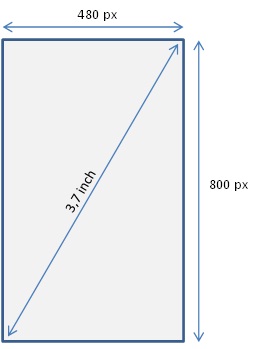
**7. Layout параметры для View-элементов.**

**Экраны**

Для начала немного теории по экранам. Экран имеет такие физические характеристики как диагональ и разрешение. Диагональ – это расстояние между противоположными углами экрана, обычно измеряется в дюймах. Разрешение – кол-во точек по горизонтали и вертикали, которое экран способен отобразить, измеряется в пикселах.

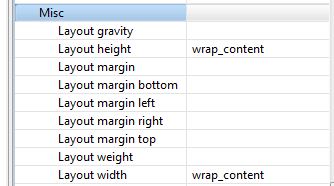
Возьмем в качестве примера экран смартфона HTC Desire. Диагональ = 3,7 дюйма, разрешение = 800х480 пикселов.



Кол-во пикселов в одном дюйме называется dpi (dot per inch). Узнаем чему равно dpi в данном случае, вспомнив классику: c2 = a2 + b2, где с – кол-во пикселей по диагонали, т.е. вмещаемое в 3,7 дюйма. a и b – стороны экрана.

c = 3,7 \* dpi  
(3,7 \* dpi)2 = 4802 + 8002dpi2 = 870400 / 13,69 = 63579  
dpi = 252. Т.е. в одном дюйме экрана помещается ряд из 252 пикселов.

Возвращаемся к теме урока. Рассмотрим подробно следующие параметры View элементов



**Layout width и Layout height**

Про ширину (layout\_width) и высоту (layout\_height) мы уже немного говорили на прошлом уроке. Мы можем указывать для них абсолютные значения, а можем использовать константы. Разберем подробнее эти возможности.

АБСОЛЮТНЫЕ ЗНАЧЕНИЯ:

Используются следующие единицы измерения (ЕИ):

**dp**или **dip**- Density-independent Pixels. Абстрактная ЕИ, позволяющая приложениям выглядеть одинаково на различных экранах и разрешениях.

**sp**- Scale-independent Pixels. То же, что и dp, только используется для размеров шрифта в View элементах

**pt**- 1/72 дюйма, определяется по физическому размеру экрана. Эта ЕИ из типографии.

**px**– пиксел, не рекомендуется использовать т.к. на разных экранах приложение будет выглядеть по-разному.

**mm**– миллиметр, определяется по физическому размеру экрана

**in**– дюйм, определяется по физическому размеру экрана

Подробней о различиях и соотношениях между этими ЕИ вы можете прочесть в этом материале сайта.

КОНСТАНТЫ

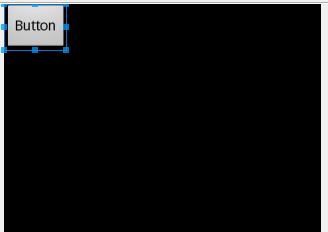
match\_parent (fill\_parent) – означает, что элемент займет всю доступную ему в родительском элементе ширину/высоту.

wrap\_content – ширна/высота элемента будет определяться его содержимым

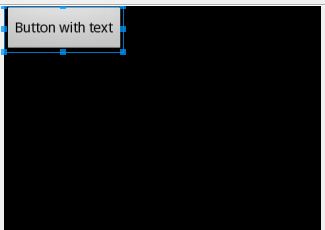
Создадим проект:

**Project name**: P0072\_LayoutProp  
**Build Target**: Android 2.3.3  
**Application name**: LayoutProp  
**Package name**: ru.startandroid.develop.layoutprop  
**Create Activity**: MainActivity

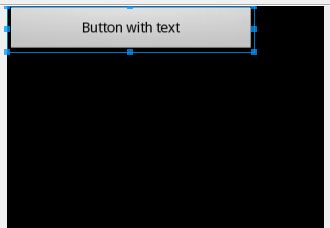
Открываем **main.xml**. Настроим корневой **LinearLayout** на горизонтальную ориентацию, удалим **TextView**, и добавим **Button**с шириной и высотой равной**wrap\_content**. Она отображается на экране и ее ширина соответствует тексту на ней.



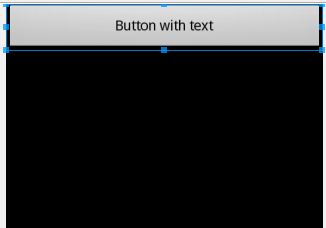
Изменим текст с «Button» на «Button with text», сохраним и посмотрим на экран.



Кнопка стала шире, т.к. ширина определяется по содержимому. Если же мы сейчас явно укажем ей ширину 250 dp, то кнопка растянется независимо от содержимого.

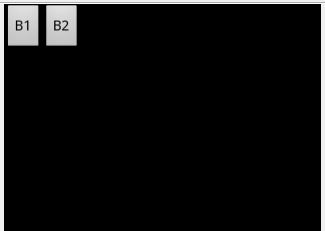


Теперь сделаем ширину равной **match\_parent**. Кнопка растянулась на всю**ширину родителя**, т.е. LinearLayout. А LinearLayout в свою очередь занимет всю ширину экрана.

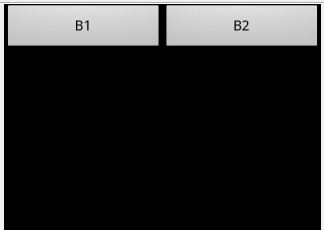


Если у нас родитель содержит несколько элементов и мы хотим, чтобы они заняли все пространство необходимо использовать параметр**Layout weight – вес**. Свободное пространство распределяется между элементами пропорционально их weight-значениям.

Изменим текст нашей кнопки на **B1**и добавим ей соседа по LinearLayout – вторую кнопку с текстом **B2**. Ширину для обоих поставьте **wrap\_content**

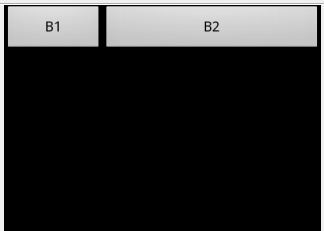


Займемся дележом. Если мы хотим, чтобы кнопки поделили пространство родителя **поровну**– то для обеих укажем **weight**= 1. В этом случае кнопки равны по ширине.

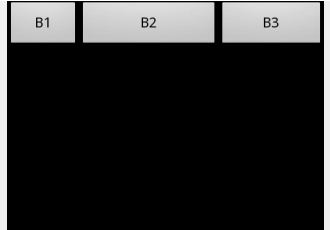


Обратите внимание, что не используются единицы измерения, указываются просто числа.

Если нужно, чтобы B1 занимала четверть, а B2 три четверти свободного пространства, то проставляем **weight**= 1 для **B1** и **weight**= 3 для **B2**.



Кол-во элементов может быть любым. Добавим еще кнопку с текстом **B3**, **weight**= 2 и **width**= wrap\_content.



xml-код получившегося экрана:

<?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"   
android:orientation="horizontal">   
<Button   
android:id="@+id/button1"   
android:layout\_height="wrap\_content"   
android:layout\_width="wrap\_content"   
android:text="B1"   
android:layout\_weight="1">   
</Button>   
<Button   
android:id="@+id/button2"   
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:id="@+id/button3"   
android:text="B3"   
android:layout\_weight="2">   
</Button>   
</LinearLayout>

Теперь для **B2**и **B3**укажите **weight**= 0. Они больше не претендуют на свободное пространство и занимают ширину по содержимому, а **B1**забирает все себе.



Разумеется, все выше сказанное применимо и для параметра высоты - **height**.

При использовании **weigth**вы можете указать значение **height**или **width**=**0dp**. В этом случае не будет учитываться содержимое элементов и результат будет более соответствующий коэффициентам веса.

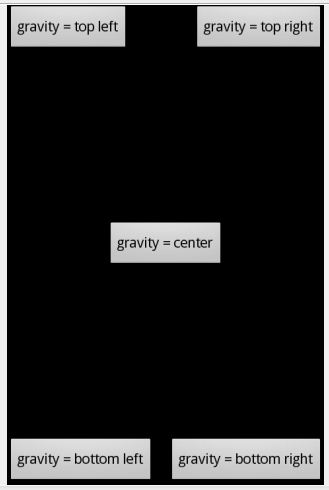
**Layout gravity**

Параметр layout\_gravity аналогичен выравниванию из Word или Excel. Удобнее всего продемонстрировать его с использованием **FrameLayout**. Я не описывал этот Layout на прошлом уроке, т.к. он совсем простой. Все помещаемые в него элементы он по умолчанию помещает в левый верхний угол и никак их не выстраивает. Нам это очень подходит для демонстрации настроек выравнивания.

Создадим **grlayout.xml**:

<?xml version="1.0" encoding="utf-8"?>   
<LinearLayout   
xmlns:android="http://schemas.android.com/apk/res/android"   
android:orientation="vertical"   
android:layout\_width="match\_parent"   
android:layout\_height="match\_parent">   
<FrameLayout   
android:id="@+id/frameLayout1"   
android:layout\_width="match\_parent"   
android:layout\_height="match\_parent">   
<Button   
android:id="@+id/button1"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content"   
android:layout\_gravity="top|left"   
android:text="gravity = top left">   
</Button>   
<Button   
android:id="@+id/button2"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content"   
android:layout\_gravity="top|right"   
android:text="gravity = top right">   
</Button>   
<Button   
android:id="@+id/button3"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content"   
android:layout\_gravity="bottom|left"   
android:text="gravity = bottom left">   
</Button>   
<Button   
android:id="@+id/button4"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content"   
android:layout\_gravity="bottom|right"   
android:text="gravity = bottom right">   
</Button>   
<Button   
android:id="@+id/button5"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content"   
android:layout\_gravity="center"   
android:text="gravity = center">   
</Button>   
</FrameLayout>   
</LinearLayout>

На экране видим:



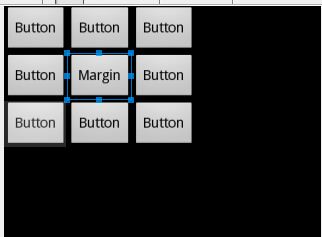
Для наглядности текст кнопки отображает ее свойства. Все очевидно и несложно.

Я честно пытался понять зачем нужны значения gravity **fill\_\*** и **clip\_\***, но так и не понял. То, что написано про них в хелпе у меня не работает. Если у вас есть сведения по этому поводу – пишите в каменты.

**Layout margin**

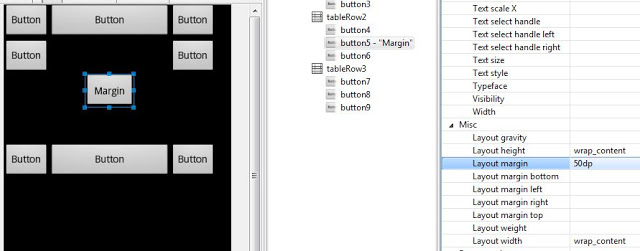
Параметры margin полностью аналогичны **margin** из **html**. Это отступ. Он может быть со всех сторон сразу, либо только с необходимых сторон. Продемонстрируем это на примере TableLayout. Создадим **marginlayout.xml** и нарисуем таблицу три на три с кнопками.

<?xml version="1.0" encoding="utf-8"?>   
<LinearLayout   
xmlns:android="http://schemas.android.com/apk/res/android"   
android:orientation="vertical"   
android:layout\_width="match\_parent"   
android:layout\_height="match\_parent">   
<TableLayout   
android:layout\_height="wrap\_content"   
android:layout\_width="match\_parent"   
android:id="@+id/tableLayout1">   
<TableRow   
android:id="@+id/tableRow1"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
<Button   
android:text="Button"   
android:id="@+id/button1"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
<Button   
android:text="Button"   
android:id="@+id/button2"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
<Button   
android:text="Button"   
android:id="@+id/button3"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
</TableRow>   
<TableRow   
android:id="@+id/tableRow2"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
<Button   
android:text="Button"   
android:id="@+id/button4"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
<Button   
android:text="Margin"   
android:id="@+id/button5"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
<Button   
android:text="Button"   
android:id="@+id/button6"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
</TableRow>   
<TableRow   
android:id="@+id/tableRow3"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
<Button   
android:text="Button"   
android:id="@+id/button7"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
<Button   
android:text="Button"   
android:id="@+id/button8"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
<Button   
android:text="Button"   
android:id="@+id/button9"   
android:layout\_width="wrap\_content"   
android:layout\_height="wrap\_content">   
</Button>   
</TableRow>   
</TableLayout>   
</LinearLayout>

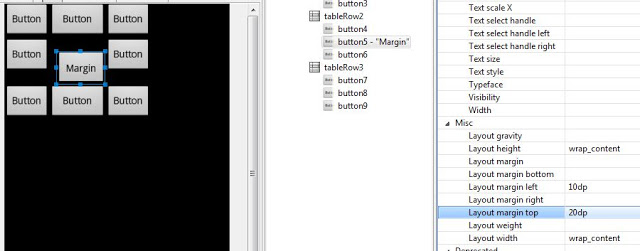


И на кнопке в центре будем экспериментировать.

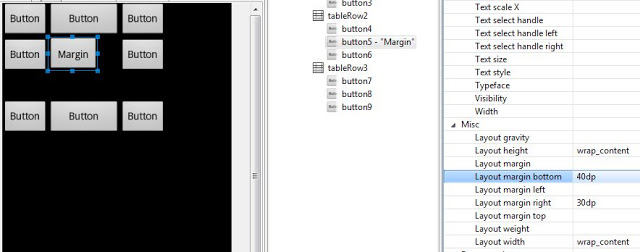
**margin**= 50 dp  
Вокруг кнопки со всех сторон образовался отступ = 50 dp.



**margin left** = 10 dp   
**margin** **top** = 20 dp  
Отступ слева и сверху.



**margin right**= 30 dp  
**margin** **bottom**= 40 dp  
Отступ справа и снизу.



Урок получился большой, но полезный. Думаю, это был последний урок дизайна, моделирования и верстки и дальше мы уже начнем кодить.

**Стили**

Если кто использовал HTML, то наверняка слышали про каскадные стили - CSS. Стили позволяют вам группировать атрибуты элементов (кнопок, таблиц, параграфов и т.д.). Далее вы просто применяете к элементам стили, и элемент рисуется с учетом всех атрибутов стиля. И нет необходимости повторять несколько раз один и тот же код для элементов, которые должны выглядеть одинаково. Особенно это удобно в случае изменения атрибутов. Вы просто меняете один раз стиль и все элементы с этим стилем меняются.

В Android тоже есть стили и они имеют точно такое же назначение. Если у вас есть несколько элементов и вам надо, чтобы они выглядели одинаково, то вы просто создаете один стиль и применяете его к нужным элементам. В принципе, вы пока можете пока не заморачиваться этим и начать использовать стили, когда наберетсь опыта. Ну а тем кому это интересно прямо сейчас - прошу в эту ветку нашего форума. Пользователь icamys на примере подробно разъясняет как использовать стили.