События ввода

В системе Android предусмотрено несколько способов перехвата событий взаимодействия пользователя с вашим приложением. Что касается событий в пользовательском интерфейсе, подход состоит в том, чтобы захватывать события из определенного представления объекта, с которым взаимодействует пользователь. Класс отображаемых объектов (View) содержит средства, позволяющие делать это.

В различных классах View, которые вы будете использовать для создания макета, вы заметите несколько общедоступных методов обратного вызова, полезных для работы с событиями пользовательского интерфейса. Эти методы вызываются платформой Android, когда с этим объектом выполняется соответствующее действие. Например, когда пользователь касается отображаемого объекта (кнопки), вызывается метод onTouchEvent() этого объекта. Однако для перехвата этого события вы должны наследовать класс и переопределить метод. Однако наследование каждого отображаемого объекта для обработки такого события было бы неудобно. Именно поэтому класс View также содержит набор вложенных интерфейсов с обратными вызовами, которые можно задать значительно проще. Эти интерфейсы, которые называются [приемниками событий](https://developer.android.com/guide/topics/ui/ui-events.html#EventListeners), и служат перехватчиками действий пользователя с вашим пользовательским интерфейсом.

Несмотря на то, что вы будете чаще использовать приемники событий для перехвата действий пользователя, может наступить момент, когда вам не захочется наследовать класс View, чтобы создать нестандартный компонент. Возможно, вы захотите наследовать класс[Button](https://developer.android.com/reference/android/widget/Button.html), чтобы сделать нечто более необычное. В этом случае вы сможете определить поведение события по умолчанию для своего класса с помощью [обработчиков событий](https://developer.android.com/guide/topics/ui/ui-events.html#EventHandlers) класса.

Приемники событий

Приемник событий — это интерфейс в классе [View](https://developer.android.com/reference/android/view/View.html), который содержит один метод обратного вызова. Эти методы будут вызываться платформой Android, когда в результате взаимодействия пользователя с объектом пользовательского интерфейса срабатывает отображаемый объект View, в котором зарегистрирован приемник.

Интерфейсы, включенные в приемник события, представляют собой следующие методы обратного вызова:

onClick()

Из объекта [View.OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html). Этот метод вызывается, когда пользователь касается элемента (в режиме касания), или переводит фокус на элемент с помощью клавиш перемещения или трекбола и нажимает соответствующую клавишу «ввода» или нажимает на трекбол.

onLongClick()

Из объекта [View.OnLongClickListener](https://developer.android.com/reference/android/view/View.OnLongClickListener.html). Этот метод вызывается, когда пользователь касается элемента и удерживает его (в режиме касания), или переводит фокус на элемент с помощью клавиш перемещения или трекбола и нажимает и удерживает соответствующую клавишу «ввода» или трекбол (в течение одной секунды).

onFocusChange()

Из объекта [View.OnFocusChangeListener](https://developer.android.com/reference/android/view/View.OnFocusChangeListener.html). Этот метод вызывается, когда пользователь перемещается в элемент или из него с помощью клавиш перемещения или трекбола.

onKey()

Из объекта [View.OnKeyListener](https://developer.android.com/reference/android/view/View.OnKeyListener.html). Этот метод вызывается, когда пользователь переносит фокус на элемент и нажимает или отпускает аппаратную клавишу на устройстве.

onTouch()

Из объекта [View.OnTouchListener](https://developer.android.com/reference/android/view/View.OnTouchListener.html). Этот метод вызывается, когда пользователь выполняет действие, считающееся событием касания, например, нажимает, отпускает или выполняет любой жест на экране (в пределах границ элемента).

onCreateContextMenu()

Из объекта [View.OnCreateContextMenuListener](https://developer.android.com/reference/android/view/View.OnCreateContextMenuListener.html). Этот метод вызывается, когда создается контекстное меню (в результате длительного «длительного нажатия»). См. обсуждение контекстных меню в руководстве для разработчиков [Меню](https://developer.android.com/guide/topics/ui/menus.html#context-menu).

Эти методы являются единственными составными частями соответствующих интерфейсов. Чтобы определить один из этих методов и обрабатывать события, реализуйте вложенный интерфейс в вашем процесс или определите его, как анонимный класс. Затем передайте экземпляр реализации в соответствующий метод View.set...Listener(). (Например, вызовите [setOnClickListener()](https://developer.android.com/reference/android/view/View.html" \l "setOnClickListener(android.view.View.OnClickListener)) и передайте ему свою реализацию [OnClickListener](https://developer.android.com/reference/android/view/View.OnClickListener.html).)

В следующем примере показано, как зарегистрировать приемник события «по клику» (on-click) для кнопки.

// Create an anonymous implementation of OnClickListener  
private OnClickListener mCorkyListener = 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(mCorkyListener);  
    ...  
}

Возможно, еще удобнее реализовать OnClickListener как часть вашей операции. При этом исключается дополнительная нагрузка класса и выделение объекта. Например:

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  
    }  
    ...  
}

Обратите внимание, что обратный вызов onClick() в примере выше не содержит возвращаемого значения, но некоторые другие методы приемника события должны возвращать логическое значение. Причина зависит от события. Для некоторых методов, которые возвращают значения, причина описана ниже:

* [onLongClick()](https://developer.android.com/reference/android/view/View.OnLongClickListener.html#onLongClick(android.view.View)) — этот метод возвращает логическое значение, указывающее, что вы обработали это событие и его более не следует хранить. А именно, верните значение *true*, чтобы указать, что вы обработали событие и его следует остановить; верните значение *false*, если вы не обработали его и/или событие должно продолжаться для любых других приемников события on-click.
* [onKey()](https://developer.android.com/reference/android/view/View.OnKeyListener.html#onKey(android.view.View, int, android.view.KeyEvent)) — этот метод возвращает логическое значение, указывающее, что вы обработали это событие и его более не следует хранить. А именно, верните значение *true*, чтобы указать, что вы обработали событие и его следует остановить; верните значение *false*, если вы не обработали его и/или событие должно продолжаться для любых других приемников события on-click.
* [onTouch()](https://developer.android.com/reference/android/view/View.OnTouchListener.html#onTouch(android.view.View, android.view.MotionEvent)) — этот метод возвращает логическое значение, указывающее, обработал ли ваш приемник это событие. Важно, что это событие может повлечь несколько действий, следующих друг за другом. Поэтому, если вы возвращаете *ложь* при приеме события нажатия, вы указываете, что вы не обработали событие и не интересуетесь последующими действиями в результате этого события. Соответственно, этот метод не будет вызываться для любых других действий в рамках этого события, таких как жесты или возможное действие отпускания.

Помните, что события аппаратных клавиш всегда попадают в отображаемый объект View, который находится в фокусе. Они отправляются, начиная с верха в иерархии отображаемых объектов, затем ниже, пока не достигнут соответствующего назначения. Если ваш отображаемый объект (или дочерний объект вашего отображаемого объекта) находится в фокусе, вы можете видеть, как событие перемещается сквозь метод [dispatchKeyEvent()](https://developer.android.com/reference/android/view/View.html" \l "dispatchKeyEvent(android.view.KeyEvent)). В качестве альтернативы к перехвату событий клавиш через отображаемый объект View можно также получать все события внутри вашей операции с помощью методов [onKeyDown()](https://developer.android.com/reference/android/app/Activity.html" \l "onKeyDown(int, android.view.KeyEvent)) и [onKeyUp()](https://developer.android.com/reference/android/app/Activity.html" \l "onKeyUp(int, android.view.KeyEvent)).

Кроме того, размышляя о вводе текста для приложения, помните, что многие устройства поддерживают только программные методы ввода. Такие методы не обязательно основаны на нажатиях клавиш. Некоторые могут использовать голосовой ввод, рукописный ввод и т. д. Даже если метод ввода представлен интерфейсом, подобным клавиатуре, он обычно **не** приводит к запуску семейства событий[onKeyDown()](https://developer.android.com/reference/android/app/Activity.html#onKeyDown(int, android.view.KeyEvent)). Не следует создавать пользовательский интерфейс, для управления которым требуется нажимать определенные клавиши, кроме случаев, когда вы хотите ограничить использование приложения только устройствами с аппаратной клавиатурой. В частности, не полагайтесь на эти методы для подтверждения ввода, когда пользователь нажимает клавишу «Ввод». Вместо этого используйте такие действия, как [IME\_ACTION\_DONE](https://developer.android.com/reference/android/view/inputmethod/EditorInfo.html#IME_ACTION_DONE), чтобы подать методу ввода сигнал о том, какой реакции ожидает приложение, чтобы он мог соответственно изменить свой пользовательский интерфейс. Избегайте предположений о работе программного метода ввода и доверьте ему ввод уже отформатированного текста в ваше приложение.

**Примечание.** Система Android сначала будет вызывать обработчики событий, а затем соответствующие обработчики по умолчанию из определения класса. Поэтому возврат значения *истина* из этих приемников событий будет останавливать передачу события остальным приемникам событий, а также будет блокировать обратный вызов обработчика событий по умолчанию в отображаемом объекте. Поэтому проверьте, что вы хотите прервать событие, когда вы возвращаете значение *true*.

Обработчики событий

Если вы создаете нестандартный компонент из отображаемого объекта, вы сможете определить несколько методов обратного вызова, используемых как обработчики события по умолчанию. В документе [Нестандартные компоненты](https://developer.android.com/guide/topics/ui/custom-components.html) можно узнать о некоторых общих обратных вызовах, используемых для обработки событий, включая следующие:

* [onKeyDown(int, KeyEvent)](https://developer.android.com/reference/android/view/View.html#onKeyDown(int, android.view.KeyEvent)) — вызывается при возникновении нового события клавиши.
* [onKeyUp(int, KeyEvent)](https://developer.android.com/reference/android/view/View.html#onKeyUp(int, android.view.KeyEvent)) — вызывается при возникновении события отпускания клавиши.
* [onTrackballEvent(MotionEvent)](https://developer.android.com/reference/android/view/View.html#onTrackballEvent(android.view.MotionEvent)) — вызывается при возникновении события перемещения трекбола.
* [onTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/View.html#onTouchEvent(android.view.MotionEvent)) — вызывается при возникновении события жеста на сенсорном экране.
* [onFocusChanged(boolean, int, Rect)](https://developer.android.com/reference/android/view/View.html#onFocusChanged(boolean, int, android.graphics.Rect)) — вызывается, когда отображаемый объект получает или теряет фокус.

Существует несколько других методов, о которых следует знать и которые не являются частью класса View, но могут напрямую влиять на доступные вам способы обработки событий. Поэтому, управляя более сложными событиями внутри макета, учитывайте и другие методы:

* [Activity.dispatchTouchEvent(MotionEvent)](https://developer.android.com/reference/android/app/Activity.html#dispatchTouchEvent(android.view.MotionEvent)) — этот метод позволяет вашей операции [Activity](https://developer.android.com/reference/android/app/Activity.html) перехватывать все события касаний перед их отправкой в окно.
* [ViewGroup.onInterceptTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/ViewGroup.html#onInterceptTouchEvent(android.view.MotionEvent)) — этот метод позволяет объекту [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup.html) просматривать события перед их отправкой в дочерние отображаемые объекты.
* [ViewParent.requestDisallowInterceptTouchEvent(boolean)](https://developer.android.com/reference/android/view/ViewParent.html#requestDisallowInterceptTouchEvent(boolean)) — вызовите этот метод в родительском отображаемом объекте, чтобы указать ему, что он не должен перехватывать события касания с помощью [onInterceptTouchEvent(MotionEvent)](https://developer.android.com/reference/android/view/ViewGroup.html" \l "onInterceptTouchEvent(android.view.MotionEvent)).

Режим касания

Когда пользователь перемещается по пользовательскому интерфейсу с помощью клавиш перемещения или трекбола, необходимо передавать фокус действующим элементам (таким как кнопки), чтобы пользователь мог видеть, какой элемент будет принимать ввод. Однако, если устройство поддерживает сенсорный ввод, и пользователь начинает взаимодействовать с интерфейсом, прикасаясь к его элементам, исчезает необходимость выделять элементы или передавать фокус определенному отображаемому объекту. Следовательно, существует режим взаимодействия, который называется «режимом касания».

Как только пользователь касается экрана, устройство, поддерживающее сенсорный ввод, переходит в режим касания. Начиная с этого момента, фокус передается только тем отображаемым объектам, для которых [isFocusableInTouchMode()](https://developer.android.com/reference/android/view/View.html" \l "isFocusableInTouchMode()) имеет значение true, таким как виджеты редактирования текста. Другие отображаемые объекты, которых можно коснуться, например, кнопки, не будут получать фокус при касании. Нажатие будет просто запускать их приемники событий on-click.

В любой момент, когда пользователь нажимает клавишу перемещения или выполняет прокрутку трекболом, устройство выходит из режима касания и находит отображаемый объект, которому передается фокус. Теперь пользователь может возобновить взаимодействие с пользовательским интерфейсом без касания экрана.

Состояние режима касания поддерживается во всей системе (для всех окон и операций). Чтобы узнать текущее состояние, можно вызвать[isInTouchMode()](https://developer.android.com/reference/android/view/View.html#isInTouchMode()) и посмотреть, находится ли устройство в режиме касания.

Фокус обработки

В ответ на пользовательский ввод система обрабатывает обычное перемещение фокуса. Сюда относится изменение фокуса, когда отображаемые объекты удаляются или скрываются, а также когда становятся доступными новые отображаемые объекты. Отображаемые объекты сообщают о своей готовности получить фокус с помощью метода [isFocusable()](https://developer.android.com/reference/android/view/View.html" \l "isFocusable()). Чтобы изменить способность объекта View получать фокус, вызовите [setFocusable()](https://developer.android.com/reference/android/view/View.html" \l "setFocusable(boolean)). В режиме касания можно узнать способность отображаемого объекта View получать фокус с помощью метода [isFocusableInTouchMode()](https://developer.android.com/reference/android/view/View.html" \l "isFocusableInTouchMode()). Для изменения вызовите [setFocusableInTouchMode()](https://developer.android.com/reference/android/view/View.html" \l "setFocusableInTouchMode(boolean)).

Перемещение фокуса основано на алгоритме, который находит ближайший элемент в заданном направлении. В редких случаях алгоритм по умолчанию может не совпадать с поведением, которое предполагает разработчик. В этих ситуациях можно задать явное переопределение с помощью следующих атрибутов XML в файле макета: *nextFocusDown*, *nextFocusLeft*, *nextFocusRight*и *nextFocusUp*. Добавьте один из этих атрибутов в отображаемый объект View, *из* которого выходит фокус. Задайте значение атрибута для идентификатора объекта View, *в* который следует передать фокус. Например:

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

Обычно в этом вертикальном макете перемещение вверх из первой кнопки не должно приводить к перемещению, как и перемещение вниз из второй кнопки. Теперь, когда верхняя кнопка задает для нижней кнопки атрибут *nextFocusUp* (и наоборот), фокус будет перемещаться циклически сверху вниз и снизу вверх.

Если вы хотите объявить, что отображаемый объект в вашем пользовательском интерфейсе способен получать фокус (тогда как обычно он не может получать фокус), добавьте XML-атрибут android:focusable в объект View в объявлении макета. Установите для него значение*true*. Можно также объявить объект View, способным получать фокус в режиме касания с помощью android:focusableInTouchMode.

Чтобы запросить передачу фокуса определенному отображаемому объекту, вызовите [requestFocus()](https://developer.android.com/reference/android/view/View.html" \l "requestFocus()).

Чтобы перехватывать события фокуса (получать уведомления, когда отображаемый объект получает или теряет фокус), используйте метод [onFocusChange()](https://developer.android.com/reference/android/view/View.OnFocusChangeListener.html" \l "onFocusChange(android.view.View, boolean)), который обсуждается в разделе [Приемники событий](https://developer.android.com/guide/topics/ui/ui-events.html#EventListeners) выше.