Android Environment
Emulator

Victor Matos
Cleveland State University

Notes are based on:
Android Emulator

• The Android SDK includes a mobile device emulator -- a virtual mobile device that runs on your computer.

• The emulator lets you prototype, develop, and test Android applications without using a physical device.

• The Android emulator mimics all of the hardware and software features of a typical mobile device, except that it can not receive or place actual phone calls.

• It provides a variety of navigation and control keys, which you can "press" using your mouse or keyboard to generate events for your application.

• It also provides a screen in which your application is displayed, together with any other Android applications running.
Android Emulator v1.5 Skin
Android Emulator v1.6 Skin
Android Emulator v1.6 Skin

Tab
Launch Pad

Status Bar – Notification Line

Volume

Power

Hang up

Call

Home

Menu

Back

5554:AVD16GoogleAPI4

Adonde
Alarm Clock
Browser
Calculator

Camcorder
Camera
Contacts
Currency1

Custom Locale
Dev Tools
Dialer
Email

Gallery
Gestures Builder
Import Contacts
Maps

1 ! 2 @ 3 # 4 $ 5 % 6 7 8 9 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M

ALT SYM @
# Android Emulator

## Controlling the Android Emulator through keyboard keys

Keypad keys only work when *NumLock* is deactivated.

<table>
<thead>
<tr>
<th>Keyboard</th>
<th>OS function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Escape</td>
<td>Back button</td>
</tr>
<tr>
<td>Home</td>
<td>Home button</td>
</tr>
<tr>
<td>F2, PageUp</td>
<td>Menu (Soft-Left) button</td>
</tr>
<tr>
<td>Shift-F2, PageDown</td>
<td>Start (Soft-Right) button</td>
</tr>
<tr>
<td>F3</td>
<td>Call/Dial button</td>
</tr>
<tr>
<td>F4</td>
<td>Hangup / EndCall button</td>
</tr>
<tr>
<td>F5</td>
<td>Search button</td>
</tr>
<tr>
<td>F7</td>
<td>Power button</td>
</tr>
<tr>
<td>Ctrl-F3, Ctrl-KEYPAD_5</td>
<td>Camera button</td>
</tr>
<tr>
<td>Ctrl-F5, KEYPAD_PLUS</td>
<td>Volume up button</td>
</tr>
<tr>
<td>Ctrl-F6, KEYPAD_MINUS</td>
<td>Volume down button</td>
</tr>
<tr>
<td>KEYPAD_5</td>
<td>DPad center</td>
</tr>
<tr>
<td>KEYPAD_4</td>
<td>DPad left</td>
</tr>
<tr>
<td>KEYPAD_6</td>
<td>DPad right</td>
</tr>
<tr>
<td>KEYPAD_8</td>
<td>DPad up</td>
</tr>
<tr>
<td>KEYPAD_2</td>
<td>DPad down</td>
</tr>
<tr>
<td>F8</td>
<td>toggle cell network on/off</td>
</tr>
<tr>
<td>F9</td>
<td>toggle code profiling (when -trace option set)</td>
</tr>
<tr>
<td>Alt-ENTER</td>
<td>toggle FullScreen mode</td>
</tr>
<tr>
<td>Ctrl-T</td>
<td>toggle trackball mode</td>
</tr>
<tr>
<td>Ctrl-F11, KEYPAD_7</td>
<td>switch to previous layout</td>
</tr>
<tr>
<td>Ctrl-F12, KEYPAD_9</td>
<td>switch to next layout</td>
</tr>
</tbody>
</table>
Android Emulator

Features - Emulating First Generation Android Phones

The Android emulator supports many hardware features likely to be found on mobile devices (such as the HTC-G1), including:

1. An ARMv5 CPU and the corresponding memory-management unit (MMU)
2. A 16-bit LCD display (mimicking 360 x 480 pixels)
3. One or more keyboards (a Qwerty-based keyboard and associated Dpad/Phone buttons)
4. A sound chip with output and input capabilities
5. Flash memory partitions (emulated through disk image files on the development machine)
6. A GSM modem, including a simulated SIM Card
Android Emulator

Nexus One (newer Google developer phone)

Some phones in the market already surpass these specs (Fall 2010)

<table>
<thead>
<tr>
<th>Size and weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
</tr>
<tr>
<td>Width</td>
</tr>
<tr>
<td>Depth</td>
</tr>
<tr>
<td>Weight</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7 inch (diagonal) widescreen WVGA AMOLED touchscreen</td>
</tr>
<tr>
<td>800 x 480 pixels</td>
</tr>
<tr>
<td>100,000:1 typical contrast ratio</td>
</tr>
<tr>
<td>1ms typical response rate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power and battery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removable 1400 mAh battery</td>
</tr>
<tr>
<td>Charges at 480 mA from USB, at 580 mA from supplied charger</td>
</tr>
<tr>
<td>Talk time</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Standby time</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Internet use</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Video playback</td>
</tr>
<tr>
<td>Audio playback</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualcomm QSD 8250 1 GHz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Mobile Technology Platform 2.1 (Eclair)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>512 MB Flash</td>
</tr>
<tr>
<td>512 MB RAM</td>
</tr>
<tr>
<td>4 GB Micro SD Card (Expandable to 32 GB)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cellular &amp; Wireless</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 UMTS bands (either 900/AWS/2100 MHz or 850/1900/2100 MHz)</td>
</tr>
<tr>
<td>HSDPA 7.2 Mbps</td>
</tr>
<tr>
<td>HSUPA 2 Mbps</td>
</tr>
<tr>
<td>GSM/EDGE (850, 900, 1800, 1900 MHz)</td>
</tr>
<tr>
<td>Wi-Fi (802.11b/g)</td>
</tr>
<tr>
<td>Bluetooth 2.1 + EDR</td>
</tr>
<tr>
<td>A2DP stereo Bluetooth</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assisted global positioning system (AGPS) receiver</td>
</tr>
<tr>
<td>Cell tower and Wi-Fi positioning</td>
</tr>
<tr>
<td>Digital compass</td>
</tr>
<tr>
<td>Accelerometer</td>
</tr>
</tbody>
</table>

---

8
Android Emulator

Working with Emulator Disk Images
The emulator uses mountable disk images (*ANDROID SYSTEM IMAGE*) stored on your development machine to simulate flash (or similar) partitions on an actual device.

For example, it uses **disk images** containing
(1) an emulator-specific kernel,
(2) the Android system,
(3) a ram-disk image, and
(4) writeable images for user data and simulated SD card.

_**By default, the Emulator always looks for the disk images in the private storage area of the AVD in use (**c:\android-sdk-windows\platform\** ...)**_
Android Emulator

Working with Emulator Disk Images

If no platform images exist there when the Emulator is launched, it creates the images in the AVD directory based on default versions stored in the SDK.

Note:
The default storage location for AVDs is in

`~/.android/avd` on OS X and Linux,
`C:\Documents and Settings\<user>\.android\avd\`... on Windows XP, and
`C:\Users\<user>\.android\` on Windows Vista.
Android Emulator

Creating an AVD using the android tool

Listing targets
To generate a list of system image targets, use this command:

```
android list targets
```
Starting – Stopping the Emulator

To start an instance of the emulator from the command line, change to the tools/ folder of the SDK. Enter emulator command like this:

```
emulator -avd <avd_name>
```

This initializes the emulator and loads an AVD configuration. After a few seconds you will see the emulator window appear on your screen.

If you are working in Eclipse, the ADT plugin for Eclipse installs your application and starts the emulator automatically, when you run or debug the application.

To stop an emulator instance, just close the emulator's window.

To list all available AVDs enter DOS command

```
android list avd
```
Android Emulator

AVD - Android Virtual Devices

Android Virtual Devices (AVDs) are configurations of emulator options that let you better model an actual device.

Each AVD is made up of:

- **A hardware profile.** You can set options to define the hardware features of the virtual device. For example, you can define whether the device has a camera, whether it uses a physical QWERTY keyboard or a dialing pad, how much memory it has, and so on.
- **A mapping to a system image.** You can define what version of the Android platform will run on the virtual device. You can choose a version of the standard Android platform or the system image packaged with an SDK add-on.
- **Other options.** You can specify the emulator skin you want to use with the AVD, which lets you control the screen dimensions, appearance, and so on. You can also specify the emulated SD card to use with the AVD.
- **A dedicated storage area** on your development machine, in which is stored the device's user data (installed applications, settings, and so on) and emulated SD card.
Android Emulator

AVD - Android Virtual Devices

You can create as many AVDs as you need, based on the types of devices you want to model and the Android platforms and external libraries you want to run your application on.
Creating an AVD using the Eclipse-ADT Tool

From Eclipse, follow the sequence: Main menu (AVD Manager) > Virtual Devices > New >

Provide a Name, choose an Android target, create a new SD card with about 2Gb, choose a screen type, add hardware devices…

Click on: Create AVD (wait, it takes several minutes to format the new SD card)
Creating an AVD using the android tool

When creating an AVD, you simply specify the -c option, like this:

```
android create avd -n <avd_name> -t <targetID> -c <size>[K|M]
```

The -t (target) argument sets up a mapping between the AVD and the system image that you want to use whenever the AVD is invoked. Later, when applications use the AVD, they'll be running on the system that you specify in the -t argument.

To specify the system image to use, you refer to its target ID — an integer — as assigned by the android tool. The target ID is not derived from the system image name, version, or API Level, or other attribute, so you need to have the android tool list the available system images and the target ID of each, as described in the next section. You should do this before you run the android create avd command.
Example: Creating an AVD using the android tool

After listing all targets (see previous image) we have decided to make a profile based on target id:4 to support SDK1.6 with Google API Mapping libraries. It should also include a 1Gig SD card. We enter the command

```
android create avd -n myAVD4SD1G -t 4 -c 1024M
```
Example: Creating an AVD using the android tool

Verifying what AVDs are available in the system:

```
Available Android Virtual Devices:
  Name: AvdApi3Id3MatosSDcard
  Path: C:\Documents and Settings\Administrator\.android\avd\AvdApi3Id3MatosSDcard.avd
  Target: Google APIs (Google Inc.)
  Based on Android 1.5 (API level 3)
  Skin: HVGA
  Sdcard: c:/Android_Emulator_Data/mysdcard.img

Name: myAVD4SD1G
  Path: C:\Documents and Settings\Administrator\.android\avd\myAVD4SD1G.avd
  Target: Google APIs (Google Inc.)
  Based on Android 1.6 (API level 4)
  Skin: HVGA
  Sdcard: 1024M
```
SD Card Emulation

- You can create a disk image and then load it to the emulator at startup, to simulate the presence of a user's SD card in the device.
- The emulator supports emulated SDHC cards, so you can create an SD card image of any size up to 128 gigabytes.
- You can browse, send files to, and copy/remove files from a simulated SD card either with `adb` or the emulator.

Creating an SD card image using `mksdcard`

Use the `mksdcard` tool, included in the SDK, to create a FAT32 disk images.

```
mksdcard <size> <file>
```

For example:

```
mksdcard 1024M c:/temp/mysdcard.iso
```
Android Emulator

**Android Emulator – How to use the SDCARD device**

The general syntax to create an SD card is

\[ \text{mksdcard [ -l label ] <size> <file>} \]

- The tool *mksdcard* is part of the Android SDK. The SD *label* is optional.
- The device’s size is expressed as an integer number followed by either K (kilobytes) or M (megabytes).

**Example**: Create a 1GB SDcard device using the following command

```
  mksdcard 1024M c:\mysdcard.img
```

Run the emulator with the command

```
  emulator -sdcard c:\mysdcard.img
```
or alternatively

```
  emulator -avd myAvdFile
```
Android Emulator

Moving Data, Music and Pictures to the Sdcard

1. Use the program **ddms** to push files into the SDcard (the emulator must be running with the SD card attached to it).

2. Click on: **Device > File Explorer**, this will open a new window and there you will select the SDcard.

3. Now you move data to the sdcard. Your options are
   - Open a Windows **Explore** panel to *drag & drop* files/folders on the card, or
   - Press on the button "**Push File onto Device**"
     (see upper left icons: *push, pull, delete*).

(DDMS stands for Dalvik Debug Monitor Services. The program is located in the /tools folder of the SDK. Also available in Eclipse perspective – Top upper right icons)
Android Emulator

Moving Data, Music and Pictures to the SDcard
Android Emulator

Moving Data, Music and Pictures to the SDcard

4. Return to the emulator. This time you will see the selected (music) files in the SDcard
Android Emulator

Moving Data, Music and Pictures to the SDcard

5. Pictures appear by clicking the *Application Pad* and invoking the *Gallery* application
Android Emulator

**Android – Login into the OS shell**

You can log into the OS Linux version of Android executing in the emulator and issue selected commands.

1. Run the Android emulator
2. Run **adb** application as follows: 
   c:> adb shell

(adb is the Android Debug Bridge app. It is Located in the /tools folder of the SDK)
Android Emulator

Android – Login into the OS shell

If more than one emulator is running (or your phone is physically connected to the computer using the USB cable) you need to identify the target.

Follow the steps:

1. Get a list of all active emulators
   
   **adb devices**
   
   List of devices attached
   
   emulator-5554  device
   
   emulator-5556  device
   
   HT845GZ45737  device

2. Run **adb** application as follows:
   
   **adb  -s emulator-5554  shell**

*(**adb** is the Android Debug Bridge app. It is Located in the /tools folder of the SDK)*
NOTE1: Emulators & Hardware Devices
You may test your applications in either a software emulator or a hardware device.

All you need to do is connect your phone to the computer via USB cable.

On a command shell type the command: “adb devices“ you should see something like “HT845GZ45737 device” indicating the presence of your hardware device.

Gaining Root Access to Your Hardware device
A developer’s phone such as the G1 comes with root access enabled and is fully opened.

Run the terminal application ( adb shell ) and see if you have the # prompt; if not try the command su. It should give you the root prompt, if you have a permission denied error then you do not have root access.
NOTE2: Moving an app from (Rooted) Hardware to Emulator

If you want to transfer an app installed in your developer’s phone to the emulator, follow the next steps:

1. Run command shell: > `adb devices` (find out the id of your hardware, say `HT845GZ45737`)

2. Pull the file from the device to your computer’s file system. Enter the command `adb -s HT845GZ45737 pull data/app/theInstalled.apk  c:/theInstalled.apk`

3. Disconnect your Android phone

4. Run an instance of the Emulator

5. Now install the app on the emulator using the command `adb -s emulator-5554  install c:\theInstalledApp.apk`

You should see a message indicating the size of the installed package, and `Success`. 
Android Emulator

Android – Login into the OS shell

3. Android accepts a number of Linux shell commands including the useful set below

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls</td>
<td>show directory (alphabetical order)</td>
</tr>
<tr>
<td>mkdir</td>
<td>make a directory</td>
</tr>
<tr>
<td>rmdir</td>
<td>remove directory</td>
</tr>
<tr>
<td>rm -r</td>
<td>to delete folders with files</td>
</tr>
<tr>
<td>rm</td>
<td>remove files</td>
</tr>
<tr>
<td>mv</td>
<td>moving and renaming files</td>
</tr>
<tr>
<td>cat</td>
<td>displaying short files</td>
</tr>
<tr>
<td>cd</td>
<td>change current directory</td>
</tr>
<tr>
<td>pwd</td>
<td>find out what directory you are in</td>
</tr>
<tr>
<td>df</td>
<td>shows available disk space</td>
</tr>
<tr>
<td>chmod</td>
<td>changes permissions on a file</td>
</tr>
<tr>
<td>date</td>
<td>display date</td>
</tr>
<tr>
<td>exit</td>
<td>terminate session</td>
</tr>
</tbody>
</table>
Android Emulator

Android – Login into the OS shell

4. There is no copy (cp) command in Android, but you could use the cat instead. For instance:

```
# cat data/app/theInstalledApp.apk > cache/theInstalledApp.apk
```
Android Emulator

Using the Emulator with “inserted” SD card from Eclipse

From Eclipse’s menu create new launch configuration:

Run > Run Configurations > New icon

On the Target panel
1. Select existing Android Virtual device (AVD)
2. Enter additional Command Line Options (see caption)
3. Apply > Run

Additional Emulator Command Line Options:
-sdcard c:\Android_Emulator_Data\mysdcard.img -datadir c:\Android_Emulator_Data
Android Emulator

Sending Text Messages to the Emulator

1. Start the emulator.
2. Open a new shell and type:
   ```cmd
   c:> adb devices
   ```
   so you know the emulator’s numeric port id (usually 5554, 5556, and so on)

3. Connect to the console using telnet command like:
   ```cmd
   c:> telnet localhost 5554
   ```

4. After receiving the telnet prompt you can send a text message with the command (no quotes needed for the message)
   ```cmd
   sms send <Sender’s phone number> <text message>
   ```
Android Emulator

Example:
Sending Text Messages to the Emulator

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600] (C) Copyright 1985-2001 Microsoft Corp.
c:\Android\tools>telnet localhost 5554
```

```
Telnet localhost
Android Console: type 'help' for a list of commands
OK
sms send 5551122 don't forget we have a ballroom class tonight at 8:30pm
OK
```

Beatriz Aurenty: don't forget we have a ballroom class tonight at 8:30pm
Sent: 2:44AM
Making a Voice Call to the Emulator

1. Start the emulator.
2. Open a new shell and type:
   `adb devices`
   to know the emulator’s numeric port id (usually 5554, 5556, and so on)

3. Connect to the console using telnet command like:
   `telnet localhost 5554` (this is the ‘number’ to be called)

4. After receiving the telnet prompt you can place a call (voice) with the command
   `gsm call <caller’s phone number>`
Android Emulator

Example: Making a Phone Call to the Emulator

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\Android\tools>telnet localhost 5554
```

```
Telnet localhost
Android Console: type 'help' for a list of commands
OK
gsm call 5551122
OK
```
Emulator Control
With these controls, you can simulate special device states and activities. Features include:

1. **Telephony Status** - change the state of the phone's Voice and Data plans (home, roaming, searching, etc.), and simulate different kinds of network Speed and Latency (GPRS, EDGE, UTMS, etc.).

2. **Telephony Actions** - perform simulated phone calls and SMS messages to the emulator.

3. **Location Controls** - send mock location data to the emulator so that you can perform location-aware operations like GPS mapping. To use the Location Controls, launch your application in the Android emulator and open DDMS. Click the Emulator Controls tab and scroll down to Location Controls. From here, you can:
   - Manually send individual longitude/latitude coordinates to the device. Click **Manual**, select the coordinate format, fill in the fields and click **Send**.
   - Use a GPX file describing a route for playback to the device.
Android Emulator

Using Eclipse to test Emulator’s Telephony Actions

[Image of a computer screen with a window open to the DDMS tool in Eclipse, showing a list of processes and a dialog box for sending a message.]
Android Emulator

Questions ?
Appendix 1 – Connecting your Hardware Device to the Computer

1. Use a mini-USB cable to link the device and your computer
2. Expand the Notification bar
3. Mount the device

You could now use the Eclipse-ADT-File Explorer panel to pull/push files to the device.
Android Emulator

Appendix 1 – Emulator to Emulator Communication

1. Run two instances of the emulator (typical IDs are: 5554, 5556, … )
2. Dial (or send SMS) from one of them (say 5554) to the other (5556)
3. Press the Green/Red call buttons to accept/terminate the call
4. Try sending SMS (use numbers 5554 and 5556)