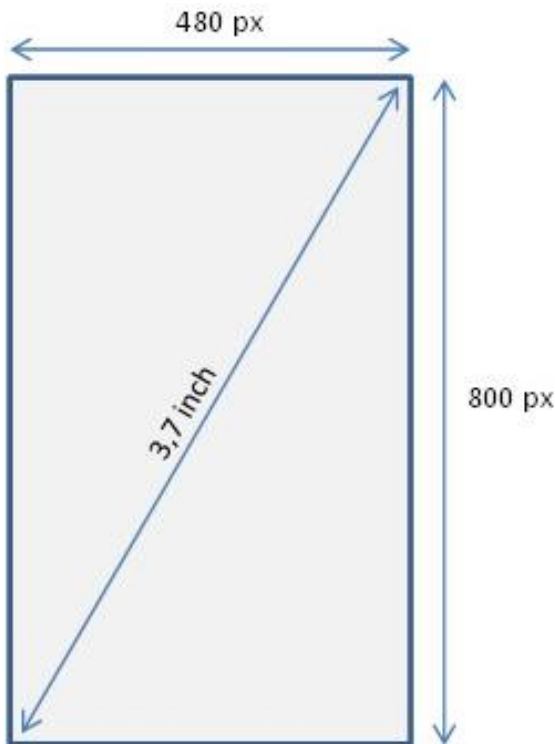


# Layout параметры для View-элементов.

## 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 пикселях. Возьмем в качестве примера экран смартфона.

Пусть диагональ = 3,7 дюйма, разрешение = 800x480 пикселей.

Кол-во пикселей в одном дюйме называется dpi (dot per inch). Узнаем чему равно dpi в данном случае, вспомнив классику:  $c^2 = a^2 + b^2$ , где  $c$  – кол-во пикселей по диагонали, т.е. вмещаемое в 3,7 дюйма.  $a$  и  $b$  – стороны экрана.

$$c = 3,7 * dpi$$

$$(3,7 * dpi)^2 = 480^2 + 800^2$$

$$dpi^2 = 870400 / 13,69 = 63579$$

$$dpi = 252.$$

Т.е. в одном дюйме экрана помещается ряд из 252 пикселей.

## XML АТРИБУТЫ

Каждый объект View и ViewGroup поддерживает свое собственное разнообразие XML атрибутов. Некоторые атрибуты являются определенными только в объекте View (например, TextView поддерживает атрибут textSize), но эти атрибуты могут также наследоваться любыми объектами View, которые расширяют этот класс. Некоторые атрибуты являются общими ко всем объектам View, потому что они унаследованы от корневого класса View (подобно атрибутам id, layout\_width, layout\_height).

## LAYOUT WIDTH И LAYOUT HEIGHT

Рассмотрим следующие параметры View элементов: ширина (layout\_width) и высота (layout\_height). Для них можно указывать как абсолютные значения, так и 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.

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

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

Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	
Layout margin bottom	
Layout margin left	
Layout margin right	
Layout margin top	
Layout weight	
Layout width	wrap_content

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

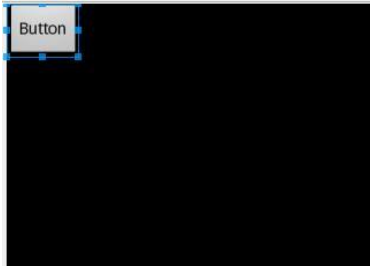
### Константы:

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

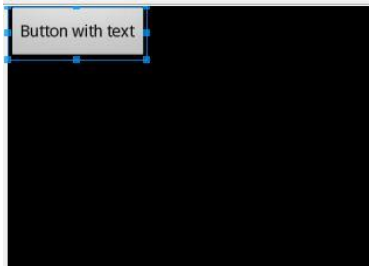
**wrap\_content** – ширина/высота элемента будет определяться его содержимым

Рассмотрим на примере: Создадим новый проект, в котором в **main.xml** настроим корневой **LinearLayout** на горизонтальную ориентацию. Далее удалим **TextView**, и добавим **Button** с шириной и высотой равной **wrap\_content**.

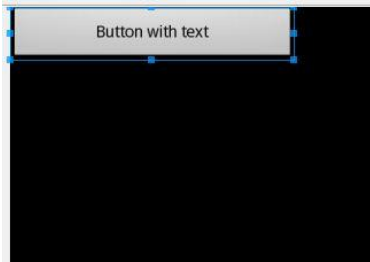
Кнопка отобразится на экране, и ее ширина соответствует тексту на ней:



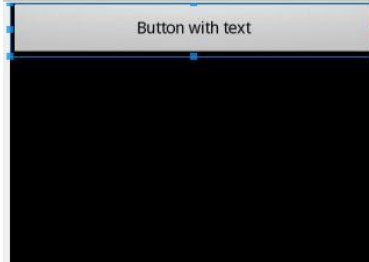
Далее изменим текст с «Button» на «Button with text», сохраним и посмотрим на экран:



Если явно указать ей ширину 250 dp, то кнопка растянется независимо от содержимого:

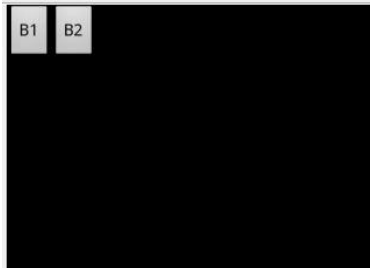


Теперь сделаем ширину равной **match\_parent**. Кнопка растянулась на всю ширину родителя, т.е. **LinearLayout**. А **LinearLayout** в свою очередь занимает всю ширину экрана:



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.

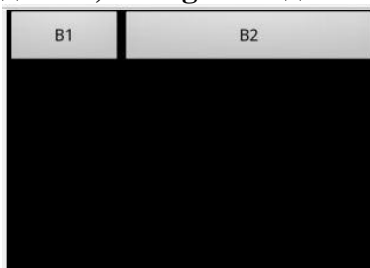
Создадим две кнопки: **B1, B2**.  
Ширину для обоих поставим **wrap\_content**:



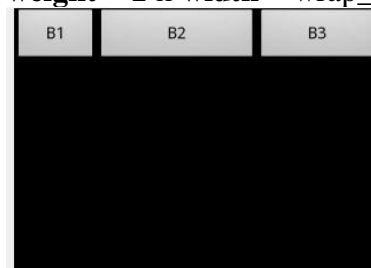
Если мы хотим, чтобы кнопки поделили пространство родителя **поровну** – то для обеих укажем **weight = 1**:



Если нужно, чтобы **B1** занимала четверть свободного пространства, то поставим **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**:



## GRAVITY

«Гравитация» может быть задана двумя способами:

- Атрибутом **gravity** у лэйаута. В таком случае она будет применена для всех дочерних элементов
- Атрибутом **layout\_gravity** у дочернего элемента. Тогда она будет применена только для этого элемента.

Gravity задает положение элемента внутри контейнера. Гравитация может быть следующей:

- **bottom** — элемент «прижимается» к нижней границе контейнера.
- **center** — элемент располагается в центре контейнера
- **center\_horizontal** — элемент находится в центре по оси X
- **center\_vertical** — элемент находится в центре по оси Y
- **end** — элемент находится «в конце» контейнера. Обычно это означает, что он будет находиться справа, но на **layout** с написанием справа-налево (RTL локалях) он будет находиться слева.
- **start** — элемент находится «в начале» контейнера. Обычно — слева, на RTL локалях — справа.
- **top** — элемент «прижимается» к верхней границе контейнера.

**left** и **right** использовать не рекомендуется, поскольку это может вызвать проблемы с версткой.

Поэкспериментируйте с уже созданными кнопками, задавая разные варианты гравитации.

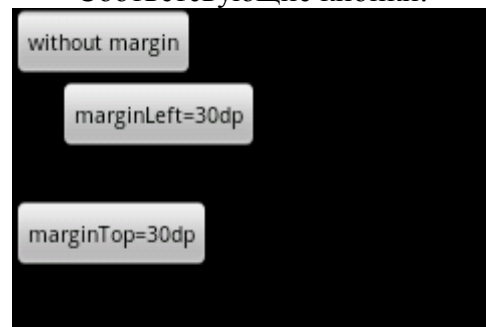
## LAYOUT MARGIN

Параметр определяет отступ элемента от соседних элементов (или от границы родительского)

Он может быть со всех сторон сразу, либо только с необходимых сторон: `layout_marginLeft`, `layout_marginRight`, `layout_marginTop`, `layout_marginBottom`. Рассмотрим xml:

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3.     android:layout_width="fill_parent"
4.     android:layout_height="fill_parent"
5.     android:orientation="vertical">
6.     <Button
7.         android:layout_width="wrap_content"
8.         android:layout_height="wrap_content"
9.         android:text="without margin"
10.    />
11.     <Button
12.         android:layout_width="wrap_content"
13.         android:layout_height="wrap_content"
14.         android:text="marginLeft=30dp"
15.         android:layout_marginLeft="30dp"
16.    />
17.     <Button
18.         android:layout_width="wrap_content"
19.         android:layout_height="wrap_content"
20.         android:text="marginTop=30dp"
21.         android:layout_marginTop="30dp"
22.    />
23. </LinearLayout>
```

Соответствующие кнопки:



Для отступа внутри элемента используются параметры padding, paddingLeft, paddingRight, paddingTop, paddingBottom:

