

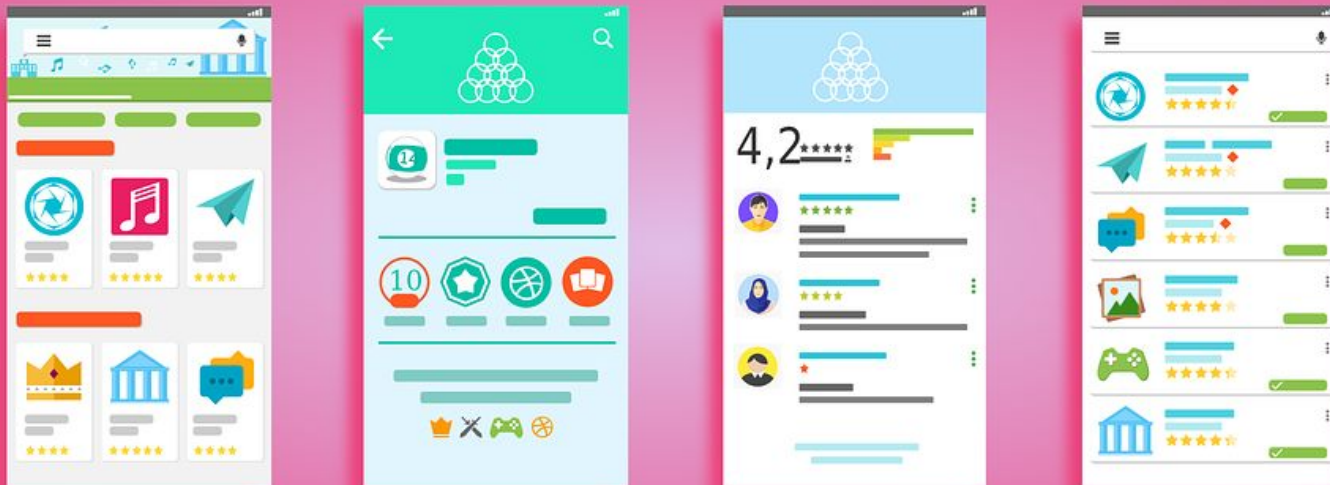


УНИВЕРСИТЕТ ИТМО

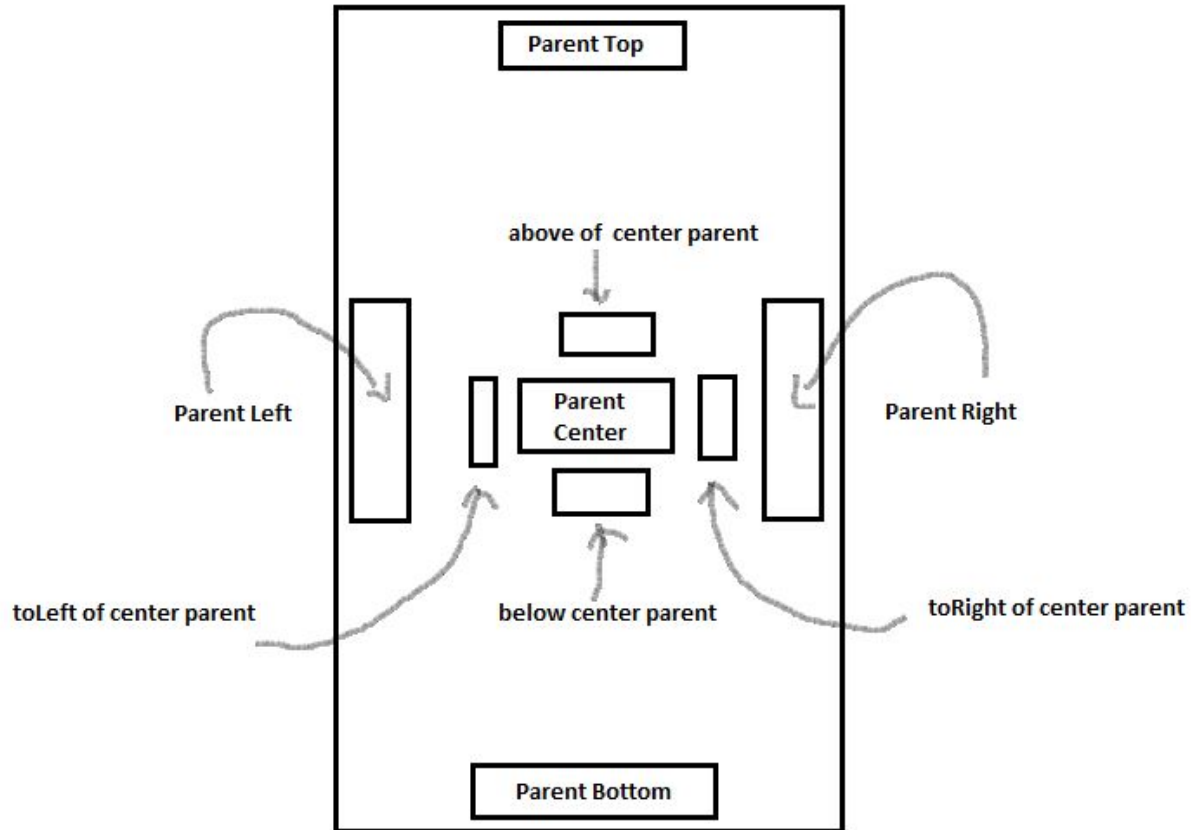
Пользовательский интерфейс

Санкт-Петербург, 2016

Пользовательский интерфейс состоит из макетов и компонентов графического интерфейса



Относительный макет (RelativeLayout)



Относительный макет (RelativeLayout)

RelativeLayout TextView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="ru.limty.timer.MainActivity">
```

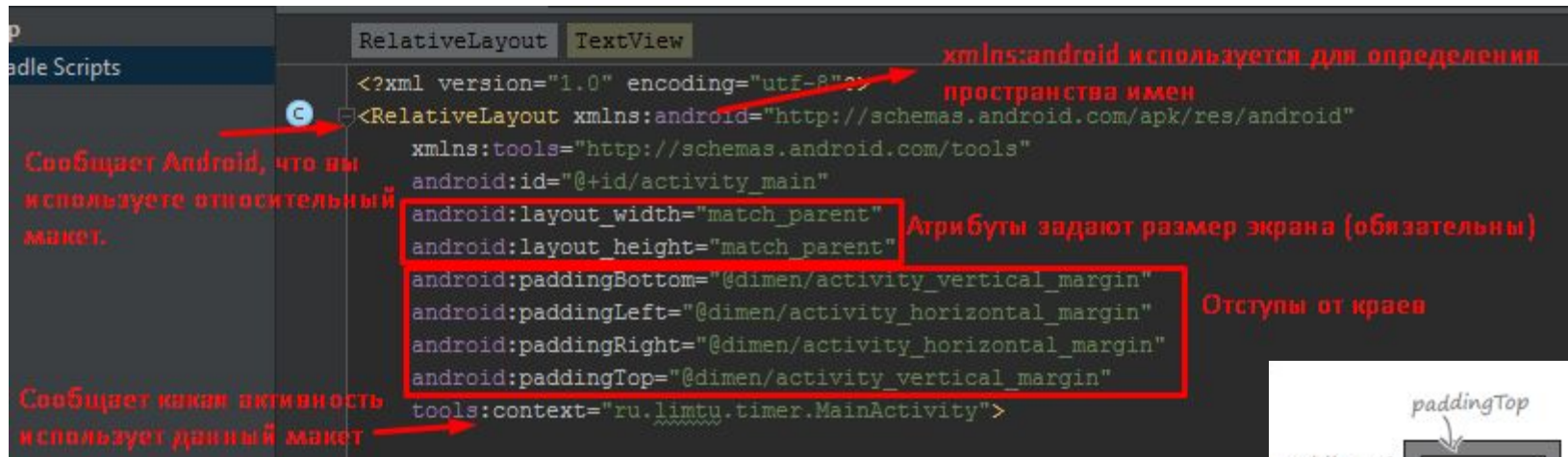
Сообщает Android, что вы используете относительный макет.

Сообщает какая активность использует данный макет

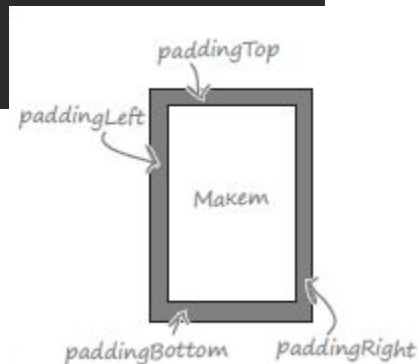
xmlns:android используется для определения пространства имен

Атрибуты задают размер экрана (обязательны)

Отступы от краев



В относительном макете представления позиционируются относительно родительского макета или относительно других представлений в макете.



Атрибуты для позиционирования представлений относительно родительского макета

android:layout_alignParentLeft

Левый край представления выравнивается по левому краю родителя.



android:layout_alignParentTop

Верхний край представления выравнивается по верхнему краю родителя.



android:layout_alignParentRight

Правый край представления выравнивается по правому краю родителя.



android:layout_alignParentLeft



android:layout_alignParentTop



android:layout_alignParentRight

android:layout_alignParentBottom

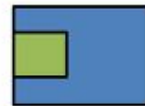
Нижний край представления выравнивается по нижнему краю родителя.



android:layout_alignParentBottom



android:layout_centerHorizontal



android:layout_centerVertical

android:layout_centerHorizontal

Выравнивается по центру внутри родителя (по горизонтали).

android:layout_centerVertical

Выравнивается по центру внутри родителя (по вертикали).

android:layout_centerInParent

Выравнивается по центру внутри родителя (по горизонтали и вертикали).



android:layout_centerInParent

Позиционирование представлений относительно других представлений

android:layout_above

Представление размещается над якорным представлением.

android:layout_below

Представление размещается под якорным представлением.

android:layout_alignTop

Верхний край представления выравнивается по верхнему краю якорного представления.

android:layout_alignBottom

Нижний край представления выравнивается по нижнему краю якорного представления.

android:layout_alignLeft

Левый край представления выравнивается по левому краю якорного представления.

android:layout_alignRight

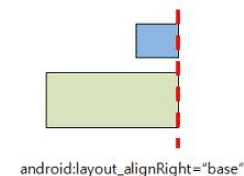
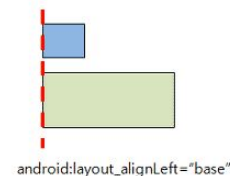
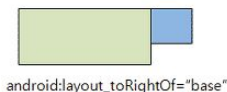
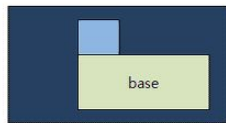
Правый край представления выравнивается по правому краю якорного представления.

android:layout_toLeftOf

Правый край представления располагается у левого края якорного представления.

android:layout_toRightOf

Левый край представления располагается у правого края якорного представления.



Создание интервалов между представлениями

Чтобы компоненты "не прилипали" друг к другу, используются атрибуты, добавляющие пространство между ними.

android:layout_marginTop

Добавляет дополнительный интервал у верхнего края представления.

android:layout_marginBottom

Добавляет дополнительный интервал у нижнего края представления.

android:layout_marginLeft

Добавляет дополнительный интервал у левого края представления.

android:layout_marginRight

Добавляет дополнительный интервал у правого края представления.



Линейный макет (LinearLayout)



В линейном макете представления размещаются рядом друг с другом по вертикали или горизонтали.

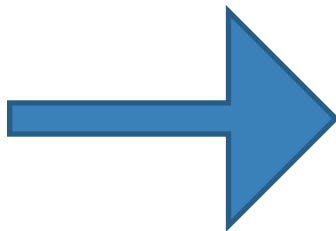
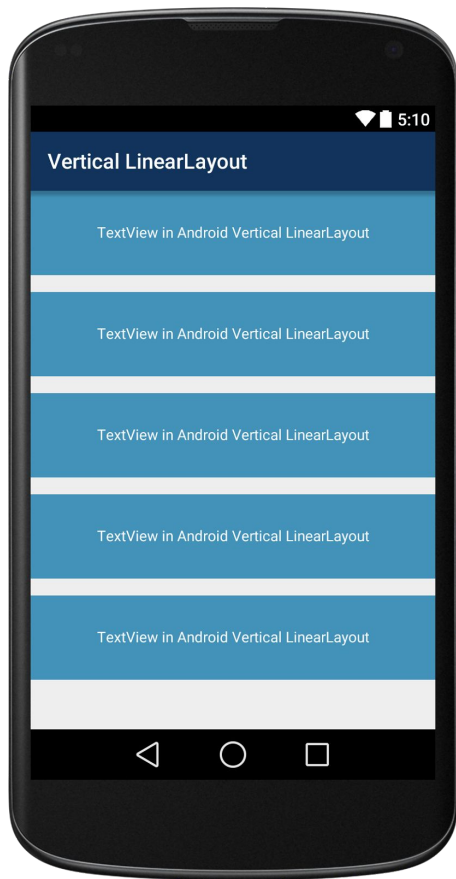
Линейный макет (LinearLayout)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context="ru.limtu.timer.MainActivity">
```

Атрибут **android:orientation** задает направление размещения представлений.
(vertical | horizontal)

Атрибуты **android:layout_width**, **android:layout_height** и **android:orientation** являются обязательными.

В линейном макете представления отображаются в порядке их следования в разметке XML



```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView1"
    android:paddingTop="20dp"/>

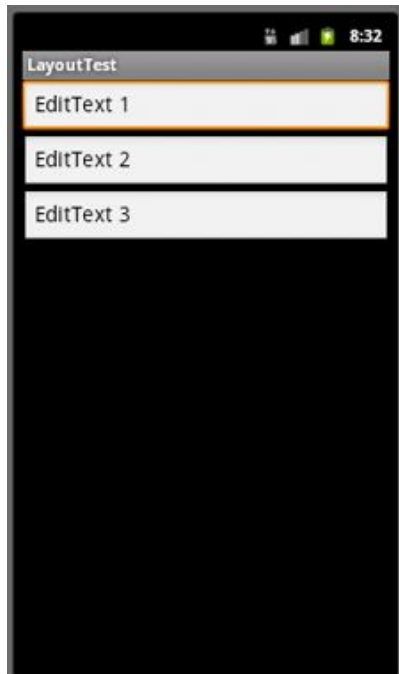
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView2"
    android:paddingTop="20dp"/>

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView3"
    android:paddingTop="10dp"/>

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView4"
    android:paddingTop="20dp"/>

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView5"
    android:paddingTop="20dp"/>
```

android:layout_weight="число"



Default weights

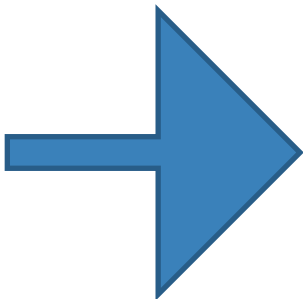
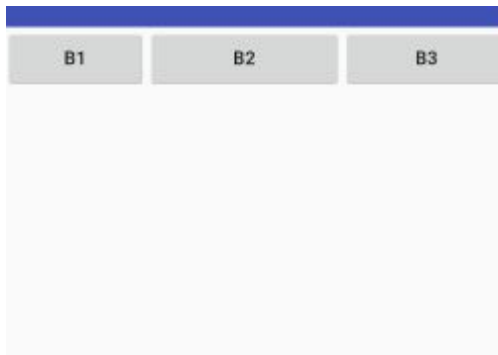


Weights – 1,1,0



Weights – 1,1,2

Пример использования `layout_weight`



```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="B1"
    android:layout_weight="1">
</Button>

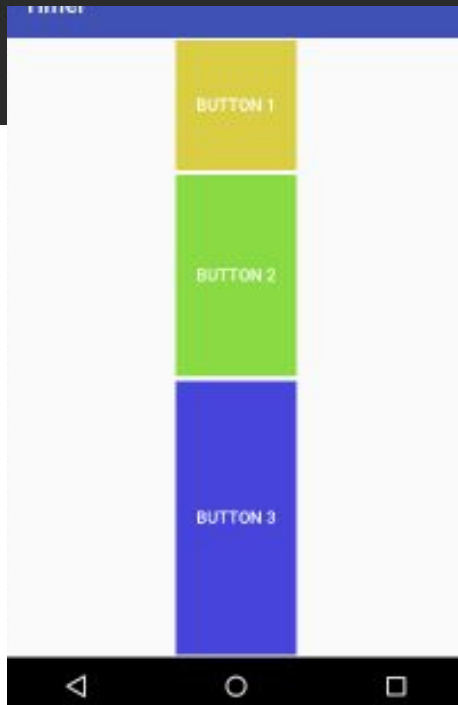
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="B2"
    android:layout_weight="3">
</Button>

<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="B3"
    android:layout_weight="2">
</Button>
```

Атрибут android:gravity: список значений

Атрибут **android:gravity** управляет размещением содержимого внутри представления.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
```



- **top**
- **bottom**
- **left**
- **right**
- **center_vertical**
- **center_horizontal**
- **center**
- **fill_vertical**
- **fill_horizontal**
- **fill**

более подробно см. заметки к слайду.

android:layout_gravity:

top, bottom, left, right

Размещает представление у верхнего, нижнего, левого или правого края контейнера.

start, end

Размещает представление в начале или в конце контейнера.

center_vertical, center_horizontal

Выравнивает представление по вертикали или по горизонтали внутри контейнера.

center

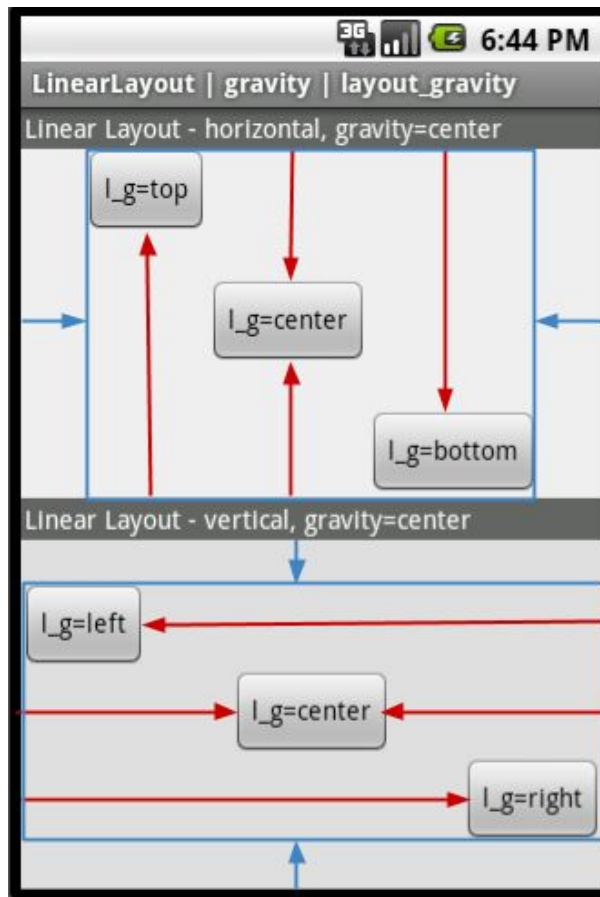
Выравнивает представление по вертикали и по горизонтали внутри контейнера.

fill_vertical, fill_horizontal

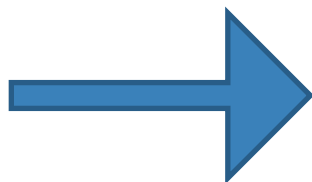
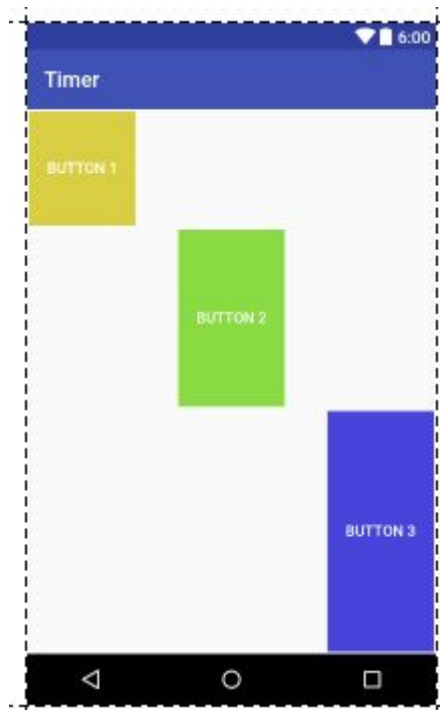
Масштабирует представление так, чтобы оно заполняло контейнер в вертикальном или горизонтальном направлении.

fill

Масштабирует представление так, чтобы оно заполняло контейнер по вертикали и по горизонтали.



Атрибут android:layout_gravity: пример



```
<Button
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="Button 1"
    android:background="#dace44"
    android:layout_margin="2dp"
    android:textColor="#fff"
    android:layout_gravity="left"
    android:layout_weight="1"/>

<Button
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="Button 2"
    android:background="#9ada44"
    android:layout_margin="2dp"
    android:textColor="#fff"
    android:layout_gravity="center"
    android:layout_weight="2"/>

<Button
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="Button 3"
    android:background="#4644da"
    android:layout_margin="2dp"
    android:textColor="#fff"
    android:layout_gravity="right"
    android:layout_weight="3"/>
```

Табличный макет (GridLayout)

Столбцы

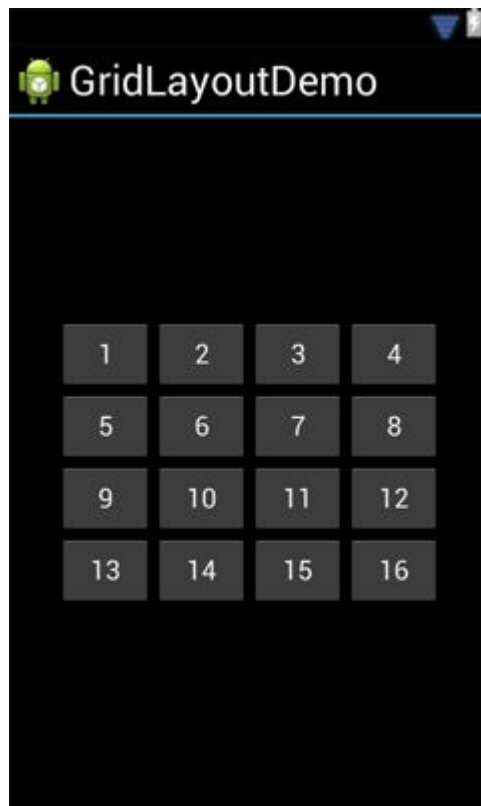
android:columnCount="число"

Строки

android:rowCount="число"

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:columnCount="3"> или rowCount.
```

GridLayout требует API уровня 14 и выше.

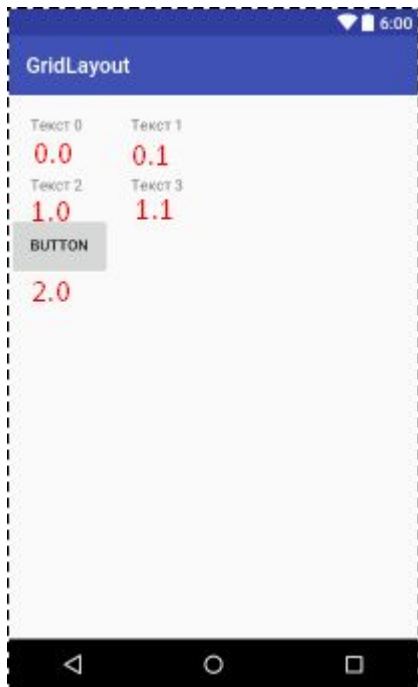


Табличный макет (GridLayout)



```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="2"
    android:columnCount="2">
    <Button
        android:text="Текст 0"
        android:textSize="14sp"
        android:padding="15dp" />
    <Button
        android:text="Текст 1"
        android:textSize="14sp"
        android:padding="15dp" />
    <Button
        android:text="Текст 2"
        android:textSize="14sp"
        android:padding="15dp" />
    <Button
        android:text="Текст 3"
        android:textSize="14sp"
        android:padding="15dp" />
</GridLayout>
```

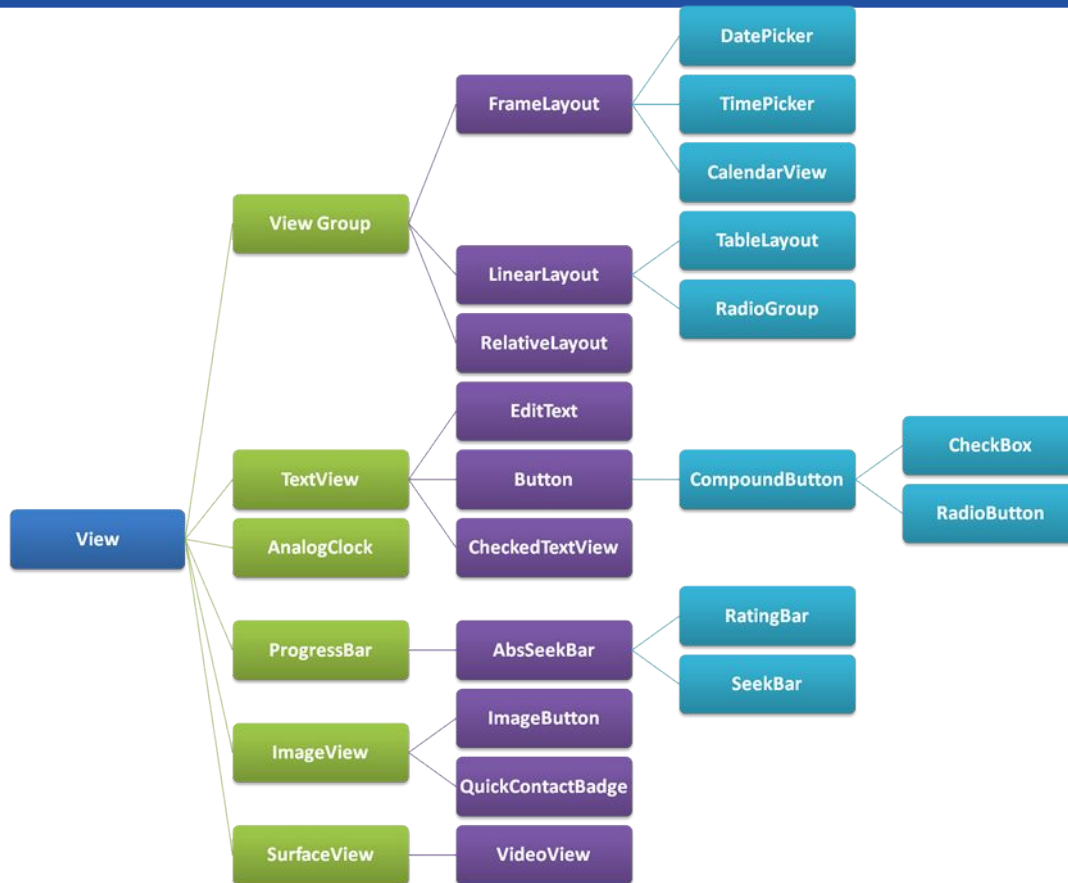
Табличный макет (GridLayout)



Если требуется задать точное расположение.

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="2">
    <TextView
        android:text="Текст 0"
        android:textSize="14sp" android:padding="20dp"
        android:layout_row="0" android:layout_column="0"/>
    <TextView
        android:text="Текст 1"
        android:textSize="14sp" android:padding="20dp"
        android:layout_row="0" android:layout_column="1"/>
    <TextView
        android:text="Текст 2"
        android:textSize="14sp" android:padding="20dp"
        android:layout_row="1" android:layout_column="0"/>
    <TextView
        android:text="Текст 3"
        android:textSize="14sp" android:padding="20dp"
        android:layout_row="1" android:layout_column="1"/>
    <Button
        android:text="@string/button"
        android:layout_row="2" android:layout_column="0"
        android:padding="20dp"/>
</GridLayout>
```

Специализации View и ViewGroup



Надпись (TextView)

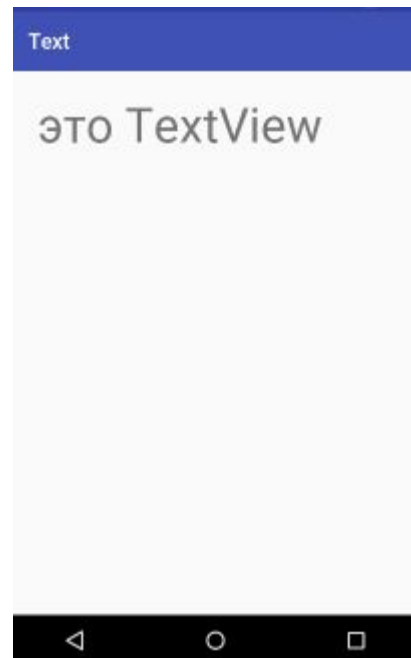
- **Используется для вывода текста.**
- **Для изменения размера используется атрибут android:textSize:**
`android:textSize="14sp"`

- **Определение в XML**

```
<TextView  
    android:id="@+id/textview"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/text" />
```

- **Использование надписи в коде активности**

```
TextView tV = (TextView) findViewById(R.id.textview);  
textView.setText("Some other string");
```



Текстовое поле (EditText)

- **Аналог надписи, но с возможностью редактирования.**
- **Определение в XML**

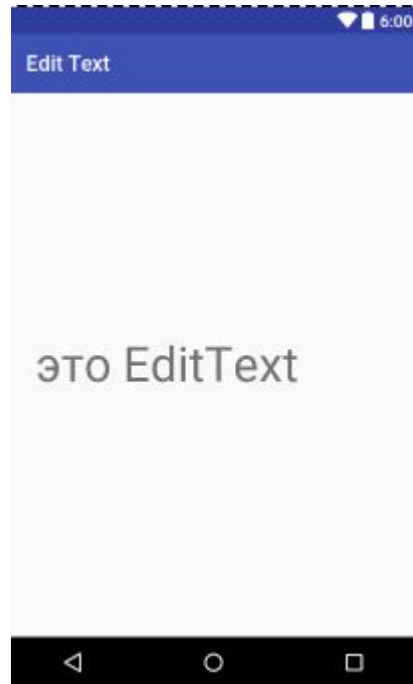
```
<EditText
    android:id="@+id/edit_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_text" />
```

- **Атрибут android:inputType="number"**

- phone** - предоставляет клавиатуру для ввода номеров.
- textPassword** - предоставляет клавиатуру для ввода пароля.
- textCapSentences** - первое слово начинается с прописной буквы.
- textAutoCorrect** - автоматически исправляет вводимый текст.

- **Использование в коде активности**

```
EditText editText = (EditText) findViewById(R.id.edit_text);
String text = editText.getText().toString();
```



Кнопка (Button)

- **Определение в XML**

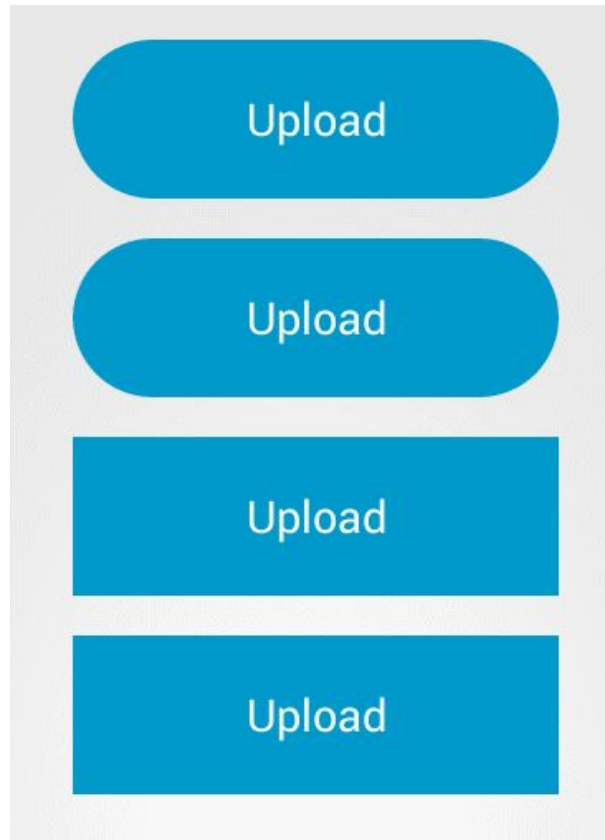
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text" />
```

- **Использование в коде активности**

```
android:onClick="onButtonClicked"
```

Затем в активности определяется метод следующего вида:

```
/** Вызывается при щелчке на кнопке */
public void onButtonClicked(View view) {
    // Сделать что-то по щелчку на кнопке
}
```



Двухпозиционная кнопка (ToggleButton)

- Щелчка на двухпозиционной кнопке, пользователь выбирает одно из двух состояний.

- Определение в XML**

```
<ToggleButton android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textOn="@string/on"  
    android:textOff="@string/off" />
```

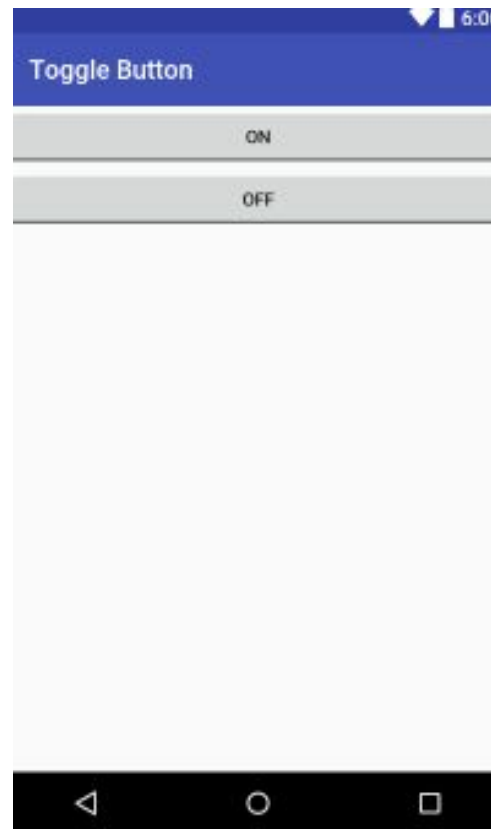
- Использование в коде активности**

```
android:onClick="onToggleButtonClicked"
```

Затем в активности определяется метод следующего вида:

```
/** Вызывается при щелчке на двухпозиционной кнопке*/
```

```
public void onToggleClicked(View view) {  
    // Получить состояние двухпозиционной кнопки.  
    boolean on = ((ToggleButton) view).isChecked();  
    if (on) {  
        // Вкл  
    } else {  
        // Выкл  
    }  
}
```



Выключатель (Switch)

- Выключатель представляет собой рычажок, который работает по тому же принципу, что и двухпозиционная кнопка.

- **Определение в XML**

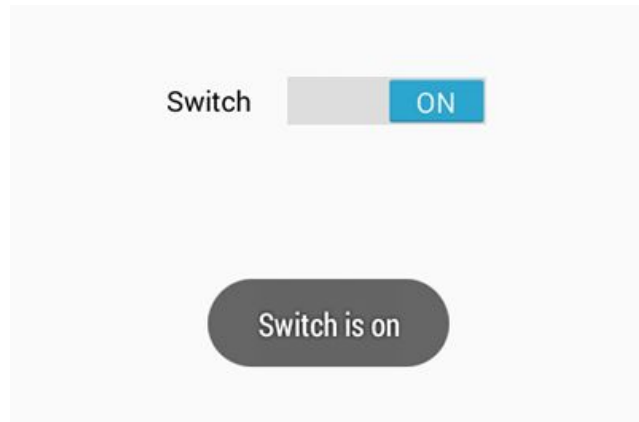
```
<Switch
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="@string/on"
    android:textOff="@string/off" />
```

- **Использование в коде активности**

```
android:onClick="onSwitchClicked"
```

Затем в активности определяется метод следующего вида:

```
/** Вызывается при щелчке на выключателе. */
public void onSwitchClicked(View view) {
    // Включенное состояние?
    boolean on = ((Switch) view).isChecked();
    if (on) {
        // Вкл
    } else {
        // Выкл
    }
}
```



Флажки (CheckBox)

- Флажки (check boxes) предоставляют пользователю набор независимых вариантов. Каждый флажок может устанавливаться или сниматься независимо от всех остальных флажков.

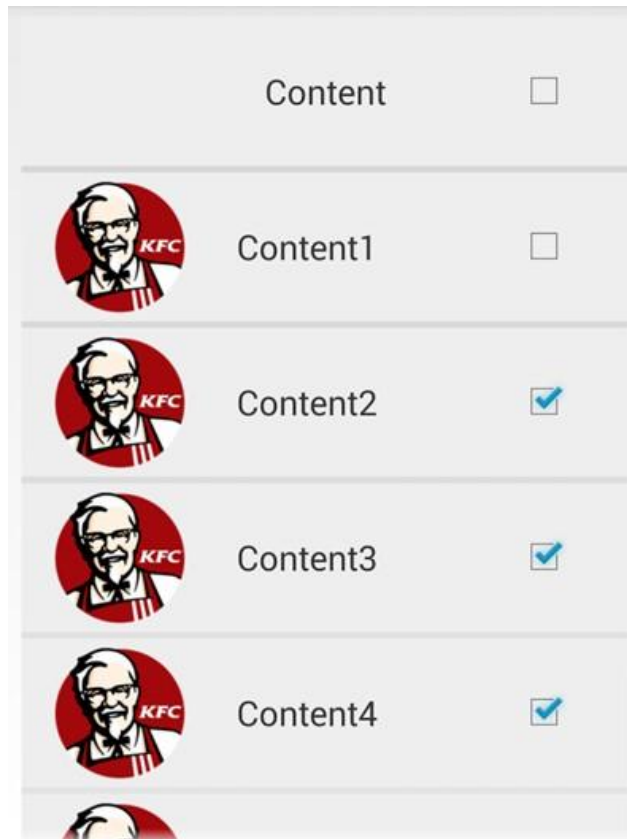
- **Определение в XML**

```
<CheckBox  
...  
android:text="@string/content1" />
```

```
<CheckBox  
...  
android:text="@string/content2" />
```

- **Использование в коде активности**

```
CheckBox cb = (CheckBox) findViewById(R.id.check);  
boolean checked = checkbox.isChecked();  
if (checked) {  
    //Действия для установленного флажка  
}
```



Флажки (продолжение...)

- Чтобы обрабатывать щелчки на флажках (по аналогии со щелчками на кнопках), включите атрибут `android:onClick` в XML макета и присвойте ему имя вызываемого метода из кода активности:

<pre><CheckBox ... android:text="@string/content1" android:onClick="onCheckboxClicked"/></pre>	<pre><CheckBox ... android:text="@string/content2" android:onClick="onCheckboxClicked"/></pre>
--	--

- Затем в активности определяется метод следующего вида:

```
public void onCheckboxClicked(View view) {
    // Был ли установлен флажок, на котором щелкнул пользователь?
    boolean checked = ((CheckBox) view).isChecked();
    // Определить, на каком флажке был сделан щелчок
    switch(view.getId()) {
        case R.id.checkbox_content1:
            if (checked)
            else
            break;
        case R.id.checkbox_content2:
            if (checked)
            else
            break;
    }
}
```

Переключатели (RadioButton)

- Переключатели предоставляют набор вариантов, из которого пользователь может выбрать ровно один вариант:

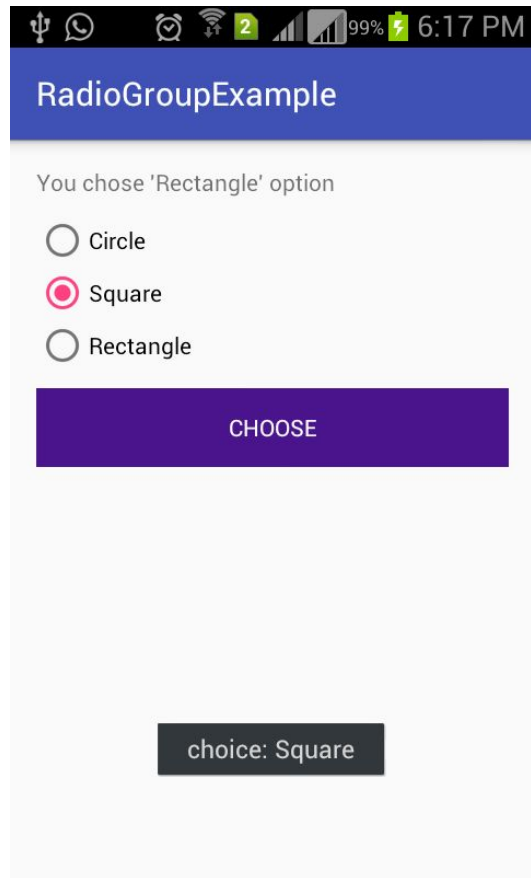
- **Определение в XML**

```
<RadioGroup
    android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:id="@+id/radio_circle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/circle" />

    <RadioButton
        android:id="@+id/radio_square"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/square" />

</RadioGroup>
```



Раскрывающийся список (Spinner)

- Раскрывающийся список содержит набор значений, из которых пользователь может выбрать только одно.

- **Определение в XML**

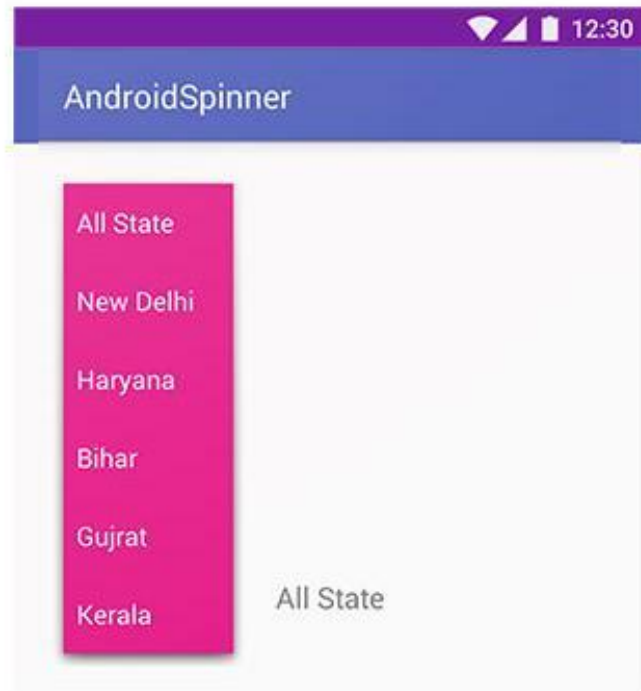
```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_values" />
```

Массив строк добавляется в файл *strings.xml* :

```
<string-array name="spinner_values">
    <item>New Delphi</item>
    <item>Haryana</item>
    <item>Bihar</item>
    <item>Guijrat</item>
</string-array>
```

- **Использование в коде активности**

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
String string = String.valueOf(spinner.getSelectedItem());
```



Графическое представление (ImageView)

- **Определение в XML макета**

```
<ImageView  
android:layout_width="200dp"  
android:layout_height="100dp"  
android:src="@drawable/logo"  
android:contentDescription="@string/logo" />
```

Ресурсы изображений снабжаются префиксом `@drawable`, который сообщает Android, что ресурс изображения хранится в одной или нескольких папках `drawable`.

- **Использование в коде активности**

```
ImageView photo = (ImageView) findViewById(R.id.photo);  
int image = R.drawable.logo;  
String description = "This is the logo";  
photo.setImageResource(image);  
photo.setContentDescription(description);
```

Этот фрагмент кода ищет ресурс изображения с именем `starbuzz_logo` в папках `drawable*` и назначает его источником данных для графического представления с идентификатором `photo`.



Вывод изображений на кнопках (Button)

- **Вывод текста и изображения на кнопке**

Чтобы вывести на кнопке текст, справа от которого находится графическое изображение, используйте атрибут `android:drawableRight` и укажите нужное изображение:

Вывести графический ресурс `android` в правой части кнопки.

```
<Button
```

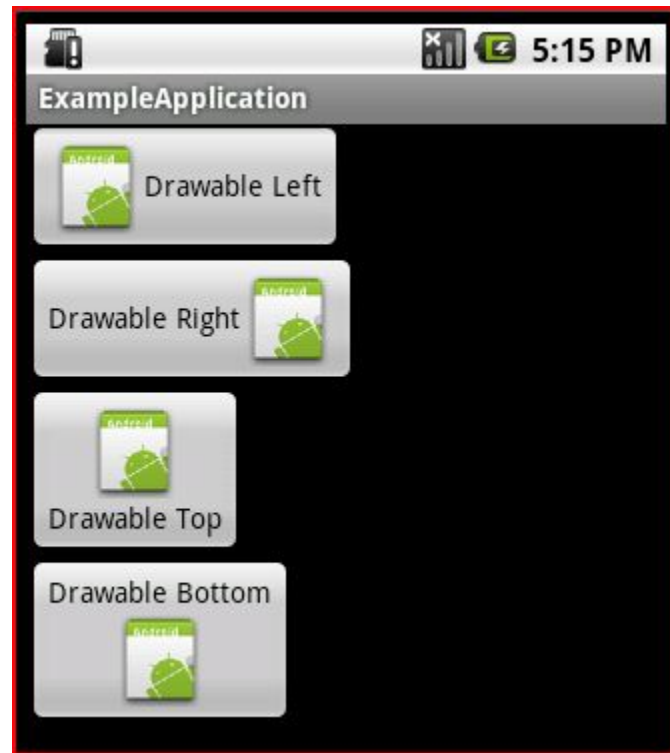
```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:drawableRight="@drawable/android"  
    android:text="@string/click_me" />
```

Чтобы изображение располагалось слева от текста, воспользуйтесь атрибутом `android:drawableLeft`:

```
<Button
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:drawableLeft="@drawable/android"  
    android:text="@string/click_me" />
```

и т.д.



Графическая кнопка (ImageButton)

- Графическая кнопка почти не отличается от обычной — просто на ней выводится только изображение, без текста.

- Определение в XML**

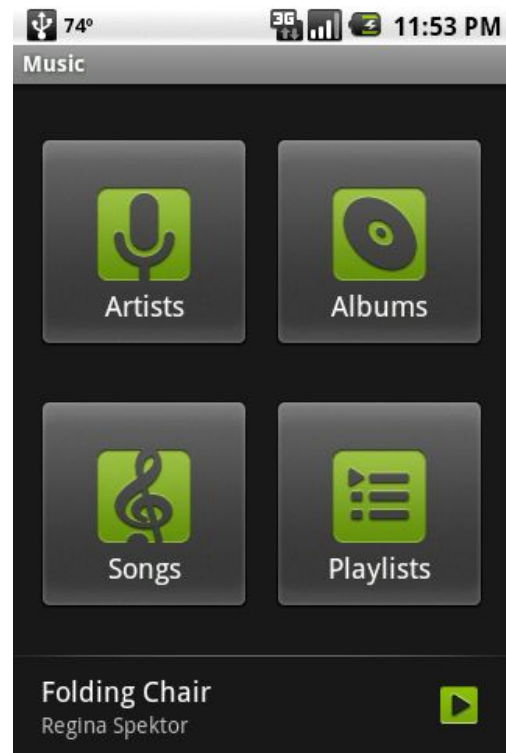
```
<ImageButton
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon" />
```

- Использование в коде активности**

```
android:onClick="onButtonClicked"
```

Затем в активности определяется метод следующего вида:

```
/** Вызывается при щелчке на кнопке */
public void onButtonClicked(View view) {
    // Сделать что-то по щелчку на кнопке
}
```



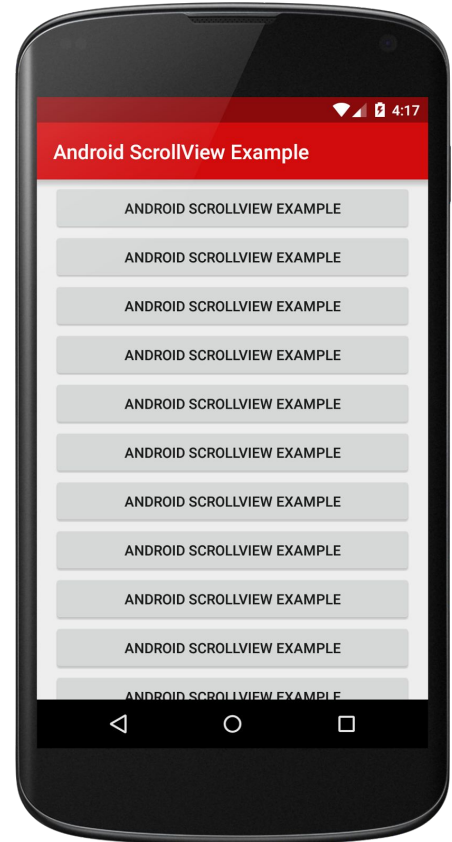
Прокручиваемые представления (ScrollView)

- Чтобы добавить вертикальную полосу прокрутки, заключите существующий макет в элемент **<ScrollView>**:

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity" >

    <LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="16dp"
android:paddingLeft="16dp"
android:paddingRight="16dp"
android:paddingTop="16dp"
android:orientation="vertical" >
        ...
    </LinearLayout>
</ScrollView>
```

- Чтобы добавить в макет горизонтальную полосу прокрутки, заключите существующий макет в элемент **<HorizontalScrollView>**.



Уведомления (Toast)

Уведомления выполняют чисто информационные функции, пользователь не может с ними взаимодействовать. Пока уведомление находится на экране, активность остается видимой и доступной для взаимодействия с пользователем. Уведомление автоматически закрывается по истечении тайм-аута.

- **Использование в коде активности**

Уведомления создаются только в коде активности; определить их в макете невозможно.

```
CharSequence text = "Hello, I'm a Toast!";  
int duration = Toast.LENGTH_SHORT;  
Toast toast = Toast.makeText(this, text, duration);  
toast.show();
```

