAccessoryAction
* Rimuove completamente l'area di testo, consentendo all'applicazione di essere vista dietro di essa.flagNoExtractUi

La visualizzazione del messaggio dell'applicazione di messaggistica istantanea precedente fornisce anche un esempio di utilizzo interessante di , per specificare l'azione di invio, ma non lasciarla visualizzare nella chiave invio:imeOptions

android:imeOptions="actionSend|flagNoEnterAction"

### API per il controllo delle IMI

Per un controllo più avanzato sull'IME, è possibile utilizzare una varietà di nuove API. A meno che non venga presa particolare attenzione (ad esempio usando la riflessione), l'utilizzo di queste API causerà l'incompatibilità dell'applicazione con le versioni precedenti di Android e dovresti assicurarti di specificare nel manifesto. Per ulteriori informazioni, vedere la documentazione relativa [all'elemento manifesto <uses-sdk>.](http://developer.android.com/guide/topics/manifest/uses-sdk-element.html)android:minSdkVersion="3"

L'API primaria è la nuova classe, che è possibile recuperare con . Consente di interagire con lo stato del metodo di input globale, ad esempio nascondendo o mostrando esplicitamente l'area di input dell'IME corrente.android.view.inputmethod.InputMethodManagerContext.getSystemService()

Esistono anche nuovi flag di finestra che controllano l'interazione del metodo di input, che è possibile controllare tramite il metodo esistente e il nuovo metodo. La classe ha aumentato i metodi corrispondenti per controllare queste opzioni nella relativa finestra. Una cosa in particolare di cui essere consapevoli è la nuova costante, che viene utilizzata per controllare se una finestra si trova sopra o dietro l'IME corrente.Window.addFlags()Window.setSoftInputMode()PopupWindowWindowManager.LayoutParams.FLAG\_ALT\_FOCUSABLE\_IM

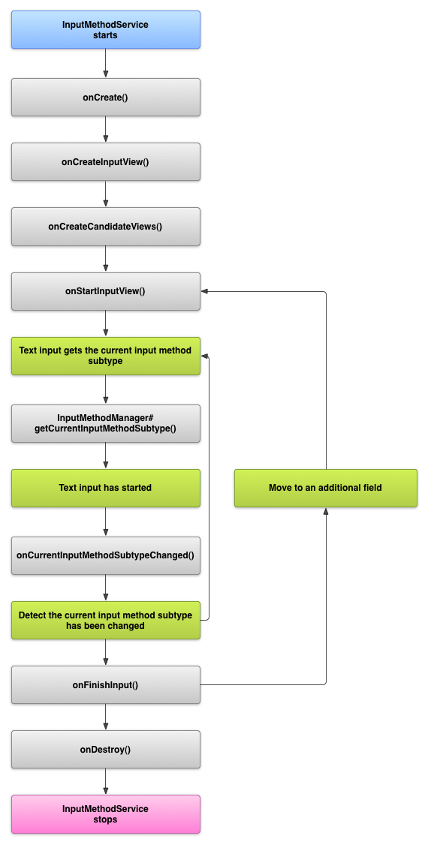
La maggior parte dell'interazione tra un IME attivo e un'applicazione viene eseguita attraverso la classe . Si tratta dell'API implementa da un'applicazione, chiamata da un IME per eseguire le operazioni di modifica appropriate sull'applicazione. Normalmente non dovrai preoccuparti di questo, poiché fornisce la propria implementazione per se stesso.android.view.inputmethod.InputConnectionTextView

Ci sono anche una manciata di nuove API, la più importante delle quali è che crea una nuova per un IME (e compila una struttura con il tipo di input, le opzioni IME e altri dati); ancora una volta, la maggior parte degli sviluppatori non dovrà preoccuparsi di questo, dal momento che TextView se ne occupa per te.View onCreateInputConnection()InputConnectionandroid.view.inputmethod.EditorInfo

**Nota:** A partire da Android 11, la piattaforma consente agli IPE di visualizzare i suggerimenti di riempimento automatico in linea, invece di utilizzare un menu a discesa. Per ulteriori informazioni su come i servizi di riempimento automatico possono supportare questa funzionalità, vedere [Integrazione del riempimento automatico con le tastiere](https://developer.android.com/guide/topics/text/ime-autofill).

## Ciclo di vita dell'IME

Il diagramma seguente descrive il ciclo di vita di un IME:



**Figura 1.** Ciclo di vita di un IME.

Nelle sezioni seguenti viene descritto come implementare l'interfaccia utente e il codice associati a un IME che segue questo ciclo di vita.

## Dichiarare componenti IME nel manifesto

Nel sistema Android, un IME è un'applicazione Android che contiene uno speciale servizio IME. Il file manifesto dell'applicazione deve dichiarare il servizio, richiedere le autorizzazioni necessarie, fornire un filtro intento che corrisponda all'azione e fornire metadati che definisce le caratteristiche dell'IME. Inoltre, per fornire un'interfaccia di impostazioni che consenta all'utente di modificare il comportamento dell'IME, è possibile definire un'attività "impostazioni" che può essere avviata dalle impostazioni di sistema. action.view.InputMethod

Il frammento seguente dichiara un servizio IME. Richiede l'autorizzazione per consentire al servizio di connettere l'IME al sistema, imposta un filtro intento che corrisponde all'azione e definisce i metadati per l'IME: [BIND\_INPUT\_METHOD](https://developer.android.com/reference/android/Manifest.permission#BIND_INPUT_METHOD)android.view.InputMethod

<!-- Declares the input method service -->  
<service android:name="FastInputIME"  
    android:label="@string/fast\_input\_label"  
    android:permission="android.permission.BIND\_INPUT\_METHOD">  
    <intent-filter>  
        <action android:name="android.view.InputMethod" />  
    </intent-filter>  
    <meta-data android:name="android.view.im"  
               android:resource="@xml/method" />  
</service>

Questo frammento successivo dichiara l'attività delle impostazioni per l'IME. Ha un filtro intento per che indica che questa attività è il punto di ingresso principale per l'applicazione IME:[ACTION\_MAIN](https://developer.android.com/reference/android/content/Intent#ACTION_MAIN)

<!-- Optional: an activity for controlling the IME settings -->  
<activity android:name="FastInputIMESettings"  
    android:label="@string/fast\_input\_settings">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN"/>  
    </intent-filter>  
</activity>

Puoi anche fornire l'accesso alle impostazioni dell'IME direttamente dalla sua interfaccia utente.

## API del metodo di input

Le classi specifiche delle IMI si trovano nei pacchetti e . La classe è importante per la gestione dei caratteri della tastiera. [android.inputmethodservice](https://developer.android.com/reference/android/inputmethodservice/package-summary)[android.view.inputmethod](https://developer.android.com/reference/android/view/inputmethod/package-summary)[KeyEvent](https://developer.android.com/reference/android/view/KeyEvent)

La parte centrale di un IME è un componente di servizio, una classe che estende . Oltre a implementare il normale ciclo di vita del servizio, questa classe dispone di callback per fornire l'interfaccia utente dell'IME, gestire l'input dell'utente e fornire testo al campo attualmente attivo. Per impostazione predefinita, la classe fornisce la maggior parte dell'implementazione per la gestione dello stato e della visibilità dell'IME e la comunicazione con il campo di input corrente. [InputMethodServiceInputMethodService](https://developer.android.com/reference/android/inputmethodservice/InputMethodService)

Importanti sono anche le seguenti classi:

[BaseInputConnection](https://developer.android.com/reference/android/view/inputmethod/BaseInputConnection)

Definisce il canale di comunicazione da un back all'applicazione che riceve il relativo input. Lo si usa per leggere il testo intorno al cursore, eseguire il commit del testo nella casella di testo e inviare eventi chiave non elaborati all'applicazione. Le applicazioni devono estendere questa classe anziché implementare l'interfaccia di base . [InputMethod](https://developer.android.com/reference/android/view/inputmethod/InputMethod)[InputConnection](https://developer.android.com/reference/android/view/inputmethod/InputConnection)

[KeyboardView](https://developer.android.com/reference/android/inputmethodservice/KeyboardView)

Estensione di che esegue il rendering di una tastiera e risponde agli eventi di input dell'utente. Il layout di tastiera è specificato da un'istanza di , che è possibile definire in un file XML. [View](https://developer.android.com/reference/android/view/View)[Keyboard](https://developer.android.com/reference/android/inputmethodservice/Keyboard)

## Progettare l'interfaccia utente del metodo di input

Esistono due elementi visivi principali per un IME: la visualizzazione **di input** e la **visualizzazione** candidati. Devi solo implementare gli elementi rilevanti per il metodo di input che stai progettando.

### Visualizzazione input

La visualizzazione di input è l'interfaccia utente in cui l'utente immette testo sotto forma di clic con tasti, grafia o gesti. Quando l'IME viene visualizzato per la prima volta, il sistema chiama il callback. Nell'implementazione di questo metodo si crea il layout che si desidera visualizzare nella finestra IME e si restituisce il layout al sistema. Questo frammento è un esempio di implementazione del metodo: [onCreateInputView()onCreateInputView()](https://developer.android.com/reference/android/inputmethodservice/InputMethodService#onCreateInputView())

[KOTLIN](https://developer.android.com/guide/topics/text/creating-input-method?hl=en#kotlin)

override fun onCreateInputView(): View {  
    return layoutInflater.inflate(R.layout.input, null).apply {  
        if (this is MyKeyboardView) {  
            setOnKeyboardActionListener(this@MyInputMethod)  
            keyboard = latinKeyboard  
        }  
    }  
}

In questo esempio, è un'istanza di un'implementazione personalizzata di che esegue il rendering di un oggetto . MyKeyboardView[KeyboardView](https://developer.android.com/reference/android/inputmethodservice/KeyboardView)[Keyboard](https://developer.android.com/reference/android/inputmethodservice/Keyboard)

### Visualizzazione Candidati

La visualizzazione candidati è l'interfaccia utente in cui l'IME visualizza potenziali correzioni di parole o suggerimenti da selezionare per l'utente. Nel ciclo di vita dell'IME, il sistema chiama quando è pronto per visualizzare la visualizzazione dei candidati. Nell'implementazione di questo metodo, restituire un layout che mostra i suggerimenti di parole o restituire null se non si desidera mostrare nulla. Una risposta null è il comportamento predefinito, quindi non è necessario implementarlo se non si forniscono suggerimenti.[onCreateCandidatesView()](https://developer.android.com/reference/android/inputmethodservice/InputMethodService#onCreateCandidatesView())

### Considerazioni sulla progettazione dell'interfaccia utente

Questa sezione descrive alcune considerazioni specifiche sulla progettazione dell'interfaccia utente per gli IMI.

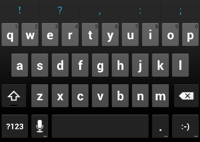
#### Gestire più dimensioni dello schermo

L'interfaccia utente per l'IME deve essere in grado di scalare in base alle diverse dimensioni dello schermo e deve anche gestire gli orientamenti orizzontale e verticale. In modalità IME non a schermo intero, lasciare spazio sufficiente all'applicazione per visualizzare il campo di testo e qualsiasi contesto associato, in modo che non più della metà dello schermo sia occupata dall'IME. In modalità IME a schermo intero questo non è un problema.

#### Gestire diversi tipi di input

I campi di testo android consentono di impostare un tipo di input specifico, ad esempio testo in formato libero, numeri, URL, indirizzi di posta elettronica e stringhe di ricerca. Quando si implementa un nuovo IME, è necessario rilevare il tipo di input di ogni campo e fornire l'interfaccia appropriata. Tuttavia, non è necessario configurare l'IME per verificare che l'utente abbia immesso testo valido per il tipo di input; questa è la responsabilità dell'applicazione che possiede il campo di testo.

Ad esempio, ecco screenshot delle interfacce fornite dall'IME latino con la piattaforma Android per gli input di testo e numeri di telefono:

**Figura 2.** Tipi di input IME latini.

Quando un campo di input riceve lo stato attivo e l'IME viene avviato, il sistema chiama , passando un oggetto contenente dettagli sul tipo di input e su altri attributi del campo di testo. In questo oggetto il campo contiene il tipo di input del campo di testo. [onStartInputView()](https://developer.android.com/reference/android/inputmethodservice/InputMethodService#onStartInputView(android.view.inputmethod.EditorInfo,%20boolean))[EditorInfo](https://developer.android.com/reference/android/view/inputmethod/EditorInfo)[inputType](https://developer.android.com/reference/android/view/inputmethod/EditorInfo#inputType)

Il campo è un oggetto che contiene modelli di bit per varie impostazioni del tipo di input. Per testarlo per il tipo di input del campo di testo, mascherarlo con la costante , in questo modo: [inputType](https://developer.android.com/reference/android/view/inputmethod/EditorInfo#inputType)int[TYPE\_MASK\_CLASS](https://developer.android.com/reference/android/text/InputType#TYPE_MASK_CLASS)

[KOTLIN](https://developer.android.com/guide/topics/text/creating-input-method?hl=en#kotlin)

inputType and InputType.TYPE\_MASK\_CLASS

Il modello di bit del tipo di input può avere uno dei diversi valori, tra cui:

[TYPE\_CLASS\_NUMBER](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_NUMBER)

Campo di testo per l'immissione di numeri. Come illustrato nella cattura di schermata precedente, l'IME latino visualizza un blocco numerico per campi di questo tipo.

[TYPE\_CLASS\_DATETIME](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_DATETIME)

Campo di testo per l'immissione di una data e di un'ora.

[TYPE\_CLASS\_PHONE](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_PHONE)

Campo di testo per l'immissione di numeri di telefono.

[TYPE\_CLASS\_TEXT](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_TEXT)

Campo di testo per l'immissione di tutti i caratteri supportati.

Queste costanti sono descritte più dettagliatamente nella documentazione di riferimento per . [InputType](https://developer.android.com/reference/android/text/InputType)

Il campo può contenere altri bit che indicano una variante del tipo di campo di testo, ad esempio: [inputType](https://developer.android.com/reference/android/view/inputmethod/EditorInfo#inputType)

[TYPE\_TEXT\_VARIATION\_PASSWORD](https://developer.android.com/reference/android/text/InputType#TYPE_TEXT_VARIATION_PASSWORD)

Una variante di per l'immissione di password. Il metodo di input visualizza i dingbat anziché il testo effettivo. [TYPE\_CLASS\_TEXT](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_TEXT)

[TYPE\_TEXT\_VARIATION\_URI](https://developer.android.com/reference/android/text/InputType#TYPE_TEXT_VARIATION_URI)

Una variante di per l'immissione di URL Web e altri URI (Uniform Resource Identifier). [TYPE\_CLASS\_TEXT](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_TEXT)

[TYPE\_TEXT\_FLAG\_AUTO\_COMPLETE](https://developer.android.com/reference/android/text/InputType#TYPE_TEXT_FLAG_AUTO_COMPLETE)

Una variante di per l'immissione di testo che l'applicazione "completa automaticamente" da un dizionario, una ricerca o un'altra funzionalità. [TYPE\_CLASS\_TEXT](https://developer.android.com/reference/android/text/InputType#TYPE_CLASS_TEXT)

Ricordarsi di mascherare con la costante appropriata quando si testano queste varianti. Le costanti mask disponibili sono elencate nella documentazione di riferimento per . [inputType](https://developer.android.com/reference/android/view/inputmethod/EditorInfo#inputType)[InputType](https://developer.android.com/reference/android/text/InputType)

**Attenzione:** Nel proprio IME, assicurarsi di gestire correttamente il testo quando lo si invia a un campo password. Nascondi la password nell'interfaccia utente sia nella visualizzazione di input che nella visualizzazione candidati. Ricorda anche che non dovresti archiviare le password su un dispositivo. Per ulteriori informazioni, vedere la guida [Designing for Security.](https://developer.android.com/guide/practices/security)

## Inviare testo all'applicazione

Quando l'utente immette testo con l'IME, è possibile inviare testo all'applicazione inviando singoli eventi chiave o modificando il testo intorno al cursore nel campo di testo dell'applicazione. In entrambi i casi, si utilizza un'istanza di per consegnare il testo. Per ottenere questa istanza, chiamare . [InputConnection](https://developer.android.com/reference/android/view/inputmethod/InputConnection)[InputMethodService.getCurrentInputConnection()](https://developer.android.com/reference/android/inputmethodservice/InputMethodService#getCurrentInputConnection())

### Modificare il testo intorno al cursore

Quando si gestisce la modifica del testo esistente in un campo di testo, alcuni dei metodi più utili in sono: [BaseInputConnection](https://developer.android.com/reference/android/view/inputmethod/BaseInputConnection)

[getTextBeforeCursor()](https://developer.android.com/reference/android/view/inputmethod/BaseInputConnection#getTextBeforeCursor(int,%20int))

Restituisce un oggetto contenente il numero di caratteri richiesti prima della posizione corrente del cursore. [CharSequence](https://developer.android.com/reference/java/lang/CharSequence)

[getTextAfterCursor()](https://developer.android.com/reference/android/view/inputmethod/BaseInputConnection#getTextAfterCursor(int,%20int))

Restituisce un oggetto contenente il numero di caratteri richiesti dopo la posizione corrente del cursore. [CharSequence](https://developer.android.com/reference/java/lang/CharSequence)

[deleteSurroundingText()](https://developer.android.com/reference/android/view/inputmethod/BaseInputConnection#deleteSurroundingText(int,%20int))

Elimina il numero specificato di caratteri prima e dopo la posizione corrente del cursore.

[commitText()](https://developer.android.com/reference/android/view/inputmethod/BaseInputConnection#commitText(java.lang.CharSequence,%20int))

Eseguire il commit di un oggetto nel campo di testo e impostare una nuova posizione del cursore. [CharSequence](https://developer.android.com/reference/java/lang/CharSequence)

Ad esempio, il frammento seguente mostra come sostituire i quattro caratteri a sinistra del cursore con il testo "Hello!":

[KOTLIN](https://developer.android.com/guide/topics/text/creating-input-method?hl=en#kotlin)

currentInputConnection.also { ic: InputConnection ->  
    ic.deleteSurroundingText(4, 0)  
    ic.commitText("Hello", 1)  
    ic.commitText("!", 1)  
}

### Composizione del testo prima del commit

Se l'IME esegue la previsione del testo o richiede più passaggi per comporre un glifo o una parola, è possibile visualizzare lo stato di avanzamento nel campo di testo fino a quando l'utente non esegue il commit della parola, quindi è possibile sostituire la composizione parziale con il testo completato. È possibile fornire un trattamento speciale al testo aggiungendo un "span" ad esso quando lo si passa a . [setComposingText()](https://developer.android.com/reference/android/view/inputmethod/InputConnection#setComposingText(java.lang.CharSequence,%20int))

Il frammento seguente mostra come mostrare lo stato di avanzamento in un campo di testo:

[KOTLIN](https://developer.android.com/guide/topics/text/creating-input-method?hl=en#kotlin)

currentInputConnection.also { ic: InputConnection ->  
    ic.setComposingText("Composi", 1)  
    ic.setComposingText("Composin", 1)  
    ic.commitText("Composing ", 1)  
}

Le schermate seguenti mostrano come appare all'utente:

**Figura 3.** Composizione del testo prima del commit.

### Intercettare gli eventi chiave hardware

Anche se la finestra del metodo di input non ha lo stato attivo esplicito, riceve prima gli eventi chiave hardware e può scegliere di consumarli o inoltrarli all'applicazione. Ad esempio, potresti voler utilizzare i tasti direzionali per navigare all'interno dell'interfaccia utente per la selezione dei candidati durante la composizione. È inoltre possibile intercettare il tasto indietro per chiudere tutti i popup provenienti dalla finestra del metodo di input.

Per intercettare le chiavi hardware, eseguire l'override e . [onKeyDown()](https://developer.android.com/reference/android/inputmethodservice/InputMethodService#onKeyDown(int,%20android.view.KeyEvent))[onKeyUp()](https://developer.android.com/reference/android/inputmethodservice/InputMethodService#onKeyUp(int,%20android.view.KeyEvent))

Ricorda di chiamare il metodo per le chiavi che non vuoi gestire da solo. super()

## Creare un sottotipo IME

I sottotipi consentono all'IME di esporre più modalità di input e linguaggi supportati da un IME. Un sottotipo può rappresentare:

* Impostazioni locali, ad esempio en\_US o fr\_FR.
* Modalità di input, ad esempio voce, tastiera o grafia.
* Altri stili di input, moduli o proprietà specifici dell'IME, ad esempio layout di tastiera a 10 tasti o qwerty.

Fondamentalmente, la modalità può essere qualsiasi testo come "tastiera", "voce" e così via. Un sottotipo può anche esporre una combinazione di questi.

Le informazioni sul sottotipo vengono utilizzate per una finestra di dialogo dello switcher IME disponibile dalla barra delle notifiche e anche per le impostazioni IME. Le informazioni consentono inoltre al framework di far emergere direttamente un sottotipo specifico di un IME. Quando si crea un IME, utilizzare la funzione sottotipo, perché consente all'utente di identificare e passare da una lingua e da una modalità IME all'altra.

È possibile definire sottotipi in uno dei file di risorse XML del metodo di input utilizzando l'elemento . Il frammento seguente definisce un IME con due sottotipi: un sottotipo di tastiera per le impostazioni locali inglesi negli Stati Uniti e un altro sottotipo di tastiera per le impostazioni locali della lingua francese per la Francia: <subtype>

<input-method xmlns:android="http://schemas.android.com/apk/res/android"  
        android:settingsActivity="com.example.softkeyboard.Settings"  
        android:icon="@drawable/ime\_icon">  
    <subtype android:name="@string/display\_name\_english\_keyboard\_ime"  
            android:icon="@drawable/subtype\_icon\_english\_keyboard\_ime"  
            android:imeSubtypeLanguage="en\_US"  
            android:imeSubtypeMode="keyboard"  
            android:imeSubtypeExtraValue="somePrivateOption=true" />  
    <subtype android:name="@string/display\_name\_french\_keyboard\_ime"  
            android:icon="@drawable/subtype\_icon\_french\_keyboard\_ime"  
            android:imeSubtypeLanguage="fr\_FR"  
            android:imeSubtypeMode="keyboard"  
            android:imeSubtypeExtraValue="foobar=30,someInternalOption=false" />  
    <subtype android:name="@string/display\_name\_german\_keyboard\_ime" ... />  
</input-method>

Per assicurarsi che i sottotipi siano etichettati correttamente nell'interfaccia utente, utilizzare %s per ottenere un'etichetta di sottotipo uguale all'etichetta delle impostazioni locali del sottotipo. Questo è dimostrato nei prossimi due frammenti. Il primo frammento mostra parte del file XML del metodo di input:

<subtype  
    android:label="@string/label\_subtype\_generic"  
    android:imeSubtypeLocale="en\_US"  
    android:icon="@drawable/icon\_en\_us"  
    android:imeSubtypeMode="keyboard" />

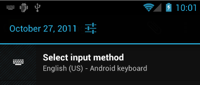
Il frammento successivo fa parte del file dell'IME. La risorsa stringa , utilizzata dalla definizione dell'interfaccia utente del metodo di input per impostare l'etichetta del sottotipo, è definita come: strings.xmllabel\_subtype\_generic

<string name="label\_subtype\_generic">%s</string>

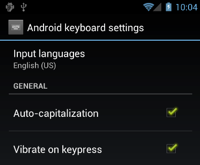
Questa impostazione fa sì che il nome visualizzato del sottotipo corrisponda all'impostazione delle impostazioni locali. Ad esempio, in qualsiasi lingua inglese, il nome visualizzato è "Inglese (Stati Uniti)".

### Scegliere i sottotipi IME dalla barra delle notifiche

Il sistema Android gestisce tutti i sottotipi esposti da tutti gli IMI. I sottotipi IME vengono trattati come modalità dell'IME a cui appartengono. Nella barra di notifica, un utente può selezionare un sottotipo disponibile per l'IME attualmente impostato, come illustrato nella schermata seguente:



**Figura 4.** Scelta di un sottotipo IME dalla barra di notifica.



**Figura 5.** Impostazione delle preferenze del sottotipo in Impostazioni di sistema.

### Scegliere sottotipi IME da Impostazioni di sistema

Un utente può controllare il modo in cui vengono utilizzati i sottotipi nel pannello impostazioni "Lingua e input" nell'area Impostazioni di sistema.



**Figura 6.** Scelta di una lingua per l'IME.

## Passare da un sottotipo IME all'altro

È possibile consentire agli utenti di passare facilmente da più sottotipi IME fornendo un tasto di commutazione, ad esempio l'icona della lingua a forma di globo, come parte della tastiera. Ciò migliora notevolmente l'usabilità della tastiera e può aiutare a evitare la frustrazione dell'utente. Per abilitare tale commutazione, eseguire la procedura seguente:

1. Dichiarare nei file di risorse XML del metodo di input. La dichiarazione dovrebbe essere simile al frammento seguente: supportsSwitchingToNextInputMethod = "true"

<input-method xmlns:android="http://schemas.android.com/apk/res/android"  
        android:settingsActivity="com.example.softkeyboard.Settings"  
        android:icon="@drawable/ime\_icon"  
        android:supportsSwitchingToNextInputMethod="true">

1. Chiamare il metodo .[shouldOfferSwitchingToNextInputMethod()](https://developer.android.com/reference/android/view/inputmethod/InputMethodManager#shouldOfferSwitchingToNextInputMethod(android.os.IBinder))
2. Se il metodo restituisce true, visualizzare un tasto di commutazione.
3. Quando l'utente tocca il tasto di commutazione, chiamare , passando false al secondo parametro. Un valore false indica al sistema di trattare tutti i sottotipi allo stesso modo, indipendentemente dall'IME a cui appartengono. Se si specifica true, il sistema deve scorrere i sottotipi nell'IME corrente.[switchToNextInputMethod()](https://developer.android.com/reference/android/view/inputmethod/InputMethodManager#switchToNextInputMethod(android.os.IBinder,%20boolean))

**Attenzione:** Prima di Android 5.0 (livello API 21), non è a conoscenza dell'attributo. Se l'utente passa a un IME senza un tasto di commutazione, potrebbe rimanere bloccato in quell'IME, incapace di spegnerlo facilmente.[**switchToNextInputMethod()**](https://developer.android.com/reference/android/view/inputmethod/InputMethodManager#switchToNextInputMethod(android.os.IBinder,%20boolean))**supportsSwitchingToNextInputMethod**

## Considerazioni generali sull'IME

Ecco alcune altre cose da considerare quando stai implementando il tuo IME:

* Offri agli utenti un modo per impostare le opzioni direttamente dall'interfaccia utente dell'IME.
* Poiché nel dispositivo possono essere installati più IME, fornire all'utente un modo per passare a un IME diverso direttamente dall'interfaccia utente del metodo di input.
* Visualizzare rapidamente l'interfaccia utente dell'IME. Precaricare o caricare su richiesta risorse di grandi dimensioni in modo che gli utenti vedano l'IME non appena toccano un campo di testo. Memorizzare nella cache le risorse e le visualizzazioni per le chiamate successive del metodo di input.
* Al contrario, è necessario rilasciare allocazioni di memoria di grandi dimensioni subito dopo che la finestra del metodo di input è nascosta, in modo che le applicazioni possano disporre di memoria sufficiente per l'esecuzione. Prendere in considerazione l'utilizzo di un messaggio ritardato per rilasciare risorse se l'IME si trova in uno stato nascosto per alcuni secondi.
* Assicurarsi che gli utenti possano immettere il maggior numero possibile di caratteri per la lingua o le impostazioni locali associate all'IME. Ricorda che gli utenti possono utilizzare la punteggiatura nelle password o nei nomi utente, quindi il tuo IME deve fornire molti caratteri diversi per consentire agli utenti di inserire una password e ottenere l'accesso al dispositivo.

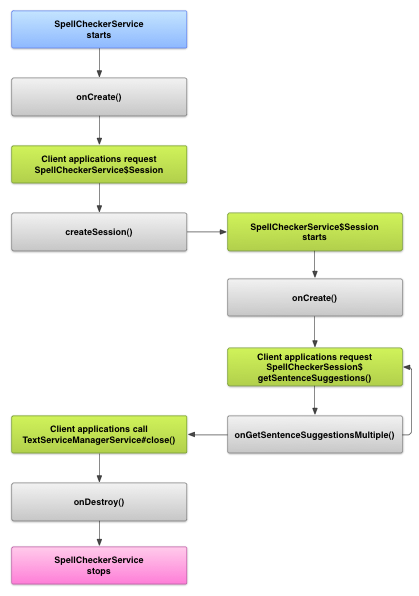
# Framework di controllo ortografico

La piattaforma Android offre un framework di controllo ortografico che ti consente di implementare e accedere al controllo ortografico nella tua applicazione. Il framework è una delle API del servizio di testo offerte dalla piattaforma Android.

Per usare il framework nella tua app, crei un tipo speciale di servizio Android che genera un oggetto sessione del **correttore ortografico.** In base al testo fornito, l'oggetto sessione restituisce suggerimenti ortografici generati dal correttore ortografico.

## Ciclo di vita del correttore ortografico

Il diagramma seguente mostra il ciclo di vita del servizio correttore ortografico:



**Figura 1.** Ciclo di vita del servizio controllo ortografico.

Per avviare il controllo ortografico, l'app avvia l'implementazione del servizio di controllo ortografico. I client dell'app, ad esempio attività o singoli elementi dell'interfaccia utente, richiedono una sessione di controllo ortografico al servizio, quindi usano la sessione per ottenere suggerimenti per il testo. Quando un client termina il funzionamento, chiude la sessione del correttore ortografico. Se necessario, l'app può arrestare il servizio di controllo ortografico in qualsiasi momento.

## Implementare un servizio di controllo ortografico

Per usare il framework del correttore ortografico nell'app, aggiungi un componente del servizio di controllo ortografico che include la definizione dell'oggetto sessione. Puoi anche aggiungere alla tua app un'attività facoltativa che controlla le impostazioni. È inoltre necessario aggiungere un file di metadati XML che descriva il servizio di controllo ortografico e aggiungere gli elementi appropriati al file manifesto.

### Classi di controllo ortografico

Definire l'oggetto servizio e sessione con le classi seguenti:

**Una sottoclasse di**[SpellCheckerService](https://developer.android.com/reference/android/service/textservice/SpellCheckerService)

Implementa sia la classe che l'interfaccia del framework del correttore ortografico. All'interno della sottoclasse è necessario implementare il metodo seguente: [SpellCheckerService](https://developer.android.com/reference/android/service/textservice/SpellCheckerService)[Service](https://developer.android.com/reference/android/app/Service)

[createSession()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService#createSession())

Metodo factory che restituisce un oggetto a un client che desidera eseguire il controllo ortografico. [SpellCheckerService.Session](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session)

Vedi l'app [di esempio Spell Checker Service](https://developer.android.com/resources/samples/SpellChecker/SampleSpellCheckerService) per saperne di più sull'implementazione di questa classe.

**L'attuazione di**[SpellCheckerService.Session](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session)

Oggetto fornito dal servizio di controllo ortografico ai client, per consente loro di passare testo al correttore ortografico e ricevere suggerimenti. All'interno di questa classe, è necessario implementare i metodi seguenti:

[onCreate()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session#onCreate())

Chiamato dal sistema in risposta a . In questo metodo è possibile inizializzare l'oggetto in base alle impostazioni locali correnti e così via. [createSession()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService#createSession())[SpellCheckerService.Session](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session)

[onGetSentenceSuggestionsMultiple()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session#onGetSentenceSuggestionsMultiple(android.view.textservice.TextInfo[],%20int))

Fa il controllo ortografico effettivo. Questo metodo restituisce una matrice di suggerimenti contenenti per le frasi passate. [SentenceSuggestionsInfo](https://developer.android.com/reference/android/view/textservice/SentenceSuggestionsInfo)

Facoltativamente, è possibile implementare , che gestisce le richieste di annullamento del controllo ortografico, , che gestisce una richiesta di suggerimento di parole, o , che gestisce batch di richieste di suggerimento di parole. [onCancel()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session#onCancel())[onGetSuggestions()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session#onGetSuggestions(android.view.textservice.TextInfo,%20int))[onGetSuggestionsMultiple()](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session#onGetSuggestionsMultiple(android.view.textservice.TextInfo[],%20int,%20boolean))

Vedi l'app [di esempio Spell Checker Client](https://developer.android.com/resources/samples/SpellChecker/HelloSpellChecker) per saperne di più sull'implementazione di questa classe.

**Nota:** È necessario implementare tutti gli aspetti del controllo ortografico come asincroni e thread-safe. Un correttore ortografico può essere chiamato contemporaneamente da thread diversi in esecuzione su core diversi. E prendersi cura di questo automaticamente. [**SpellCheckerService**](https://developer.android.com/reference/android/service/textservice/SpellCheckerService)[**SpellCheckerService.Session**](https://developer.android.com/reference/android/service/textservice/SpellCheckerService.Session)

### Manifesto e metadati del correttore ortografico

Oltre al codice, è necessario fornire il file manifesto appropriato e un file di metadati per il correttore ortografico.

Il file manifesto definisce l'applicazione, il servizio e l'attività per il controllo delle impostazioni, come illustrato nel frammento seguente:

<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.example.android.samplespellcheckerservice" >  
    <application  
        android:label="@string/app\_name" >  
        <service  
            android:label="@string/app\_name"  
            android:name=".SampleSpellCheckerService"  
            android:permission="android.permission.BIND\_TEXT\_SERVICE" >  
            <intent-filter >  
                <action android:name="android.service.textservice.SpellCheckerService" />  
            </intent-filter>  
  
            <meta-data  
                android:name="android.view.textservice.scs"  
                android:resource="@xml/spellchecker" />  
        </service>  
  
        <activity  
            android:label="@string/sample\_settings"  
            android:name="SpellCheckerSettingsActivity" >  
            <intent-filter >  
                <action android:name="android.intent.action.MAIN" />  
            </intent-filter>  
        </activity>  
    </application>  
</manifest>

Si noti che i componenti che desiderano utilizzare il servizio devono richiedere l'autorizzazione per assicurarsi che solo il sistema si associa al servizio. La definizione del servizio specifica anche il file di metadati, descritto nella sezione successiva. [BIND\_TEXT\_SERVICE](https://developer.android.com/reference/android/Manifest.permission#BIND_TEXT_SERVICE)spellchecker.xml

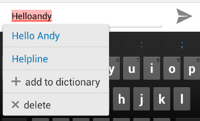
Il file di metadati contiene il codice XML seguente: spellchecker.xml

<spell-checker xmlns:android="http://schemas.android.com/apk/res/android"  
        android:label="@string/spellchecker\_name"  
        android:settingsActivity="com.example.SpellCheckerSettingsActivity">  
    <subtype  
            android:label="@string/subtype\_generic"  
            android:subtypeLocale="en”  
    />  
    <subtype  
            android:label="@string/subtype\_generic"  
            android:subtypeLocale="fr”  
    />  
</spell-checker>

I metadati specificano l'attività utilizzata dal correttore ortografico per il controllo delle impostazioni. Definisce inoltre i sottotipi per il correttore ortografico; in questo caso, i sottotipi definiscono le impostazioni locali che il correttore ortografico può gestire.

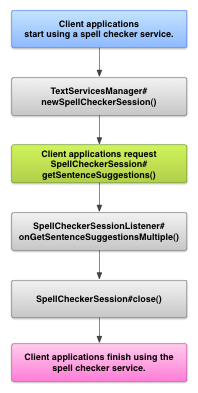
## Accedere al servizio di controllo ortografico da un client

Le applicazioni che usano le visualizzazioni beneficiano automaticamente del controllo ortografico, perché usa automaticamente un correttore ortografico. Le schermate seguenti mostrano questo: [TextViewTextView](https://developer.android.com/reference/android/widget/TextView)

**Figura 2.** Controllo ortografico in TextView.

Tuttavia, potresti voler interagire direttamente con un servizio di controllo ortografico anche in altri casi. Il diagramma seguente mostra il flusso di controllo per l'interazione con un servizio di controllo ortografico:



**Figura 3.** Interagire con un servizio di controllo ortografico.

[L'editor di metodi di input LatinIME in Android Open Source Project contiene](https://android.googlesource.com/platform/packages/inputmethods/LatinIME/) un esempio di controllo ortografico.