**ОСНОВИ ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ ПРОГРАМИ**

**5.1 Компоненти екрану**

Графічний інтерфейс користувача є ієрархією об’єктів android.view.View і android.view.ViewGroup. Кожен об’єкт ViewGroup представляє собоє контейнер, який містить і впорядковує дочірні об’єкти View. Зокрема, до контейнерів відносять такі елементи, як RelativeLayout, LinearLayout, GridLayout, ConstraintLayout і ряд інших.

Прості об’єкти View є елементами управління та інші віджети, наприклад, кнопки, текстові поля і т. д., через які користувач взаємодіє з програмою.

Більшість візуальних елементів успадковуються від класу View, такі як кнопки, текстові поля та інші, і розташовуються в пакеті android.widget.

Розмітка визначає візуальну структуру користувацького інтерфейсу. Встановити розмітку можна декількома способами:

1. Створити елементи управління програмно в коді Java;

2. Оголосити елементи інтерфейсу в XML

3. Поєднання обох способів – базові елементи розмітки визначити в XML, а решта додавати під час виконання додатку програмним способом.

Найкращим є підхід, за якого візуальний інтерфейс описується в файлах xml, а вся пов’язана з ним логіка – в класі activity. Таким чином ми досягаємо розмежування інтерфейсу і логіки додатка, їх легше розробляти і модифікувати.

**5.2 Визначення інтерфейсу у файлі xml. Файли layout**

У додатках під Android візуальний інтерфейс часто завантажується із спеціальних файлів xml, які зберігають розмітку. Ці файли є ресурсами розмітки. Подібний підхід нагадує створення веб-сайтів, коли інтерфейс описується в файлах html, а логіка програми – в коді javascript.

Файли розмітки графічного інтерфейсу розташовуються в проекті в каталозі

*res/layout*.

При створенні розмітки в XML слід дотримуватися деяких правил: кожен файл розмітки повинен містити один кореневий елемент, який повинен представляти об’єкт View або ViewGroup.

При компіляції кожен XML-файл розмітки компілюється в ресурс View. Завантаження ресурсу розмітки здійснюється в методі *Activity.onCreate()*. Щоб встановити розмітку для поточного об’єкта activity, треба в метод *setContentView*() як параметр передати посилання на ресурс розмітки.

Для отримання посилання на ресурс в коді Java необхідно застосувати вираз R.layout.[Назва\_ресурсу]. Назва ресурсу layout повинна збігатися з ім’ям файлу. Наприклад:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

Зазвичай в проекті є декілька різних ресурсів layout (рис. 5.1). Як правило, кожен

окремий клас Activity використовує свій файл layout. Але для одного класу Activity можна також використовуватися декілька різних файлів layout.

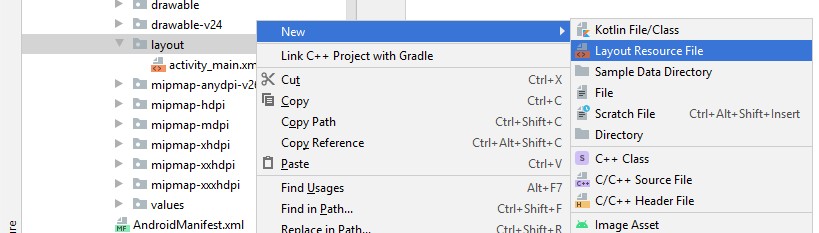


Рисунок 5.1 Додавання нового файла layout в проект

Для доступу до елементів по атрибуту id клас Activity має метод *findViewById*(). У цей метод передається ідентифікатор ресурсу у вигляді *R.id.[ідентифікатор\_елементу]*. Цей метод повертає об’єкт View – об’єкт базового класу для всіх елементів, тому результат методу ще необхідно привести до відповідного типу. Далі ми можемо маніпулювати цим елементом. Наприклад, для текстового поля

<TextView android:id="@+id/header" android:text="Second Activity" android:textSize="26dp" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" />

можна змінити текст:

public class MainActivity extends AppCompatActivity {

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// встановлюємо в якості інтерфейсу файл second\_layout.xml setContentView(R.layout.second\_layout);

// отримуємо елемент textView

TextView textView = (TextView) findViewById(R.id.header);

// змінюємо текст

textView.setText("Hello Android!");

}

}

Тут важливо, що пошук елемента відбувається після того, як в методі

*setContentView*() була встановлена розмітка, в якій цей візуальний елемент був визначений.

**5.3 Графічні можливості Android Studio**

Android Studio має потужний інструментарій, який полегшує розробку графічного інтерфейсу (рис. 5.2).

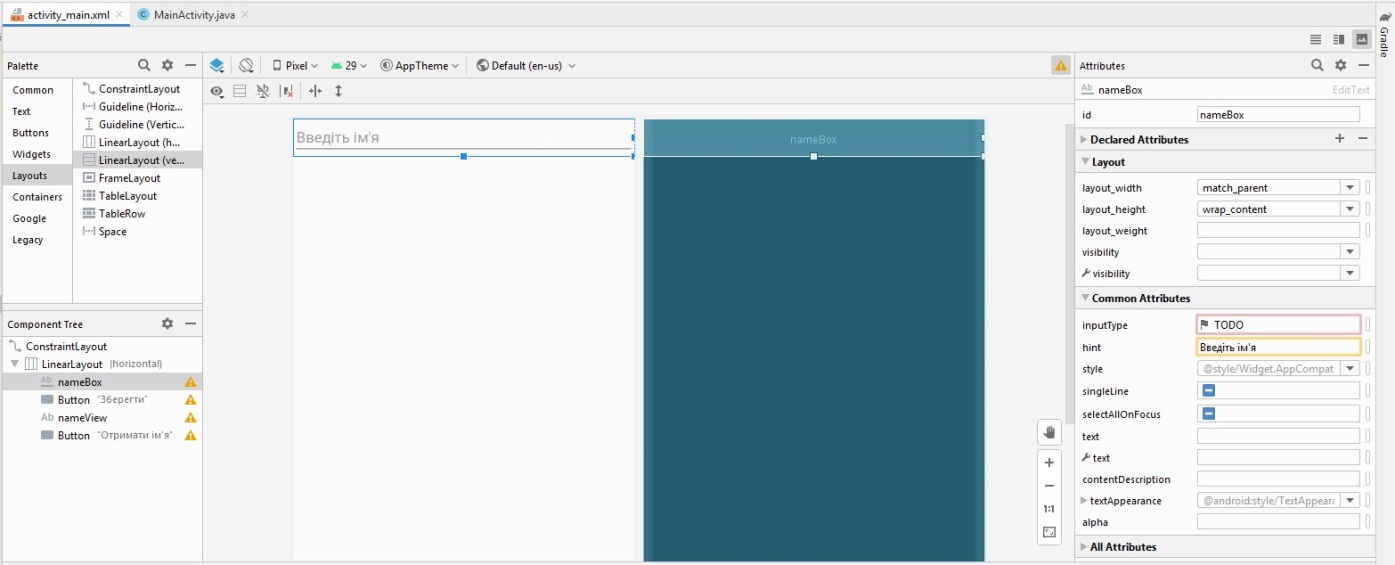


Рисунок 5.2 Частина Android Studio, призначена для розробки інтерфейсу програми

Ми можемо відкрити файл *activity\_main.xml* і внизу за допомогою кнопки Design переключитися в режим дизайнера із графічним представленням інтерфейсу у вигляді ескізу смартфона.

Зліва буде знаходитися панель інструментів, із якої ми можемо переносити потрібний елемент мишкою на ескіз смартфона. При цьому всі перенесені елементи будуть автоматично додаватися в файл *activity\_main.xml*. За допомогою миші ми можемо змінювати позиціонування вже доданих елементів.

Справа буде вікно Properties – панель властивостей виділеного елемента. Тут ми можемо змінити значення властивостей елемента. І знову ж таки після зміни властивостей зміниться і вміст файлу *activity\_main.xml*.

Тобто при будь-яких змінах в режимі дизайнера відбуватиметься синхронізація з файлом *activity\_main.xml*. Це все одно, що ми вручну змінювали б код безпосередньо в файлі *activity\_main.xml*.

**5.4 Різні варіанти компонування елементів інтерфейсу (Layout**)

**5.4.1 LinearLayout**

Контейнер LinearLayout представляє об’єкт ViewGroup, який впорядковує всі дочірні елементи в одному напрямку: по горизонталі або по вертикалі. Все елементи розташовуються один за одним. Напрямок розмітки вказується за допомогою атрибута *android:orientation*.

Якщо, наприклад, орієнтація розмітки вертикальна (*android:orientation = "vertical"*), то всі елементи розташовуються в стовпчик – по одному елементу на кожен рядок. Якщо орієнтація горизонтальна (*android:orientation="horizontal"*), то елементи розташовуються в один рядок:

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

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/android)" android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="horizontal" >

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Hello" android:textSize="26sp" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Android" android:textSize="26sp" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Nougat" android:textSize="26sp" />

</LinearLayout>

LinearLayout підтримує таку властивість, як вага елемента, яка задається атрибутом *android:layout\_weight*. Це властивість вказує, яку частину залишку вільного місця контейнера по відношенню до інших об’єктів займе даний елемент. Наприклад, якщо один елемент у нас буде мати для властивості *android:layout\_weight* значення 2, а інший – значення 1, то в сумі вони дадуть 3, тому перший елемент буде займати 2/3 простору контейнера, а другий – 1/3.

Якщо всі елементи мають значення *android:layout\_weight="1"*, то всі вони будуть рівномірно розподілені по всій площі контейнера.

**5.4.2 RelativeLayout**

RelativeLayout представляє об’єкт ViewGroup, який розміщує дочірні елементи відносно щодо позиції інших дочірніх елементів розмітки або щодо області самої розмітки RelativeLayout. Використовуючи відносне позиціонування, ми можемо встановити елемент по правому краю або в центрі або іншим способом, який надає даний контейнер. Для установки елемента в файлі xml ми можемо застосовувати атрибути групи *android:layout\_* (*android:layout\_above*, *android:layout\_below*, *android:layout\_centerHorizontal*, *android:layout\_alignTop*, *android:layout\_alignParentLeft* тощо). Наприклад:

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

<RelativeLayout xmlns:android="[http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/activity\_main"

android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<TextView android:text="Left Top" android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" android:textSize="26sp" android:layout\_alignParentLeft="true" android:layout\_alignParentTop="true" />

<TextView android:text="Right Top" android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" android:textSize="26sp" android:layout\_alignParentRight="true" android:layout\_alignParentTop="true" />

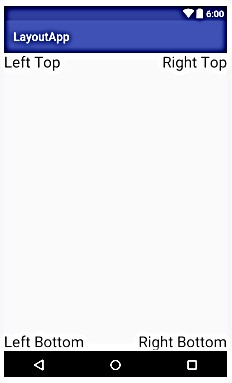
<TextView android:text="Left Bottom" android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" android:textSize="26sp" android:layout\_alignParentLeft="true" android:layout\_alignParentBottom="true" />

<TextView android:text="Right Bottom" android:layout\_height="wrap\_content"

android:layout\_width="wrap\_content" android:textSize="26sp" android:layout\_alignParentRight="true" android:layout\_alignParentBottom="true" />

</RelativeLayout>

На екрані ми побачимо:



**5.4.3 TableLayout**

Контейнер TableLayout впорядковує дочірні елементи управління по стовпцях і рядках. Визначимо в файлі *activity\_main.xml* елемент TableLayout, який буде включати два рядки і два стовпці:

<TableLayout xmlns:android=["http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<TableRow>

<TextView android:layout\_weight="0.5" android:text="Логин" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" />

<EditText android:layout\_weight="1" android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</TableRow>

<TableRow>

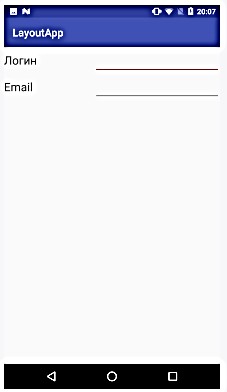
<TextView android:layout\_weight="0.5" android:text="Email" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" />

<EditText android:layout\_weight="1" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" />

</TableRow>

</TableLayout>

На екрані ми побачимо:



**5.4.4 FrameLayout**

Контейнер FrameLayout призначений для виведення на екран одного поміщеного в нього візуального елемента. Якщо ж ми помістимо декілька елементів, то вони будуть накладатися один на одного. Припустимо ми вклали в FrameLayout два елементи TextView:

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

<FrameLayout xmlns:android="[http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android)

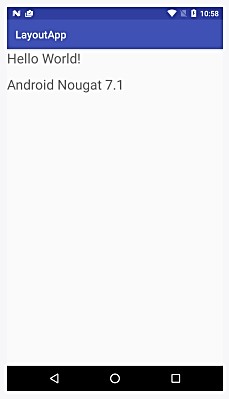
android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Hello World!" android:textSize="26sp"/>

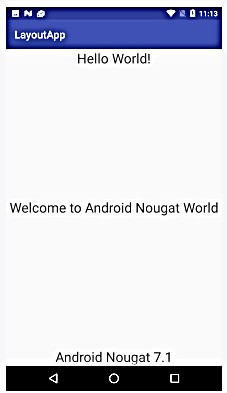
<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Android Nougat 7.1" android:textSize="26sp" android:layout\_marginTop="50dp"/>

</FrameLayout>

Тут обидва елементи позиціонуються в одне і той же місце – в лівий верхній кут контейнера FrameLayout. Щоб уникнути накладання, в даному випадку в другого TextView встановлюється відступ зверху в 50 одиниць:



Елементи управління всередині FrameLayout можуть встановлювати своє позиціонування за допомогою атрибута android: layout\_gravity. При заданні значення цього атрибуту ми можемо комбінувати ряд значень, розділяючи їх вертикальною лінією: bottom | center\_horizontal. В результаті можна отримати таку розмітку:



**5.4.5 ConstraintLayout**

ConstraintLayout представляє собою новий тип контейнерів, який є розвитком RelativeLayout і дозволяє створювати гнучкі та масштабовані інтерфейси. Починаючи з версії Android Studio 2.3 ConstraintLayout був доданий в список стандартних компонентів і навіть є контейнером, який використовується в файлах layout за замовчуванням. Розглянемо приклад його застосування:

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

<android.support.constraint.ConstraintLayout xmlns:android="[http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) xmlns:app="[http://schemas.android.com/apk/res-auto"](http://schemas.android.com/apk/res-auto) xmlns:tools="[http://schemas.android.com/tools"](http://schemas.android.com/tools) android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<ImageView android:text="TextView" android:background="#3F51B5" android:layout\_width="50dp" android:layout\_height="50dp" android:id=["@+id](mailto:@+id)/imageView" app:layout\_constraintLeft\_toLeftOf="parent" android:layout\_marginLeft="16dp" app:layout\_constraintTop\_toTopOf="parent"

android:layout\_marginTop="16dp" />

<TextView android:layout\_width="0dp" android:layout\_height="0dp" android:textSize="20sp" android:id=["@+id](mailto:@+id)/textView"

android:text="Lorem Ipsum is simply dummy text of the printing and typesetting industry ... remaining essentially unchanged"

app:layout\_constraintLeft\_toRig[htOf="@+id](mailto:@+id)/imageView" android:layout\_marginLeft="16dp" app:layout\_constraintTop\_toBottomOf=["@+id](mailto:@+id)/imageView" android:layout\_marginTop="16dp" app:layout\_constraintRight\_toRightOf="parent" android:layout\_marginRight="16dp" app:layout\_constraintBottom\_toTopO[f="@+id](mailto:@+id)/button2" android:layout\_marginBottom="16dp" />

<Button android:text="Cancel" android:layout\_width="93dp" android:layout\_height="53dp" android:id=["@+id](mailto:@+id)/button1" app:layout\_constraintRight\_toRightOf="parent" android:layout\_marginRight="16dp"

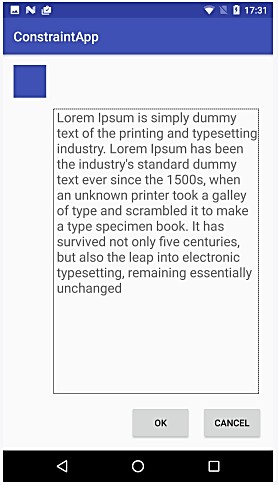
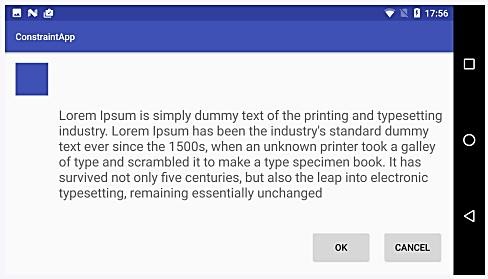
app:layout\_constraintBottom\_toBottomOf="parent" android:layout\_marginBottom="16dp" />

<Button android:text="OK" android:layout\_width="93dp" android:layout\_height="53dp" android:id=["@+id](mailto:@+id)/button2"

app:layout\_constraintRight\_toLeftO[f="@+id](mailto:@+id)/button1" android:layout\_marginRight="16dp" app:layout\_constraintBaseline\_toBaselineOf=["@+id](mailto:@+id)/button1" />

</android.support.constraint.ConstraintLayout>

На екрані це виглядатиме так:



По-перше, тут елемент позиціонується відносно контейнера ConstraintLayout: від верхньої і лівої межі контейнера до відповідних меж ImageView по 16 dip.

По-друге, відносно контейнера позиціонується також кнопка з *id=button1*: від правої і нижньої межі контейнера до відповідних меж Button також по 16 dip.

По-третє, друга кнопка з *id=button2* позиціонується відносно першої кнопки: від правої межі другої кнопки до лівої межі першої кнопки (*app:layout\_constraintRight\_toLeftOf=*[*"@+id*](mailto:@+id)*/button1"*) також 16 dip. І щоб обидві кнопки знаходилися на одному рівні, у них задано вирівнювання по базовій лінії: *app:layout\_constraintBaseline\_toBaselineOf=*[*"@+id*](mailto:@+id)*/button1"*.

Нарешті елемент TextView із фрагментом тексту позиціонується одразу відносно як до контейнера, так і до елемента ImageView і до другої кнопки.

**5.5 Одиниці вимірювання розміру елементів екрану**

Розрізняють декілька одиниць вимірювання розміру елементів екрану:

 **px**: пікселі поточного екрану. Дана одиниця вимірювання не рекомендується, оскільки реальне представлення зовнішнього вигляду екрану програми може змінюватися залежно від конкретного пристрою; кожен пристрій має індивідуально визначену кількість пікселів на один дюйм. Відповідно, кількість пікселів на екрані є різною для різних пристроїв;

 **dp**: (device-independent pixels) – пікселі, незалежні від щільності екрану. Це абстрактна одиниця вимірювання, яка базується на фізичній щільності екрану з роздільною здатністю 160 dpi (точок на дюйм). У цьому випадку 1dp=1px. Якщо ж розмір екрана більший або менший, ніж 160dpi, кількість пікселів, які застосовуються для представлення 1dp, відповідно? збільшується або зменшується. Наприклад, на екрані з 240 dpi 1dp=1,5px, а на екрані з 320dpi матимемо: 1dp=2px. Загальна формула для обчислення кількості фізичних пікселів з dp така: px=dp\*(dpi/160);

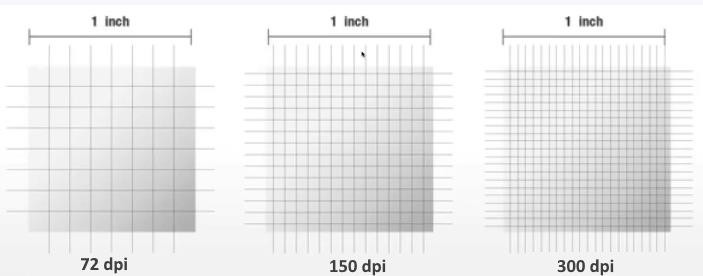
 **sp**: (scale-independent pixels) – пікселі, незалежні від масштабування. Допускає налаштування розмірів користувачем. Рекомендується для роботи із шрифтами;

 **pt**: 1/72 дюйма, базується на фізичних розмірах екрану;

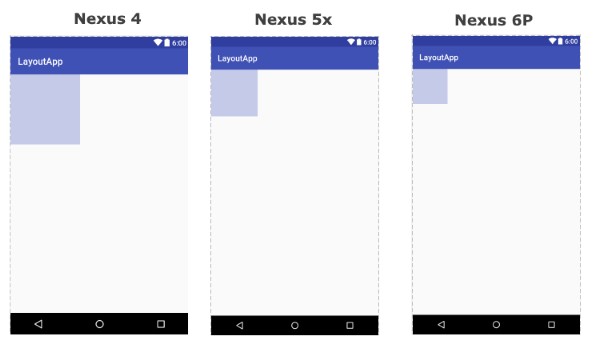
 **mm**: міліметри;

 **in**: дюйми.

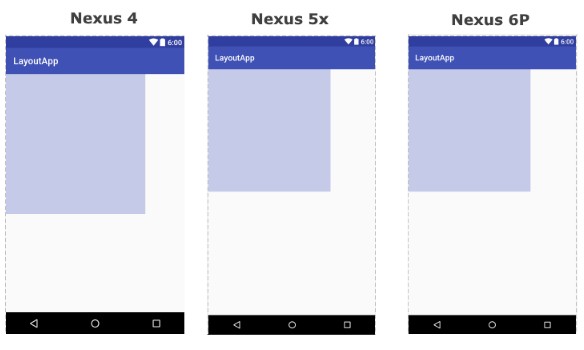
Рекомендованою одиницею для використання є dp. Це пов’язано з тим, що світ мобільних пристроїв на Android досить сильно фрагментований в контексті широкого різноманіття розширення та розмірів екрану. І чим більшою є щільність (кількість пікселів на один дюйм), тим більше пікселів буде на екрані:



Використовуючи стандартні фізичні пікселі, ми неминуче стикнемося із тою проблемою, що розміри елементів сильно варіюватимуться для різних екранів. Наприклад, для 3 пристроїв з різними характеристиками екрану Nexus 4, Nexus 5X та Nexus 6P, квадрат розмірами 300px на 300px на екрані виглядатиме так:



Тепер візьмемо квадрат із сторонами 300dp на 300dp. Як бачимо, тепер квадрат виглядає більш-менш коректно.



Для спрощення роботи всі розміри розбито на декілька груп:

 **ldpi** (low): ~120dpi;

 **mdpi** (medium): ~160dpi;

 **hdpi** (high): ~240dpi (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).