**Обзор FrameLayout**

Диспетчер компоновки FrameLayout в основном используется для динамического вывода одного представления, но его также можно заполнить множеством элементов, наложенных друг на друга. Можно определить один элемент как видимый, а остальные - невидимыми.

Если добавить в компоновку много элементов, то диспетчер FrameLayout просто уложит их в стопку, один поверх другого, причём последний элемент будет наверху. Таким образом, можно получить достаточно интересный интерфейс, чем сейчас и займёмся.

**Пример 1**

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

**android:layout\_width**=**"fill\_parent"**

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

<**ImageView**

**android:layout\_width**=**"fill\_parent"**

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

**android:scaleType**=**"center"**

**android:src**=**"@drawable/pic2"**/>

<**TextView**

**android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:layout\_marginBottom**=**"20dip"**

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

**android:padding**=**"12dip"**

**android:background**=**"#AA000000"**

**android:textColor**=**"#ffffffff"**

**android:text**=**"Клёвые туфли"**/>

<**ImageView** **android:layout\_width**=**"wrap\_content"**

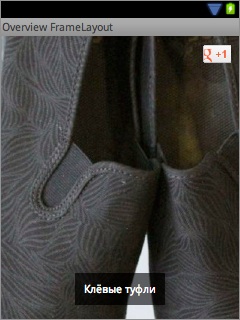
**android:layout\_height**=**"wrap\_content"**

**android:src**=**"@drawable/google\_plus"**

**android:layout\_gravity**=**"right|top"**

**android:layout\_margin**=**"12dip"**/>

</**FrameLayout**>



Пример очень простой и наглядно показывает преимущества компоновки.  
Для начала мы добавили картинку во всю ширину/высоту экрана и добавили текст с подписью и картинку кнопки Google+, а далее используя атрибут android:layout\_gravity, выровняли в макете. Текст выровнен по центру горизонтали и по низу экрана android:layout\_gravity="center\_horizontal|bottom", а кнопка Google+ выровнена по правому краю и сверху экрана android:layout\_gravity="right|top".  Как видно, они оказались поверх картинки.

**Пример 2**

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

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

**android:layout\_width**=**"fill\_parent"**

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

**android:background**=**"#fff"**

>

<**View** **android:layout\_width**=**"250dp"**

**android:layout\_height**=**"330dp"**

**android:background**=**"#dddddd"**

**android:layout\_gravity**=**"center"**

/>

<**View** **android:layout\_width**=**"150dp"**

**android:layout\_height**=**"250dp"**

**android:background**=**"#aaaaaa"**

**android:layout\_gravity**=**"center"**

/>

<**View** **android:layout\_width**=**"100dp"**

**android:layout\_height**=**"150dp"**

**android:background**=**"#666666"**

**android:layout\_gravity**=**"center"**

/>

<**View** **android:layout\_width**=**"50dp"**

**android:layout\_height**=**"100dp"**

**android:background**=**"#333333"**

**android:layout\_gravity**=**"center"**

/>

<**View** **android:layout\_width**=**"20dp"**

**android:layout\_height**=**"50dp"**

**android:background**=**"#000000"**

**android:layout\_gravity**=**"center"**

/>

<**View** **android:layout\_width**=**"5dp"**

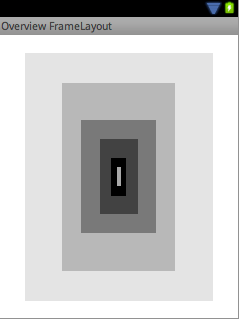
**android:layout\_height**=**"25dp"**

**android:background**=**"#999999"**

**android:layout\_gravity**=**"center"**

/>

</**FrameLayout**>



В этом примере, мы один за другим добавляем элемент View с фоновой заливкой - всего 6 штук, но с разными шириной и высотой и выровненной по центру. Получились разноцветные прямоугольники, располагающиеся друг на друге. Элементы, добавленные позже, располагаются поверх тех, что были добавлены раньше. Если бы мы в конце добавили элемент на всю ширину и высоту экрана, то он бы перекрыл все предыдущие элементы и был бы только один видим, пока бы его не сделали невидимым с помощью android:visibility="gone". Помните в начале статьи, я писал об этом, что можно определить один элемент как видимый, а остальные - невидимыми. Реализуем этот пример:

**Пример 3 - xml компоновка**

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

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

**android:layout\_width**=**"fill\_parent"**

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

>

<**ImageView** **android:layout\_width**=**"fill\_parent"**

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

**android:scaleType**=**"fitCenter"**

**android:src**=**"@drawable/pic1"**

**android:id**=**"@+id/pic1"**

**android:onClick**=**"onClick"**

/>

<**ImageView** **android:layout\_width**=**"fill\_parent"**

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

**android:scaleType**=**"fitCenter"**

**android:src**=**"@drawable/pic2"**

**android:id**=**"@+id/pic2"**

**android:visibility**=**"gone"**

**android:onClick**=**"onClick"**

/>

<**ImageView** **android:layout\_width**=**"fill\_parent"**

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

**android:scaleType**=**"fitCenter"**

**android:src**=**"@drawable/pic3"**

**android:id**=**"@+id/pic3"**

**android:visibility**=**"gone"**

**android:onClick**=**"onClick"**

/>

</**FrameLayout**>

Тут мы добавили 3 изображения друг на друга и 2 из них скрыли атрибутом android:visibility="gone", в итоге у нас получилось видимым только 1 изображение.



Напишем обработку нажатия по изображению. В XML макете у элементов ImageView добавим атрибут android:onClick="onClick". Это обозначает, что при клике будет вызван метод onClick(View view). Зададим в коде, что при клике текущее изображение будет скрыто, а следующее отображено.

**Пример 3 - java code**

**package** ru.devpad.framelayout;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.ImageView;

**public** **class** ExampleActivity3 **extends** Activity {

**private** ImageView pic1;

**private** ImageView pic2;

**private** ImageView pic3;

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

**super**.onCreate(savedInstanceState);

setContentView(R.layout.framelayout\_ex3);

pic1 = (ImageView) findViewById(R.id.pic1);

pic2 = (ImageView) findViewById(R.id.pic2);

pic3 = (ImageView) findViewById(R.id.pic3);

}

**public** **void** onClick(View view) {

**switch** (view.getId()) {

**case** R.id.pic1:

pic1.setVisibility(View.GONE);

pic2.setVisibility(View.VISIBLE);

pic3.setVisibility(View.GONE);

**break**;

**case** R.id.pic2:

pic1.setVisibility(View.GONE);

pic2.setVisibility(View.GONE);

pic3.setVisibility(View.VISIBLE);

**break**;

**case** R.id.pic3:

pic1.setVisibility(View.VISIBLE);

pic2.setVisibility(View.GONE);

pic3.setVisibility(View.GONE);

**break**;

}

}

}

В коде, находим все 3 изображения по их id. В методе onClick(View view) обрабатываем нажатие для соответствующей картинки по её id. Скрываем текущую картинку, показываем следующую и так далее.  
  
Посмотрите на видео работы:

**Атрибут android:measureAllChildren**

Атрибут android:measureAllChildren указывает учитывать ли размер всех элементов, в том числе и скрытых (android:visibility="gone") или учитывать только элементы с видимостью visible и invisible.

Давайте рассмотрим на примере. Зададим для компоновки FrameLayout, ширину и высоту только необходимую wrap\_content и зададим фоновый цвет в чёрный. Далее добавим 4 квадрата ~~Малевича~~ разных размеров и 2 самых крупных скроем атрибутом android:visibility="gone".

**Пример 4**

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

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

**android:layout\_width**=**"fill\_parent"**

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

<**FrameLayout** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:background**=**"#000"**

>

<**View** **android:layout\_width**=**"300dp"**

**android:layout\_height**=**"300dp"**

**android:visibility**=**"gone"**

/>

<**View** **android:layout\_width**=**"200dp"**

**android:layout\_height**=**"200dp"**

**android:visibility**=**"gone"**

/>

<**View** **android:layout\_width**=**"100dp"**

**android:layout\_height**=**"100dp"**

/>

<**View** **android:layout\_width**=**"50dp"**

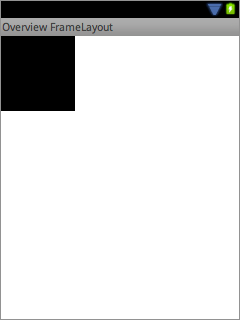
**android:layout\_height**=**"50dp"**

/>

</**FrameLayout**>

</**LinearLayout**>

Самым большим видимым элементом получился элемент с шириной и высотой в 100dp.



Теперь давайте добавим для контейнра FrameLayout атрибут android:measureAllChildren.

<**FrameLayout** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

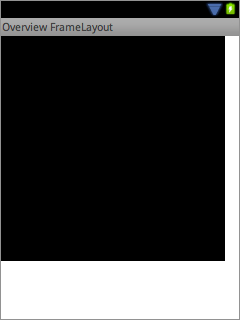
**android:background**=**"#000"**

**android:measureAllChildren**=**"true"**

**android:id**=**"@+id/frame"**

>

Теперь размер FrameLayout должен учитываться даже со скрытыми элементами, т. е. теперь он должен быть равен самому большому элементу в компоновке – 300dp по ширине и высоте.   
Запускаем и получаем:



В JAVA этот атрибут можно установить при помощи метода setMeasureAllChildren(boolean measureAll).

FrameLayout frameLayout = (FrameLayout) findViewById(R.id.frame);

frameLayout.setMeasureAllChildren(**true**);

**Атрибут android:foreground**

Данный атрибут принимает объект вида Drawable и отображает его, как изображение переднего фона, т. е. выводит поверх всего содержимого.

**Пример 5.1**

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

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

**android:orientation**=**"vertical"**

**android:gravity**=**"center"**

**android:layout\_width**=**"fill\_parent"**

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

<**FrameLayout** **android:layout\_width**=**"250dp"**

**android:layout\_height**=**"250dp"**

**android:background**=**"#bbbeee"**

**android:foreground**=**"@drawable/android"**

>

<**ImageView** **android:layout\_width**=**"fill\_parent"**

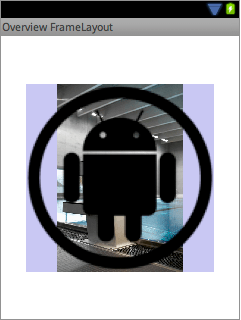
**android:layout\_height**=**"fill\_parent"**

**android:scaleType**=**"fitCenter"**

**android:src**=**"@drawable/pic1"**/>

</**FrameLayout**>

</**LinearLayout**>



Здесь мы создали контейнер FrameLayout с картинкой внутри и добавили атрибут android:foreground с объектом drawable,  в котором содержится картинка с андроидом чёрного цвета. Картинка с андроидом стала на переднем фоне, растянувшись на всю ширину и высоту нашего контейнера и перекрыв собой задний фон.

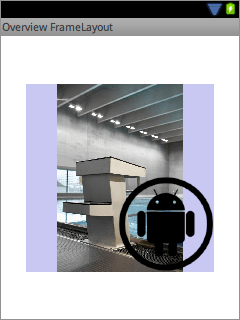
**Атрибуты android:foreground и android:foregroundGravity**

С помощью атрибута android:foregroundGravity можно выровнять изображение перднего фона. Как и в атрибутах gravity и layout\_gravity, значения можно задавать через *побитовое или*|.

Изменим предыдущий код,  добавив новый атрибут.

android:foregroundGravity="right|bottom"

Мы задали выравнивание по правому краю и по низу нашей компоновки.  
Вот,  что теперь у нас получилось:



При добавлении атрибута android:foregroundGravity, картинка переднего плана уже не растягивается на весь экран, а принимает значение wrap\_content для ширины и высоты.  
  
Полный код примера:

**Пример 5.2**

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

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

**android:orientation**=**"vertical"**

**android:gravity**=**"center"**

**android:layout\_width**=**"fill\_parent"**

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

<**FrameLayout** **android:layout\_width**=**"250dp"**

**android:layout\_height**=**"250dp"**

**android:background**=**"#bbbeee"**

**android:foreground**=**"@drawable/android"**

**android:foregroundGravity**=**"right|bottom"**

>

<**ImageView** **android:layout\_width**=**"fill\_parent"**

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

**android:scaleType**=**"fitCenter"**

**android:src**=**"@drawable/pic1"**/>

</**FrameLayout**>

</**LinearLayout**>

**Программное создание компоновки FrameLayout**

Рассмотрим создание контейнера компоновки FrameLayout программно на JAVA.  
Для этого возьмём [пример #1](https://devpad.ru/post/item/8/android-overview-framelayout.html#framelayout_ex1) и создадим тоже самое, но полностью java кодом.

**Пример 6**

**public** **class** ExampleActivity6 **extends** Activity {

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

**super**.onCreate(savedInstanceState);

setTitle(**"FrameLayout programmatically"**);

ImageView image = **new** ImageView(**this**);

image.setLayoutParams(

**new** FrameLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.FILL\_PARENT

)

);

image.setScaleType(ImageView.ScaleType.CENTER);

image.setImageResource(R.drawable.pic2);

TextView description = **new** TextView(**this**);

FrameLayout.LayoutParams paramDescription = **new** FrameLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

paramDescription.bottomMargin = 20;

paramDescription.gravity = Gravity.CENTER\_HORIZONTAL | Gravity.BOTTOM;

description.setLayoutParams(paramDescription);

description.setPadding(12, 12, 12, 12);

description.setBackgroundColor(0xAA000000);

description.setTextColor(0xffffffff);

description.setText(**"Клёвые туфли"**);

ImageView googlePlus = **new** ImageView(**this**);

FrameLayout.LayoutParams paramGooglePlus = **new** FrameLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

paramGooglePlus.setMargins(12, 12, 12, 12);

paramGooglePlus.gravity = Gravity.RIGHT | Gravity.TOP;

googlePlus.setLayoutParams(paramGooglePlus);

googlePlus.setImageResource(R.drawable.google\_plus);

FrameLayout frameLayout = **new** FrameLayout(**this**);

frameLayout.setLayoutParams(

**new** FrameLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.FILL\_PARENT

)

);

frameLayout.addView(image);

frameLayout.addView(description);

frameLayout.addView(googlePlus);

setContentView(frameLayout);

}

}

Исходники обзора, как всегда на гитхабе - <https://github.com/devpad/Android-FrameLayout>