Mobile Application Development

Introduction To Android Platform

(Professional Android 4 Application Development chapter 1,2)

Outline

- Overview of an Android platform
- Android SDK features
- Which devices Android runs on
- Why android for mobile development
- Comparison of Android with other platform
- Android development framework
- Understanding Android software stack
- Android application architecture
- Installing and Updating Android SDK,
- Android development tools: AVD manager, Android SDK manager, Emulator, Dalvik debug monitor service, Debug bridge, Hierarchy view and Lint Tool, Monkey and Monkey runner

Overview

- Android is an open-source software stack that includes the operating system, middleware, and key mobile applications, along with a set of API libraries for writing applications that can shape the look, feel, and function of the devices on which they run.
- Today mobile devices have become powerful tools that incorporate touch screens, cameras, media players, Global Positioning System (GPS) receivers, and Near Field Communications (NFC) hardware.

What Android Isn't

- A Java ME implementation — Android applications are written using the Java language, but they are not run within a Java ME (Mobile Edition) VM, and Java-compiled classes and executables will not run natively in Android.
- Part of the Linux Phone Standards Forum (LiPS) or the Open Mobile Alliance (OMA) — Android runs on an open-source Linux kernel, but, while their goals are similar, Android's complete software stack approach goes further than the focus of these standards-defining organizations.
- Simply an application layer (such as UIQ or S60) — Although Android does include an application layer, "Android'' also describes the entire software stack, encompassing the underlying operating system, the API libraries, and the applications themselves.

What Android Isn't

- A mobile phone handset — Android includes a reference design for mobile handset manufacturers, but there is no single "Android phone." Instead, Android has been designed to support many alternative hardware devices.
- Google's answer to the iPhone — The iPhone is a fully proprietary hardware and software platform released by a single company (Apple), whereas Android is an open-source software stack produced and supported by the Open Handset Alliance (OHA) and designed to operate on any compatible device.

Android

- Google's Andy Rubin describes Android as follows:

  "The first truly open and comprehensive platform for mobile devices. It includes an operating system, user-interface and applications — all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation."
- Android has expanded beyond a pure mobile phone platform to provide a development platform for an increasingly wide range of hardware, including tablets, televisions and wearable devices like watch, glass etc.

Android

- Android is an ecosystem made up of a combination of three components:
  - A free, open-source operating system for embedded devices
  - An open-source development platform for creating applications
  - Devices, particularly mobile phones, that run the Android operating system and the applications created for it

Android Parts

- A Compatibility Definition Document (CDD) and Compatibility Test Suite (CTS) that describe the capabilities required for a device to support the software stack.
- A Linux operating system kernel that provides a low-level interface with the hardware, memory management, and process control, all optimized for mobile and embedded devices.
- Open-source libraries for application development, including SQLite, WebKit, OpenGL, and a media manager.
- A run time used to execute and host Android applications, including the Dalvik Virtual Machine (VM) and the core libraries that provide Android-specific functionality. The run time is designed to be small and efficient for use on mobile devices.

Android Parts

- An application framework that agnostically exposes system services to the application layer, including the window manager and location manager, databases, telephony, and sensors.
- A user interface framework used to host and launch applications.
- A set of core pre-installed applications.
- A software development kit (SDK) used to create applications, including the related tools, plug-ins, and documentation.

History

1. Android 1.0: Alpha (API 1)

Features

Google Maps, browser, calendar

Camera and scroll down the notification bar

Gmail integration, Contacts, and Google Synchronization.

Wireless supports – Wi-Fi and Bluetooth.

History

2. Android 1.1: Beta (API 2)

Features

- Display details and reviews for locations
- Add Save attachment in the message
- Provide detailed information by clicking on the business

History

3. Android 1.5: Cupcake (API 3)

Features

- on-screen keyboard and search function
- Uploading videos and images
- Copy and paste facility and video recordings
- Support for MPEG4 and 3GP formats

History

4. Android 1.6: Donut (API 4)

Features

- Power Control widget for handling Wi-Fi, Bluetooth, GPS, etc.
- Gallery and Camera quick toggling features
- WVGA screen resolution speed
- Technology support for CDMA/EVDO, 802.1x, VPNs, and a text-to-speech engine
- Speed improvements for camera and searching applications
- Quick Search Box

History

5. Android 2.0: Eclair (API 5)

Features

- Update UI
- Support Bluetooth 2.1
- Improve Google map
- Minor API Changes
- Support Live and animated Wallpapers
- Ability to add contact’s photo and select to call, message or email

History

6. Android 2.2: Froyo (API 8)

Features

- Support for Animated GIF and multiple keyboard languages
- Speed and performance improvements
- Upload file support in the browser
- Support numeric & alphanumeric passwords to enhance security
- Increased Compatibility with car kits and headsets
- Wi-Fi Support Hotspot functionality

History

7. Android 2.3: Gingerbread (API 9)

Features

- Improve Copy and Paste Facility
- Updated UI design
- Support for VP8 and WebM video format
- Video calling and Social Networking Supports
- Easy to use a keyboard with faster and intuitive typing

History

8. Android 3.0: Honeycomb (API 11)

Features

- Gmail, contacts, camera and gallery improvements
- Support for passwords with complex characters
- encrypted storage and updated 3D UI
- Supports multiprocessors and recent apps for easy visual multitasking
- Media Sync from SD Card
- Action bar for application control
- System bar for global status and notifications
- Google eBooks and Talk Video Chat
- Support Adobe Flash in Browser
- More sensor support
- High-performance Wi-Fi Connections and Lock
- Chinese handwriting and redesigned keyboard

History

9. Android 4.0: Ice Cream Sandwich (API 14)

Features

- Spelling check feature
- Wi-Fi direct
- Photo Decor facility and on-screen buttons
- Unlocking with face-fixing.
- Card-like appearance for app-switching
- Improved video recording with high resolution
- Better Camera performance
- Ability to open up to 16 tabs in the web browser

History

10. Android 4.1: Jelly Bean (API 16)

Features

- Voice search and typing
- Panorama
- Project Butter
- Expandable notifications
- Daydream as a screensaver
- Power control
- Support USB audio
- Improved camera app
- Security improvements
- New gestures and accessibility features
- Multiple user accounts (Only for tablets)
- 4k resolution support
- Supporting Bluetooth with low energy
- Bi-directional text and different language support
- Set or adjust the volume of incoming calls and show a message alert
- Google displays relevant content based on your search history
- Native emoji support

History

11. Android 4.4: KitKat (API 19)

Features

- Screen Recording
- Contact Prioritization
- GPS Support
- Smarter Caller ID
- Offline music support
- UI updates for alarm and google map navigations
- Cartoonish ideograms and emojis to the Google keyboard
- KitKat has ‘OK Google’ feature that allows access to Google to the users without touching your smartphones.

History

12. Android 5.0: Lollipop (API 21)

Features

- Support ART
- Better device protection
- Notifications can be flicked away from the lock screen
- Better and improved UI
- Built-in battery saver feature
- New material design
- Revamped navigation bar
- Support for Multiple sim cards
- High definition of voice call.

History

13. Android 6.0: Marshmallow (API 23)

Features

- Support for Fingerprint readers
- Type C USB support
- Multi-window experience
- 'Sleep Mode' for saving battery life
- Clear permission system
- Custom google tabs and improved Copy-pasting

History

14. Android 7.0: Nougat (API 24)

Features

- Provide multitasking and split-screen mode
- Storage manager enhancements
- Quick setting toggles
- Display touch enhancements
- Better setting application
- Inline reply to messages and notifications without opening applications

History

15. Android 8.0: Oreo (API 26)

Features

- Password autofill
- Auto-enable Wi-Fi
- Downloadable fonts
- Multi-display support
- Support Picture-in-Picture
- Notification channels and snooze notification
- Google Play support and new emoji styling
- Adaptive icons and smart text selection

History

16. Android 9: Pie (API 28)

Features

- Sound amplifier with select to speak options
- Artificial intelligence (AI) compatibility
- Adaptive Battery and Brightness with background restrictions
- Multi-camera support with the external camera compatibility
- New Gesture Navigation and App Actions
- New Screenshot Shortcut key and accessibility menu
- Easier Screen Rotation and edge to edge screens support
- Volume and Sound enhancements
- Selectable Dark Mode
- HDR, HD audio, multiple Bluetooth connections
- Slices and long press to overview selection
- Improved Security features for extra protection
- Digital Wellbeing with app timers, dashboard and do not disturb options
- Android backups and privacy enhancements
- More Notification Information and easier text selection

History

17. Android 10: Android Q (API 29)

Features

- Support for foldable smartphones with flexible displays
- Dark mode for eyes comfortability
- Navigation control over gesture quicker and intuitive ever
- Sound amplifier with more clear sound
- Smart reply suggestions for all messaging apps
- Live caption for media playing on a smartphone
- Undo app removal
- Better notification control with many options

History

18. Android 11: (API 30)
    Native screen recording
    Muting notifications during video
    Increase touch sensitivity
    Notification History
    Auto revoke app permissions

Android SDK features

- GSM, EDGE, 3G, 4G, and LTE networks for telephony or data transfer, enabling you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks
- Comprehensive APIs for location-based services such as GPS and network-based location detection
- Full support for applications that integrate map controls as part of their user interfaces
- Wi-Fi hardware access and peer-to-peer connections
- Full multimedia hardware control, including playback and recording with the camera and microphone
- Media libraries for playing and recording a variety of audio/video or still-image formats

Android SDK features

- APIs for using sensor hardware, including accelerometers, compasses, and barometers
- Libraries for using Bluetooth and NFC hardware for peer-to-peer data transfer
- IPC message passing
- Shared data stores and APIs for contacts, social networking, calendar, and multi-media
- Background Services, applications, and processes
- Home-screen Widgets and Live Wallpaper
- The ability to integrate application search results into the system searches

Android SDK features

- An integrated open-source HTML5 WebKit-based browser
- Mobile-optimized, hardware-accelerated graphics, including a path-based 2D graphics library and support for 3D graphics using OpenGL ES 2.0
- Localization through a dynamic resource framework
- An application framework that encourages the reuse of application components and the replacement of native applications

Android SDK features

- Access to hardware, including camera, GPS, sensor
- Data transfers using Wi-Fi, Bluetooth, and NFC
- Maps, Geocoding, and Location-based services
- Background services
- SQLite database for storage and retrieval
- Shared data and inter-application communication
- Using widgets and live wallpaper to enhance the home screen
- Extensive media support and 2D/3D graphics
- Cloud to device messaging
- Optimized memory and process management.

WHAT DOES ANDROID RUN ON?

- Not just a mobile OS created for a single hardware implementation, Android is designed to support a large variety of hardware platforms, from smartphones to tablets and televisions etc.
- With no licensing fees or proprietary software, the cost to handset manufacturers for providing Android devices is comparatively low.

WHY DEVELOP FOR MOBILE?

- Today modern mobile smartphones is multifunction devices including a phone but featuring a full-featured web browser, cameras, media players, Wi-Fi, and location-based services.
- Mobile-phone ownership easily surpasses computer ownership in many countries, with more than 3 billion mobile phone users worldwide.
- 2009 marked the year that more people accessed the Internet for the first time from a mobile phone rather than a PC. Now more people are access the Internet by mobile phone rather than using personal computers.
- The increasing popularity of modern smartphones, combined with the increasing availability of highspeed mobile data and Wi-Fi hotspots, has created a huge opportunity for advanced mobile applications.

Comparison of android with other platforms

- Many of the features such as 3D graphics and native database support, are also available in other native mobile SDKs, as well as becoming available on mobile browsers.
- The following non comprehensive list details some of the features available on Android that may not be available on all modern mobile development platforms.
    - **Google Maps applications** — Android offers a Google Map as an atomic, reusable control for use in your applications. The Map View lets you display, manipulate, and annotate a Google Map within your Activities to build map-based applications using the familiar Google Maps interface.

Comparison of android with other platforms

- **Background services and applications** — Full support for background applications and services lets you create applications based on an event-driven model, working silently while other applications are being used or while your mobile sits ignored until it rings, flashes, or vibrates to get your attention.
- A service that changes your ringtone or volume depending on your current location, the time of day, and the identity of the caller.
- **Shared data and inter-process communication** — Using Intents and Content Providers, Android lets your applications exchange messages, perform processing, and share data.
- A full permission-based security mechanism should be develop.

Comparison of android with other platforms

- **All applications are created equal —** Android doesn't differentiate between native applications and those developed by third parties.
- This gives consumers unprecedented power to change the look and feel of their devices by letting them completely replace every native application with a third-party alternative that has access to the same underlying data and hardware.
- **Wi-Fi Direct and Android Beam —** inter-device communication APIs, you can include features such as instant media sharing and streaming.
- Android Beam is an NFC-based API that lets you provide support for proximity-based interaction, while Wi-Fi Direct offers a wider range peer-to-peer for reliable, high-speed communication between devices.

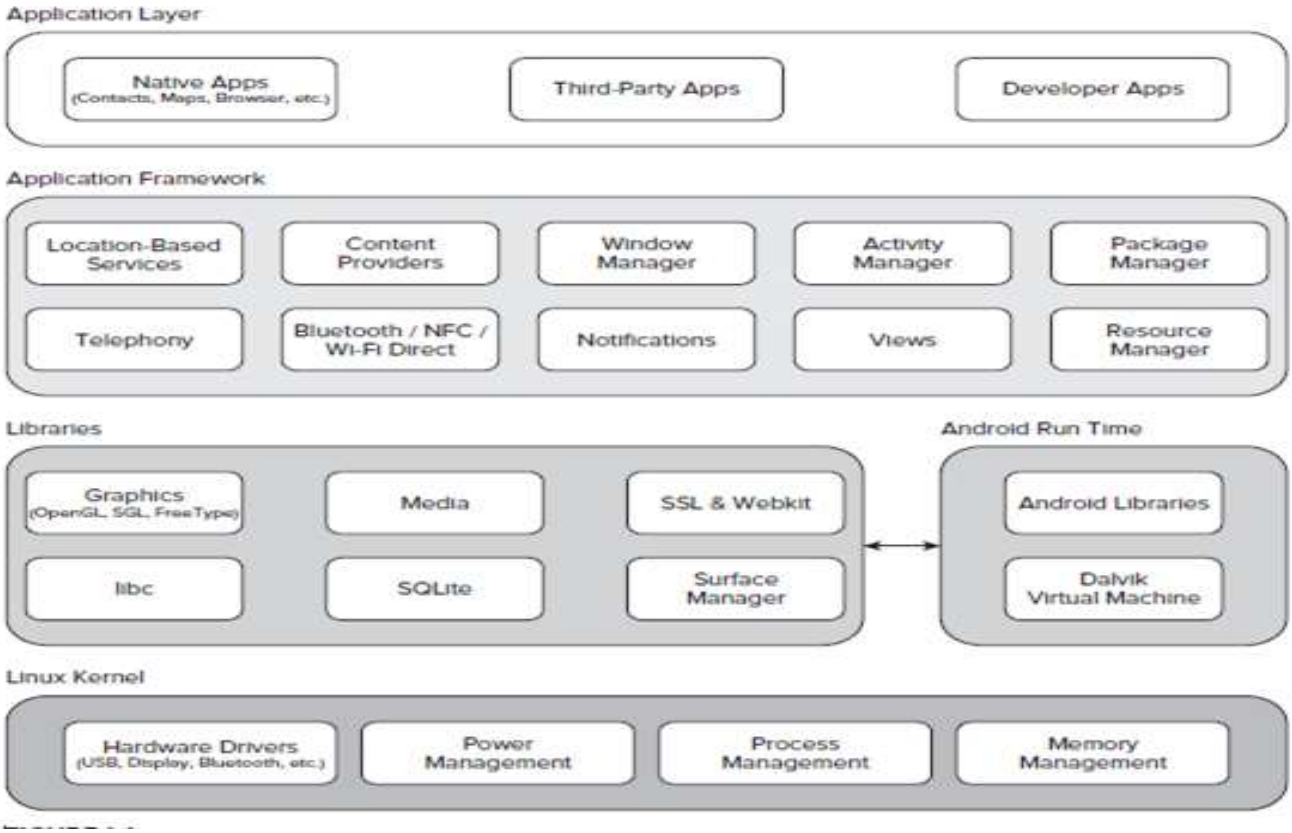
Comparison of android with other platforms

- **Home-screen Widgets, Live Wallpaper, and the quick search box —** Using Widgets and Live Wallpaper, you can create windows into your application from the phone's home screen.
- The quick search box lets you integrate search results from your application directly into the phone's search functionality

Android Software Stack

- The Android software stack is, simply, a Linux kernel and a collection of C/C++ libraries exposed through an application framework that provides services for, and management of, the run time and applications.
- The Android software stack is composed of the elements shown in Figure below

Android Software Stack



Android Software Stack

- Linux kernel — Core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.
- Libraries — Running on top of the kernel, Android includes various C/C++ core libraries such as libc and SSL, as well as the following:
  - A media library for playback of audio and video media
  - A surface manager to provide display management
  - Graphics libraries that include SGL and OpenGL for 2D and 3D graphics
  - SQLite for native database support
  - SSL and WebKit for integrated web browser and Internet security

Android Software Stack

- Android run time — The run time is what makes an Android phone an Android phone rather than a mobile Linux implementation.
- Including the core libraries and the Dalvik VM.
  - Core libraries — Although most Android application development is written using the Java language, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries, as well as the Android specific libraries.
  - Dalvik VM — Dalvik is a register-based Virtual Machine that's been optimized to ensure that a device can run multiple instances efficiently.
  - It relies on the Linux kernel for threading and low-level memory management.

Android Software Stack

- Application framework — The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- Application layer — All applications, both native and third-party, are built on the application layer by means of the same API libraries.

Android Application Architecture

- Android's architecture encourages component reuse, enabling you to publish and share Activities, Services, and data with other applications, with access managed by the security restrictions you define.
- The following application services are the architectural cornerstones of all Android applications, providing the framework you'll be using for your own software:
  - Activity Manager and Fragment Manager — Control the lifecycle of your Activities and Fragments, respectively, including management of the Activity stack
  - Views — Used to construct the user interfaces for your Activities and Fragments

Android Application Architecture

Notification Manager — Provides a consistent and nonintrusive mechanism for signaling your users

Content Providers — Lets your applications share data.

Resource Manager — Enables non-code resources, such as strings and graphics, to be externalized

Intents — Provides a mechanism for transferring data between applications and their components.

Android design philosophy demands

- Performance
- Responsiveness
- Freshness
- Security
- Seamlessness
- Accessibility

Types of Android Applications

- Foreground — An application that's useful only when it's in the foreground and is effectively suspended when it's not visible. Games are the most common examples.
- Background — An application with limited interaction that, apart from when being configured, spends most of its lifetime hidden. These applications are less common, but good examples include call screening applications, SMS auto-responders, and alarm clocks.
- Intermittent — Most well-designed applications fall into this category. At one extreme are applications that expect limited interactivity but do most of their work in the background. A common example would be a media player. At the other extreme are applications that are typically used as foreground applications but that do important work in the background. Emails and news application are examples.

Types of Android Applications

- Widgets and Live Wallpapers — Some applications are represented only as a home-screen Widget or as a Live Wallpaper.

Android Development Tools

- The Android SDK includes several tools and utilities to help you create, test, and debug your projects.
- The Android Virtual Device and SDK Managers — Used to create and manage AVDs and to download SDK packages, respectively.
  - The AVD hosts an Emulator running a particular build of Android, letting you specify the supported SDK version, screen resolution, amount of SD card storage available, and available hardware capabilities (such as touchscreens and GPS).
- The Android Emulator — An implementation of the Android VM designed to run within an AVD on your development computer. Use the Emulator to test and debug your Android applications.

Android Development Tools

- Dalvik Debug Monitoring Service (DDMS) — Use the DDMS to monitor and control the Emulators on which you're debugging your applications.
- Android Debug Bridge (ADB) — A client-server application that provides a link to virtual and physical devices. It lets you copy files, install compiled application packages (.apk), and run shell commands.
- Logcat — A utility used to view and filter the output of the Android logging system.
- Android Asset Packaging Tool (AAPT) — Constructs the distributable Android package files (.apk).

Android Development Tools

- The following additional tools are also available:
- SQLite3 — A database tool that you can use to access the SQLite database files created and used by Android.
- Traceview and dmtracedump — Graphical analysis tools for viewing the trace logs from your Android application.
- Hprof-conv — A tool that converts HPROF profiling output files into a standard format to view in your preferred profiling tool.
- MkSDCard — Creates an SD card disk image that can be used by the Emulator to simulate an external storage card.
- Dx — Converts Java .class bytecode into Android .dex bytecode.
- Hierarchy Viewer — Provides both a visual representation of a layout's View hierarchy to debug and optimize your UI, and a magnified display to get your layouts pixel-perfect.

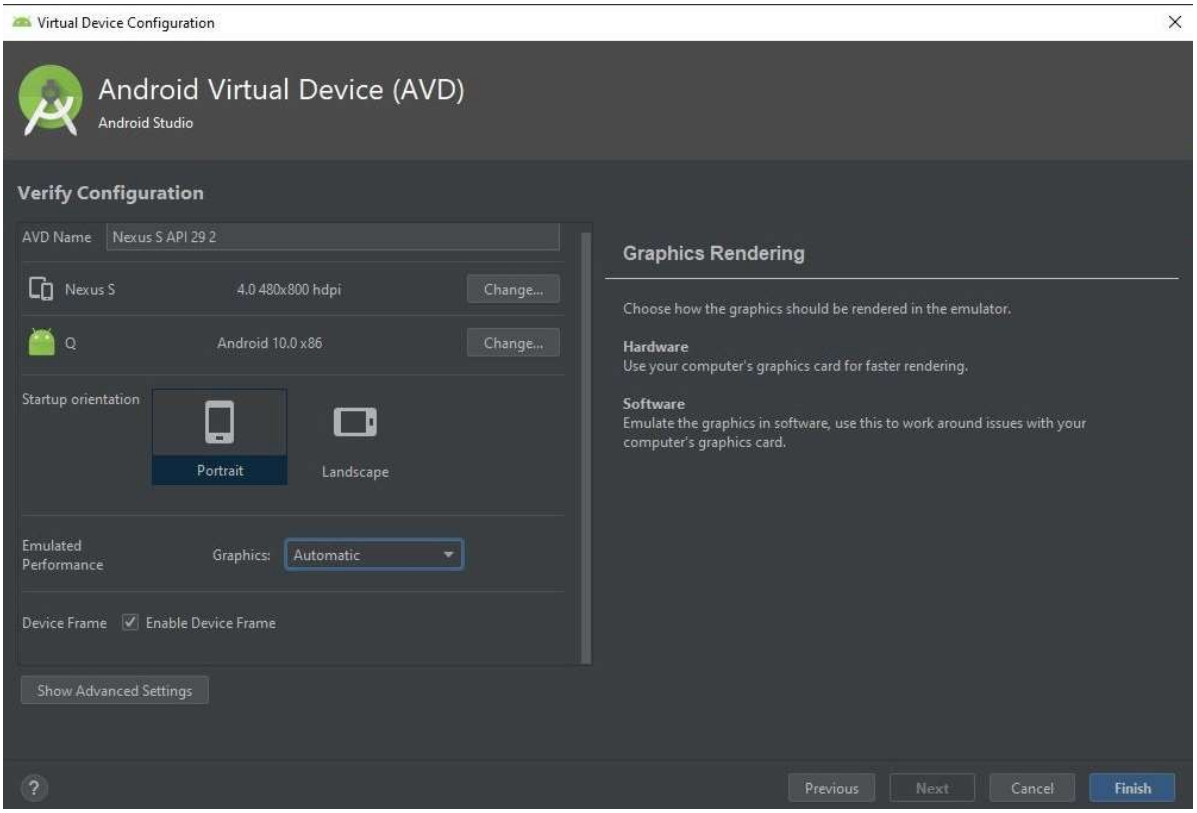
Android Development Tools

- Lint — A tool that analyzes your application and its resources to suggest improvements and optimizations.
- Draw9patch: A handy utility to simplify the creation of NinePatch graphics using a WYSIWYG editor.
- Monkey and Monkey Runner: Monkey runs within the VM, generating pseudo-random user and system events. Monkey Runner provides an API for writing programs to control the VM from outside your application.
- ProGuard — A tool to shrink and obfuscate (unclear) your code by replacing class, variable, and method names with semantically meaningless alternatives. This is useful to make your code more difficult to reverse engineer.

Android Virtual Device Manager

- The Android Virtual Device Manager is used to create and manage the virtual devices that will host instances of the Emulator.
- AVDs are used to simulate the software builds and hardware configurations available on different physical devices. This lets you test your application on a variety of hardware platforms without needing to buy a variety of phones.
- Each virtual device is configured with a name, a target build of Android (based on the SDK version it supports), an SD card capacity, and screen resolution, as shown in the Create new Android Virtual Device (AVD) dialog in Figure

Android Virtual Device Manager



Android Virtual Device Manager

- Each virtual device also supports a number of specific hardware settings and restrictions that can be added in the form of name-value pairs (NVPs) in the hardware table.
- The additional settings include the following:
  - Maximum VM heap size
  - Screen pixel density
  - SD card support
  - Existence of D-pad, touchscreen, keyboard, and trackball hardware

Android Virtual Device Manager

- Accelerometer, GPS, and proximity sensor support
- Available device memory
- Camera hardware (and resolution)
- Support for audio recording
- Existence of hardware back and home keys

Android SDK Manager

- The Android SDK Manager can be used to see which version of the SDK you have installed and to install new SDKs when they are released.
- Each platform release is displayed, along with the platform tools and a number of additional support packages.
- Each platform release includes the SDK platform, documentation, tools, and examples corresponding to that release.

Android Emulator

- The Emulator is available for testing and debugging your applications.
- The Emulator is an implementation of the Dalvik VM, making it as valid a platform for running Android applications as any Android phone.
- Because it's decoupled from any particular hardware, it's an excellent baseline to use for testing your applications.
- Full network connectivity is provided along with the ability to tweak the Internet connection speed and latency while debugging your applications.
- You can also simulate placing and receiving voice calls and SMS messages.

Dalvik Debug Monitor Service

- The Emulator enables you to see how your application will look, behave, and interact, but to actually see what's happening under the surface, you need the Dalvik Debug Monitoring Service.
- The DDMS is a powerful debugging tool that lets you interrogate active processes, view the stack and heap, watch and pause active threads, and explore the filesystem of any connected Android device.

Android Debug Bridge

- The Android Debug Bridge (ADB) is a client-service application that lets you connect with an Android device (virtual or actual).
- It's made up of three components:
  - A daemon running on the device or Emulator
  - A service that runs on your development computer
  - Client applications (such as the DDMS) that communicate with the daemon through the service
- As a communications conduit between your development hardware and the Android device/Emulator, the ADB lets you install applications, push and pull files, and run shell commands on the target device.

Android Debug Bridge

- Using the device shell, you can change logging settings and query or modify SQLite databases available on the device.
- The ADT tool automates and simplifies a lot of the usual interaction with the ADB, including application installation and updating, file logging, and file transfer (through the DDMS perspective).

Hierarchy Viewer and Lint tool

- To build applications that are fast and responsive, you need to optimize your UI.
- The Hierarchy Viewer and Lint tools help you analyze, debug, and optimize the XML layout definitions used within your application.
- The Hierarchy Viewer displays a visual representation of the structure of your UI layout.
- Starting at the root node, the children of each nested View (including layouts) is displayed in a hierarchy.
- Each View node includes its name, appearance, and identifier.
- To optimize performance, the performance of the layout, measure, and draw steps of creating the UI of each View at runtime is displayed.

Hierarchy Viewer and Lint tool

- Using these values, you can learn the actual time taken to create each View within your hierarchy, with colored “traffic light” indicators showing the relative performance for each step.
- You can then search within your layout for Views that appear to be taking longer to render than they should.
- The Lint tool helps you to optimize your layouts by checking them for a series of common inefficiencies that can have a negative impact on your application’s performance.
- Common issues include a surplus of nested layouts, a surplus of Views within a layout, and unnecessary parent Views.

Monkey and Monkey Runner

- Monkey and Monkey Runner can be used to test your applications stability from a UI perspective.
- Monkey works from within the ADB shell, sending a stream of pseudo-random system and UI events to your application.
- It's particularly useful to stress test your applications to investigate edgecases you might not have anticipated through unconventional use of the UI.
- Alternatively, Monkey Runner is a Python scripting API that lets you send specific UI commands to control an Emulator or device from outside the application.
- It's extremely useful for performing UI, functional, and unit tests in a predictable, repeatable fashion.