

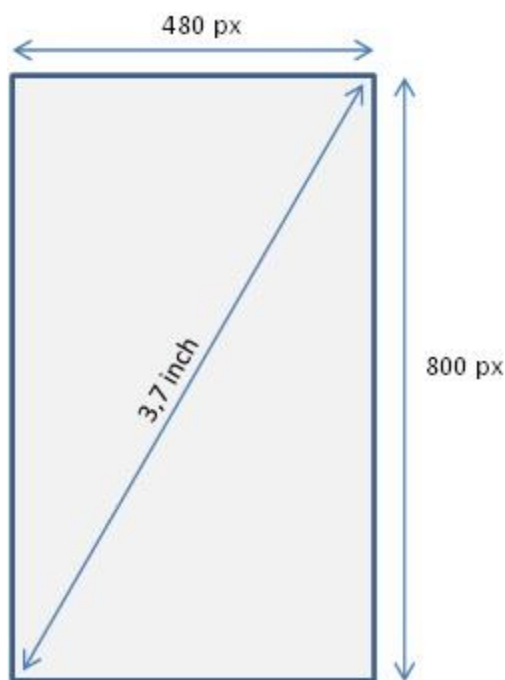
Урок 6. «Layout параметры для View-элементов»¹.

На этом уроке мы разберёмся в характеристиках экрана и рассмотрим layout-параметры (высота, ширина, отступ, гравитация, вес).

Экраны

Экран имеет такие физические характеристики как диагональ и разрешение. Диагональ – это расстояние между противоположными углами экрана, обычно измеряется в дюймах. Разрешение – количество точек по горизонтали и вертикали, которое экран способен отобразить, измеряется в пикселях.

Возьмем в качестве примера экран смартфона HTC Desire. 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.

$$c = 3,7 * dpi$$

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

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

$$dpi = 252. \text{ Т.е. в одном дюйме экрана помещается ряд из 252 пикселей.}$$

Возвращаемся к теме урока.

Рассмотрим подробно следующие параметры View элементов

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

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 значения²

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 экрана³.

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

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

Создадим проект:

Project name: P0072_LayoutProp

Build Target: Android 2.3.3

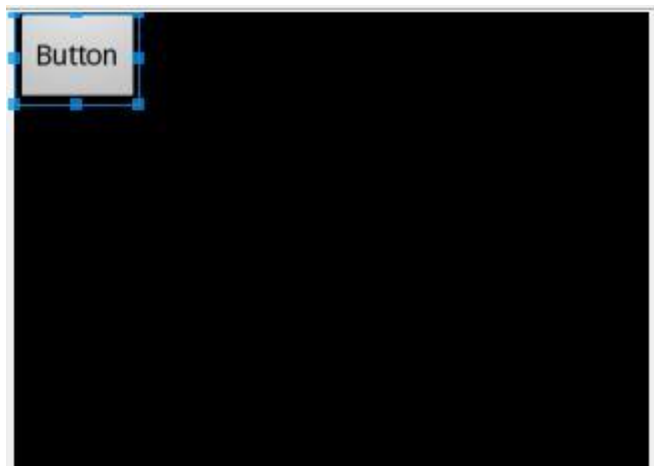
Application name: LayoutProp

Package name: ru.startandroid.develop.layoutprop

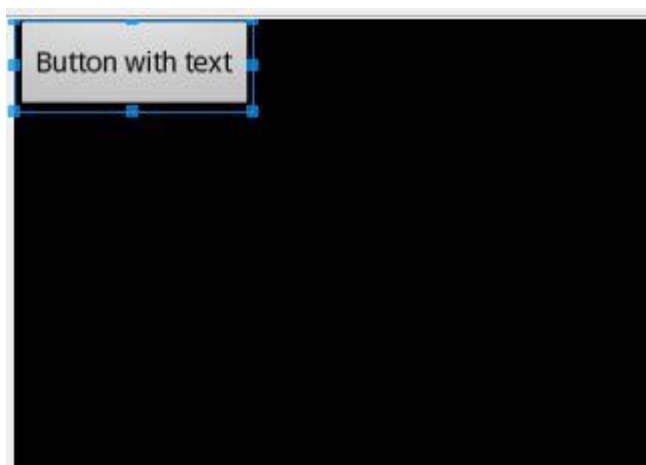
Create Activity: MainActivity

Открываем main.xml.

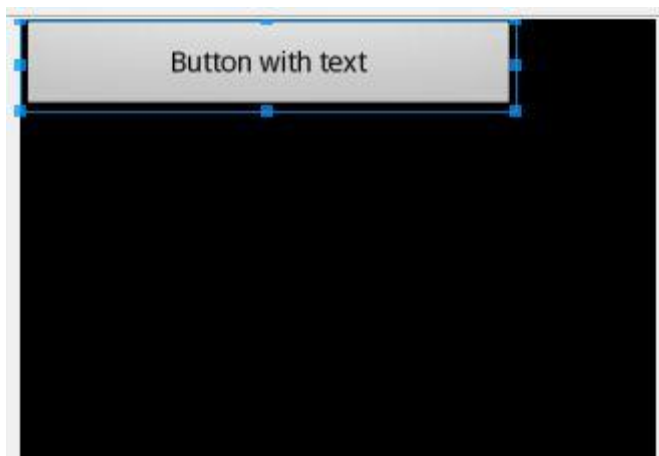
Настроим корневой LinearLayout на горизонтальную ориентацию, удалим TextView, и добавим Button с шириной и высотой равной wrap_content. Она отображается на экране, и ее ширина соответствует тексту на ней.



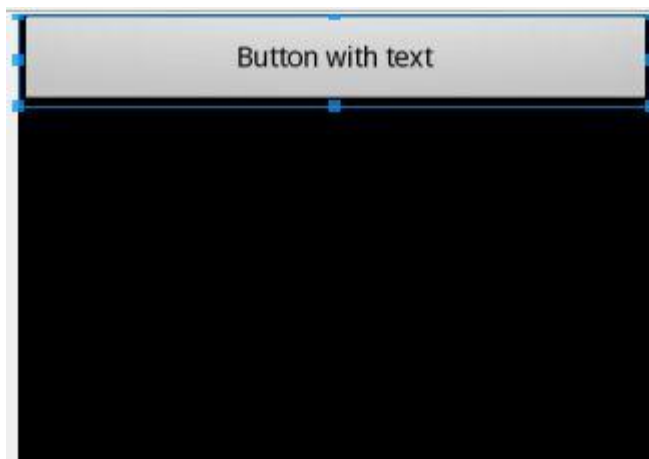
Изменим текст с «Button» на «Button with text», сохраним и посмотрим на экран.



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 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.

Изменим текст нашей кнопки на B1 и добавим ей соседа по `LinearLayout` – вторую кнопку с текстом B2. Ширину для обоих поставьте `wrap_content`

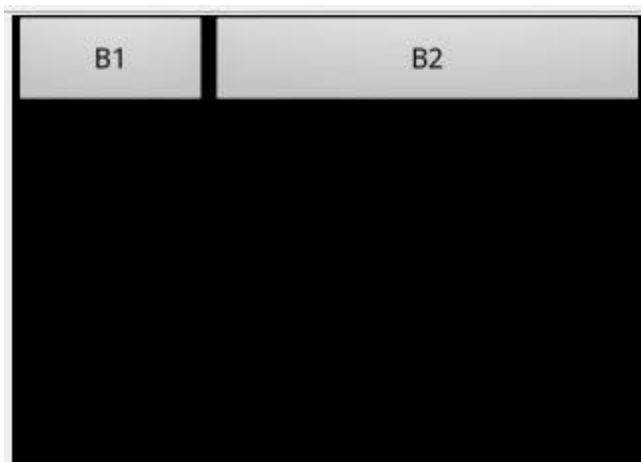


Займемся дележом. Если мы хотим, чтобы кнопки поделили пространство родителя поровну – то для обеих укажем `weight = 1`. В этом случае кнопки равны по ширине.

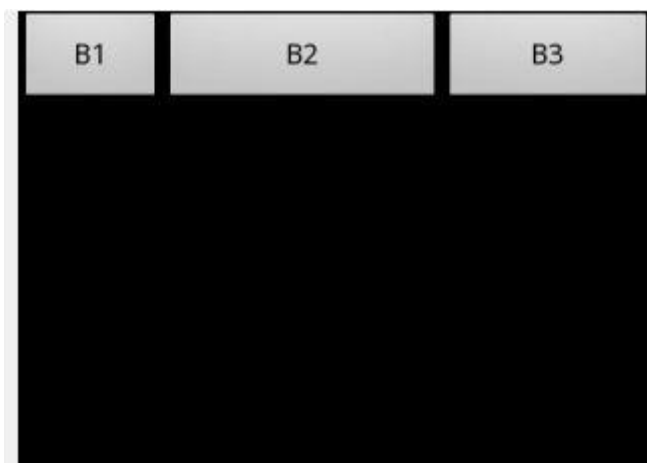


Обратите внимание, что не используются единицы измерения, указываются просто числа.

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:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="fill_parent"
5      android:layout_height="fill_parent"
6      android:orientation="horizontal">
7      <Button
8          android:id="@+id/button1"
9          android:layout_height="wrap_content"
10         android:layout_width="wrap_content"
11         android:text="B1"
12         android:layout_weight="1">
13     </Button>
14     <Button
15         android:id="@+id/button2"
16         android:layout_width="wrap_content"
17         android:layout_height="wrap_content"
18         android:text="B2"
19         android:layout_weight="3">
20     </Button>
21     <Button
22         android:layout_height="wrap_content"
23         android:layout_width="wrap_content"
24         android:id="@+id/button3"
25         android:text="B3"
26         android:layout_weight="2">
27     </Button>
28 </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 `weight` вы можете указать значение `height` или `width = 0dp`. В этом случае не будет учитываться содержимое элементов. Результат будет более соответствующим коэффициентам веса.

Layout gravity

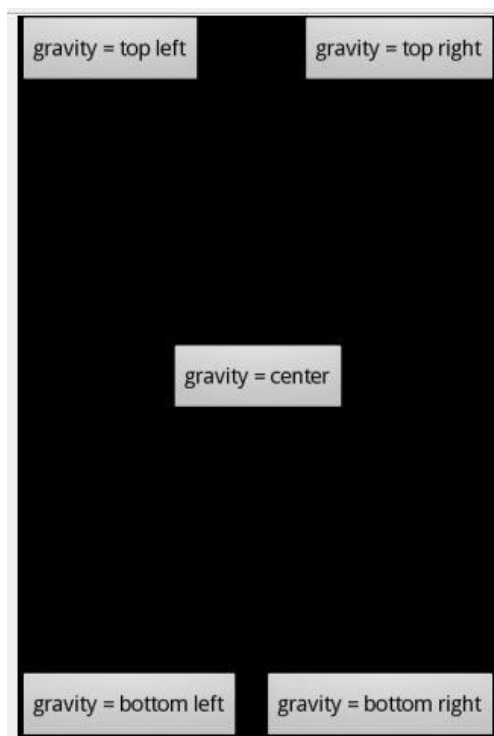
Параметр `layout_gravity` аналогичен выравниванию из Word или Excel. Удобнее всего продемонстрировать его с использованием `FrameLayout`.

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 *glayout.xml*:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:orientation="vertical"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent">
7      <FrameLayout
8          android:id="@+id/frameLayout1"
9          android:layout_width="match_parent"
10         android:layout_height="match_parent">
11         <Button
12             android:id="@+id/button1"
13             android:layout_width="wrap_content"
14             android:layout_height="wrap_content"
15             android:layout_gravity="top|left"
16             android:text="gravity = top left">
17         </Button>
18         <Button
19             android:id="@+id/button2"
20             android:layout_width="wrap_content"
21             android:layout_height="wrap_content"
22             android:layout_gravity="top|right"
23             android:text="gravity = top right">
24         </Button>
25         <Button
26             android:id="@+id/button3"
27             android:layout_width="wrap_content"
28             android:layout_height="wrap_content"
29             android:layout_gravity="bottom|left"
30             android:text="gravity = bottom left">
31         </Button>
32         <Button
33             android:id="@+id/button4"
34             android:layout_width="wrap_content"
35             android:layout_height="wrap_content"
36             android:layout_gravity="bottom|right"
37             android:text="gravity = bottom right">
38         </Button>
39         <Button
40             android:id="@+id/button5"
41             android:layout_width="wrap_content"
42             android:layout_height="wrap_content"
43             android:layout_gravity="center"
44             android:text="gravity = center">
45         </Button>
46     </FrameLayout>
47 </LinearLayout>
```

На экране видим:



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

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.

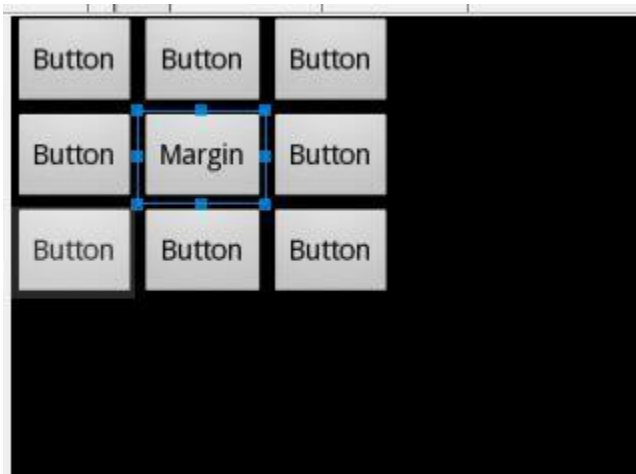
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:orientation="vertical"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent">
7      <TableLayout
8          android:layout_height="wrap_content"
9          android:layout_width="match_parent"
10         android:id="@+id/tableLayout1">
11         <TableRow
12             android:id="@+id/tableRow1"
13             android:layout_width="wrap_content"
14             android:layout_height="wrap_content">
15             <Button
16                 android:text="Button"
17                 android:id="@+id/button1"
18                 android:layout_width="wrap_content"
19                 android:layout_height="wrap_content">
20             </Button>
21             <Button
22                 android:text="Button"
23                 android:id="@+id/button2"
24                 android:layout_width="wrap_content"
25                 android:layout_height="wrap_content">
26             </Button>
27             <Button
28                 android:text="Button"
29                 android:id="@+id/button3"
30                 android:layout_width="wrap_content"
31                 android:layout_height="wrap_content">
32             </Button>
33         </TableRow>
34         <TableRow
35             android:id="@+id/tableRow2"
36             android:layout_width="wrap_content"
37             android:layout_height="wrap_content">
38             <Button
39                 android:text="Button"
40                 android:id="@+id/button4"
41                 android:layout_width="wrap_content"
42                 android:layout_height="wrap_content">
43             </Button>
44             <Button
45                 android:text="Margin"
46                 android:id="@+id/button5"
47                 android:layout_width="wrap_content"
48                 android:layout_height="wrap_content">
49             </Button>
50             <Button
51                 android:text="Button"
52                 android:id="@+id/button6"
53                 android:layout_width="wrap_content"
54                 android:layout_height="wrap_content">
55             </Button>
56         </TableRow>
57         <TableRow
58             android:id="@+id/tableRow3"
59             android:layout_width="wrap_content"
60             android:layout_height="wrap_content">
61             <Button
62                 android:text="Button"
63                 android:id="@+id/button7"
64                 android:layout_width="wrap_content"
65                 android:layout_height="wrap_content">
66             </Button>
```



```

67         <Button
68             android:text="Button"
69             android:id="@+id/button8"
70             android:layout_width="wrap_content"
71             android:layout_height="wrap_content">
72         </Button>
73         <Button
74             android:text="Button"
75             android:id="@+id/button9"
76             android:layout_width="wrap_content"
77             android:layout_height="wrap_content">
78         </Button>
79     </TableRow>
80 </TableLayout>
81 </LinearLayout>

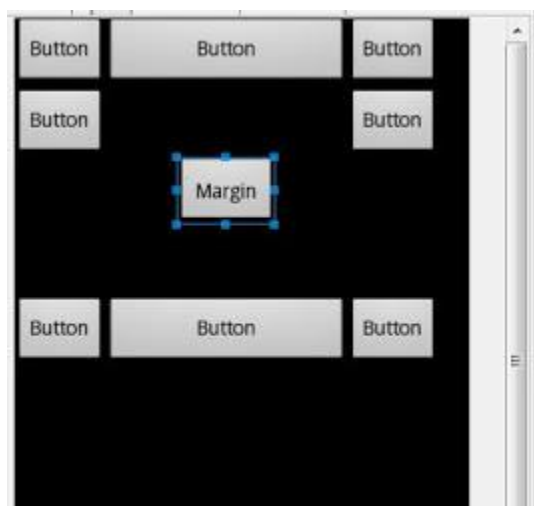
```



И на кнопке в центре будем экспериментировать.

margin = 50 dp

Вокруг кнопки со всех сторон образовался отступ = 50 dp.

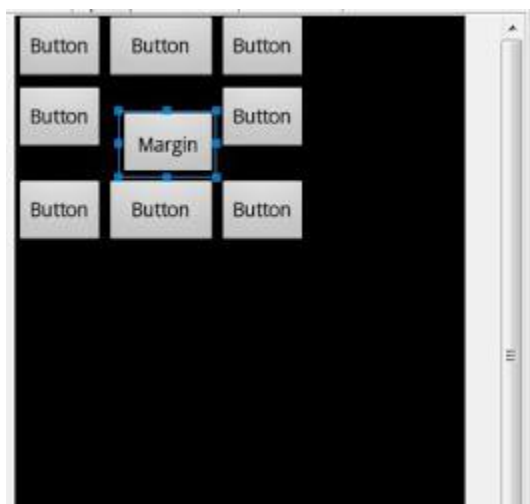


Text scale X	
Text select handle	
Text select handle left	
Text select handle right	
Text size	
Text style	
Typeface	
Visibility	
Width	
Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	50dp
Layout margin bottom	
Layout margin left	
Layout margin right	
Layout margin top	
Layout weight	
Layout width	wrap_content

margin left = 10 dp

margin top = 20 dp

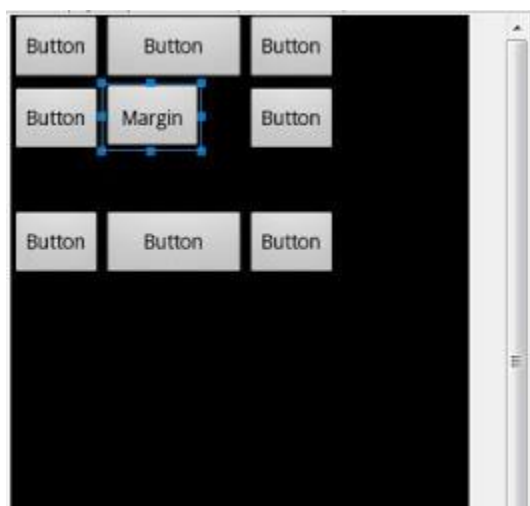
Отступ слева и сверху.



Text scale X	
Text select handle	
Text select handle left	
Text select handle right	
Text size	
Text style	
Typeface	
Visibility	
Width	
Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	
Layout margin bottom	
Layout margin left	10dp
Layout margin right	
Layout margin top	20dp
Layout weight	
Layout width	wrap_content

margin right = 30 dp
margin bottom = 40 dp

Отступ справа и снизу.



Text scale X	
Text select handle	
Text select handle left	
Text select handle right	
Text size	
Text style	
Typeface	
Visibility	
Width	
Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	
Layout margin bottom	40dp
Layout margin left	
Layout margin right	30dp
Layout margin top	
Layout weight	
Layout width	wrap_content

Стили

Если кто использовал HTML, то наверняка слышал про каскадные стили – CSS. Стили позволяют группировать атрибуты элементов (кнопок, таблиц, параграфов и т.д.). Далее вы просто применяете к элементам стили и элемент рисуется с учетом всех атрибутов стиля. Необходимость повторять несколько раз один и тот же код для элементов, которые должны выглядеть одинаково, пропадает. 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.