



Мобильная разработка

Пирская Любовь Владимировна,
к.т.н., доцент кафедры МОП ЭВМ
lpirskaya@sfedu.ru

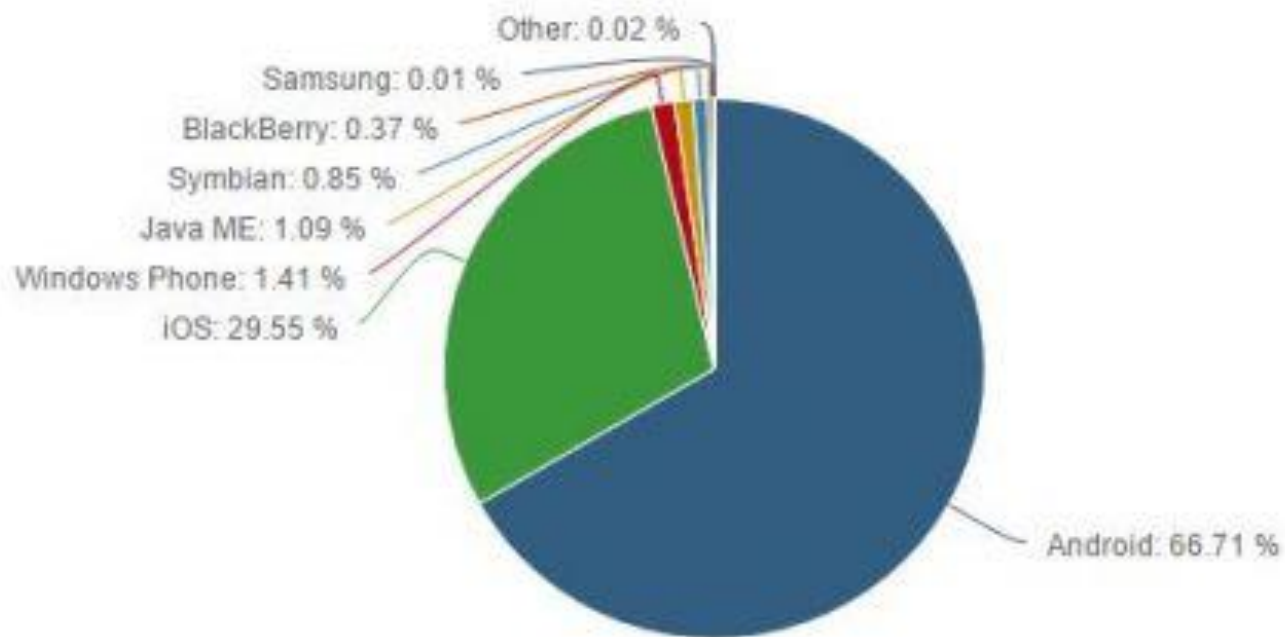
Дисциплины

- Разработка приложений для мобильных устройств
 - бакалавриат 09.03.04 «Программная инженерия»
- Программное обеспечение для мобильных платформ и системы цифровой обработки сигналов
 - магистратура 09.04.04 «Программная инженерия»

Распределение баллов

Виды контрольных мероприятий	Бакалавры (экзамен)	Магистры (диф. зачет)
Лабораторная работа №1	10	15
Лабораторная работа №2	10	15
Посещение лекций	10	-
Индивидуальное задание		10
Экзамен	20	-
Доп. баллы	10	-
Общее количество баллов	50 (10)	40

Мобильные ОС

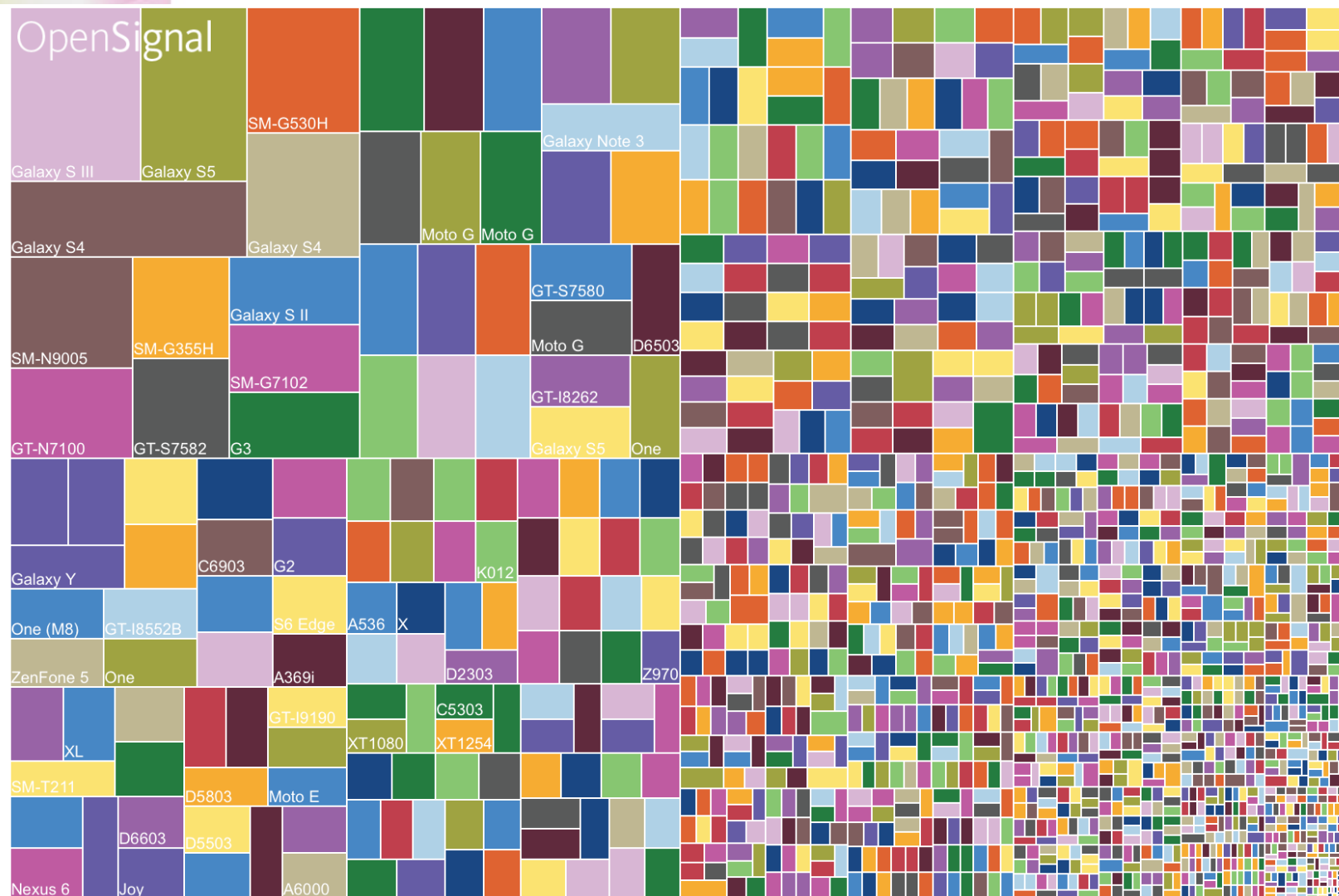


Почему Android?

- Открытая операционная система
- Open Handset Alliance
- Google
- Можно делать все что угодно
- Динамично развивается
- Быстро растёт



Фрагментация



August 2014

August 2015

Фрагментация

Недостатки:

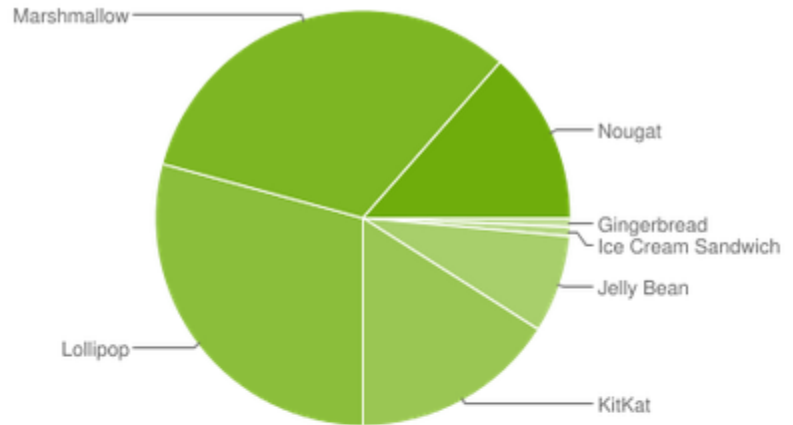
- Все формы и размеры
- Разное качество исполнения
- Невозможно протестировать на всех целевых устройствах

Достоинства:

- Охват аудитории
- Девайс на любой вкус

Dashboards

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.7%
4.1.x	Jelly Bean	16	2.7%
4.2.x		17	3.8%
4.3		18	1.1%
4.4	KitKat	19	16.0%
5.0	Lollipop	21	7.4%
5.1		22	21.8%
6.0	Marshmallow	23	32.3%
7.0	Nougat	24	12.3%
7.1		25	1.2%

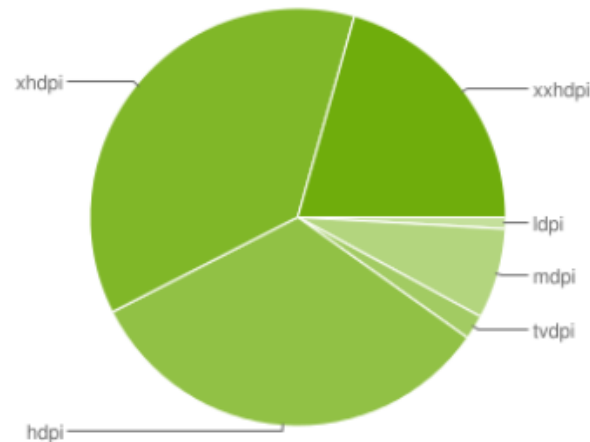
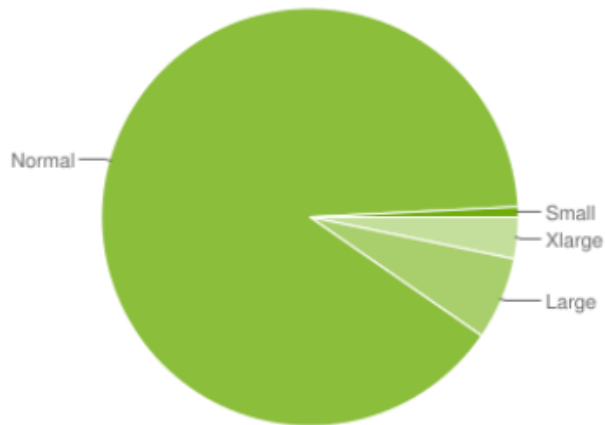


Data collected during a 7-day period ending on August 8, 2017.

Any versions with less than 0.1% distribution are not shown.

Dashboards

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.8%						0.8%
Normal		1.7%	0.2%	31.7%	35.5%	20.5%	89.6%
Large	0.1%	3.1%	1.7%	0.5%	0.9%	0.1%	6.4%
Xlarge		2.2%		0.5%	0.5%		3.2%
Total	0.9%	7.0%	1.9%	32.7%	36.9%	20.6%	



Data collected during a 7-day period ending on August 8, 2017.

Any screen configurations with less than 0.1% distribution are not shown.

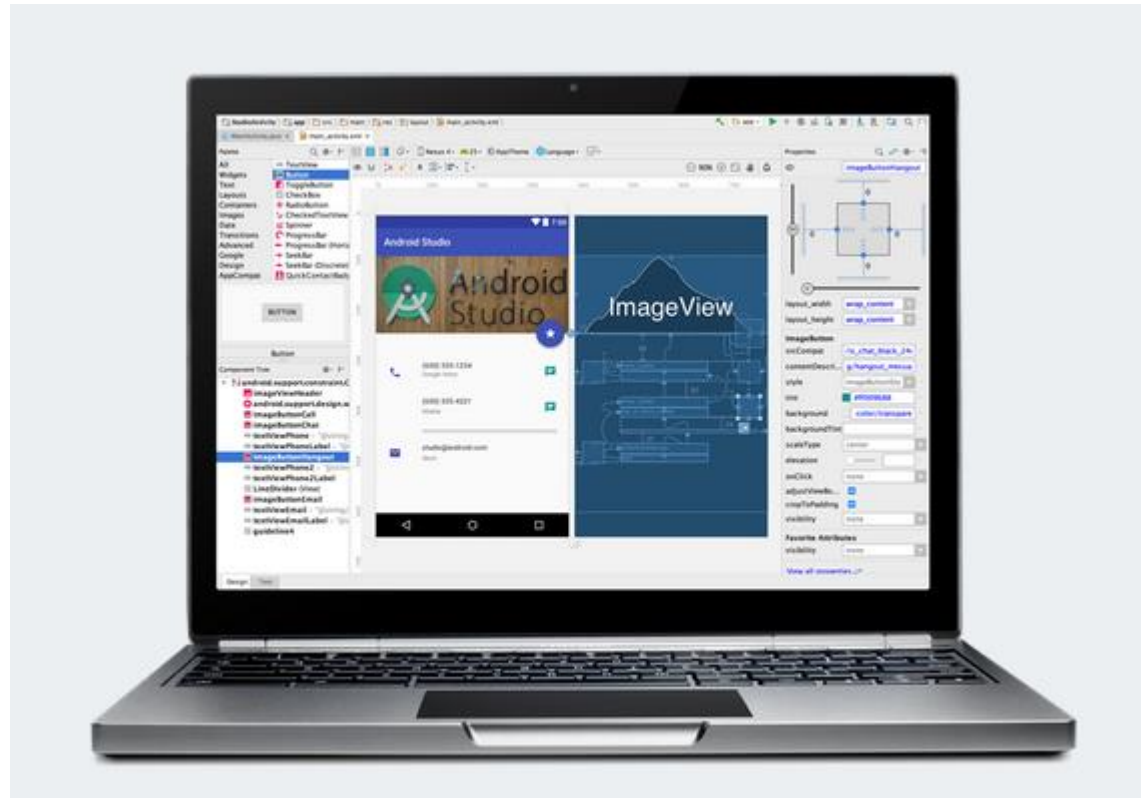
Архитектура Android



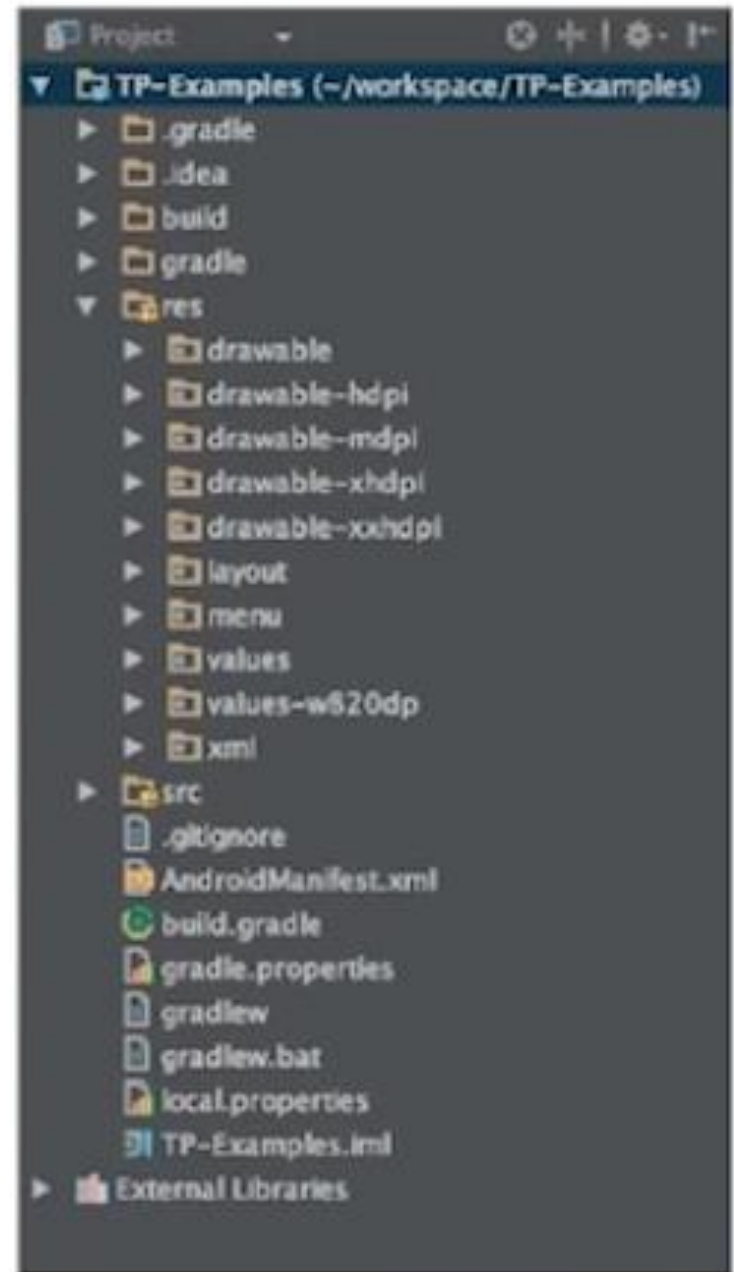
Среда разработки

Android Studio

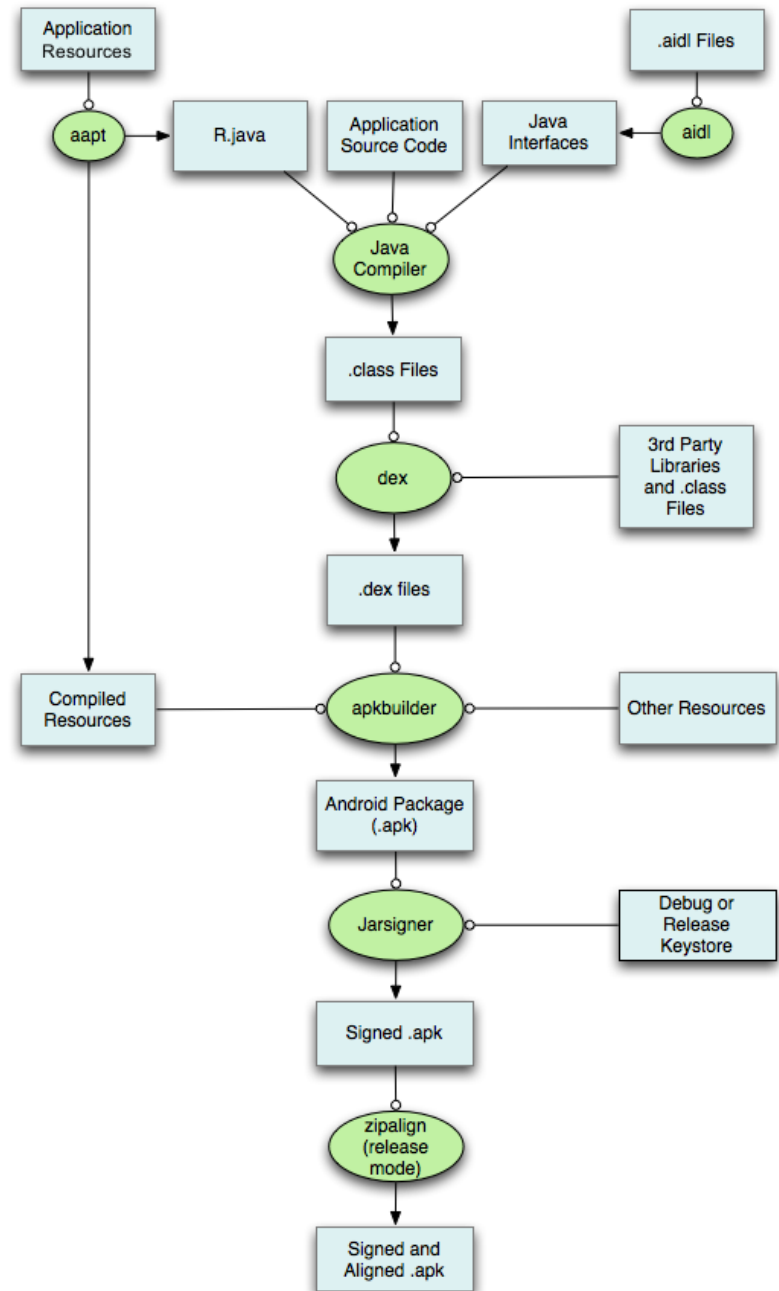
<https://developer.android.com/studio/index.html>



Структура приложения



Gradle



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.contactsdiff"
    android:versionCode="1"
    android:versionName="1.0" >

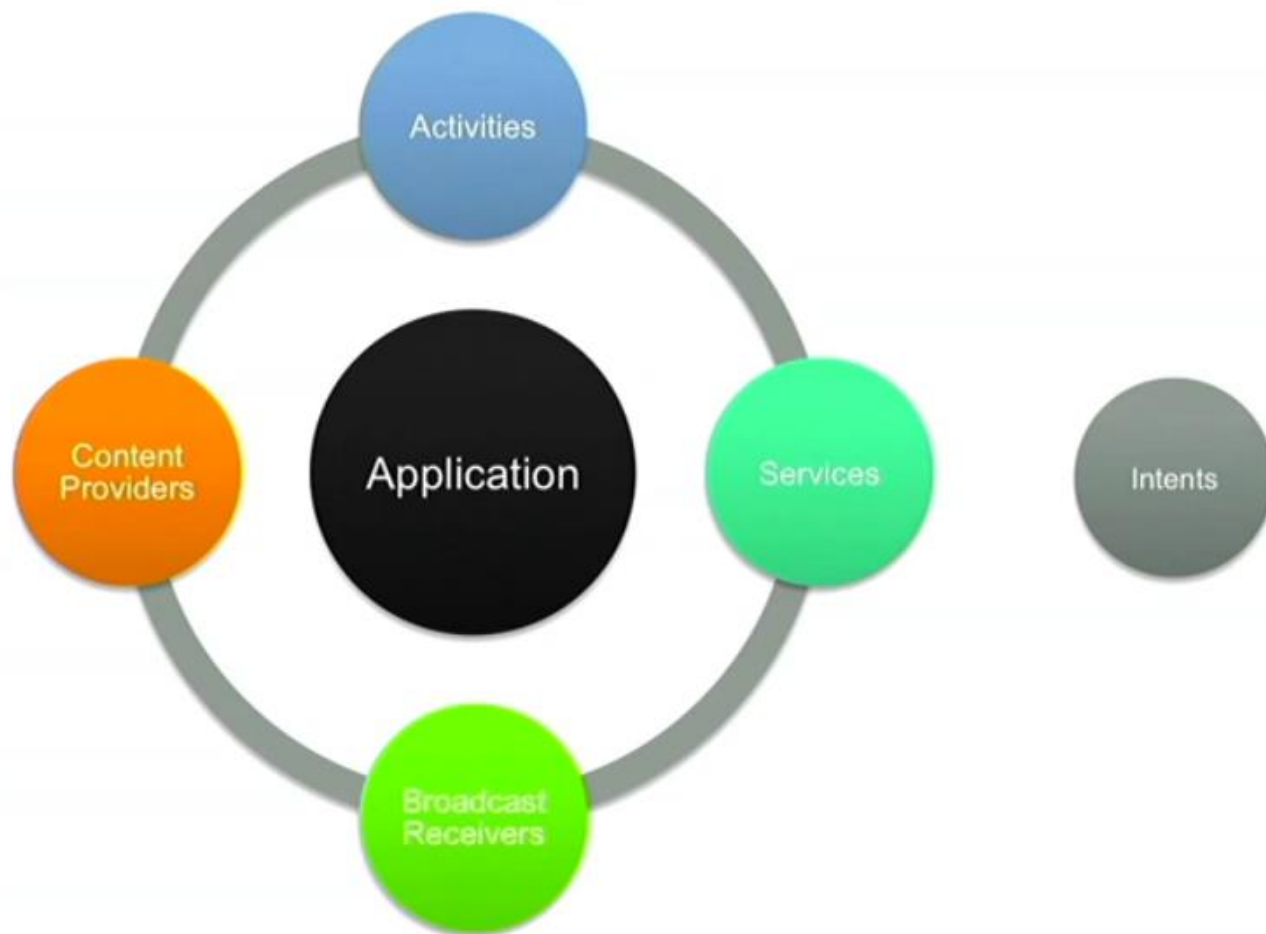
    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="14" />

    <uses-permission android:name="android.permission.READ_CONTACTS"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
```

- объявляет имя Java-пакета приложения;
- описывает компоненты приложения;
- содержит список необходимых разрешений для обращения к защищенным частям API и взаимодействия с другими приложениями;
- объявляет минимальный уровень API Android;
- перечисляет связанные библиотеки;

Компоненты



Activity

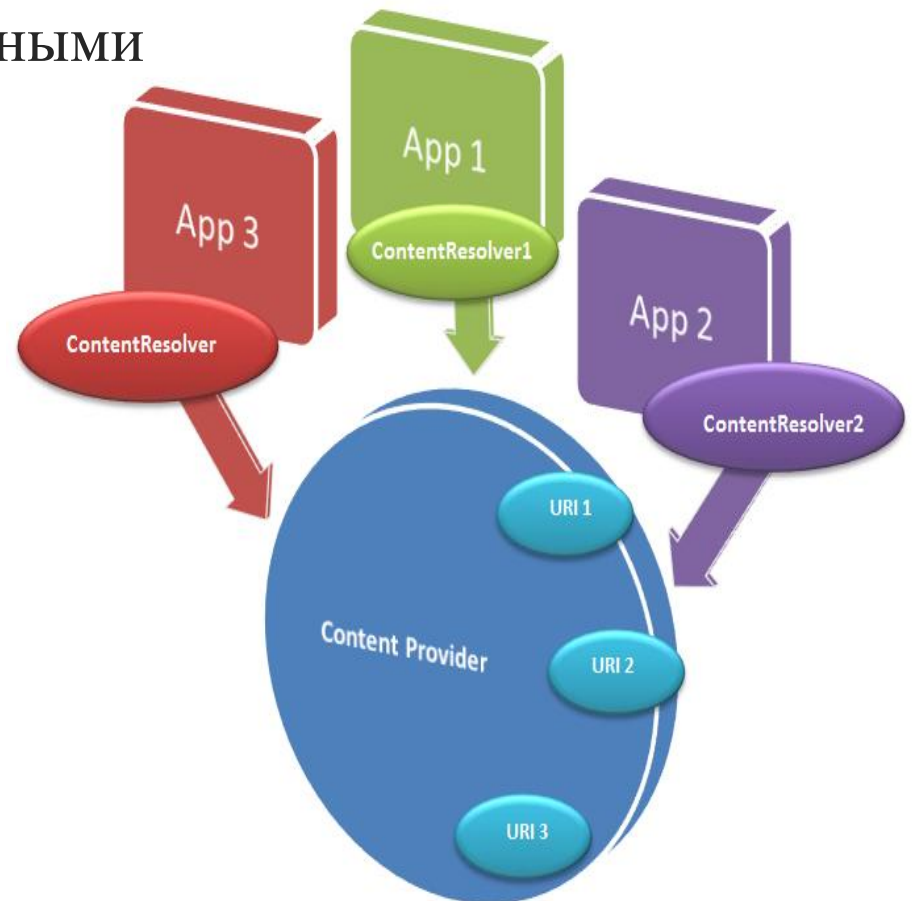
- Основной компонент
- Экран с интерфейсом
- Не обязательно весь экран
- Одна activity вызывает другую
- Точка входа в приложение
- Описывается в манифесте

Service

- Без интерфейса
- Фоновый режим
- Работа с remote процессами
- Application layers

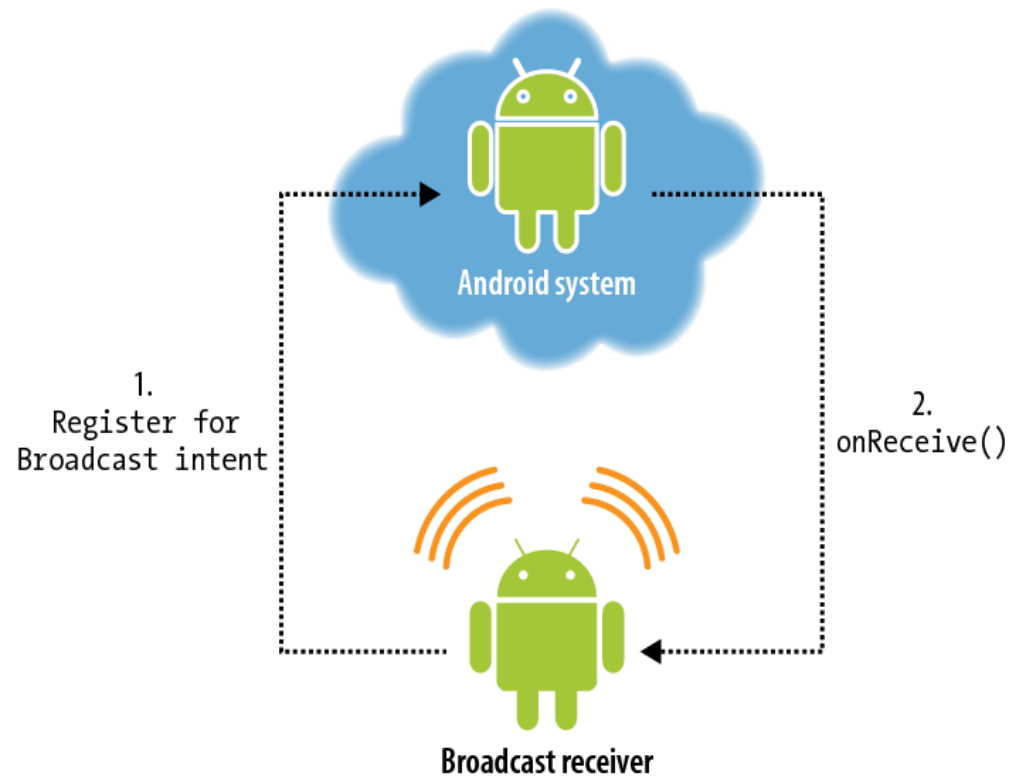
Content provider

- Шаринг данных
- Фасад для хранилищ (БД, Сервер, файловая система и проч.)
- Управление данными



Broadcast Receiver

- Оповещение
- Системные
- Пользовательские
- Могут создавать нотификации
- Ресурсоемкие

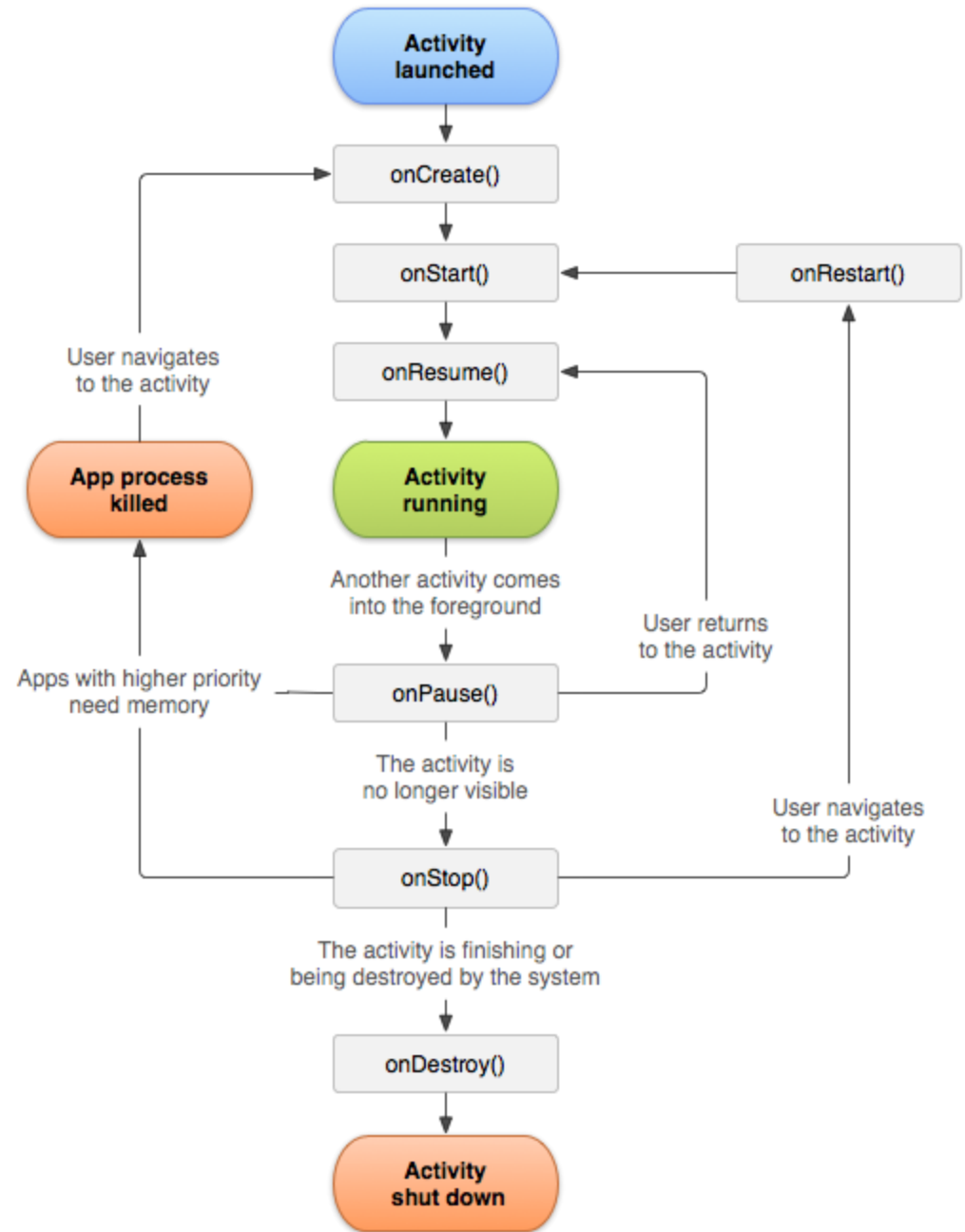


Важность процессов

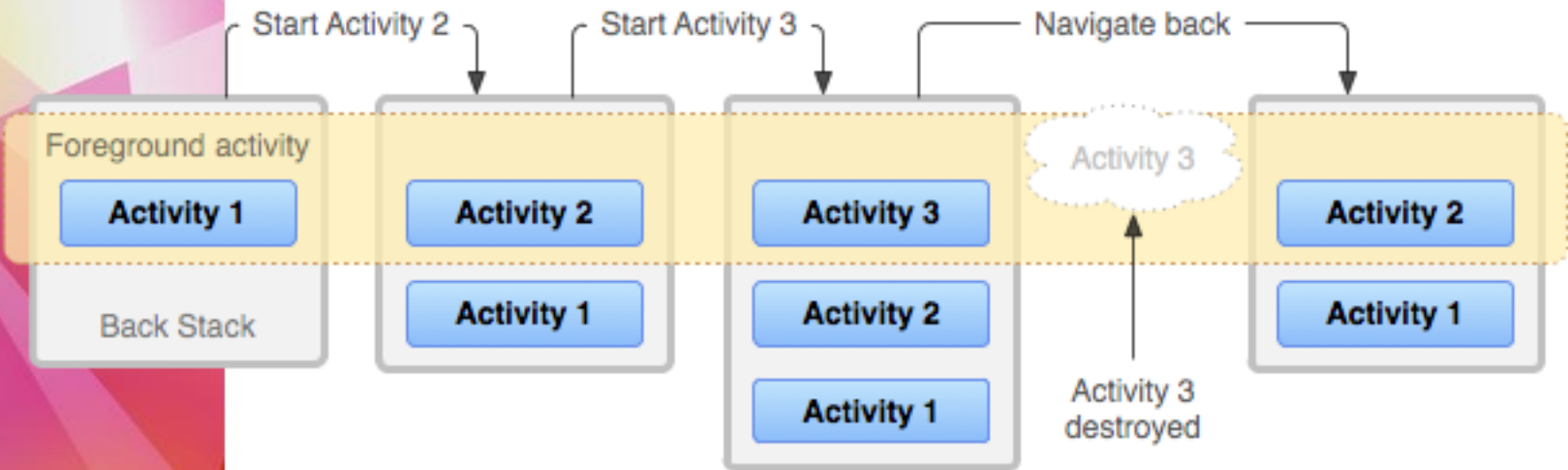


Activity

- Life Cycle
- Tasks & Back Stack
- Manifest
- Save instance state
- Intents



Tasks & Back Stack



- Из activity можно вызвать другую activity
- Можно вызвать activity другого приложения
- Task – коллекция activities, которые складываются в стек
- Новая activity пушится в стек, а у предыдущей вызывается `onStop()`
- По кнопке back верхняя activity достается из стека и уничтожается, а у activity под ней вызывается `onResume()`
- Activities в стеке никогда не меняются местами!

Методы жизненного цикла Activity

- `OnCreate(Bundle savedInstanceState)`
Вызывается когда создается activity
Получает сохраненное состояние(если оно есть)
- `OnResume()`
Вызывается перед тем как activity станет видимым пользователю
- `OnPause()`
Вызывается перед тем как у другой activity вызовется `onResume()`
Здесь все завершающие операции
Не делать долгих операций!
- `OnStop()`
Вызывается, когда activity уже не видима пользователю
- `onDestroy()`
Вызывается перед уничтожением activity

Saving instance state

- Если выйти из приложения по кнопке “Home”, или запустить другую activity, то предыдущая activity остается в памяти, поэтому когда (если) вы вернетесь в нее она полностью восстановит свое состояние
- Система может убить activity в back stack
- Пользователь ничего об этом не знает
- `onSaveInstanceState(Bundle outState)`
- Восстановит только если activity была убита системой!(кнопка Back не считается)
- Существует реализация по умолчанию
- Вызовется перед `onStop()`
- Не использовать для хранения данных!

Intent

- Сообщение
- Объект, содержащий описание запрашиваемой операции
- Объект, оповещающий о произошедшем событии
- Запуск activity, service

Layout

- Определяет порядок расположения элементов интерфейса
- Обычно создается с помощью xml
- Большое количество layouts в библиотеке android

LinearLayout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

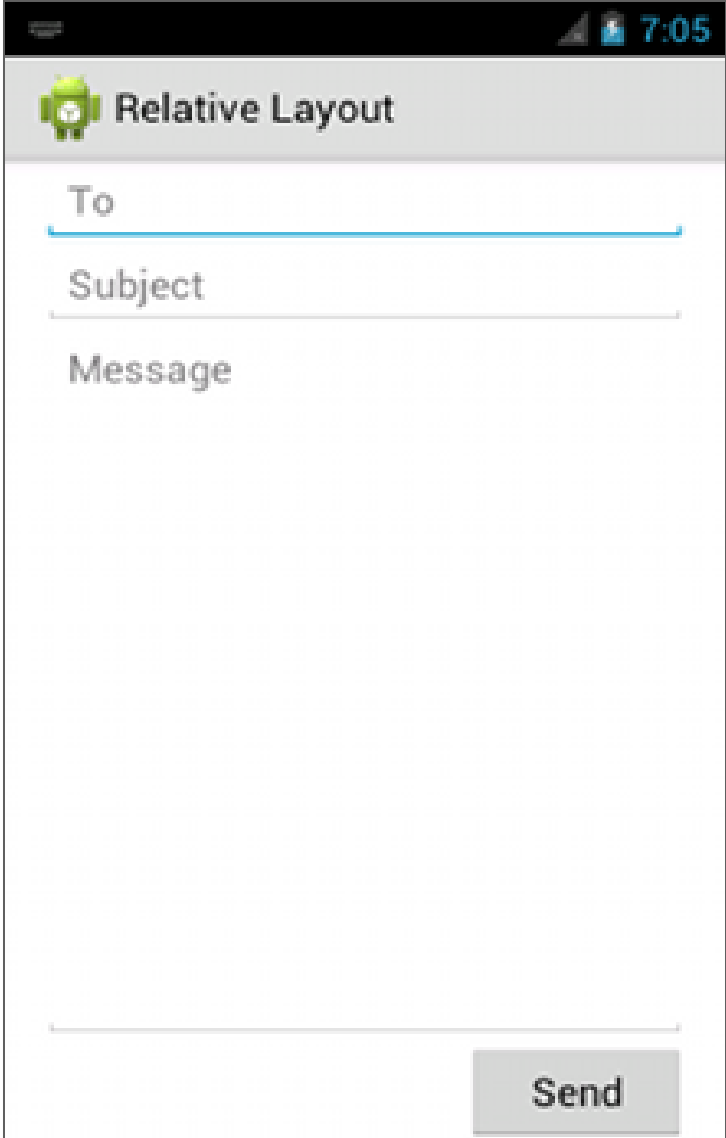
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="@drawable/wallpaper_gallery_background">

        <Gallery android:id="@+id/gallery"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:spacing="-4dp" />

        <Button android:id="@+id/set"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/wallpaper_instructions"
            android:layout_gravity="center_horizontal" />
    </LinearLayout>
</RelativeLayout>
```

LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



The screenshot shows an Android application window with the title "Relative Layout". The interface consists of three vertically stacked text input fields. The first field is labeled "To", the second is labeled "Subject", and the third is labeled "Message". At the bottom right of the screen is a grey button labeled "Send". The status bar at the top shows the time as 7:05 and some system icons.

Gravity

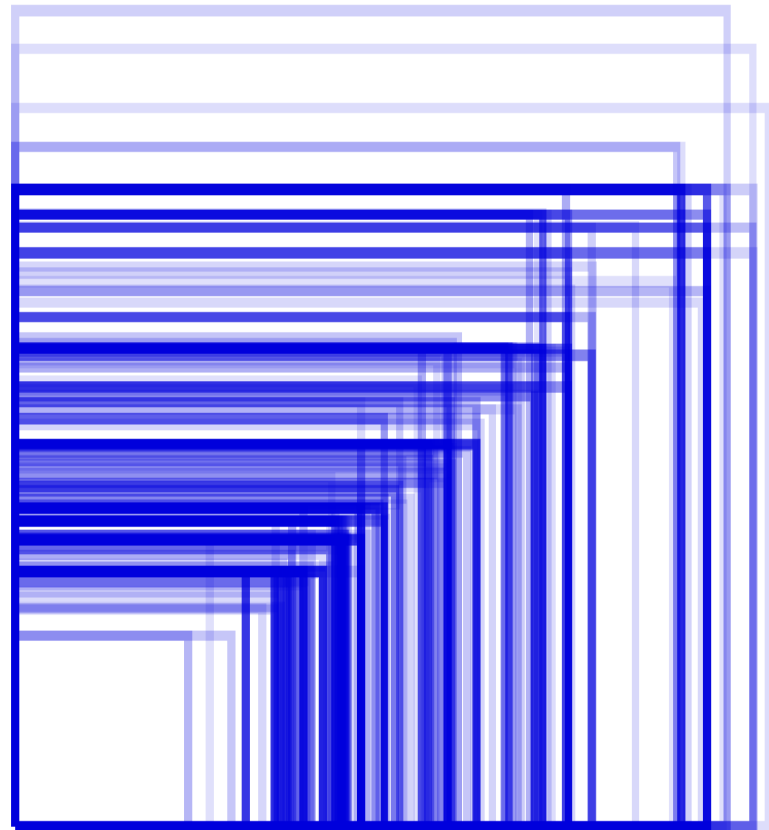
- `android:gravity` – расположение контента внутри контейнера
- `android:layout_gravity` – расположение относительно родителя

Layout weight

- Параметр `android:layout_weight` позволяет распределять пространство между компонентами в долевом отношении
- Если используются вес, то соответствующее измерение компонента должно равняться 0
- $$\text{space assign to child} = (\text{child individual weight}) / (\text{sum of weight of every child in Linear Layout})$$
- По умолчанию вес = 0, это означает, что элемент займет минимум места, которого потребует контент
- Не использовать `weight` во вложенных `LinearLayout`!

Multiple screens

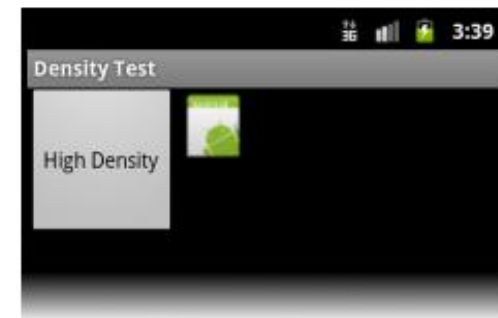
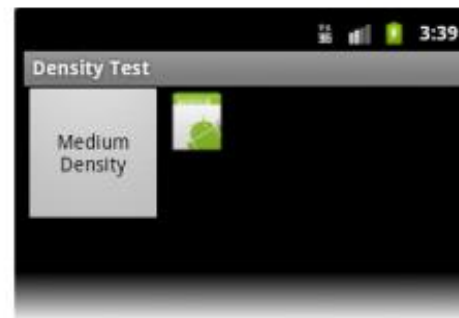
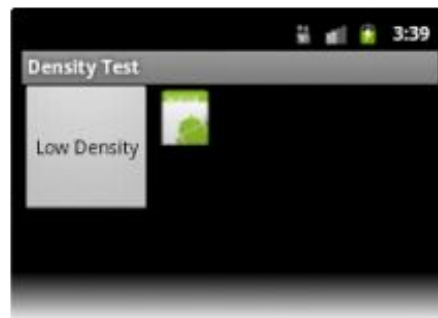
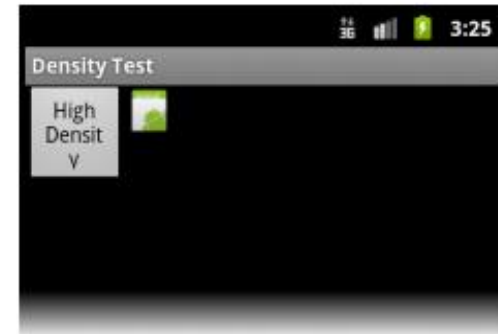
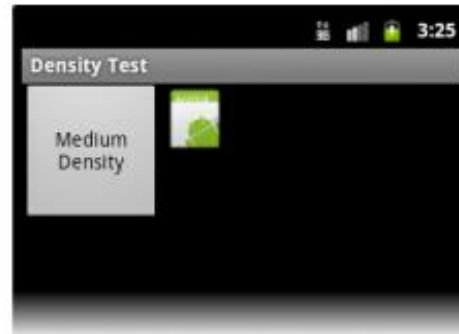
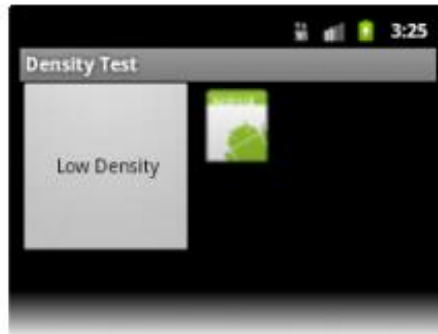
- Огромное количество экранов с разными размерами, ориентацией, разрешениями и проч.
- Android предоставляет методы для удобной организации ресурсов



Multiple screens

- Размер экрана(физический размер)
- Screen density(dpi - точек на дюйм) – low, medium, high, extra high
- Ориентация(портрет, ландшафт)
- Разрешение(не работаем с разрешением)
- Density-independent pixel(dp) – виртуальный пиксель
- $1 \text{ dp} = 1 \text{ px}$ на 160dpi экране – medium экран
- $\text{px} = \text{dp} * (\text{dpi} / 160)$ – например на экране 240dpi $1 \text{ dp} = 1.5 \text{ px}$
- Всегда использовать dp, и никогда px

Multiple screens. Density independence



Multiple screens. Правила

- Все только в dp
- Разные layout для разных размеров экранов (small, normal, large, xlarge)
- Разные изображения для экранов с разной плотностью
- Использовать квалификаторы
- http://developer.android.com/guide/practices/screens_support.html
- <http://developer.android.com/guide/topics/resources/providing-resources.html>