Определение размеров

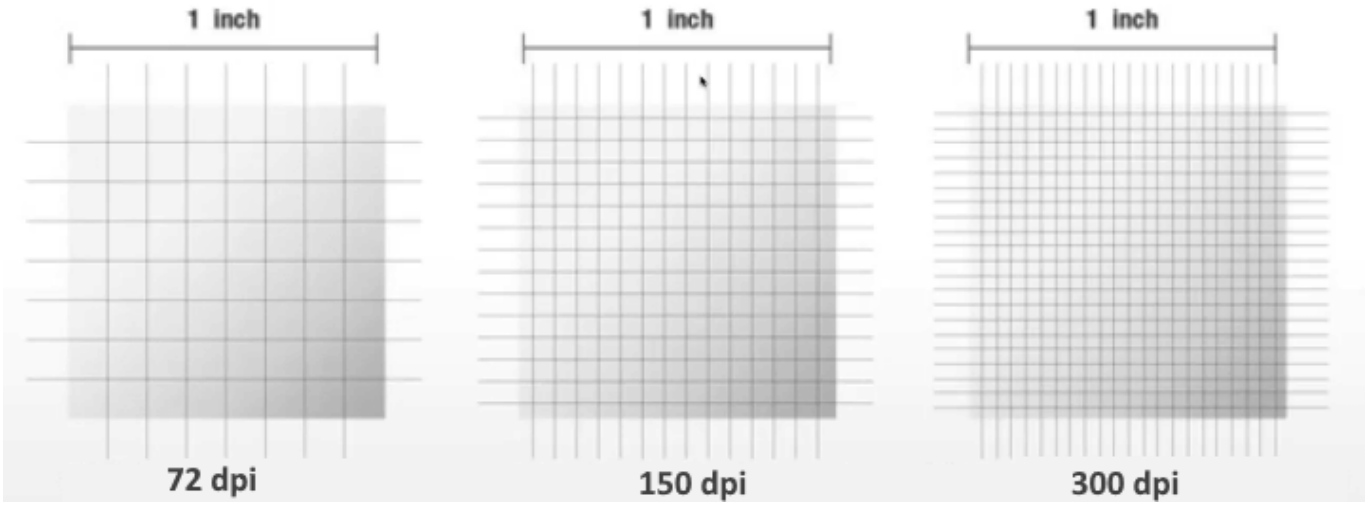
Последнее обновление:

При разработке приложений под Android мы можем использовать различные типы измерений:

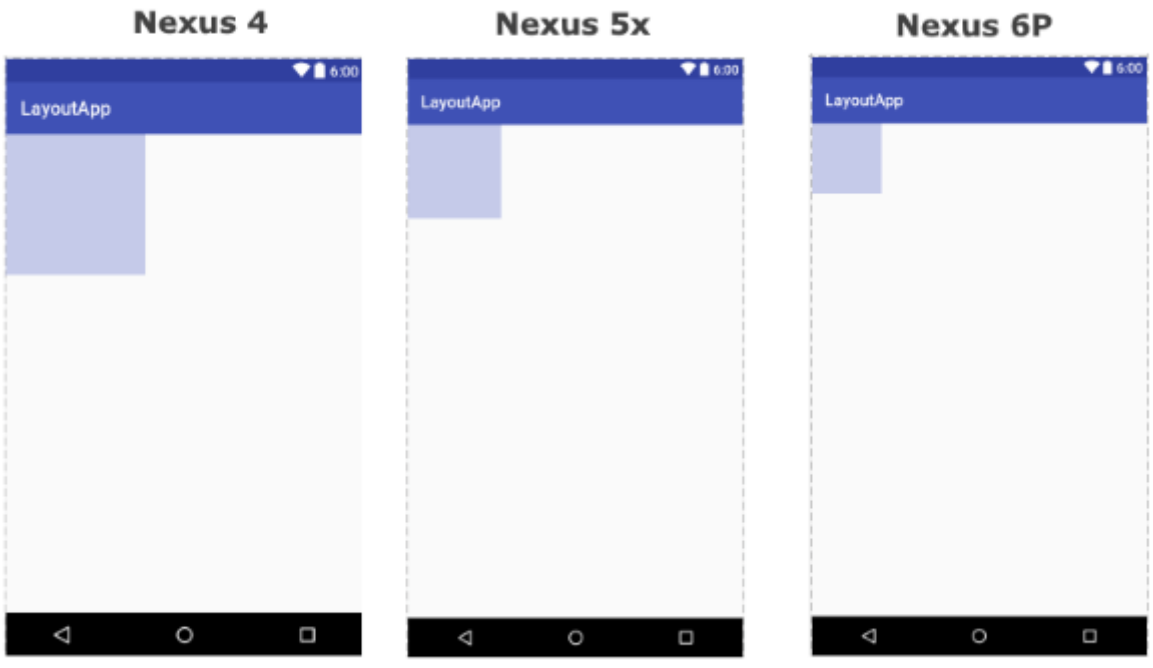
- px: пиксели текущего экрана. Однако эта единица измерения не рекомендуется, так как реальное представление внешнего вида может изменяться в зависимости от устройства; каждое устройство имеет определенный набор пикселей на дюйм, поэтому количество пикселей на экране может также меняться
- dp: (device-independent pixels) независимые от плотности экрана пиксели. Абстрактная единица измерения, основанная на физической плотности экрана с разрешением 160 dpi (точек на дюйм). В этом случае 1dp = 1px. Если размер экрана больше или меньше, чем 160dpi, количество пикселей, которые применяются для отрисовки 1dp соответственно увеличивается или уменьшается. Например, на экране с 240 dpi 1dp=1,5px, а на экране с 320dpi 1dp=2px. Общая формула для получения количества физических пикселей из dp: **px = dp * (dpi / 160)**
- sp: (scale-independent pixels) независимые от масштабирования пиксели. Допускают настройку размеров, производимую пользователем. Рекомендуются для работы со шрифтами.
- pt: 1/72 дюйма, базируются на физических размерах экрана

- `mm`: миллиметры
- `in`: дюймы

Предпочтительными единицами для использования являются dp. Это связано с тем, что мир мобильных устройств на Android сильно фрагментирован в плане разрешения и размеров экрана. И чем больше плотность пикселей на дюйм, тем соответственно больше пикселей нам будет доступно:

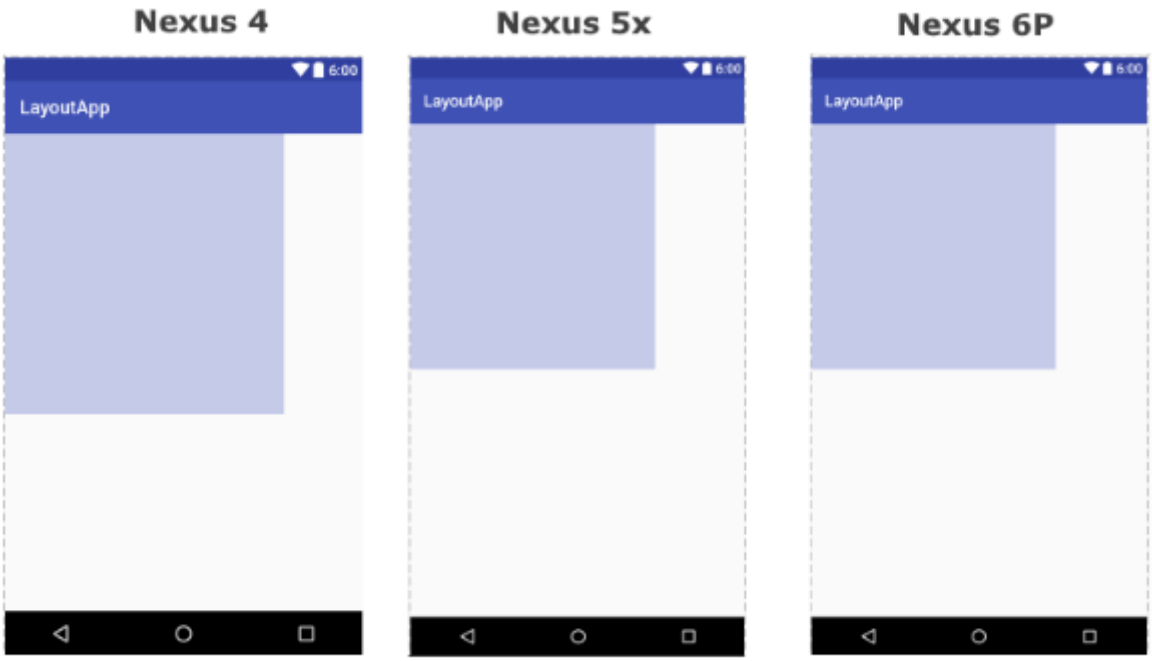


Используя же стандартные физические пиксели мы можем столкнуться с проблемой, что размеры элементов также будут сильно варьироваться в зависимости от плотности пикселей устройства. Например, возьмем 3 устройства с различными характеристиками экрана Nexus 4, Nexus 5X и Nexus 6P и выведем на экран квадрат размером 300px на 300px:



В одном случае квадрат по ширине будет занимать 40%, в другом - треть ширины, в третьем - 20%.

Теперь также возьмем квадрат со сторонами 300x300, но теперь вместо физических пикселей используем единицы dp:



Теперь же размеры квадрата на разных устройствах выглядят более консистентно.

Для упрощения работы с размерами все размеры разбиты на несколько групп:

- **ldpi (low)**: ~120dpi
- **mdpi (medium)**: ~160dpi
- **hdpi (high)**: ~240dpi (к данной группе можно отнести такое древнее устройство как Nexus One)
- **xhdpi (extra-high)**: ~320dpi (Nexus 4)
- **xxhdpi (extra-extra-high)**: ~480dpi (Nexus 5/5X, Samsung Galaxy S5)
- **xxxhdpi (extra-extra-extra-high)**: ~640dpi (Nexus 6/6P, Samsung Galaxy S6)

Установка размеров

Основная проблема, связанная с размерами, связана с их установкой в коде Java. Например, некоторые методы принимают в качестве значения физические пиксели, а не device-independent pixels. В этом случае может потребоваться перевести значения из

одного типа единиц в другой. Для этого требуется применить метод **TypedValue.applyDimension()**, который принимает три параметра:

```
public static float applyDimension(int unit,
                                   float value,
                                   android.util.DisplayMetrics metrics)
```

Параметр `unit` представляет тип единиц, из которой надо получить значение в пикселях. Тип единиц описывается одной из констант **TypedValue**:

- `COMPLEX_UNIT_DIP` - dp или независимые от плотности экрана пиксели
- `COMPLEX_UNIT_IN` - in или дюймы
- `COMPLEX_UNIT_MM` - mm или миллиметры
- `COMPLEX_UNIT_PT` - pt или точки
- `COMPLEX_UNIT_PX` - px или физические пиксели
- `COMPLEX_UNIT_SP` - sp или независимые от масштабирования пиксели (scale-independent pixels)

Параметр `value` представляет значение, которое надо преобразовать

.

Параметр `metrics` представляет информацию о метрике, в рамках коорой надо выполнить преобразование.

В итоге метод возвращает преобразованное значение. Рассмотрим абстрактный пример. Например, нам надо получить из 60dp обычные физические пиксели:

```
int valueInDp = 60;
int valueInPx = (int) TypedValue.applyDimension(
                TypedValue.COMPLEX_UNIT_DIP, valueInDp, getResources().getDi
```

В качестве третьего аргумента передается вызов метода `getResources().getDisplayMetrics()`, который позволяет получить информацию о метрике, связанной с текущим устройством. В итоге мы получим из 60dp некоторое количество пикселей.