Introducing Mobile Application Development for Android

Presented by:

Ahmed Misbah

Agenda

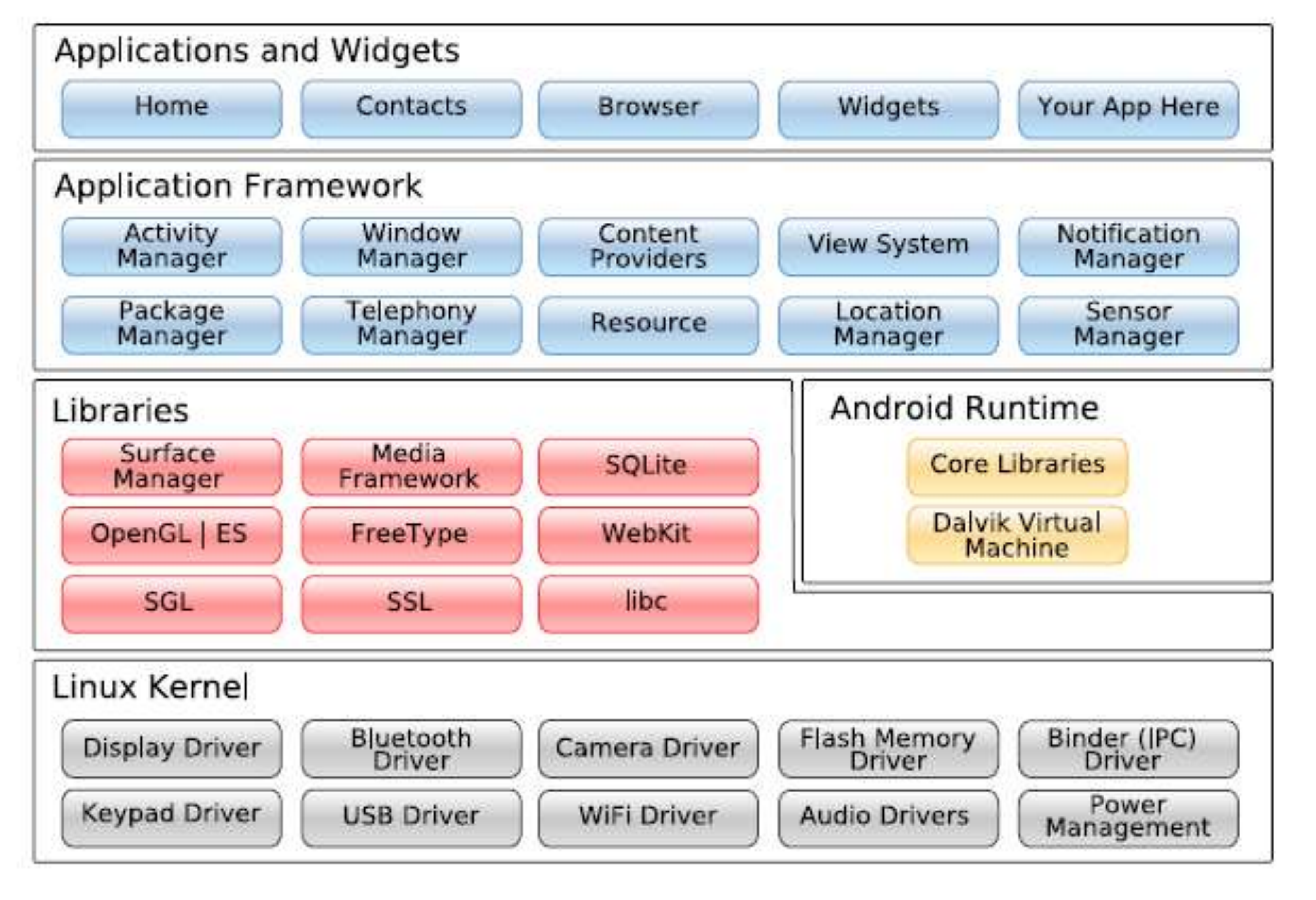
- Introduction
- Android SDK Features
- Developing an Android Application
- Android Market
- Android Application Trends

INTRODUCTION

What is Android?

- Google’s mobile operating system
- Based on Linux Kernel
- Offers an SDK and NDK
- Latest SDK version is 3.0/3.1 (Honeycomb)

Architecture Overview



Applications and Widgets

Home | Contacts | Browser | Widgets | Your App Here

Application Framework

Activity Manager | Window Manager | Content Providers | View System | Notification Manager

Package Manager | Telephony Manager | Resource | Location Manager | Sensor Manager

Libraries

Surface Manager | Media Framework | SQLite

OpenGL | ES | FreeType | WebKit

SGL | SSL | libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver | Bluetooth Driver | Camera Driver | Flash Memory Driver | Binder (IPC) Driver

Keypad Driver | USB Driver | WiFi Driver | Audio Drivers | Power Management

Linux Kernel

- Android uses Linux for its memory management, process management, networking, and other operating system services

Native Libraries

- Shared libraries all written in C or C++
- Compiled for the particular hardware architecture used by the phone
- Preinstalled by the phone vendor
- Can be developed using NDK

Native Libraries (cont’d)

- Surface Manager
- 2D, 3D Graphics
- Media Codecs
- SQL Database
- Browser Engine

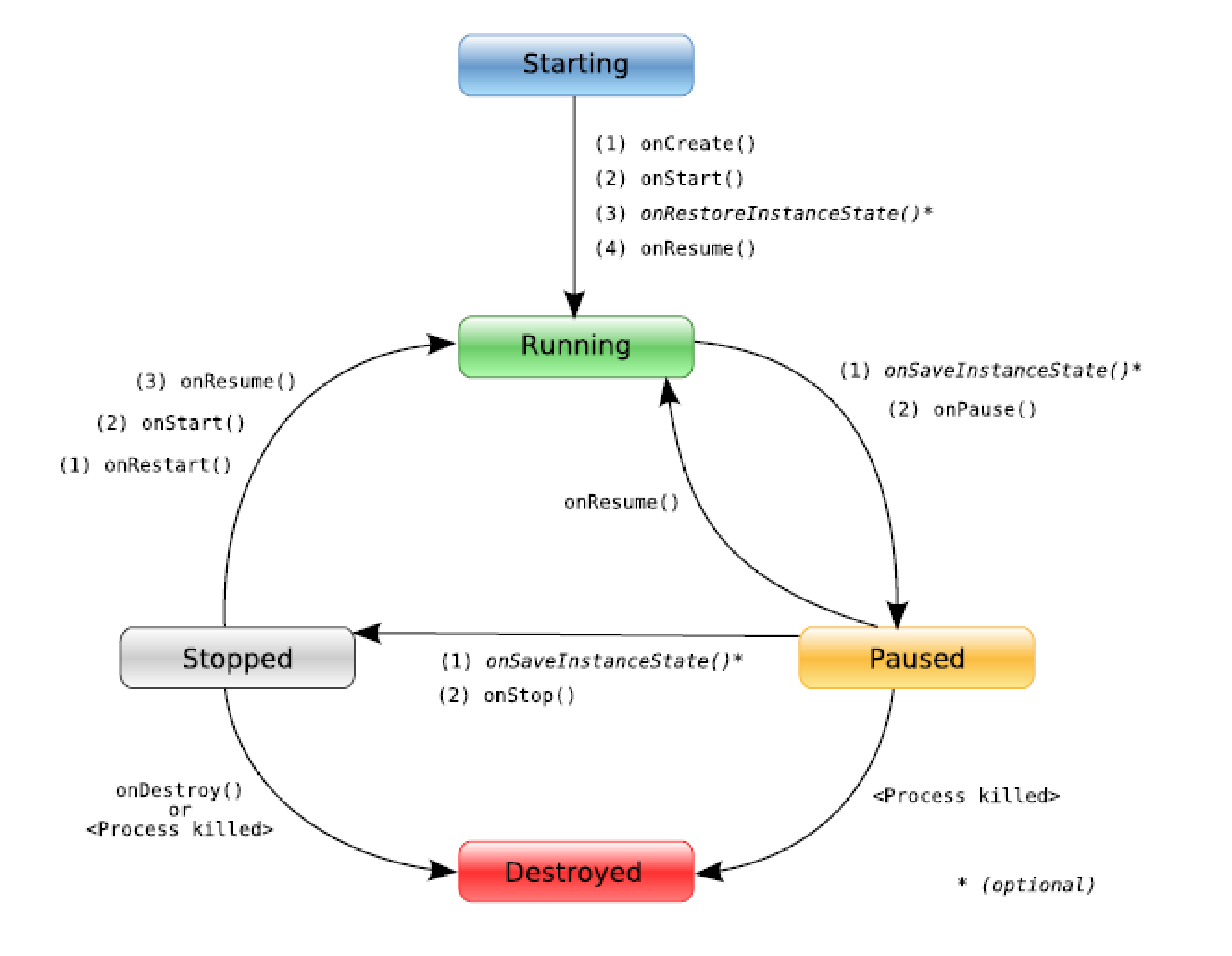
Android Runtime

- Dalvik VM
  - Google’s implementation of Java
  - Optimized for mobile devices
  - Runs .dex files which are more compact and efficient than standard .class files
- Core Java libraries
  - Not those of JSE or JME but have some similarities

Application Framework

- Activity Manager
- Content providers
- Resource Manager
- Location Manager
- Notification Manager

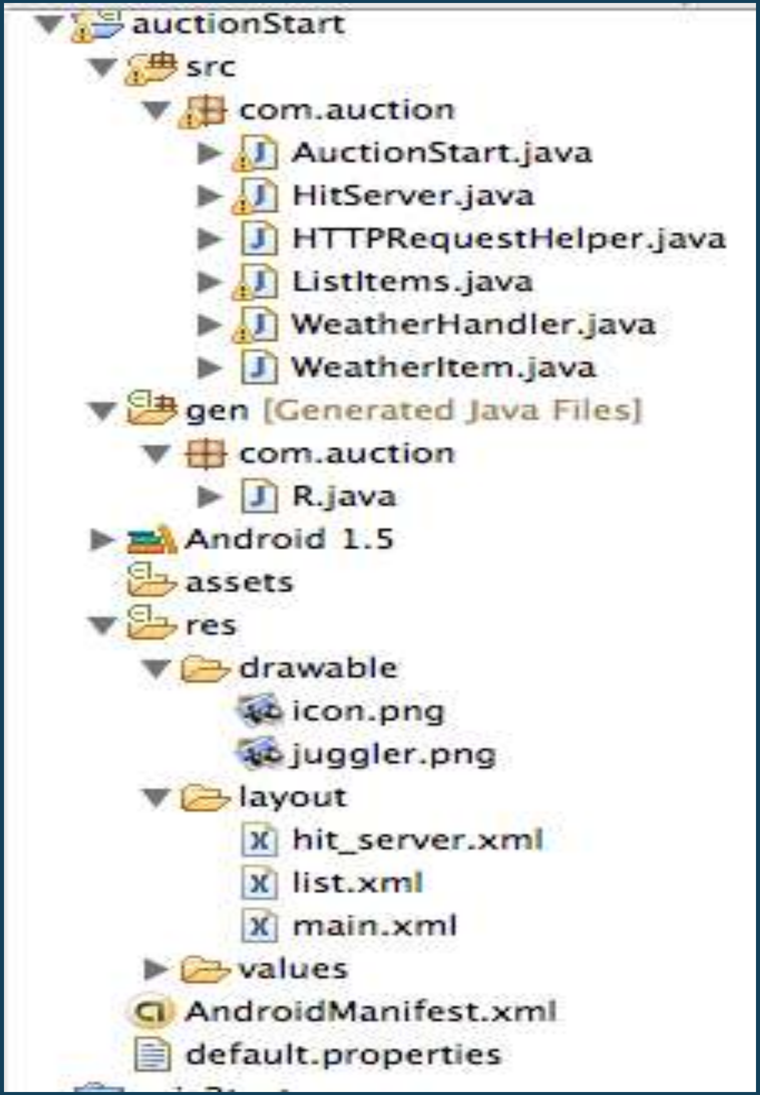
Application Lifecycle



Building Blocks

- **Activities** : User Interface
- **Intent**: A mechanism for describing a specific action
- **Service**: A task that runs in the background without user interaction
- **Content providers**: is a set of data wrapped up in a custom API to read and write it

Application Structure



```
auctionStart
  src
    com.auction
      AuctionStart.java
      HitServer.java
      HTTPRequestHelper.java
      ListItems.java
      WeatherHandler.java
      WeatherItem.java
  gen [Generated Java Files]
    com.auction
      R.java
  Android 1.5
  assets
  res
    drawable
      icon.png
      juggler.png
    layout
      hit_server.xml
      list.xml
      main.xml
    values
  AndroidManifest.xml
  default.properties
```

Resources

- Stored in **res** folder
- Includes all non code information (e.g. localized text and images)
- Resources compiler compresses and packs all resources in a class named **R**

Android Manifest

- Every application must have an **AndroidManifest.xml** file in its root directory
- Manifest presents essential information about the application to the Android system:
  - Java package
  - Components of the application (Activities, Services, etc.)
  - Permissions the application
  - Minimum level of the Android
  - Libraries that the application utilizes

Security

- Stored in **Android-Manifest.xml**

- Contains following permissions:
  - INTERNET
  - READ_CONTACTS
  - WRITE_CONTACTS
  - RECEIVE_SMS
  - ACCESS_COARSE_LOCATION
  - ACCESS_FINE_LOCATION
  - WRITE_EXTERNAL_STORAGE

ANDROID SDK FEATURES

Android SDK Features

- User Interface
- Graphics
- Multimedia
- Data Storage
- Networking
- Locating and Sensing
- Telephony, Messaging and Notification
- I18N and Localization

USER INTERFACE

Overview

- Design Methods
  - Declare UI elements in XML (Declarative design)
  - Instantiate UI elements at runtime

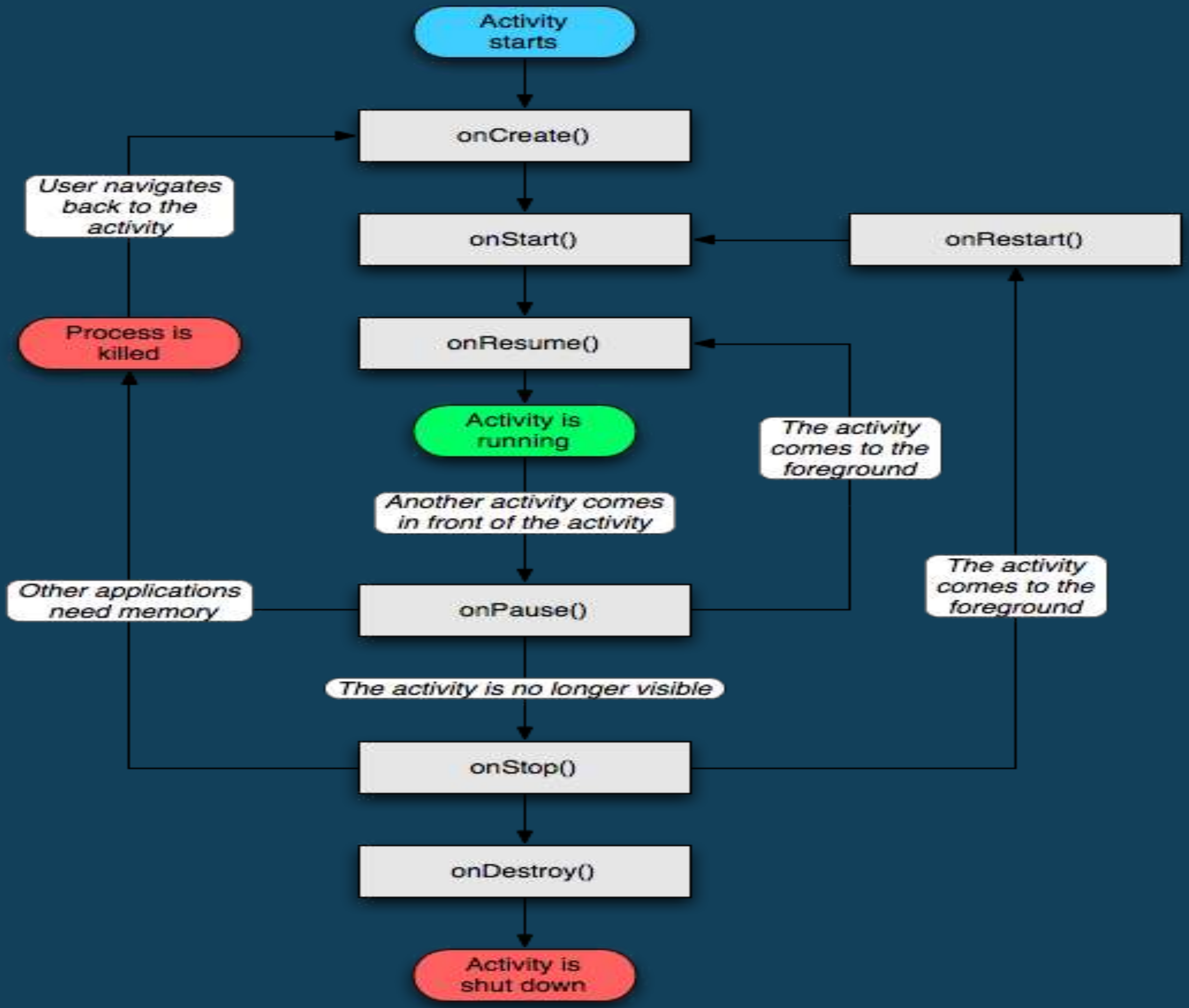
Activity Class

- **Activity** class takes care of creating a window in which UI can be placed

- There is a **one-to-one** relationship between an Activity and a UI screen

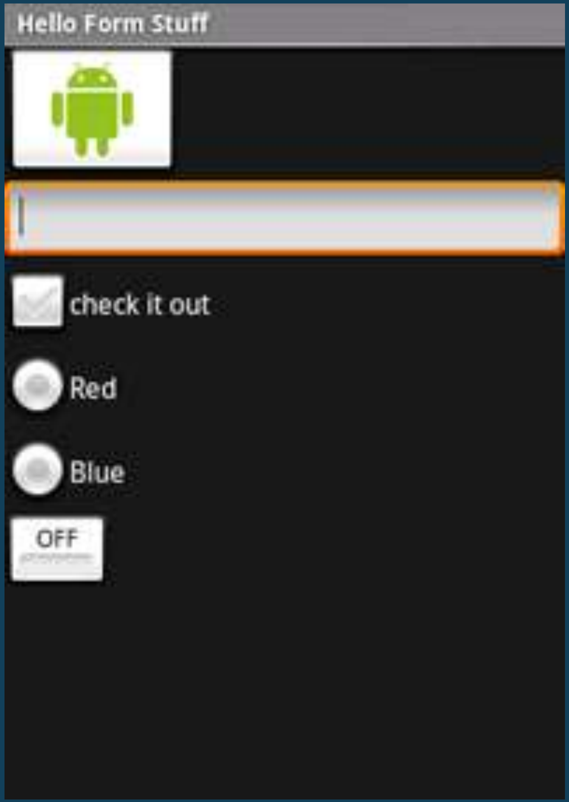
- Activities are made up of subcomponents called ***Views***

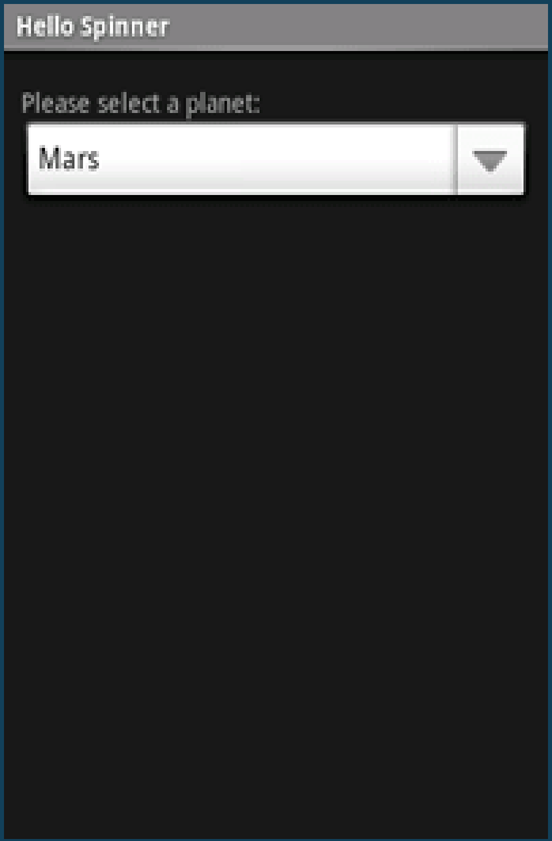
Activity Lifecycle


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

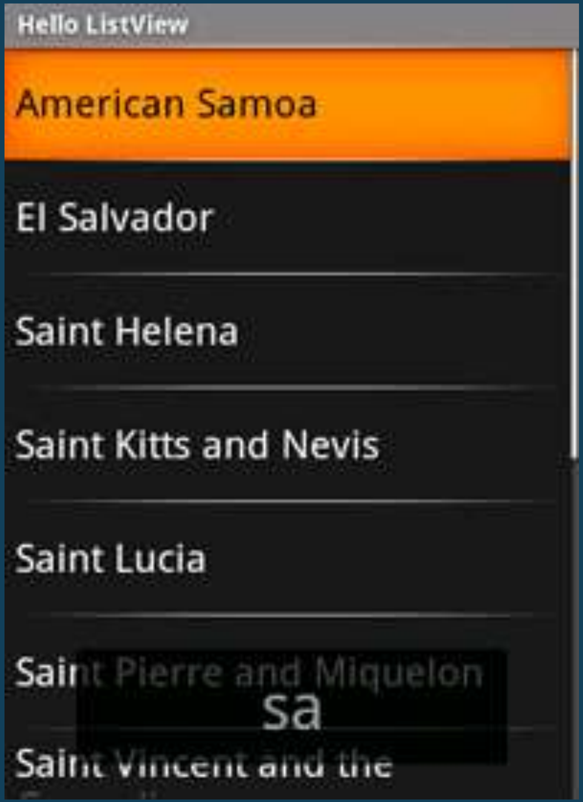
Views

- **Views** are what your users will see and interact with

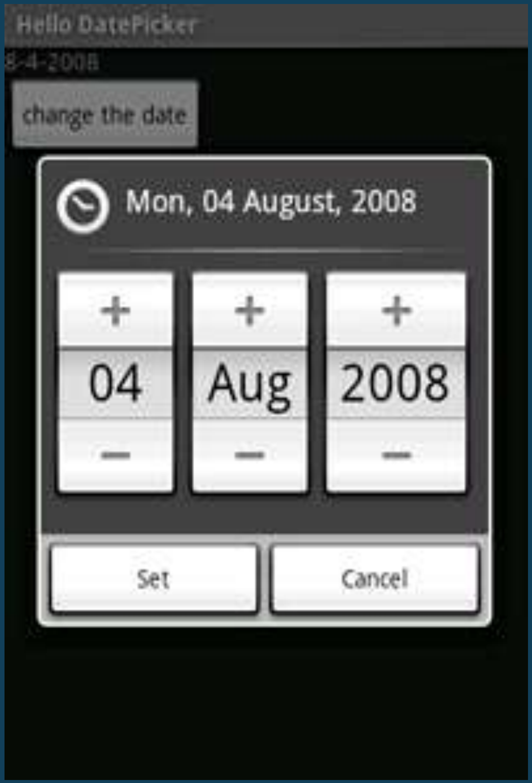




Please select a planet: Mars



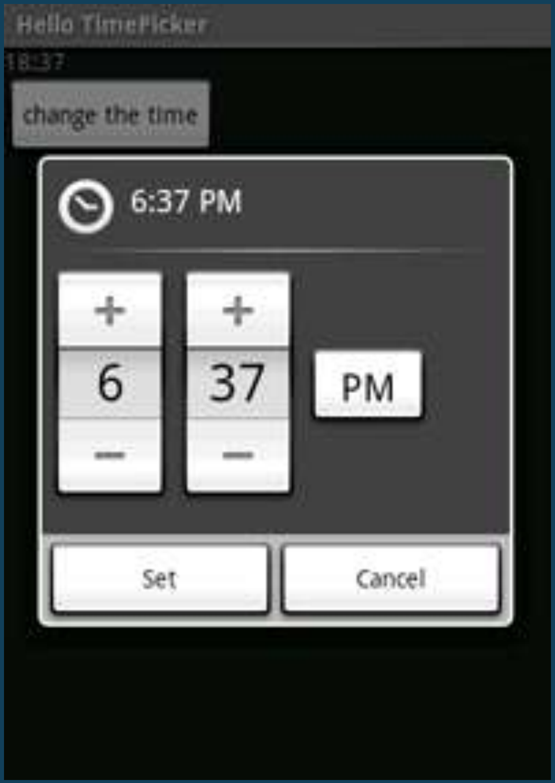
Views (cont'd)


Hello DatePicker

8-4-2008

Mon, 04 August, 2008

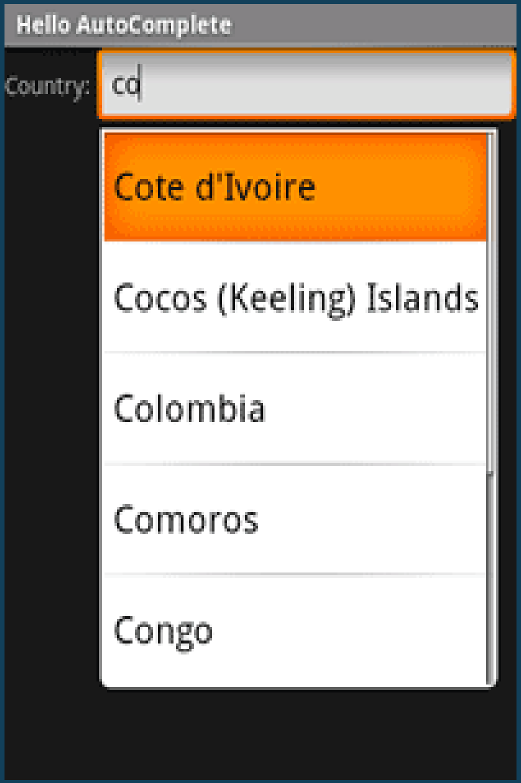
04 Aug 2008


Hello TimePicker

18:37

6:37 PM

6 37 PM


Hello AutoComplete

Country: co

Cote d'Ivoire

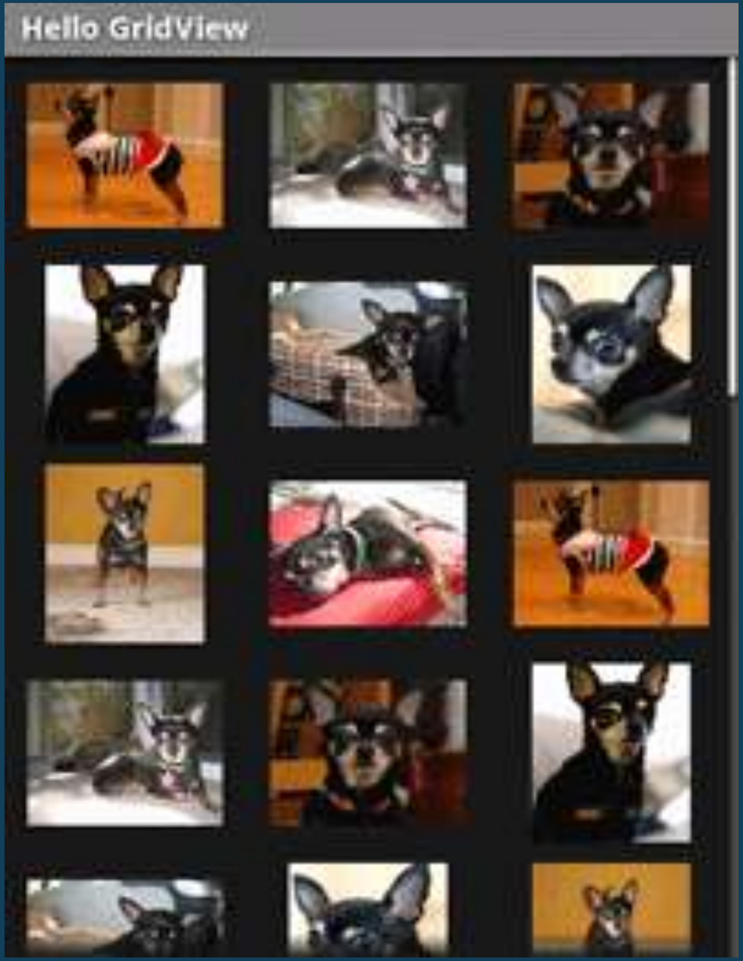
Cocos (Keeling) Islands

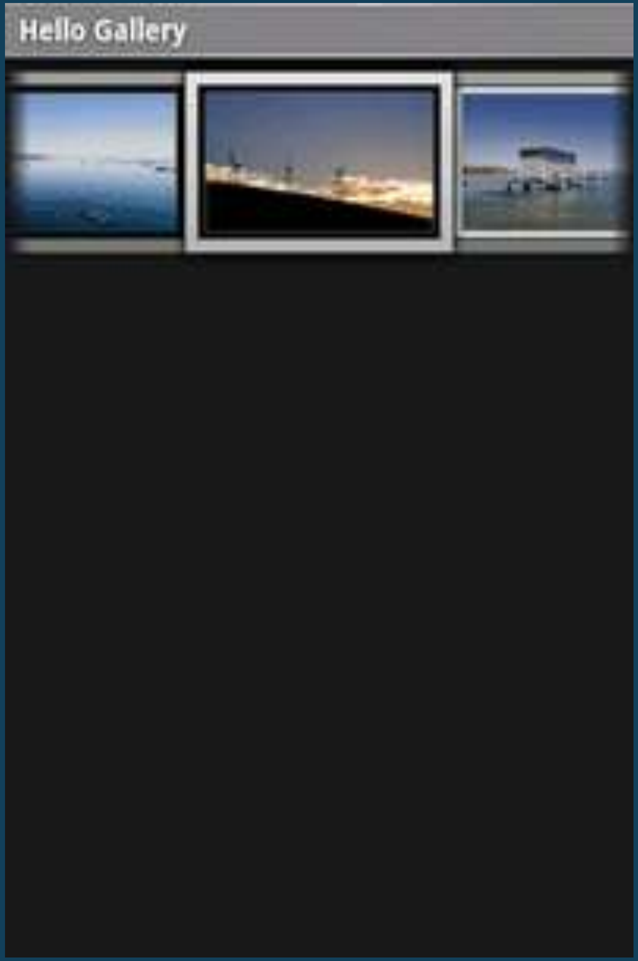
Colombia

Comoros

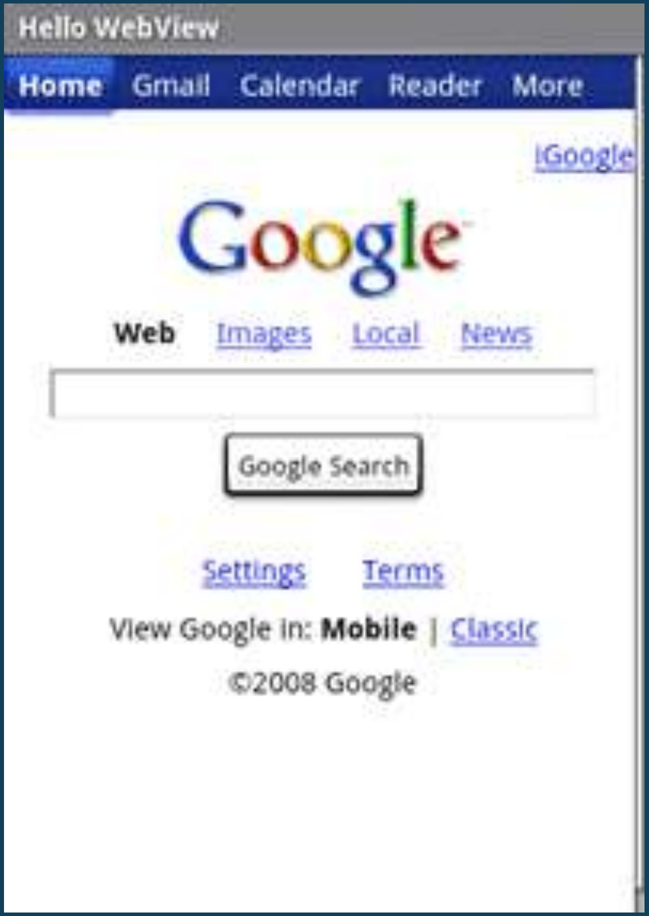
Congo

Views (cont'd)



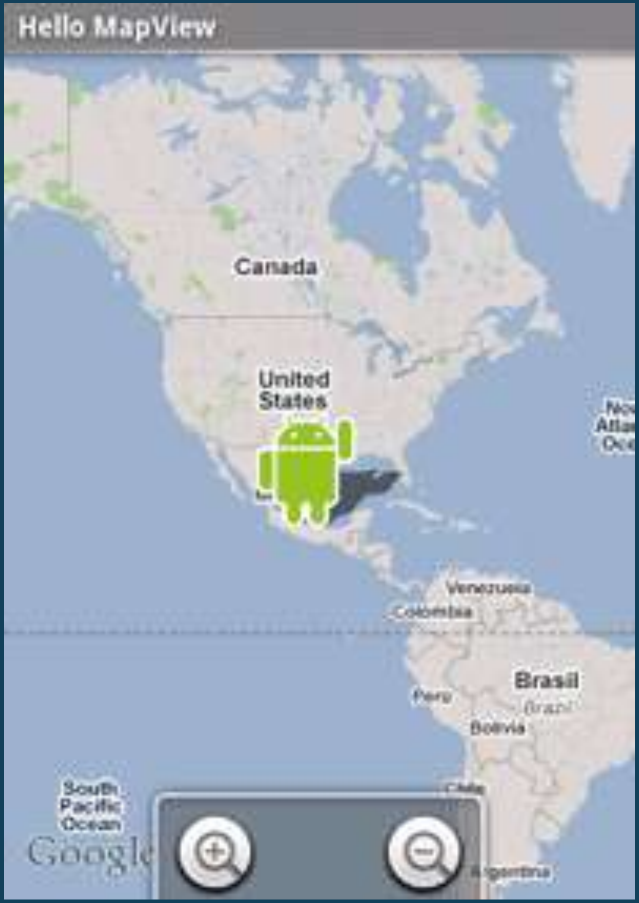


Views (cont’d)



View Google in: **Mobile** | Classic

©2008 Google



Resources

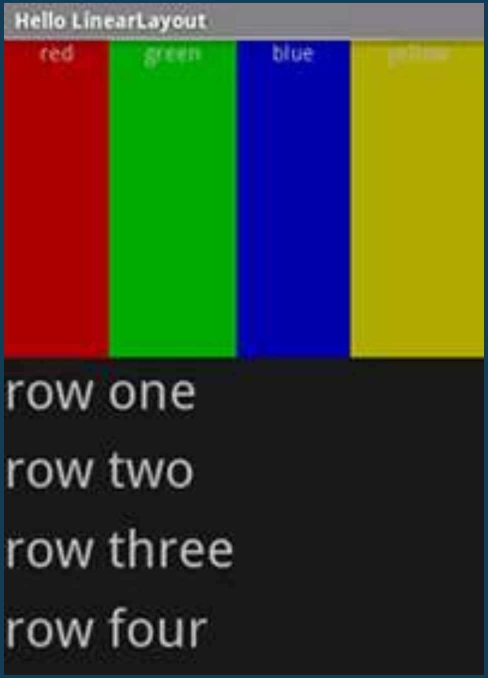
- Some important resource files
  - /res/layout/main.xml
  - /res/layout-land/main.xml
  - /res/values/strings.xml
  - /res/values/colors.xml
  - /res/values/styles.xml
  - /res/menu/menu.xml

Layouts

- Layouts are defined in **/res/layout/main.xml**
- Layouts are automatically converted to a member in the **layout** inner class in **R** class

Layouts (cont'd)

- Linear Layout: Arranges its children in a single column or row. This is the most common layout you will use



Hello LinearLayout

red green blue yellow

row one

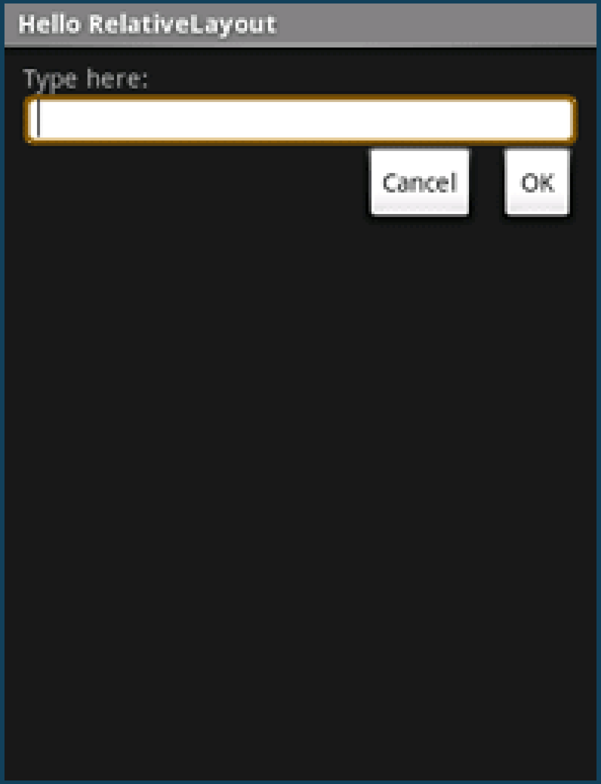
row two

row three

row four

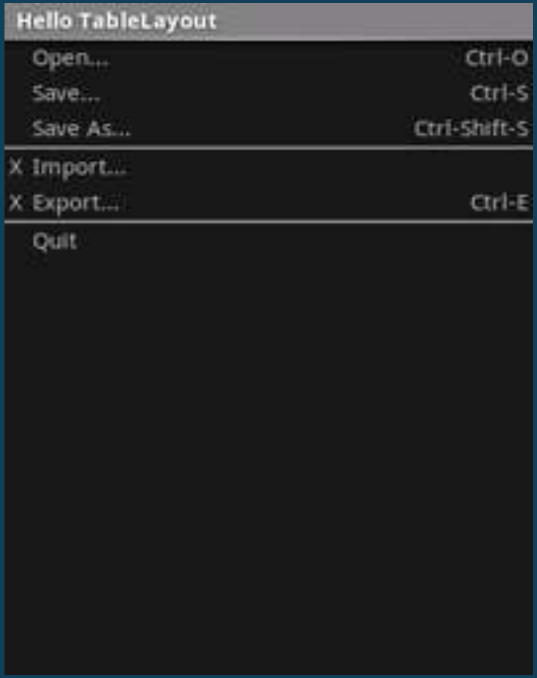
Layouts(cont’d)

- Relative Layout: Arranges its children in relation to each other or to the parent. This is often used in forms



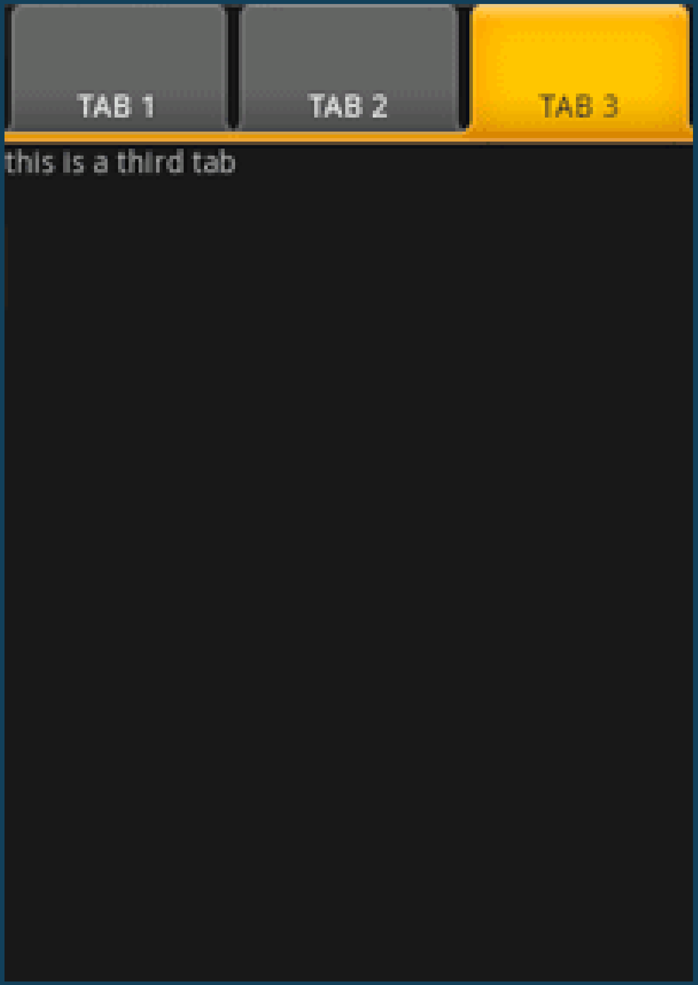
Layouts(cont'd)

- Table Layout: Arranges its children in rows and columns, similar to an HTML table



```
Hello TableLayout
  Open...                         Ctrl-O
  Save...                         Ctrl-S
  Save As...                Ctrl-Shift-S
X Import...
X Export...                       Ctrl-E
  Quit
```

Tab Activity



this is a third tab

Listeners

- Tell Android which object to callback when the user touches or clicks the view

- Use setOnClickListener() method that needs to be passed an object that implements the OnClickListener Java interface

- Set android:onClick property with the method name that handles the click action

Applying a Theme

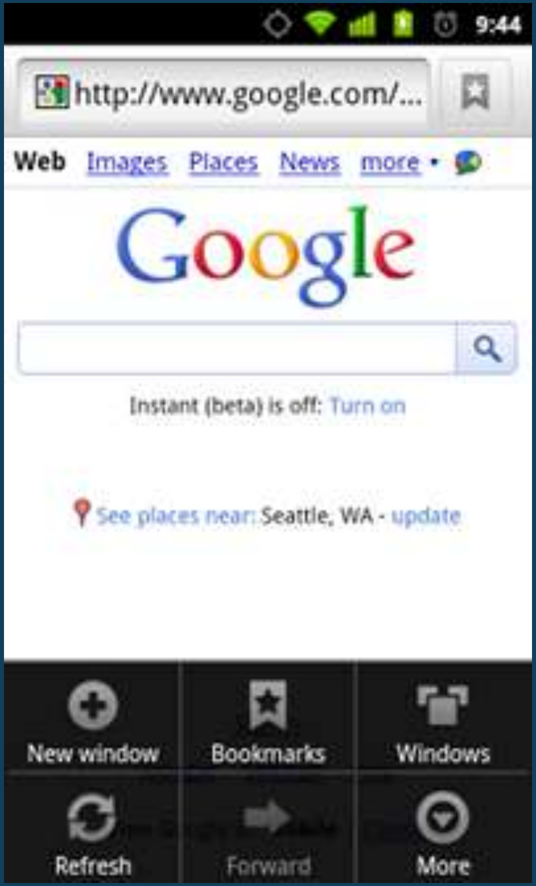
- Android is packaged with several themes that you can reference by name, or you can make up your own theme by extending existing ones and overriding their default values

- You can define your own custom theme in **res/values/styles.xml**

Menus

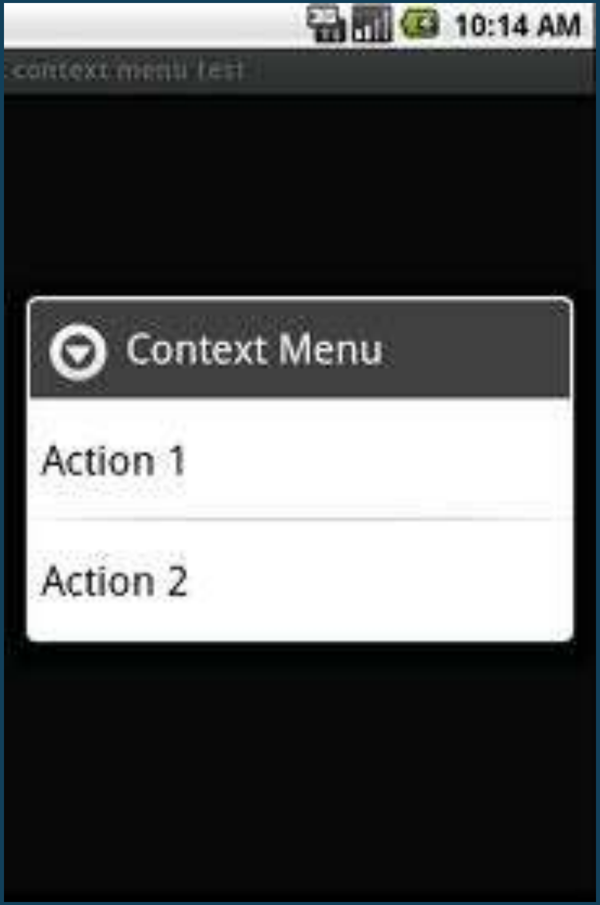
- Android supports three kinds of menus:
  - **Options Menu:** the menu you get when you press the physical Menu button
  - **Context Menu:** that pops up when you press and hold your finger on the screen
  - **Sub Menu:** a floating list of menu items that the user opens by pressing a menu

Menus (cont'd)



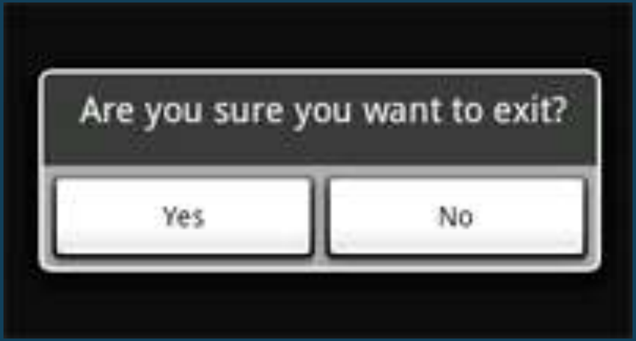
Instant (beta) is off: Turn on

See places near: Seattle, WA - update

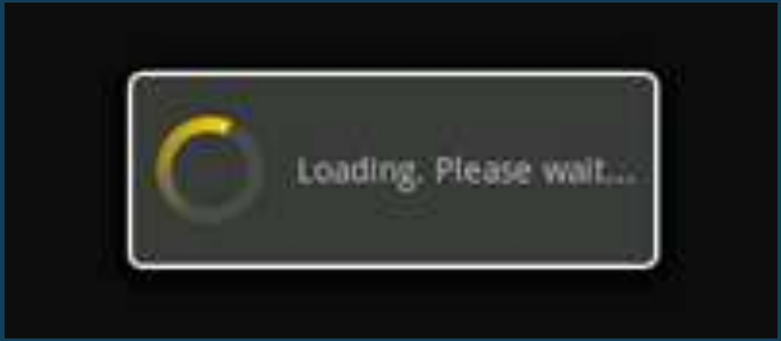


Dialogs

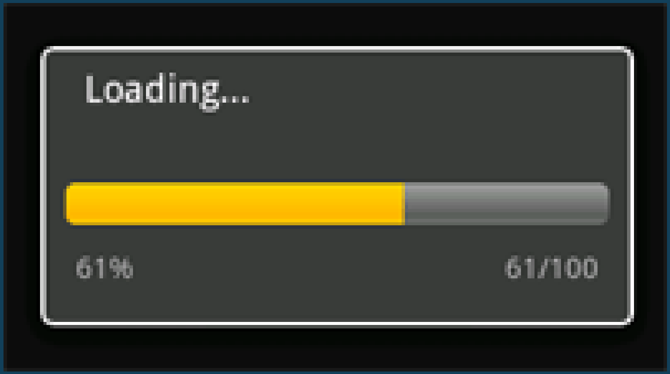
- A small window that appears in front of the current Activity



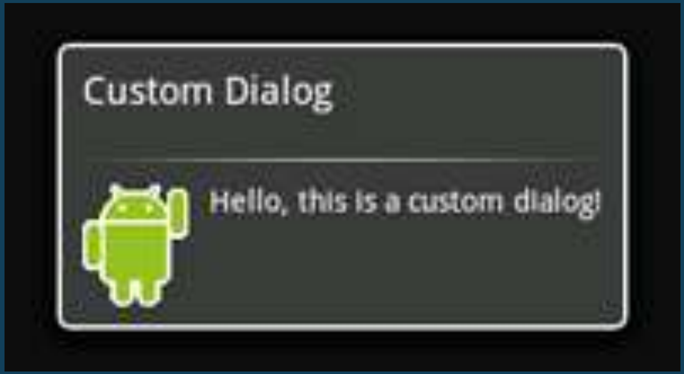
Are you sure you want to exit?



Loading. Please wait...



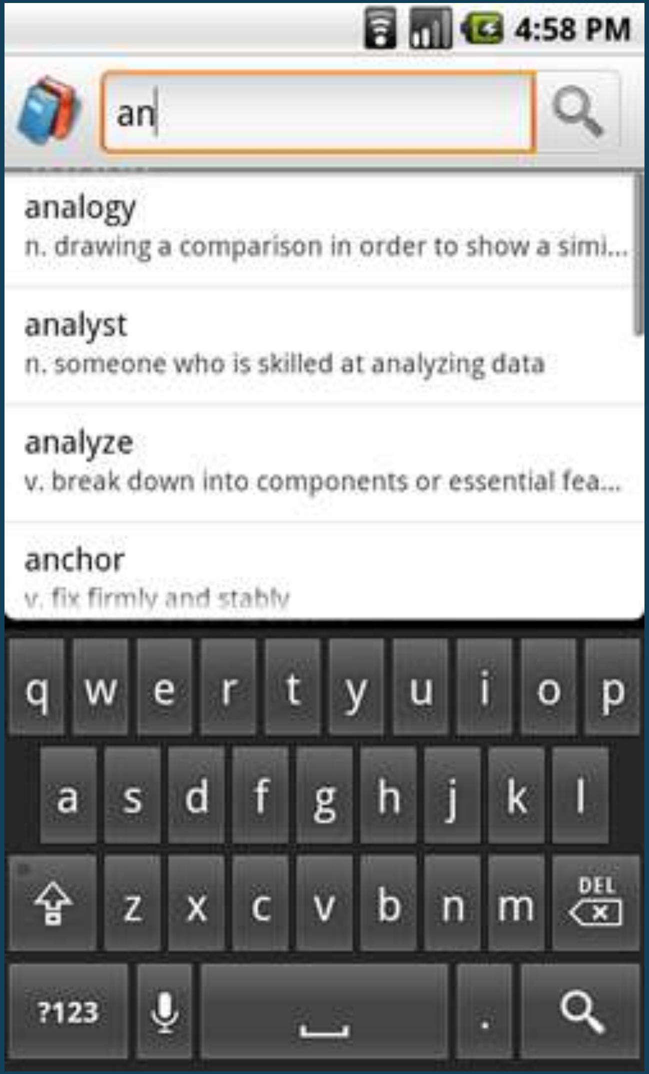
Loading...



Custom Dialog

Hello, this is a custom dialog!

Search Activity



4:58 PM

an

analogy

n. drawing a comparison in order to show a simi...

analyst

n. someone who is skilled at analyzing data

analyze

v. break down into components or essential fea...

anchor

v. fix firmly and stably

GRAPHICS

Overview

- Android provides a powerful graphics library that supports drawing of 2D shapes and developing animations

- 2D Graphics since version 3.0 can also be hardware accelerated

- For 3D Graphics, android provides an implementation based on OpenGL ES 1.0 APIs

2D Graphics

- Android offers a custom 2D graphics library for drawing and animating shapes and images

- The **android.graphics.drawable** and **android.view.animation** packages are where you'll find the common classes used for drawing and animating in two-dimensions

Drawable class

- A **Drawable** is a general abstraction for "something that can be drawn."
- Subclasses include **BitmapDrawable**, **ShapeDrawable**, **PictureDrawable**, etc.
- **draw** method takes a **Canvas** which handles drawing of primitive shapes (Bitmap, rectangle, line, circle, etc.)

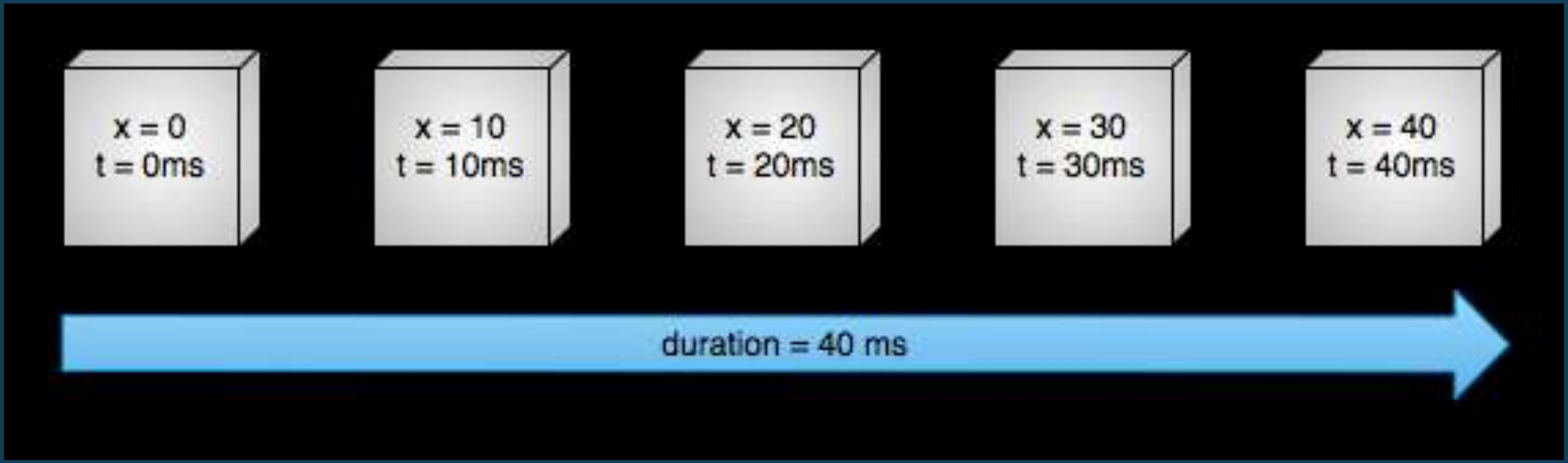
Animations

- Android support 2 animation frameworks:
  - **Property Animation**: latest animation framework that allows developers to animate almost anything
  - **View Animation**: provides the capability to only animate View objects

Property Animation

- Available since version 3.0

- Changes a property's (a field in an object) value over a specified length of time



View Animation

- Tween Animation: can perform a series of simple transformations (position, size, rotation, and transparency) on the contents of a View object
- Frame Animation: a traditional animation in the sense that it is created with a sequence of different images, played in order, like a roll of film

Live Wallpaper

- Introduced in version 2.1
- Like any normal application, can use any feature (MapView, Accelerometer, GPS, …)
- Provides an Engine for handling rendering of Wallpaper
- Provide “settings screen”

MULTIMEDIA

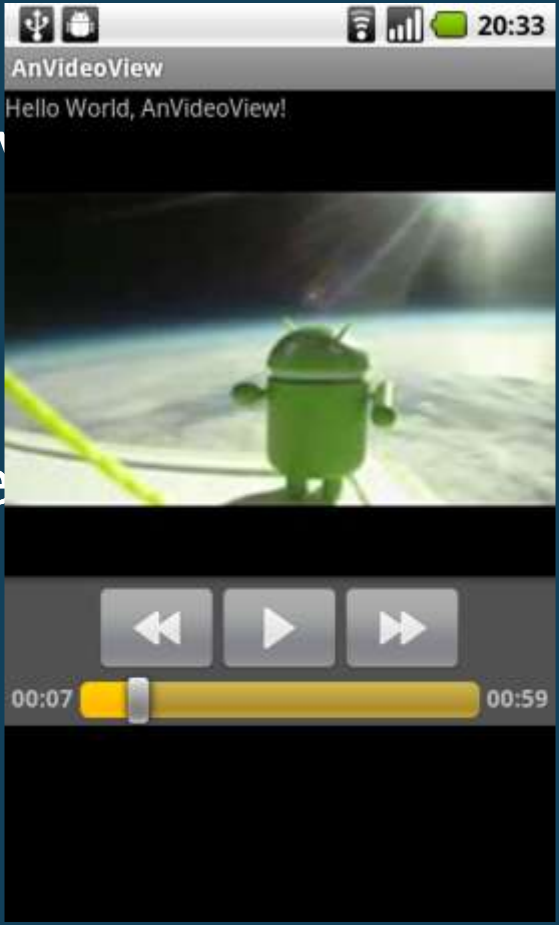
Audio

- Steps for playing Audio:
  1. Put sound files in **res/raw** directory
  2. Create **android.media.MediaPlayer** instance
  3. **mediaPlayer.start()**
     - **stop(), pause(), reset(), prepare(), setLooping(), …**
- Useful methods:
  - **setVolumeControlStream(AudioManager.STREAM_MUSIC)**
  - **setOnCompletionListener( )**

Video

- Exactly similar to Audio
  - MediaPlayer => start(), stop()
  - Just add “**Surface**” to preview the

- Or simply use **VideoView**:
  - video.setVideoPath("/data/sample
  - video.start();



Hello World, AnVideoView!

DATA STORAGE

Preferences

- Out of the box preference screen
- Allows reading and writing application resources
- Preference screen components written in resource XML
- Preference screen loaded from class which extends **PreferenceActivity**

Accessing Internal File System

- Allows access to package private directory created at install time (**/data/data/packagename**)

- Few helper methods are provided on the Context:
  - **deleteFile( )**
  - **fileList( )**
  - **openFileInput( )**
  - **openFileOutput( )**

Accessing SD Card

- Requires **WRITE_EXTERNAL_STORAGE** permission

- Uses **/sdcard/** instead of **/data/**

```
// Load and start the movie
video.setVideoPath("/sdcard/samplevideo.3gp" );
video.start();
```

- Use standard **java.io** to access files

Database access

- Android utilizes **SQLite**
- A SQLite database is just a single file
- Android stores the file in the **/data/data/packagename/databases** directory
- Uses standard SQL DML and DDL scripts

Database access (cont'd)

- DB is accessible through a class that extends **SQLiteOpenHelper**

- Provides an object of **SQLiteDatabase** that exposes methods like:

  - **db.execSQL(String sql)**
  - **db.insert(String tablename, String nullColumnHack, ContentValues *values);***
  - **db.query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)**

Database access (cont’d)

- **query** methods returns an object of **Cursor** class over a result set

- Data binding is possible using **ListActivity**

BREAK

NETWORKING

Checking Network Status

- Available using **ConnectivityManager**

```
ConnectivityManager cMgr =
    (ConnectivityManager)this.getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo netInfo = cMgr.getActiveNetworkInfo();

this.status.setText(netInfo.toString());
```

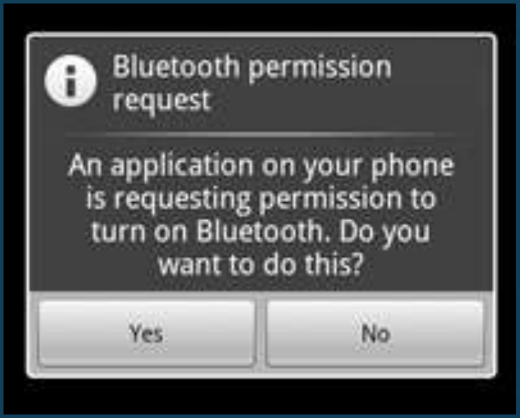
Sockets

- Similar to JSE socket programming

Bluetooth Socket

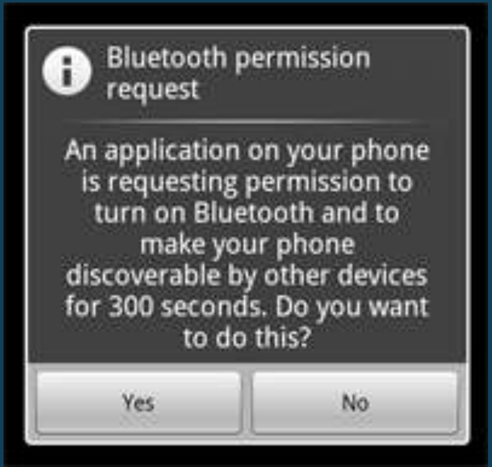
- Requires permission **android.permission.BLUETOOTH**

- Setting up Bluetooth:
  - Enabling Bluetooth
  - Finding Paired Devices
  - Searching for Devices
  - Enabling Discoverability



Bluetooth permission request

An application on your phone is requesting permission to turn on Bluetooth. Do you want to do this?



Bluetooth permission request

An application on your phone is requesting permission to turn on Bluetooth and to make your phone discoverable by other devices for 300 seconds. Do you want to do this?

Bluetooth Socket (cont’d)

- You can connect as a Server using **BluetoothServerSocket**

- You can also connect as a client using **BluetoothDevice** and **BluetoothSocket**

- Connections are managed by **BluetoothSocket** using **InputStream** and **OutputStream**

Working with HTTP

- Similar to JSE using **HttpURLConnection** and **java.net**
- Robust HTTP with HttpClient

```
HttpClient httpclient = new DefaultHttpClient();
 HttpPost httppost = new HttpPost("http://www.website.org/service.php");

List<NameValuePair> pairs = new ArrayList<NameValuePair>(2);
pairs.add(new BasicNameValuePair("ID", "VALUE"));
httppost.setEntity(new UrlEncodedFormEntity(pairs));

HttpResponse webServerAnswer = httpclient.execute(httppost);
```

Working with Web Services

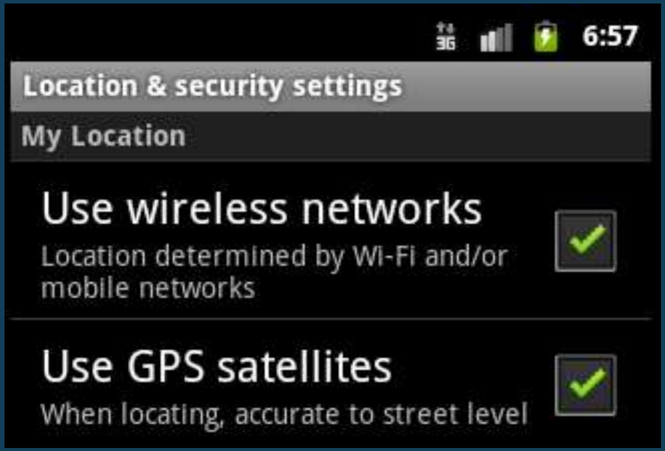
- SOAP Web Services can be invoked using 3rd party library such as **org.ksoap2**

- RESTful Web Service can be implemented using **HttpURLConnection** and XML parser and/or JSON library

LOCATING AND SENSING

Locating Overview

- Supported Providers:
  - GPS
  - Cell Towers
  - WI-FI



6:57

Location & security settings

My Location

Use wireless networks ☑

Location determined by Wi-Fi and/or mobile networks

Use GPS satellites ☑

When locating, accurate to street level

- Access to location information is protected by Android permissions:
  - ACCESS_COARSE_LOCATION
  - ACCESS_FINE_LOCATION

Location Manager

- Provides access to the system location services

- Retrieved through Context.getSystemService(**Context.LOCATION_SERVICE**)

Location Manager(cont’d)

- Useful Methods:
  - getAllProviders()
  - getBestProvider(Criteria criteria, boolean enabledOnly)
  - getLastKnownLocation(String provider)
  - requestLocationUpdates(String provider, long minTime, float minDistance, LocationListener listener)

Location Listener

- Used for receiving notifications from the **LocationManager** when the location is updated

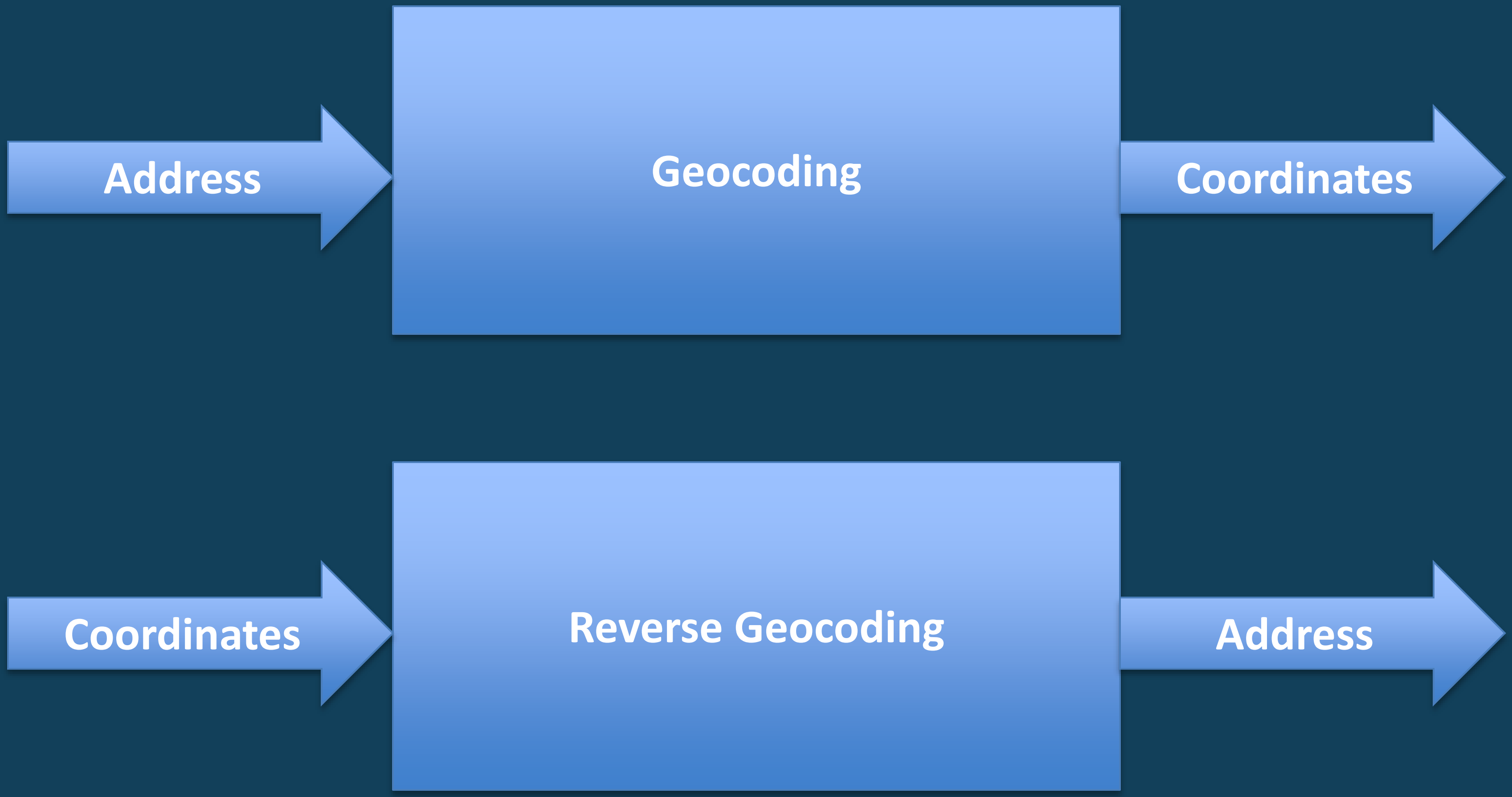
- Location Listener methods:
  - onLocationChanged(Location location)
  - onProviderDisabled(String provider)
  - onProviderEnabled(String provider)
  - onStatusChanged(String provider, int status, Bundle extras)

Geocoding

- The process of finding associated geographic coordinates (often expressed as latitude and longitude) from other geographic data, such as street addresses, or zip codes (postal codes)

- Reverse Geocoding performs the opposite operation

Geocoding (cont’d)



Address

Geocoding

Coordinates

Coordinates

Reverse Geocoding

Address

Geocoder Class

- A class for handling Geocoding and Reverse Geocoding

- Useful methods:
  - **getFromLocation**(double latitude, double longitude, int maxResults)
  - **getFromLocationName**(String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude)
  - **getFromLocationName**(String locationName, int maxResults)

Sensors

- Android supports many different types of sensor devices:
  - **TYPE_ACCELEROMETER**: Measures acceleration in the x-, y-, and z axes
  - **TYPE_LIGHT**: Tells you how bright your surrounding area is
  - **TYPE_MAGNETIC_FIELD**: Returns magnetic attraction in the x-, y-, and z-axes
  - **TYPE_ORIENTATION**: Measures the yaw, pitch, and roll of the device
  - **TYPE_PRESSURE**: Senses the current atmospheric pressure
  - **TYPE_PROXIMITY**: Provides the distance between the sensor and some object
  - **TYPE_TEMPERATURE**: Measures the temperature of the surrounding area

Sensor Manager

- Allows utilizing the device's sensors

- An instance of this class is retrieved by calling **Context.getSystemService(Context. SENSOR_SERVICE)**

- Specific sensors are retrieved using **getDefaultSensor(Sensor.TYPE_ACCELEROMETER**)

SensorEventListener

- Receives notifications from the **SensorManager** when sensor values are updated

- Callback Methods:
  - onAccuracyChanged(Sensor sensor, int accuracy)
  - onSensorChanged(SensorEvent event)

TELEPHONY, MESSAGING AND NOTIFICATIONS

Telephony Manager

- Provides access to information about the telephony services on the device
- Requires **READ_PHONE_STATE** permission
- Get an instance of this class by calling Context.getSystemService(**Context.TELEPHONY_SERVICE**)

Telephony Manager(cont’d)

- **PhoneStateListener** A listener class for monitoring changes in specific telephony states on the device, including service state, signal strength, message waiting indicator (voicemail), and others

SMS Messages Support

- Android API supports developing applications that can send and receive SMS messages

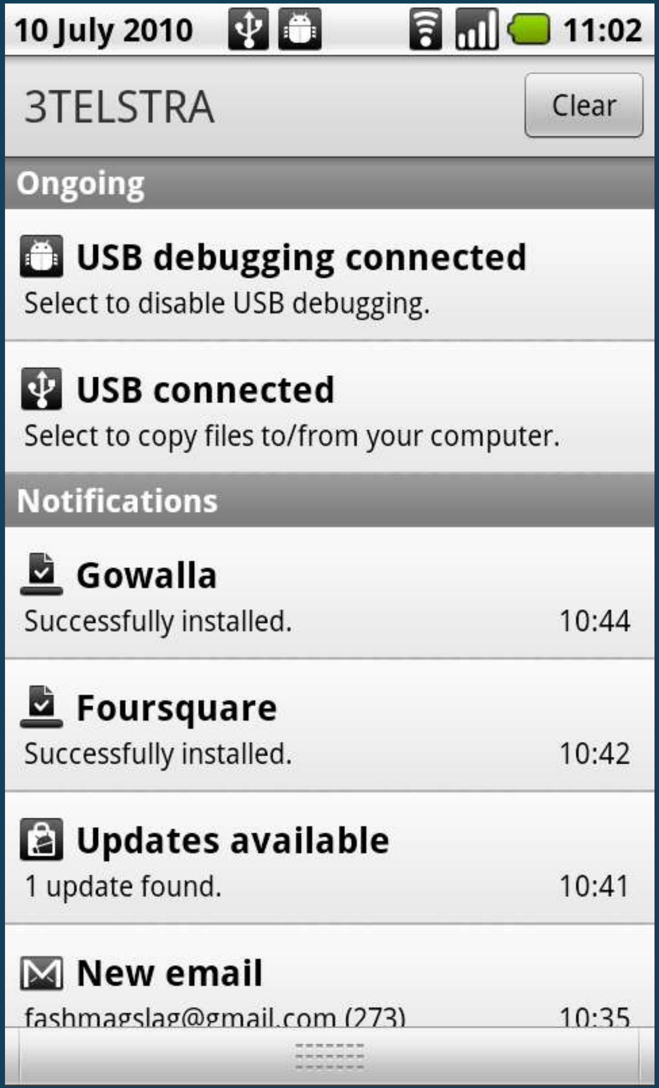
- **SmsManager** Manages SMS operations such as sending data, text, and PDU SMS messages

- Requires **SEND_SMS** permission

Notifications

- A **Notification** is a persistent message that not only shows up in the status bar but stays in a notification area until the user deletes it

- Managed by **Notification** and **NotificationManager** Classes



10 July 2010 11:02

3TELSTRA

Ongoing

USB debugging connected
Select to disable USB debugging.

USB connected
Select to copy files to/from your computer.

Notifications

Gowalla
Successfully installed. 10:44

Foursquare
Successfully installed. 10:42

Updates available
1 update found. 10:41

New email
fashmagslag@gmail.com (273) 10:35

I18N AND LOCALIZATION

Localization

- All resources in Android can be configured to support localization

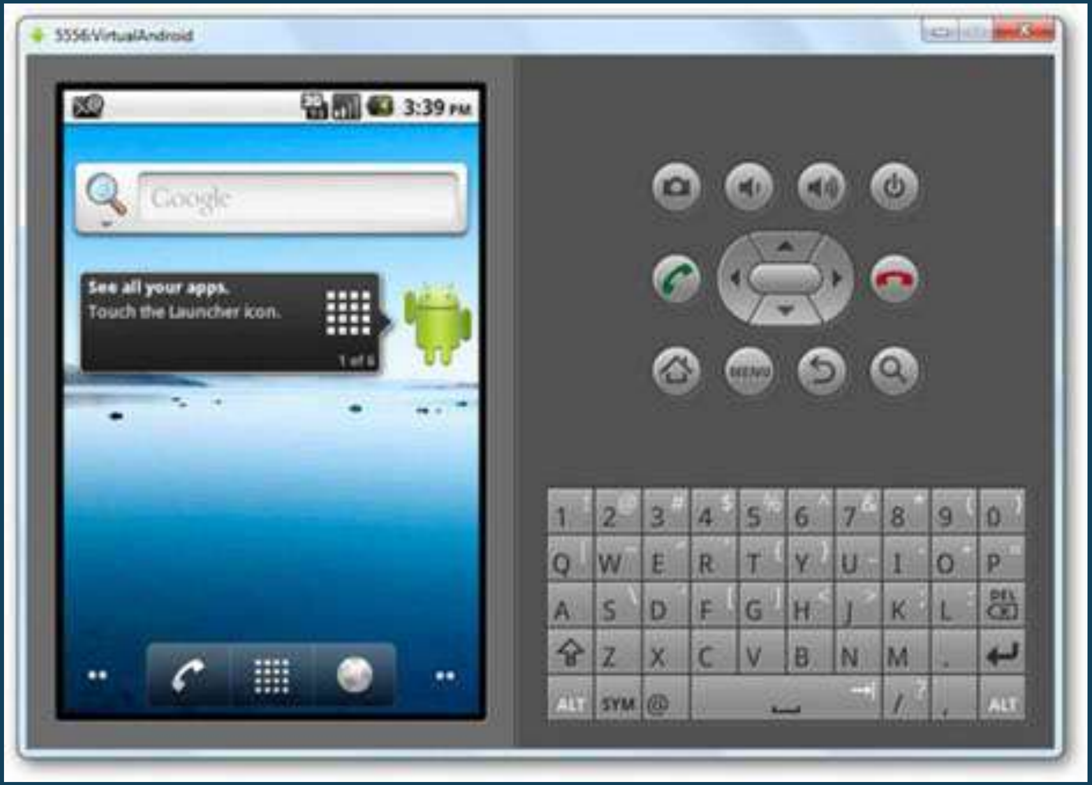
- Example:
  - Default (English): **res/values/strings.xml**
  - Arabic: **res/values-ar/strings.xml**
  - French: **res/values-fr/strings.xml**

- Use Android context to change locale
  - Locale locale = context.getResources().getConfiguration().locale

DEVELOPING AN ANDROID APP

SDK

- Contains Dalvik VM, Java libraries and Emulator

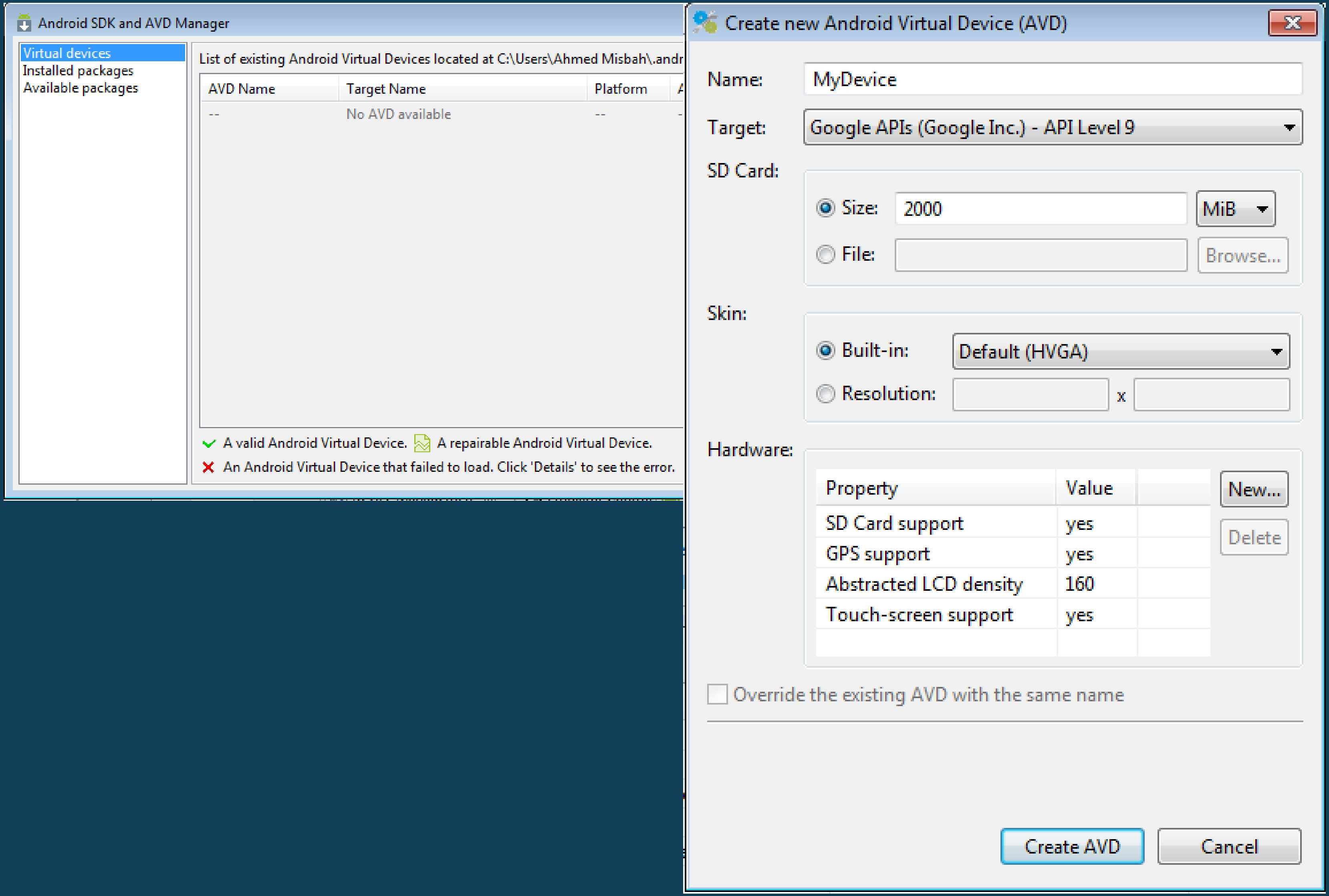


IDE

- An Android plugin, called Android Development Tools (ADT) (https://dl-ssl.google.com/android/eclipse/), is available for Eclipse IDE

- MotoDev is an Eclipse based IDE with tremendous features for Android Development

Create an AVD



Android SDK and AVD Manager

- Virtual devices
- Installed packages
- Available packages

List of existing Android Virtual Devices located at C:\Users\Ahmed Misbah\.andr

AVD Name	Target Name	Platform	A
--	No AVD available	--	-

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Create new Android Virtual Device (AVD)

Name: MyDevice

Target: Google APIs (Google Inc.) - API Level 9

SD Card:

● Size: 2000 MiB

○ File:

Skin:

● Built-in: Default (HVGA)

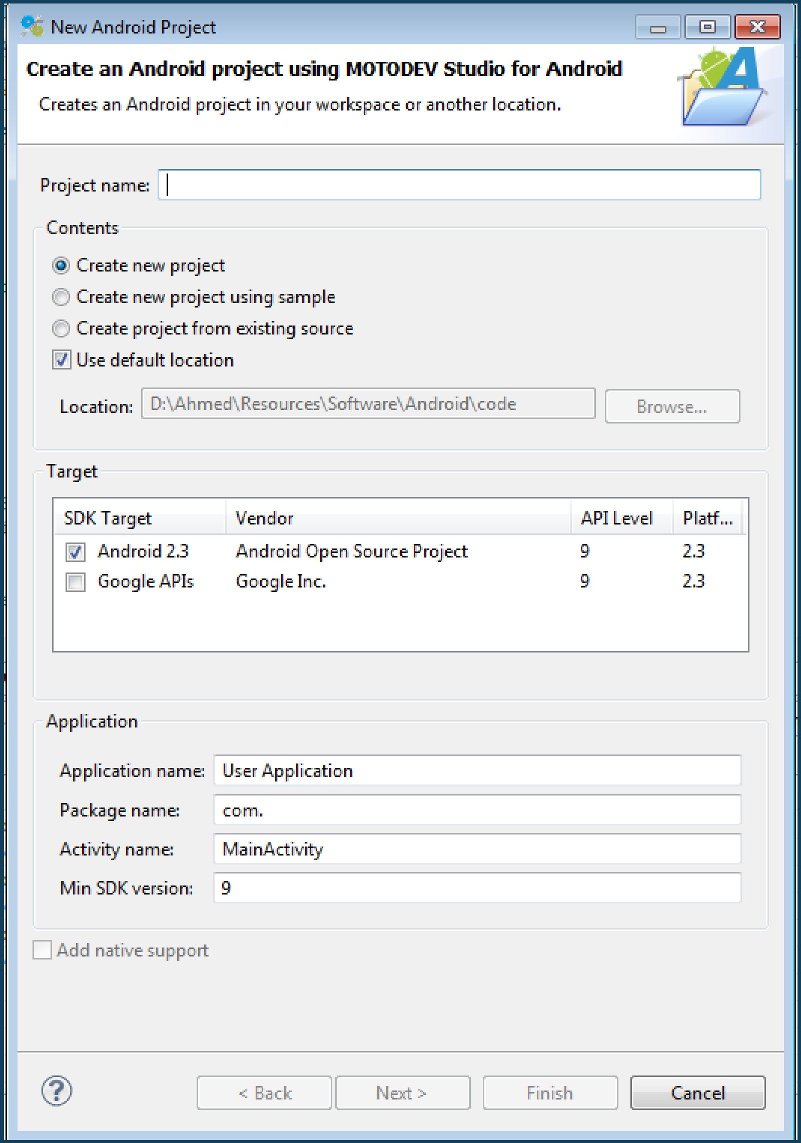
○ Resolution: x

Hardware:

Property	Value
SD Card support	yes
GPS support	yes
Abstracted LCD density	160
Touch-screen support	yes

□ Override the existing AVD with the same name

Create new project



New Android Project

Create an Android project using MOTODEV Studio for Android

Creates an Android project in your workspace or another location.

Project name:

Contents

● Create new project

○ Create new project using sample

○ Create project from existing source

☑ Use default location

Location: D:\Ahmed\Resources\Software\Android\code

Target

SDK Target	Vendor	API Level	Platf...
☑ Android 2.3	Android Open Source Project	9	2.3
□ Google APIs	Google Inc.	9	2.3

Application

Application name: User Application

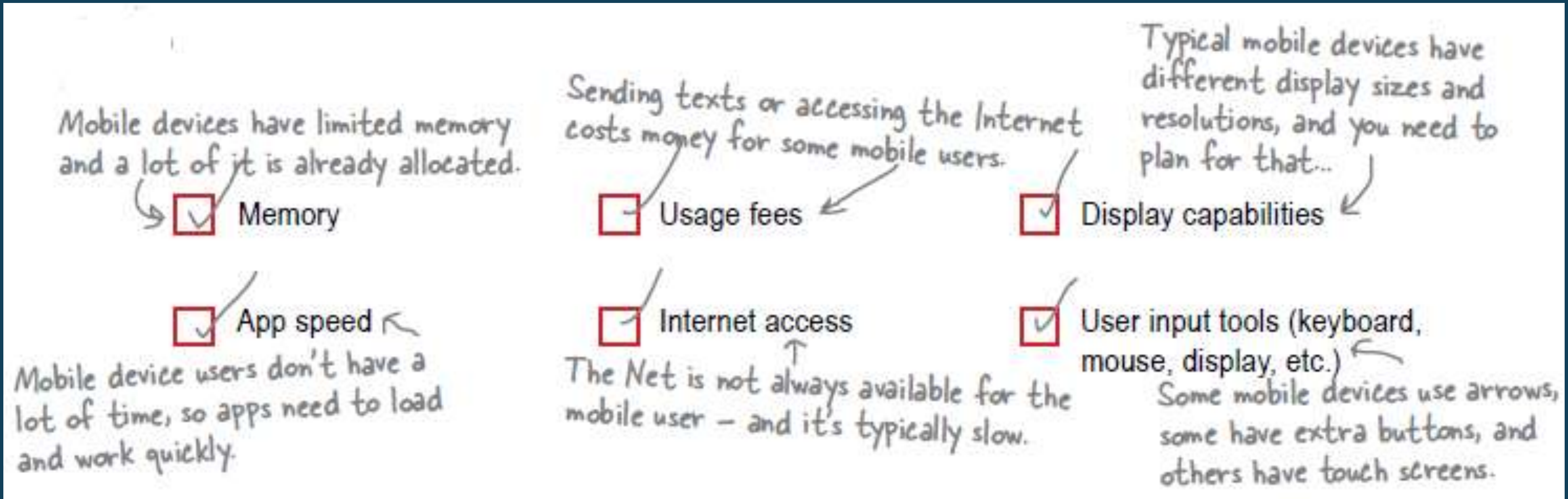
Package name: com.

Activity name: MainActivity

Min SDK version: 9

□ Add native support

Development Checklist



Mobile devices have limited memory and a lot of it is already allocated.

☑ Memory

☑ App speed

Mobile device users don't have a lot of time, so apps need to load and work quickly.

Sending texts or accessing the Internet costs money for some mobile users.

☑ Usage fees

☑ Internet access

The Net is not always available for the mobile user – and it's typically slow.

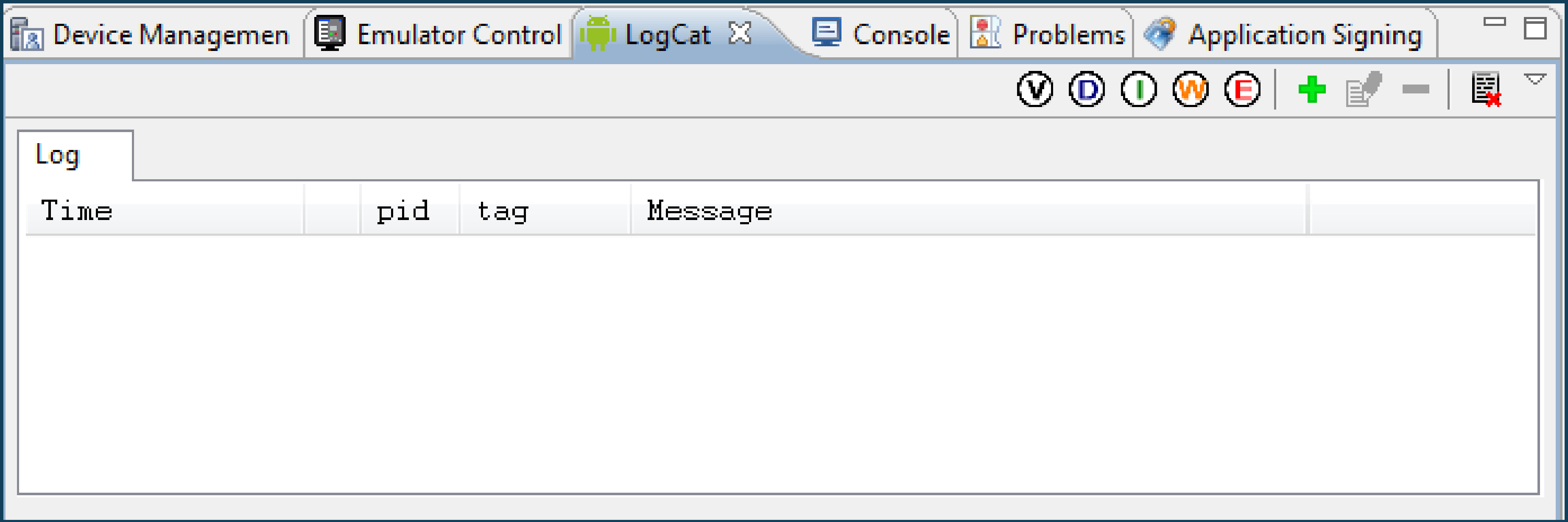
Typical mobile devices have different display sizes and resolutions, and you need to plan for that...

☑ Display capabilities

☑ User input tools (keyboard, mouse, display, etc.)

Some mobile devices use arrows, some have extra buttons, and others have touch screens.

Debugging



Time		pid	tag	Message

Package and deploy

- Sign application using Eclipse Export Wizard
- Choose a strong password to sign your application
- Application is exported to an APK file

Publish to market

- Publishing checklist:
  1. Test your application extensively on an actual device
  2. Consider adding an End User License Agreement in your application
  3. Consider adding licensing support
  4. Specify an icon and label in the application's manifest
  5. Turn off logging and debugging and clean up data/files

Publish to market (cont’d)

6. Version your application
7. Obtain a suitable cryptographic key
8. Register for a Maps API Key, if your application is using MapView elements
9. Sign your application
10. Obfuscate your code using ProGuard

- Follow MotoDev publishing steps

Support and Resources

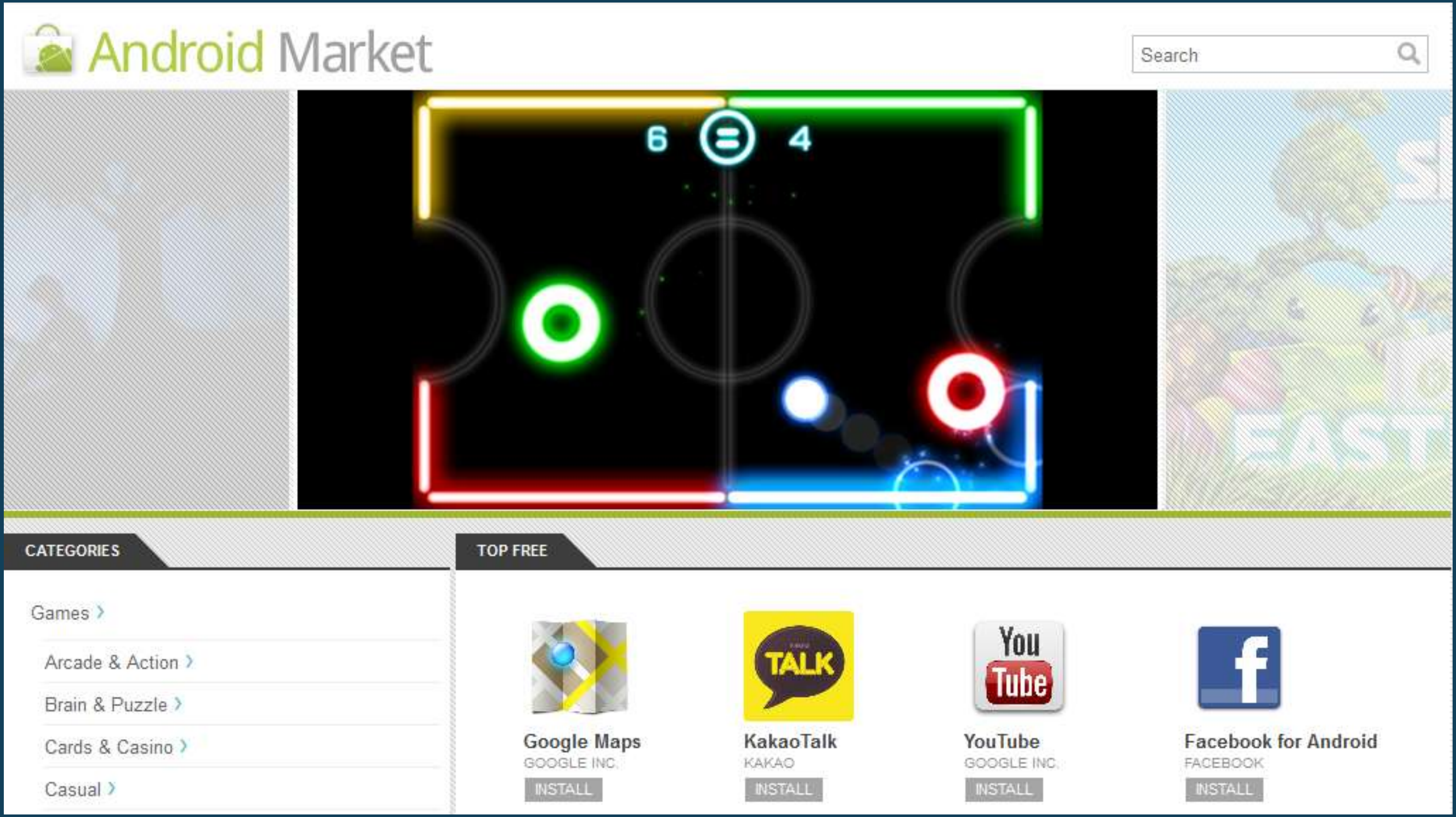
- Android Developers (http://developer.android.com/index.html)
- Offers SDK downloads, Reference (JAVADOCs), Resources and Dev Guide

ANDROID MARKET

Overview

- Android’s application repository
- Similar to Apple’s App Store and Nokia’s Ovi Store
- By August 2010, there were over 80,000 applications available for download, with over 1 billion application downloads

Overview (cont'd)



Android Market

Search

CATEGORIES

Games ›

- Arcade & Action ›
- Brain & Puzzle ›
- Cards & Casino ›
- Casual ›

TOP FREE

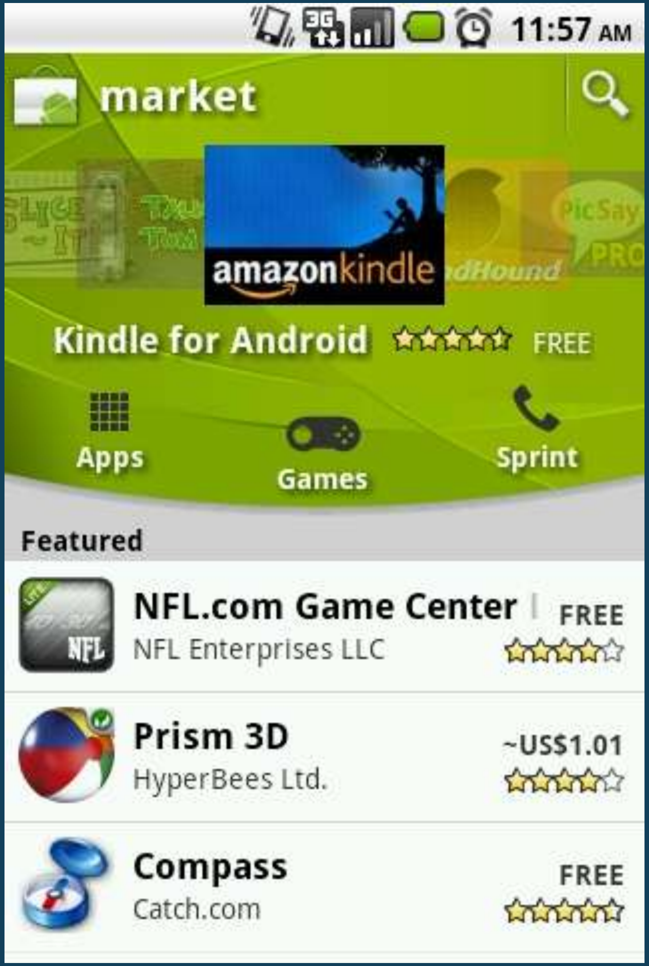
Google Maps
GOOGLE INC.
INSTALL

KakaoTalk
KAKAO
INSTALL

YouTube
GOOGLE INC.
INSTALL

Facebook for Android
FACEBOOK
INSTALL

Overview (cont'd)



Kindle for Android FREE

Featured

NFL.com Game Center FREE
NFL Enterprises LLC

Prism 3D ~US$1.01
HyperBees Ltd.

Compass FREE
Catch.com

Publishing on Android Market

1. Create a developer profile using a Google account

2. Pay a registration fee of 25$

3. For paid applications, Google receives a 30% transaction fee

4. Google handles version updates

ANDROID APPLICATION TRENDS

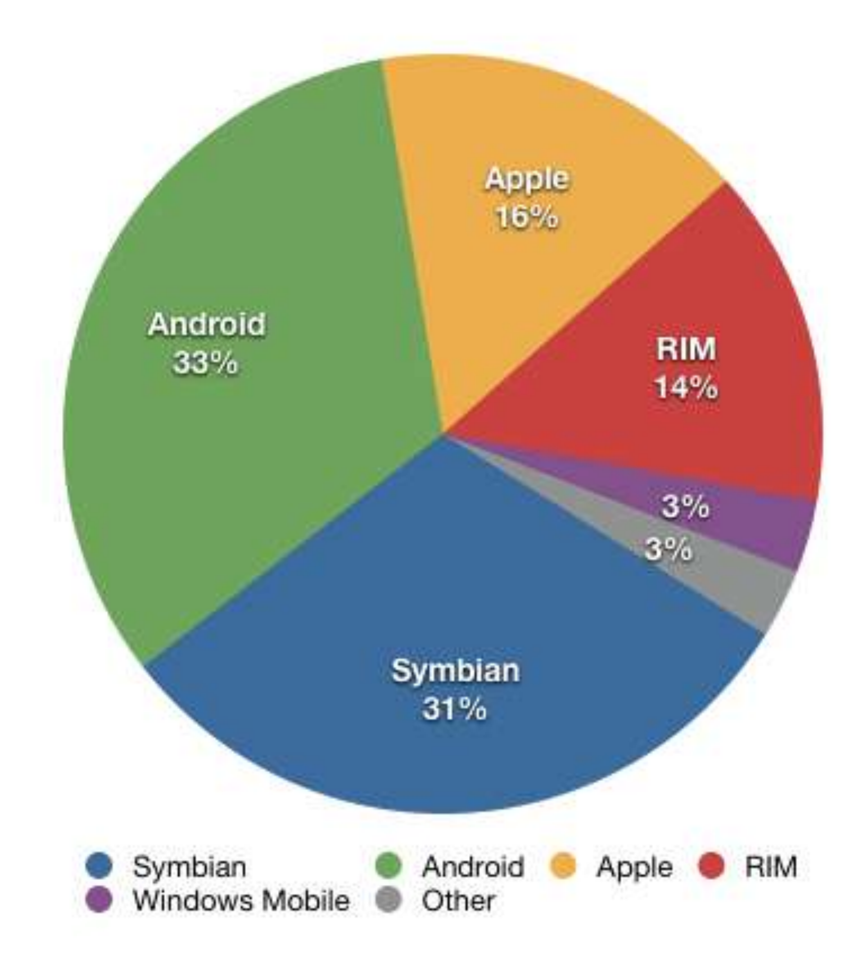
What are analysts saying?

- *"Android Is Destroying Everyone, Especially RIM -- iPhone Dead In Water"* **- Business Insider**

- *"Android market share to near 50 percent"* **- Gartner**

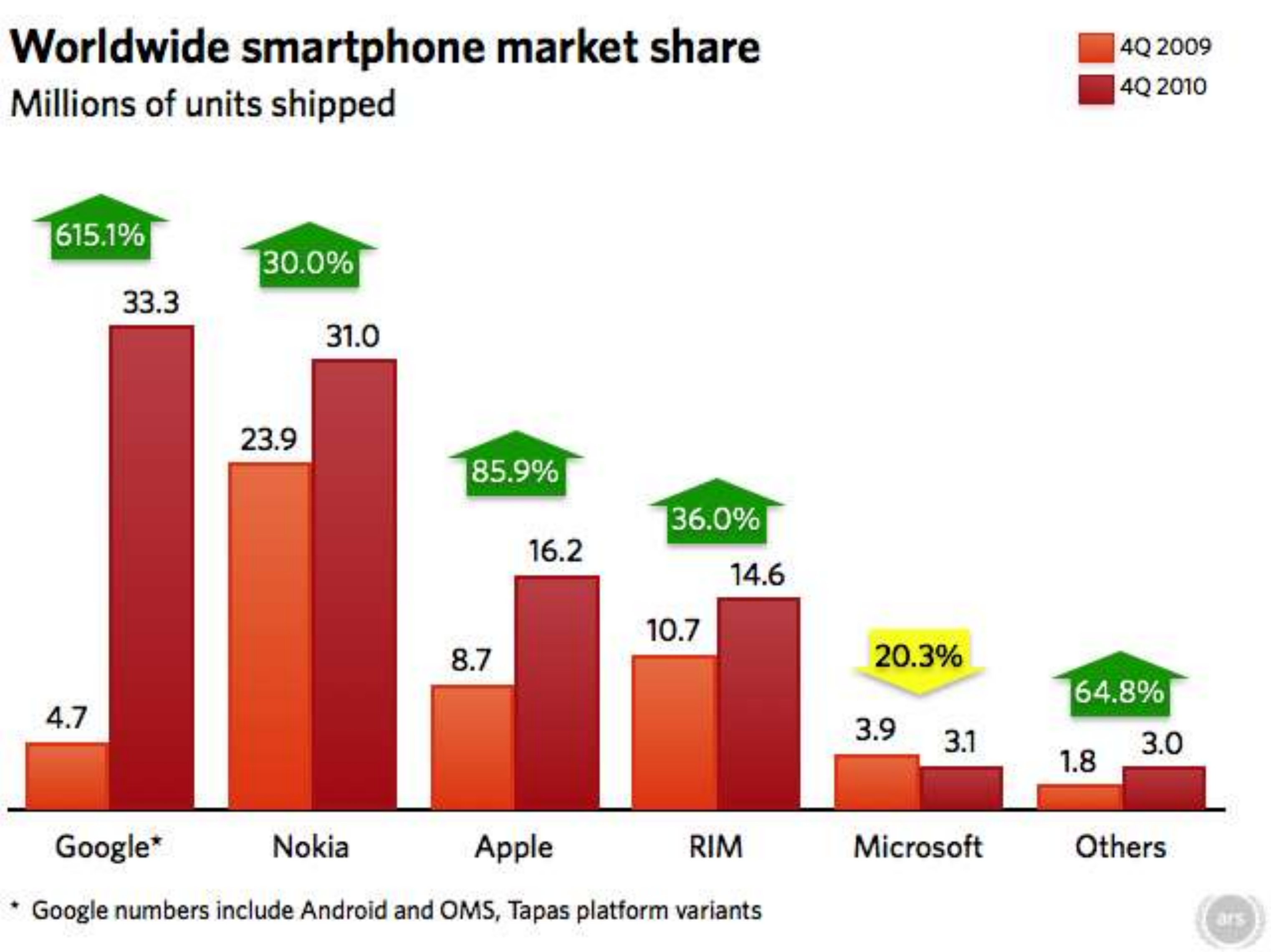
- *"Android's Market Share Soars Ahead Of Apple iPhone's"* **- The Huffington Post**

Market Share



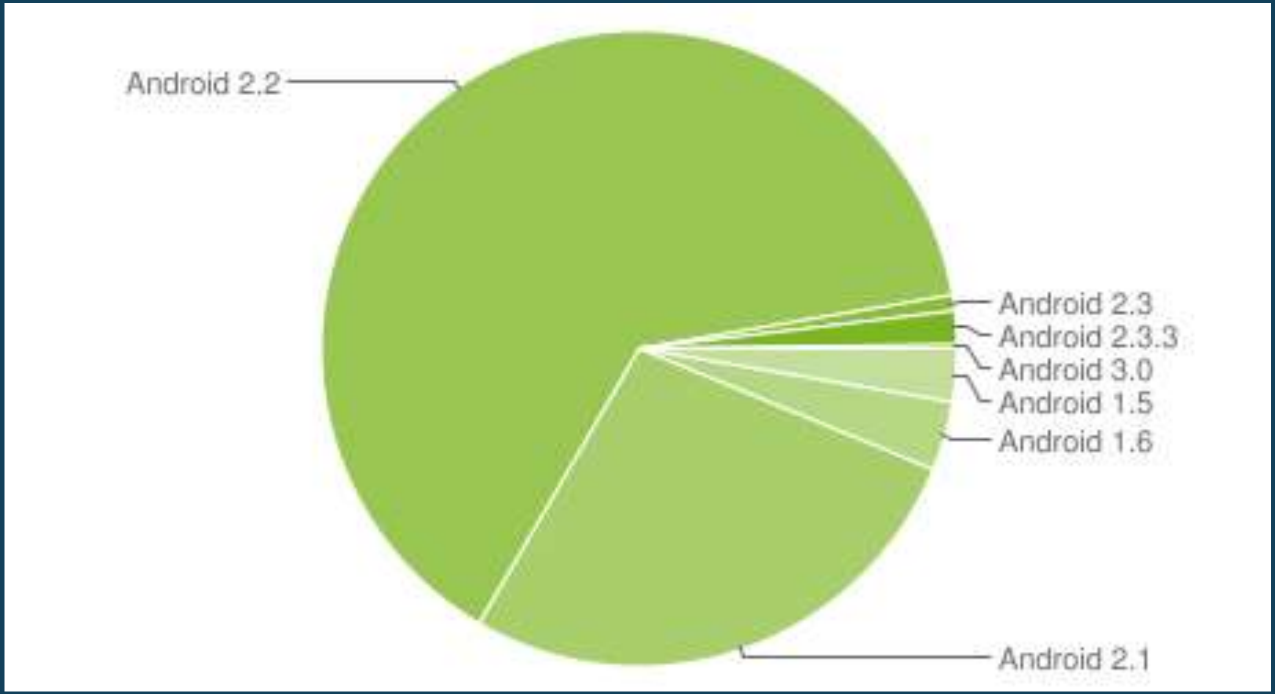
Data collected on Q4 2010

Market Share (cont'd)



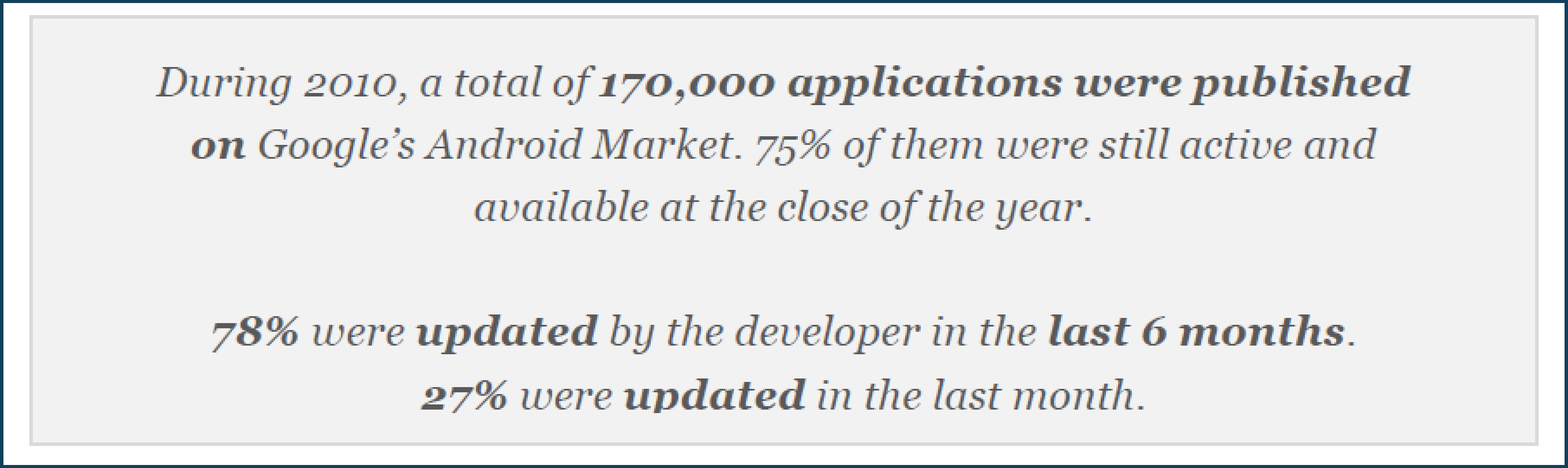
* Google numbers include Android and OMS, Tapas platform variants

Usage Share



Data collected on May 2011

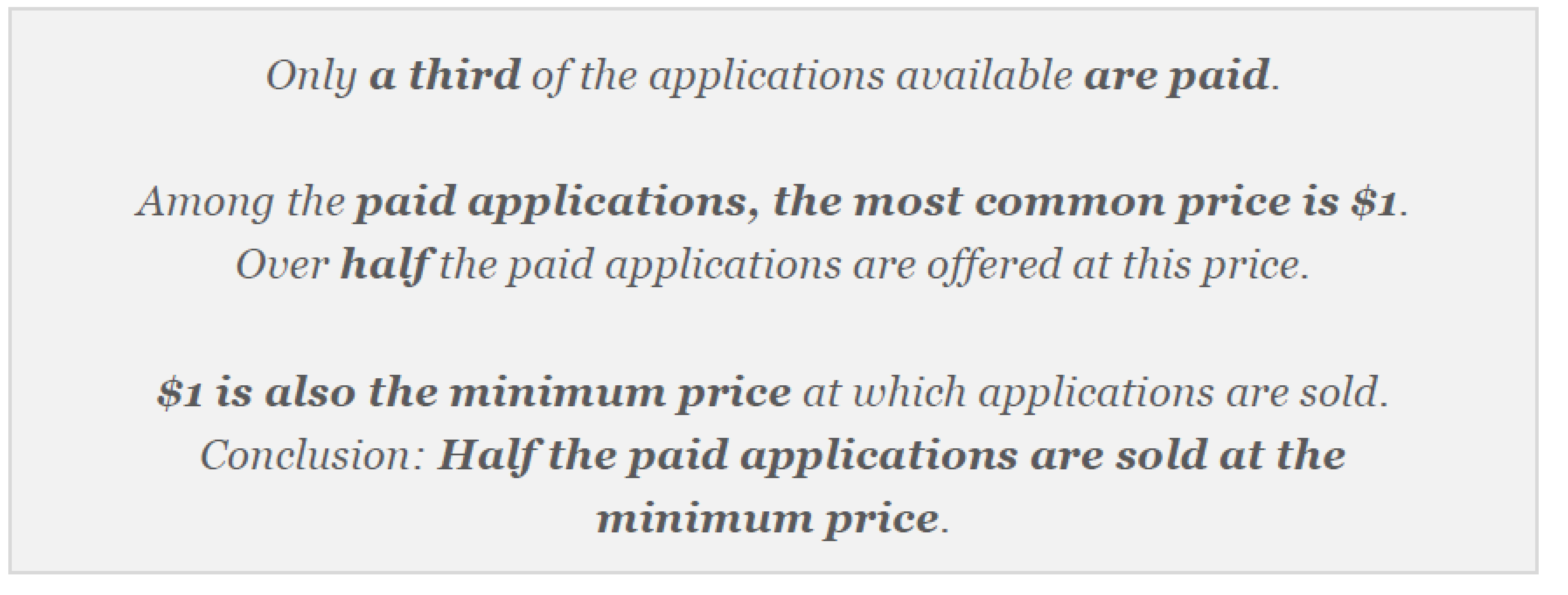
Available Applications



During 2010, a total of ***170,000 applications were published on*** *Google’s Android Market. 75% of them were still active and available at the close of the year.*

78% *were* ***updated*** *by the developer in the* ***last 6 months****.*
27% *were* ***updated*** *in the last month.*

Paid vs. Free

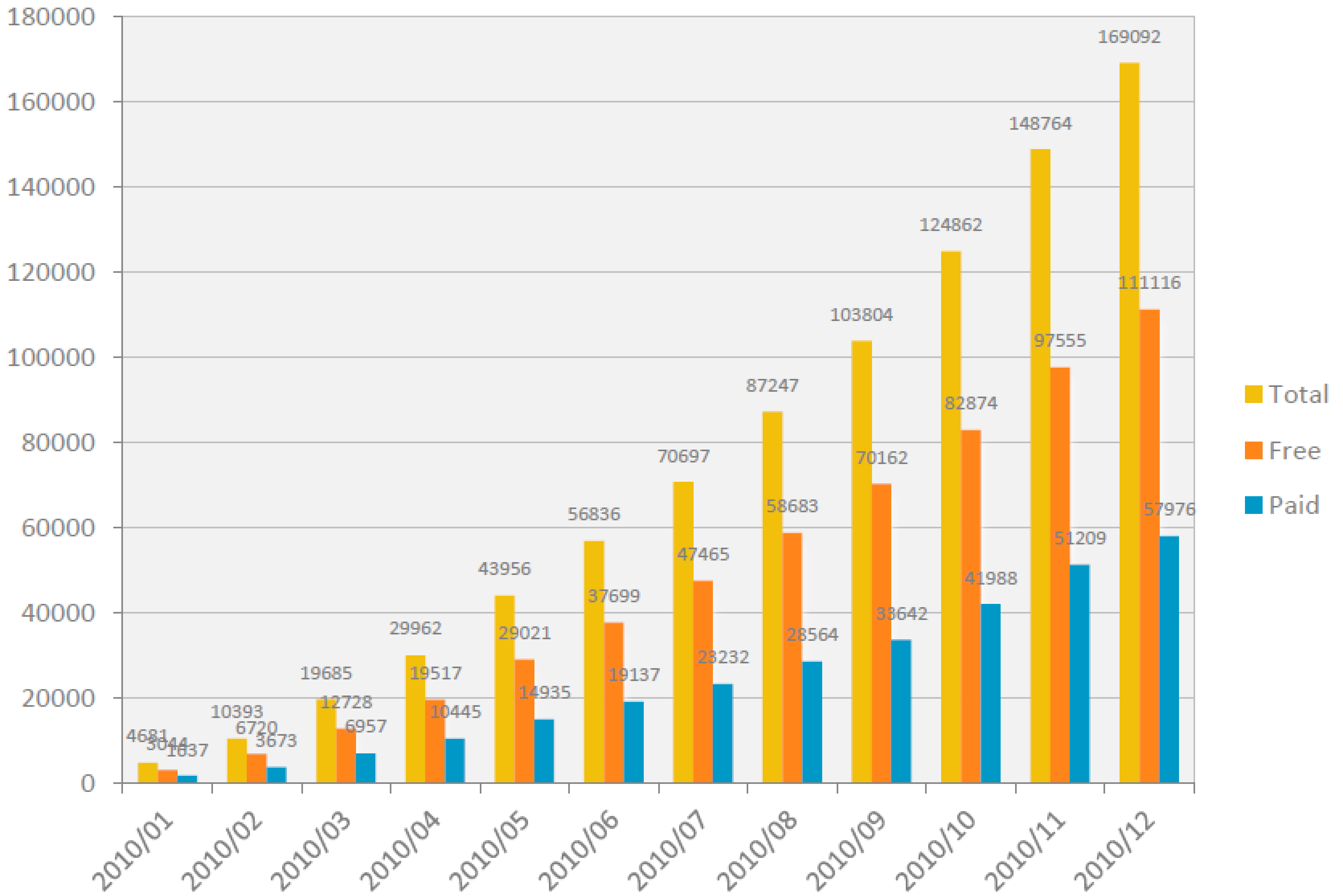


Only **a third** of the applications available **are paid**.

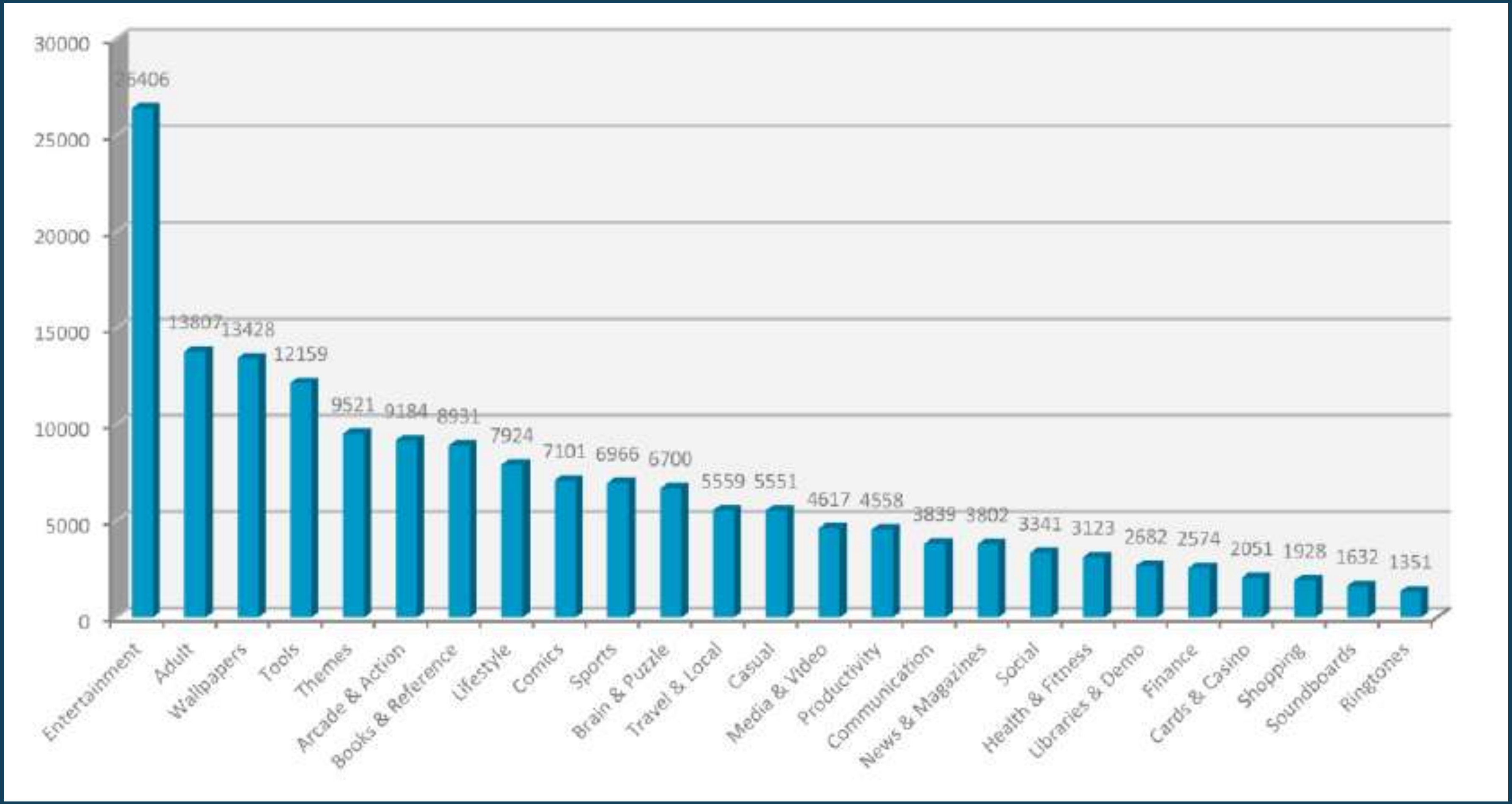
Among the **paid applications, the most common price is $1**.
Over **half** the paid applications are offered at this price.

$1 is also the minimum price at which applications are sold.
Conclusion: **Half the paid applications are sold at the minimum price**.

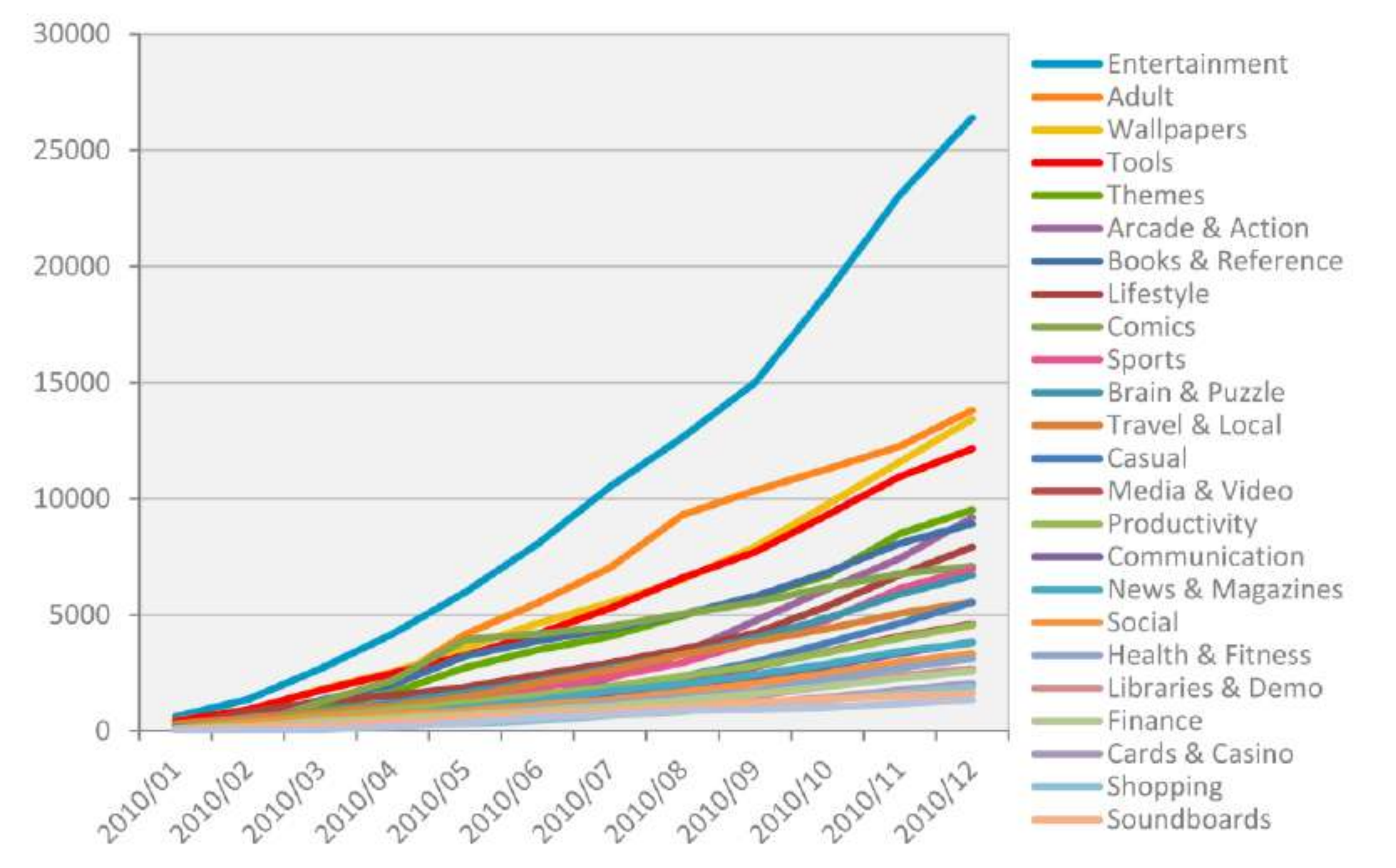
Applications Evolution, 2010



Category Analysis



Category Analysis (cont'd)



Key factors for 2010

- Entertainment category will remain most popular
- Free applications will continue to dominate
- The rise of books and reference categories

Future of Android Apps

- Localized content
- More mature business applications
- Applications for Tablet devices
- Applications utilizing location and maps
- Social Network aggregators
- Satallite Systems (SSTL)
- Software Development process for mobile applications

Gartner Top 10 Mobile Applications for 2012

- Mobile Money Transfer
- Location-Based Services
- Mobile Search
- Mobile Browsing
- Mobile Health Monitoring
- Mobile Payment
- Near Field Communication Services
- Mobile Advertising
- Mobile Instant Messaging
- Mobile Music

Thank you