**Продвинутые примеры с Button**

Рассмотрим продвинутые примеры с [кнопкой](http://developer.alexanderklimov.ru/android/views/button.php).

[Системные стили](http://developer.alexanderklimov.ru/android/views/button-advanced.php#systemstyle)  
[Самоуничтожающаяся кнопка](http://developer.alexanderklimov.ru/android/views/button-advanced.php#destroy)  
[Стильные кнопки](http://developer.alexanderklimov.ru/android/views/button-advanced.php#style)  
[Программное управление внешним видом кнопки](http://developer.alexanderklimov.ru/android/views/button-advanced.php#coding)  
[Анимация для кнопок](http://developer.alexanderklimov.ru/android/views/button-advanced.php#animation)  
[Вверх тормашками](http://developer.alexanderklimov.ru/android/views/button-advanced.php#ups)  
[Манипуляции с кнопкой](http://developer.alexanderklimov.ru/android/views/button-advanced.php#change)  
[Быстрая инициализация кнопок через массив](http://developer.alexanderklimov.ru/android/views/button-advanced.php#array)  
[Круглая кнопка](http://developer.alexanderklimov.ru/android/views/button-advanced.php#circlebutton)  
[Программное добавление кнопок](http://developer.alexanderklimov.ru/android/views/button-advanced.php#addbutton)

Системные стили

Ниже будет рассказано, как применять собственные стили для кнопки. Также можно подключать системные стили. Добавим в файл **styles.xml** следующие строки.

<style name="MyButton" parent="Theme.AppCompat.Light">

<item name="colorControlHighlight">@android:color/holo\_green\_light</item>

<item name="colorButtonNormal">@android:color/holo\_orange\_dark</item>

</style>

Подключим через атрибут **android:theme**:

<Button

android:id="@+id/button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Button"

android:theme="@style/MyButton" />

Цвет **colorButtonNormal** относится к кнопке в обычном состоянии. При нажатии цвет поменяется на **colorControlHighlight**.

Самоуничтожающаяся кнопка

Хотите удалить кнопку при нажатии на неё? А вам не страшно? Ну ладно, вот код уничтожения:

public void onClick(View view) {

Button buttonDestroy = (Button)findViewById(R.id.button1);

// получим родителя кноппки

ViewGroup parent = (ViewGroup) buttonDestroy.getParent();

// Убираем кнопку у родителя

parent.removeView(buttonDestroy);

}

Если удаления несчастной кнопки вам недостаточно, то можете удалить вообще все элементы родительского объекта:

parent.removeAllViews(); // Галактика в опасности! Подумайте ещё раз!

Стильные кнопки

Стандартная кнопка выглядит скучно и малоинтересно. Но у неё есть атрибуты, позволяющие изменить свой внешний вид. Вы можете изменить форму и цвет кнопки при помощи свойств **background**, **drawableTop**, **drawableBottom**, **drawableLeft** и **drawableRight**, присвоив им подходящие графические ресурсы.

Кстати, совсем не очевидно для многих, что если одному из последних четырёх свойств присвоить изображение, то получим кнопку с картинкой.

<Button

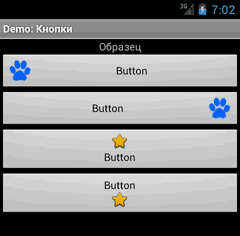
android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

**android:drawableLeft="@drawable/paw\_on"**

android:text="Button" />



Управлять свойствами **drawableXXX** можно программно. Причём, мы можем не только их установить в коде, но и задать нужную прозрачность. Для примера добавим на экран пять элементов **SeekBar**, которые будут отвечать за прозрачность всей кнопки и по отдельности за каждый элемент **Drawable**.

<LinearLayout 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"

android:orientation="vertical"

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=".MainActivity" >

<SeekBar

android:id="@+id/opacitybuttonSeekBar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="100"

android:progress="100"/>

<SeekBar

android:id="@+id/opacitydrawableleftSeekBar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="255"

android:progress="255"/>

<SeekBar

android:id="@+id/opacitydrawabletopSeekBar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="255"

android:progress="255"/>

<SeekBar

android:id="@+id/opacitydrawablerightSeekBar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="255"

android:progress="255"/>

<SeekBar

android:id="@+id/opacitydrawablebottomSeekBar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="255"

android:progress="255"/>

<Button

android:id="@+id/button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Opacity Button"/>

</LinearLayout>

Ресурсы для **Drawable** будем брать у системы для экономии кода.

package ru.alexanderklimov.test;

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

import android.app.Activity;

import android.graphics.drawable.Drawable;

import android.os.Bundle;

import android.widget.Button;

import android.widget.SeekBar;

import android.widget.SeekBar.OnSeekBarChangeListener;

public class MainActivity extends Activity {

private SeekBar mOpacityButtonSeekBar;

private SeekBar mDrawableLeftSeekBar, mDrawableTopSeekBar,

mDrawableRightSeekBar, mDrawableBottomSeekBar;

private Button mButton;

private Drawable mDrawableLeft, mDrawableTop, mDrawableRight, mDrawableBottom;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Получим Drawable из ресурсов

mDrawableLeft = getResources().getDrawable(android.R.drawable.ic\_menu\_day);

mDrawableTop = getResources().getDrawable(android.R.drawable.ic\_menu\_gallery);

mDrawableRight = getResources().getDrawable(android.R.drawable.ic\_menu\_compass);

mDrawableBottom = getResources().getDrawable(android.R.drawable.ic\_menu\_camera);

mButton = (Button) findViewById(R.id.button);

// устанавливаем CompoundDrawable на кнопке программно

mButton.setCompoundDrawablesWithIntrinsicBounds(mDrawableLeft,

mDrawableTop, mDrawableRight, mDrawableBottom);

mOpacityButtonSeekBar = (SeekBar) findViewById(R.id.opacitybuttonSeekBar);

mDrawableLeftSeekBar = (SeekBar) findViewById(R.id.opacitydrawableleftSeekBar);

mDrawableTopSeekBar = (SeekBar) findViewById(R.id.opacitydrawabletopSeekBar);

mDrawableRightSeekBar = (SeekBar) findViewById(R.id.opacitydrawablerightSeekBar);

mDrawableBottomSeekBar = (SeekBar) findViewById(R.id.opacitydrawablebottomSeekBar);

mOpacityButtonSeekBar.setOnSeekBarChangeListener(opacityButtonOnSeekBarChangeListener);

mDrawableLeftSeekBar.setOnSeekBarChangeListener(opacityDrawableOnSeekBarChangeListener);

mDrawableTopSeekBar.setOnSeekBarChangeListener(opacityDrawableOnSeekBarChangeListener);

mDrawableRightSeekBar.setOnSeekBarChangeListener(opacityDrawableOnSeekBarChangeListener);

mDrawableBottomSeekBar.setOnSeekBarChangeListener(opacityDrawableOnSeekBarChangeListener);

}

OnSeekBarChangeListener opacityButtonOnSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

float alpha = (float) progress / (float) seekBar.getMax();

mButton.setAlpha(alpha); // for API Level 11+

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

};

OnSeekBarChangeListener opacityDrawableOnSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

mDrawableLeft.setAlpha(mDrawableLeftSeekBar.getProgress());

mDrawableTop.setAlpha(mDrawableTopSeekBar.getProgress());

mDrawableRight.setAlpha(mDrawableRightSeekBar.getProgress());

mDrawableBottom.setAlpha(mDrawableBottomSeekBar.getProgress());

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

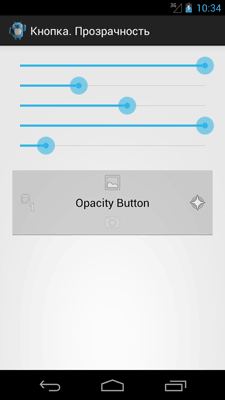
public void onStopTrackingTouch(SeekBar seekBar) {

}

};

}

На рисунке видно, что у правого значка прозрачность нулевая, а у остальных уровень прозрачности немного изменён.



Чтобы сделать кнопку более стильной, можно в XML-файле описать все красоту и связать описание с кнопкой. Создадим в папке /res папку **drawable-nodpi** (использование ресурсов при любом разрешении) или просто **drawable**. В ней расположим два файла buttonstyle1.xml и buttonstyle2.xml:

buttonstyle1.xml

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

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

<item android:state\_pressed="true">

<shape>

<gradient android:angle="90.0"

android:endColor="#BA55D3"

android:startColor="#800080"

android:type="linear" />

<corners android:radius="5.0dp" />

</shape>

</item>

<item>

<shape>

<gradient android:angle="90.0"

android:endColor="#DDA0DD"

android:startColor="#9370D8"

android:type="linear" />

<corners android:radius="5.0dp" />

</shape>

</item>

</selector>

buttonstyle2.xml

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

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

<item android:state\_pressed="true">

<shape>

<gradient android:angle="90.0"

android:endColor="#483D8B"

android:startColor="#00008B"

android:type="linear" />

<corners android:radius="5.0dp" />

</shape>

</item>

<item>

<shape>

<gradient android:angle="90.0"

android:endColor="#6495ED"

android:startColor="#0000FF"

android:type="linear" />

<corners android:radius="5.0dp" />

</shape>

</item>

</selector>

Теперь создайте на форме две кнопки для тестирования и присвойте свойствам **Background** значения **@drawable/buttonstyle1** и **@drawable/buttonstyle2** соответственно.

Сохранив изменения, вы увидите, что теперь у вас вместо серых скучных кнопок настоящие глаМУРные кнопочки. Разберем, что мы тут понаписали.

Мы использовали градиент, они бывают разных типов (linear, radial, sweep). В нашем случае использовался **linear** с поворотом на 90 градусов. Для округления углов кнопок используем специальный тег **corners**. Также можно описать внешний вид кнопки в момент нажатия (**state\_pressed**). Про последний атрибут стоит рассказать подробнее. При создании собственных профессиональных кнопок нужно подготовить несколько ресурсов для различных состояний кнопок - когда она нажата, в фокусе, выбрана, по умолчанию. Для каждого из этих состояний есть специальные атрибуты. Например.

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

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

<item android:drawable="@drawable/button\_pressed"

**android:state\_pressed** = "true" />

<item android:drawable="@drawable/button\_focused"

**android:state\_focused** = "true" />

<item android:drawable="@drawable/button\_default" />

</selector>

В этом примере используются три состояния - нажато, в фокусе и по умолчанию. Следите, чтобы атрибут **android:state\_pressed** всегда был первым, иначе другие атрибуты могут перекрыть эффект нажания и вы его не увидите. Состояние по умолчанию используйте последним.

Программное управление внешним видом кнопки

Для программного управления описаний для кнопок нужно задействовать метод **setBackgroundResource**. Например. если мы хотим программно поменять описания для кнопок между собой, то можно сделать так:

final Button styledButton1 = (Button)findViewById(R.id.button1);

final Button styledButton2 = (Button)findViewById(R.id.button2);

styledButton1.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

styledButton1.setBackgroundResource(R.drawable.buttonstyle2);

styledButton2.setBackgroundResource(R.drawable.buttonstyle1);

}

});

styledButton2.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub 1098

styledButton1.setBackgroundResource(R.drawable.buttonstyle1);

styledButton2.setBackgroundResource(R.drawable.buttonstyle2);

}

});

Используя подобную технику, вы можете создавать кнопки с закругленными сторонами, увеличив радиусы углов (roundbutton.xml):

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

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

android:padding="10dp"

android:shape="rectangle" >

<solid android:color="@android:color/background\_light" />

<corners

android:bottomLeftRadius="20dp"

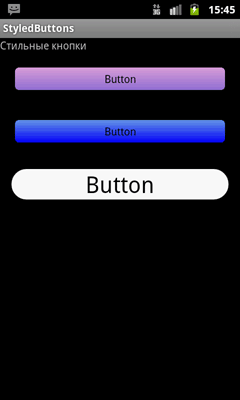
android:bottomRightRadius="20dp"

android:topLeftRadius="20dp"

android:topRightRadius="20dp" />

</shape>

Неплохо получилось, а?



На [одном сайте](http://www.dibbus.com/2011/02/gradient-buttons-for-android/) нашёл несколько шаблонов для градиентных кнопок, которые можно использовать в качестве отправной точки. Создадим ресурсы в папке **drawable** для кнопок с различными цветами:

yellow.xml

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

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

<item android:state\_pressed="true"><shape>

<solid android:color="#f3ae1b" />

<stroke android:width="1dp" android:color="#bb6008" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

<item><shape>

<gradient android:angle="270" android:endColor="#bb6008" android:startColor="#f3ae1b" />

<stroke android:width="1dp" android:color="#bb6008" />

<corners android:radius="4dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

</selector>

red.xml

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

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

<item android:state\_pressed="true"><shape>

<solid android:color="#ef4444" />

<stroke android:width="1dp" android:color="#992f2f" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

<item><shape>

<gradient android:angle="270" android:endColor="#992f2f" android:startColor="#ef4444" />

<stroke android:width="1dp" android:color="#992f2f" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

</selector>

green.xml

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

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

<item android:state\_pressed="true"><shape>

<solid android:color="#70c656" />

<stroke android:width="1dp" android:color="#53933f" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

<item><shape>

<gradient android:angle="270" android:endColor="#53933f" android:startColor="#70c656" />

<stroke android:width="1dp" android:color="#53933f" />

<corners android:radius="4dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

</selector>

purple.xml

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

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

<item android:state\_pressed="true"><shape>

<solid android:color="#a276eb" />

<stroke android:width="1dp" android:color="#6a3ab2" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

<item><shape>

<gradient android:angle="270" android:endColor="#6a3ab2" android:startColor="#a276eb" />

<stroke android:width="1dp" android:color="#6a3ab2" />

<corners android:radius="4dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

</selector>

blue.xml

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

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

<item android:state\_pressed="true"><shape>

<solid android:color="#449def" />

<stroke android:width="1dp" android:color="#2f6699" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

<item><shape>

<gradient android:angle="270" android:endColor="#2f6699" android:startColor="#449def" />

<stroke android:width="1dp" android:color="#2f6699" />

<corners android:radius="4dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp" />

</shape></item>

</selector>

black.xml

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

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

<item android:state\_pressed="true"><shape>

<solid android:color="#343434" />

<stroke android:width="1dp" android:color="#171717" />

<corners android:radius="3dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp"/>

</shape></item>

<item><shape>

<gradient android:angle="270" android:endColor="#171717" android:startColor="#343434" />

<stroke android:width="1dp" android:color="#171717" />

<corners android:radius="4dp" />

<padding android:bottom="10dp" android:left="10dp" android:right="10dp" android:top="10dp"/>

</shape></item>

</selector>

Эти свойства подключаем к кнопкам через атрибут **android:background="@drawable/..."**.

Также зададим стиль для текста к кнопке в файле styles.xml:

<resources>

<style name="AppTheme" parent="android:Theme.Light" />

<style name="ButtonText">

<item name="android:layout\_width">fill\_parent</item>

<item name="android:layout\_height">wrap\_content</item>

<item name="android:textColor">#ffffff</item>

<item name="android:gravity">center</item>

<item name="android:layout\_margin">3dp</item>

<item name="android:textSize">20sp</item>

<item name="android:textStyle">bold</item>

<item name="android:shadowColor">#000000</item>

<item name="android:shadowDx">1</item>

<item name="android:shadowDy">1</item>

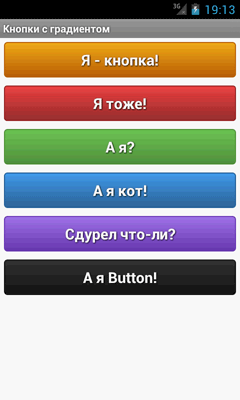
<item name="android:shadowRadius">2</item>

</style>

</resources>

Подключим стили через атрибут **style="@style/ButtonText"**.

Смотрим, что получилось:



Посмотрите на сайте огромное число градиентных кнопок на любой вкус: [Even more gradient buttons for Android | Dibbus.com](http://www.dibbus.com/2011/08/even-more-gradient-buttons-for-android/)

Анимация для кнопок

Применим анимационные эффекты к кнопкам

Эффект Alpha: Изменение прозрачности

Создадим анимацию изменения прозрачности кнопки до полного исчезновения и обратно

res/anim/alpha.xml

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

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

android:interpolator="@android:anim/linear\_interpolator" >

<alpha

android:duration="500"

android:fromAlpha="1.0"

android:repeatCount="1"

android:repeatMode="reverse"

android:toAlpha="0.1" />

</set>

Нажимаем на кнопку

Button buttonAlpha = (Button)findViewById(R.id.buttonAlpha);

final Animation animAlpha = AnimationUtils.loadAnimation(this, R.anim.alpha);

buttonAlpha.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

v.startAnimation(animAlpha);

}

});

Эффект Rotate: Вращение кнопки

Заставим кнопку вращаться при нажатии. Карлсону, который живет на крыше, эта кнопка понравится.

res/anim/rotate.xml

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

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

android:interpolator="@android:anim/linear\_interpolator" >

<rotate

android:duration="500"

android:fromDegrees="0"

android:pivotX="50%"

android:pivotY="50%"

android:repeatCount="1"

android:repeatMode="reverse"

android:startOffset="0"

android:toDegrees="360" />

</set>

Нажимаем на кнопку

Button buttonRotate = (Button)findViewById(R.id.buttonRotate);

final Animation animRotate = AnimationUtils.loadAnimation(this, R.anim.rotate);

buttonRotate.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

v.startAnimation(animRotate);

}

});

Эффект Scale: Изменение размеров

Делаем из мухи слона, т.е. меняем размеры кнопки

res/anim/scale.xml

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

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

android:interpolator="@android:anim/linear\_interpolator" >

<scale

android:duration="500"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:repeatCount="1"

android:repeatMode="reverse"

android:toXScale="3.0"

android:toYScale="3.0" />

</set>

Нажимаем на кнопку

Button buttonScale = (Button)findViewById(R.id.buttonScale);

final Animation animScale = AnimationUtils.loadAnimation(this, R.anim.scale);

buttonScale.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

v.startAnimation(animScale);

}

});

Эффект Translate: Перемещение кнопки

Переместим кнопку на новое место

res/anim/translate.xml

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

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

android:interpolator="@android:anim/linear\_interpolator" >

<translate

android:duration="500"

android:fromXDelta="0"

android:repeatCount="1"

android:repeatMode="reverse"

android:toXDelta="100%p" />

</set>

Щелчок на кнопке

Button buttonTranslate = (Button)findViewById(R.id.buttonTranslate);

final Animation animTranslate = AnimationUtils.loadAnimation(this, R.anim.translate);

buttonTranslate.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

v.startAnimation(animTranslate);

}

});

Посмотреть видео

Источник примеров для анимаций кнопок: [Android-er: Apply animation on Button](http://android-er.blogspot.com/2012/02/apply-animation-on-button.html)

В Android 3.1 (API 12) можно сделать анимацию через новый класс **ViewPropertyAnimator**. Пример [смотрите здесь](http://developer.alexanderklimov.ru/android/animation/viewpropertyanimator.php#hideshow).

Вверх тормашками

А теперь немного похулиганим и создадим необычную кнопку, расположенную вверх тормашками!

Создадим новый класс, который будет наследоваться от класса Button:

// файл InvertedButton.java

package ru.alexanderklimov.views;

import android.content.Context;

import android.graphics.Canvas;

import android.util.AttributeSet;

import android.view.Gravity;

import android.widget.Button;

public class InvertedButton extends Button {

public InvertedButton(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

setWillNotDraw(false);

setGravity(Gravity.CENTER);

}

public InvertedButton(Context context, AttributeSet attrs) {

super(context, attrs);

setWillNotDraw(false);

setGravity(Gravity.CENTER);

}

public InvertedButton(Context context) {

super(context);

setWillNotDraw(false);

setGravity(Gravity.CENTER);

}

@Override

protected void onDraw(Canvas canvas) {

int left = getPaddingLeft();

int top = getPaddingTop();

int right = getPaddingRight();

int bottom = getPaddingBottom();

int width = getWidth() - left - right;

int height = getHeight() - top - bottom;

int saveCount = canvas.getSaveCount();

canvas.translate(left + width / 2, top + height / 2);

canvas.rotate(-180);

canvas.translate((-width / 2)-left, (-height / 2)-top);

canvas.restoreToCount(saveCount);

super.onDraw(canvas);

}

}

Теперь в простом макете вашего шаблона добавим стандартную и нашу необычную кнопку:

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal|bottom"

android:text="Нормальная кнопка" />

<ru.alexanderklimov.views.InvertedButton

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal|bottom"

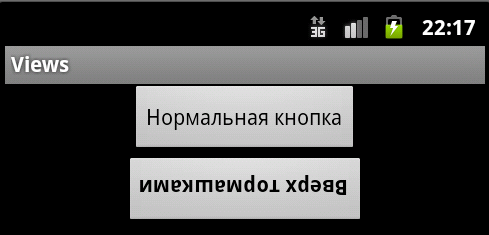
android:text="Вверх тормашками"

android:textStyle="bold" >

</ru.alexanderklimov.views.InvertedButton>

</LinearLayout>

Уже сейчас в графическом режиме просмотра макета вы можете увидеть удивительную кнопку. Запускаем проект и смотрим на нее уже удивленными глазами.



Вся магия сосредоточена в методе **onDraw()**, где мы вращаем объект Canvas. Также проводим некоторые вычисления для точного позиционирования текста.

Зачем это нужно? Чтобы взглянуть на проблему под другим углом!

Манипуляции с кнопкой

Начиная с Android 3.0, кнопку можно переворачивать, вращать и перемещать с помощью готовых методов одной строчкой кода. В документации к Android есть пример с кнопкой, который показывает новые возможности. Так как официальную документацию никто не читает, то продублирую код здесь:

Создадим разметку с кнопкой и ползунками:

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

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

android:id="@+id/container"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:splitMotionEvents="true" >

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="20dip"

android:orientation="horizontal"

android:splitMotionEvents="true" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="5dip"

android:paddingRight="5dip"

android:text="TX"

android:textStyle="bold" />

<SeekBar

android:id="@+id/translationX"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="15dip"

android:paddingRight="5dip"

android:text="TY"

android:textStyle="bold" />

<SeekBar

android:id="@+id/translationY"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="20dip"

android:orientation="horizontal"

android:splitMotionEvents="true" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="5dip"

android:paddingRight="5dip"

android:text="SX"

android:textStyle="bold" />

<SeekBar

android:id="@+id/scaleX"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="15dip"

android:paddingRight="5dip"

android:text="SY"

android:textStyle="bold" />

<SeekBar

android:id="@+id/scaleY"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="20dip"

android:orientation="horizontal"

android:splitMotionEvents="true" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="5dip"

android:paddingRight="5dip"

android:text="X"

android:textStyle="bold" />

<SeekBar

android:id="@+id/rotationX"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="15dip"

android:paddingRight="5dip"

android:text="Y"

android:textStyle="bold" />

<SeekBar

android:id="@+id/rotationY"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:paddingLeft="15dip"

android:paddingRight="5dip"

android:text="Z"

android:textStyle="bold" />

<SeekBar

android:id="@+id/rotationZ"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="horizontal" />

</LinearLayout>

<Button

android:id="@+id/rotatingButton"

android:layout\_width="200dip"

android:layout\_height="150dip"

android:layout\_marginLeft="50dip"

android:layout\_marginTop="50dip"

android:onClick="onClick"

android:text="Click Me" />

</LinearLayout>

И код для примера:

package ru.alexanderklimov.test;

import android.animation.ObjectAnimator;

import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.SeekBar;

public class TestActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

final Button rotatingButton = (Button) findViewById(R.id.rotatingButton);

SeekBar seekBar;

seekBar = (SeekBar) findViewById(R.id.translationX);

seekBar.setMax(400);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setTranslationX((float)progress);

}

});

seekBar = (SeekBar) findViewById(R.id.translationY);

seekBar.setMax(800);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setTranslationY((float)progress);

}

});

seekBar = (SeekBar) findViewById(R.id.scaleX);

seekBar.setMax(50);

seekBar.setProgress(10);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setScaleX((float)progress/10f);

}

});

seekBar = (SeekBar) findViewById(R.id.scaleY);

seekBar.setMax(50);

seekBar.setProgress(10);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setScaleY((float)progress/10f);

}

});

seekBar = (SeekBar) findViewById(R.id.rotationX);

seekBar.setMax(360);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setRotationX((float)progress);

}

});

seekBar = (SeekBar) findViewById(R.id.rotationY);

seekBar.setMax(360);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setRotationY((float)progress);

}

});

seekBar = (SeekBar) findViewById(R.id.rotationZ);

seekBar.setMax(360);

seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onStopTrackingTouch(SeekBar seekBar) {

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

rotatingButton.setRotation((float)progress);

}

});

}

}

Быстрая инициализация кнопок через массив

Один из читателей спросил, как можно быстро инициализировать кнопки в цикле, так как лень набирать код:

Button[] btns = new Button[9];

btns[0] = (Button) findViewById(R.id.btn1);

btns[1] = (Button) findViewById(R.id.btn2);

btns[2] = (Button) findViewById(R.id.btn3);

btns[3] = (Button) findViewById(R.id.btn4);

btns[4] = (Button) findViewById(R.id.btn5);

btns[5] = (Button) findViewById(R.id.btn6);

btns[6] = (Button) findViewById(R.id.btn7);

btns[7] = (Button) findViewById(R.id.btn8);

Как вариант:

Button[] btns = new Button[9];

int[] butIDs = { R.id.btn0, R.id.btn1, R.id.btn2, R.id.btn3,

R.id.btn4, R.id.btn5, R.id.btn6, R.id.btn7,

R.id.btn8 };

for (int i = 0; i < 9; i++) {

btns[i] = (Button) findViewById(butIDs[i]);

btns[i].setText("Я кнопка " + i);

}

В этом примере всё равно приходится создавать массив с идентификаторами. Хотя количество кода уменьшилось. Можно программно также получить идентификаторы кнопок и обойтись без создания массива идентификаторов, но этот способ считается медленным и не особо рекомендуется:

for (int i = 0; i < 9; i++) {

btns[i] = (Button) findViewById(this.getResources().getIdentifier(

"btn" + i, "id", this.getPackageName()));

btns[i].setText("Я кнопка " + i);

}

Круглая кнопка

Используя знакомую нам технику применения drawable-ресурсов, мы можем создать круглую кнопку. Для этого нам понадобится фигура **oval** и установить одинаковые размеры для ширины и высоты кнопки.

Создадим общий файл для круглой кнопки - **res/drawable/circular\_button\_selector.xml**:

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

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

<item android:drawable="@drawable/circular\_button\_pressed" android:state\_pressed="true"/>

<item android:drawable="@drawable/circular\_button\_selected" android:state\_selected="true"/>

<item android:drawable="@drawable/circular\_button"/>

</selector>

В файле упоминаются три состояния кнопки: нажатая, выбранная, по умолчанию. Давайте их создадим.

res/drawable/circular\_button\_pressed.xml

Кнопка должна оставаться овальной, меняем только цвет при нажатии.

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

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

android:shape="oval" >

<solid android:color="@android:color/holo\_purple" />

<stroke

android:width="1dp"

android:color="#AAA" />

</shape>

res/drawable/circular\_button\_selected.xml

Аналогично, форма овальная, меняем цвет.

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

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

android:shape="oval" >

<solid android:color="@android:color/holo\_blue\_dark" />

<stroke

android:width="1dp"

android:color="#AAA" />

</shape>

res/drawable/circular\_button.xml

Стандартный вид кнопки на экране. Задаём овальную фигуру и цвет кнопки.

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

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

android:shape="oval" >

<solid android:color="@android:color/holo\_blue\_bright" />

<stroke

android:width="1dp"

android:color="#AAA" />

</shape>

Прописываем у кнопки нужный фон и устанавливаем одинаковые размеры.

<Button

android:id="@+id/button1"

android:layout\_width="96dp"

android:layout\_height="96dp"

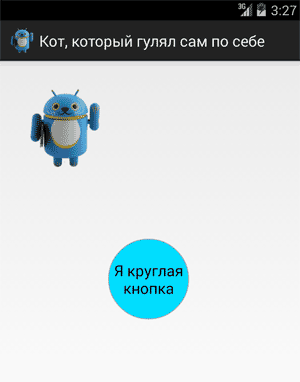
android:layout\_centerVertical="true"

android:layout\_toRightOf="@+id/imageView"

android:background="@drawable/circular\_button\_selector"

android:onClick="onClick"

android:text="Я круглая кнопка" />



Если вы хотите также изменять цвет текста в кнопке, то аналогично создаём в папке **res** новую папку **color** и в ней создаём файл, который присваиваем атрибуту кнопки **android:textColor="@color/button\_text\_selector"**. Сам файл для различных состояний кнопки будет следующим.

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

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

<item android:state\_pressed="true" android:color="#FFF"/>

<item android:state\_selected="true" android:color="#FFF"/>

<item android:color="#777"/>

</selector>

Теперь при нажатии цвет текста у кнопки будет меняться на белый.

Программное добавление кнопок

Добавление любых компонентов в разметку во время выполнения программы осуществляется с помощью метода **addView()**. Рассмотрим пример программного добавления трёх кнопок с равномерным распределением.

Добавьте в свою разметку контейнер **LinearLayout**, в котором будут возникать новые кнопки.

<LinearLayout

android:id="@+id/buttonsContainer"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/button1"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="41dp"

android:orientation="horizontal" >

</LinearLayout>

Код. Мы не только создаём кнопки, но и присваиваем им идентификаторы и текст, а также запоминаем, какая из них является активной. Это может пригодится в том случае, если вы задали для кнопки стили для различных состояний.

package ru.alexanderklimov.test;

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

import ...

public class MainActivity extends Activity implements View.OnClickListener {

private ViewGroup mButtonsContainer;

private Button mActiveButton = null;

private final int MAX\_BUTTONS = 3;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

this.mButtonsContainer = (ViewGroup) findViewById(R.id.buttonsContainer);

int buttonsSpacing = (int) getResources().getDimension(

R.dimen.activity\_horizontal\_margin);

int buttonSize = (int) getResources().getDimension(R.dimen.button\_size);

for (int i = 0; i < MAX\_BUTTONS; i++) {

Button button = new Button(this);

button.setText("Кнопка " + (i + 1));

button.setId(i + 1);

button.setOnClickListener(this);

mButtonsContainer.addView(button);

// добавим пространства между кнопками

if (i != MAX\_BUTTONS - 1) {

mButtonsContainer.addView(new Space(this),

new ViewGroup.LayoutParams(buttonsSpacing, buttonSize));

}

}

selectButton((Button) mButtonsContainer.getChildAt(0));

}

private void selectButton(Button button) {

if (mActiveButton != null) {

mActiveButton.setSelected(false);

mActiveButton = null;

}

mActiveButton = button;

button.setSelected(true);

}

@Override

public void onClick(View view) {

selectButton((Button) view);

Toast.makeText(getApplicationContext(), "Clicked" + view.getId(), Toast.LENGTH\_SHORT)

.show();

}

}

Я вставил контейнер в проект из предыдущих примеров. Три кнопки добавлены при загрузке активности в нижней части экрана.

