Descripción general de los eventos de entrada

En Android, existe más de una forma de interceptar los eventos desde una interacción del usuario con tu aplicación. Al considerar los eventos dentro de tu interfaz de usuario, el enfoque consiste en capturar los eventos desde el objeto de vista específico con el que interactúa el usuario. La clase de vista proporciona los medios para hacerlo.

Dentro de las diversas clases de vista que utilizarás para componer tu diseño, quizá observes varios métodos de devolución de llamada públicos que pueden ser útiles para eventos de IU. Estos métodos son llamados por el marco de trabajo de Android cuando la acción respectiva ocurre en ese objeto. Por ejemplo, cuando se toca una vista (por ejemplo, un botón), se llama al método onTouchEvent() en ese objeto. Sin embargo, para interceptar esto, debes extender la clase y reemplazar el método. No obstante, extender todos los objetos de vista para manejar tal evento no sería práctico. Por eso, la clase de vista también contiene una colección de interfaces anidadas con devoluciones de llamada que puedes definir más fácilmente. Estas interfaces, llamadas [receptores de eventos](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#EventListeners), te permiten capturar la interacción del usuario con tu IU.

Si bien generalmente utilizarás los receptores de eventos para escuchar la interacción del usuario, quizás en algún momento desees extender una clase de vista a fin de crear un componente personalizado. Quizás desees extender la clase [Button](https://developer.android.com/reference/android/widget/Button.html?hl=es-419) para crear algo más elaborado. En este caso, podrás definir los comportamientos de eventos predeterminados para tu clase utilizando los [controladores de eventos](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#EventHandlers) de la clase.

Receptores de eventos

Un receptor de eventos es una interfaz de la clase [View](https://developer.android.com/reference/android/view/View.html?hl=es-419) que contiene un solo método de devolución de llamada. A estos métodos los llamará el marco de trabajo de Android cuando la vista con la que se registró el receptor sea iniciada por la interacción del usuario con el elemento de la IU.

En las interfaces de los receptores de eventos, se incluyen los siguientes métodos de devolución de llamada:

onClick()

Desde [View.OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html?hl=es-419). Se llama a este método cuando el usuario toca el elemento (en el modo táctil), o selecciona el elemento con las teclas de navegación o la bola de seguimiento y presiona la tecla Intro o la bola de seguimiento adecuada.

onLongClick()

Desde [View.OnLongClickListener](https://developer.android.com/reference/android/view/View.OnLongClickListener.html?hl=es-419). Se llama a este método cuando el usuario toca y mantiene presionado el elemento (en el modo táctil), o selecciona el elemento con las teclas de navegación o la bola de seguimiento y mantiene presionada la tecla Intro o la bola de seguimiento adecuada (durante un segundo).

onFocusChange()

Desde [View.OnFocusChangeListener](https://developer.android.com/reference/android/view/View.OnFocusChangeListener.html?hl=es-419). Se llama a este método cuando el usuario navega hacia el elemento o sale de él utilizando las teclas de navegación o la bola de seguimiento.

onKey()

Desde [View.OnKeyListener](https://developer.android.com/reference/android/view/View.OnKeyListener.html?hl=es-419). Se llama a este método cuando el usuario se centra en el elemento y presiona o suelta una tecla física del dispositivo.

onTouch()

Desde [View.OnTouchListener](https://developer.android.com/reference/android/view/View.OnTouchListener.html?hl=es-419). Se llama a este método cuando el usuario realiza una acción calificada como un evento táctil, por ejemplo, presionar, soltar o cualquier gesto de movimiento en la pantalla (dentro de los límites del elemento).

onCreateContextMenu()

Desde [View.OnCreateContextMenuListener](https://developer.android.com/reference/android/view/View.OnCreateContextMenuListener.html?hl=es-419). Se llama a este método cuando se crea un menú contextual (como resultado de un "clic largo" sostenido). Consulta la explicación sobre menús contextuales en la guía para desarrolladores [Menús](https://developer.android.com/guide/topics/ui/menus.html?hl=es-419#context-menu).

Estos métodos son los únicos habitantes de su respectiva interfaz. Para definir uno de estos métodos y manejar sus eventos, implementa la interfaz anidada en su actividad o defínela como una clase anónima. Luego, pasa una instancia de tu implementación al método View.set...Listener() respectivo. (Por ejemplo, llama a [setOnClickListener()](https://developer.android.com/reference/android/view/View.html?hl=es-419#setOnClickListener(android.view.View.OnClickListener)) y pásale tu implementación de [OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html?hl=es-419)).

En el ejemplo siguiente, se muestra cómo registrar un receptor onClick para un botón.

[KOTLIN](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#kotlin)[JAVA](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#java)

// Create an anonymous implementation of OnClickListener  
private OnClickListener corkyListener = new OnClickListener() {  
    public void onClick(View v) {  
      // do something when the button is clicked  
    }  
};  
  
protected void onCreate(Bundle savedValues) {  
    ...  
    // Capture our button from layout  
    Button button = (Button)findViewById(R.id.corky);  
    // Register the onClick listener with the implementation above  
    button.setOnClickListener(corkyListener);  
    ...  
}

Quizás también te parezca más conveniente implementar OnClickListener como parte de tu actividad. Esto evitará la carga extra de la clase y la asignación de objetos. Por ejemplo:

[KOTLIN](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#kotlin)[JAVA](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#java)

public class ExampleActivity extends Activity implements OnClickListener {  
    protected void onCreate(Bundle savedValues) {  
        ...  
        Button button = (Button)findViewById(R.id.corky);  
        button.setOnClickListener(this);  
    }  
  
    // Implement the OnClickListener callback  
    public void onClick(View v) {  
      // do something when the button is clicked  
    }  
    ...  
}

Ten en cuenta que la devolución de llamada onClick() del ejemplo anterior no tiene un valor para mostrar, pero algunos otros métodos de receptores de eventos deben mostrar un valor booleano. El motivo depende del evento. A continuación, se explican los motivos de los pocos casos que lo hacen:

* [onLongClick()](https://developer.android.com/reference/android/view/View.OnLongClickListener.html?hl=es-419#onLongClick(android.view.View)): Este método muestra un valor booleano para indicar si has consumido el evento y si no debe continuar. Es decir, muestra un valor *true* para indicar que has usado el evento y que debe detenerse aquí, y muestra *false* si no has usado el evento o si el evento debe continuar para otros receptores de clic.
* [onKey()](https://developer.android.com/reference/android/view/View.OnKeyListener.html?hl=es-419#onKey(android.view.View,%20int,%20android.view.KeyEvent)): Este método muestra un valor booleano para indicar si has consumido el evento y si no debe continuar. Es decir, muestra un valor *true* para indicar que has usado el evento y que debe detenerse aquí, y muestra *false* si no has usado el evento o si el evento debe continuar para otros receptores de tecla.
* [onTouch()](https://developer.android.com/reference/android/view/View.OnTouchListener.html?hl=es-419#onTouch(android.view.View,%20android.view.MotionEvent)): Este método muestra un valor booleano para indicar si tu receptor consume este evento. Lo importante es que este evento puede tener varias acciones una después de la otra. Por lo tanto, si muestra *false* cuando se recibe el evento de acción de abajo, tú indicas que no has consumido el evento y también que no estás interesado en las acciones subsiguientes de este evento. Por ende, no se te llamará para otras acciones dentro del evento, como un gesto de dedo, o el evento de acción de arriba final.

Recuerda que los eventos de teclas de hardware siempre se entregan a la vista actualmente en foco. Se distribuyen comenzando desde la parte superior de la jerarquía de vistas y luego hacia abajo, hasta llegar al destino correspondiente. Si tu vista (o un elemento secundario de tu vista) actualmente tiene foco, puedes ver cómo pasa el evento a través del método [dispatchKeyEvent()](https://developer.android.com/reference/android/view/View.html?hl=es-419#dispatchKeyEvent(android.view.KeyEvent)). Como alternativa a capturar eventos de teclas mediante tu vista, también puedes recibir todos los eventos dentro de tu actividad con [onKeyDown()](https://developer.android.com/reference/android/app/Activity.html?hl=es-419#onKeyDown(int,%20android.view.KeyEvent)) y [onKeyUp()](https://developer.android.com/reference/android/app/Activity.html?hl=es-419#onKeyUp(int,%20android.view.KeyEvent)).

Además, al considerar la entrada de texto para tu aplicación, recuerda que muchos dispositivos solo tienen métodos de entrada de software. No es necesario que esos métodos se basen en teclas; algunos pueden utilizar entrada de voz, escritura a mano, etc. Incluso si un método de entrada presenta una interfaz similar a un teclado, generalmente **no** desencadenará la familia de eventos [onKeyDown()](https://developer.android.com/reference/android/app/Activity.html?hl=es-419#onKeyDown(int,%20android.view.KeyEvent)). Nunca debes construir una IU que requiera que se presionen teclas específicas para controlarla, salvo que desees limitar tu aplicación a dispositivos con un teclado de hardware. En particular, no utilices estos métodos para validar la entrada cuando el usuario presiona la tecla de entrada; en cambio, utiliza acciones como [IME\_ACTION\_DONE](https://developer.android.com/reference/android/view/inputmethod/EditorInfo.html?hl=es-419#IME_ACTION_DONE) para señalar al método de entrada cómo espera reaccionar tu aplicación a fin de que pueda cambiar su IU de forma significativa. Evita los supuestos sobre la forma en la que un método de entrada de software debe funcionar y simplemente confía en él para proporcionar texto ya formateado a tu aplicación.

**Nota:** Android llamará en primer lugar a los controladores de eventos y en segundo lugar a los controladores predeterminados correspondientes de la definición de clase. Por lo tanto, al mostrar *true* desde estos receptores de eventos, se detendrá la propagación del evento a otros receptores y también se bloqueará la devolución de llamada al controlador de eventos predeterminado en la vista. Por eso, asegúrate de que desees finalizar el evento cuando muestres *true*.

Controladores de eventos

Si estás creando un componente personalizado desde un elemento de vista, podrás definir varios métodos de devolución de llamada utilizados como controladores de eventos predeterminados. En el documento [Componentes personalizados de vistas](https://developer.android.com/guide/topics/ui/custom-components.html?hl=es-419), aprenderás sobre algunas de las devoluciones de llamada que se utilizan comúnmente para controlar eventos, incluidas las siguientes:

* [onKeyDown(int, KeyEvent)](https://developer.android.com/reference/android/view/View.html?hl=es-419#onKeyDown(int,%20android.view.KeyEvent)): Se llama cuando se produce un nuevo evento de tecla.
* [onKeyUp(int, KeyEvent)](https://developer.android.com/reference/android/view/View.html?hl=es-419#onKeyUp(int,%20android.view.KeyEvent)): Se llama cuando se produce un evento KeyUp.
* [onTrackballEvent(MotionEvent)](https://developer.android.com/reference/android/view/View.html?hl=es-419#onTrackballEvent(android.view.MotionEvent)): Se llama cuando se produce un evento de movimiento de la bola de seguimiento.
* [onTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/View.html?hl=es-419#onTouchEvent(android.view.MotionEvent)): Se llama cuando se produce un evento de movimiento de la pantalla táctil.
* [onFocusChanged(boolean, int, Rect)](https://developer.android.com/reference/android/view/View.html?hl=es-419#onFocusChanged(boolean,%20int,%20android.graphics.Rect)): Se llama cuando la vista gana o pierde foco.

Existen otros métodos que debes conocer, que no forman parte de la clase de vista, pero que pueden afectar directamente la forma en la que puedes manejar eventos. Por ende, al administrar eventos más complejos dentro de un diseño, considera estos otros métodos:

* [Activity.dispatchTouchEvent(MotionEvent)](https://developer.android.com/reference/android/app/Activity.html?hl=es-419#dispatchTouchEvent(android.view.MotionEvent)): Este método permite que tu [Activity](https://developer.android.com/reference/android/app/Activity.html?hl=es-419) intercepte todos los eventos táctiles antes de que se envíen a la ventana.
* [ViewGroup.onInterceptTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/ViewGroup.html?hl=es-419#onInterceptTouchEvent(android.view.MotionEvent)): Este método permite que un [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html?hl=es-419) observe los eventos mientras se envían a las vistas secundarias.
* [ViewParent.requestDisallowInterceptTouchEvent(boolean)](https://developer.android.com/reference/android/view/ViewParent.html?hl=es-419#requestDisallowInterceptTouchEvent(boolean)): Llama a este método en una vista primaria para indicar que no debes interceptar eventos táctiles con [onInterceptTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/ViewGroup.html?hl=es-419#onInterceptTouchEvent(android.view.MotionEvent)).

Modo táctil

Cuando un usuario navega en una interfaz de usuario con teclas direccionales o una bola de seguimiento, es necesario poner el foco en los elementos que se pueden accionar (como los botones) para que el usuario pueda ver lo que aceptará una entrada. Sin embargo, si el dispositivo tiene funciones táctiles y el usuario comienza a interactuar con la interfaz tocándola, ya no es necesario destacar los elementos ni poner el foco en una vista determinada. Por ende, existe un modo de interacción denominado "modo táctil".

En el caso de un dispositivo con funciones táctiles, una vez que el usuario toca la pantalla, el dispositivo entra en el modo táctil. A partir de este momento, solo serán enfocables las vistas para las cuales [isFocusableInTouchMode()](https://developer.android.com/reference/android/view/View.html?hl=es-419#isFocusableInTouchMode()) sea verdadero (por ejemplo, los widgets de edición de texto). Otras vistas que son táctiles, como los botones, no se enfocarán cuando se las toque; simplemente, activarán sus receptores onClick cuando se las presione.

Cuando un usuario toque una tecla direccional o se desplace con una bola de seguimiento, el dispositivo saldrá del modo táctil y encontrará una vista para poner el foco. Ahora, el usuario puede reanudar la interacción con la interfaz de usuario sin tocar la pantalla.

El estado del modo táctil se mantiene en todo el sistema (todas las ventanas y actividades). Para consultar el estado actual, puedes llamar a [isInTouchMode()](https://developer.android.com/reference/android/view/View.html?hl=es-419#isInTouchMode()) para ver si el dispositivo está en el modo táctil.

Cómo manejar el foco

El marco de trabajo manejará el movimiento del foco de rutina en respuesta a las entradas del usuario. Esto incluye cambiar el foco a medida que se quiten o se oculten las vistas, o a medida que haya nuevas vistas disponibles. Las vistas indican su disposición a permanecer enfocadas a través del método [isFocusable()](https://developer.android.com/reference/android/view/View.html?hl=es-419#isFocusable()). Para cambiar si una vista puede permanecer enfocada o no, llama a [setFocusable()](https://developer.android.com/reference/android/view/View.html?hl=es-419#setFocusable(boolean)). En el modo táctil, puedes consultar si una vista permite el foco con [isFocusableInTouchMode()](https://developer.android.com/reference/android/view/View.html?hl=es-419#isFocusableInTouchMode()). Puedes cambiar esto con [setFocusableInTouchMode()](https://developer.android.com/reference/android/view/View.html?hl=es-419#setFocusableInTouchMode(boolean)).

En dispositivos que ejecutan Android 9 (nivel de API 28) o una versión posterior, las actividades no asignan un foco inicial. En cambio, debes solicitar un foco inicial de forma explícita si lo deseas.

El movimiento del foco se basa en un algoritmo que encuentra el vecino más próximo en una dirección determinada. En casos poco frecuentes, es posible que el algoritmo predeterminado no coincida con el comportamiento previsto del desarrollador. En estas situaciones, puedes proporcionar anulaciones explícitas con los siguientes atributos XML en el archivo de diseño: ***nextFocusDown***, ***nextFocusLeft***, ***nextFocusRight*** y ***nextFocusUp***. Agrega uno de estos atributos a la vista *desde* la que se elimina el foco. Define el valor del atributo que será el ID de la vista *en* la que se pondrá el foco. Por ejemplo:

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

Generalmente, en este diseño vertical, al navegar hacia arriba desde el primer botón, no se llegaría a ningún lado, como tampoco navegando hacia abajo desde el segundo botón. Ahora que el botón superior ha definido al botón inferior como ***nextFocusUp*** (y viceversa), el foco de navegación pasará de arriba hacia abajo y de abajo hacia arriba.

Si deseas declarar una vista como enfocable en tu IU (cuando tradicionalmente no lo es), agrega el atributo XML android:focusable a la vista en tu declaración de diseño. Establece el valor ***true***. También puedes declarar una vista como enfocable en el modo táctil con android:focusableInTouchMode.

Para solicitar que se enfoque una vista en particular, llama a [requestFocus()](https://developer.android.com/reference/android/view/View.html?hl=es-419#requestFocus()).

Para escuchar eventos de foco (ser notificado cuando una vista recibe o pierde foco), utiliza [onFocusChange()](https://developer.android.com/reference/android/view/View.OnFocusChangeListener.html?hl=es-419#onFocusChange(android.view.View,%20boolean)), como se explica en la sección [Receptores de eventos](https://developer.android.com/guide/topics/ui/ui-events.html?hl=es-419#EventListeners).