

[PRODUCTS](#) [DOWNLOADS](#) [TUTORIALS](#) [SUPPORT](#) [CONTACT](#)

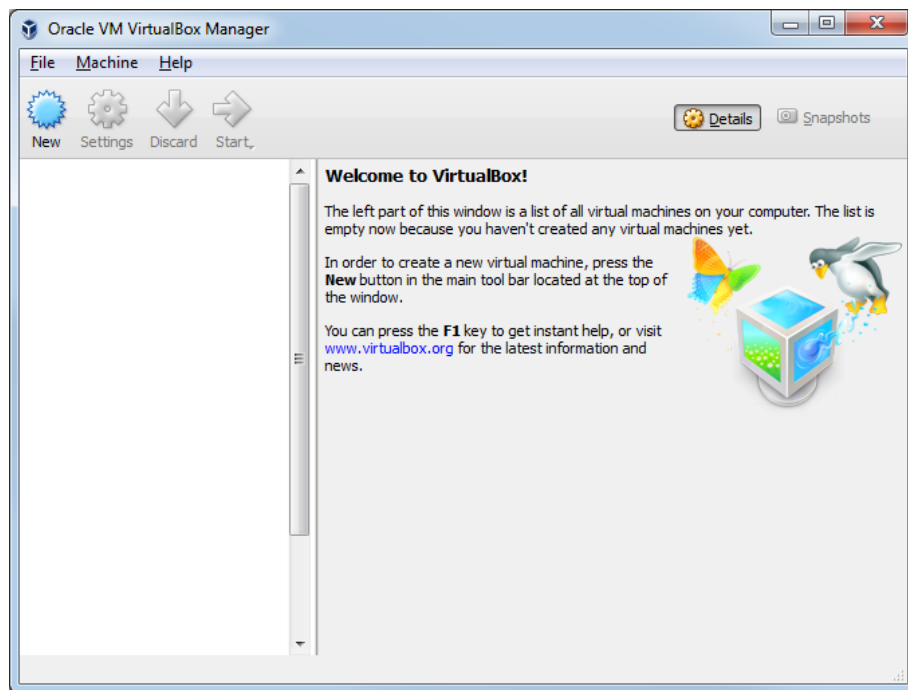
Tutorials > Android > Integration with other tools > Accelerating Android App debugging with VirtualBox

Accelerating Android App debugging with VirtualBox

📅 September 23, 2015 📁 android, opengl, virtualbox

This tutorial shows how to use [VirtualBox](#) to accelerate debugging of Android apps with native components. We will show how to create an Android VirtualBox VM, configure it for debugging and use VisualGDB to debug the San-Angeles project. Before you begin, download the latest VirtualBox and get an x86 Android ISO from the [android-x86.org](#) website.

1. Install and run VirtualBox. Click the “New” button to create a new VM:



2. Select “Linux -> Other Linux (32-bit)” as the machine type:



Categories

Tutorials

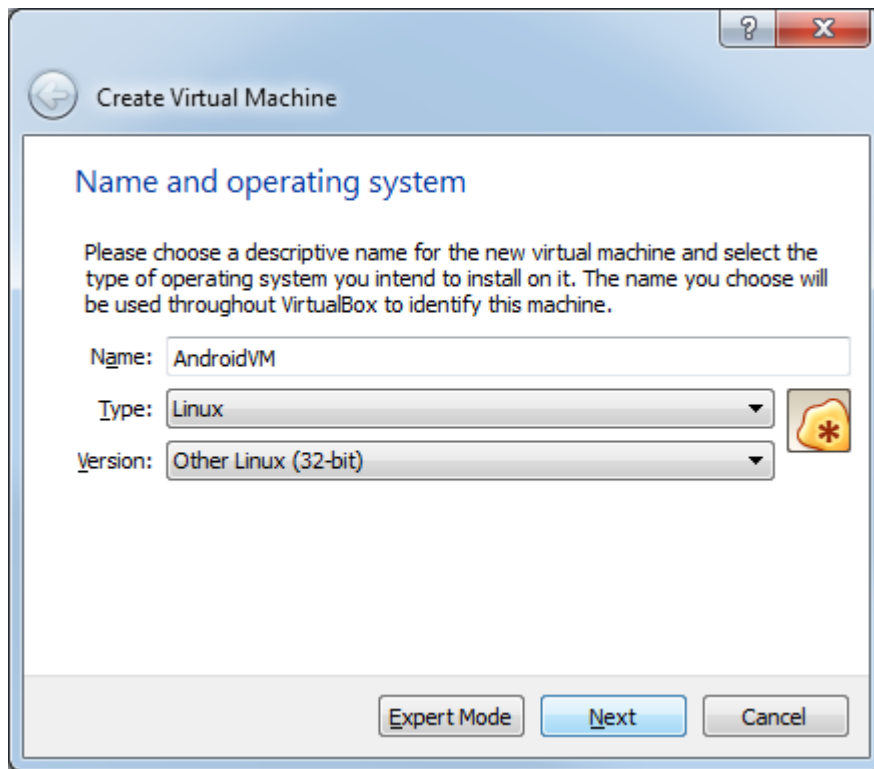
[Android](#)[Cocos2d-x](#)[Integration with other tools](#)

Archive

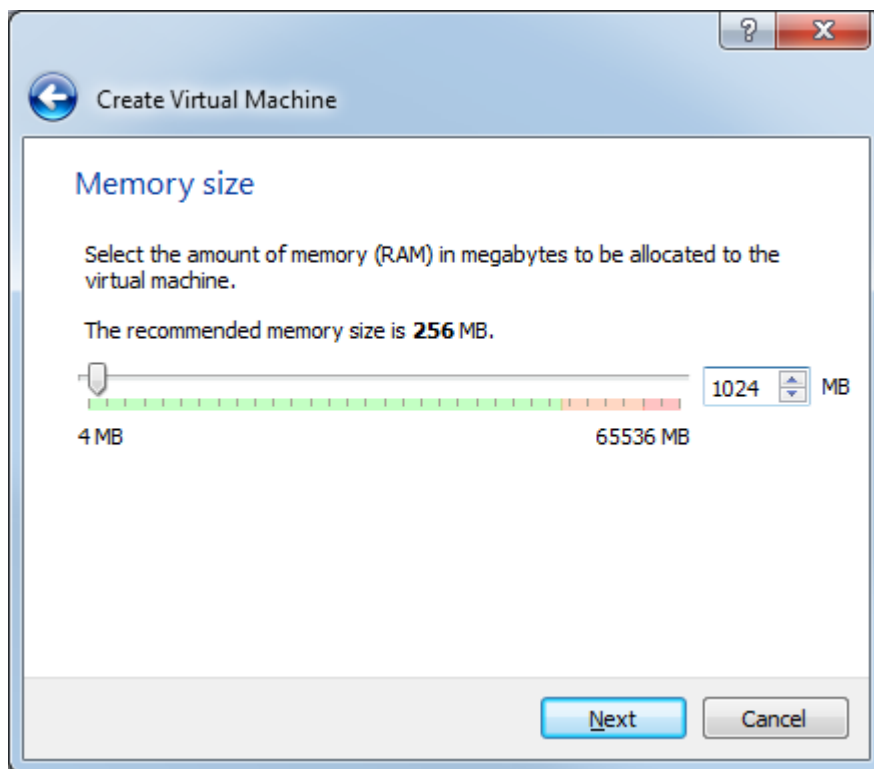
[Android](#)[Embedded](#)[Linux](#)[Arduino](#)[CMake](#)[Continuous Integration](#)[Customization](#)[FreeBSD](#)[SDK](#)

Embedded

[ARM Features](#)[ESP8266/ESP32](#)[Getting Started with Boards](#)[Internet of Things](#)[mbed](#)[MSP430](#)[RTOS](#)



3. Proceed with the default memory size:



4. Select "Create a virtual hard disk now":

[STM32 Boards & Tools](#)

[STM32 Peripherals](#)

[IntelliSense](#)

[Linux](#)

[Beaglebone](#)

[Cubieboard](#)

[Linux Frameworks & Tools](#)

[Raspberry Pi](#)

[Porting](#)

[Profiler](#)

[Embedded](#)

[Linux](#)

[Real-Time Watch](#)

[Unit Tests](#)

[Windows](#)

[Cygwin](#)

[MinGW](#)

[Uncategorised](#)

[Tags](#)

[android](#) [android](#)

[samples](#) [arduino](#) [arm](#)

[beaglebone](#) [bluetooth](#)

[cmake](#) [cross-compile](#)

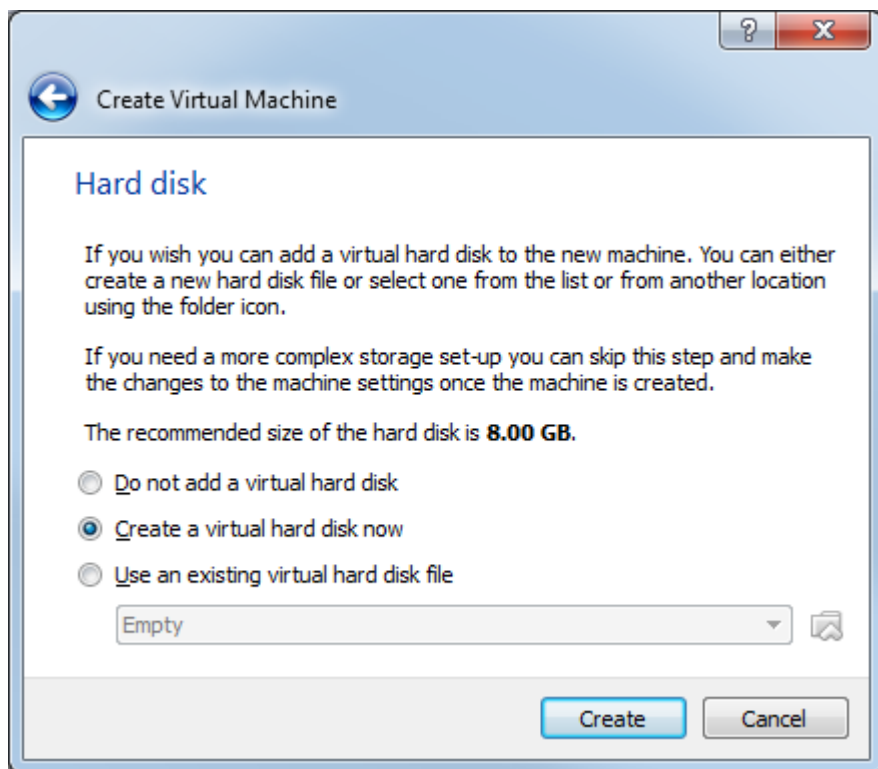
[custom](#) [cygwin](#) [embedded](#)

[esp32](#) [esp8266](#) [freertos](#)

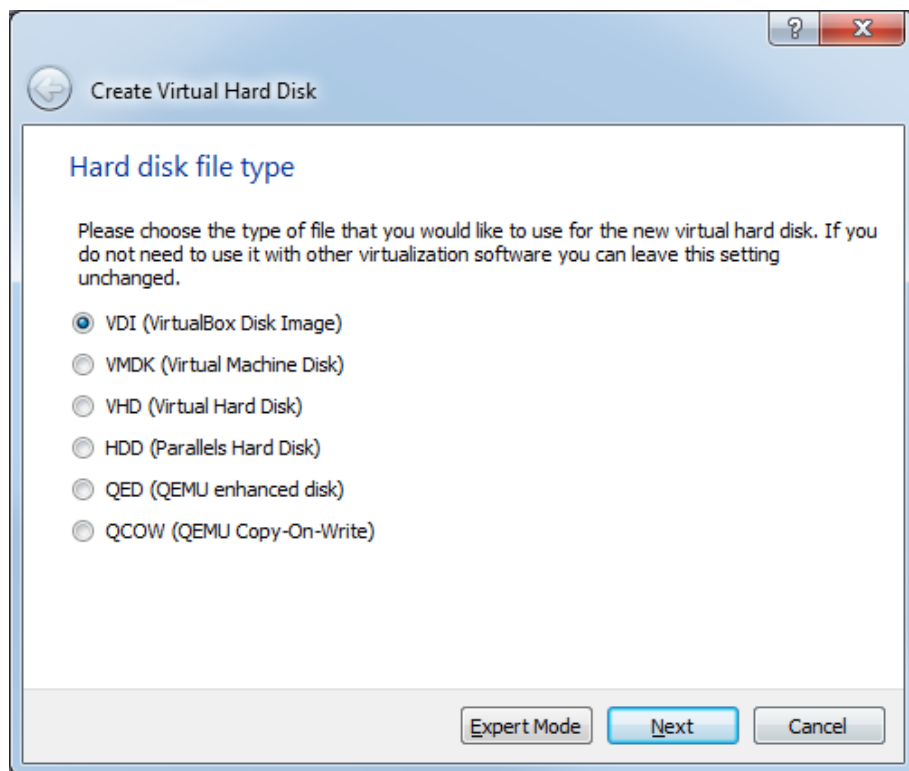
[HTTP](#) [import](#) [IoT](#) [keil](#)

[kinetis](#) [led](#) [library](#) [linux](#)

[mbed](#) [mingw](#) [msbuild](#) [msp430](#)

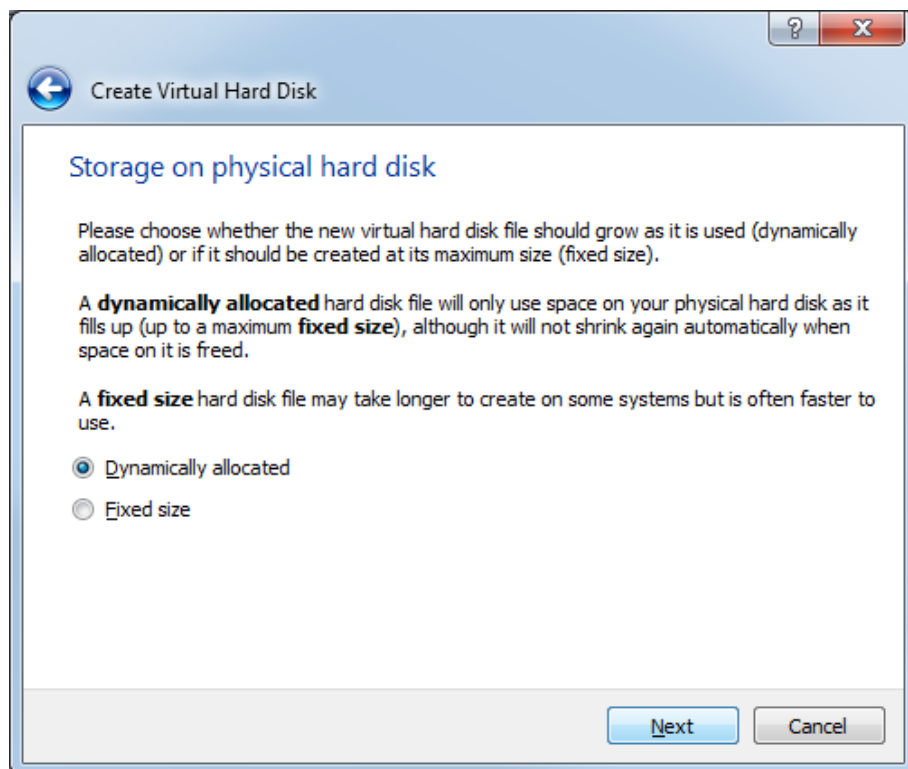


5. Proceed with the default virtual disk format:

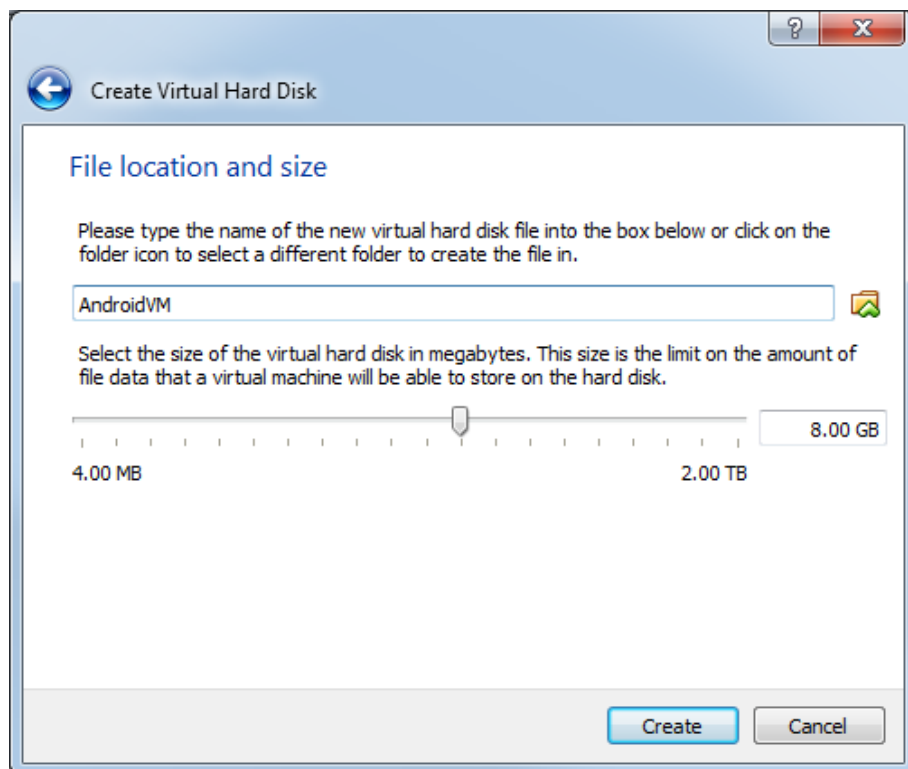


6. On the next page select "dynamically allocated":

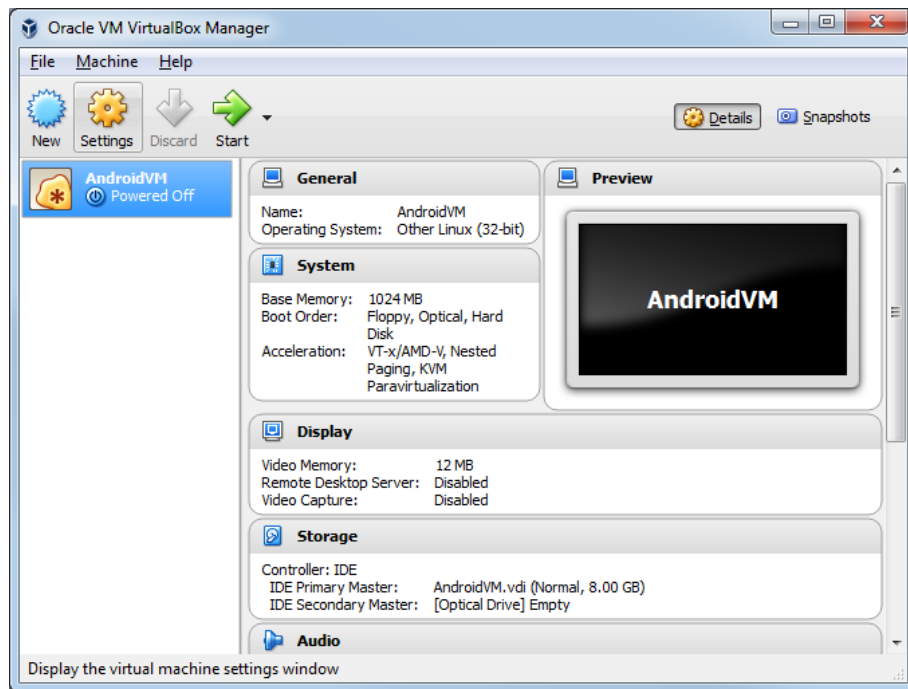
[nrf51](#) [openocd](#) [porting](#)
[profiler](#) [python](#) [qt](#) [quickdebug](#)
[quickstart](#) [raspberrypi](#)
[raspberrypi](#) [rtx](#) [ssh](#)
[stm32](#) [tests](#) [uart](#)
[v51_features](#)
[v52_features](#) [WiFi](#) [win32](#)



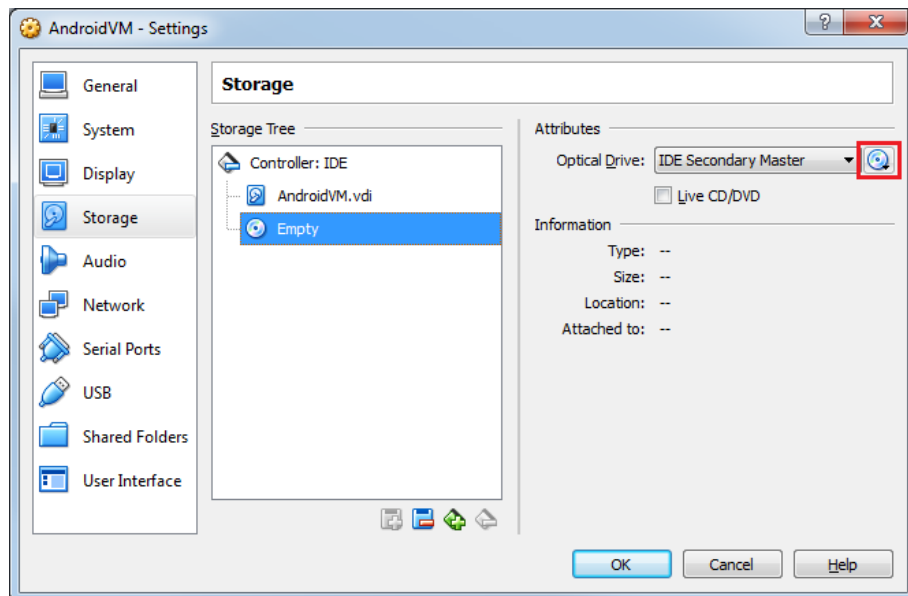
7. Finally you can customize the disk size. The default value of 8GB should be enough for most cases:



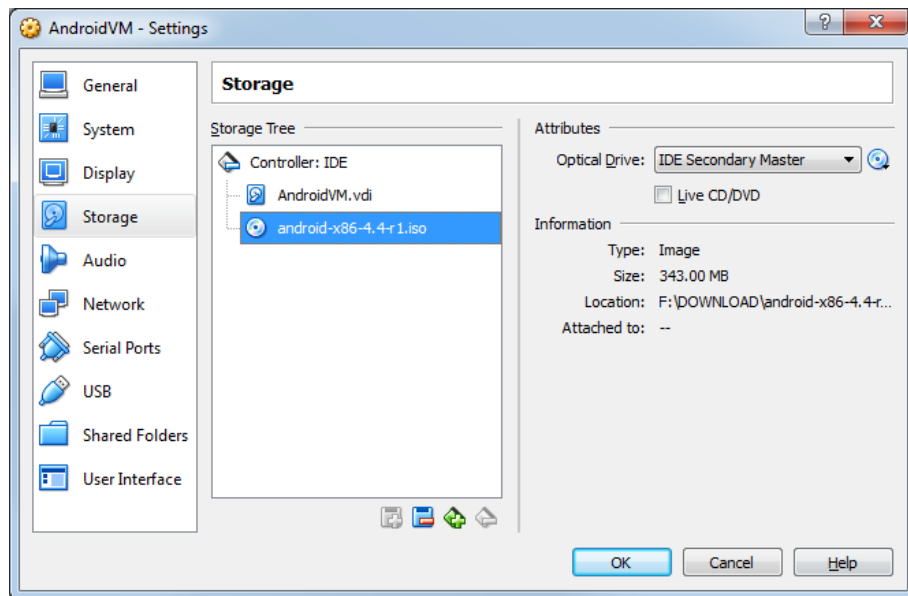
8. Before you can start installing the Android OS into your VM, you need to mount the ISO file in it. Click "Settings" to open the VM settings:



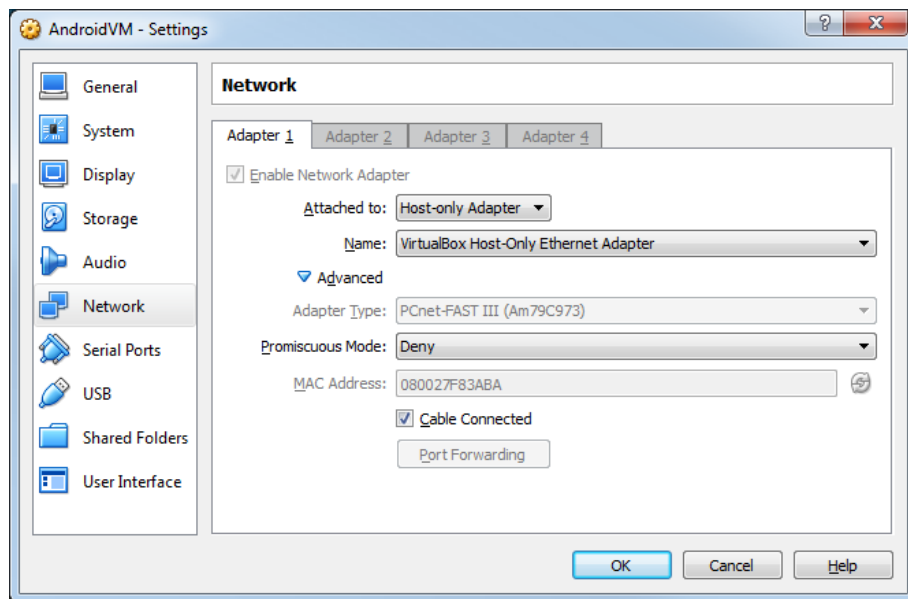
9. Go to the “Storage” tab, select the empty CD-ROM device and click the disc icon to browse for a disc image:



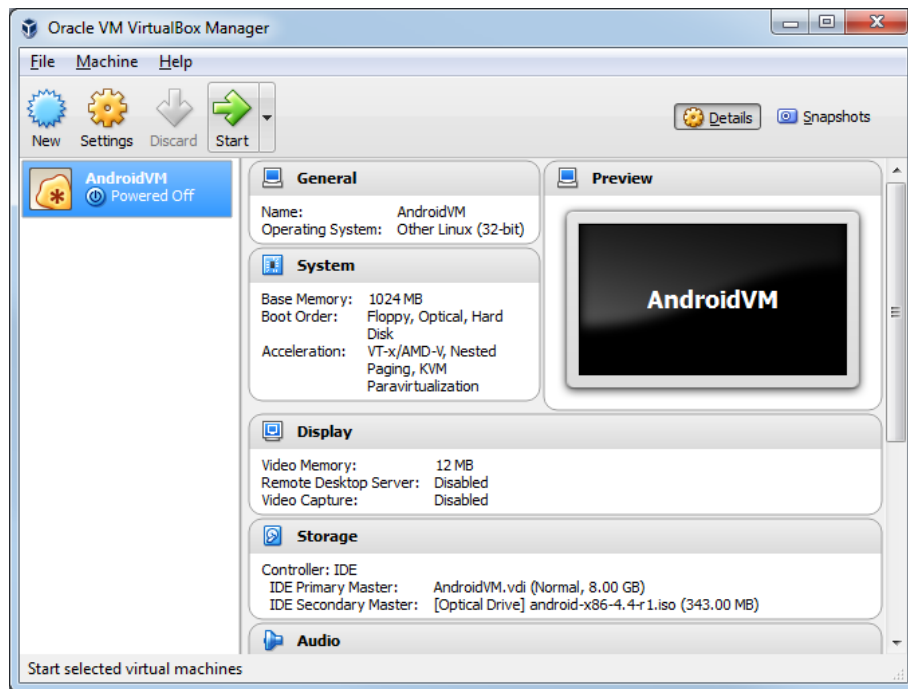
10. Specify the path to the Android image you downloaded. It will appear in the Settings window:



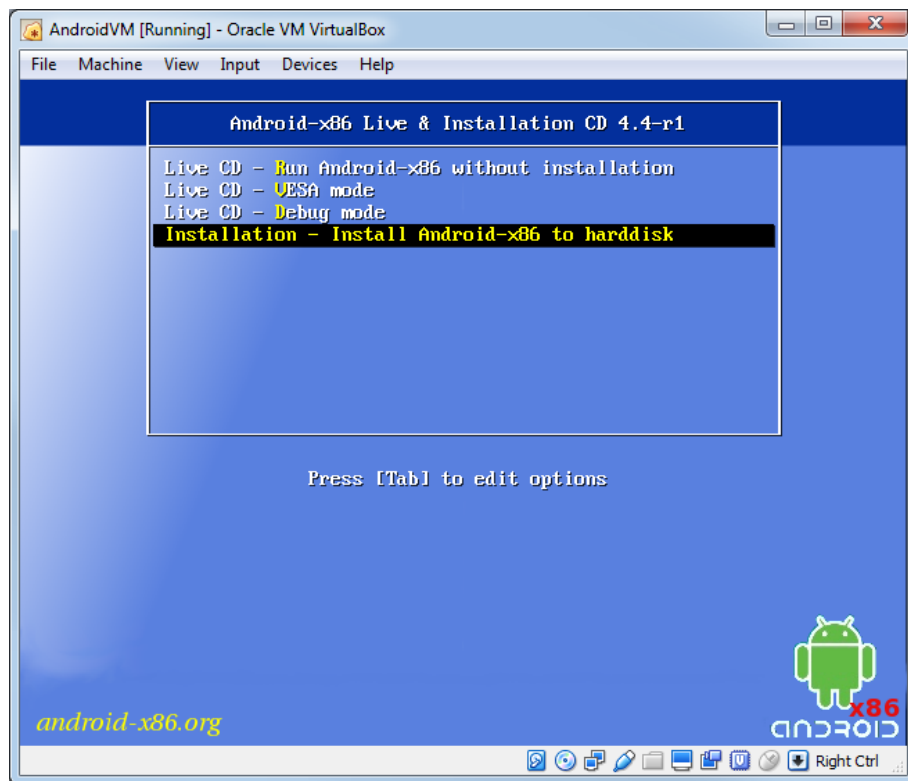
11. Go to the network settings and select either Host-only adapter (if you don't have a local DHCP server) or Bridged mode (if your network has a DHCP server):



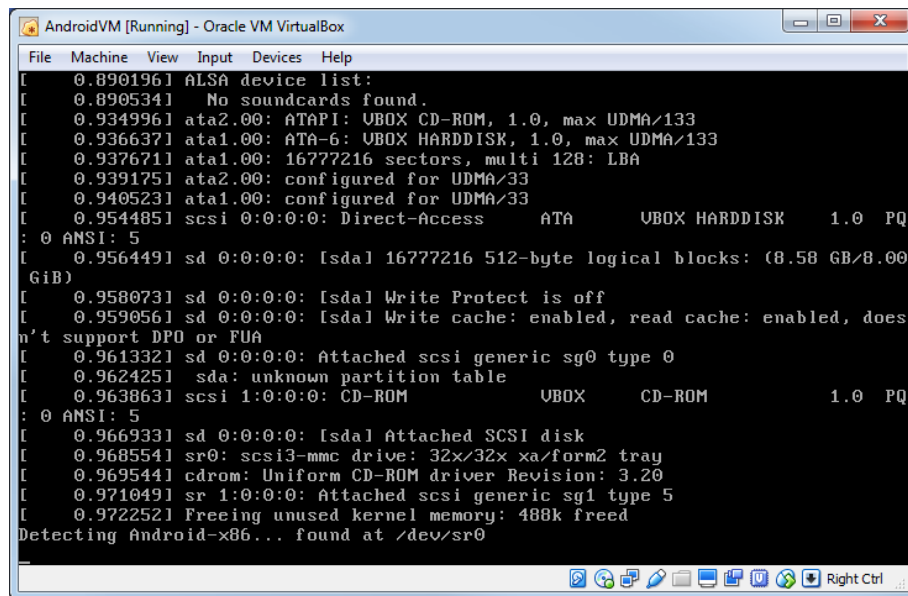
12. Finally press OK to save the settings and start the VM:



13. In the boot menu select “Installation” and press Enter:

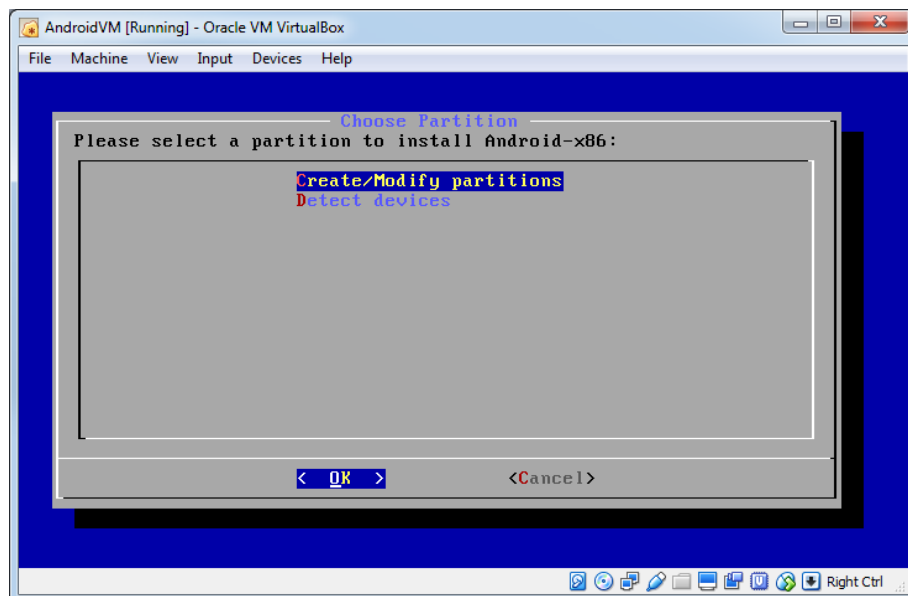


14. Wait until the Android OS boots:

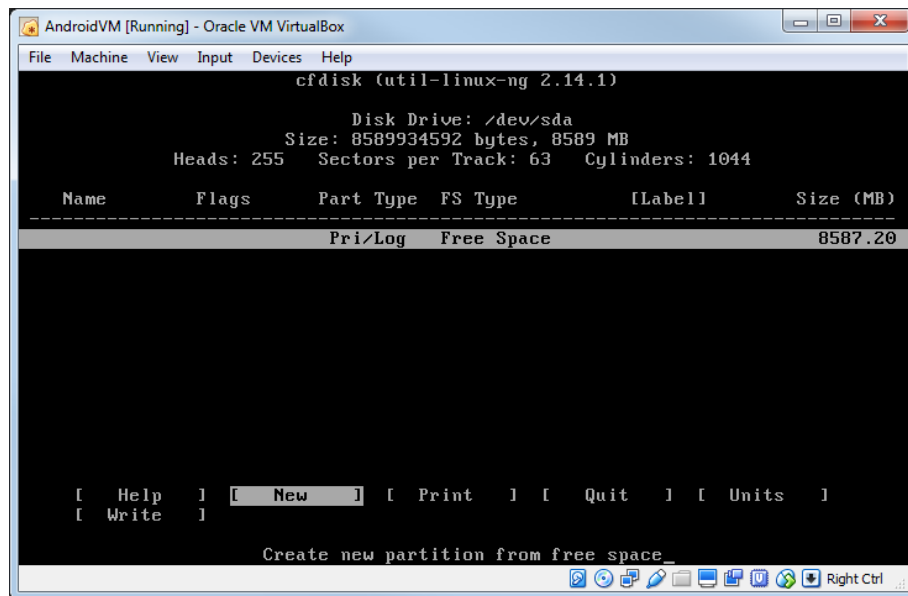


```
AndroidVM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
[ 0.890196] ALSA device list:
[ 0.890534]   No soundcards found.
[ 0.934996] ata2.00: ATAPI: VBOX CD-ROM, 1.0, max UDMA/133
[ 0.936637] ata1.00: ATA-6: VBOX HARDDISK, 1.0, max UDMA/133
[ 0.937671] ata1.00: 16777216 sectors, multi 128: LBA
[ 0.939175] ata2.00: configured for UDMA/33
[ 0.940523] ata1.00: configured for UDMA/33
[ 0.954485] scsi 0:0:0:0: Direct-Access      ATA          VBOX HARDDISK    1.0   PQ
: 0 ANSI: 5
[ 0.956449] sd 0:0:0:0: [sda] 16777216 512-byte logical blocks: (8.58 GB/8.00
GiB)
[ 0.958073] sd 0:0:0:0: [sda] Write Protect is off
[ 0.959056] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, does
n't support DPO or FUA
[ 0.961332] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 0.962425] sda: unknown partition table
[ 0.963863] scsi 1:0:0:0: CD-ROM           VBOX          CD-ROM          1.0   PQ
: 0 ANSI: 5
[ 0.966933] sd 0:0:0:0: [sda] Attached SCSI disk
[ 0.968554] sr0: scsi3-mmc drive: 32x/32x xa/form2 tray
[ 0.969544] cdrom: Uniform CD-ROM driver Revision: 3.20
[ 0.971049] sr 1:0:0:0: Attached scsi generic sg1 type 5
[ 0.972252] Freeing unused kernel memory: 488k freed
Detecting Android-x86... found at /dev/sr0
```

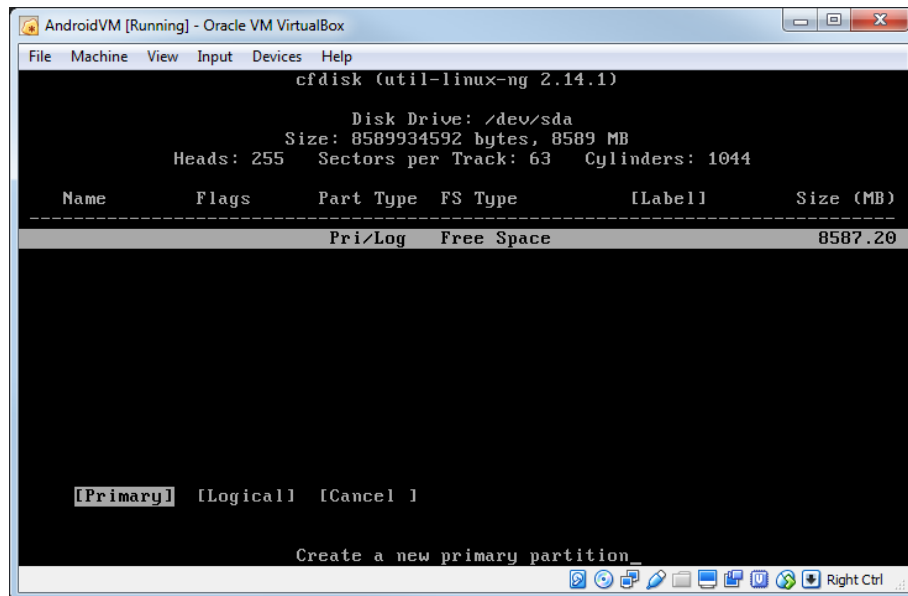
15. Select "Create/modify partitions":



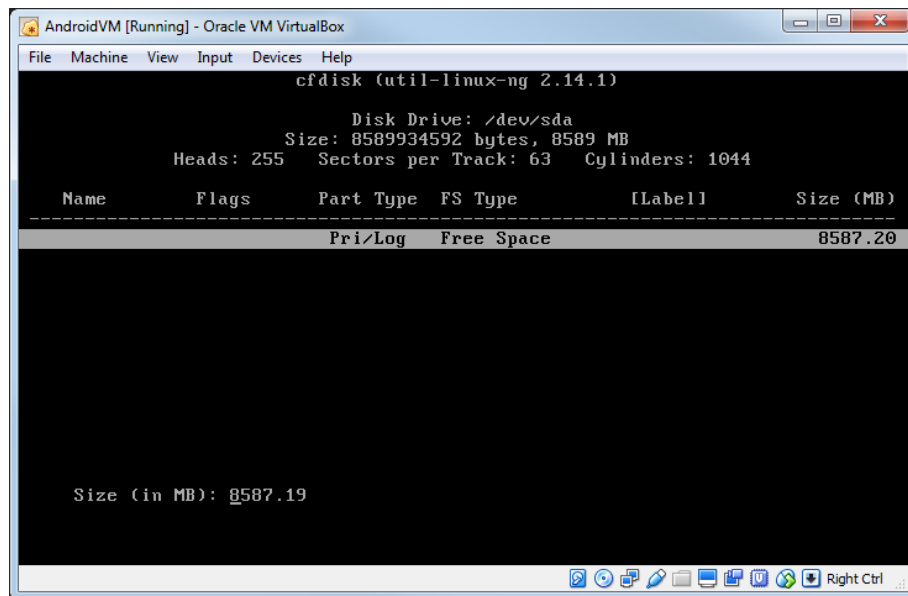
16. Select "New" to create a new partition:



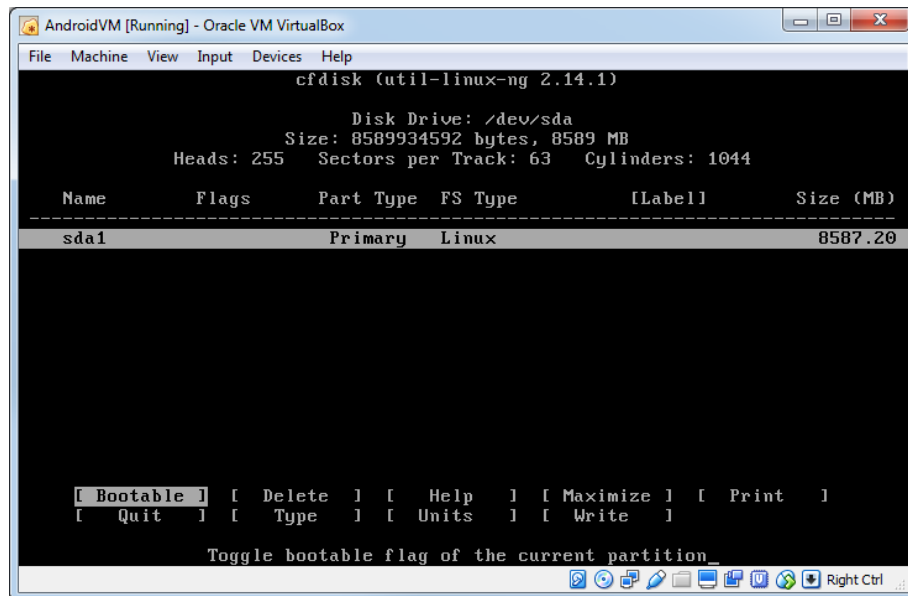
17. Select "Primary" to create a partition that can be used to store a bootable OS:



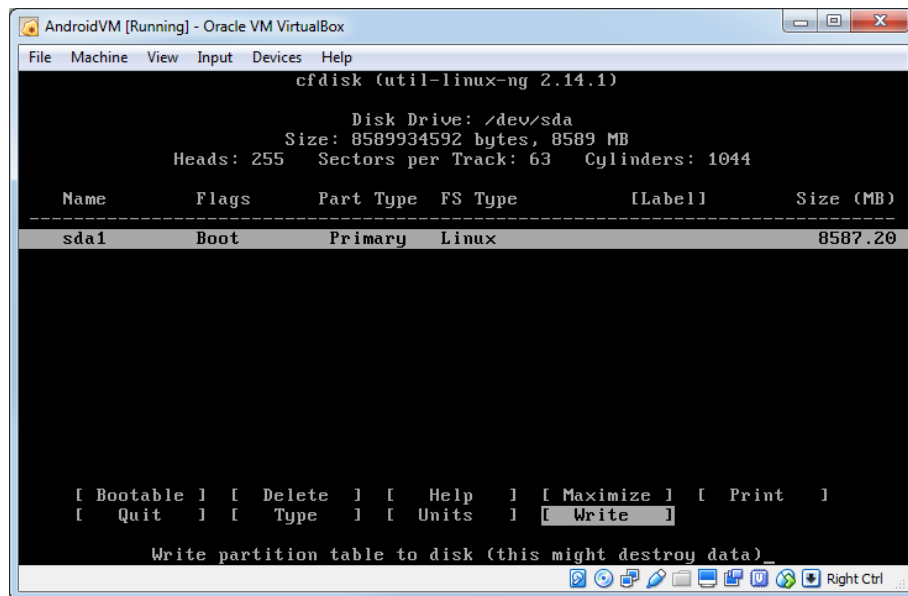
18. Proceed with the size suggested by the partition utility. By default the new partition will cover the entire disk:



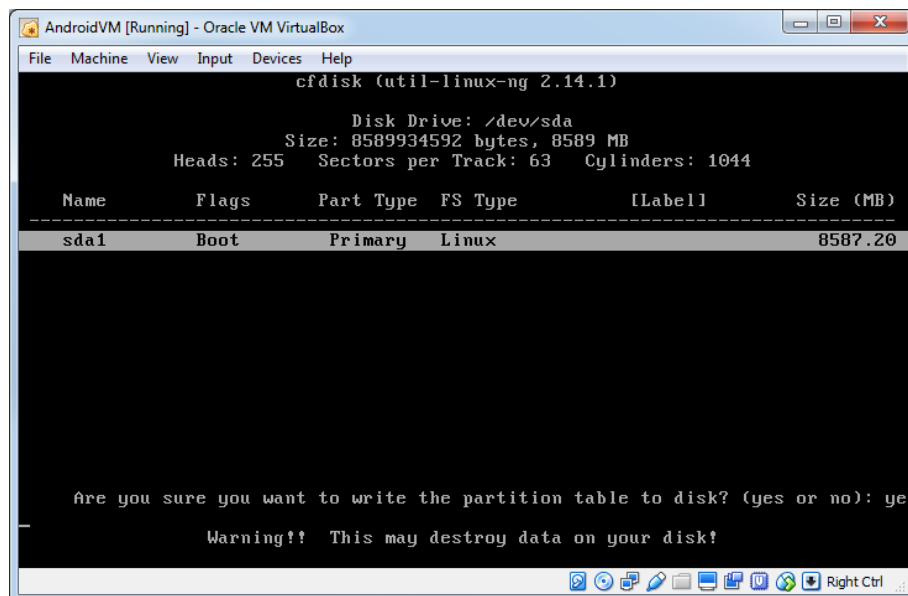
19. Select the “Bootable” button and press Enter to mark the partition as bootable:



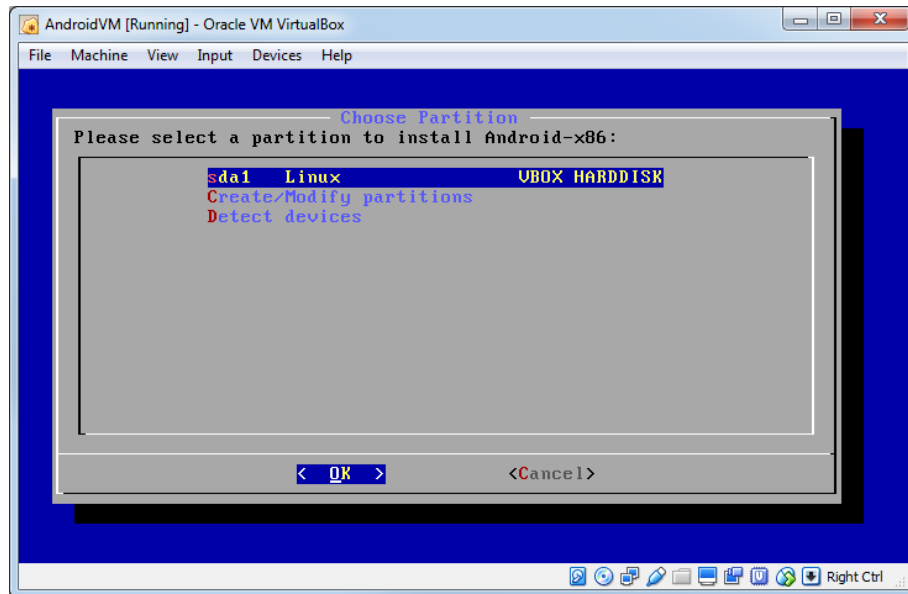
20. Finally select “Write” and press Enter to save the partition table to the disk:



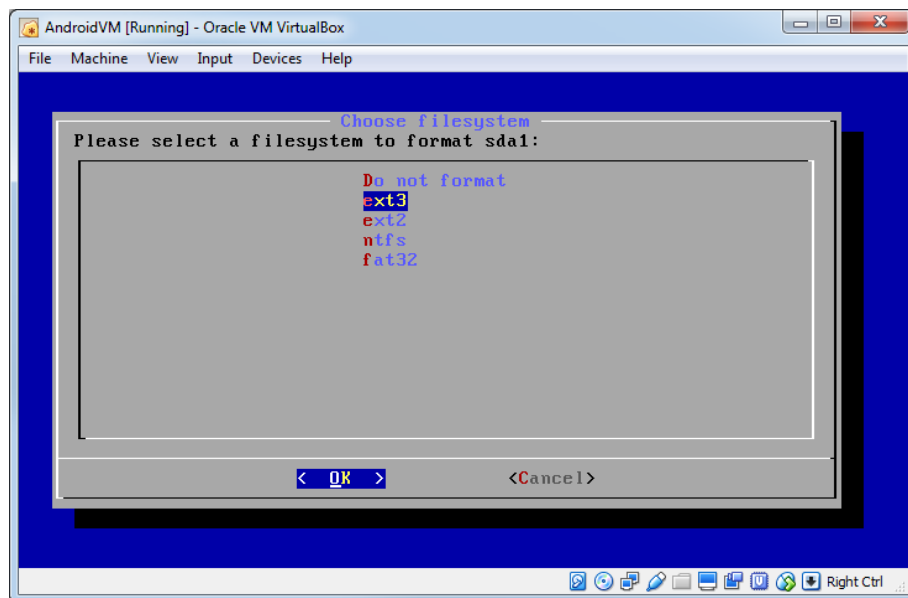
21. Type “yes” and press Enter to confirm the write:



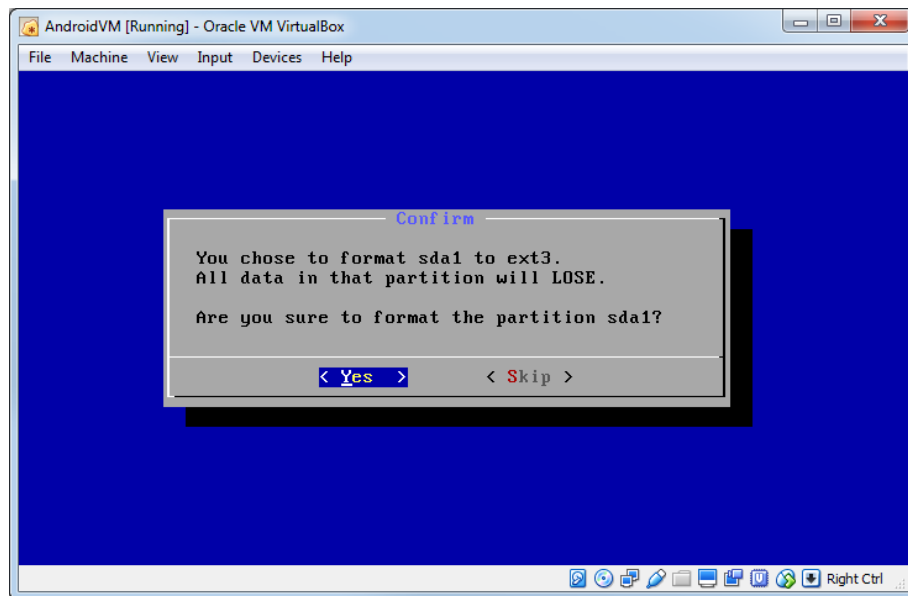
22. Now you can select the newly created partition in the partition list and press OK:



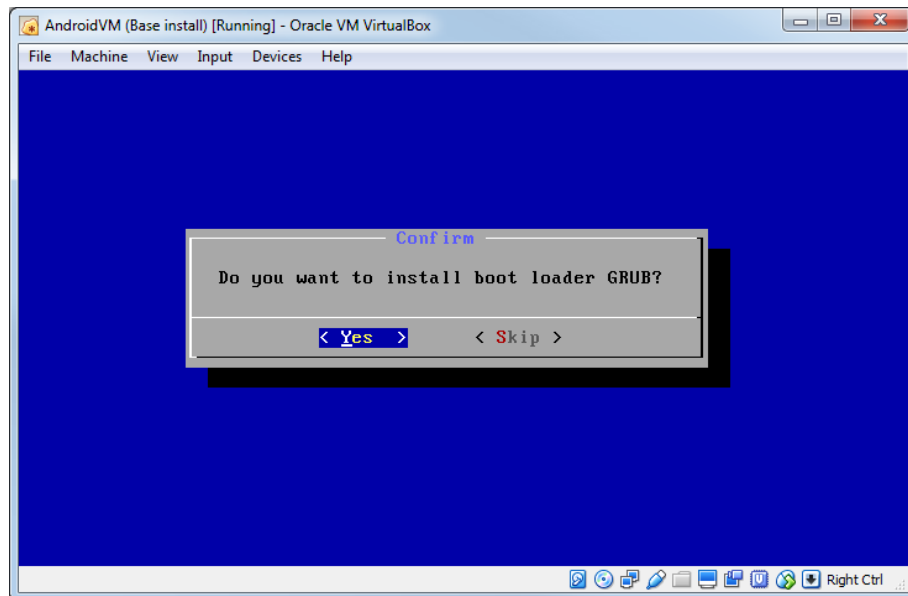
23. Choose to format it with the ext3 filesystem:



24. Confirm the format operation:



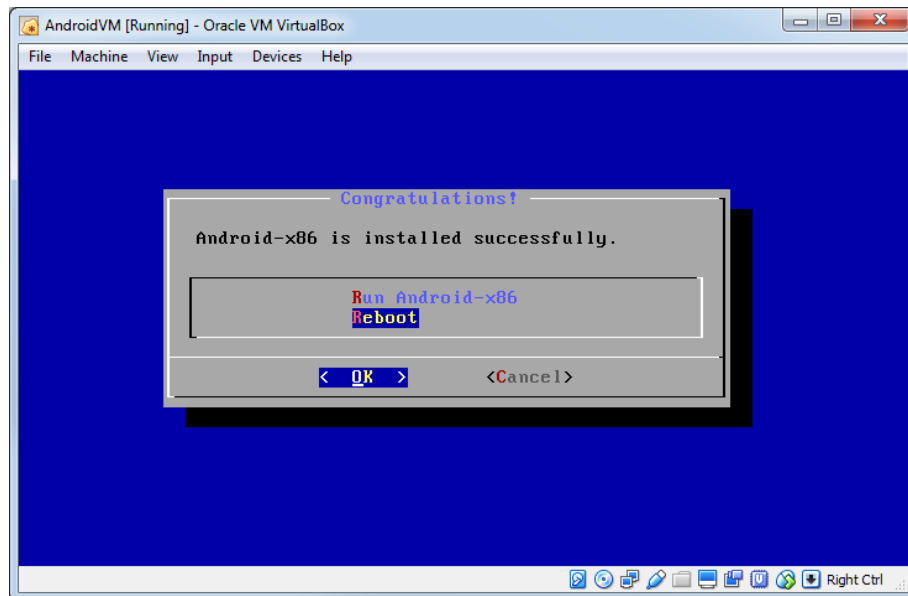
25. Choose “Yes” to install the boot loader so that you can boot into the operating system:



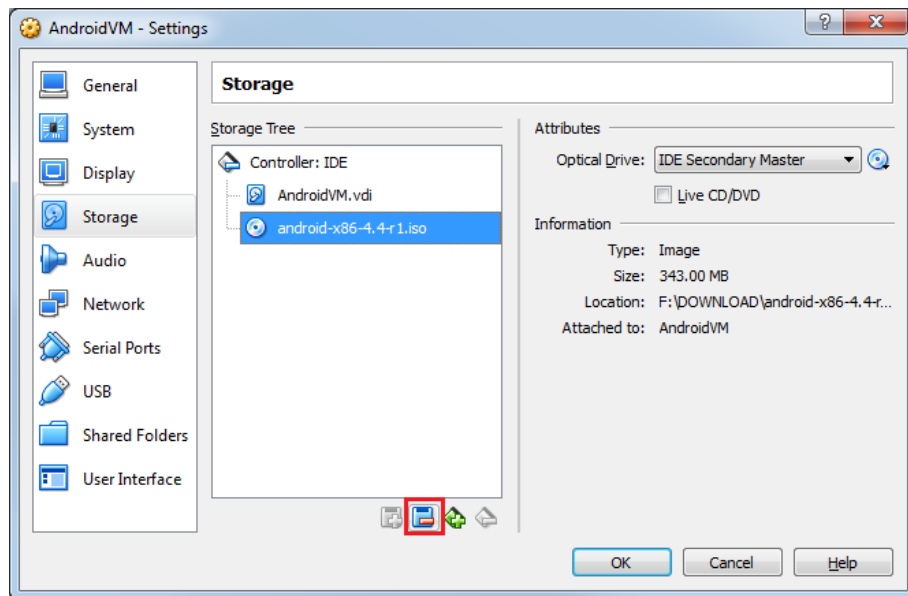
26. Choose to make the system directory writable as this will allow fixing permission bug described later:



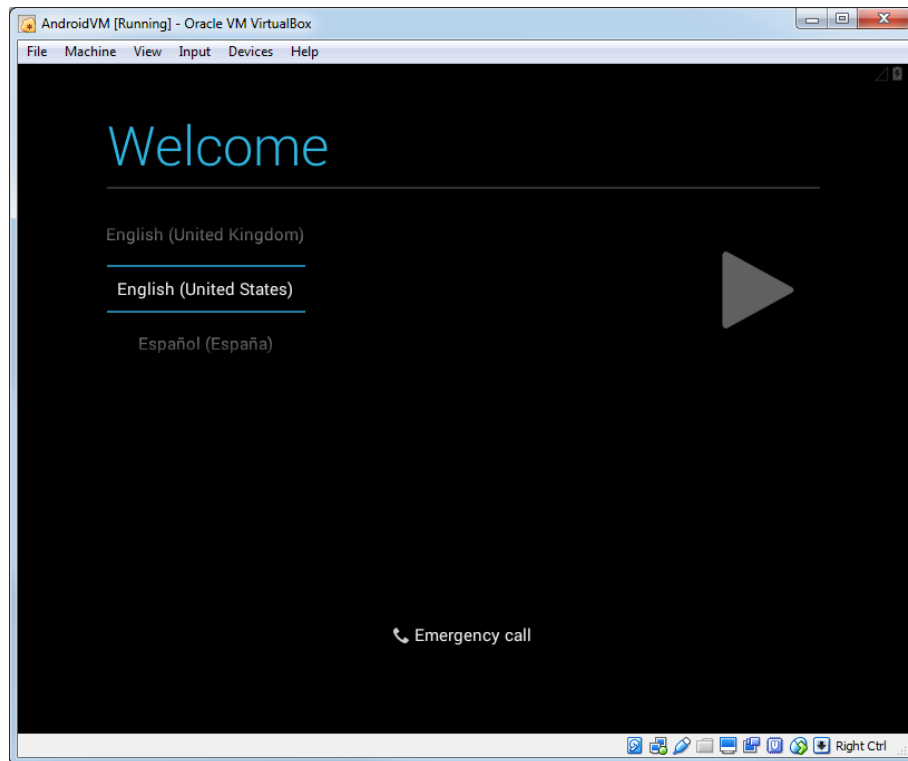
27. The Android OS will now be installed. Once the installation completes, choose "Reboot":



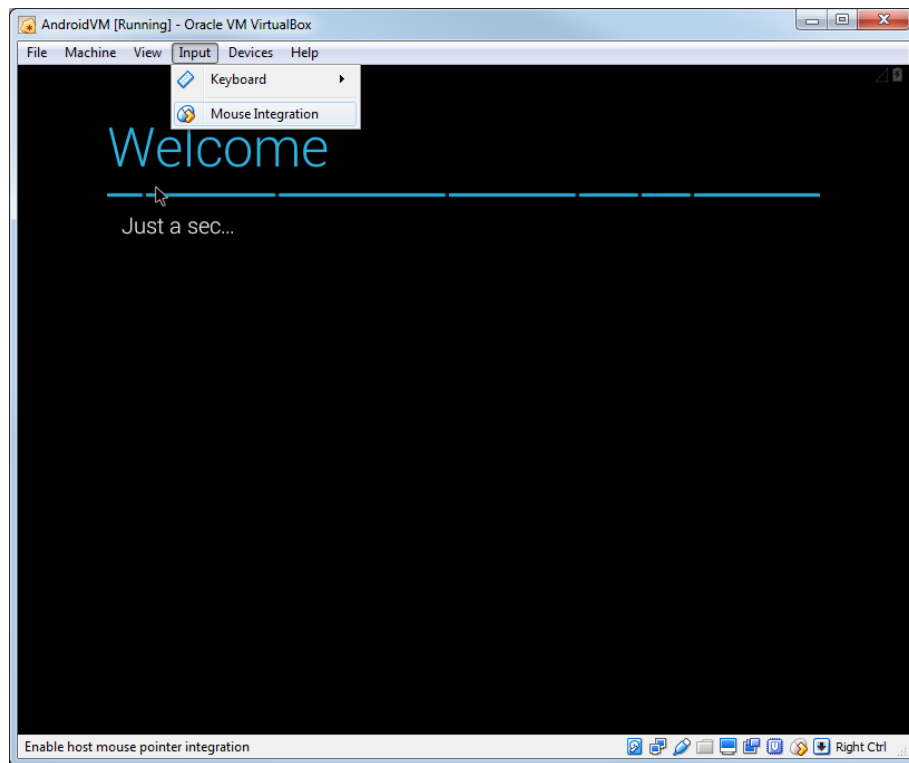
28. To avoid booting into the installer again, open VM properties and remove the virtual CD-ROM device:



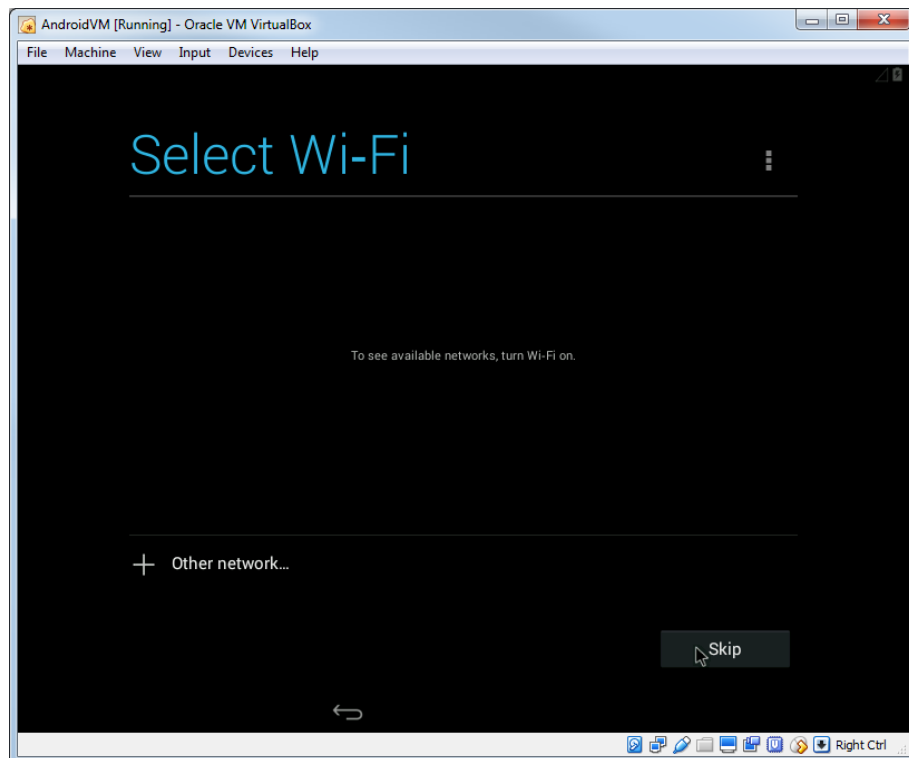
29. Once the OS boots, select your language and press the button to the right to continue:



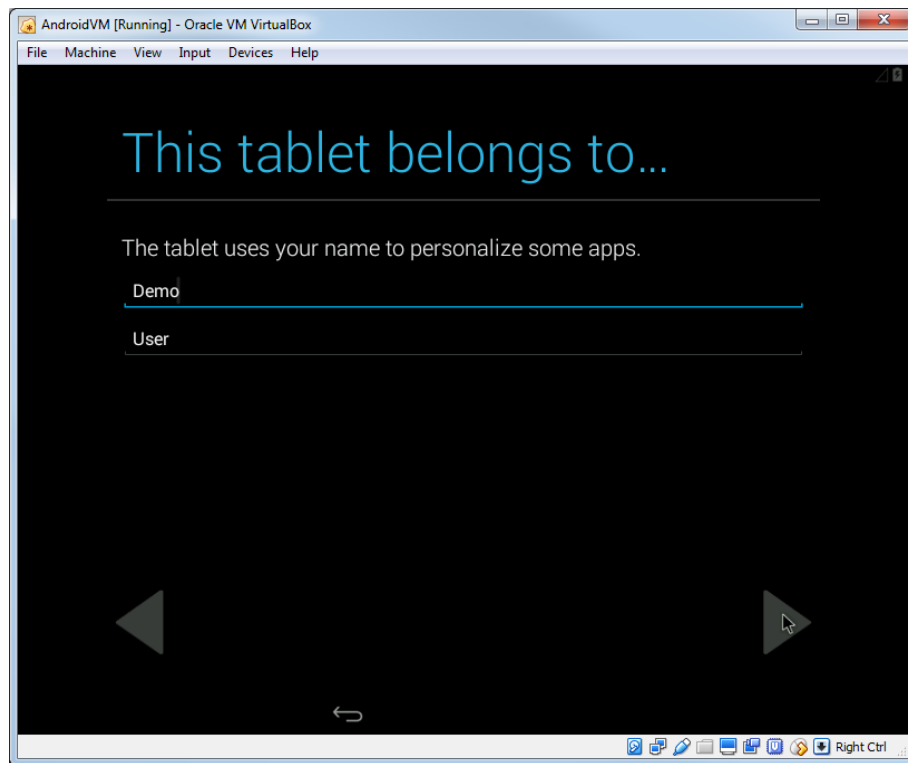
30. If you cannot see the mouse pointer, try toggling the mouse integration in the Input menu:



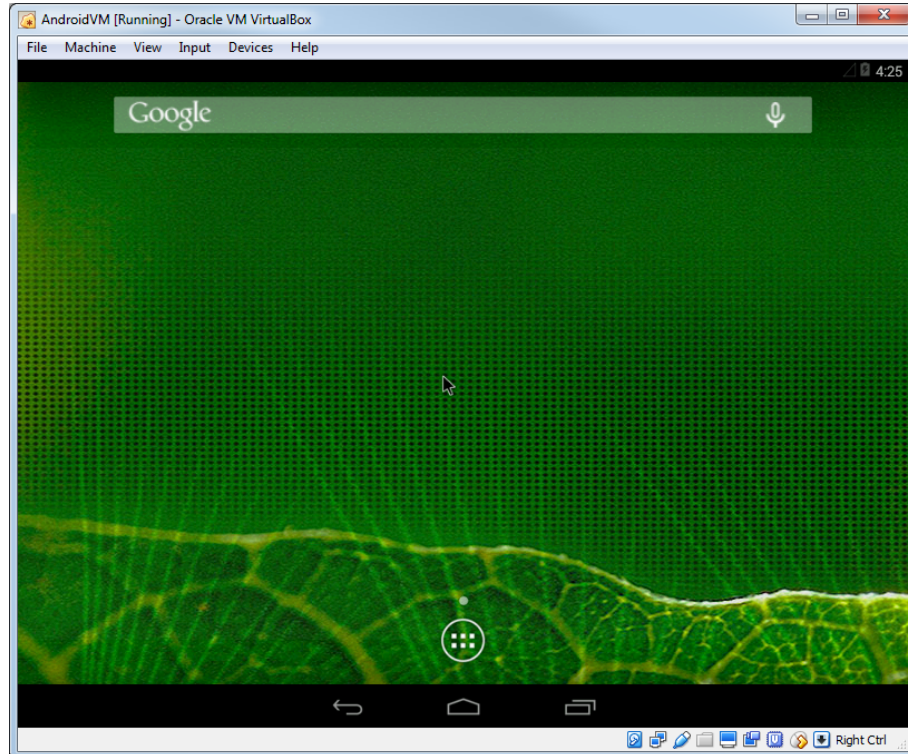
31. Skip the WiFi selection as we will be using a virtual network adapter anyway:



32. Enter the user name that will be associated with your virtual OS installation:



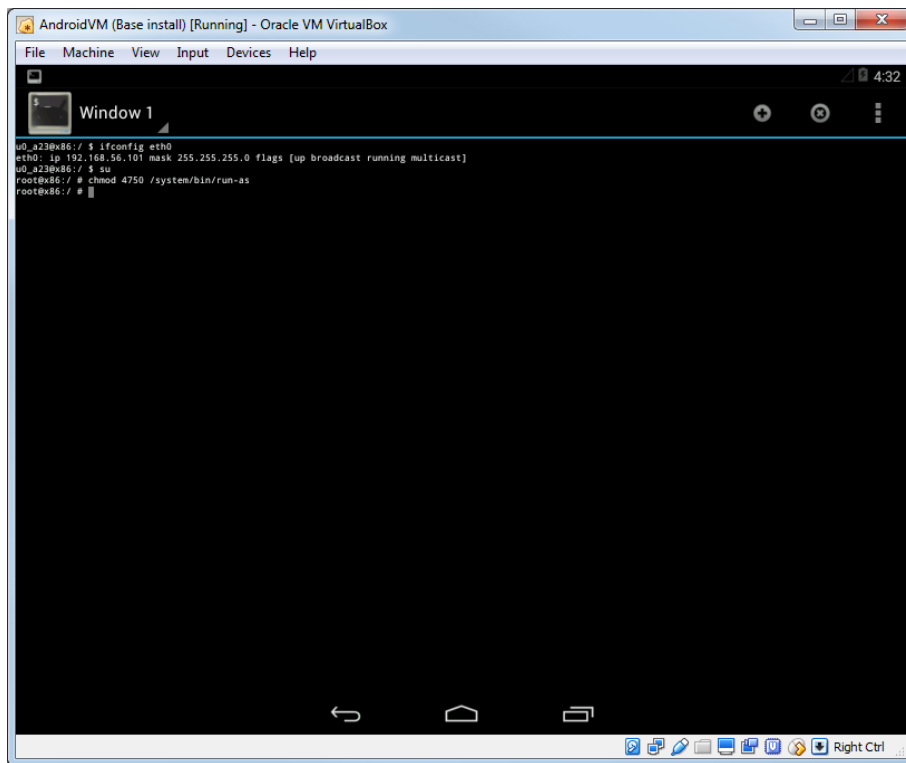
33. Now the installation is complete and you can finally start using your OS. It is recommended to create a VM snapshot at this point so that you can return to it later if something breaks:



34. Open the Terminal app from the apps list and run the following commands to fix the run-as permission bug and determine the current IP address:

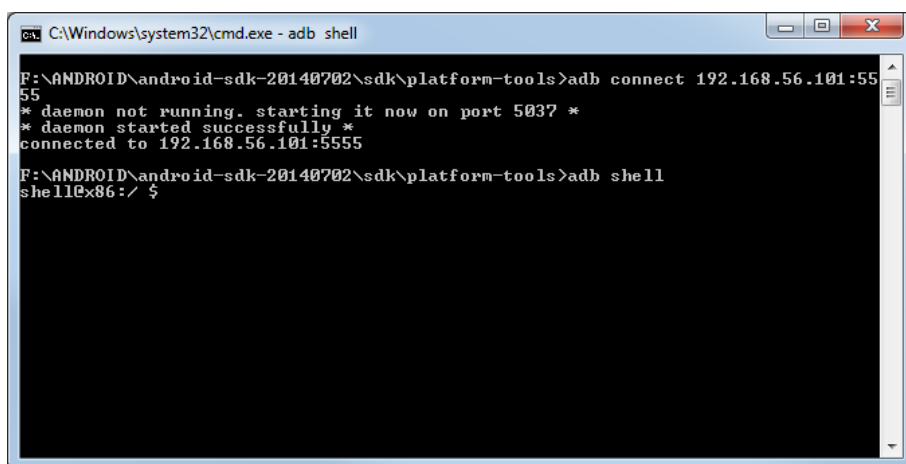
```
1 ifconfig eth0
```

```
2 su
3 chmod 4750 /system/bin/run-as
```



Unless you update the permission on the run-as tool, you won't be able to debug native Android code, as the Android system won't be able to launch the gdbserver under the correct user account.

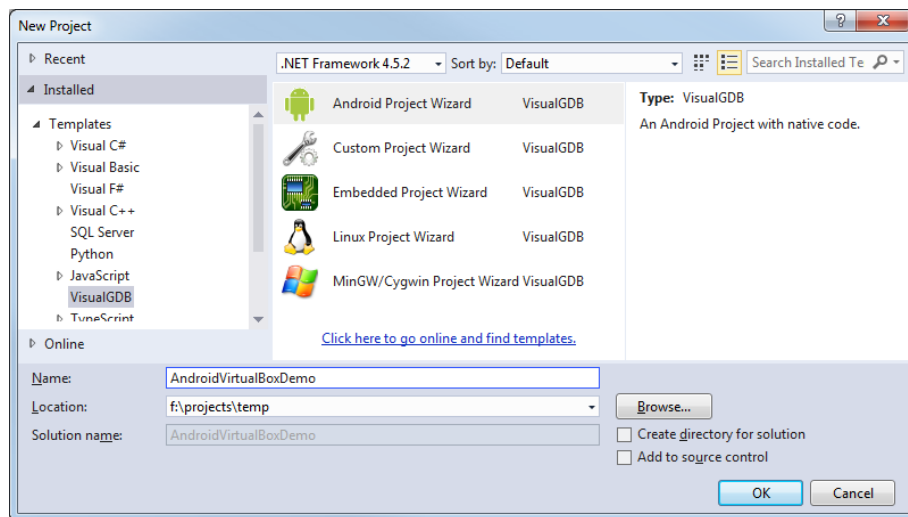
35. Open a command prompt window on your Windows machine and run the "adb connect <IP ADDRESS>:5555" command followed by "adb shell". You should see the Android shell of your virtual device:



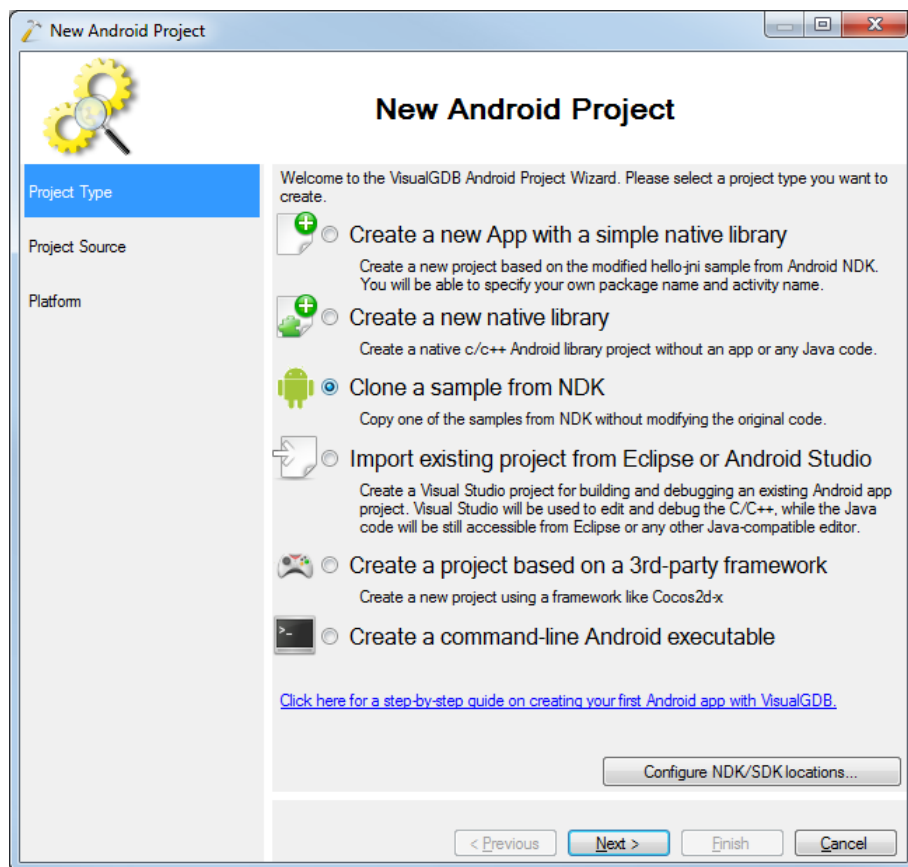
You can close the shell by pressing Ctrl-D and exit the command prompt window.

36. Now we will show how to create and debug a basic OpenGL app with Visual Studio and VisualGDB. Start Visual Studio

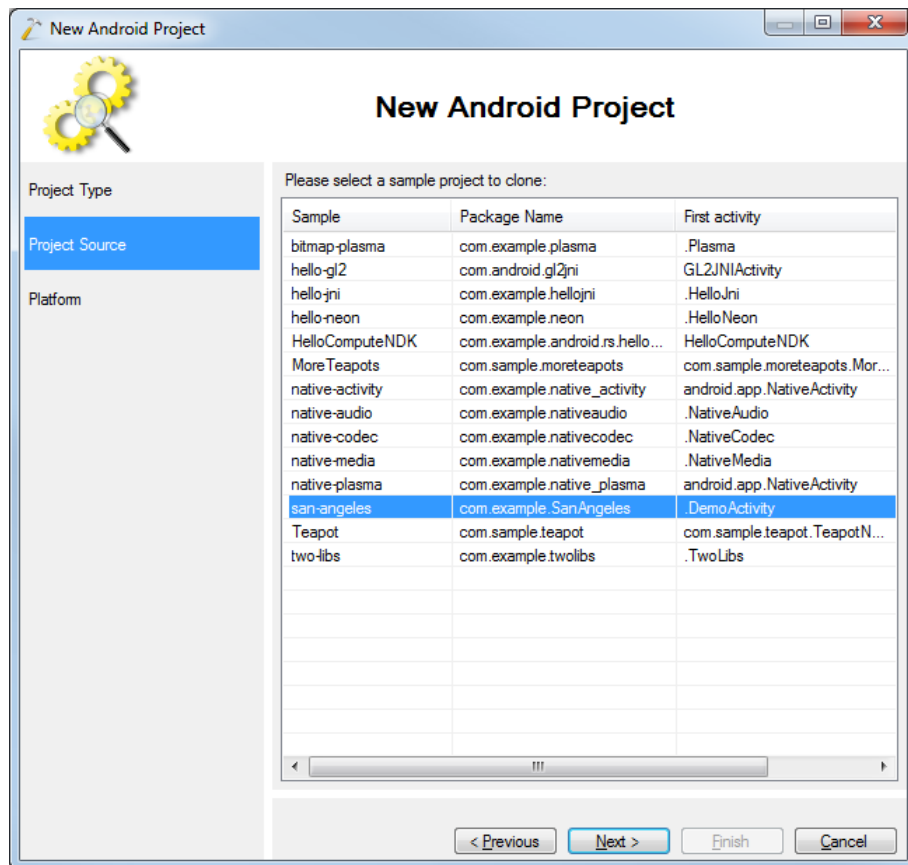
and launch the VisualGDB Android Project Wizard:



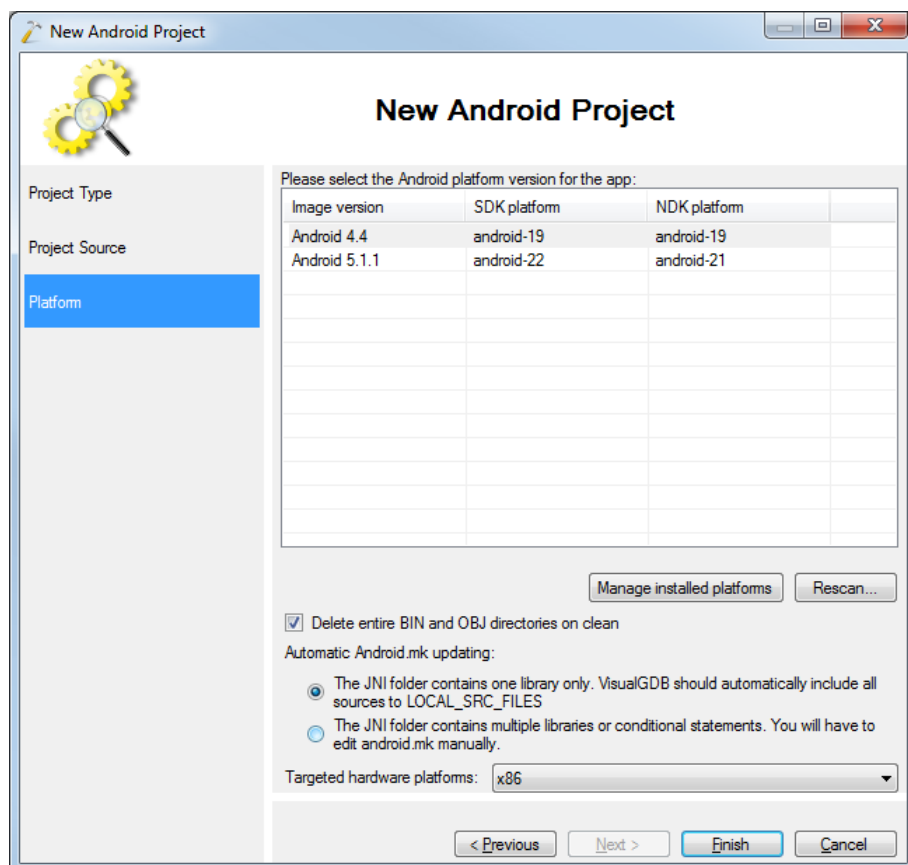
37. Select "Clone a sample from NDK":



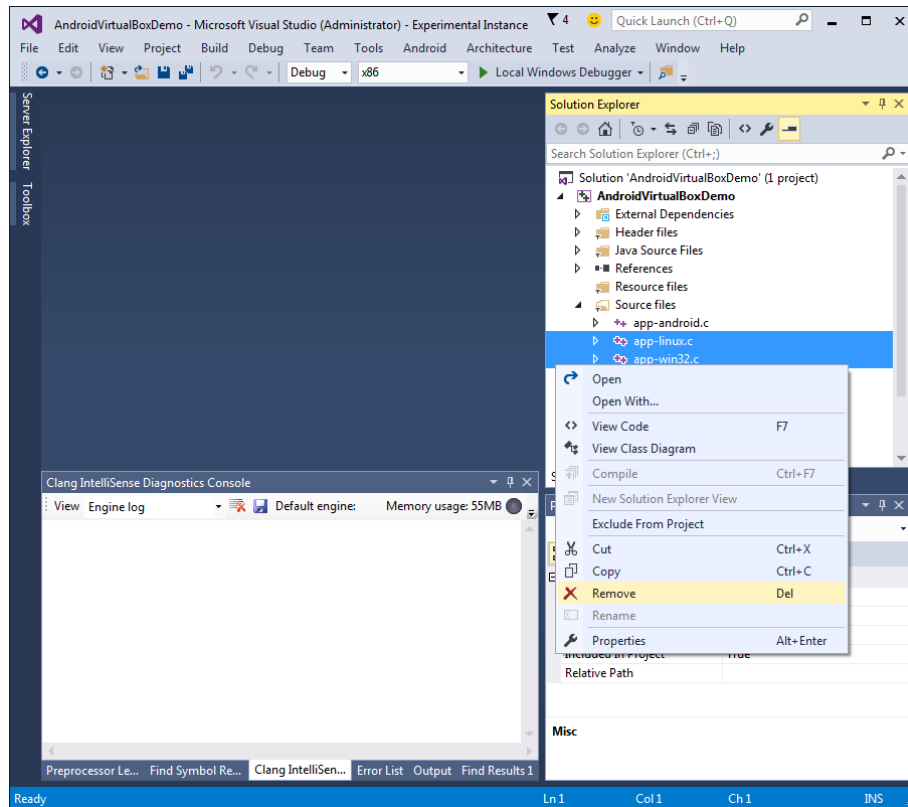
38. Select the san-angeles project from the sample list:



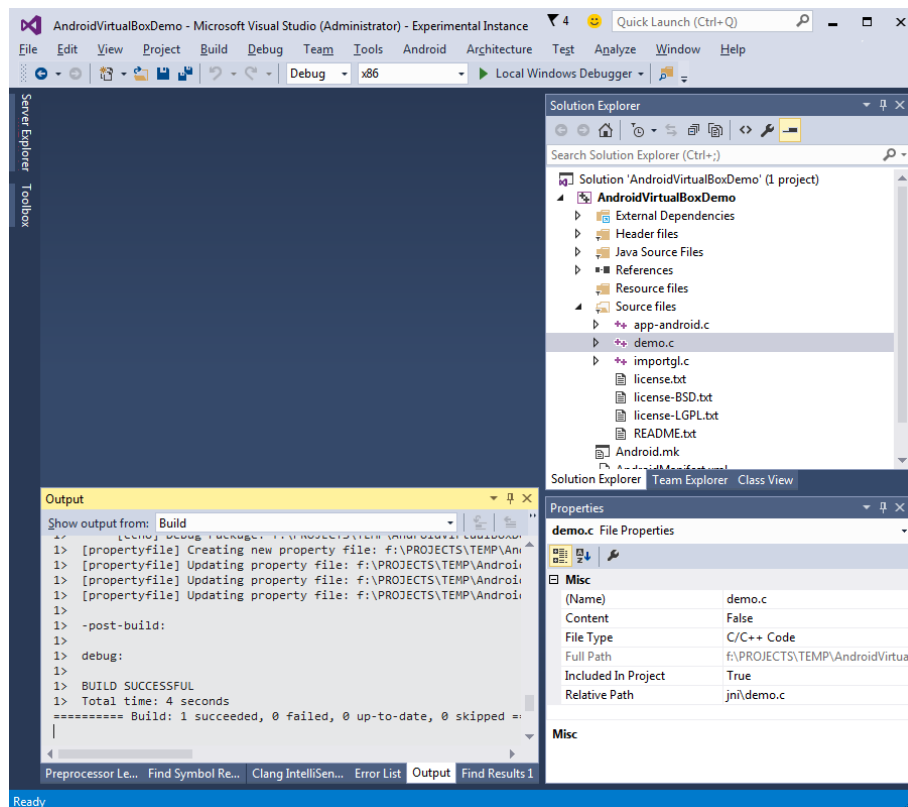
39. Select the Android OS version that matches the version you installed into VirtualBox and select "x86" as the hardware platform:



40. Once the project is created, remove the **app-linux.c** and **app-win32.c** files:

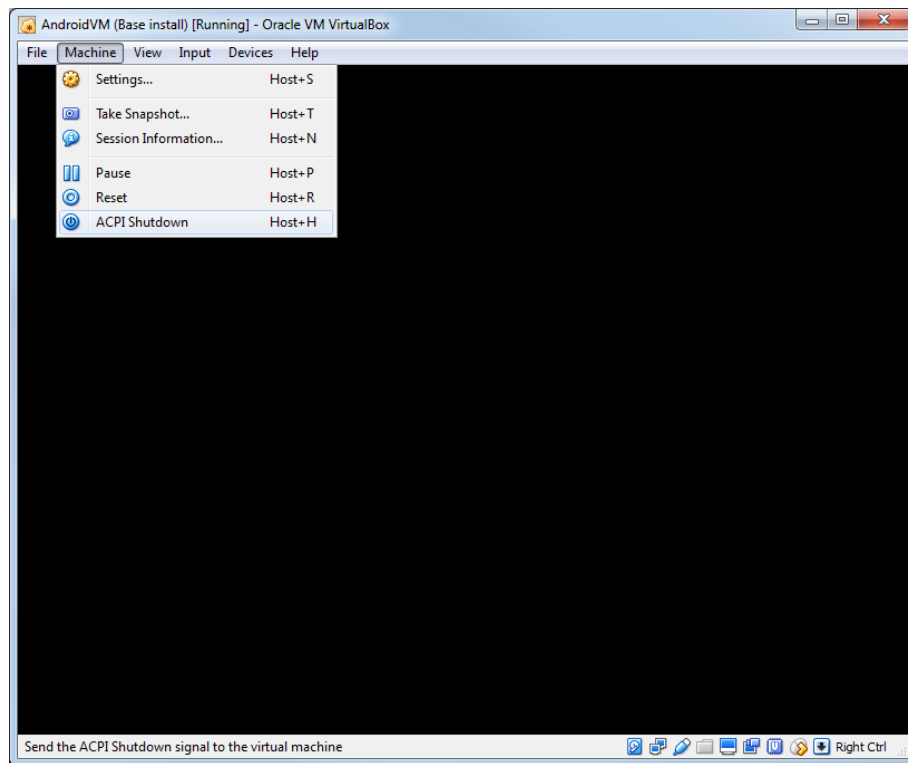


41. Build the project with Ctrl-Shift-B:

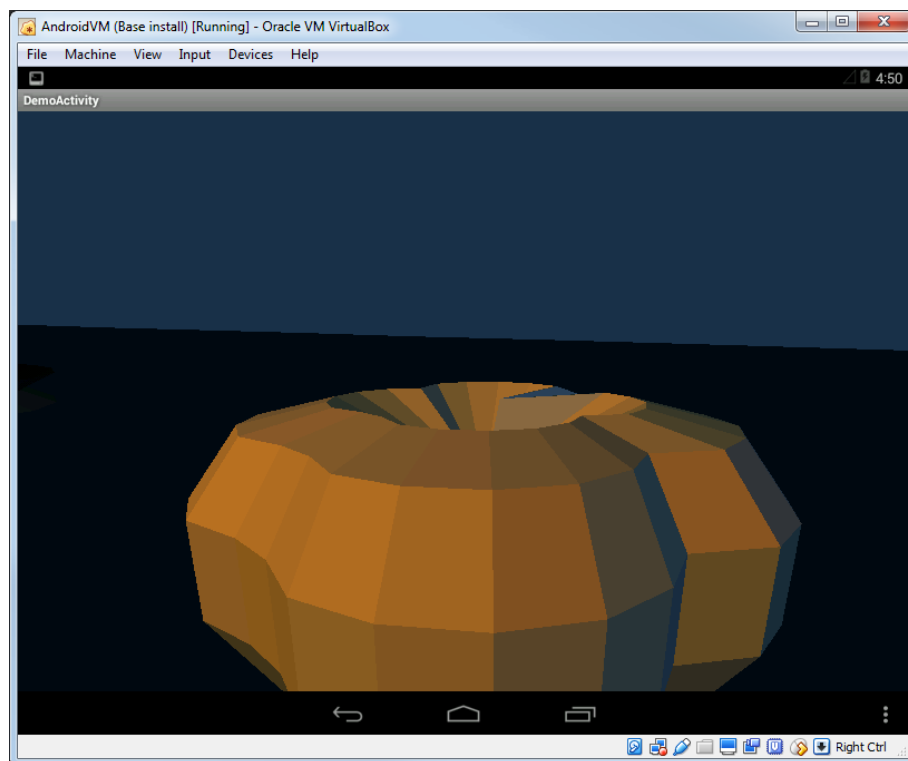


42. If your Android VM screen appears blank, the Android OS has most likely gone into the sleep mode and won't react on

keyboard and mouse events. Use the Machine->ACPI shutdown command to wake it up by emulating the power button press:

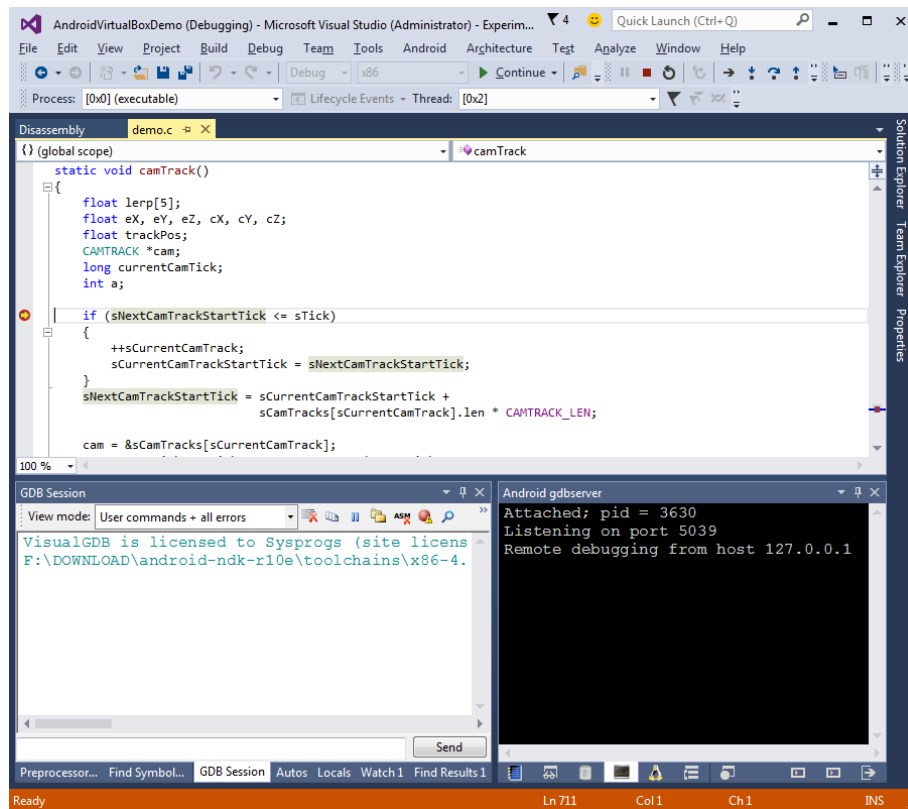


43. Hit F5 in Visual Studio to start debugging. You will see the san-angeles demo being rendered in the VirtualBox window:

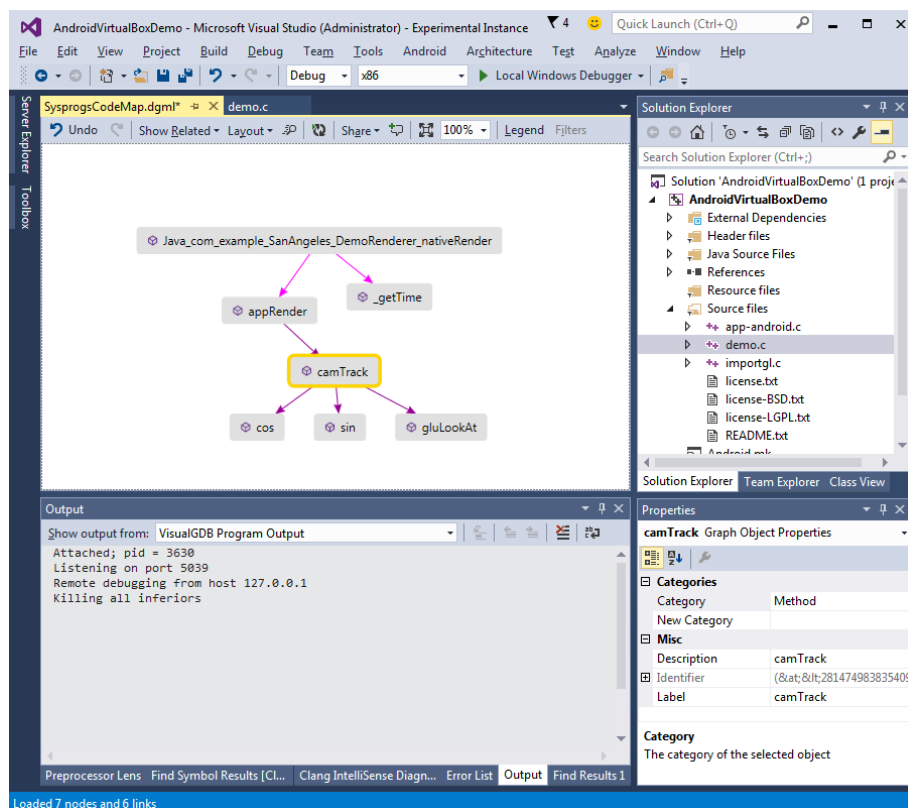


44. Set a breakpoint somewhere in the code (e.g. in the `camTrack()` function). It should hit as soon as the function

gets executed:



45. You can now use the normal Visual Studio debugging techniques to debug your application. If you are using the Clang-based IntelliSense that comes with VisualGDB, you can also use advanced features like Preprocessor Lens and C++ Code Map:



© 2012-2017 Sysprogs OÜ. All rights reserved.

[Terms of Use](#) | [Copyright](#) | [Privacy Policy](#)