FrameLayout

**FrameLayout** является самым простым типом разметки. Обычно это пустое пространство на экране, которое можно заполнить только дочерними объектами **View** или **ViewGroup**. Все дочерние элементы **FrameLayout** прикрепляются к верхнему левому углу экрана.

В разметке **FrameLayout** нельзя определить различное местоположение для дочернего объекта. Последующие дочерние объекты **View** будут просто рисоваться поверх предыдущих компонентов, частично или полностью затеняя их, если находящийся сверху объект непрозрачен, поэтому единственный дочерний элемент для **FrameLayout** обычно растянут до размеров родительского контейнера и имеет атрибуты:

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

Также можно использовать свойства **Gravity** для управления порядком размещения.

**FrameLayout** можно использовать для интересного эффекта, когда верхняя часть (**TextView**) остаётся неизменной, а основной контент прокручивается.

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<TextView

android:id="@+id/tvTop"

android:gravity="top|center\_horizontal"

android:layout\_width="match\_parent"

android:layout\_height="20dip"

android:text="FLOATING VIEW - TOP"

android:textColor="#000000"

android:textStyle="bold"

android:background="#FFFFFF" />

<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<ListView

android:id="@+id/lvMain"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

</FrameLayout>

**Поочередная смена картинки**

Создадим разметку, когда на экран динамически выводится только одно изображение.

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

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

android:id="@+id/FrameLayout1"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<ImageView

android:id="@+id/imageView1"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:scaleType="fitCenter"

android:src="@drawable/cat" />

<ImageView

android:id="@+id/imageView2"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:scaleType="fitCenter"

android:src="@drawable/ic\_launcher"

android:visibility="gone" />

</FrameLayout>

Теперь код для активности, где при нажатии на изображении, оно будет заменяться на другое.

private ImageView img1;

private ImageView img2;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

img1 = (ImageView)findViewById(R.id.imageView1);

img2 = (ImageView)findViewById(R.id.imageView2);

img1.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

img2.setVisibility(View.VISIBLE);

img1.setVisibility(View.GONE);

}

});

img2.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

img2.setVisibility(View.GONE);

img1.setVisibility(View.VISIBLE);

}

});

}

В разметке мы указали, что один из компонентов **ImageView** должен быть невидимым. Если бы мы этого не сделали, то увидели, что компоненты просто накладываются друг на друга в стопку. И если элементы разных размеров, то можно увидеть, как одно изображение частично перекрывает другое.

Имейте в виду, что размер компоновки **FrameLayout** определяется наибольшим элементом в контейнере. Поэтому более маленькое изображение может растянуться.

Если часть элементов сверху стопки являются невидимыми, то могут наблюдаться проблемы с выводом на экран. В этих случаях рекомендуется вызывать метод **setMeasureAllChildren()** с аргументом *true* (аналог метода - атрибут **android:measureAllChildren="true"**).

**Программное создание FrameLayout**

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

TextView tv1 = new TextView(this);

tv1.setText(R.string.top\_text);

tv1.setTextSize(40);

tv1.setTextColor(Color.BLACK);

TextView tv2 = new TextView(this);

tv2.setLayoutParams(new LayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.WRAP\_CONTENT, Gravity.BOTTOM));

tv2.setTextSize(50);

tv2.setGravity(Gravity.RIGHT);

tv2.setText(R.string.bottom\_text);

tv2.setTextColor(Color.WHITE);

ImageView iv1 = new ImageView(this);

iv1.setImageResource(R.drawable.lake);

iv1.setLayoutParams(new LayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.FILL\_PARENT));

iv1.setScaleType(ScaleType.MATRIX);

FrameLayout fl = new FrameLayout(this);

fl.setLayoutParams(new LayoutParams(LayoutParams.FILL\_PARENT, LayoutParams.FILL\_PARENT));

fl.addView(iv1);

fl.addView(tv1);

fl.addView(tv2);

setContentView(fl);

}