GridLayout

[Цифровая клавиатура](http://developer.alexanderklimov.ru/android/layout/gridlayout.php#digitalkey)

В Android 4.0 появился новый вид макета под именем **GridLayout**. На первый взгляд он может показаться похожим на **TableLayout**. Но на самом деле он гораздо удобнее и функциональнее. И очень рекомендуется изучить и использовать его в своих новых проектах, которые разрабатываются под новую платформу.

Разметка относится к классу **android.widget.GridLayout** и имеет колонки, ряды, клетки как в **TableLayout**, но при этом элементы могут гибко настраиваться.

В **GridLayout** для любого компонента можно указать строку и колонку, и в этом месте таблицы он будет размещён. Указывать элемент для каждой ячейки не понадобится, это нужно делать только для тех ячеек, где действительно должны быть компоненты. Компоненты могут растягиваться на несколько ячеек таблицы. Более того, в одну ячейку можно поместить несколько компонентов.

В данной разметке нельзя использовать атрибут веса, поскольку он не сработает в дочерних представлениях **GridLayout**. Используйте атрибут **layout\_gravity**.

Обратите внимание на атрибуты **layout\_column** и **layout\_columnSpan**, используемые для указания самой левой колонки и количества занимаемых компонентом колонок. Также доступны атрибуты **layout\_row** и **layout\_rowSpan**.

Для дочерних элементов не нужно указывать атрибуты **layout\_height** и **layout\_width**, по умолчанию они устанавливаются в значение **wrap\_content**.

Количество колонок и рядов используются атрибуты **android:columnCount="number"** и **android:rowCount="number"**.

Цифровая клавиатура

Для демонстрации создадим пример цифровой клавиатуры или калькулятора, чтобы понять преимущества нового шаблона.

Итак, мы хотим создать разметку, напоминающую цифровую панель на многих моделях клавиатур для настольного компьютера или обычный калькулятор. Если вы посмотрите на цифровую панель, то увидите, что часть клавиш имеет больший размер и вытянуты в длину или ширину. Подобный дизайн практически невозможно реализовать старыми способами - придётся постоянно использовать вложенные конструкции **LinearLayout**, **TableLayout** и др.

Но, теперь у нас есть **GridLayout**. Во многом его поведение схоже с **LinearLayout** - у него есть горизонтальная и вертикальная ориентации, которые определяют вывод следующего элемента.

Для цифровой клавиатуры, если начнем с клавиши слэша (/) с позиции 4 колонки, то используя горизонтальную ориентацию, нам не нужно пропускать клетки. Выбирая горизонтальную ориентацию, мы ограничиваем число колонок для автоматического переноса новой клетки на следующий ряд. В нашем примере используется четыре колонки. В каждой клетке макета будет находиться кнопка, отцентрированая относительно клетки. Сама разметка должна занимать весь контент экрана.

Дочерние элементы настраиваются не совсем привычном образом. Нам не нужно явно задавать размеры (ширину и высоту) каждого элемента. Согласно нашему плану мы должны использовать четыре колонки для 16 кнопок. Попробуем...

Вот как будет выглядеть первоначальная XML-разметка для нашей цели:

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

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

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:columnCount="4"

android:orientation="horizontal" >

<Button android:text="1" />

<Button android:text="2" />

<Button android:text="3" />

<Button android:text="4" />

<Button android:text="5" />

<Button android:text="6" />

<Button android:text="7" />

<Button android:text="8" />

<Button android:text="9" />

<Button android:text="10" />

<Button android:text="11" />

<Button android:text="12" />

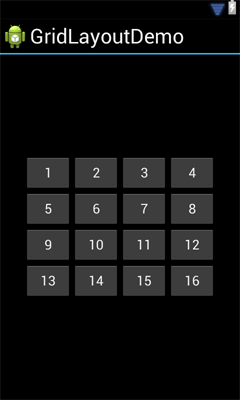
<Button android:text="13" />

<Button android:text="14" />

<Button android:text="15" />

<Button android:text="16" />

А так будет выглядеть форма в графическом дизайнере:



Поиграйтесь с параметром **android:columnCount**, чтобы увидеть, как ведёт себя шаблон при разных значениях.

Теперь займёмся более точной настройкой. Например, первая клавиша клавиатуры - символ слеша (/) будет иметь такой же размер, как большинство кнопок, но начинаться с четвёртой колонки. Клавиша (+) идёт после клавиши (9) и занимает три воображаемых ряда. Клавиша (0) должна занять две колонки. Клавиша (=) должна занять три колонки. Внесём требуемые изменения:

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

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

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:columnCount="4"

android:orientation="horizontal" >

<Button

android:layout\_column="3"

android:text="/" />

<Button android:text="1" />

<Button android:text="2" />

<Button android:text="3" />

<Button android:text="\*" />

<Button android:text="4" />

<Button android:text="5" />

<Button android:text="6" />

<Button android:text="-" />

<Button android:text="7" />

<Button android:text="8" />

<Button android:text="9" />

<Button

android:layout\_rowSpan="3"

android:text="+" />

<Button

android:layout\_columnSpan="2"

android:text="0" />

<Button android:text="00" />

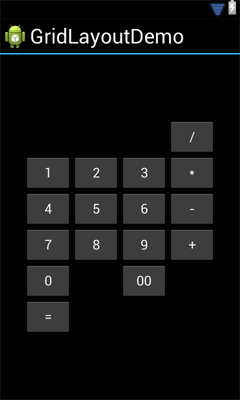
<Button

android:layout\_columnSpan="3"

android:text="=" />

</GridLayout>

Теперь мы видим такую картину:



Это не совсем то, что нам нужно. Вроде клавиши зарезервировали себе место, но не растянулись до нужных размеров. Исправляем ситуацию. Здесь на помощь нам придёт атрибут **layout\_gravity**. Применим его у клавиш, которые необходимо растянуть:

<Button

android:layout\_gravity="fill"

android:layout\_rowSpan="3"

android:text="+" />

<Button

android:layout\_gravity="fill"

android:layout\_columnSpan="2"

android:text="0" />

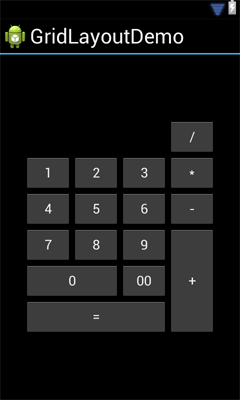
<Button

android:layout\_gravity="fill"

android:layout\_columnSpan="3"

android:text="=" />

Финальный результат радует глаз:



Как видите, если разобраться, то ничего сложного. Разметка получилась лаконичной и удобной для чтения.

Использованные материалы: