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 пикселей.

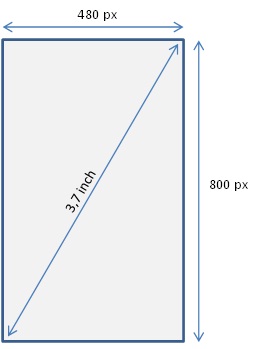


Рис. 1.

Кол-во пикселей 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 + 8002

dpi2 = 870400 / 13,69 = 63579

dpi = 252.

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

Рассмотрим подробно следующие параметры View элементов

Layout width и Layout height

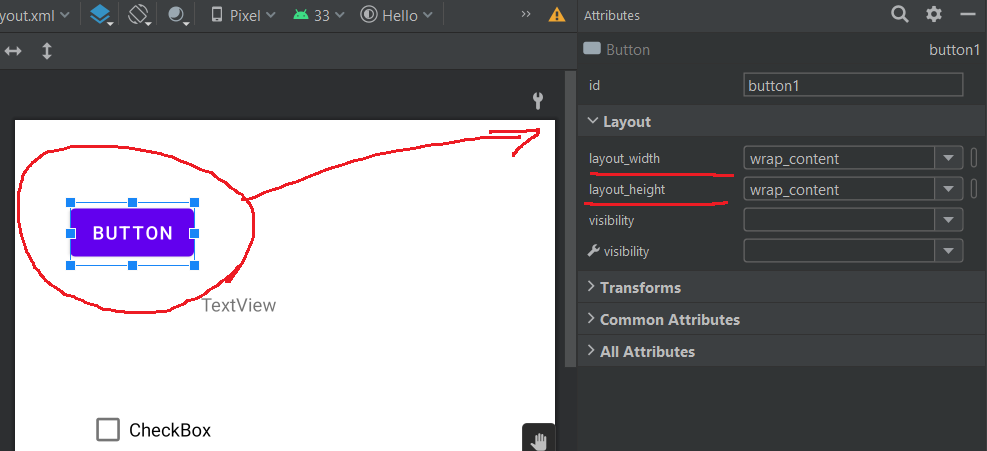


Рис. 2.

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

Абсолютные значения:

Используются следующие [единицы измерения](http://developer.android.com/guide/topics/resources/more-resources.html#Dimension) (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](http://en.wikipedia.org/wiki/Point_(typography)).

**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:

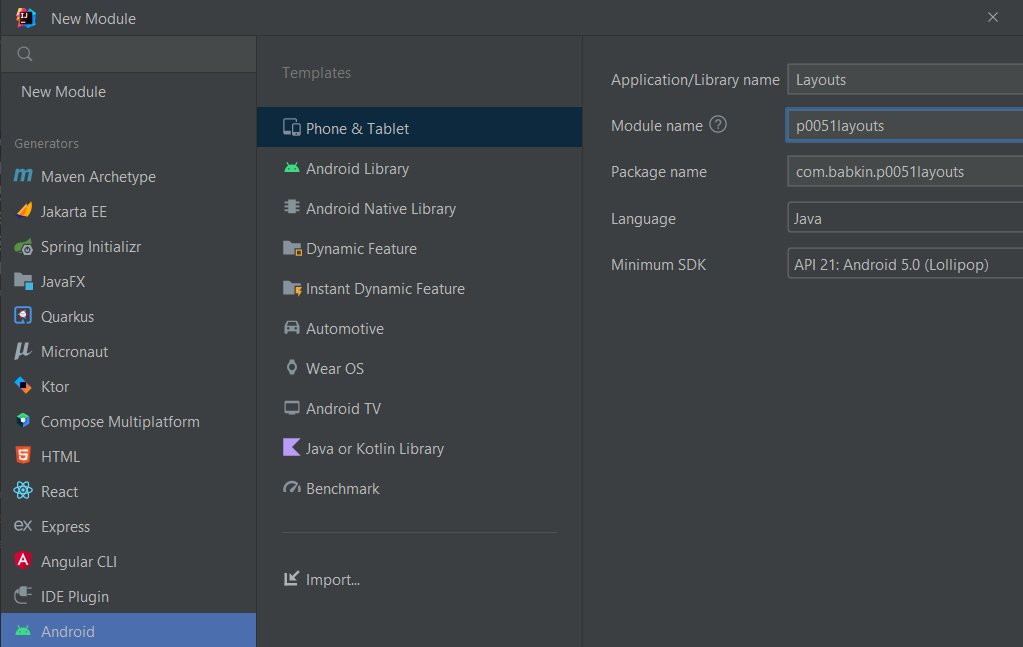


Рис. 3.

Откроем layout­файл main.xml, и поместите в него следующий код:

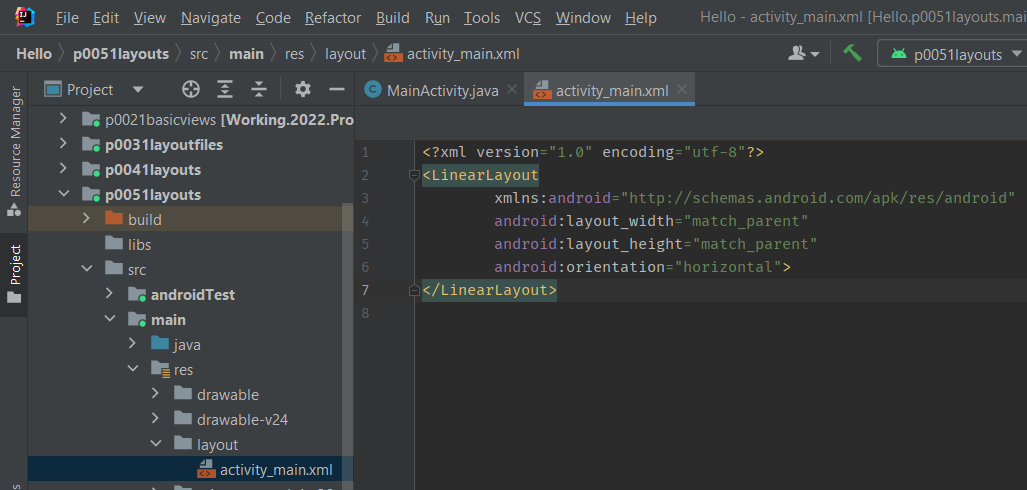


Рис. 4.

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

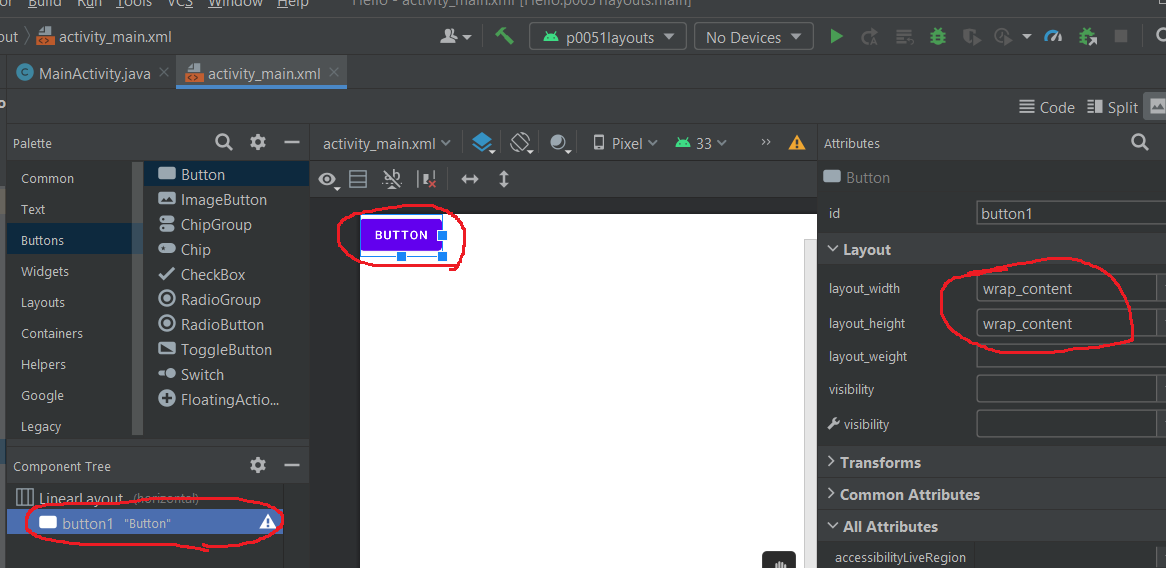


Рис. 5.

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.

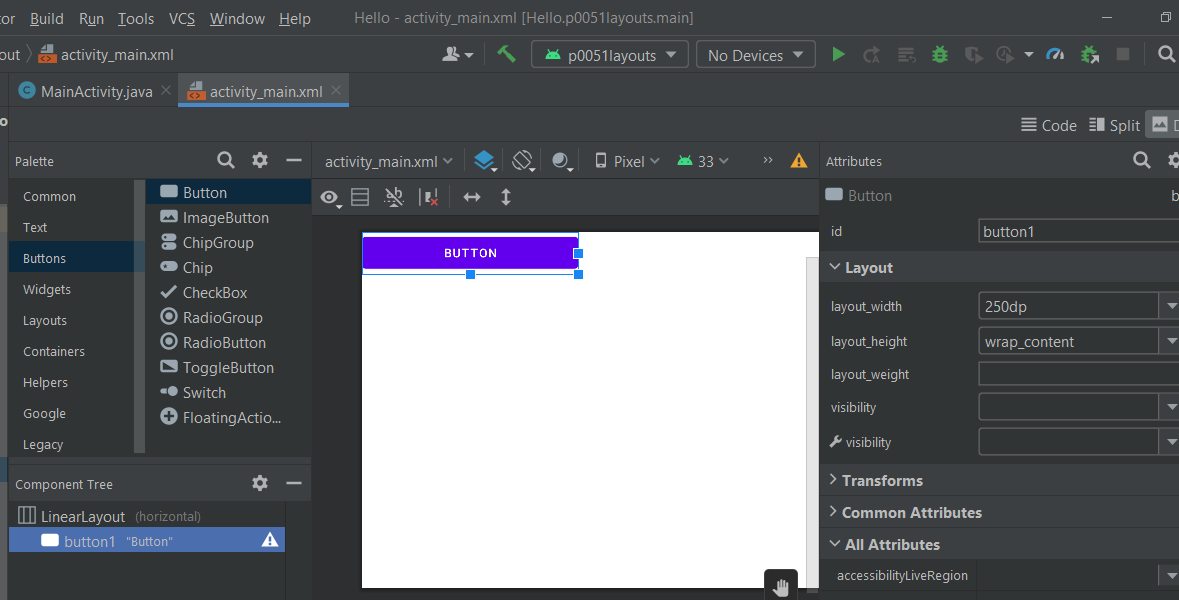


Рис. 6.

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

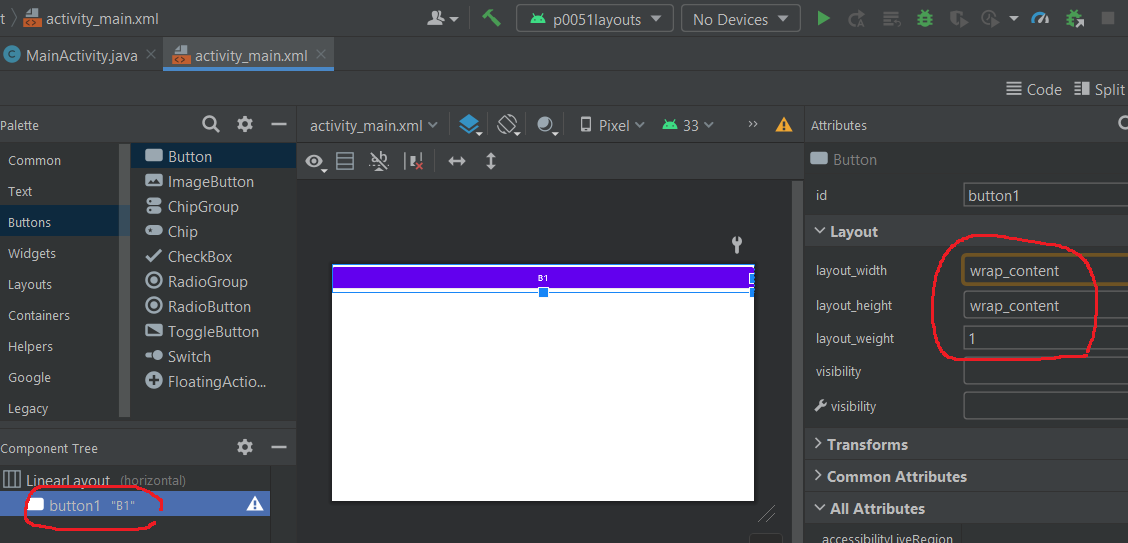


Рис. 7.

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 weight­значениям. 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.

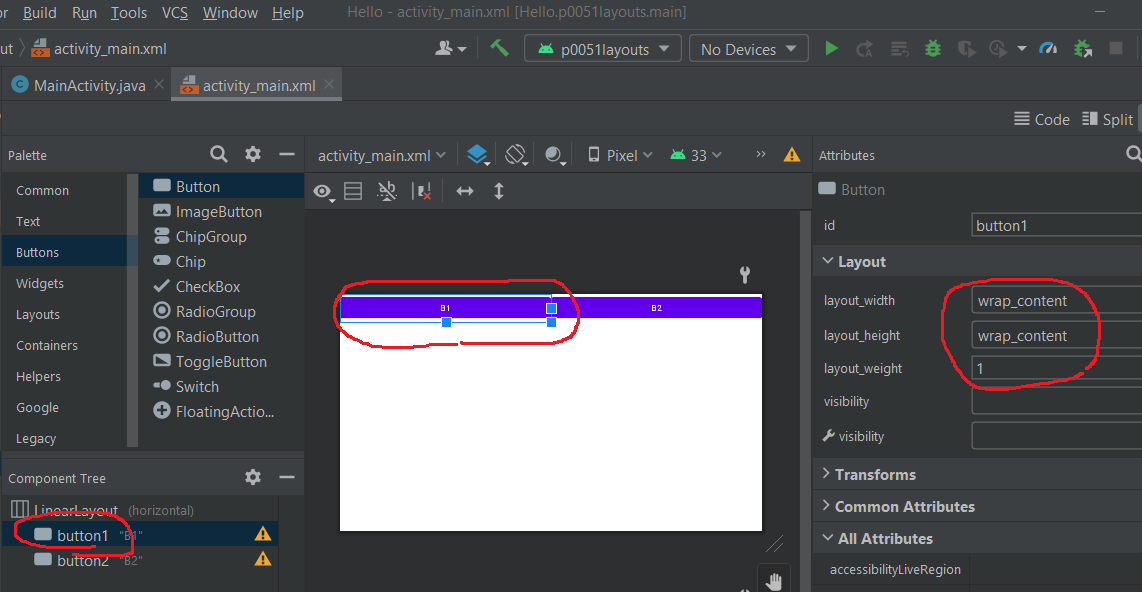


Рис. 8.

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

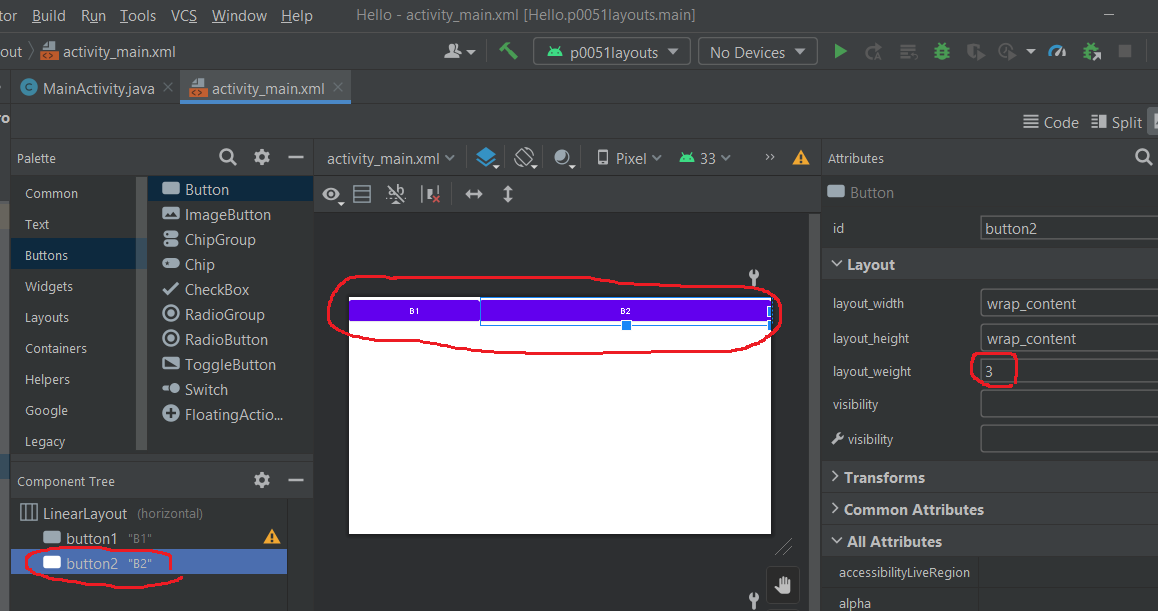


Рис. 9.

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

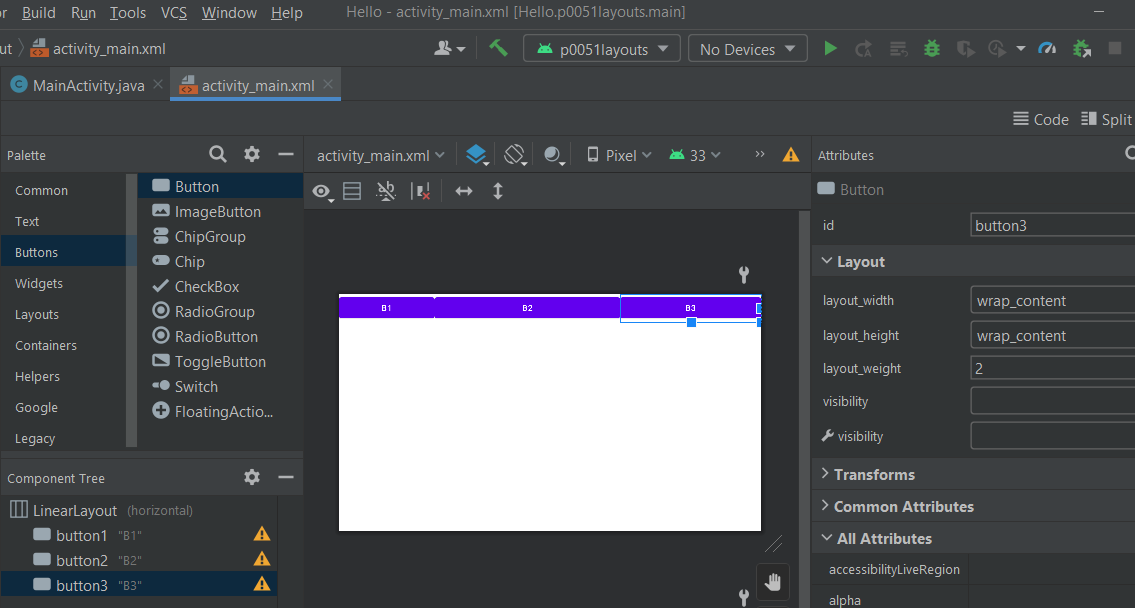


Рис. 10.

xml­код получившегося экрана:

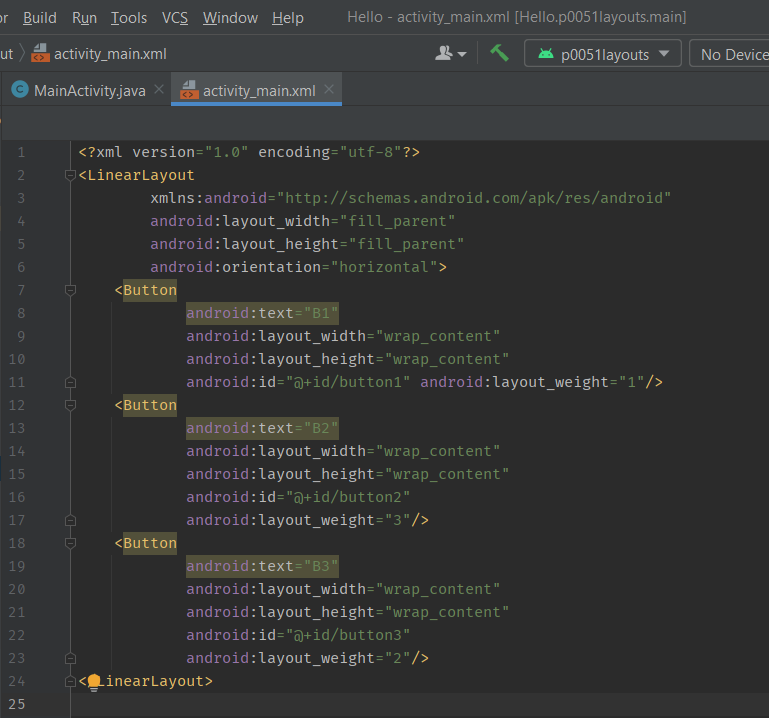


Рис. 11.

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 и результат будет более соответствующий коэффициентам веса.

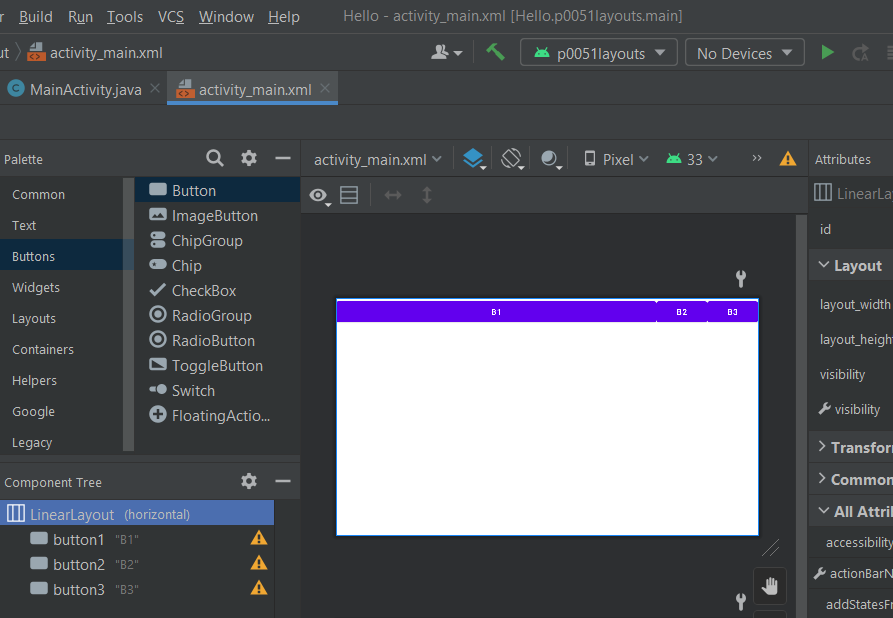


Рис. 12.

Layout gravity

Параметр layout\_gravity аналогичен выравниванию из Word или Excel. Удобнее всего продемонстрировать его с использованием FrameLayout. Я не описывал этот Layout в прошлый раз, 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.

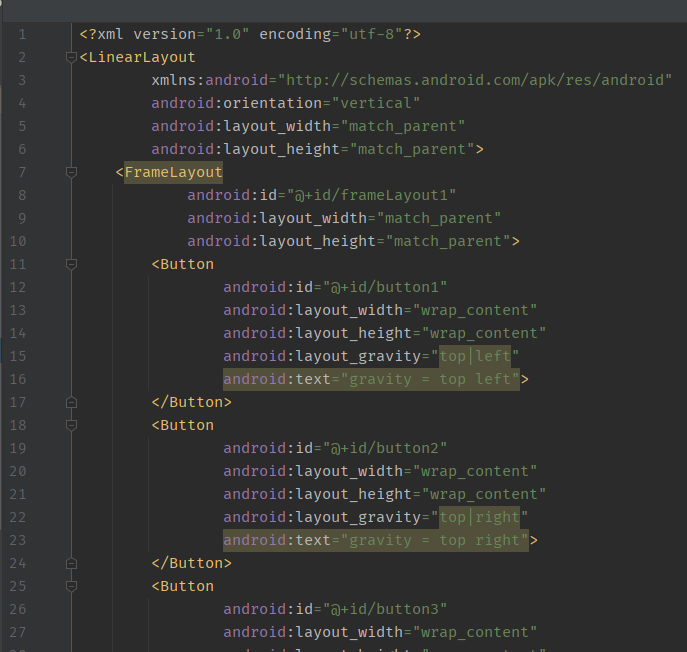


Рис. 13.



Рис. 14.

На экране видим:

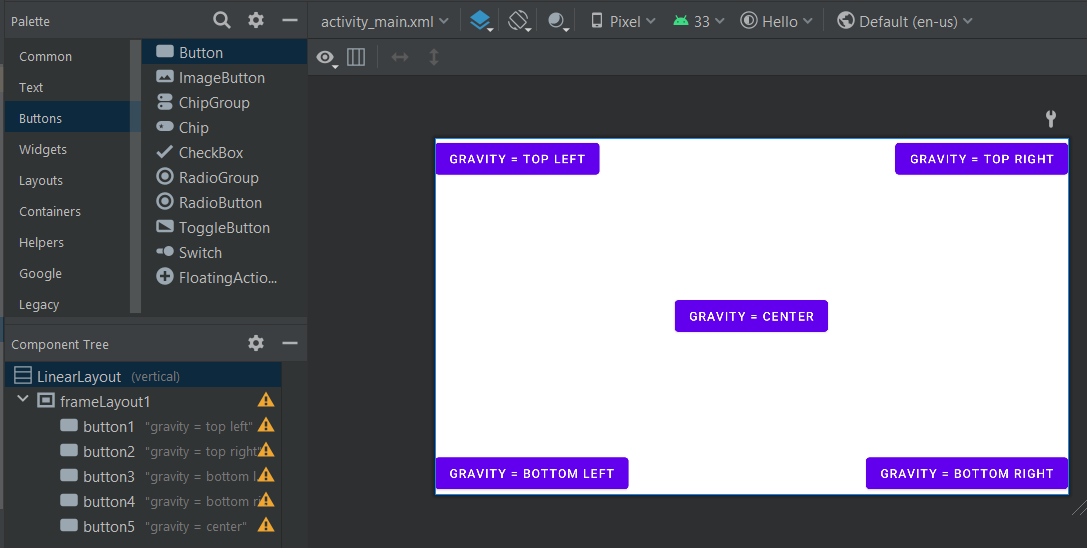


Рис. 15.

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.

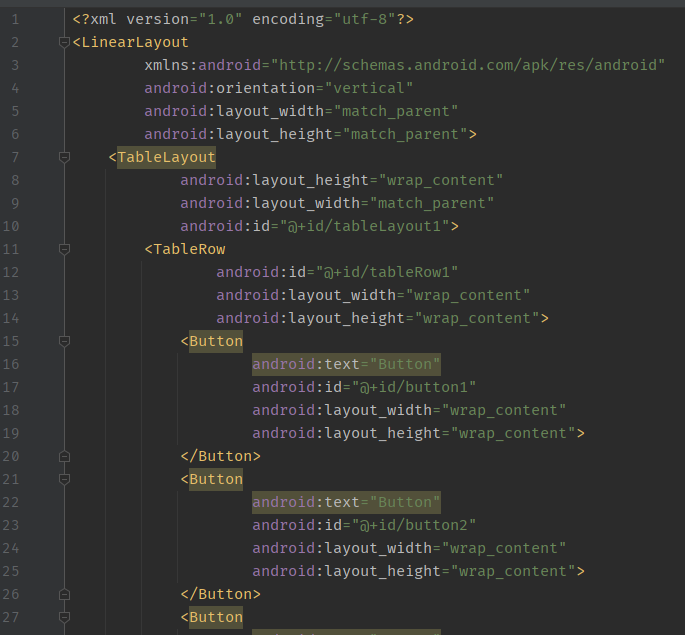


Рис. 16.



Рис. 17.

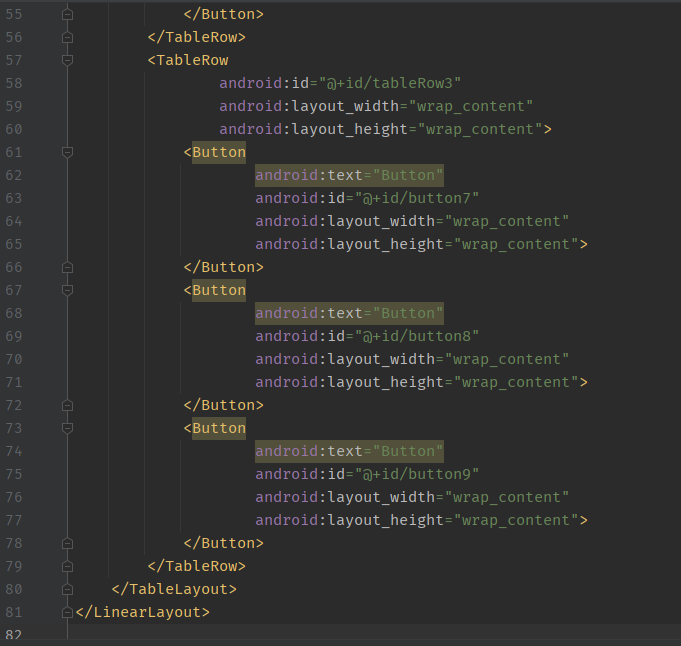


Рис. 18.

И на кнопке в центре будем экспериментировать. margin = 50 dp Вокруг кнопки со всех сторон образовался отступ = 50 dp.

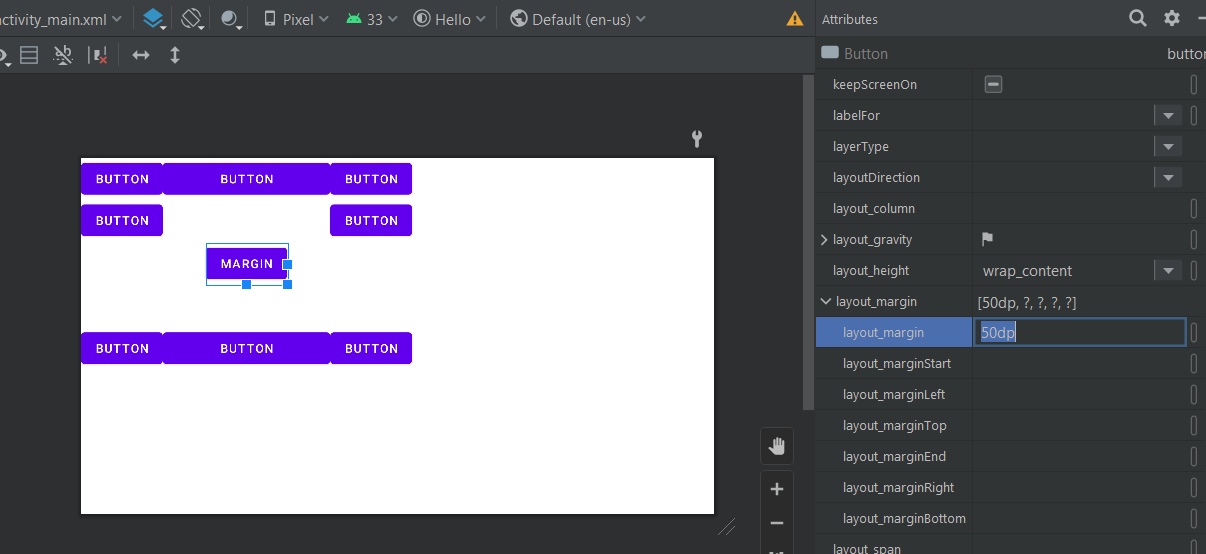


Рис. 19.

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

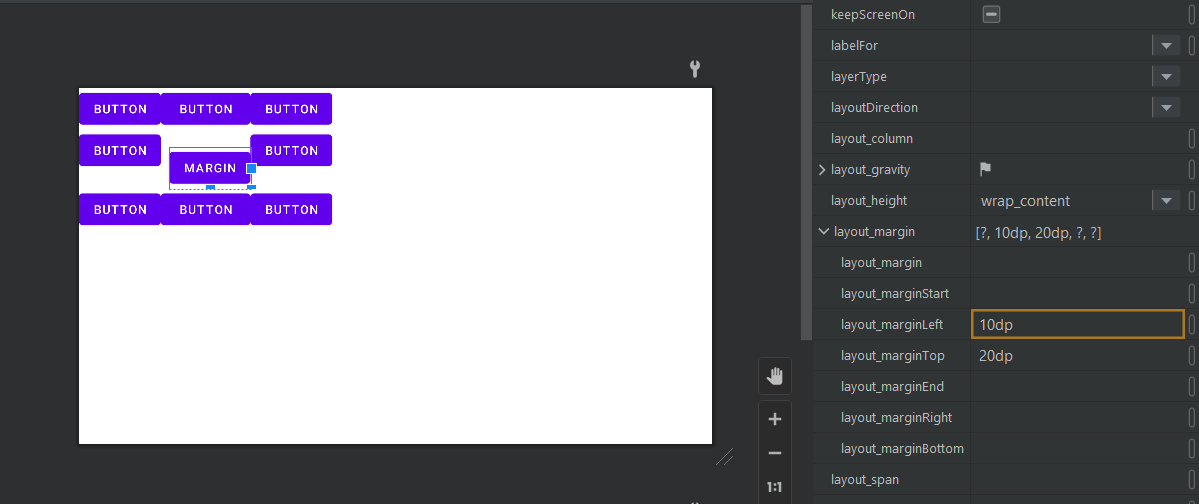


Рис.20.

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

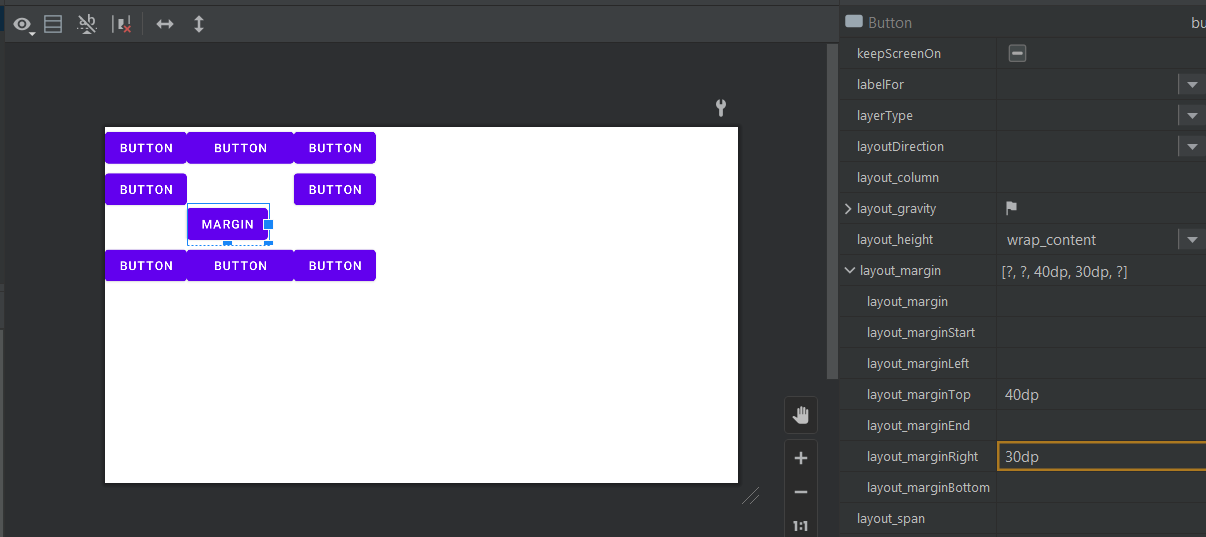


Рис. 21.