Floating toolbar for text selection в Android Marshmallow: разбор нововведения

В Andriod при выделении текста появляется меню с действиями, которые можно выполнить: «Вырезать», «Скопировать», «Отправить».

В Android Marshmallow (SDK 23) появилась возможность расширять это меню и давать пользователю легкий доступ к дополнительным возможностям при работе с текстом: «Перевести», «Прокомментировать», «Процитировать».  
  
В процессе подготовке к выступлению на конференции [GDG в Нижнем Новгороде](http://habrahabr.ru/company/google/blog/270287/) я обнаружил, что эта новая возможность крайне плохо документирована, единственная доступная статья не во всем соответствует действительности, и в сети находится исчезающе мало примеров использования этой возможности. Пришлось разбираться самому. Результатами проведенного исследования и хочу поделиться. Это может сэкономить вам достаточно много времени.  
  
Поскольку различные меню в Android появлялись эволюционно, проще всего рассказ начать «от печки». Разработчики с опытом могут смело листать сразу в раздел [«Новое»](https://habrahabr.ru/post/271737/#new) без риска что-нибудь пропустить. Если что – потом вернетесь. Разработчики помоложе могут под спойлерами найти полезные на практике примеры.

Старое

Первое. Меню вульгарис

Рассказ про самое старое обычное меню. С примерами и скриншотами

Меню в Android устроены достаточно однотипно: они описываются с помощью [XML](http://developer.android.com/guide/topics/resources/menu-resource.html), затем в нужный момент в соответствии с этим описанием с помощью [MenuInflater](http://developer.android.com/reference/android/view/MenuInflater.html) создаются соответствующие объекты в памяти, это меню отображается, и информация о нажатии на соответствующий элемент приходит в колбэк.  
  
Обычное меню, которое во времена оно вызывалось нажатием на аппаратную клавишу «Menu», описывается в ресурсах приложения в виде XML-файла.

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item

android:id="@+id/action\_foo"

android:orderInCategory="100"

android:title="@string/word\_foo"/>

<item

android:id="@+id/action\_foobar"

android:orderInCategory="101"

android:title="@string/word\_bar"/>

<item

android:id="@+id/action\_baz"

android:orderInCategory="102"

android:title="@string/word\_baz"/>

</menu>

Пункты меню будут отсортированы по полю android:orderInCategory.  
Создается меню вызовом инфлейтера в методе [onCreateOptionsMenu()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreateOptionsMenu(android.view.Menu)).  
  
**@Override**

**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.my\_menu, menu);

**return** **true**;

}

Информация о выборе пункта меню попадает в метод [onOptionsItemSelected()](http://developer.android.com/reference/android/app/Activity.html" \l "onOptionsItemSelected(android.view.MenuItem)).  
  
**@Override**

**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

*// Handle item selection*

**switch** (item.getItemId()) {

**case** R.id.action\_foo:

toast("foo");

**return** **true**;

**case** R.id.action\_foobar:

toast("bar");

**return** **true**;

**case** R.id.action\_baz:

toast("baz");

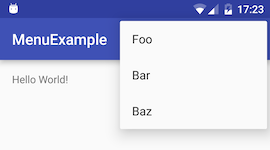
**return** **true**;

**default**:

**return** **super**.onOptionsItemSelected(item);

}

}

Все описанное выше приводит к появлению такого меню:  
  
  
  
Все просто и привычно.

Второе. Контекстное меню

Рассказ про контекстное меню. С примерами и скриншотами

Контекстное меню – это плавающее меню, которое появляется при длительном нажатии на какой-либо элемент интерфейса. Этот элемент должен быть предварительно зарегистрирован для работы с контекстным меню.   
  
Контекстные меню чаще всего применяются при работе со списками. Но этот рассказ был бы больше про списки, чем про меню, поэтому оставим за рамками статьи. Примеры будут без них.   
  
Рассмотрим простой пример.   
Зарегистрировать элемент можно, например, в [onCreate()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)):   
  
**@Override**

**protected** **void** **onCreate**(Bundle savedInstanceState) {

...

registerForContextMenu(findViewById(R.id.text\_view\_one));

}

Дальше все аналогично предыдущему пункту, только внутри других методов ([onCreateContextMenu()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreateContextMenu(android.view.ContextMenu, android.view.View, android.view.ContextMenu.ContextMenuInfo)) и [onContextItemSelected()](http://developer.android.com/reference/android/app/Activity.html" \l "onContextItemSelected(android.view.MenuItem))):  
  
**@Override**

**public** **void** **onCreateContextMenu**(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {

**super**.onCreateContextMenu(menu, v, menuInfo);

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.my\_menu, menu);

}

**@Override**

**public** **boolean** **onContextItemSelected**(MenuItem item) {

**switch** (item.getItemId()) {

**case** R.id.action\_foo:

toast("foo");

**return** **true**;

**case** R.id.action\_foobar:

toast("bar");

**return** **true**;

**case** R.id.action\_baz:

toast("baz");

**return** **true**;

**default**:

**return** **super**.onContextItemSelected(item);

}

}

При долгом нажатии на элемент с текстом появляется контекстное меню.

Скриншот



Третье. Меню в AppBar и Toolbar

Рассказ про меню с Toolbar. С примерами и скриншотами

Начиная с Android Honeycomb (он же 3.0, он же SDK 11, то есть тоже достаточно давно) аппаратная кнопка Menu была упразднена, а меню стало отображаться в «строке действий», она же «application bar». Появилась возможность часть пунктов меню показать в этой строке в виде иконок, а часть оставить скрытыми до нажатия на находящийся справа значок меню.   
  
Поля для описания иконок и расположения дополнили XML с описанием меню. В остальном это осталось все то же меню с известными нам [onCreateOptionsMenu()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreateOptionsMenu(android.view.Menu)) и [onOptionsItemSelected()](http://developer.android.com/reference/android/app/Activity.html" \l "onOptionsItemSelected(android.view.MenuItem)).

Меню с иконками и расположением. Добавились android:icon и android:orderInCategory

|  |  |
| --- | --- |
| https://habrastorage.org/files/7b2/51a/57e/7b251a57ea4c4cc0a9470149bc3f3fa1.png Видимая часть меню | https://habrastorage.org/files/fa2/756/066/fa2756066e284dd99dd1760099d5ff0c.png Скрытая часть меню,  если нажать на «три точки» |

На этом уровне про меню обычно знают все после первой же новичковой книжки. Спасибо, что хватило терпения дочитать досюда, дальше будет интересней.

Четвертое. Режим контекстных действий (Contextual action mode)

Рассказ про contextual action mode. С примерами и скриншотами

Режим контекстных действий, он же «Contextual action mode» позволяет показать пользователю набор действий, которые можно выполнить над выбранным элементом. Как и контекстное меню, этот инструмент удобен при работе со списками, но списки в этой статье мы не рассматриваем.   
  
Для того, чтобы рассмотреть работу этого режима, нам потребуется элемент, который можно «выбрать». Возьмем для примера[ToggleButton](http://developer.android.com/reference/android/widget/ToggleButton.html). Данный режим приложение активирует само, соответственно, сделаем это при изменении статуса ToggleButton:

checkedListener =

**new** CompoundButton.OnCheckedChangeListener() {

**@Override**

**public** **void** **onCheckedChanged**(CompoundButton buttonView,

**boolean** isChecked)

{

**if** (isChecked) {

actionMode = startActionMode(actionModeCallback);

actionMode.setTitle("Action Mode");

} **else** {

**if** (actionMode != **null**) {

actionMode.finish();

}

}

}

};

toggleButton.setOnCheckedChangeListener(checkedListener);

actionModeCallback – это экземпляр класса [ActionMode.Callback](http://developer.android.com/reference/android/view/ActionMode.Callback.html), который содержит колбэки для работы с меню. Меню при этом осталось все тем же, все с той же старой доброй механикой:

actionModeCallback = **new** ActionMode.Callback() {

**@Override**

**public** **boolean** **onCreateActionMode**(ActionMode mode, Menu menu) {

MenuInflater inflater = mode.getMenuInflater();

inflater.inflate(R.menu.my\_menu\_two, menu);

**return** **true**;

}

...

**@Override**

**public** **void** **onDestroyActionMode**(ActionMode mode) {

actionMode = **null**;

toggleButton.setChecked(**false**);

}

};

Обратите внимание на значение, которое мы сохраняем в переменной actionMode. Это экземпляр класса [ActionMode](http://developer.android.com/reference/android/view/ActionMode.html), нам он нужен для того, чтобы была возможность изменить заголовок ([actionMode.setTitle()](http://developer.android.com/reference/android/view/ActionMode.html" \l "setTitle(int))), подзаголовок ([actionMode.setSubtitle()](http://developer.android.com/reference/android/view/ActionMode.html" \l "setSubtitle(int))), а также завершить этот режим ([actionMode.finish()](http://developer.android.com/reference/android/view/ActionMode.html" \l "finish())). После выполнения действия режим автоматически не завершается, и если нам нужно, то завершить его мы должны сами.   
  
Обрабатывается выбор пользователя тоже привычным способом:

actionModeCallback = **new** ActionMode.Callback() {

...

**@Override**

**public** **boolean** **onActionItemClicked**(ActionMode mode, MenuItem item) {

**switch** (item.getItemId()) {

**case** R.id.action\_compass:

toast("compass");

**return** **true**;

**case** R.id.action\_camera:

toast("camera");

**return** **true**;

**default**:

**return** **false**;

}

}

};

Меню для контекстных действий, очевидно, должно отличаться от основного, поэтому создадим еще один XML c описанием.

XML с описанием меню. Такой же по структуре, но с другими пунктами

**<?xml version="1.0" encoding="utf-8"?>**

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:id="@+id/action\_camera"

android:icon="@android:drawable/ic\_menu\_camera"

android:orderInCategory="100"

android:title="@string/word\_camera"

app:showAsAction="ifRoom"

/>

<item

android:id="@+id/action\_compass"

android:icon="@android:drawable/ic\_menu\_compass"

android:orderInCategory="101"

android:title="@string/word\_compass"

app:showAsAction="ifRoom"

/>

</menu>

Получаем вот такое поведение:

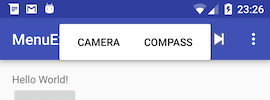
|  |  |
| --- | --- |
| https://habrastorage.org/files/1ae/a4f/5d7/1aea4f5d7c53497793e268156c26a13e.png | https://habrastorage.org/files/152/e34/521/152e3452195c4cf8baccc662ecfb71e0.png |

Все старенькое, то есть известное по крайней мере с SDK 11 теперь закончилось, началось неизвестное.

**Новое**

В Android Marshmallow (он же 6.0, он же SDK 23) в работе меню появилось два новшества. Оба эти новшества еще не поддерживаются Support Library и работают только на устройствах с SDK 23. Поэтому прежде чем вызывать появившиеся методы, необходимо проверять номер SDK устройства, на котором запущено приложение.   
  
Для удобства рассказа мы избавимся от этих проверок, указав minSdkVersion 23.

Пятое. Еще одно контекстное меню

Метод для создания [«Режима контекстных действий»](https://habrahabr.ru/post/271737/#contextaction) (описан выше) был расширен. Вторым параметром [startActionMode()](http://developer.android.com/reference/android/app/Activity.html" \l "startActionMode(android.view.ActionMode.Callback, int)) можно передать константу, которая задает тип отображения.  
  
Значение [ActionMode.TYPE\_PRIMARY](http://developer.android.com/reference/android/view/ActionMode.html" \l "TYPE_PRIMARY) соответствует старому поведению. То есть [startActionMode(actionModeCallback)](http://developer.android.com/reference/android/app/Activity.html" \l "startActionMode(android.view.ActionMode.Callback)) и[startActionMode(actionModeCallback, ActionMode.TYPE\_PRIMARY)](http://developer.android.com/reference/android/app/Activity.html#startActionMode(android.view.ActionMode.Callback, int)) – одно и тоже.   
  
Если задать тип [ActionMode.TYPE\_FLOATING](http://developer.android.com/reference/android/view/ActionMode.html" \l "TYPE_FLOATING), то меню приобретает следующий вид:   
  
  
  
Иконок нет. Расположение – горизонтальное.   
  
Если пунктов много настолько, что не влезают по ширине, то появятся уже знакомые «три точки»:

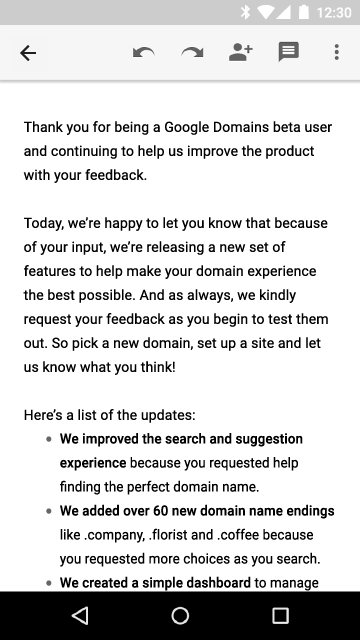
|  |  |
| --- | --- |
| https://habrastorage.org/files/279/48b/4ee/27948b4eefc84563adcf0a7181ea336d.png | https://habrastorage.org/files/a99/53b/fd8/a9953bfd84384747864b39930989b873.png |

Все остальное – точно также, как в [«Contextual action mode»](https://habrahabr.ru/post/271737/#contextaction) (см. выше).

Шестое. Расширение меню при выделении текста

Ну и, наконец, то, что было обещано в [статье о нововведениях в Android Marshmallow](http://developer.android.com/intl/ru/about/versions/marshmallow/android-6.0-changes.html#behavior-text-selection). Появилась возможность дополнить меню, появляющееся при выделении текста, своими пунктами. Анимированная картинка этого чуда была в самом начале статьи, но если хотите, она еще раз под спойлером.

Анимированная картинка с меню



Реализация

Чтобы добиться такого эффекта, сначала нужно создать унаследованный от [ActionMode.Callback](http://developer.android.com/reference/android/view/ActionMode.Callback.html) колбэк. То есть полностью аналогичный тому, что мы создавали двумя разделами выше для [«Contextual action mode»](https://habrahabr.ru/post/271737/#contextaction).   
  
Далее для всех элементов, в которых мы хотим расширить меню при редактировании, нужно указать этот колбэк:

textView.setTextIsSelectable(**true**);

textView.setCustomSelectionActionModeCallback(actionModeCallback);

editText.setCustomSelectionActionModeCallback(actionModeCallback);

editText.setCustomInsertionActionModeCallback(actionModeCallback);

Колбэк, указанный в [setCustomSelectionActionModeCallback()](http://developer.android.com/reference/android/widget/TextView.html" \l "setCustomSelectionActionModeCallback(android.view.ActionMode.Callback)), будет использоваться, если есть выделенный текст. Указаный в[setCustomInsertionActionModeCallback()](http://developer.android.com/reference/android/widget/TextView.html#setCustomInsertionActionModeCallback(android.view.ActionMode.Callback)) – если текста нет. Разделили их потому, что не все действия имеют смысл, когда ничего не выделено, и, соответственно, содержимое появляющихся меню должно быть разным.

|  |  |
| --- | --- |
| https://habrastorage.org/files/4e6/16c/36b/4e616c36b1954b09bf89d5f05e804943.png | https://habrastorage.org/files/30e/ed7/51b/30eed751b8ba41df99affdc52ab45291.png |

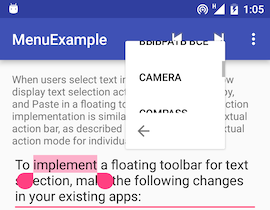
На пустом EditText выглядит вот так:

|  |  |
| --- | --- |
| https://habrastorage.org/files/57d/236/751/57d2367516824e2b8af5d3bb01a29f10.png | https://habrastorage.org/files/419/227/8b4/4192278b4f7f4298b391a210ca0d055c.png |

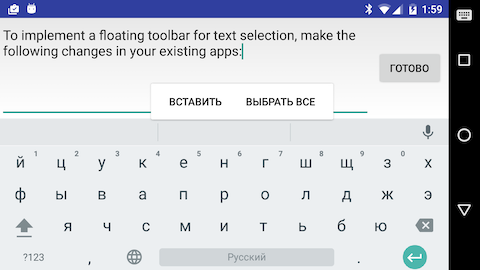
Ой. Куда делось мое меню?

Посмотрите на скриншоты ниже. Есть идея, куда делись все добавленные пункты меню?

|  |  |
| --- | --- |
| https://habrastorage.org/files/44d/5b8/589/44d5b8589ac549b2b41e2211d1797552.png | https://habrastorage.org/files/c6f/d69/c99/c6fd69c996374252be53855f7026037b.png |

Подсказку можно найти здесь:  
  
  
  
Поясню. Достаточно велика вероятность то, что во вторую, вертикальную часть меню оставшиеся пункты тоже не поместятся. И там появится скролл. На скриншотах вверху так и случилось.  
  
Проблема в том, никаих признаков скролла пользователю не видно: значок скролла почти сразу исчезает, а край непоместившегося пункта не выглядывает. Завел багу, посмотрим, что создатели этой фичи скажут. [code.google.com/p/android/issues/detail?id=195043](https://code.google.com/p/android/issues/detail?id=195043)

Куда опять делось мое меню?

  
  
На этот раз другая проблема: меню при fullscreen input mode описанным выше способом не расширяется. Воркэраунд я нашел только один: выключить fullscreen режим с помощью android:imeOptions=«flagNoExtractUi».

Как разместить свой пункт перед стандартными?

Родные пункты меню имеют параметры order от 1 до 5. Поэтому с помощью android:orderInCategory в описании меню задать положение перед родными пунктами нельзя. Но можно изменить порядок пунктов в уже сформированном меню, например, так:

**@Override**

**public** **boolean** **onPrepareActionMode**(ActionMode mode, Menu menu) {

*// родные пункты меню "нумеруются" с 1,*

*// дополнительные я "пронумеровал" со 100*

**while** (menu.getItem(0).getOrder() < 100) {

MenuItem item = menu.getItem(0);

menu.removeItem(item.getItemId());

*// теперь родные будут "нумероваться" с 200, то есть*

*// станут после дополнительных*

menu.add(item.getGroupId(), item.getItemId(),

item.getOrder() + 200, item.getTitle());

}

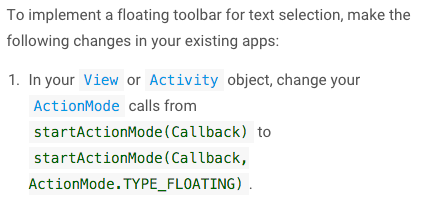
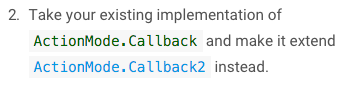
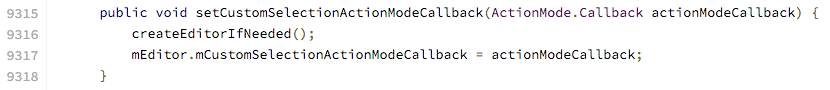
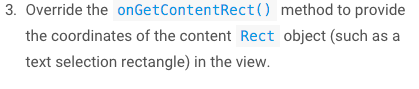
**return** **true**;

}

Получаем результат:

|  |  |
| --- | --- |
| https://habrastorage.org/files/c77/54b/352/c7754b35275c4192a0e5f7de2a204dc0.png | https://habrastorage.org/files/c5d/f68/4fa/c5df684fa47248cfae4248273b825541.png |

Странности в документации

Нормальной документации на расширение меню при выделении текста нет. Есть уже упоминавшаяся статья [статья о новинках в Android Marshmallow](http://developer.android.com/intl/ru/about/versions/marshmallow/android-6.0-changes.html" \l "behavior-text-selection).  
  
Я несколько десятков раз перечитывал это место, но так и не смог соотнести то, что там написано, с практикой. Рассказываю по порядку.   
  
  
  
Если вы прочитали эту статью или попробовали сами, то заметили, что [startActionMode(callback, ActionMode.TYPE\_FLOATING)](http://developer.android.com/reference/android/app/Activity.html" \l "startActionMode(android.view.ActionMode.Callback, int))действительно создает floating toolbar, но совсем не для selection. А для selection его создает [setCustomSelectionActionModeCallback()](http://developer.android.com/reference/android/widget/TextView.html" \l "setCustomSelectionActionModeCallback(android.view.ActionMode.Callback)).  
  
  
  
Вторая непонятная инструкция. Метод [setCustomSelectionActionModeCallback()](http://developer.android.com/reference/android/app/Activity.html#startActionMode(android.view.ActionMode.Callback, int)) класса [android.widget.TextView](http://developer.android.com/reference/android/widget/TextView.html) ожидает получить[ActionMode.Callback](http://developer.android.com/reference/android/view/ActionMode.Callback.html).  
  
  
  
[android.googlesource.com/platform/frameworks/base/+/master/core/java/android/widget/TextView.java](https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/widget/TextView.java)  
  
Если посмотреть в [android.widget.Editor](https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/widget/Editor.java), то упомянутое поле mCustomSelectionActionModeCallback также имеет тип[ActionMode.Callback](http://developer.android.com/reference/android/view/ActionMode.Callback.html):  
  
https://habrastorage.org/files/63a/2ca/45d/63a2ca45d68f41f2a5c9869f78b8622e.png  
  
И нигде в этом коде не ожидается, что custom callback будет типа [ActionMode.Callback2](http://developer.android.com/reference/android/view/ActionMode.Callback2.html).   
[android.googlesource.com/platform/frameworks/base/+/master/core/java/android/widget/Editor.java](https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/widget/Editor.java)  
  
Статья же продолжает упорно рассказывать о том, как использовать [ActionMode.Callback2](http://developer.android.com/reference/android/view/ActionMode.Callback2.html):  
  
  
  
Мое предположение, что в статью по недосмотру редактора попал фрагмент какой-то внутренней документации по[ActionMode.Callback2](http://developer.android.com/reference/android/view/ActionMode.Callback2.html). Есть у вас есть другие гипотезы, напишите об этом.

Пример

См. прилагаемый архив…