Sviluppo di Applicazioni Mobili

Vincenzo Gervasi

Dipartimento di Informatica

Email: gervasi@di.unipi.it
Sito web: http://www.di.unipi.it/~gervasi
Ufficio: Stanza 305, Dipartimento di Informatica

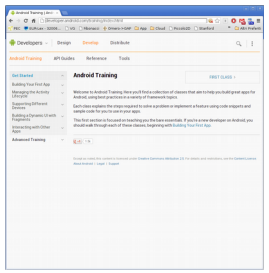
Logistica del corso

- 6 CFU – 2º semestre
- INF-L complementare (3° anno)
- Orario delle lezioni:
  - Martedì 14:00-16:00 aula Fib A1
  - Giovedì 14:00-16:00 aula Fib C
- Orario di ricevimento:
  - Da fissare: forse Martedì 16:00-17:00, stanza 305 (Dipartimento di Informatica)?
  - Negoziabile!

Testi raccomandati

- **Reto Meier**, *Professional Android 4|2|ε Application Development*, Wrox Publishing, 2012|2010|2008

- **E. Di Saverio, S. Sanna**, *Android. Programmazione avanzata*, Edizioni FAG, 2012



- Tutorial online: http://developer.android.com/training

Programma di massima del corso

1. Introduzione, storia del mercato mobile, storia di Android.
2. Architettura di Android; rapporto con Linux, visione a strati
3. Dalvik VM, ambiente di sviluppo, deploy di applicazioni e Market
4. Il sistema delle risorse e degli asset; dispatching a runtime
5. Activity e ciclo di vita; il dispatching degli Intent; Layout e View; scrivere una custom View
6. Listview e DataAdapter; dialog, notifiche e alert
7. Drawable e sue sottoclassi; approfondimenti su 9patch
8. Tematiche di storage: Bundle e Parcelable; preferenze; file system; caching; SQLlite e classi helper; ContentProvider e ContentResolver
9. Services
10. Broadcast receiver e servizi di sistema (telefonia, sensori, ecc.)
11. Esecuzione asincrona e in background
12. Programmazione nativa in C

Modalità d'esame

- Sviluppo di una app
  - Tema proposto dallo studente
  - Dettagli concordati in anticipo con il docente
- Esame orale consistente in
  - Presentazione della app
  - Ispezione del codice
  - Domande “di teoria” su aspetti non coperti nel progetto
- Non sono previsti “compitini” o altre attività di verifica intermedia

Lezione 1

20 Febbraio 2018

Programmazione Android

- Breve storia di Android
- Ambienti di sviluppo
  - Eclipse + ADT
  - Android Studio
  - Dettagli sull'installazione
- Architettura di un sistema Android
  - Kernel
  - La macchina virtuale
  - Librerie e Framework

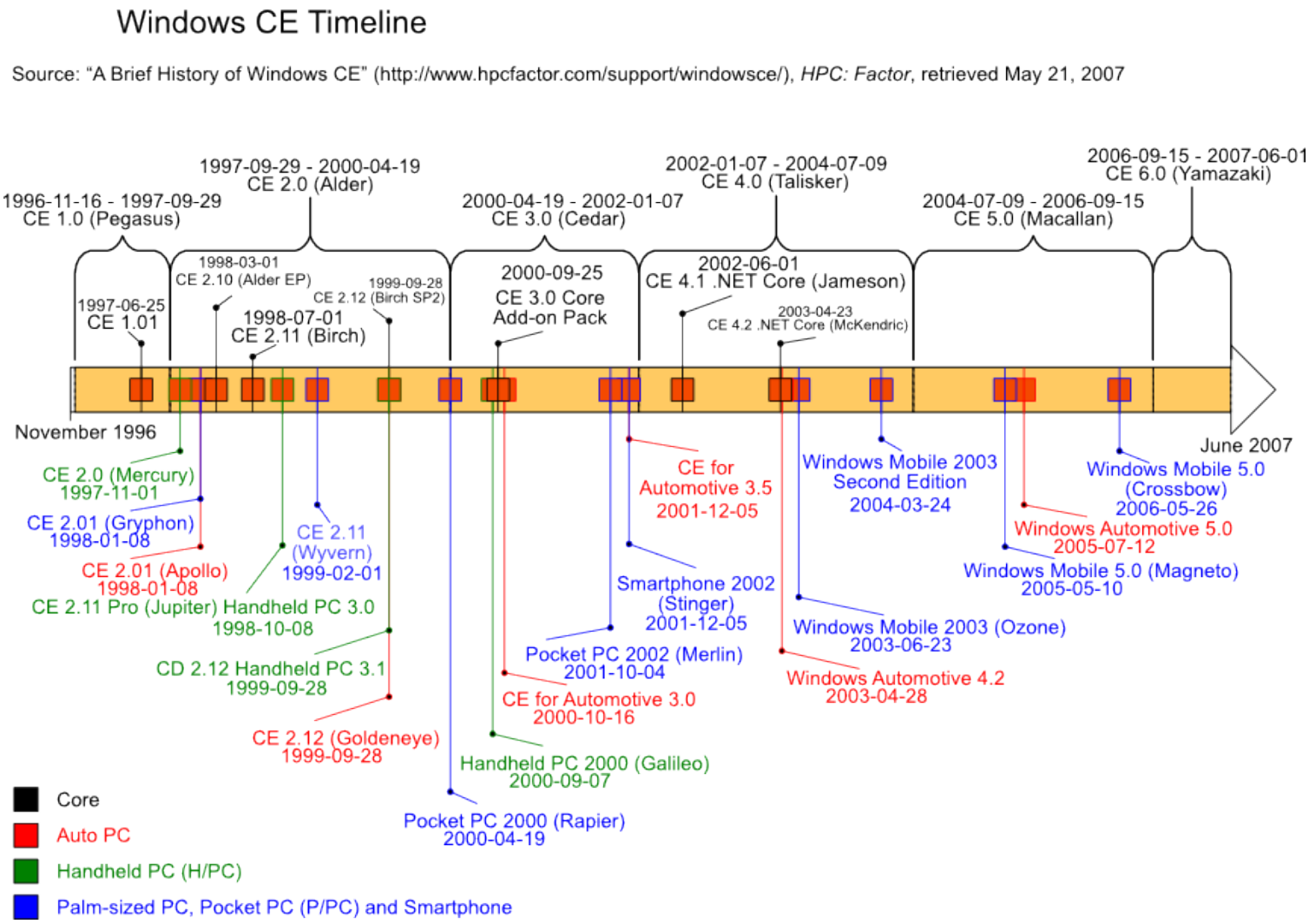
Breve storia di Android

Breve storia di Android
Episode I

- Rewind al 2007
  - Palm († 2006)
  - Windows CE (1996-2011)
  - Blackberry (1999-~~vivente~~)

Breve storia di Android Episode I

- Dilemma per i produttori
- Tutti sistemi **fortemente proprietari**
- JavaME?
  - Portabile
  - Molto limitato



Breve storia di Android Episode I

- Novembre 2007: un gruppo di **produttori di telefoni** forma la **Open Handset Alliance**

Industry Leaders Announce Open Platform for Mobile Devices

November 5, 2007

Group Pledges to Unleash Innovation for Mobile Users Worldwide

MOUNTAIN VIEW, Calif.; BONN, Germany; TAOYUAN, Taiwan; SAN DIEGO, Calif.; SCHAUMBERG, Ill., November 5, 2007 — A broad alliance of leading technology and wireless companies today joined forces to announce the development of Android, the first truly open and comprehensive platform for mobile devices. Google Inc., T-Mobile, HTC, Qualcomm, Motorola and others have collaborated on the development of Android through the Open Handset Alliance, a multinational alliance of technology and mobile industry leaders.

Open Handset Alliance Founding Members

Aplix (www.aplixcorp.com), Ascender Corporation (www.ascendercorp.com), Audience (www.audience.com), Broadcom (www.broadcom.com), **China Mobile** (www.chinamobile.com), **eBay** (www.ebay.com), Esmertec (www.esmertec.com), **Google** (www.google.com), **HTC** (www.htc.com), **Intel** (www.intel.com), KDDI (www.kddi.com), Living Image (www.livingimage.jp), **LG** (www.lge.com), Marvell (www.marvell.com), **Motorola** (www.motorola.com), NMS Communications (www.nmscommunications.com), Noser (www.noser.com), **NTT DoCoMo, Inc.** (www.nttdocomo.com), Nuance (www.nuance.com), **Nvidia** (www.nvidia.com), PacketVideo (www.packetvideo.com), **Qualcomm** (www.qualcomm.com), **Samsung** (www.samsung.com), SiRF (www.sirf.com), SkyPop (www.skypop.com), SONiVOX (www.sonivoxrocks.com), **Sprint Nextel** (www.sprint.com), Synaptics (www.synaptics.com), TAT - The Astonishing Tribe (www.tat.se), **Telecom Italia** (www.telecomitalia.com), **Telefónica** (www.telefonica.es), **Texas Instruments** (www.ti.com), **T-Mobile** (www.t-mobile.com), Wind River (www.windriver.com)

Breve storia di Android Episode I

Open Handset Alliance Releases Android SDK

November 12, 2007

The Open Handset Alliance, a group of mobile and technology leaders, today announced the availability of the Android Software Development Kit (SDK). Available now as an early look, the Android SDK will enable developers to create innovative and compelling applications for the platform. The early look will also provide developers with the opportunity to participate in the evolution of the Android platform by providing feedback throughout the development process.

The Android platform was built from the ground up to enable developers to create new and innovative mobile applications that take full advantage of all the capabilities of a handset connected to the internet. It is a complete mobile platform built on the Linux 2.6 kernel that exposes a robust operating system, a comprehensive set of libraries, a rich multimedia user interface, and a complete set of phone applications. Android's innovative application model makes it easy for developers to extend, replace, and reuse existing software components to create rich and integrated mobile services for consumers.

The Android platform also includes the Dalvik virtual machine to maximize application performance, portability, and security. The entire platform will be made available under the very liberal, developer-friendly Apache v2 open-source license in 2008.

Android Software Development Kit

The SDK contains a rich set of tools for developers to build applications for the Android platform. Included are advanced development and debugging tools, a rich set of libraries, a true device emulator, in-depth documentation, sample projects, tutorials, FAQs, and more. For developers looking for a seamless development experience, an Eclipse plugin is included to integrate these tools with the Eclipse integrated development environment. The site hosting the kit will also feature a blog and discussion groups, to make it easier for everyone contributing to the platform to interact and share knowledge.

Requirements

To begin building applications for Android, developers will need to download the Android SDK to an x86-based computer running Windows XP or Vista; Mac OS 10.4.8 or later; or Linux Ubuntu Dapper Drake or later (other modern distributions of Linux will also likely work but are not directly supported).

Developers will also need Eclipse 3.2 or later, with Java Development Tools and the Android SDK's plugin, or Java and Javac 1.5 or 1.6; Apache Ant; an integrated development environment; and Python 2.2 or later.

- 7 giorni dopo, viene rilasciato il primo SDK
  - Licenza Apache
- Basato su
  - Linux 2.6
- Sviluppo su
  - Eclipse
  - Java
  - Python (!)

Breve storia di Android Episode I

- In realtà, il software era stato già sviluppato
- Da Android Inc., la classica startup californiana
  - Nata nel 2003 a Palo Alto
  - Acquistata da Google nel 2005
  - Brevetti registrati nel 2007
- Quasi tutti i giochi erano già fatti a fine 2007
  - Architettura complessiva
  - Ambiente di sviluppo
  - Licensing
  - Partner (telefoni e carrier)

Breve storia di Android
Episode I

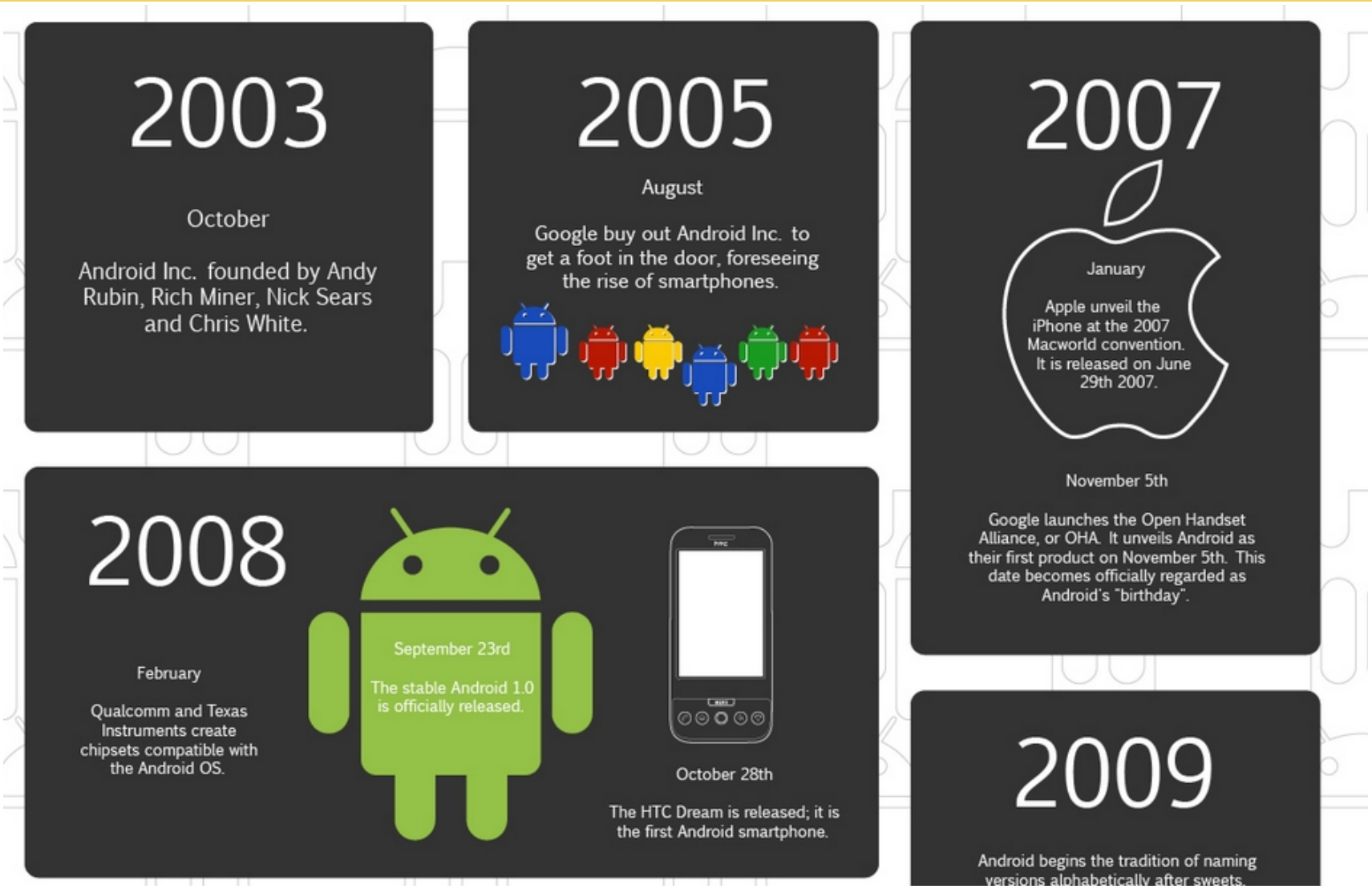
- In realtà, il software era stato c
- Da Android Inc., la classica s
  - Nata nel 2003 a Palo Alto
  - Acquistata da Google nel 20
  - Brevetti registrati nel 2007
- Quasi tutti i giochi erano già
  - Architettura complessiva
  - Ambiente di sviluppo
  - Licensing
  - Partner (telefoni e carrier)

Fondata da **Andy Rubin**

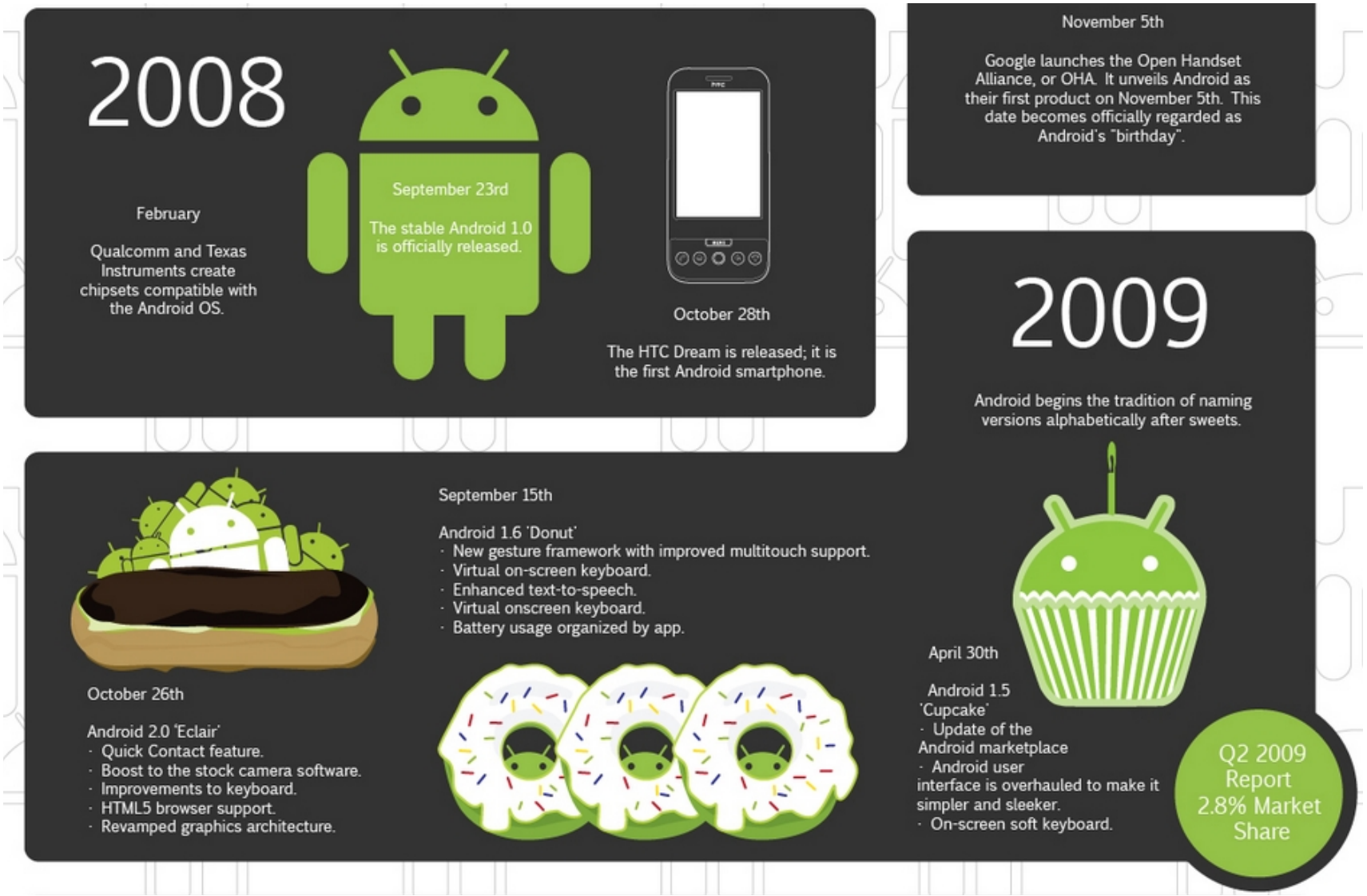
- 1986-1989 Carl Zeiss AG, robotics engineer
- 1989-1992 **Apple Inc.**, manufacturing engineer
- 1992-1995 General Magic, engineer. An Apple spin-off where he participated in developing Magic Cap, an operating system and interface for **hand-held mobile devices**.
- 1995-1999 **MSN** TV, engineer. When Magic Cap failed, Rubin joined Artemis Research, founded by Steve Perlman, which became WebTV and was eventually acquired by **Microsoft**.
- 1999-2003 Danger Inc., co-founder. Founded with Matt Hershenson and Joe Britt. Firm is most notable for the Danger Hiptop, often branded as the **T-Mobile** Sidekick, which is a phone with PDA-like abilities. Firm was later acquired by **Microsoft** in February 2008.
- 2003-2005 **Android Inc.**, co-founder.
- 2005-2014 **Google**. Senior Vice President in charge of **Android** for most of his tenure. Since December 2013, managing the robotics division of Google (which includes companies bought by Google, such as Boston Dynamics).
- 2014 Left Google to start an “incubator for hardware startups”
- 2015 Founder of Playground Global, an incubator “to help make advances in Artificial Intelligence (AI)” – $300M capital
- 2016 Joined Essential, Inc. - maker of smartphones

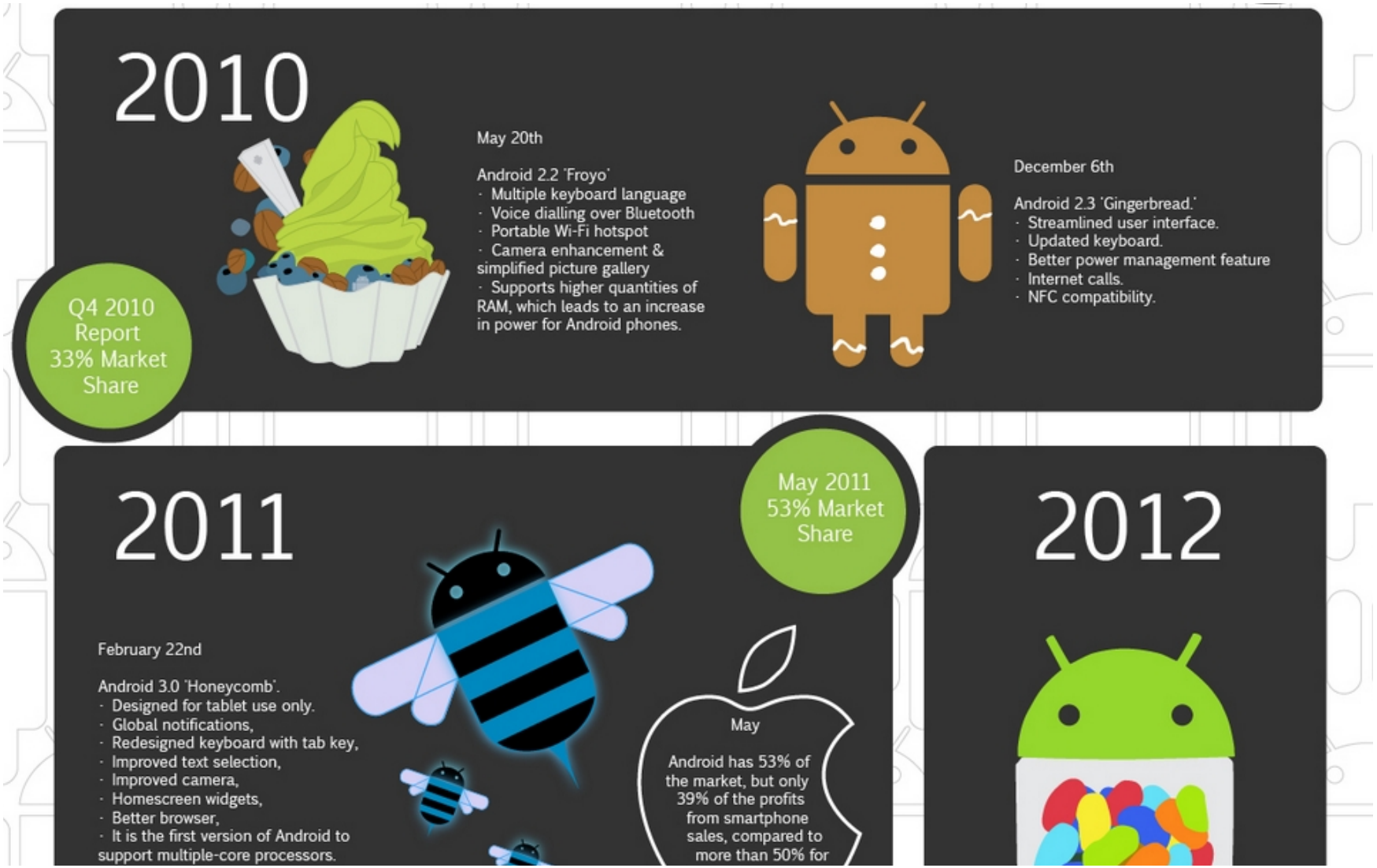
Breve storia di Android
Episode II

- Dal 2007, sono state rilasciate numerose versioni
- Numero di versione e *codename*
  - Nomi di dolciumi
  - In ordine alfabetico!



www.mobilemadhouse.co.uk


2008
February
Qualcomm and Texas Instruments create chipsets compatible with the Android OS.
September 23rd
The stable Android 1.0 is officially released.
October 28th
The HTC Dream is released; it is the first Android smartphone.
November 5th
Google launches the Open Handset Alliance, or OHA. It unveils Android as their first product on November 5th. This date becomes officially regarded as Android's "birthday".
2009
Android begins the tradition of naming versions alphabetically after sweets.
September 15th
Android 1.6 'Donut'
· New gesture framework with improved multitouch support.
· Virtual on-screen keyboard.
· Enhanced text-to-speech.
· Virtual onscreen keyboard.
· Battery usage organized by app.
October 26th
Android 2.0 'Eclair'
· Quick Contact feature.
· Boost to the stock camera software.
· Improvements to keyboard.
· HTML5 browser support.
· Revamped graphics architecture.
April 30th
Android 1.5 'Cupcake'
· Update of the Android marketplace
· Android user interface is overhauled to make it simpler and sleeker.
· On-screen soft keyboard.
Q2 2009 Report 2.8% Market Share
www.mobilemadhouse.co.uk


2010
May 20th
Android 2.2 'Froyo'
· Multiple keyboard language
· Voice dialling over Bluetooth
· Portable Wi-Fi hotspot
· Camera enhancement & simplified picture gallery
· Supports higher quantities of RAM, which leads to an increase in power for Android phones.
December 6th
Android 2.3 'Gingerbread.'
· Streamlined user interface.
· Updated keyboard.
· Better power management feature
· Internet calls.
· NFC compatibility.
Q4 2010 Report 33% Market Share
2011
May 2011 53% Market Share
February 22nd
Android 3.0 'Honeycomb'.
· Designed for tablet use only.
· Global notifications,
· Redesigned keyboard with tab key,
· Improved text selection,
· Improved camera,
· Homescreen widgets,
· Better browser,
· It is the first version of Android to support multiple-core processors.
May
Android has 53% of the market, but only 39% of the profits from smartphone sales, compared to more than 50% for
2012

2011

May 2011
53% Market Share

February 22nd

Android 3.0 'Honeycomb'.
- Designed for tablet use only.
- Global notifications,
- Redesigned keyboard with tab key,
- Improved text selection,
- Improved camera,
- Homescreen widgets,
- Better browser,
- It is the first version of Android to support multiple-core processors.

May

Android has 53% of the market, but only 39% of the profits from smartphone sales, compared to more than 50% for Apple.

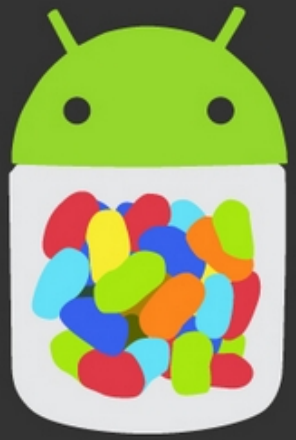
October 5th

Apple founder Steve Jobs dies after a long battle with pancreatic cancer. He is succeeded by Tim Cook.

October 19th

Android 4.0 'Ice Cream Sandwich'
- It converges the tablet and smartphone Android devices.
- Homescreen folders.
- Resizable and refreshed widgets.
- A more dynamic lock page.
- Multiple customizable shortcuts.
- Quick responses to calls.
- Improved stock keyboard.
- Better video editing software.
- Android Beam NFC functionality.

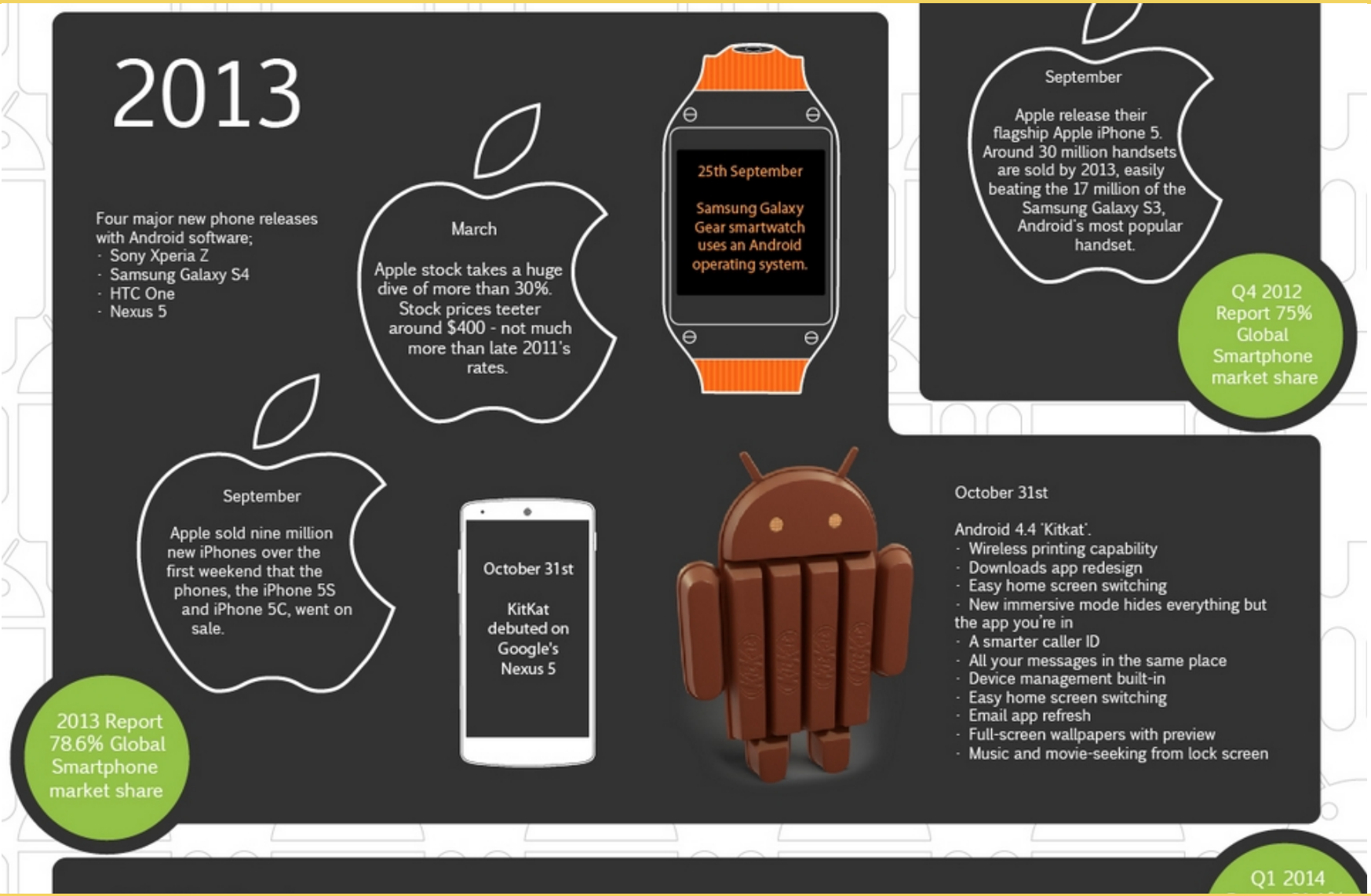
2012



May - June

Android 4.1 'Jelly Bean'
- Increased performance
- Streamline usability.
- "Project Butter" Software to makeAndroid devices run more smoothly.
- Google Now - a voice search function.
- Better support for non-English speaking users.
- Larger & richer notifications.
- Automatically resizing app widgets.
- Simplified task navigation.

www.mobilemadhouse.co.uk


2013
Four major new phone releases with Android software;
· Sony Xperia Z
· Samsung Galaxy S4
· HTC One
· Nexus 5
March
Apple stock takes a huge dive of more than 30%. Stock prices teeter around $400 - not much more than late 2011's rates.
25th September
Samsung Galaxy Gear smartwatch uses an Android operating system.
September
Apple release their flagship Apple iPhone 5. Around 30 million handsets are sold by 2013, easily beating the 17 million of the Samsung Galaxy S3, Android's most popular handset.
Q4 2012 Report 75% Global Smartphone market share
September
Apple sold nine million new iPhones over the first weekend that the phones, the iPhone 5S and iPhone 5C, went on sale.
October 31st
KitKat debuted on Google's Nexus 5
October 31st
Android 4.4 'Kitkat'.
· Wireless printing capability
· Downloads app redesign
· Easy home screen switching
· New immersive mode hides everything but the app you're in
· A smarter caller ID
· All your messages in the same place
· Device management built-in
· Easy home screen switching
· Email app refresh
· Full-screen wallpapers with preview
· Music and movie-seeking from lock screen
2013 Report 78.6% Global Smartphone market share
Q1 2014
www.mobilemadhouse.co.uk


2014
New phone releases with Android software;
· Sony Xperia Z2
· Samsung Galaxy S5
· HTC One (M8)
· LG G3
· Sony Xperia Z3
· Nexus 6
· OnePlus One
September
Apple release iPhone 6 and 6 Plus
October
Android 5.0 'Lollipop'
· Nexus 6 and 9 released as flagship Android devices.
· Songs, photos, apps and recent searches from one Android device can be accessed across all of your Android devices.
· Project Volta Battery Life – savings of over 36% by instructing apps to carry out less important tasks during charging time.
· Intuitive system that prioritises notifications by learning from your interaction habits.
Q1 2014 Report 81.1% Global Smart phone market share
October
Lolipop debuted on Google's Nexus 6 & Nexus 9
android wear
Announced on March 18th, Android Wear is Androids operating system for smart watches and other wearables.
June 25th LG G Watch
June 25th Samsung Gear Live
Summer Time Motorola Moto 360
LG G Watch
Samsung Gear
moto 360
Q2 2014 84.7% Global Smartphone market share
www.mobilemadhouse.co.uk

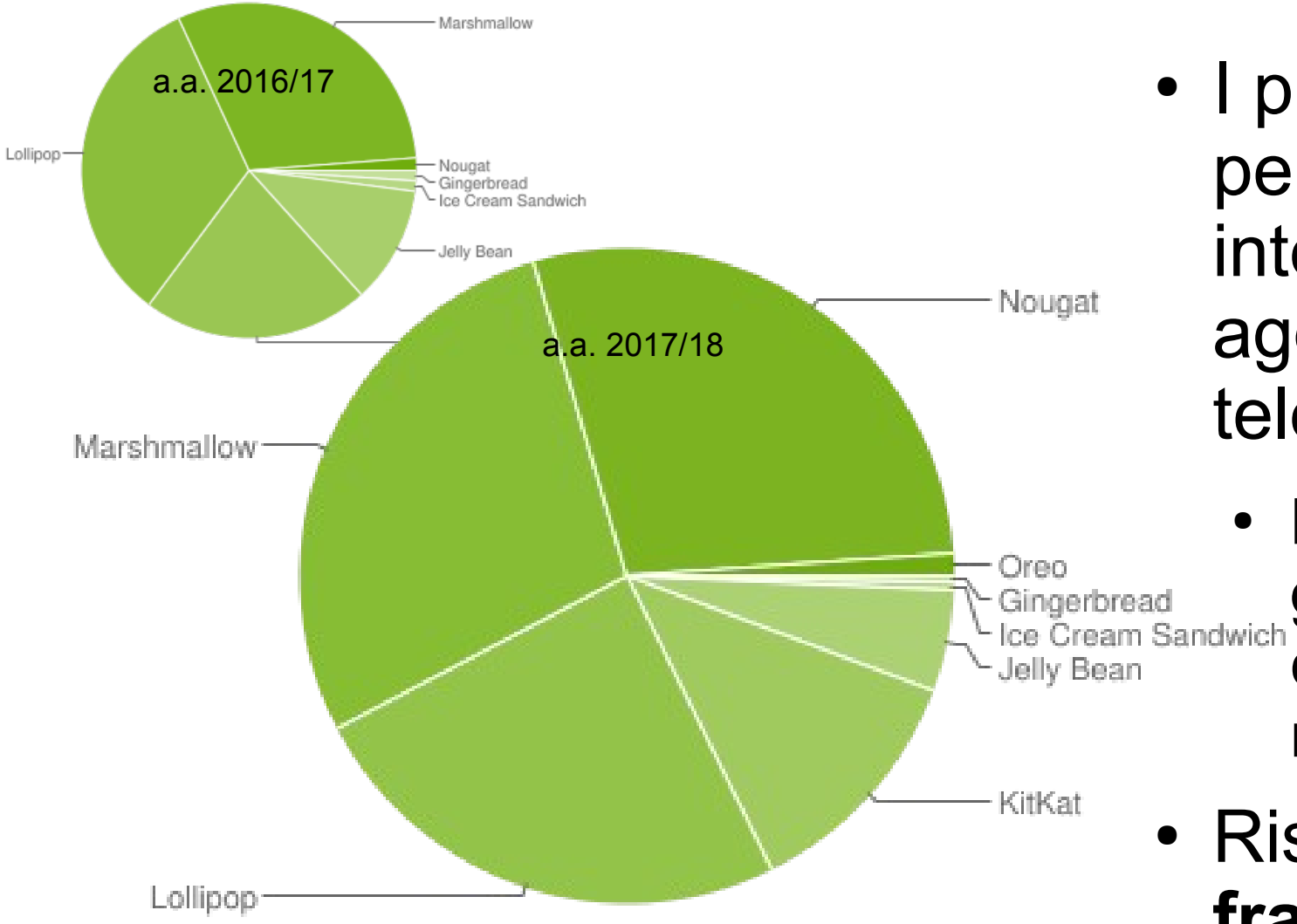
Release “corrente”

- A ottobre 2015 è stato rilasciato Android 6 (Marshmallow)
  - Significativi cambiamenti al sistema dei permessi
  - Numerose altre modifiche lato-utente
    - Che però non interessano lo sviluppatore di app
- 2016: Android 7 “Nougat”
  - Split screen, API Vulkan, ottimizzazioni batteria e dati
  - Android 7.1.1 di Agosto 2016
- 2017: Android 8 “Oreo”
- 2018: Android “P…”, probabilmente fra Marzo e Maggio

Breve storia di Android
Episode II

- Da qui in avanti ci occuperemo quasi esclusivamente di **software**
- … ma non bisogna dimenticare il ruolo dell'**hardware**!
  - Potenza di calcolo
  - Efficienza della batteria
  - Sensori
  - Schermi

Breve storia di Android
Episode II



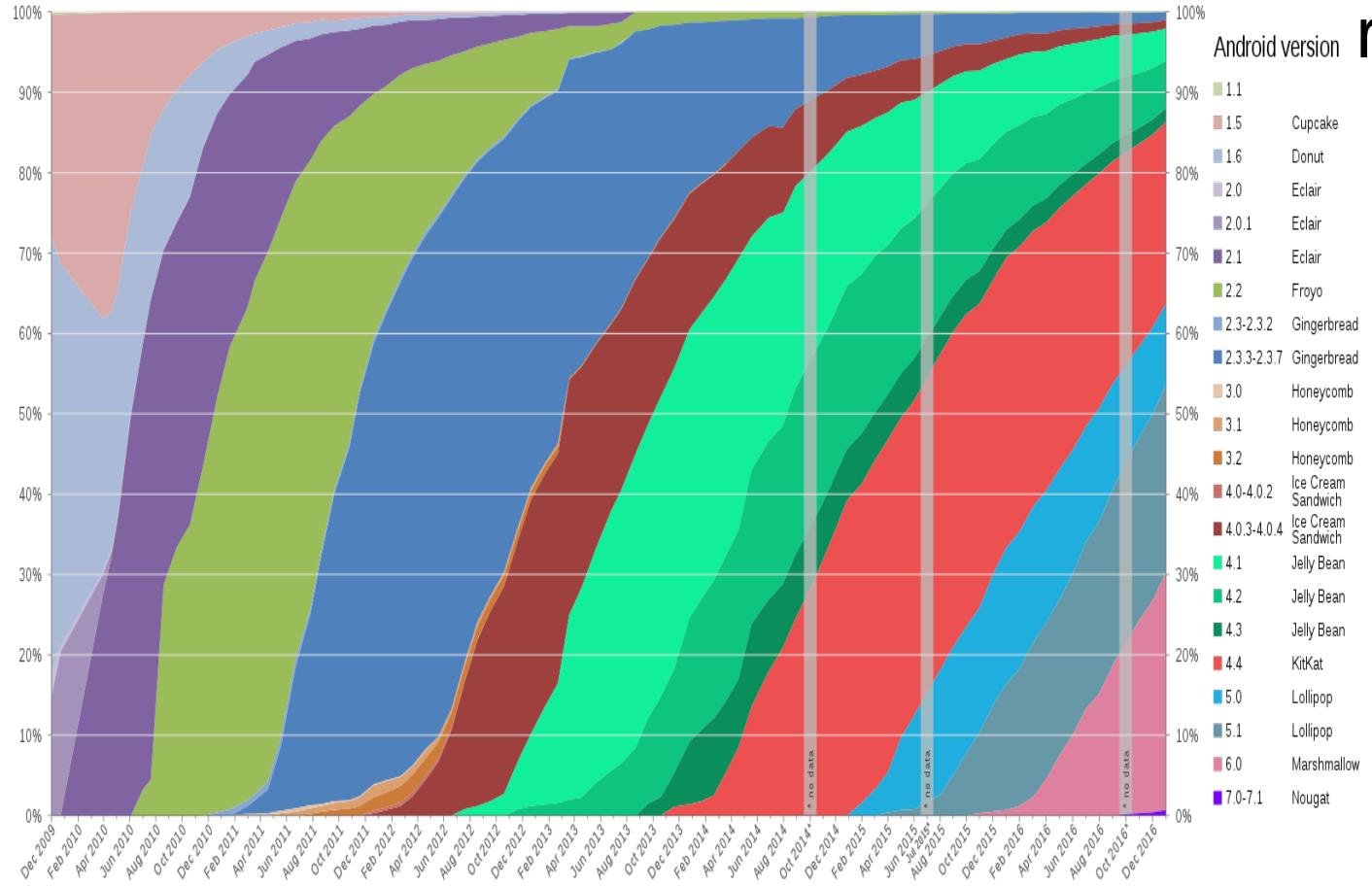
Dati aggiornati al 20 Feb 2018

- I produttori hanno però poco interesse ad aggiornare i telefonini “vecchi”
  - Meglio spingere gli utenti a comprarne di nuovi!
- Risultato: **frammentazione**

Breve storia di Android Episode II

- Ogni versione successiva è (quasi) sempre pienamente compatibile con le precedenti
- Cambiamenti nelle API sono identificate da un **API Level**
- Le applicazioni possono dichiarare
  - Un API Level **minimo** di cui necessitano per funzionare
  - Un API Level **target** per cui sono state scritte
  - Un API Level **massimo** oltre il quale non funzionano più
    - Pessima idea, sconsigliato, obsoleto, ignorato dopo Android 2.0.1
- Il **Market** e la **procedura di aggiornamento del S.O.** verificano il rispetto dei vincoli

Breve storia di Android
Episode II



Da Wikipedia; dati al 1 Febbraio 2017

- Diverso panorama rispetto a iOS
  - I device iOS vengono (quasi) sempre aggiornati alla versione più recente
  - Android tende a diffondere aggiornamenti più lentamente
  - L'Android più recente è *cool*, ma è sempre una nicchia!

Breve storia di Android Episode II

- Google fa un *tentativo* di supportare più o meno all'infinito vecchie versione del S.O.
  - Con le **librerie di compatibilità**
    - Codice che le applicazioni possono includere nel loro "eseguibile"
    - Simula le funzioni delle versioni più recenti su versioni antiche
  - Con i **Google Play Services**
    - Funzioni incorporate in una libreria aggiornabile da Market
- Grosso ostacolo: **customizzazione** (skinning)

Ambiente di sviluppo

Ambiente di sviluppo

- Possiamo considerare due livelli di sviluppo
  - Programmazione “nativa”
    - Si programma in C in ambiente Linux
    - GCC, librerie standard (libc, libm) e relativi header, librerie custom (liblogm, libjnigraphics) e relativi header
    - Tools per impacchettare codice nativo in formato **`.apk`**
  - Programmazione “standard”
    - Si programma in Java in ambiente Android (**non** Java!)
    - Javac, parte delle librerie J2SE, molte librerie custom
    - Tools per trasformare il bytecode Java in DEX e impacchettare le classi in formato **`.apk`**

PER ORA

Ambienti di sviluppo integrati

- Ci sono sostanzialmente tre “IDE” in uso per Android
- CLI (toolchain)
  - Si sviluppa su command line, eseguendo in una shell ogni comando (compilatore, linker, ecc.)
- Eclipse
  - Si usa Eclipse con dei plug-in ad-hoc per Android
- **Android Studio**
  - Versione customizzata di IntelliJ IDEA

Raramente utile
(non lo vedremo)

Utile se vi serve Android + altri plugin, ma sempre più raro

Default per Android “liscio”

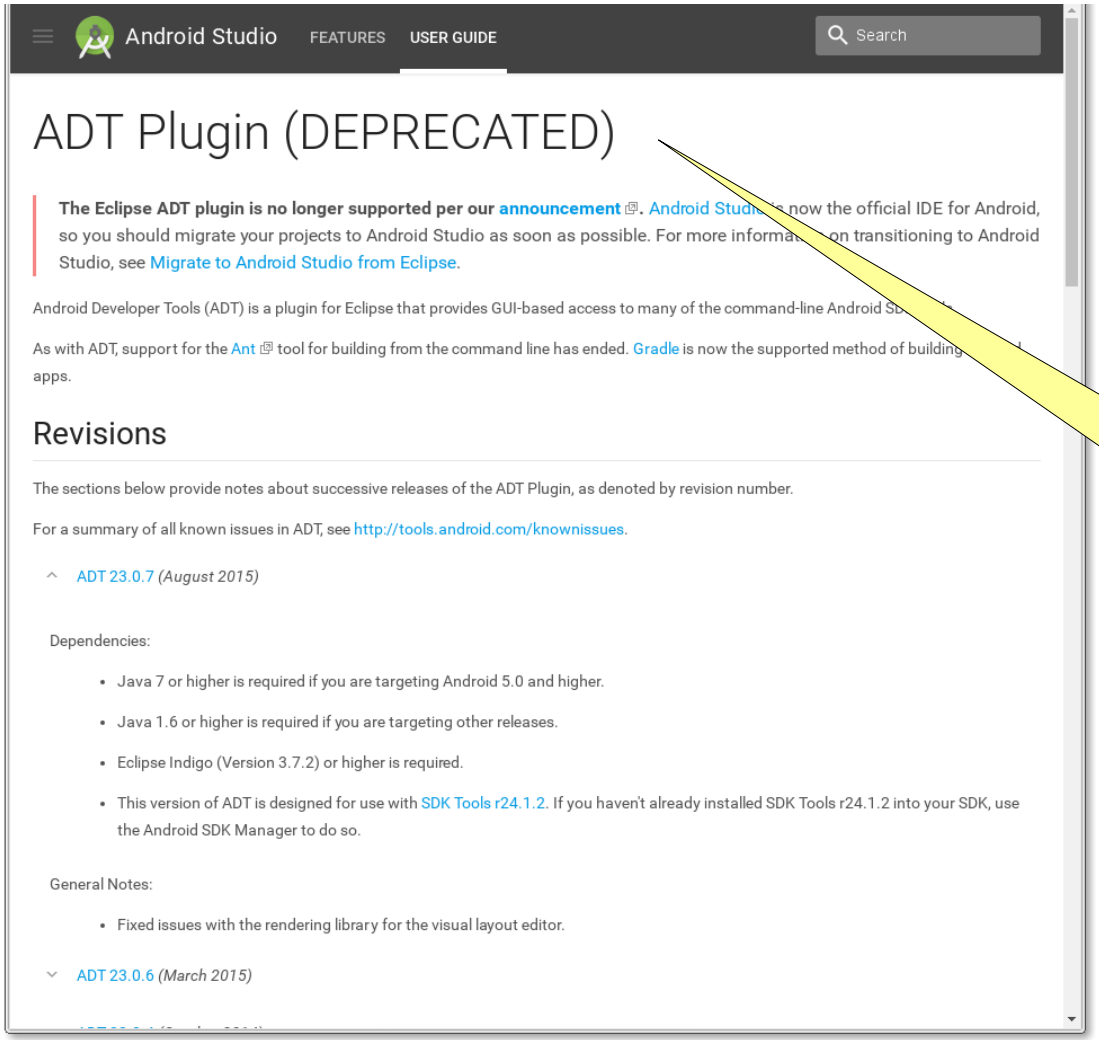
Componenti dell'ambiente di sviluppo tipico - Eclipse

- **Java Development Kit (JDK)**
  - Il più testato è quello Sun (Oracle), in teoria potrebbero funzionare anche altre implementazioni – ma solo in teoria
- **Eclipse**
  - Più testati i package *Eclipse IDE for Java Developers* e *Eclipse Classic*
- **Android Development Tools (ADT)**
  - Plug-in per Eclipse per aggiungere il supporto ad Android
- **Android starter Software Development Kit (SDK)**
  - I tool specifici “core” per la programmazione in Android
- **Android Platform** e altri componenti simili (extra tools)
  - Contengono le immagini eseguibili delle varie versioni di Android, nonché altre librerie contenenti package di utilità (es., per accedere al Market)

Installazione - Eclipse

- È possibile installare separatamente JDK, Eclipse, ADT, ecc. e poi collegarli fra di loro
  - Utile solo se avete una installazione di Eclipse già molto configurata pre-esistente
- Ma è molto più comodo installare l'**ADT Bundle**
  - Eclipse + ADT plugin
  - Android SDK Tools
  - Android Platform-tools
  - A version of the Android platform
  - A version of the Android system image for the emulator

Installazione – Eclipse



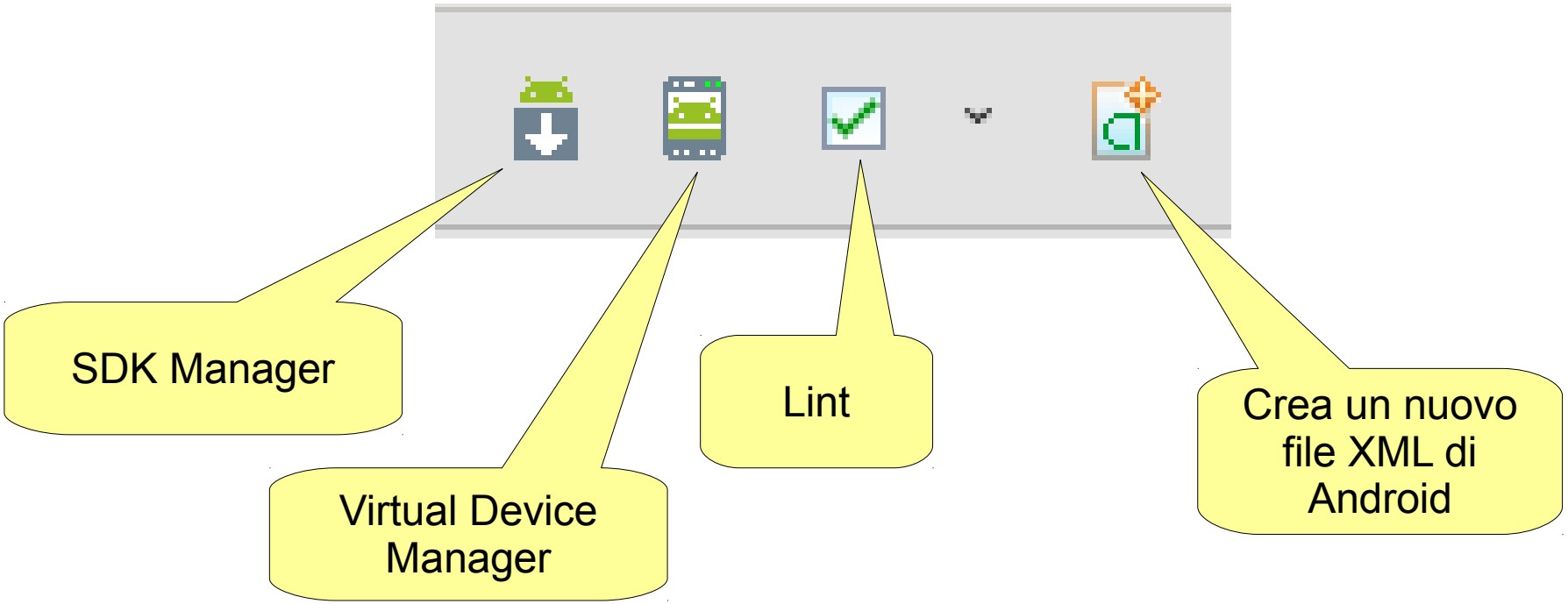
https://developer.android.com/studio/tools/sdk/eclipse-adt.html

Versione Linux 64 bit: 355Mb
Altri S.O. variano leggermente

Per chi non avesse capito da che parte va il mondo…

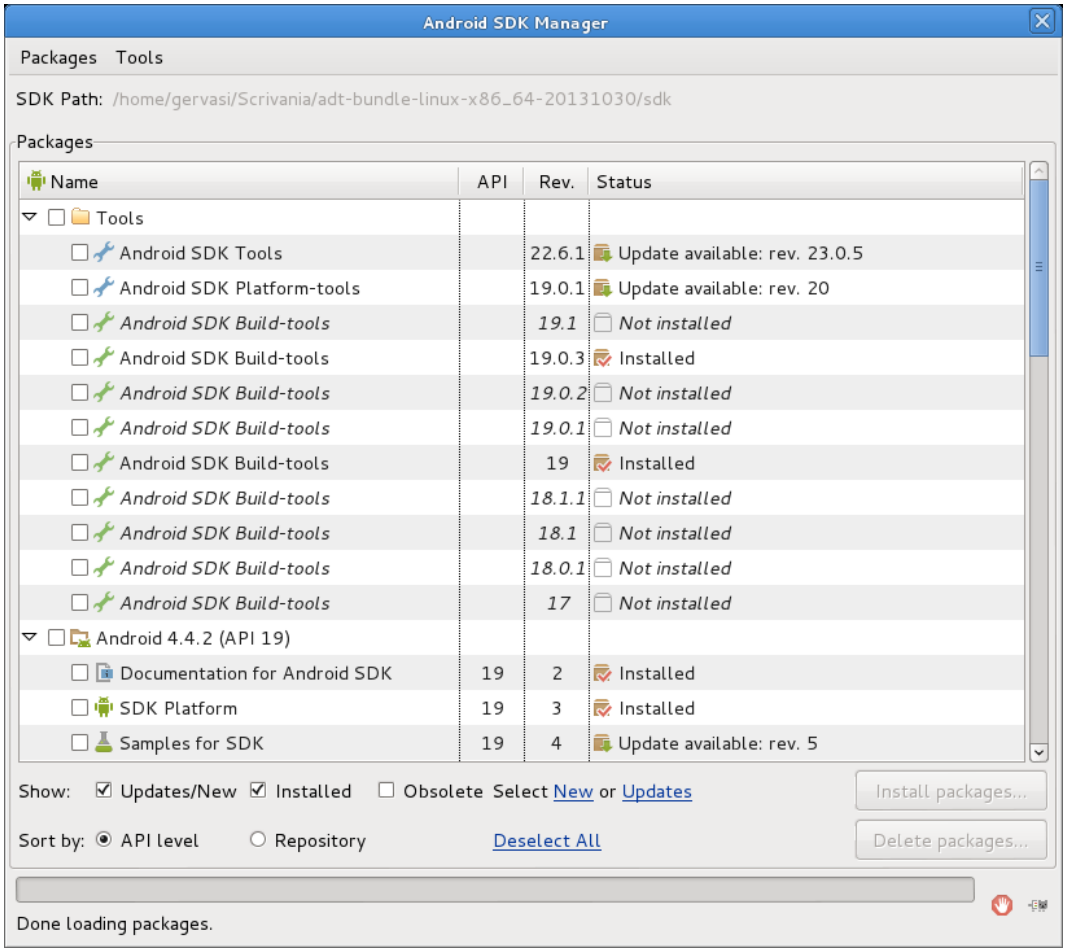
ADT su Eclipse

- Una volta installato l'ADT Bundle, si può aprire Eclipse
- Sulla toolbar troviamo quattro nuovi pulsanti:



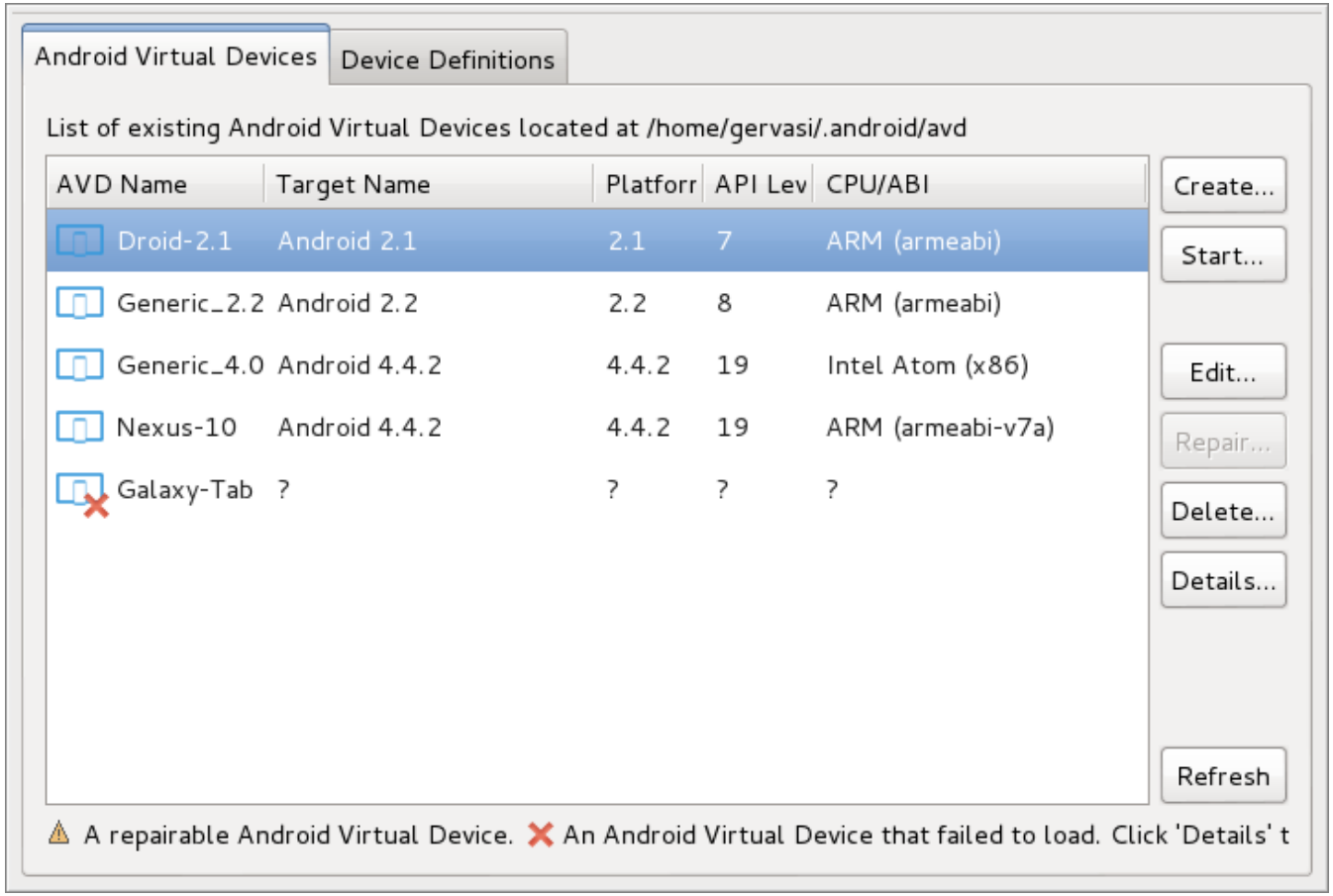
ADT su Eclipse

- L'**SDK Manager** è un **gestore di pacchetti** interno
  - Tipi di pacchetti:
    - Tool
    - Versioni di SDK Android
      - Librerie
      - Immagini virtuali per l'emulatore
      - Esempi
      - Documentazione
    - Altro
      - Librerie di terze parti
      - Librerie “esterne” di Google
        - Compatibilità
        - Google Play Services
        - AdSense
        - Ecc.

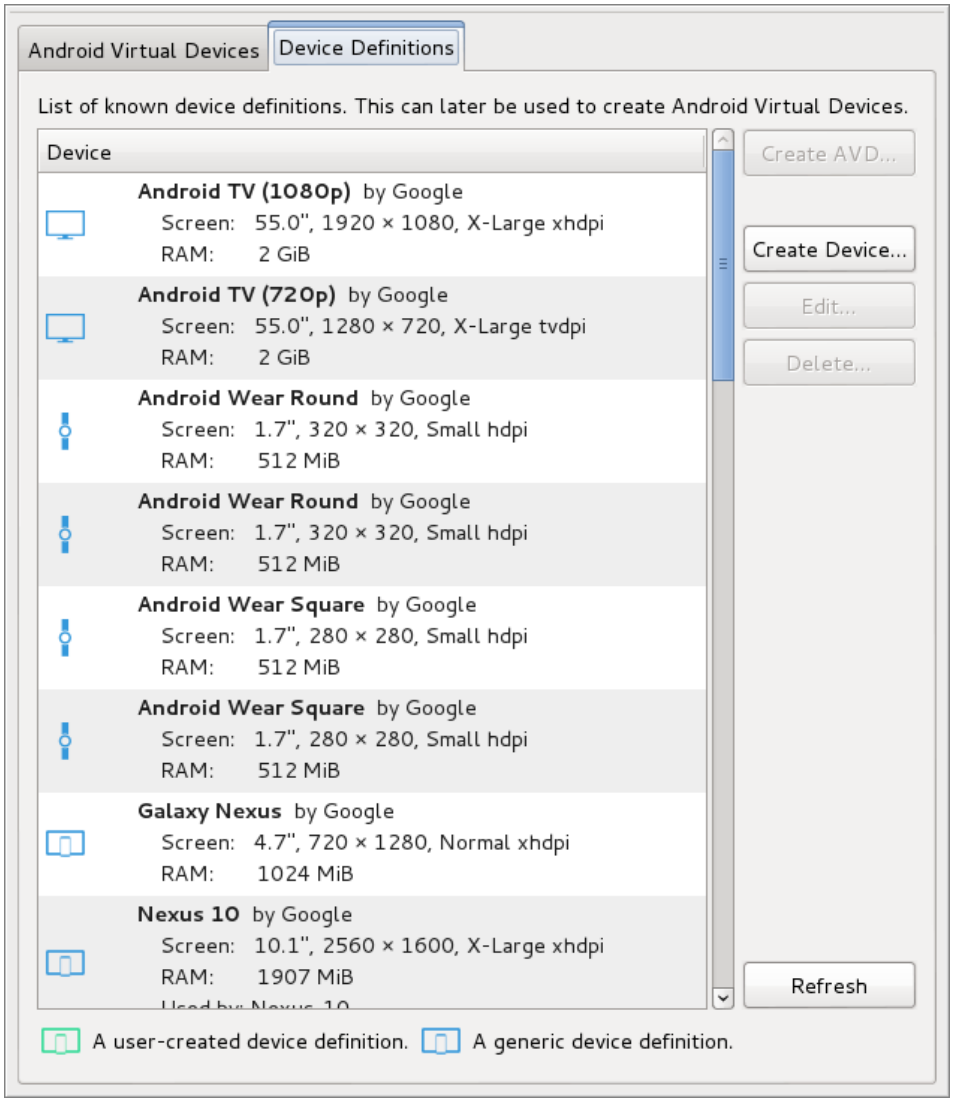


ADT su Eclipse

- L'**AVD Manager** gestisce le immagini dei vari dispositivi virtuali
  - **Immagine** = file contenente una copia della memoria
  - **Dispositivo virtuale** = configurazione per l'emulatore Android



ADT su Eclipse



- Vengono fornite **definizioni di dispositivo** per i casi più comuni
  - Android TV
  - Android Wear
  - Tutti i modelli NEXUS
  - Altri casi tipici
- Servono da base per creare i dispositivi virtuali (ciascuno con la sua immagine)

ADT su Eclipse

- Il **lint** è un venerabile tool per la ricerca **statica** di problemi nel codice
  - Vi segnala i casi più spesso problematici
    - Nel codice, nei file XML, nella struttura delle directory...
  - Non sempre sono **errori!**
- Il wizard "**Crea nuovo file XML**" vi offre dei template per i casi comuni
  - File di layout, menu, animazioni, preferenze, ecc.
  - Comodo, non indispensabile

Installazione – Android Studio

- In questo caso, esiste solo la versione “bundle”
- Include l'IDE vero e proprio, l'SDK, alcune immagini di default
- Funzionalmente, offre tutto quello che offre Eclipse
  - ambiente più moderno
  - un sistema di build più avanzato (e complicato)
  - tool per l’editing grafico di IU quasi utile
- È in sviluppo attivo (nuove release ogni settimana)

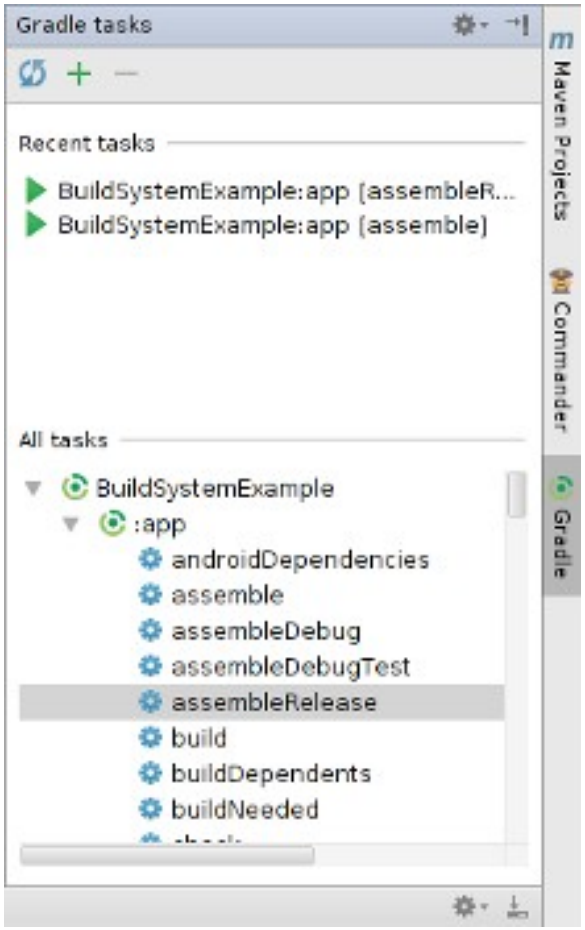
Installazione – Android Studio

- Download da https://developer.android.com/studio/index.html
  - Altri 440Mb di .zip, consigliato 1Gb di spazio disco
- Offre i soliti tool
  - SDK Manager
  - AVD Manager
- Diversi l'editor, i wizard di refactoring, il sistema di build
  - Eclipse usa Ant, Android Studio Gradle

Android Studio e Gradle

- **Gradle** è un sistema di build avanzato, molto configurabile, adatto allo sviluppo distribuito
- Rispetto ad Ant, offre
  - **Varianti** multiple del prodotto finale
    - Per esempio: per diverse architetture
  - Dipendenze **remote** (tramite Maven)
    - Un artefatto può dipendere da una libreria di terze parti di cui si ha l'URL; il sistema controlla, scarica e aggiorna se necessario
  - **Fill-in del manifest**
    - “aggiusta” i file XML del manifest secondo il particolare build

Android Studio e Gradle



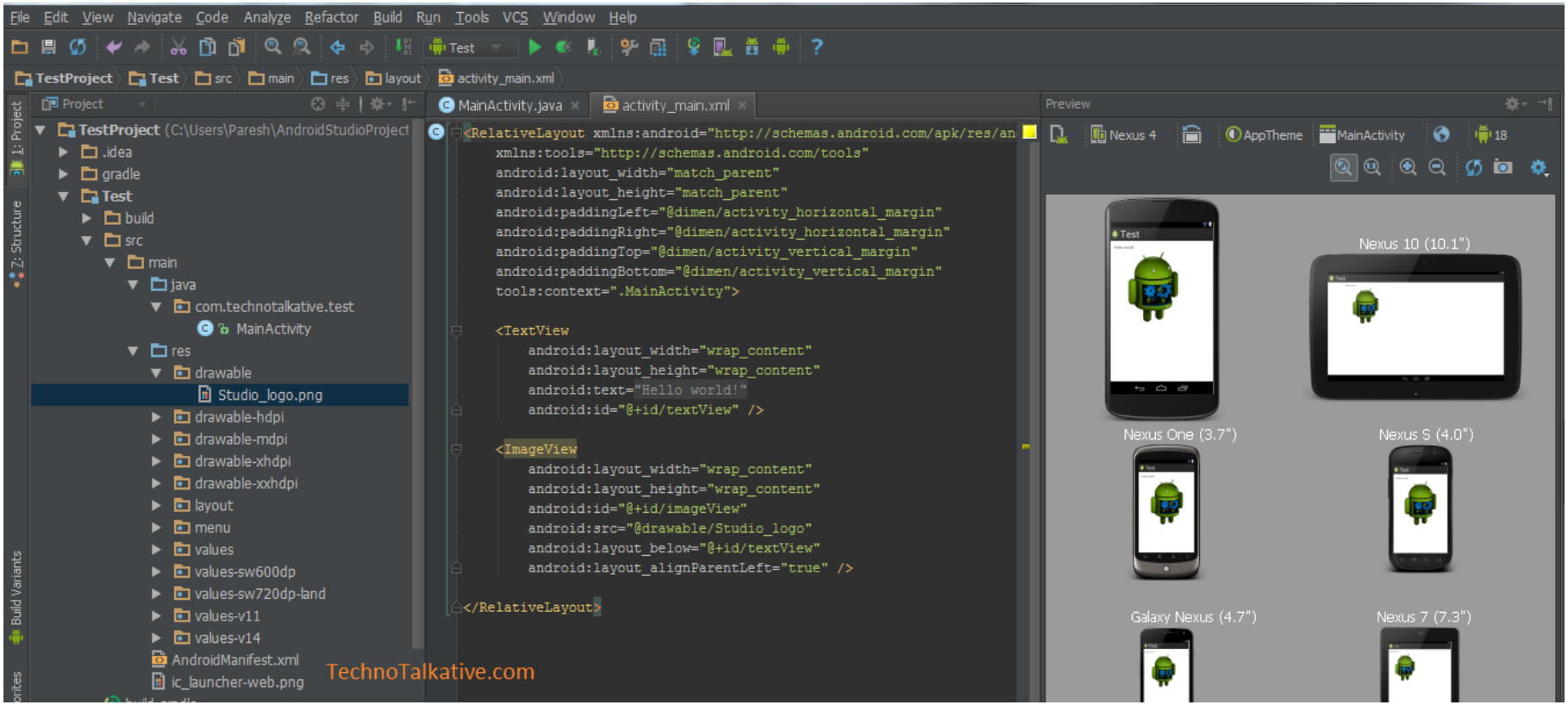

```
apply plugin: 'android'

android {
    compileSdkVersion 19
    buildToolsVersion "19.0.0"

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 19
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            runProguard true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), \
            'proguard-rules.txt'
        }
    }
}

dependencies {
    compile project(":lib")
    compile 'com.android.support:appcompat-v7:19.0.1'
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Scotto da pagare: una **maggiore complessità** nella configurazione del build system.

(Per fortuna molti default sono ragionevoli...)

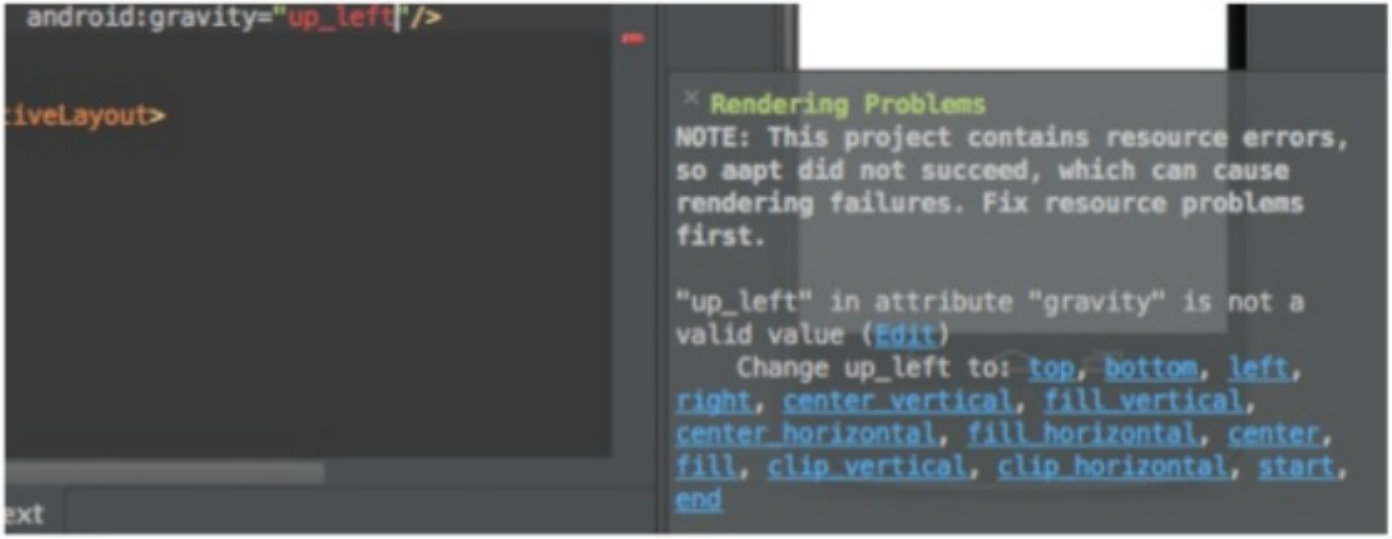
Android Studio

- Android Studio ha importanti vantaggi:
  - Design della UI in parallelo su più form factor

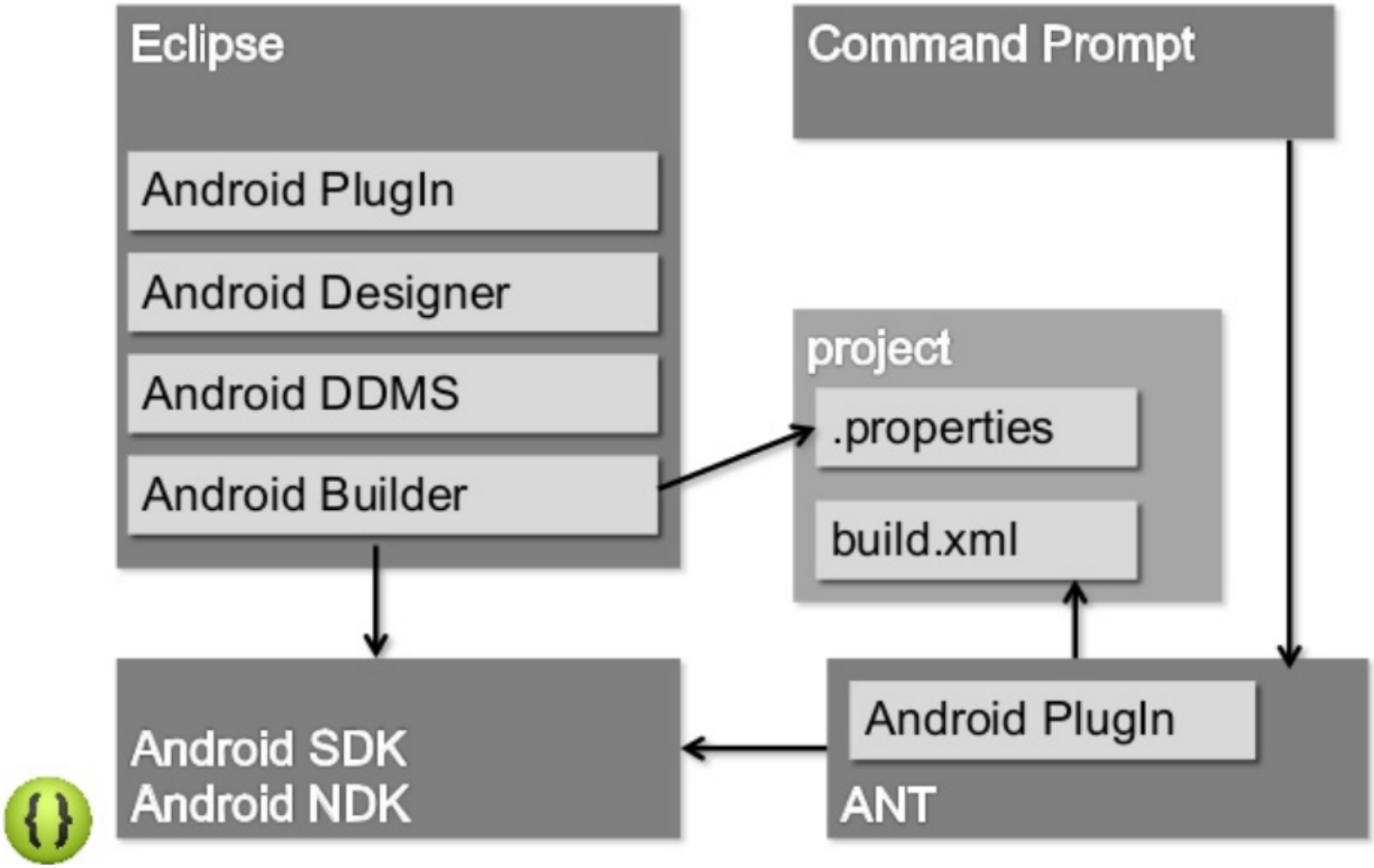


Android Studio

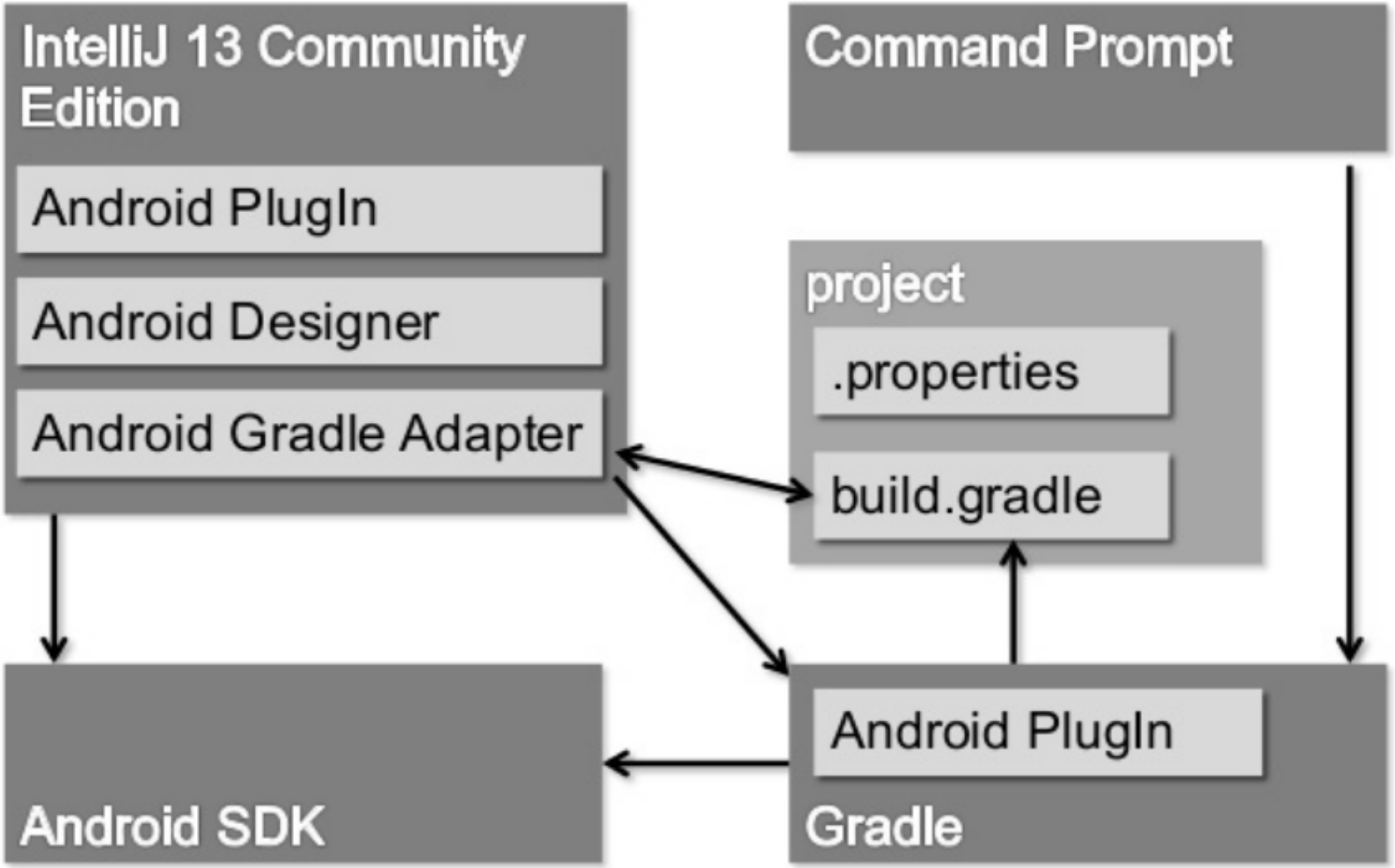
- Android Studio ha importanti vantaggi:
  - Design della UI in parallelo su più form factor
  - Molto più veloce durante l'uso e la compilazione
  - Maggiore integrazione con gli strumenti di debug
  - Integrazione più stretta con Lint



Architetture a confronto

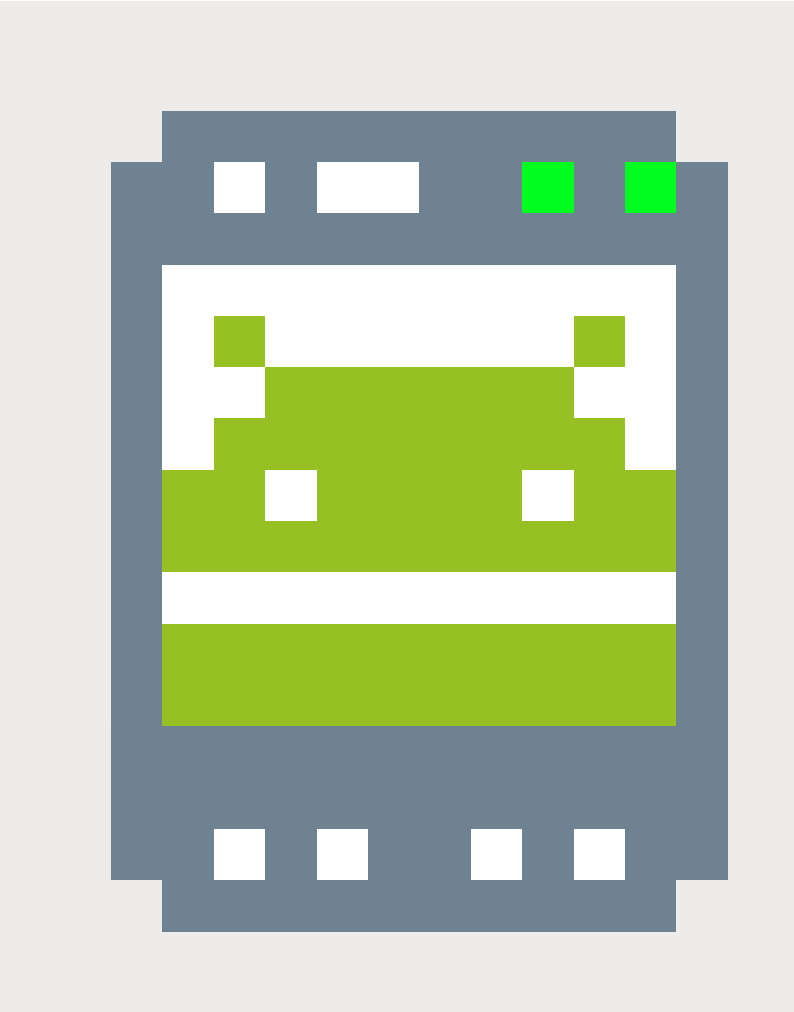


Architetture a confronto



L'emulatore

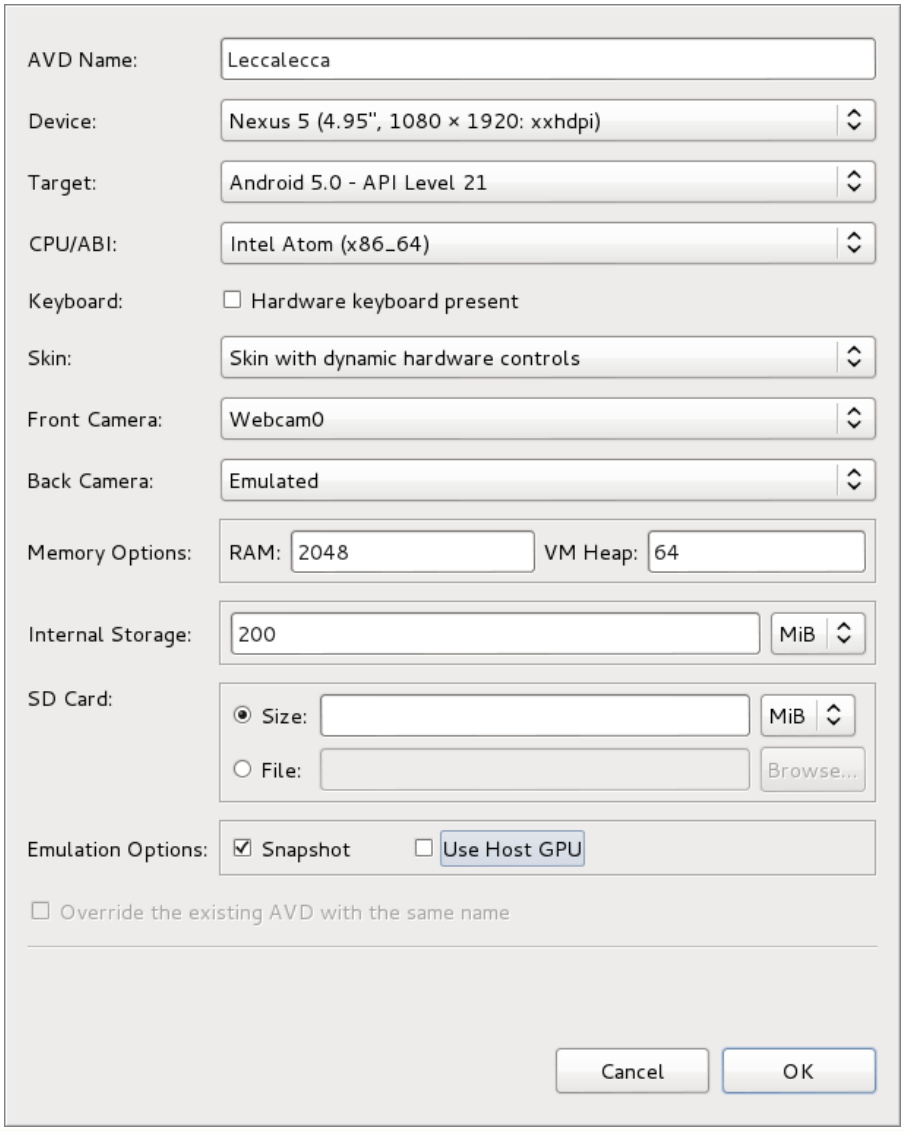
- L'**AVD Manager** è il punto di avvio per l'**emulatore** di Android
  - Consente di eseguire applicazioni senza necessità di un dispositivo fisico
- Gestisce vere e proprie **macchine virtuali**
  - Diverso hw/os simulati
  - Possibiltà di snapshot



Interprete vs. Esecuzione VM

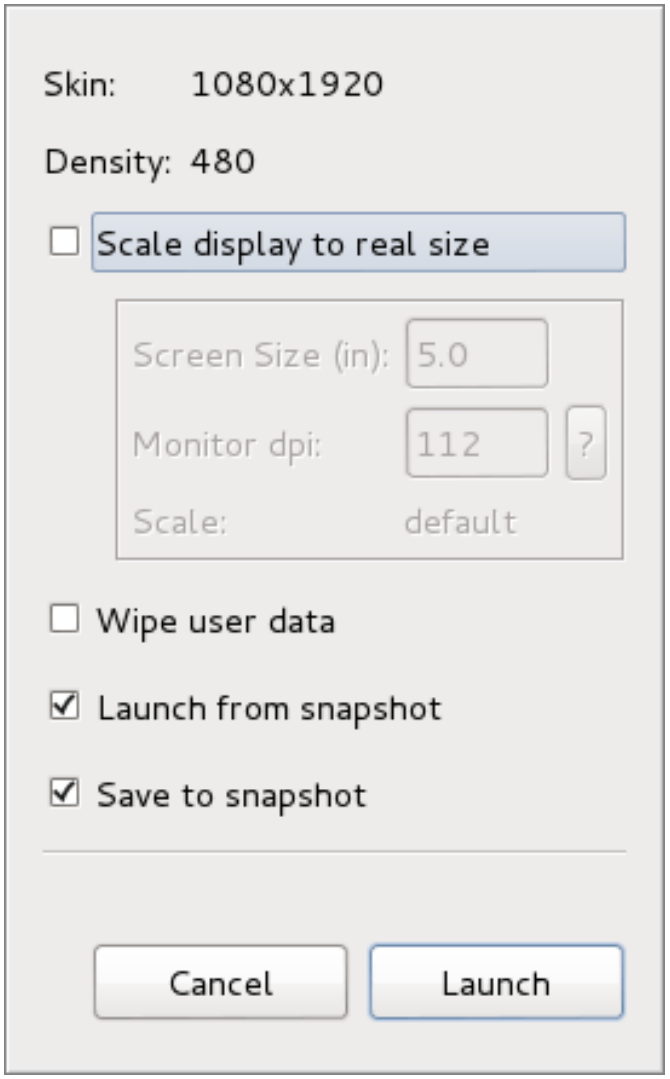
- Problema
  - La mamma va al mercato. Compra un PC e uno smartphone. Vuole sviluppare su Android
  - Il PC ha un chip Intel, ma lo smartphone ha un ARM
  - Come può simulare su PC l'esecuzione su ARM?
- Risoluzione
  - Se l'emulatore è configurato con un'immagine ARM, viene eseguita l'app in modo **interpretato** (lento)
  - Se l'emulatore è configurato con un'immagine x86 Atom, viene eseguita l'app in modo **nativo** (veloce)
    - Dentro una macchina virtuale a livello hardware – su Linux richiede il modulo **KVM** installato nel kernel

L'emulatore



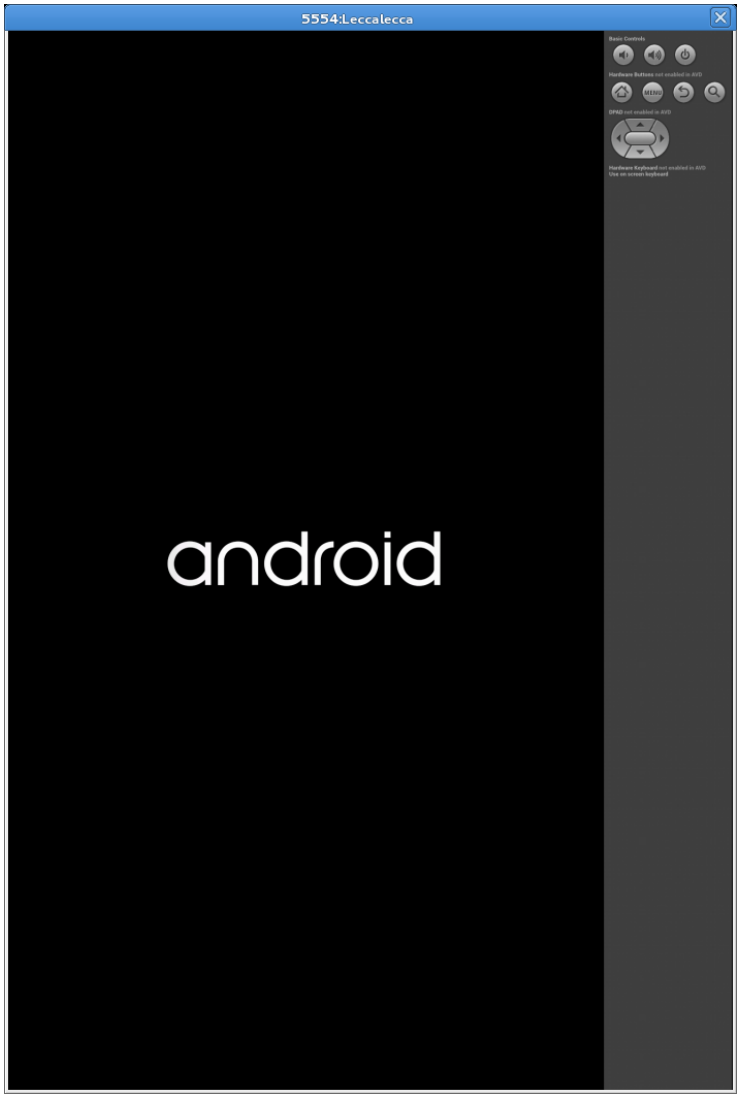
- Selezionare l'AVD Manager
  - Clic sull'icona!
- Selezionare Create...
- Definire le caratteristiche della VM desiderata
  - In particolare: il Target (espresso come API Level)
- Selezionare OK

L'emulatore



- Il dispositivo virtuale appena creato si aggiunge alla lista dei dispositivi dell'AVD
- Con Start... si apre una ulteriore finestra di configurazione
  - Si noti l'uso degli snapshot
- Con Launch, si avvia finalmente l'emulatore

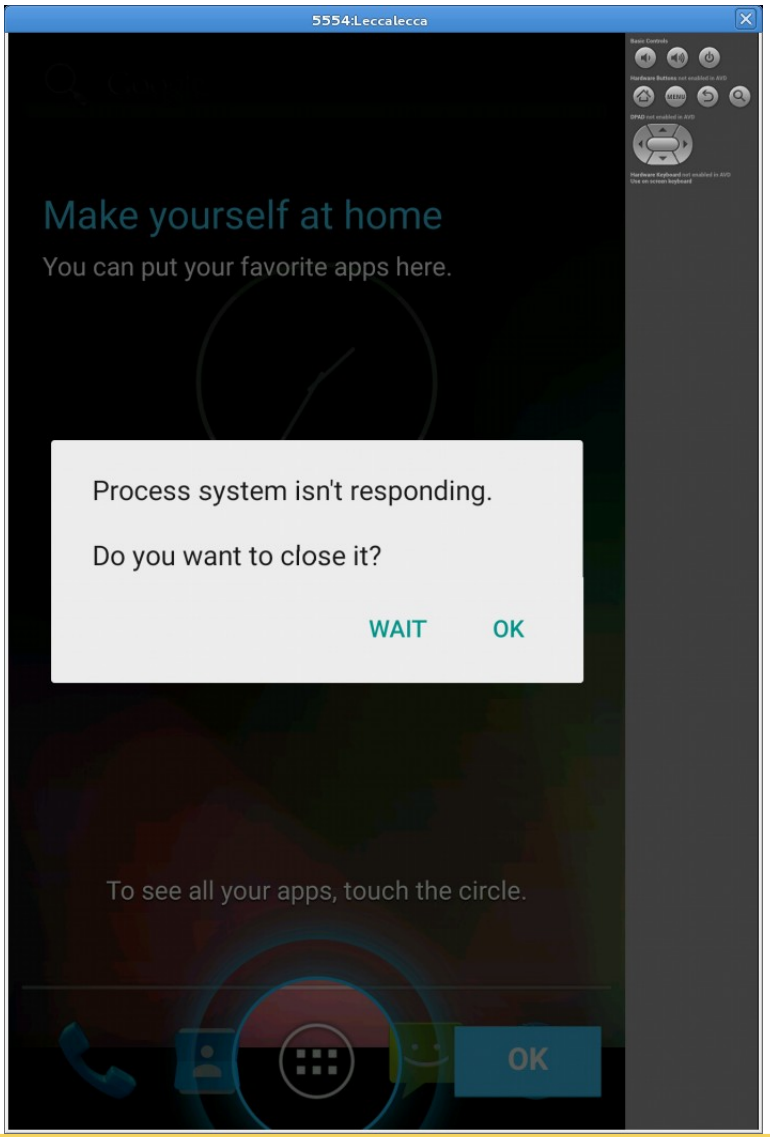
Test dell'ambiente



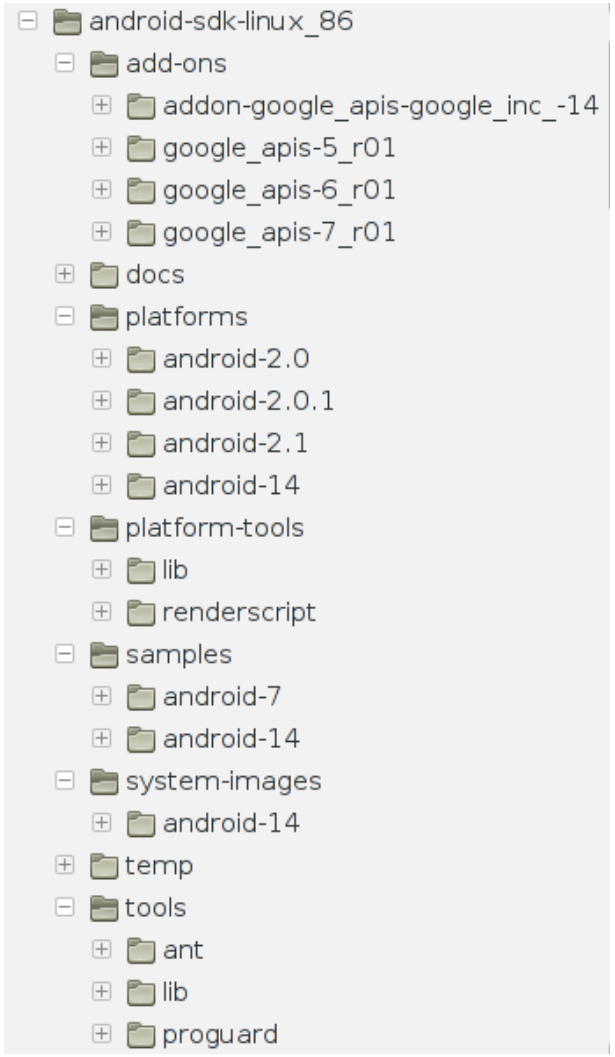
- Pazienza, pazienza...
  - Se usare ARM ABI, l'emulatore sta interpretando il codice ARM istruzione per istruzione!
  - Il primo avvio è un'operazione lenta anche sui telefoni veri
  - Per fortuna, si può abilitare lo snapshot
    - Restore immediato le volte successive

Test dell'ambiente

- **Congratulazioni!**
- Parte il “wizard di primo avvio” di Android
- Se siamo arrivati fin qui... pronti per programmare!
  - Sempre se avete *molta* pazienza, potete provare subito Android 7



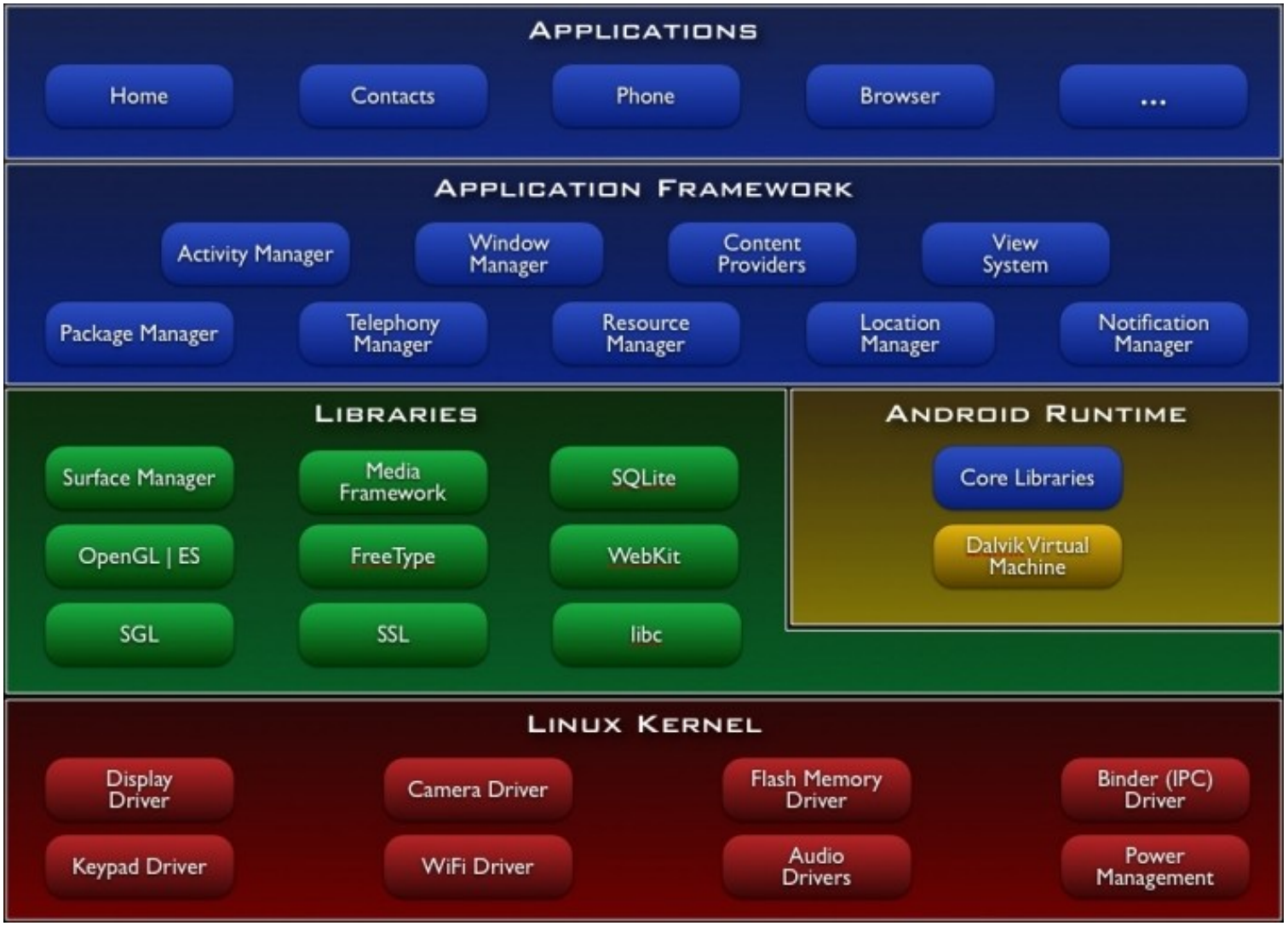
Esplorare l'SDK



- Potete anche esaminare il contenuto dell'SDK
  - Tool di base
  - Tutti i componenti aggiuntivi installati tramite l'SDK Manager
  - I docs includono copia della documentazione di riferimento disponibile online
    - Stile “Javadocs”, con qualche aggiustamento

Un po' di architettura

The big picture



Studieremo a fondo

Studieremo per diletto

Largamente invisibili

Utile da sapere

Il Kernel

- Alla base di tutto c'è un **kernel Linux**
- Si tratta di un kernel completo, con tutte le primitive UNIX a cui siamo abituati
  - Processi, gestione della memoria, IPC, thread
  - Filesystem, utenti, diritti
  - Librerie, shell, comandi
  - Driver (sotto forma di moduli) per vari device
    - Tipicamente, quelli presenti in ogni particolare dispositivo
    - SD card, reti, telefonia, sensori, ecc.

Applicazioni native

- È possibile scrivere applicazioni che chiamano direttamente il kernel
  - Direttamente (via syscall) o via librerie (es., stdio.h)
  - Il codice deve essere compilato per il particolare processore in uso su un certo telefonino
    - In genere, ARM – ma non forzosamente
  - Deve poi essere “impacchettato” in un formato specificato per la distribuzione/installazione
    - Non troppo diverso dai vari .rpm, .deb, e simili
- Sconsigliato! (noi lo vedremo in fondo)

Dalvik & ART

- La stragrande maggioranza delle applicazioni gira in una **macchina virtuale**: **Dalvik**
- È una versione di JVM con importanti differenze
  - Basata su registri (non su stack)
  - Set di istruzioni ottimizzato per risparmiare memoria e aumentare la velocità di esecuzione
  - Formato dei file eseguibili ottimizzato per risparmiare memoria
  - Eseguibile da più processi con una sola istanza
    - Tutto codice **rientrante** – sharing del codice di Dalvik via mmap()
  - **Non** sotto il controllo di Oracle (che infatti è in causa)

Dalvik & ART

- **Dalvik** è la macchina virtuale di default su Android
  - Dalla creazione fino ad Android 4.4
- Su Android 4.4, era disponibile un'opzione per sviluppatori per passare su ART
- Da Android 5, **ART** è la macchina virtuale di default
- Differenze:
  - ART pre-compila a install-time, non interpreta
  - Più lenta l'installazione, più veloce l'esecuzione
  - Largamente **invisibile** a programmatore e utente

Dalvik & ART

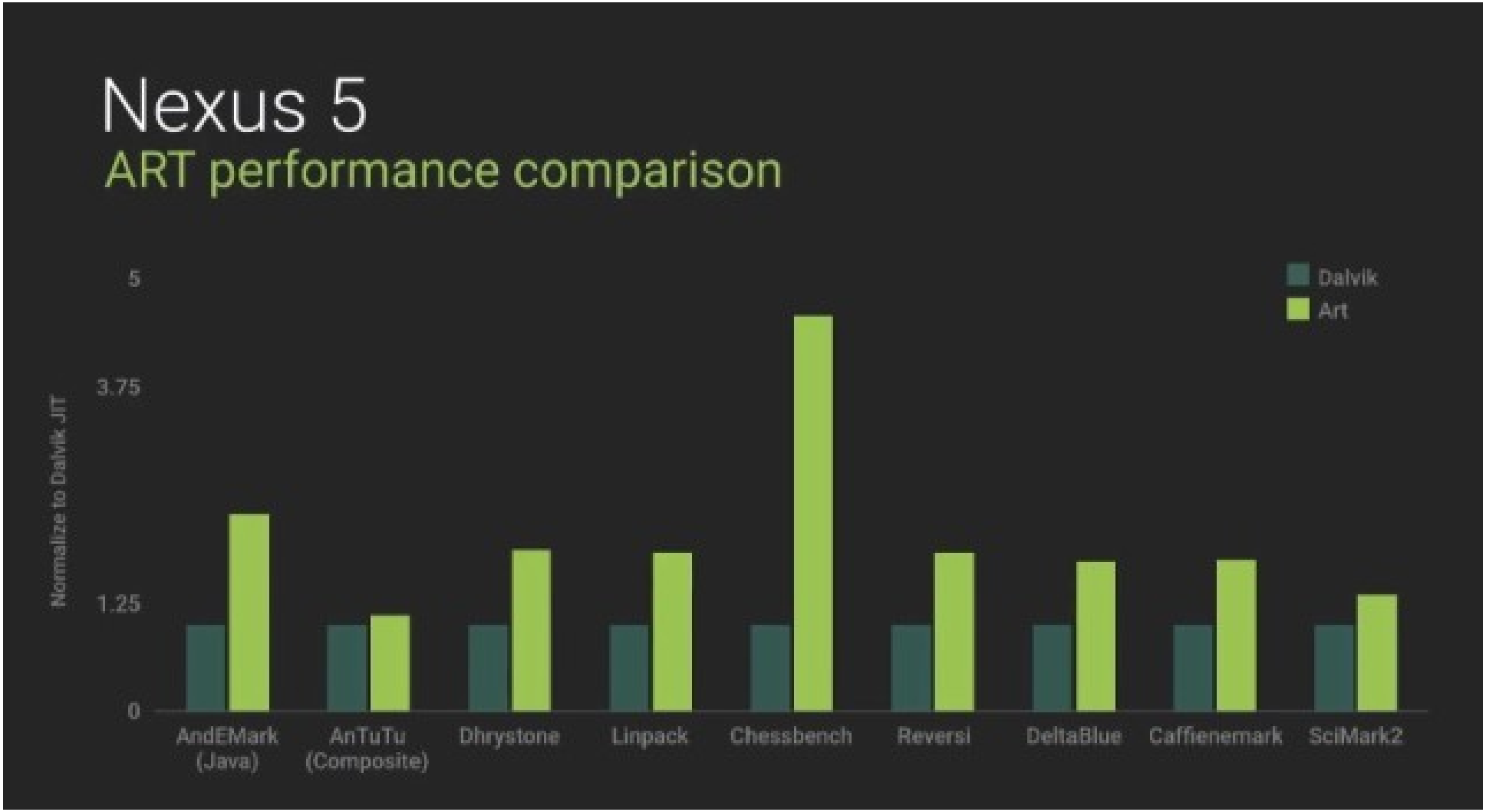
- **Dalvik** è la macchina virtuale di default su Android
  - Dalla creazione fino ad Android 4.4
- Su Android 4.4, era disponibile un'opzione per sviluppatori per passa
- Da Android 5, **ART** è l ault
- Differenze:
  - ART pre-compila a install-time, non interpreta
  - Più lenta l'installazione, più veloce l'esecuzione
  - Largamente **invisibile** a programmatore e utente

Casualmente, ART produce codice nativo, non bytecode... ulteriore assicurazione contro le cause di Oracle!

Dalvik & ART

- Sia Dalvik che ART sono entrambe rilevanti
  - Dalvik, perché è usato da praticamente il 100% dei dispositivi attualmente nelle mani degli utenti
  - ART, perché Google ha già detto che in futuro svilupperà e supporterà soltanto questa
    - Più veloce in esecuzione
    - Migliore gestione della *garbage collection*
      - Meno pause, grafica più fluida
    - Maggiore integrazione con profiling e debugging
    - Minor consumo di energia
      - Carica della batteria dura più a lungo

Dalvik & ART


Nexus 5
ART performance comparison
Dalvik
Art
Normalize to Dalvik JIT
5
3.75
1.25
0
AndEMark (Java)
AnTuTu (Composite)
Dhrystone
Linpack
Chessbench
Reversi
DeltaBlue
Caffienemark
SciMark2

Linux, VM, e sicurezza

- Ogni App viene eseguita dal kernel Linux
  - **In un processo separato**
    - Che esegue Dalvik che esegue il bytecode dell'app
    - Controllo dei **permessi** di accesso alle risorse logiche fatto dalla VM (i permessi sono concessi dall'utente-umano)
  - **Con uno user ID distinto**
    - Tutti i file creati dall'applicazione appartengono al “suo” user ID; altre applicazioni non possono accedere alla “sua” directory, né leggere i “suoi” file
    - È possibile che applicazioni *amiche* condividano processo e user ID – occorre però che siano firmate dallo stesso autore
    - Controllo dei **diritti** di accesso alle risorse fisiche fatto dal kernel (i diritti **non** sono controllati dall'utente-umano)

Linux, Dalvik, e sicurezza

- Risultato complessivo
  - Notevole grado di **separazione** e **isolamento** delle Applicazioni
    - Ci può sempre essere un buco di sicurezza non patchato nel kernel Linux, ma l'uso dello stesso kernel usato per tutte le altre applicazioni rende l'eventualità remota
  - Android è un sistema piuttosto **sicuro**, ma...
    - Sono sempre possibili exploit basati sull'**ingegneria sociale**
      - **Una App convince l'utente darle particolari permessi**
      - **La App usa poi questi permessi per uno scopo diverso da quello pubblicizzato**
    - Molto più difficili gli attacchi veri

Linux, Dalvik, e sicurezza

- In ambiente UNIX tradizionale, abbiamo
  - Le applicazioni (comandi) appartengono al sistema
  - Più utenti (umani) usano il sistema
    - Mutuamente malfidati
  - Il dominio di protezione è l'**utente**
- Su Android invece
  - C'è un solo utente umano, fidato ma inaffidabile
  - Le applicazioni sono mutuamente malfidate
  - Il dominio di protezione è l'**applicazione**

Linux, Dalvik, e sicurezza

- I veri utenti (nel senso UNIX) sono **i programmatori delle varie App**
- Il proprietario del telefonino è (nel senso UNIX) quasi un **device** :-)
  - Non ha un suo userID
  - Non è proprietario di nessun file nel file system
  - Non è titolare di nessun processo
  - Non ha nemmeno un login e una password!
- L'utente non fa niente, se non tramite App!

Hello, world!