Laboratory 1
Introduction to OpenGL ES 1.x

OpenGL ES

OpenGL ES is a 3D graphics standard based on the OpenGL library developed in the labs of Silicon Graphics in 1992. It is widely used across the industry in everything from pocketable machines running games up to supercomputers running fluid dynamics simulations for NASA. The ES variety stands for Embedded Systems, meaning small, portable, low-power devices.

OpenGL for Embedded Systems (OpenGL ES or GLES) is a subset of the OpenGL computer graphics rendering application programming interface (API) for rendering 2D and 3D computer graphics such as those used by video games, typically hardware-accelerated using a graphics processing unit (GPU). It is designed for embedded systems like smartphones, computer tablets, video game consoles and PDAs. OpenGL ES is the most widely deployed 3D graphics API in history.

In this laboratory, we will be using the OpenGL ES 1.0 and 1.1 versions, generically termed OpenGL ES 1.x.

The OpenGL ES 1.0/1.1 API provides a fixed function pipeline and convenience functions which are not available in the OpenGL ES 2.0 or 3.0 APIs. The latter are oriented toward lower-level operations, sacrificing some of the ease-of-use utility routines for flexibility and control. The OpenGL ES 2.0 and 3.0 APIs provide a higher degree of control by providing a fully programmable pipeline through the use of shaders. With more direct control of the graphics processing pipeline, developers can create effects that would be very difficult to generate using the 1.0/1.1 API.

Android Studio

The OpenGL ES 1.0 and 1.1 API specification is supported by *Android 1.0* and higher.

Android Studio is the official Android IDE that provides multiple features including

- Android Studio IDE
- Android SDK tools
- Android 7.0 (Nougat) Platform
- Android 7.0 emulator system image with Google APIs.

Android Studio can be installed by the following steps: (**DO NOT** do this in the laboratory, Android Studio is already installed)

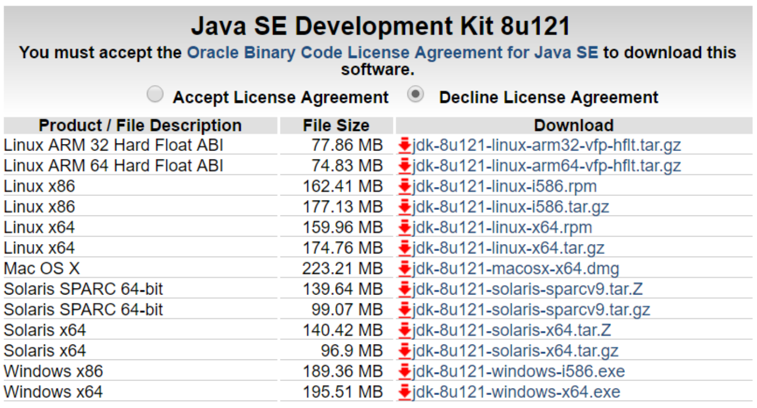
1. (Optional) Upon starting your computer, enter the **BIOS** Utility (*F10*), and, under the **Security** > **System Security** menu, *Enable* the *Intel Virtualization Technology.* (This applies for the HP machines in the laboratory. The names and locations of the

menus might differ, depending on your machine.) This is the so-called *Intel Virtualization Technology (VT-x)* which speeds up the Android emulator. Note that this step is optional for the necessities of the present laboratory, since we will be working with an Android 2.2 emulator, which runs rather fast even if the Virtualization Technology is not used. But for running the latest Android 7 emulator, this step is mandatory.

2. From the web page

   http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html,

   download and install the latest **Java SE Development Kit 8u121**. Choose the appropriate download for your machine. Make sure to select the *Accept License Agreement* radio button.



Java SE Development Kit 8u121

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

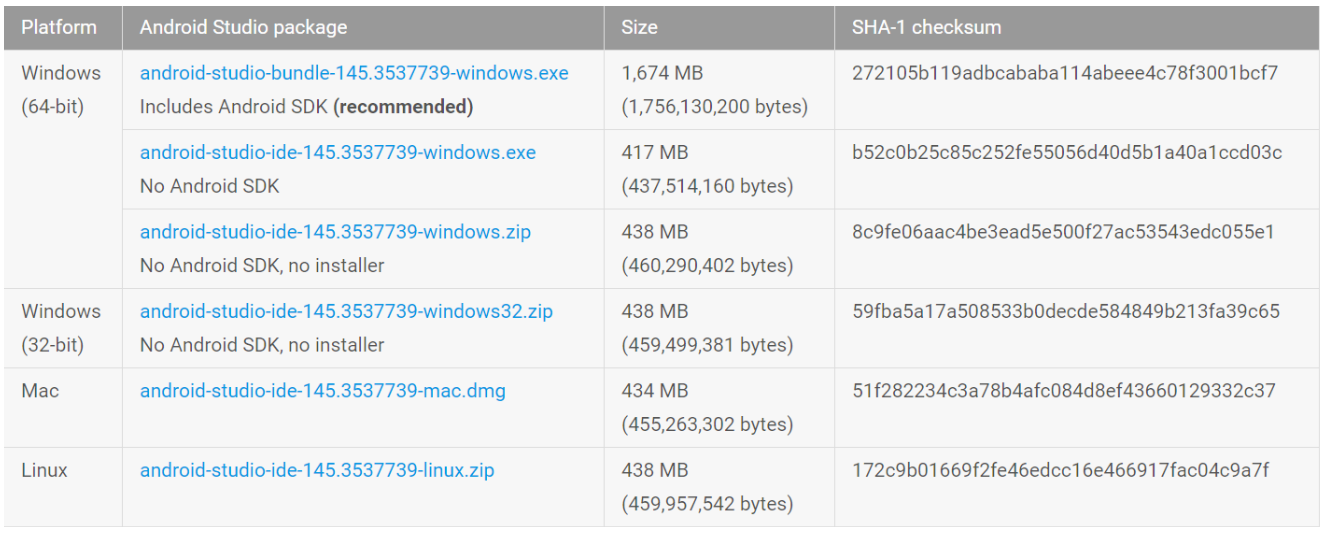
Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.86 MB	jdk-8u121-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u121-linux-arm64-vfp-hflt.tar.gz
Linux x86	162.41 MB	jdk-8u121-linux-i586.rpm
Linux x86	177.13 MB	jdk-8u121-linux-i586.tar.gz
Linux x64	159.96 MB	jdk-8u121-linux-x64.rpm
Linux x64	174.76 MB	jdk-8u121-linux-x64.tar.gz
Mac OS X	223.21 MB	jdk-8u121-macosx-x64.dmg
Solaris SPARC 64-bit	139.64 MB	jdk-8u121-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.07 MB	jdk-8u121-solaris-sparcv9.tar.gz
Solaris x64	140.42 MB	jdk-8u121-solaris-x64.tar.Z
Solaris x64	96.9 MB	jdk-8u121-solaris-x64.tar.gz
Windows x86	189.36 MB	jdk-8u121-windows-i586.exe
Windows x64	195.51 MB	jdk-8u121-windows-x64.exe

3. From the webpage

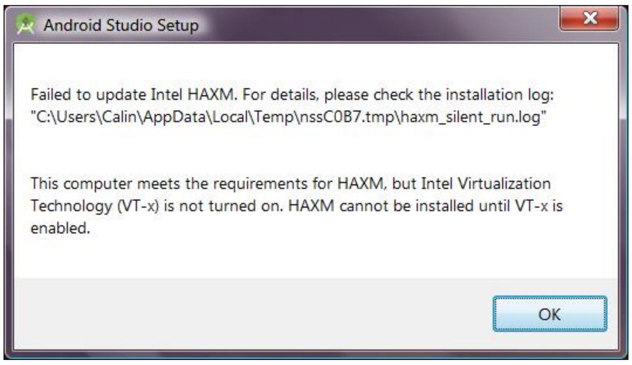
   http://developer.android.com/studio/index.html,

   download and install the latest **Android Studio and SDK Tools**. Choose the appropriate download for your machine, from the *All Android Studio Packages* section. Make sure to check the *I have read and agree with the above terms and conditions* check box.



Platform	Android Studio package	Size	SHA-1 checksum
Windows (64-bit)	android-studio-bundle-145.3537739-windows.exe Includes Android SDK **(recommended)**	1,674 MB (1,756,130,200 bytes)	272105b119adbcababa114abeee4c78f3001bcf7
	android-studio-ide-145.3537739-windows.exe No Android SDK	417 MB (437,514,160 bytes)	b52c0b25c85c252fe55056d40d5b1a40a1ccd03c
	android-studio-ide-145.3537739-windows.zip No Android SDK, no installer	438 MB (460,290,402 bytes)	8c9fe06aac4be3ead5e500f27ac53543edc055e1
Windows (32-bit)	android-studio-ide-145.3537739-windows32.zip No Android SDK, no installer	438 MB (459,499,381 bytes)	59fba5a17a508533b0decde584849b213fa39c65
Mac	android-studio-ide-145.3537739-mac.dmg	434 MB (455,263,302 bytes)	51f282234c3a78b4afc084d8ef43660129332c37
Linux	android-studio-ide-145.3537739-linux.zip	438 MB (459,957,542 bytes)	172c9b01669f2fe46edcc16e466917fac04c9a7f

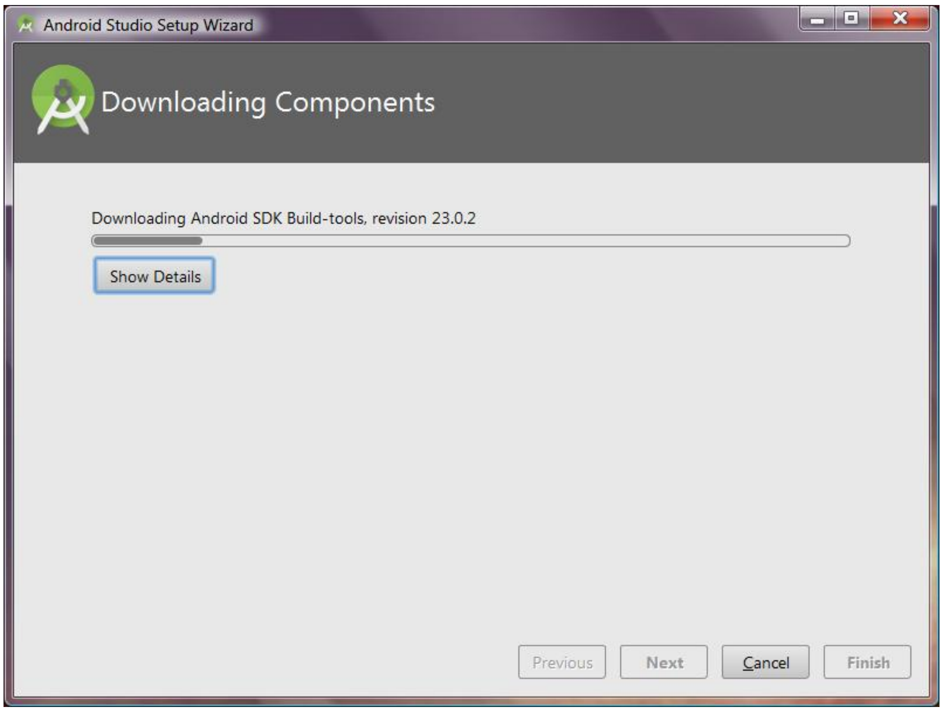
If you successfully completed Step 1, *Intel Hardware Accelerated Execution Manager (Intel HAXM)* will be automatically installed, else a warning message will be given stating either that the processor does not support *Intel Virtualization Technology*, or, more likely, that the *Intel Virtualization Technology* is not enabled. *Intel Hardware Accelerated Execution Manager (Intel HAXM)* is a hardware-assisted virtualization engine (hypervisor) that uses *Intel Virtualization Technology (Intel VT)* to speed up Android app emulation on a host machine.


Android Studio Setup

Failed to update Intel HAXM. For details, please check the installation log: "C:\Users\Calin\AppData\Local\Temp\nssC0B7.tmp\haxm_silent_run.log"

This computer meets the requirements for HAXM, but Intel Virtualization Technology (VT-x) is not turned on. HAXM cannot be installed until VT-x is enabled.

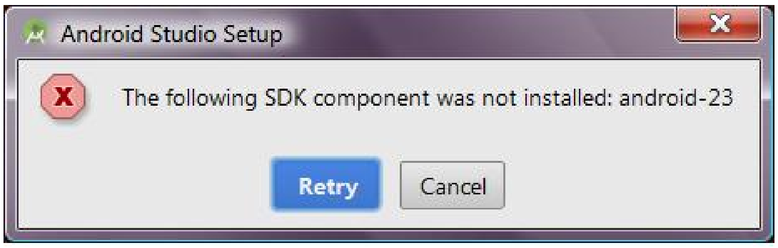
4. After successfully installing **Android Studio**, you are ready to run it for the first time. Setup needs to install some more components:


Android Studio Setup Wizard

Downloading Components

Downloading Android SDK Build-tools, revision 23.0.2

If the error


Android Studio Setup

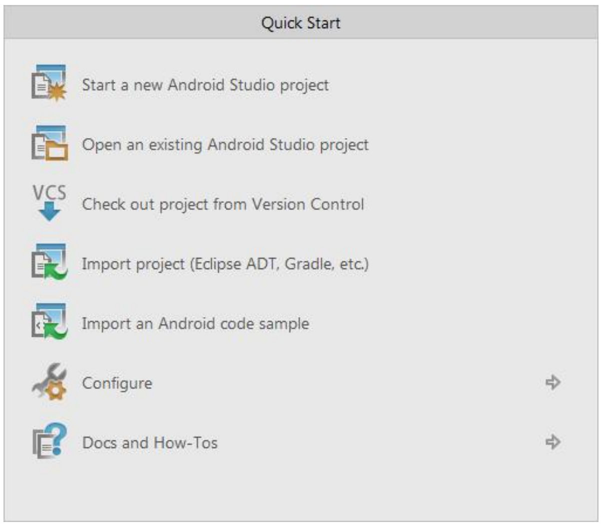
The following SDK component was not installed: android-23

occurs, just click **Retry**, and the component will install successfully.

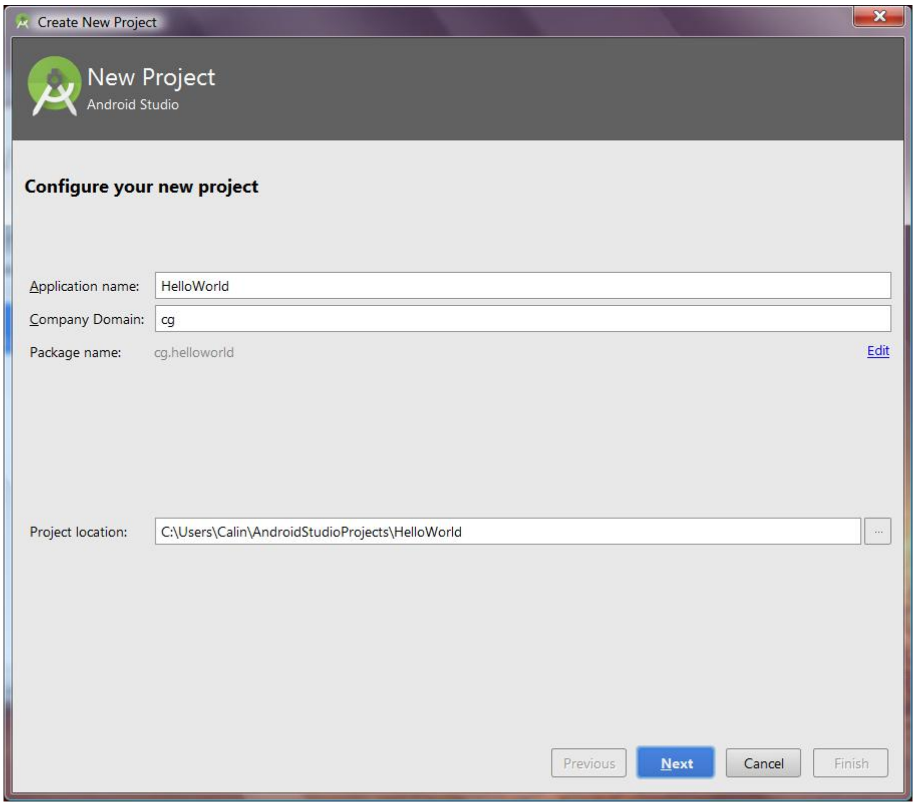
HelloWorld Project

Next, we will develop our first Android Project using OpenGL ES 1.x, generically called *HelloWorld*.

1. In the **Quick Start** dialog, choose **Start a new Android Studio project**:


Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Docs and How-Tos

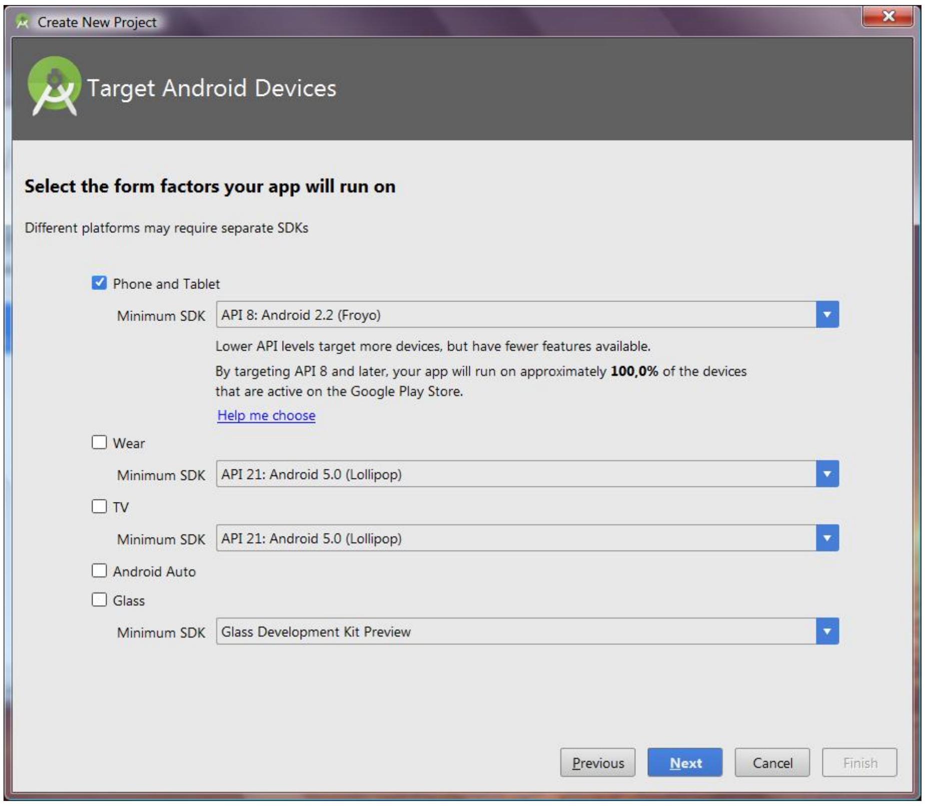
2. The **Create New Project** dialog appears. Write *HelloWorld* in the **Application name:** box and *cg* in the **Company Domain:** box. Click **Next**.



Configure your new project

Application name: HelloWorld
Company Domain: cg
Package name: cg.helloworld
Project location: C:\Users\Calin\AndroidStudioProjects\HelloWorld

3. Then, leaving the **Phone and Tablet** checked, select the **Minimum SDK** as being *API 8: Android 2.2 (Froyo)* from the drop-down list. Click **Next**.



Create New Project

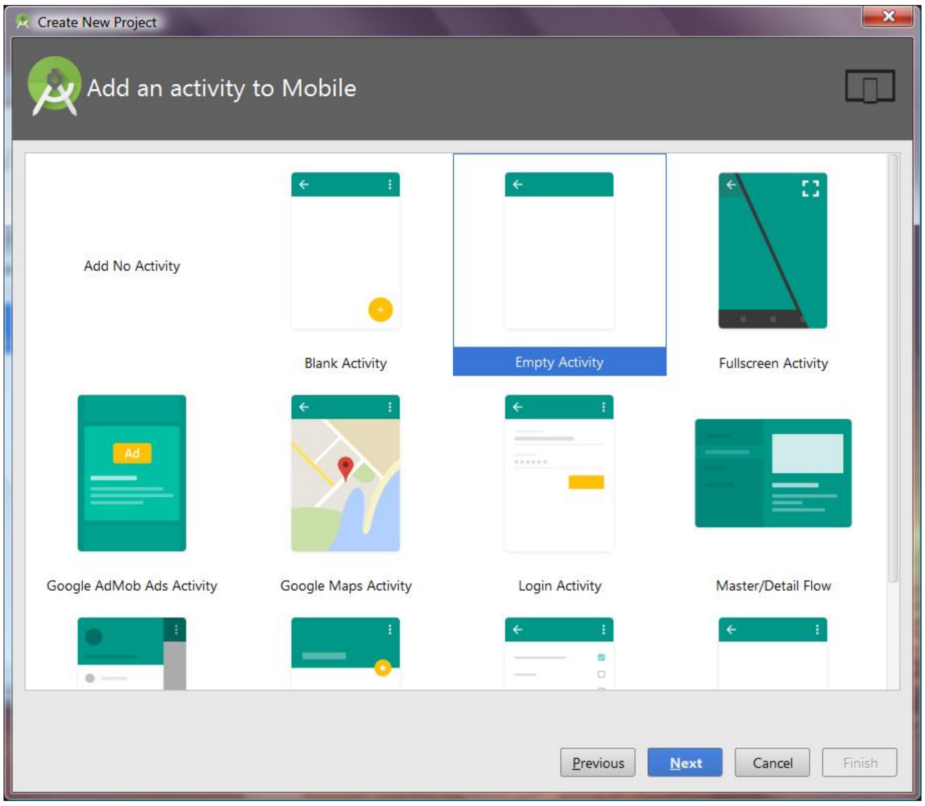
Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

- [x] Phone and Tablet
  - Minimum SDK: API 8: Android 2.2 (Froyo)
  - Lower API levels target more devices, but have fewer features available.
  - By targeting API 8 and later, your app will run on approximately **100,0%** of the devices that are active on the Google Play Store.
  - Help me choose
- [ ] Wear
  - Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] TV
  - Minimum SDK: API 21: Android 5.0 (Lollipop)
- [ ] Android Auto
- [ ] Glass
  - Minimum SDK: Glass Development Kit Preview

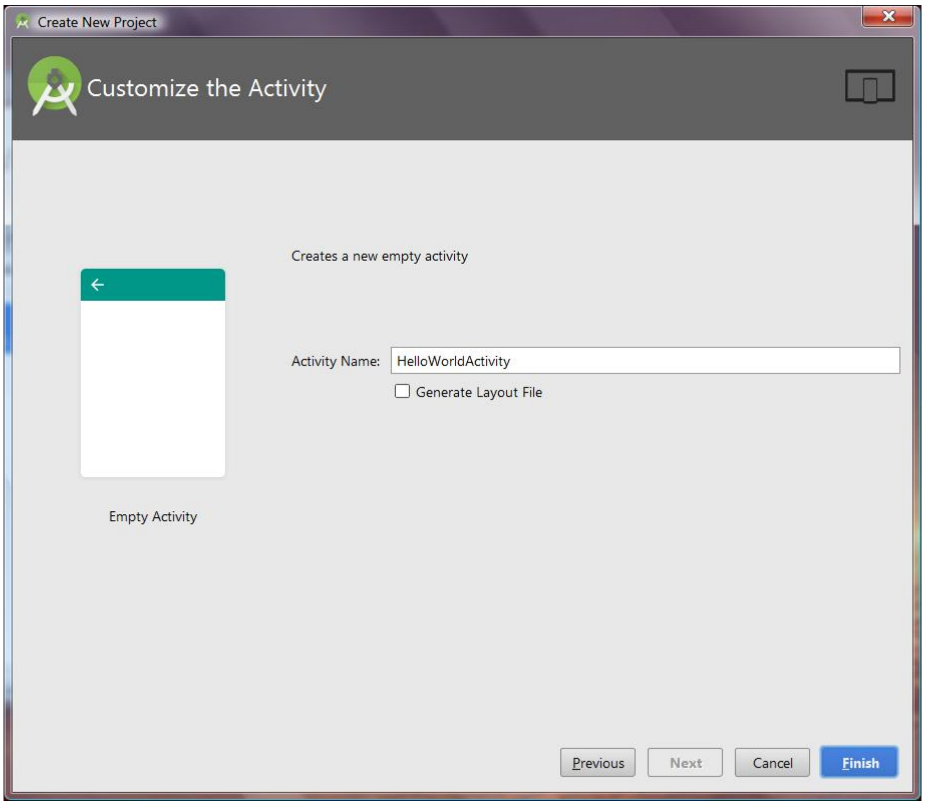
4. Choose the **Empty Activity** and click **Next**.



Create New Project

Add an activity to Mobile

5. In the **Activity Name:** box, write *HelloWorldActivity* and uncheck the **Generate Layout File**. Click **Finish**.


Create New Project

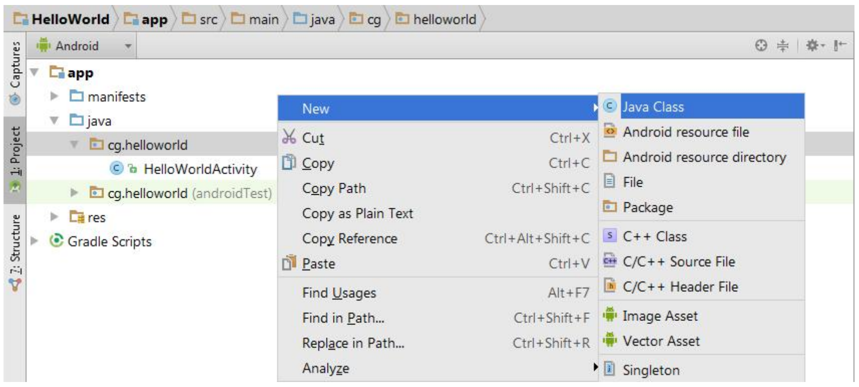
Customize the Activity

Creates a new empty activity

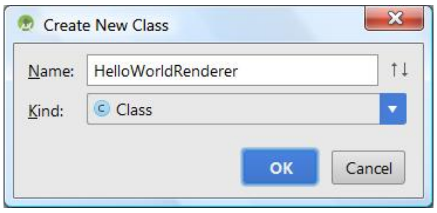
Activity Name: HelloWorldActivity

☐ Generate Layout File

6. Now, right click the *cg.helloworld* subdirectory of the *java* directory, and then choose the **New** > **Java Class** menu.



7. Name the new class *HelloWorldRenderer*.


Create New Class

Name: HelloWorldRenderer

Kind: Class

8. Type the following code in the *HelloWorldRenderer* class:

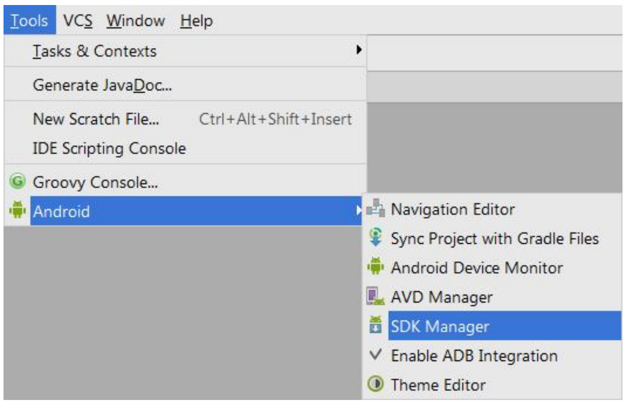
```
package cg.helloworld;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView;
```

```
public class HelloWorldRenderer implements GLSurfaceView.Renderer
{
    public void onSurfaceCreated(GL10 gl, EGLConfig config)
    {
        gl.glClearColor(1,0,0,1);
    }
    public void onSurfaceChanged(GL10 gl, int width, int height)
    {
        gl.glViewport(0, 0, width, height);
    }

    public void onDrawFrame(GL10 gl)
    {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    }
}
```

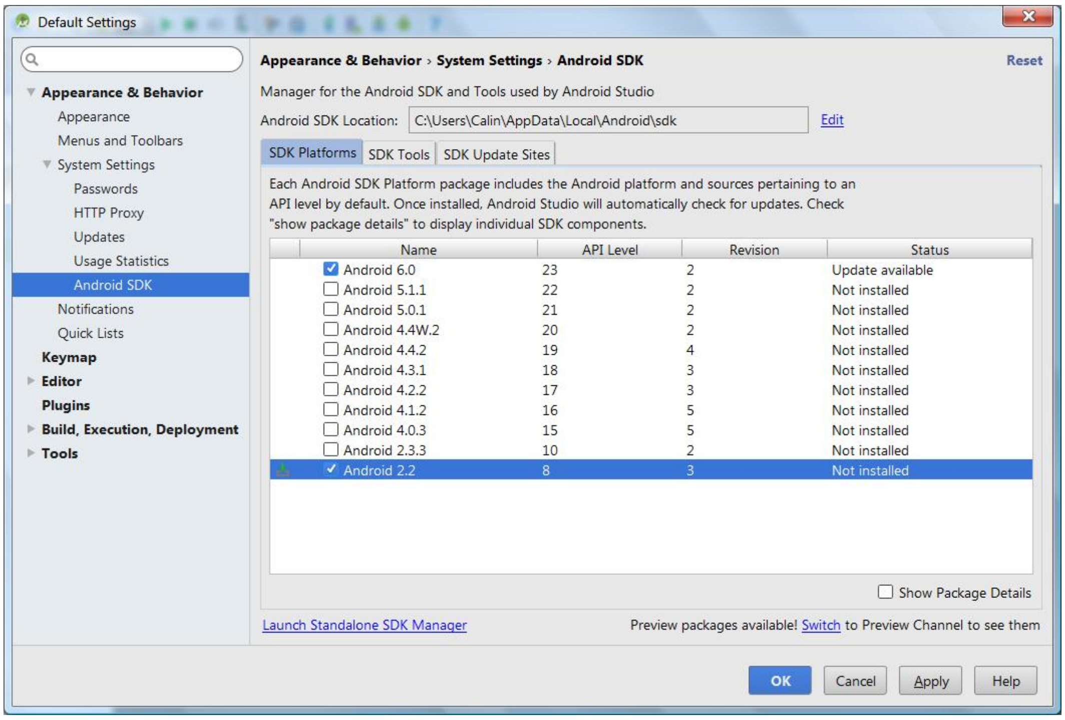
9. Type the following code in the *HelloWorldActivity* class:

```
package cg.helloworld;

import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;

public class HelloWorldActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        GLSurfaceView view = new GLSurfaceView(this);
        view.setRenderer(new HelloWorldRenderer());
        setContentView(view);
    }
}
```

10. (**DO NOT** do steps 10-18 in the laboratory, because the emulator is already set up; skip directly to step 19) Now, we need to set up the Android 2.2 emulator. For this, in the **Tools** menu, choose **Android** > **SDK Manager**.



11. Check the *Android 2.2* from the **SDK Platforms** tab:



Default Settings

Appearance & Behavior
- Appearance
- Menus and Toolbars
- System Settings
  - Passwords
  - HTTP Proxy
  - Updates
  - Usage Statistics
  - Android SDK
- Notifications
- Quick Lists

Keymap
Editor
Plugins
Build, Execution, Deployment
Tools

Appearance & Behavior › System Settings › Android SDK — Reset

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\Calin\AppData\Local\Android\sdk — Edit

SDK Platforms | SDK Tools | SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

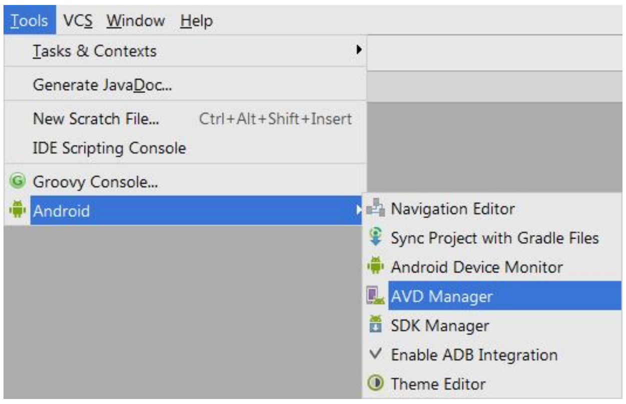
Name	API Level	Revision	Status
✓ Android 6.0	23	2	Update available
☐ Android 5.1.1	22	2	Not installed
☐ Android 5.0.1	21	2	Not installed
☐ Android 4.4W.2	20	2	Not installed
☐ Android 4.4.2	19	4	Not installed
☐ Android 4.3.1	18	3	Not installed
☐ Android 4.2.2	17	3	Not installed
☐ Android 4.1.2	16	5	Not installed
☐ Android 4.0.3	15	5	Not installed
☐ Android 2.3.3	10	2	Not installed
✓ Android 2.2	8	3	Not installed

☐ Show Package Details

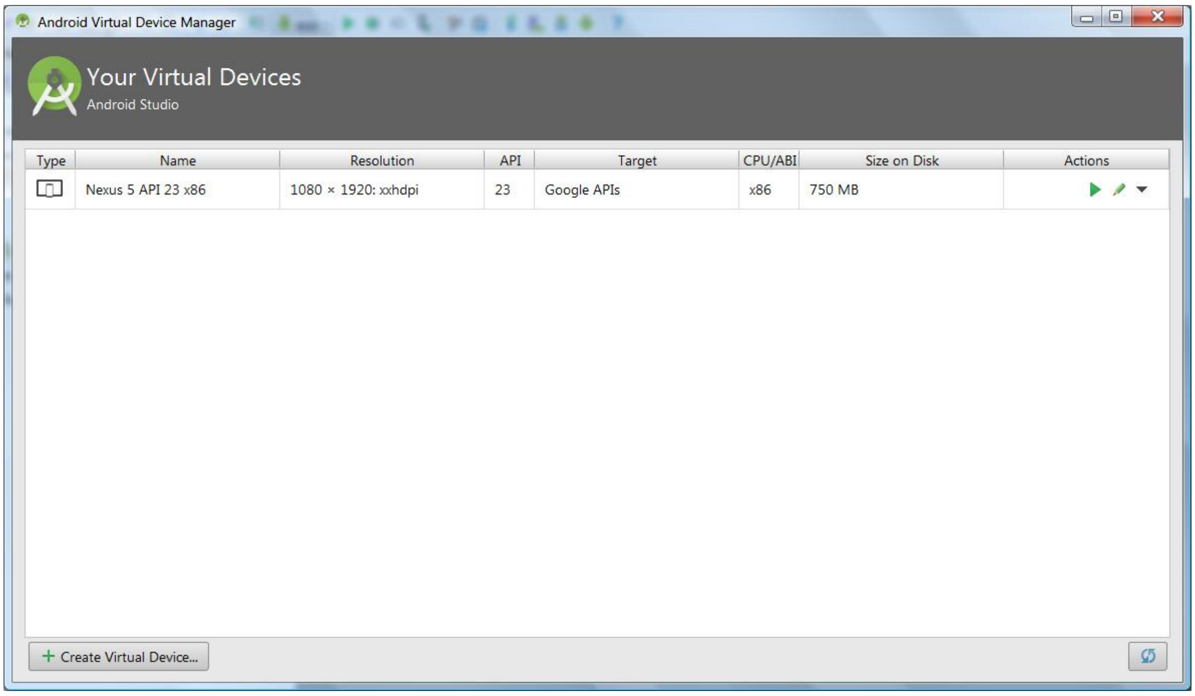
Launch Standalone SDK Manager — Preview packages available! Switch to Preview Channel to see them

OK | Cancel | Apply | Help

12. Now, from the **Tools** menu, choose **Android** > **AVD Manager**.



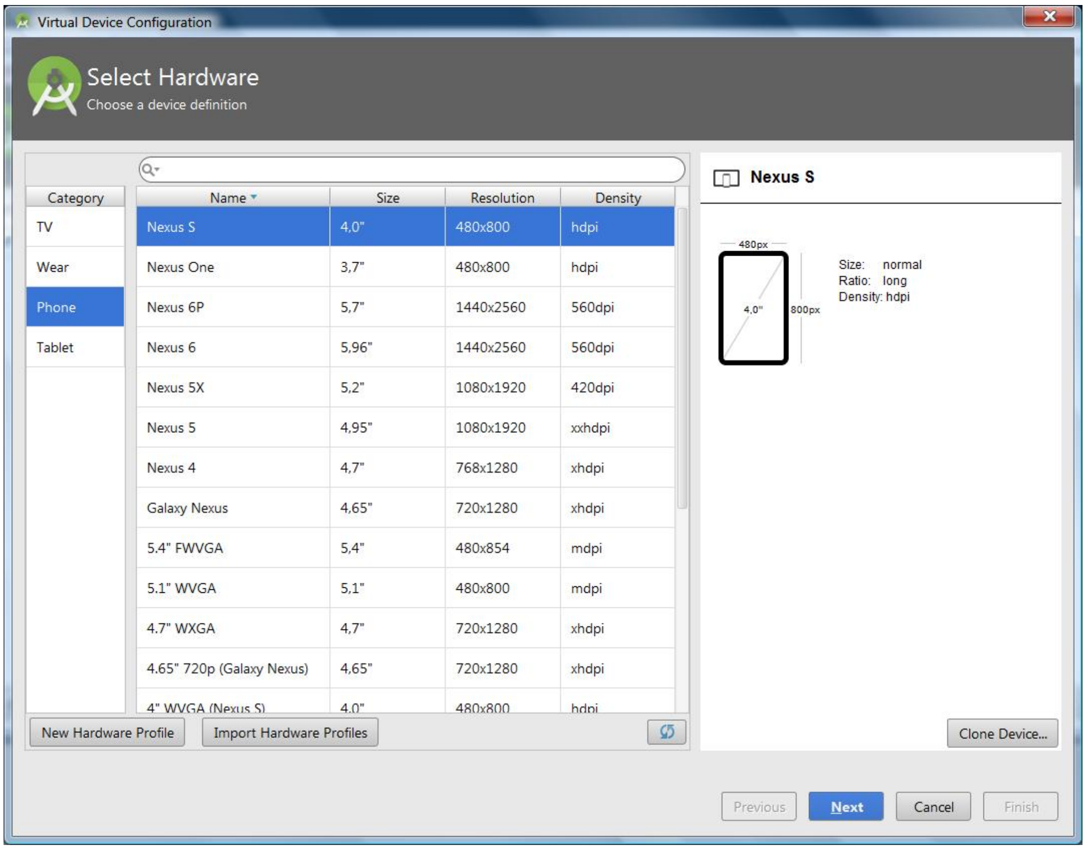
13. The **Android Virtual Device Manager** dialog appears. Click the **Create Virtual Device...** button.



Your Virtual Devices
Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5 API 23 x86	1080 × 1920: xxhdpi	23	Google APIs	x86	750 MB	

14. Choose the *Nexus S* **Phone** and click **Next**.



Select Hardware
Choose a device definition

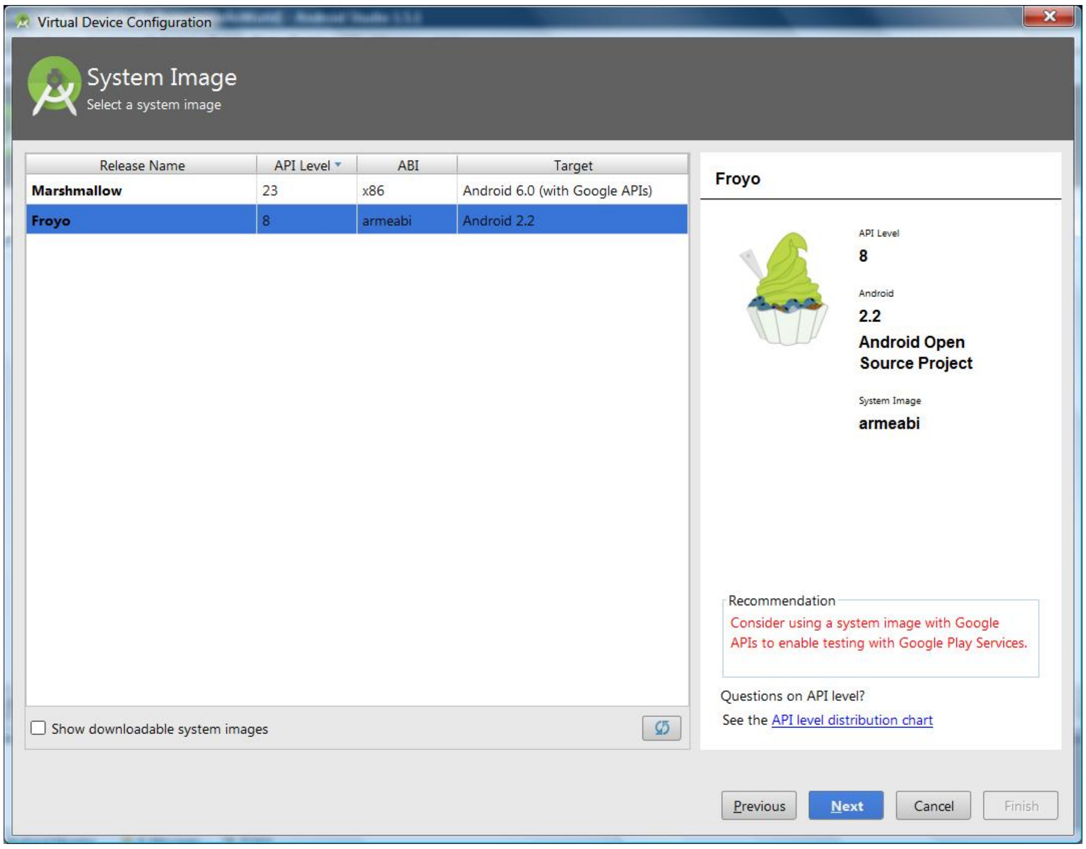
Category
TV
Wear
Phone
Tablet

Name	Size	Resolution	Density
Nexus S	4,0"	480x800	hdpi
Nexus One	3,7"	480x800	hdpi
Nexus 6P	5,7"	1440x2560	560dpi
Nexus 6	5,96"	1440x2560	560dpi
Nexus 5X	5,2"	1080x1920	420dpi
Nexus 5	4,95"	1080x1920	xxhdpi
Nexus 4	4,7"	768x1280	xhdpi
Galaxy Nexus	4,65"	720x1280	xhdpi
5.4" FWVGA	5,4"	480x854	mdpi
5.1" WVGA	5,1"	480x800	mdpi
4.7" WXGA	4,7"	720x1280	xhdpi
4.65" 720p (Galaxy Nexus)	4,65"	720x1280	xhdpi
4" WVGA (Nexus S)	4,0"	480x800	hdpi

Nexus S

Size: normal
Ratio: long
Density: hdpi

15. Choose the *Froyo (Android 2.2)* **System Image** and click **Next**.



Release Name	API Level	ABI	Target
Marshmallow	23	x86	Android 6.0 (with Google APIs)
Froyo	8	armeabi	Android 2.2

Show downloadable system images

Froyo

API Level
8

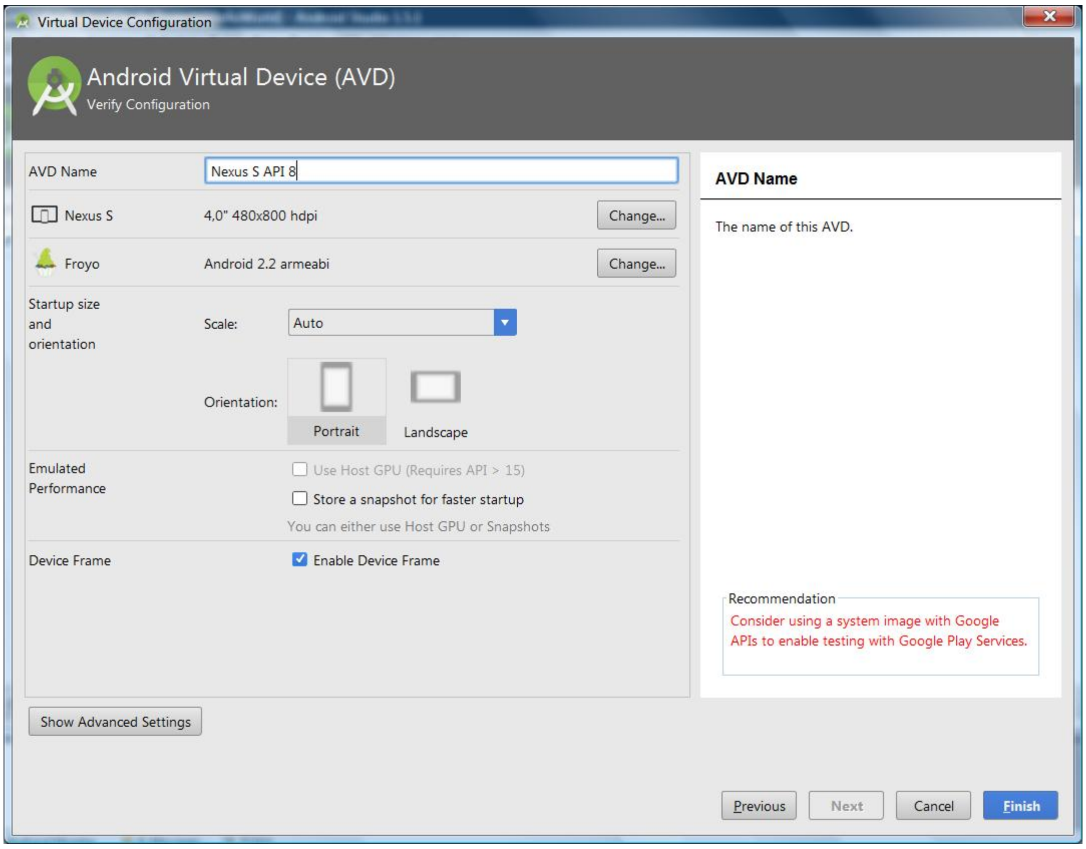
Android
2.2
Android Open Source Project

System Image
armeabi

Recommendation
Consider using a system image with Google APIs to enable testing with Google Play Services.

Questions on API level?
See the API level distribution chart

16. Click **Finish**.



AVD Name: Nexus S API 8

Nexus S: 4,0" 480x800 hdpi

Froyo: Android 2.2 armeabi

Startup size and orientation: Scale: Auto; Orientation: Portrait / Landscape

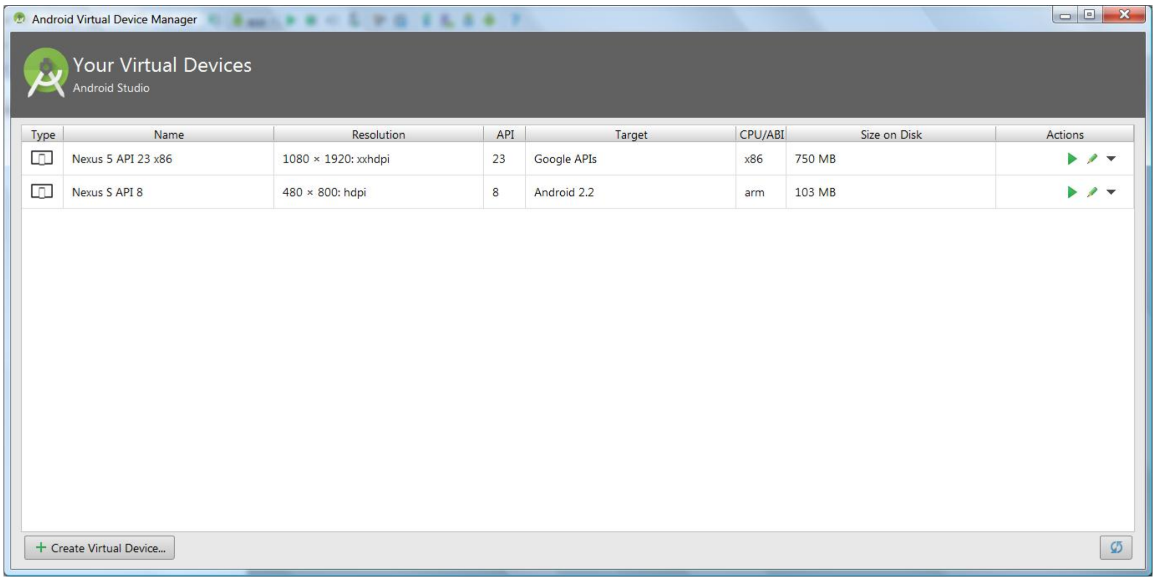
Emulated Performance: Use Host GPU (Requires API > 15); Store a snapshot for faster startup
You can either use Host GPU or Snapshots

Device Frame: Enable Device Frame ✓

AVD Name
The name of this AVD.

Recommendation
Consider using a system image with Google APIs to enable testing with Google Play Services.

17. Now, the **Android Virtual Device Manager** dialog looks like this:

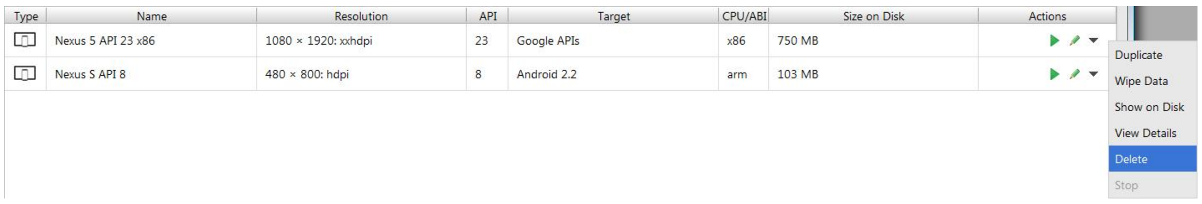


Android Virtual Device Manager

Your Virtual Devices
Android Studio

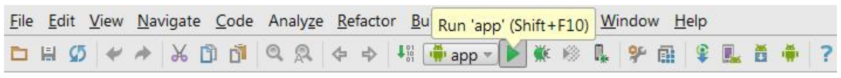
Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5 API 23 x86	1080 × 1920: xxhdpi	23	Google APIs	x86	750 MB	
	Nexus S API 8	480 × 800: hdpi	8	Android 2.2	arm	103 MB	

18. For convenience, you can delete the *Nexus 5 API 23 x86* default emulator.

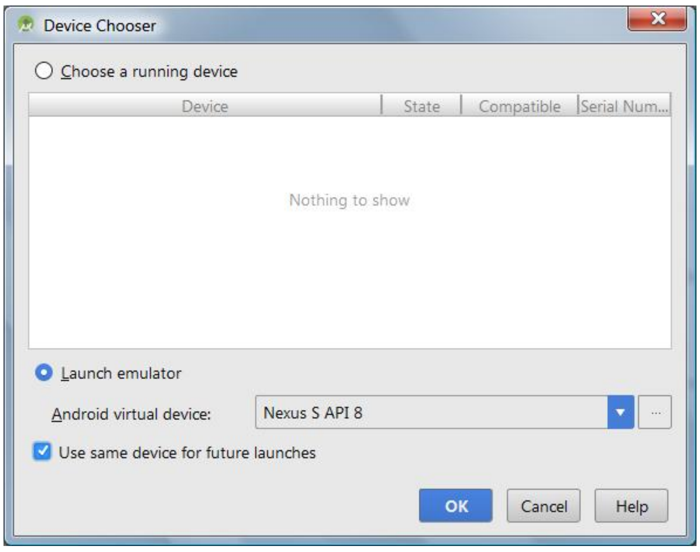


Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5 API 23 x86	1080 × 1920: xxhdpi	23	Google APIs	x86	750 MB	
	Nexus S API 8	480 × 800: hdpi	8	Android 2.2	arm	103 MB	

19. Now, it's time to **Run** our *HelloWorld* project. Click the **Run** button on the toolbar:



20. Choose the only remaining emulator, namely *Nexus S API 8*. Be sure to check the **Use same device for future launches**.



Device Chooser

( ) Choose a running device

Device	State	Compatible	Serial Num...

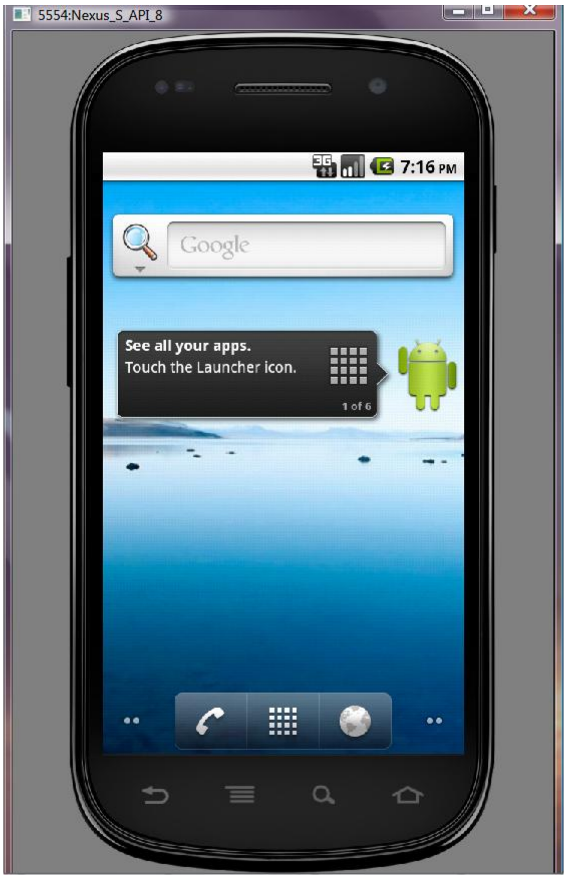
Nothing to show

(●) Launch emulator

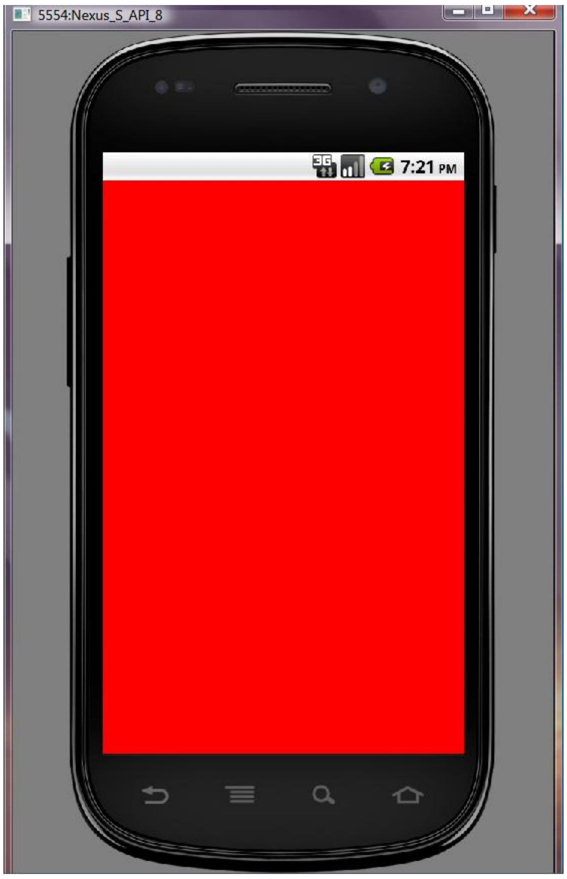
Android virtual device: Nexus S API 8

[✓] Use same device for future launches

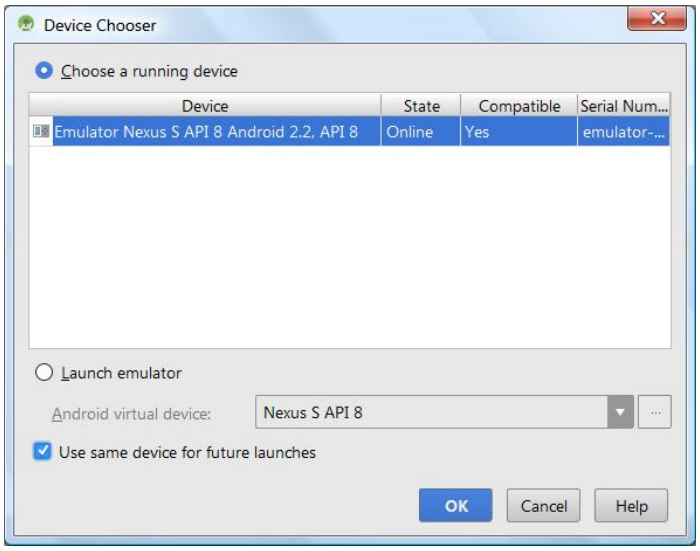
21. The *Nexus_S_API_8* emulator appears:



22. And, shortly after, the *HelloWorld* project produces the following output:



23. You can modify the background color, and **Run** the project again, using the opened emulator. Be sure to check the **Use same device for future launches** again.



Device Chooser

(●) Choose a running device

Device	State	Compatible	Serial Num...
Emulator Nexus S API 8 Android 2.2, API 8	Online	Yes	emulator-...

( ) Launch emulator

Android virtual device: Nexus S API 8

[✓] Use same device for future launches