Tutorial – Operating Systems and Virtual Machines

Note: This tutorial requires that a number of pieces of software are downloaded and installed. To save time in class it would be beneficial to download these in advance if possible.

In this tutorial we will set up an Android Virtual Device and deploy a Unity game to it. This can be a lengthy process the first time through as there are many parts which must be configured correctly to interoperate with one another, but it will get easier with practice and as you gain a better understanding of each of the components. This will also help you in deploying games to real Android devices, as much of the process is the same.

Take your time, read the instructions in detail, and follow them carefully. If you get stuck, then don't be afraid to ask for help or clarification.

Preparation - You will need:

- **Several gigabytes of free space.**
    - Requirements will vary depending on software versions. 10GB or more is recommended.

- **Java SE Development Kit**, or "JDK".

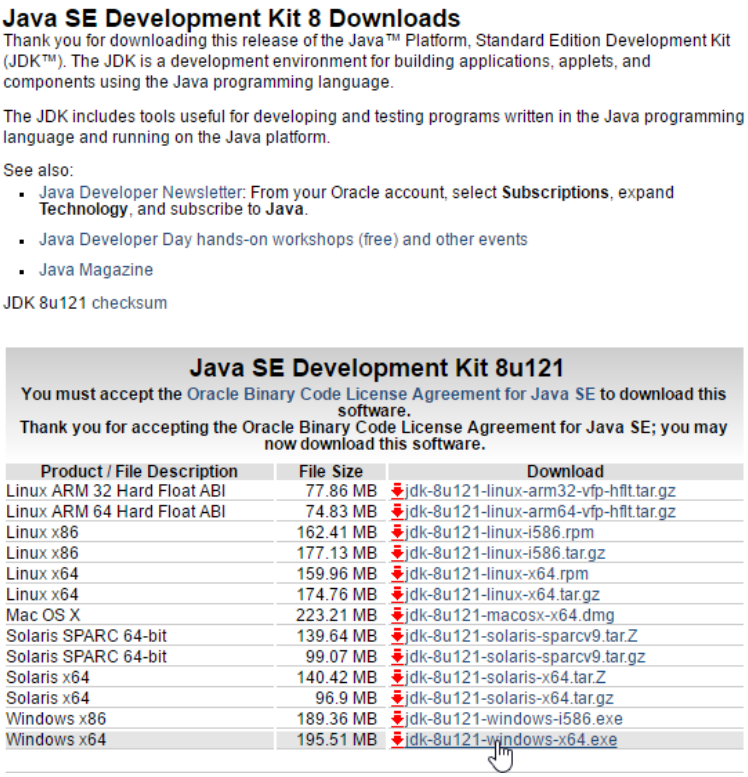
  Note: Downloading and installing the Java JDK is only necessary if you want to build your Unity games for Android phones and deploy them to either the emulator or a real phone.

  Creating Android games in Unity is discussed in the second part of this tutorial.

  This is not necessary of any assessment item. Only installing and running the emulator, discussed in the first part of this tutorial, is an assessable item.

  If you do not which to complete the second part of this tutorial then it is not necessary to download and install the JDK

    - This tutorial is written using JDK 8u121, accessible at the time of writing here: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
    - You will need to download the appropriate version for your operating system - Mac OS X, Windows x86 (32 bit) or Windows x64 (64 bit).



Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter: From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK 8u121 checksum

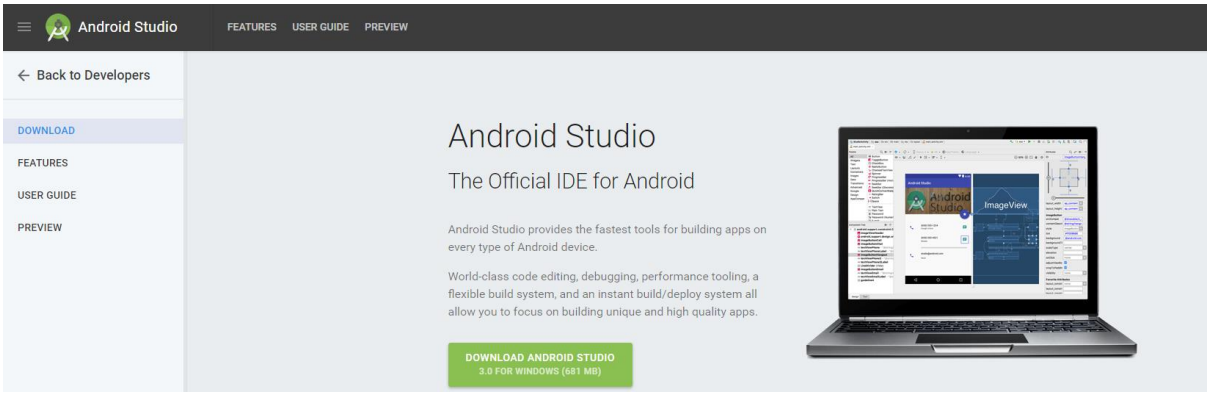
Java SE Development Kit 8u121

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

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

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

- **The Android Studio.**
  - Can be downloaded from here: https://developer.android.com/studio/index.html
    This tutorial covers Android Studio version 3.0



- **Computer specifications.**
  - If you have multiple computers, this tutorial is best performed on the most powerful one you have available.
  - Ideally you will want 4+gb of RAM and a fast CPU.

Software Installation and Configuration:

The first thing we need to do is install our software. These installations provide us with two things. The first is the relevant libraries and tools which are required to create and deploy software to Android devices. The second is the ability to create Android Virtual Devices, which we will use for our testing.

Installing the Java Development Kit:

1. If you haven't done so already, download the **Java SE Development Kit** from the link listed in the above *Preparation* section.

2. Install the Java SE Development Kit you have downloaded by running the installer and following the prompts. When the installation is complete click "Close" - we don't need to look at the "Next Steps" provided by Oracle.

   Make sure you make a note of where you installed this. We'll need the installation path later when setting up Unity.

Installing the Android Studio:

Getting the *Android Studio 3.0* set up with all the necessary components is an involved process. While not necessarily difficult, it is time consuming and will involve anywhere up to 5GB of downloads.

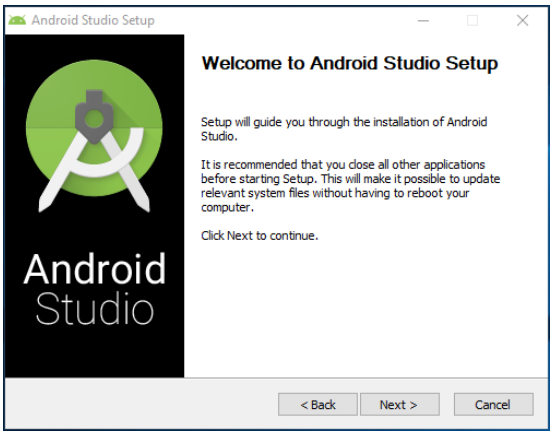
The *Android Studio* installer is around 700MB, but does not include all the components we will need for this tutorial. As you set up and create projects, *Android Studio* will download components as necessary. For this reason, make sure you have enough bandwidth when completing this step.

Our basic approach when setting up *Android Studio* will be to install the base program, then create a blank test project. As we build the test project, *Android Studio* will download the additional components we need to make and run an *Android* phone emulator. Although there are a lot of steps, this process is the easiest way to install everything we need.

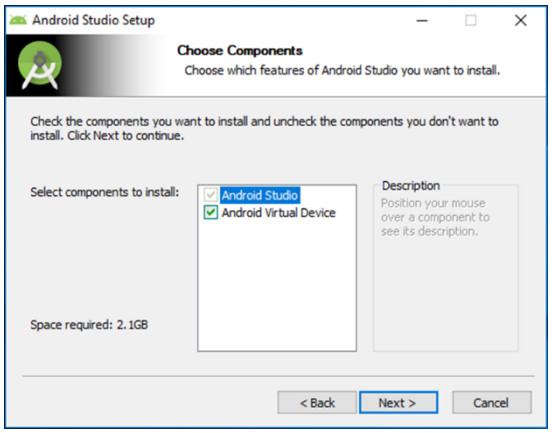
1. If you haven't done so already, download the **Android Studio** listed above in the *Preparation* section.

2. Run the installer for the Android Studio. You can install this program anywhere on your computer.

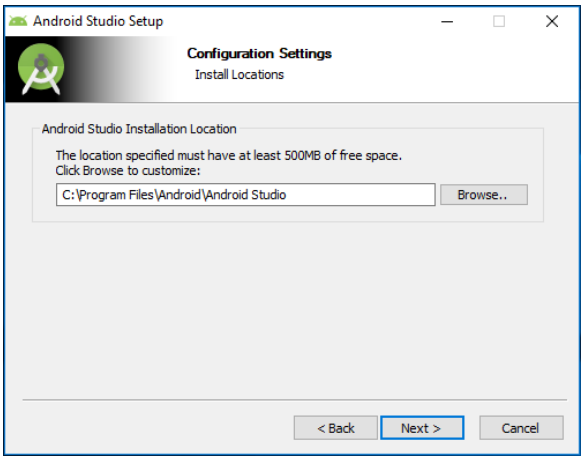
The first screen is the installer splash screen. Simply press the *Next* button.



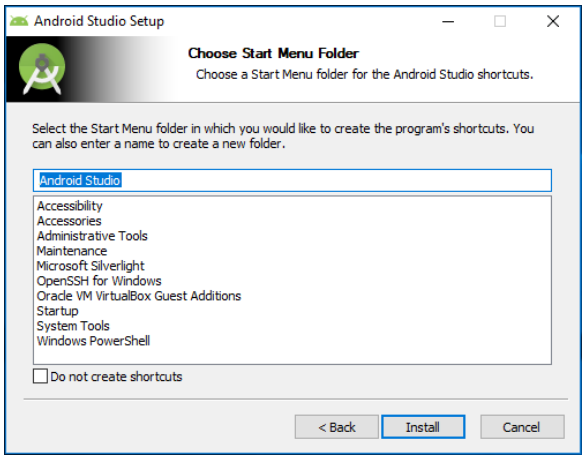
3. The next screen to display is for choosing the components to install. Make sure all boxes are ticked and press *Next*.



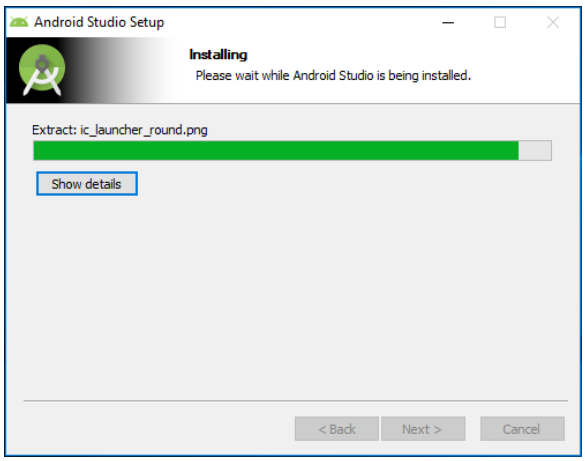
4. Next, choose the installation directory.
   Make a note of this location. We'll need the installation path later when setting up Unity.



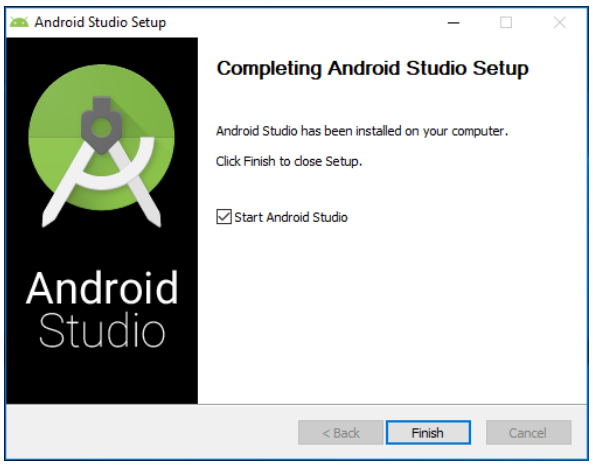
5. Choose the *Start Menu* folder. Selecting the default option on this screen is fine.



6. The *Android Studio* will now install. This should not take too long.

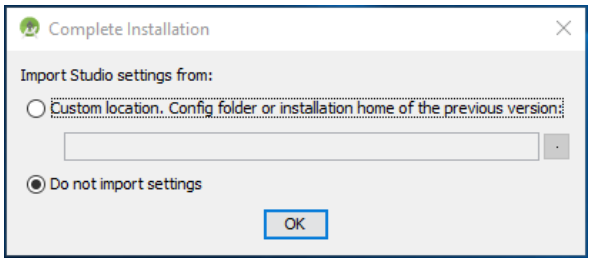


7. Once installation has completed, start *Android Studio*

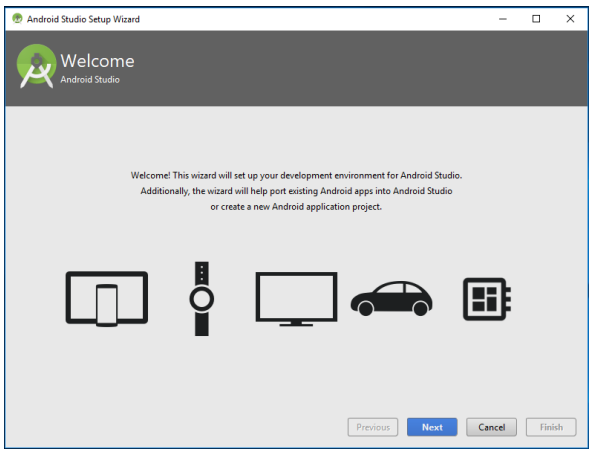


8. When *Android Studio* starts for the first time, it will give you the option to import settings from any previously installed versions.

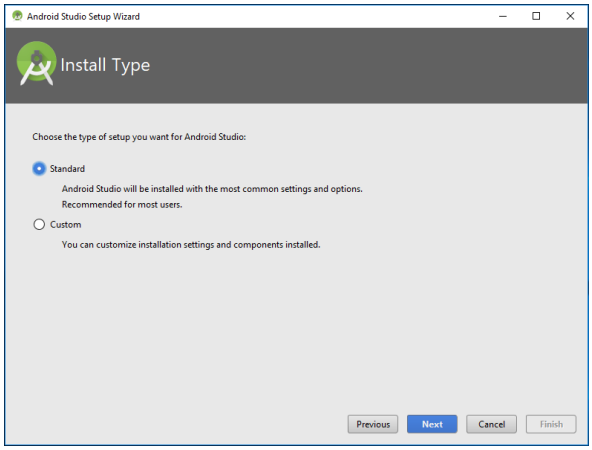
You can select '*Do not import settings*'.



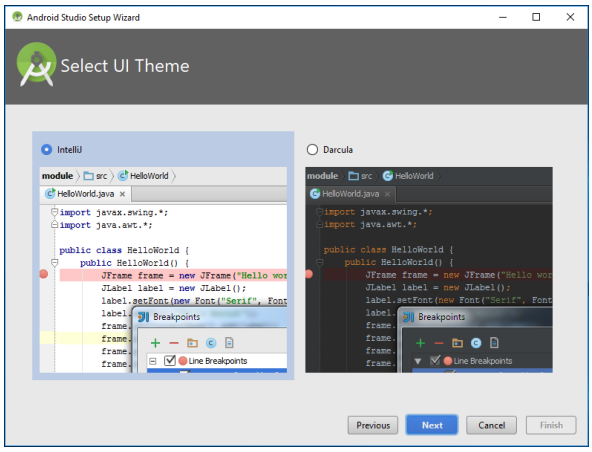
9. *Android Studio* will then launch its *Setup Wizard*. Click through these screens accepting the default options.



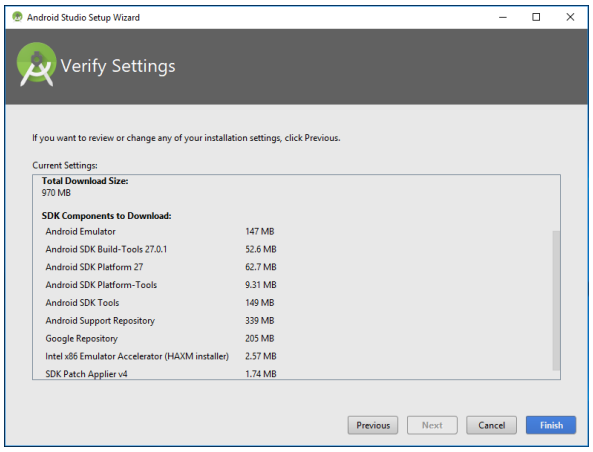
Choose the *Standard* setup option:



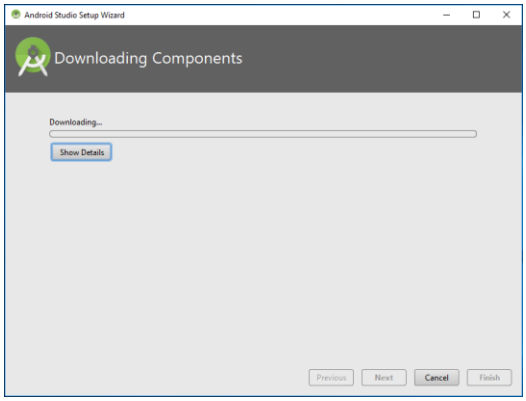
Select your preferred theme, or leave the default selected:



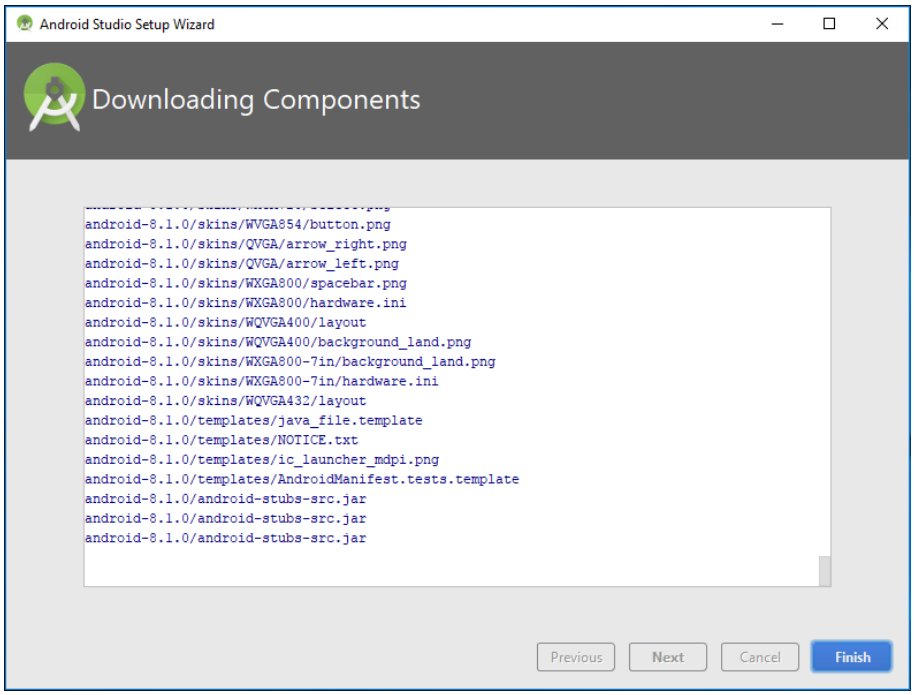
Finish the installation:



Additional components will be downloaded automatically:



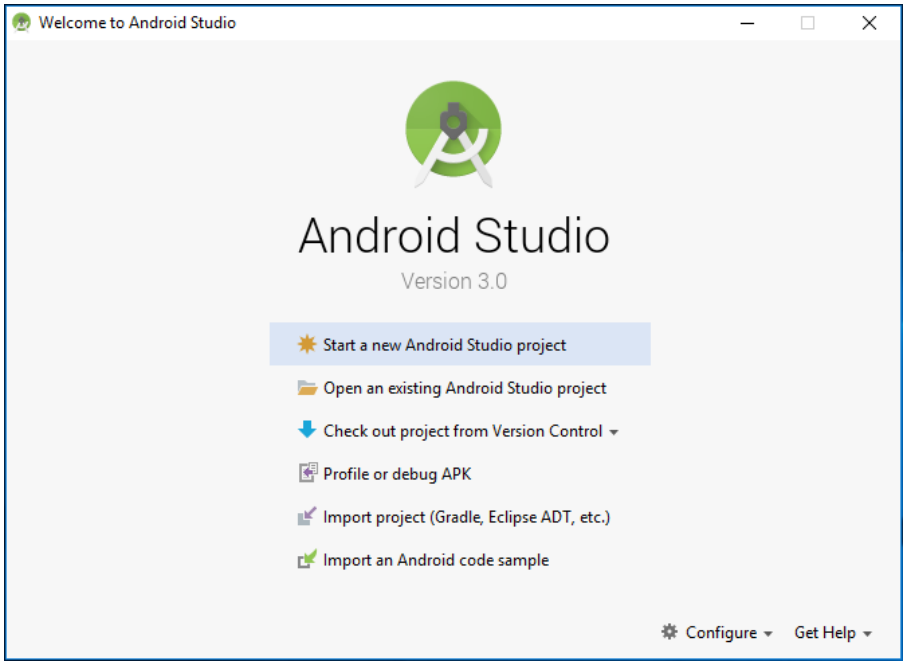
10. When all components have downloaded and installed, press *Finish* to start *Android Studio*.



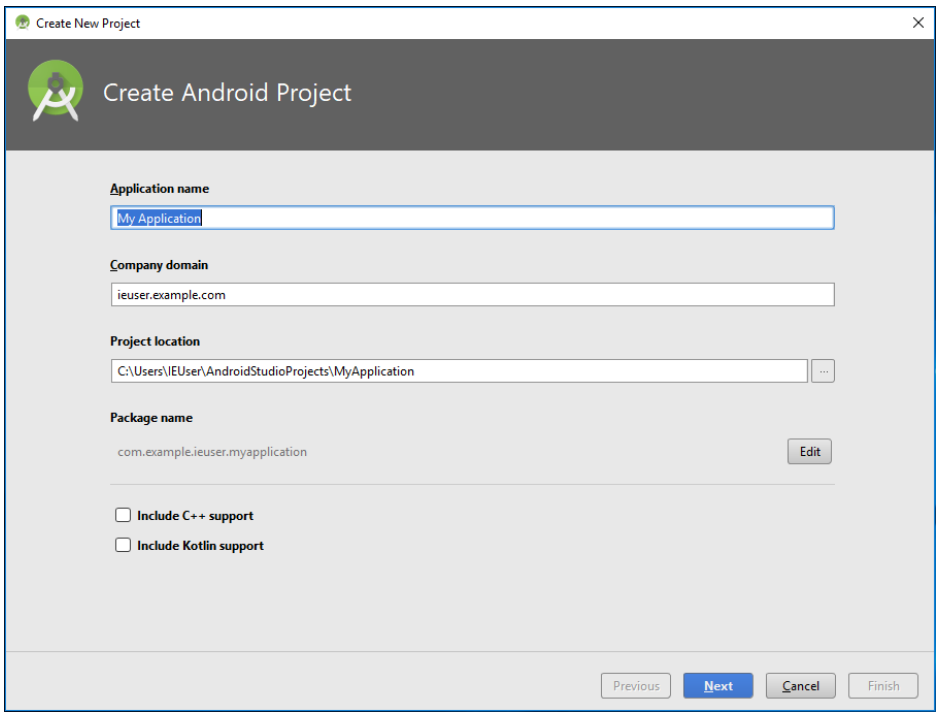
11. When *Android Studio* starts, you'll be presented with a screen listing several options. To get anywhere, we're going to need to make a project.

We aren't actually going to use the project, but we need to make it so that we can get access to the tools we want to use.

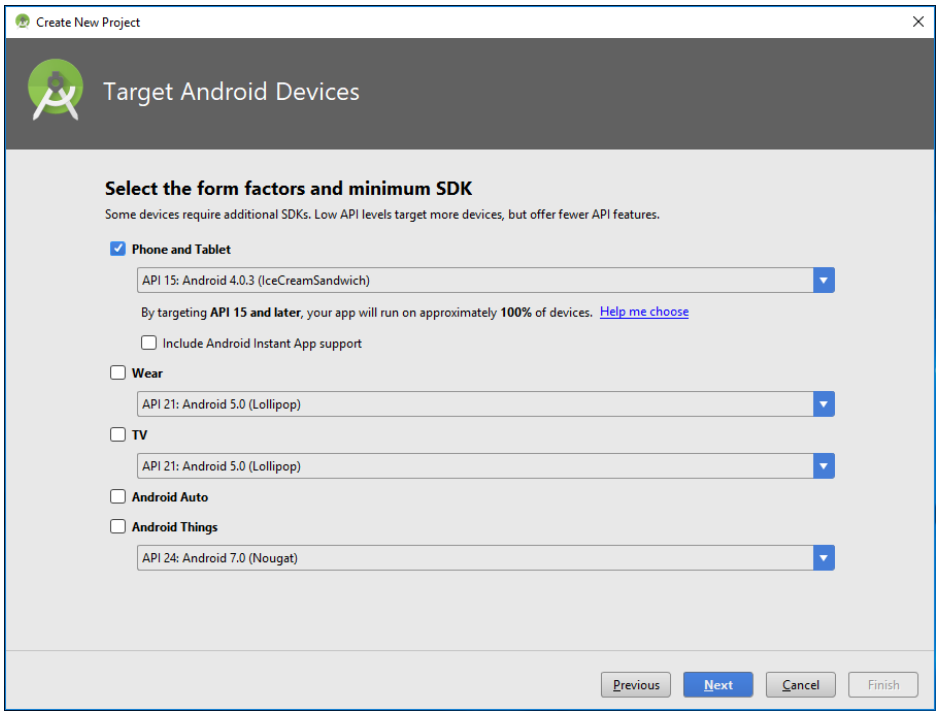
Select the first menu option, '*Start a new Android Studio project*'.



12. On the *Create New Project* screen, select the default options:



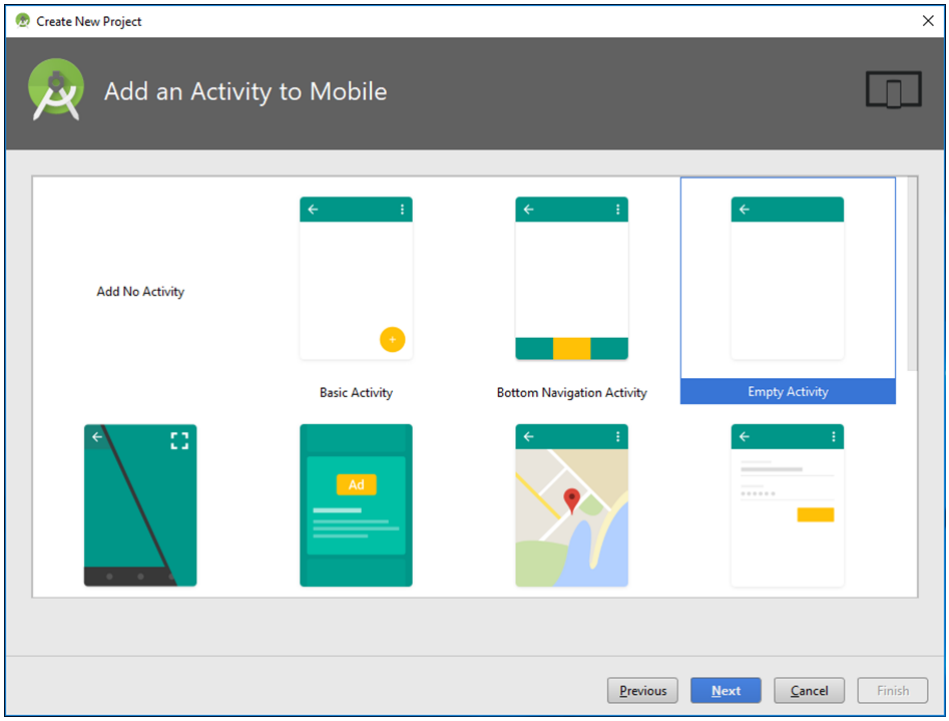
13. For *Target Android Devices*, ensure only *Phone and Tablet* is selected. This will be the default option:



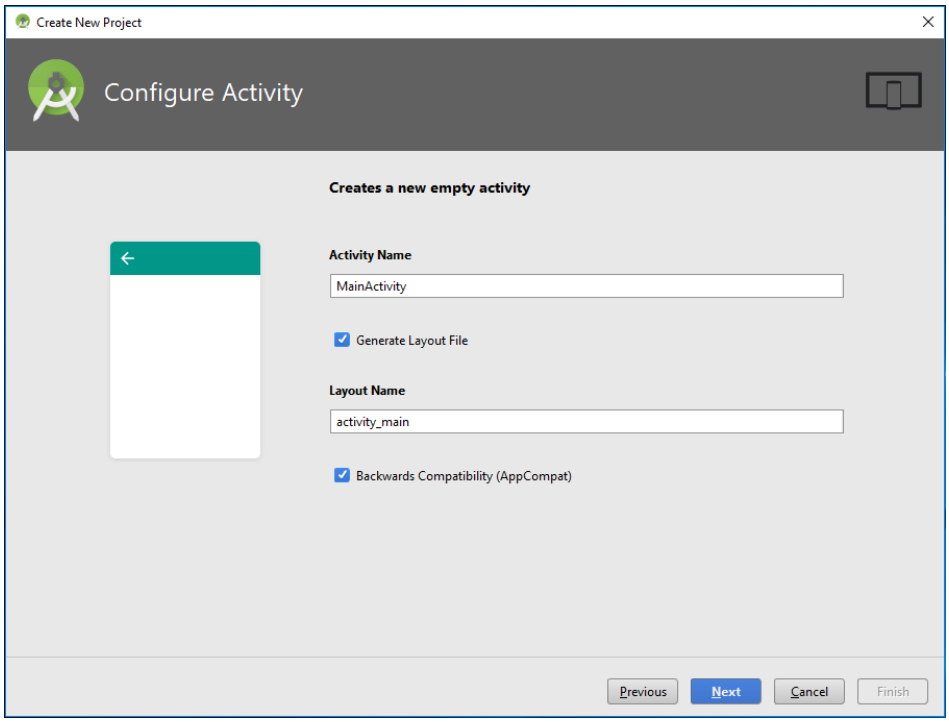
14. The next screens allow you to select what type of mobile project to create.

    Since we don't actually want to use the project for anything, we don't really care.

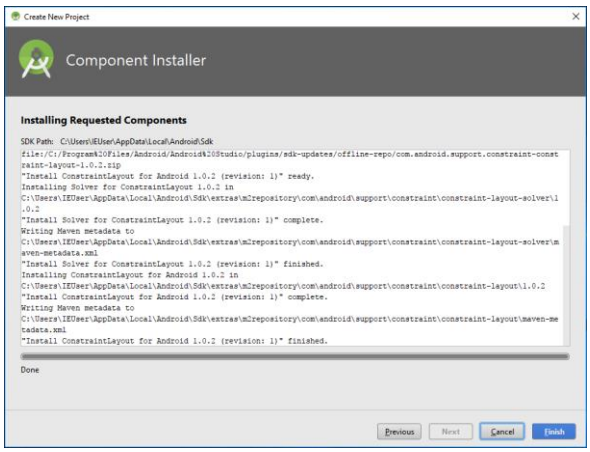
    You will want to select the default options on these screens to ensure our project builds fast and doesn't require too many additional components to be installed.
    Select the default option on this screen:



Leave the default values in place on this screen too:

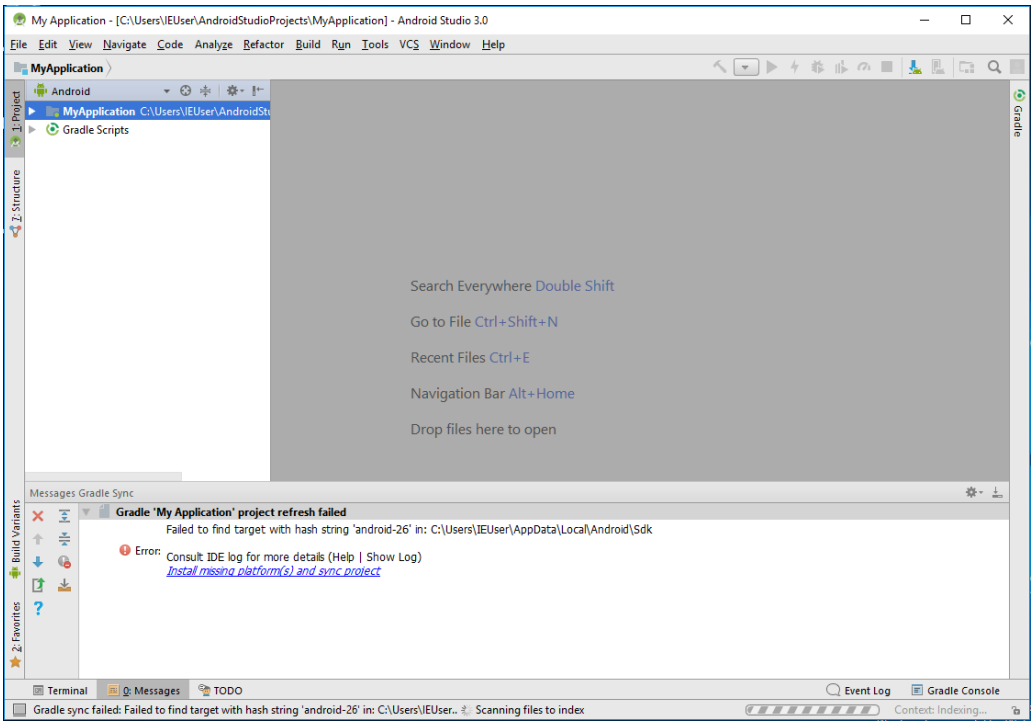


15. *Android Studio* will now install the components required to build this project:



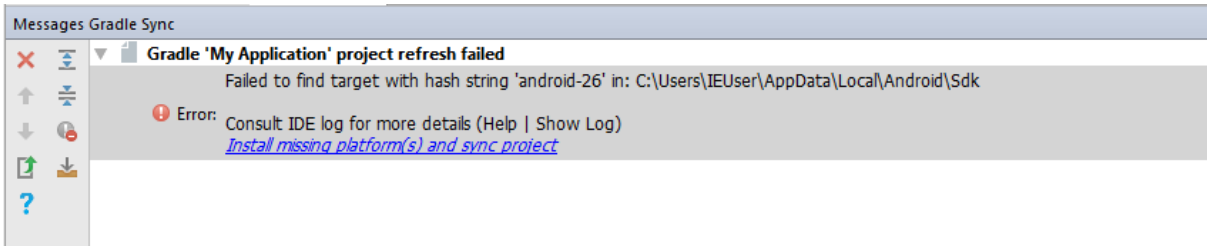
Press *Finish* once components have been installed, and your project will start building:

16. Finally, *Android Studio* will open:



Although we just installed the program, *Android Studio* still doesn't have everything it requires to completely build and execute our project on an *Android Emulator*.

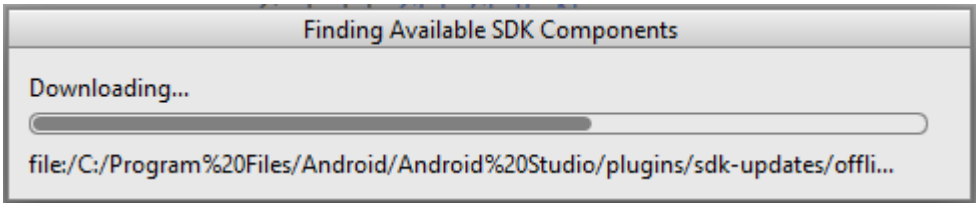
17. At the bottom of the window, you will see this error message:



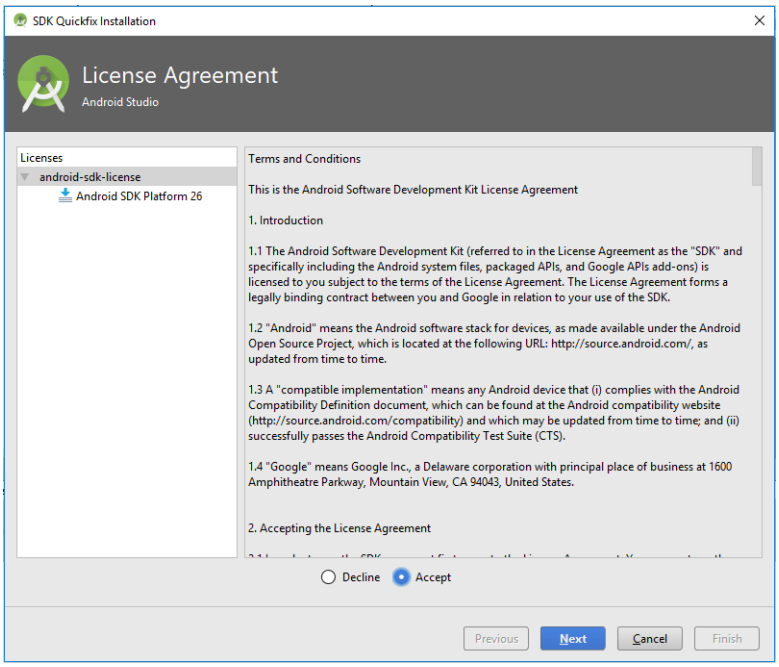
This means that there are still missing components that are preventing our project from running.

Click on the blue text '*Install missing platform(s) and sync project*'

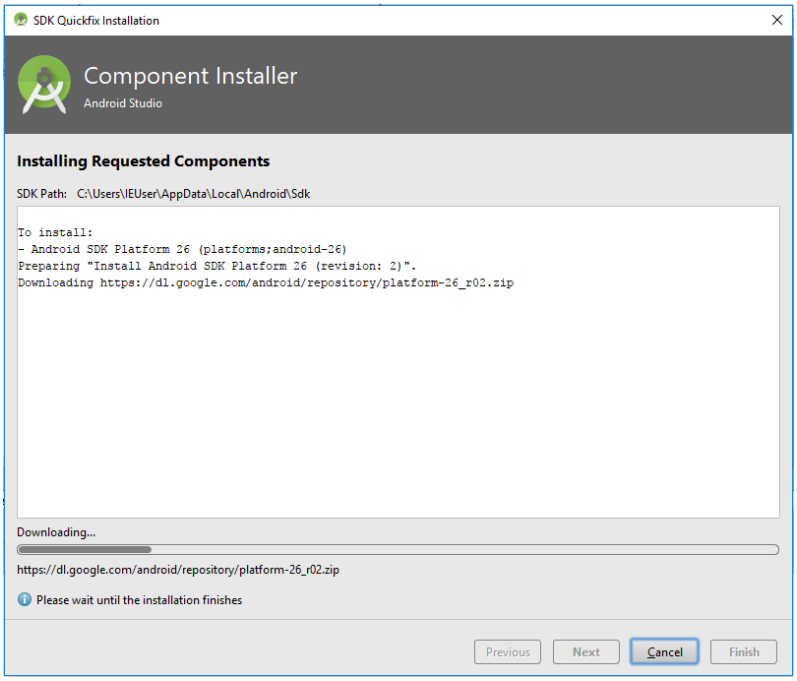
By pressing this link, *Android Studio* will start to automatically resolve the problem by downloading the missing components:



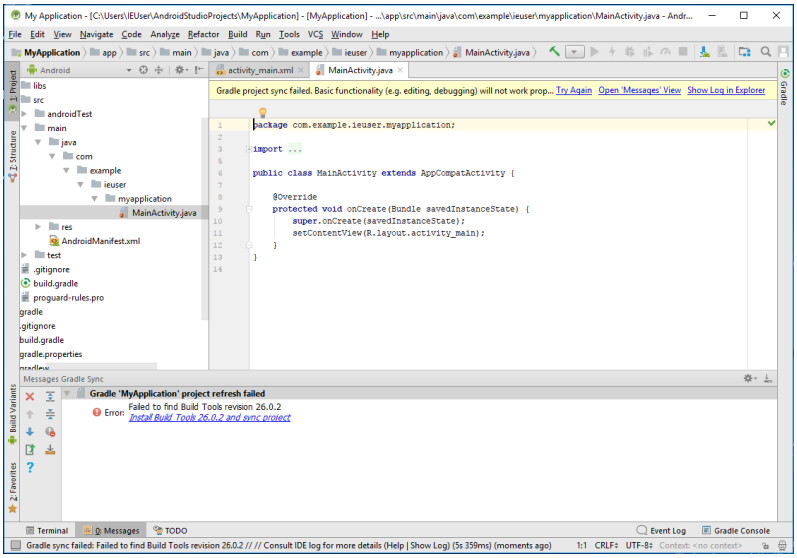
18. Accept the licence agreements and the SDK components will be installed:

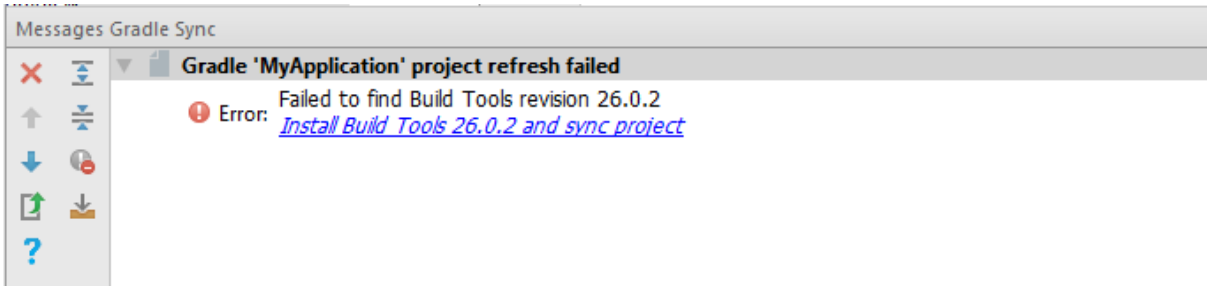


19. Once installation has completed, press *Finish* to return to *Android Studio*



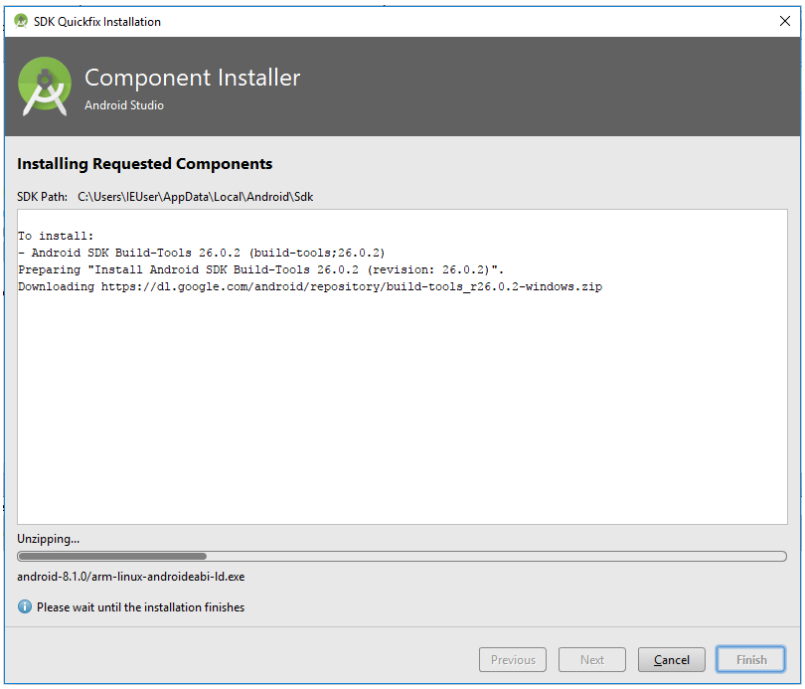
20. Your project will attempt to build again, and again you'll be presented with an error message (this might take a minute or two):





21. Like before, click the blue text *'Install Build Tools 26.0.2 and sync project'*.

    The build tools will be installed.



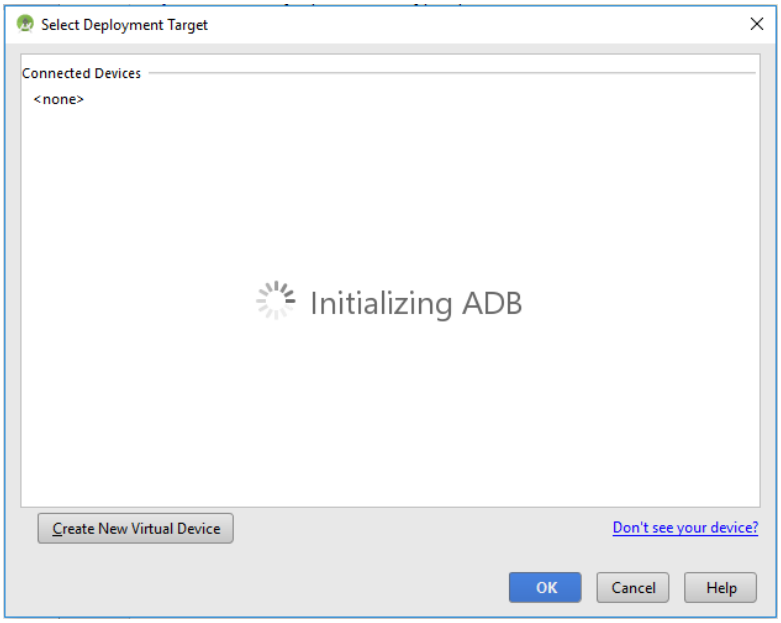
    Once this has finished, press the *Finish* button.

22. We are now ready to run our project.

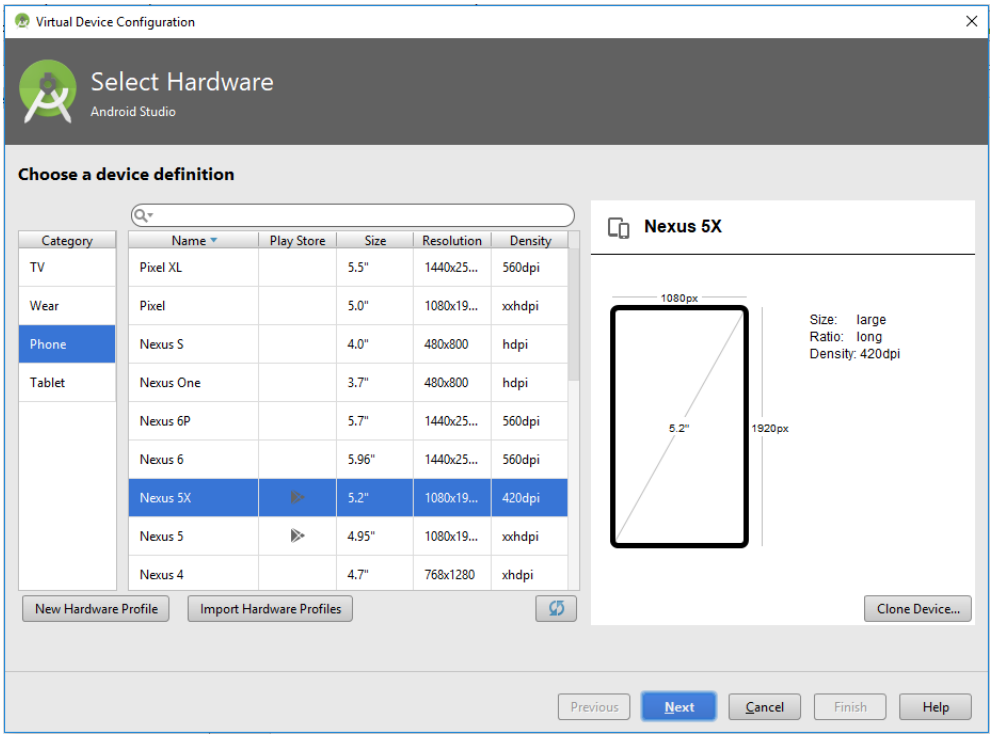
Find the toolbar at the top of the main window containing a *Play* button:

Press the *Play* button: 

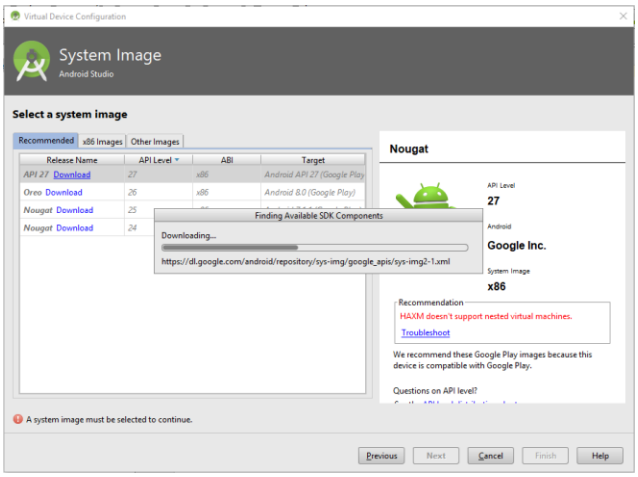
23. We'll now be presented with a screen that will allow us to set up an emulated device (phone) to use to run our simple mobile project.



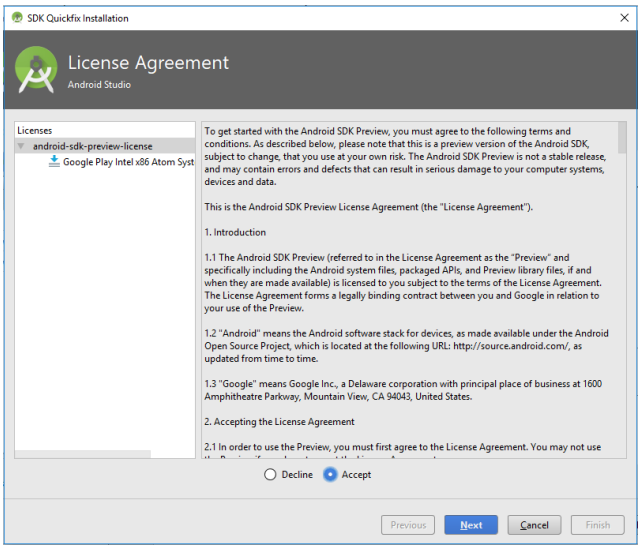
24. Select the default options. Here, the *Nexus 5X* is selected.
You may want to experiment with different devices, but it is best to do this after you have completed this tutorial.



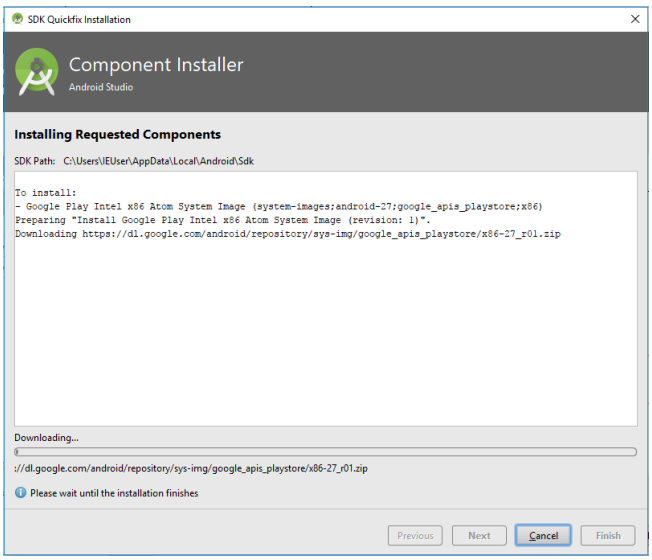
The device emulator will then be downloaded and installed:



And you'll be asked once again to accept a licence agreement:

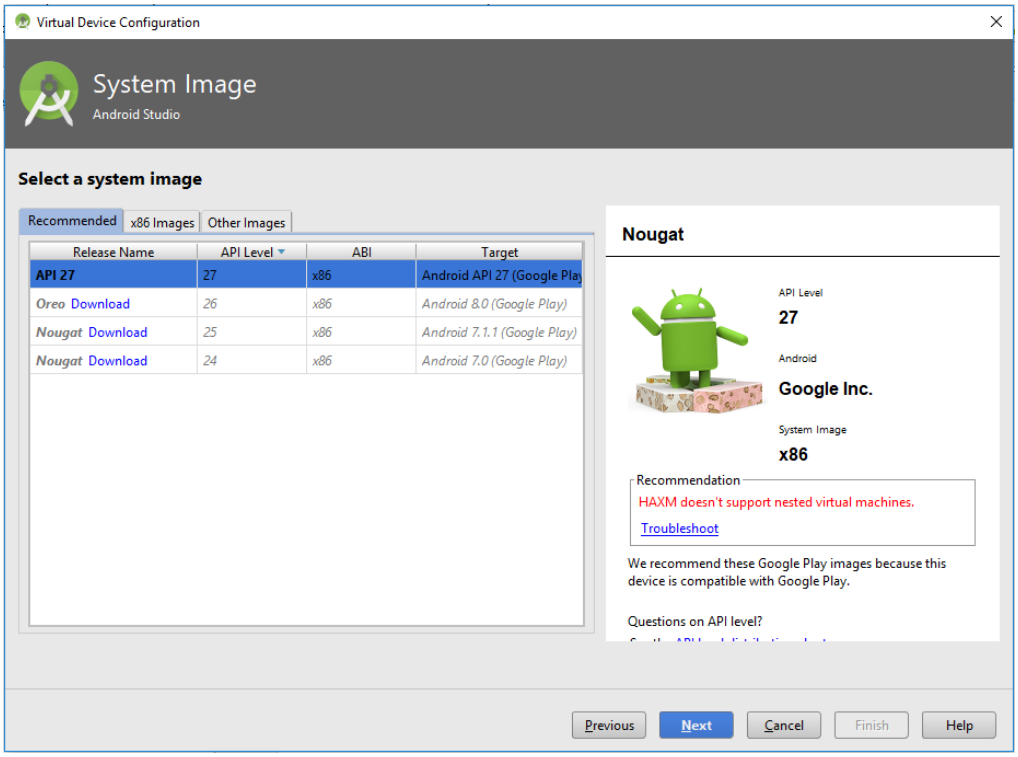


When the components have finished installing, press *Finish*:

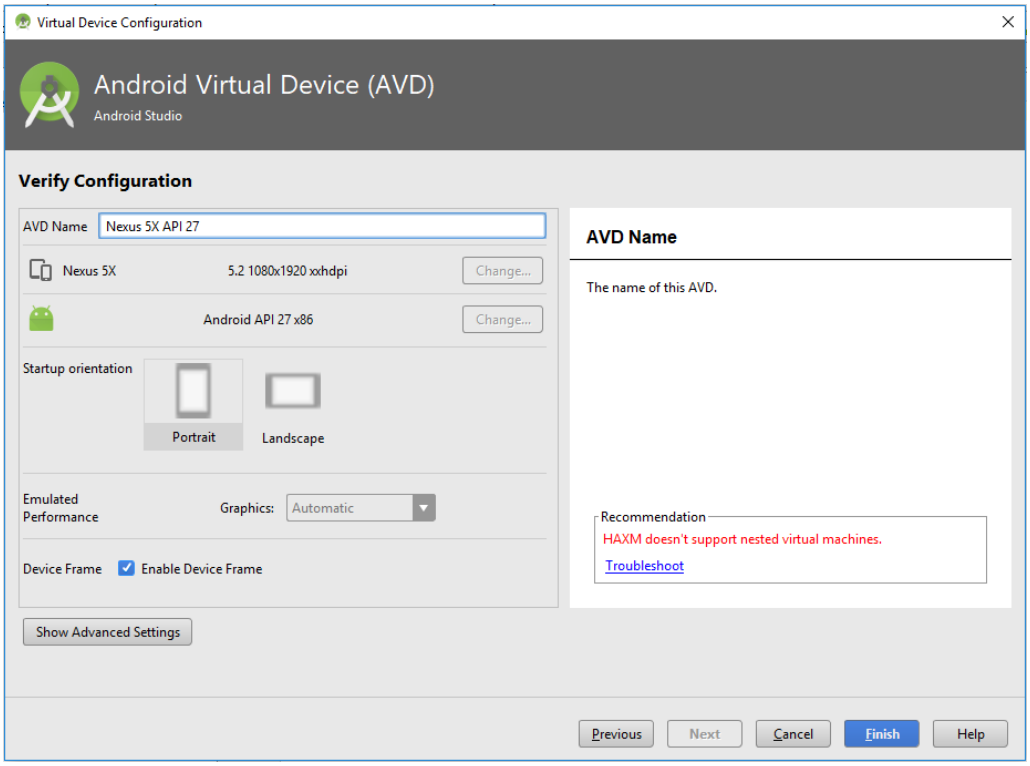


25. You'll then need to select a *System Image* to install onto the emulator. This is the actual *Android* operating system.

    Once again, stick with the default selection and press *Next*:

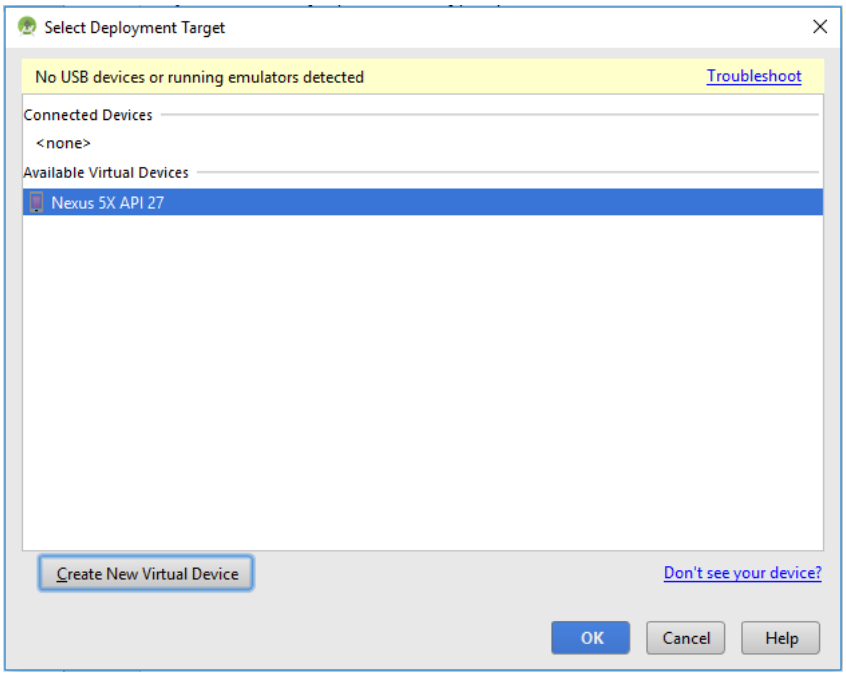


26. Finally, select *Finish*:



27. The *Virtual Device Configuration* wizard will now close, and you'll be taken back to a previous screen.

Select your newly created device emulator from the list, and press *OK*.



28. The *Android* emulator will now start.
    You should see a basic mobile app displaying the text 'Hello World'.

After you've verified that the emulator is functioning, you can close the emulator window.

Launching an Android Virtual Device:

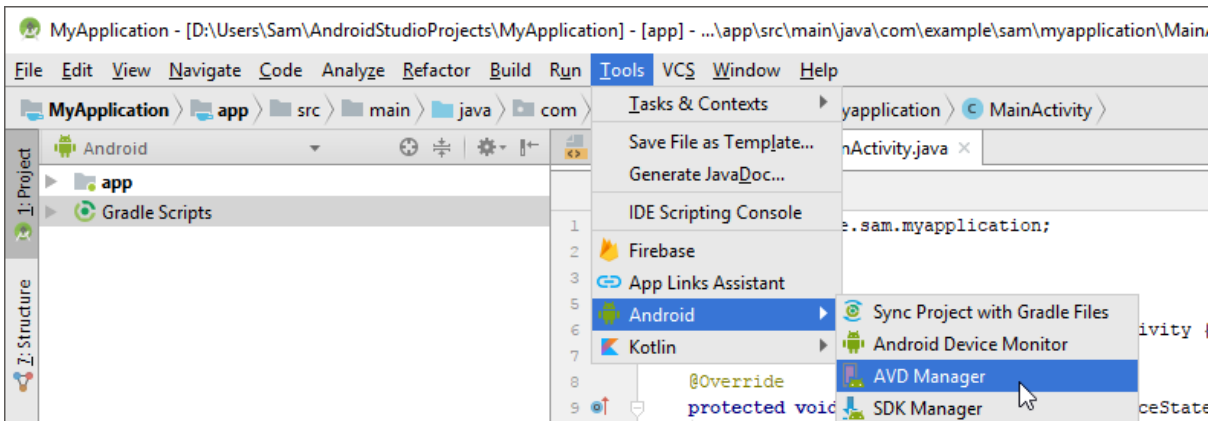
In the steps above, we created a new *Android Virtual Device* (i.e., an emulator), and verified that it worked using a simple mobile app created in *Android Studio*.

In the next section, we will explain deploying our *Unity* game to the emulator. But first, we should know how to open the emulator so we can access it whenever we need it.

Whenever you open *Android Studio* from now on, it will open the last project you worked on.

This is fine, as we'll need to have an open project to actually access the tools we need.

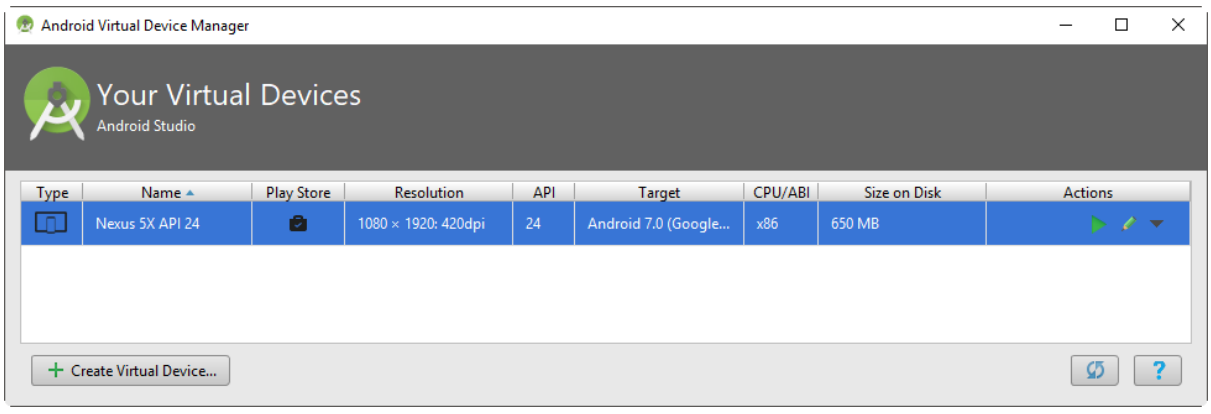
From the *Tools* menu, select *Android -> AVD Manager*



This will open the *Android Virtual Device Manager* dialog, where you can access all the virtual devices (emulators) you have made.

Your *Android Virtual Device Manager* should now list one device – the *Nexus 5X*.

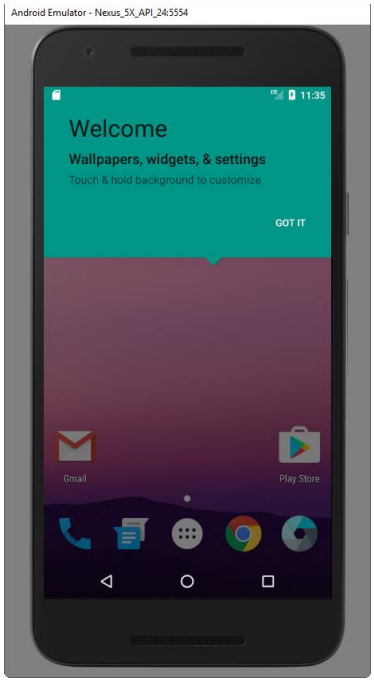
In the *Actions* column on the right, press the *Play* icon.



This will launch the emulator. It may take a few seconds before you see anything. Wait for your virtual device to load. If you have multiple screens or several applications open the view screen may initially be hidden.

Occasionally the device may fail to load properly at this step or the next step. If this happens, the first thing to try is completely closing the virtual device and then loading it again.

Once your emulator has launched, you should end up with a screen similar to the following:



You now have a working emulated Android device. Your mouse works as a finger for touch input. Unlock the device by dragging the lock icon. Note that because it is emulated the device will be much slower than a real Android device would be. You may have to be patient with it!

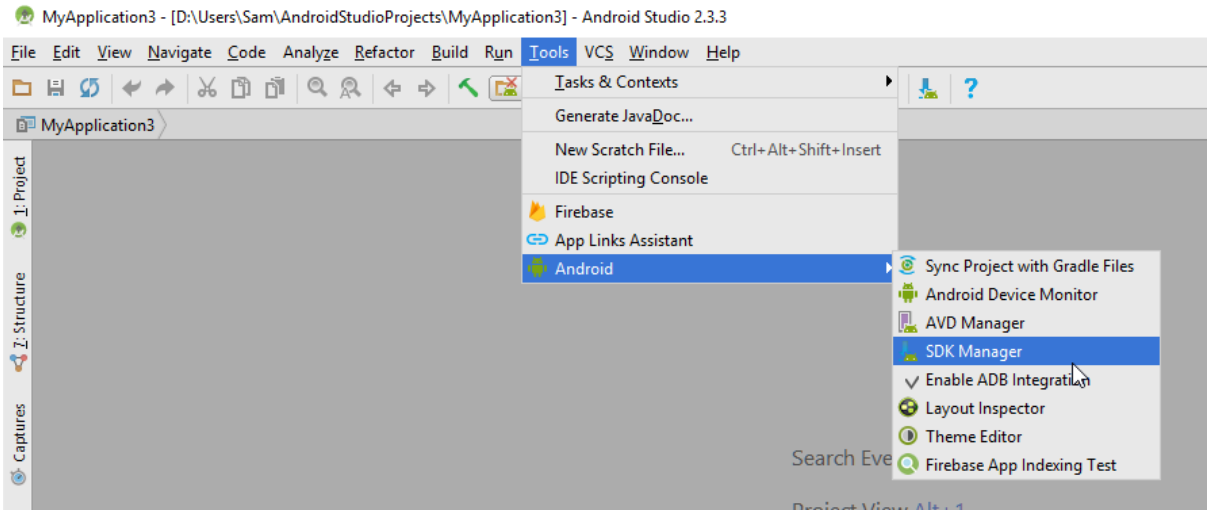
Deploy a Unity Game to your Virtual Device:

Now that we've got an Android Virtual Device for testing, lets deploy a game to it.

1. Open one of your game projects in Unity. For our purposes, it doesn't matter which project.
2. In *Unity*, we'll need to set the Java Development Kit and Android SDK Paths (which we noted back in Activity 1). If you don't remember what these paths are, we can access them from the *Android Studio* (see below).
    a. In *Unity*, Go to "Edit" -> "Preferences" -> "External Tools".
    b. We need to enter the *Android SDK* path.

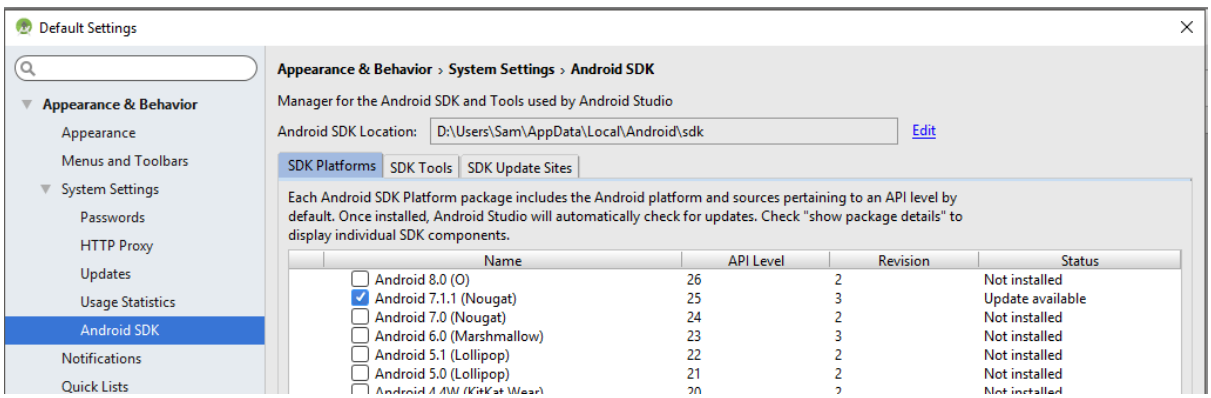
    When we created an *Android Virtual Device*, Android Studio downloaded an SDK for us. We will need to use Android Studio to obtain the path to this SDK.

    If it isn't open already, open *Android Studio* and from the *Tools* menu select *SDK Manager*.



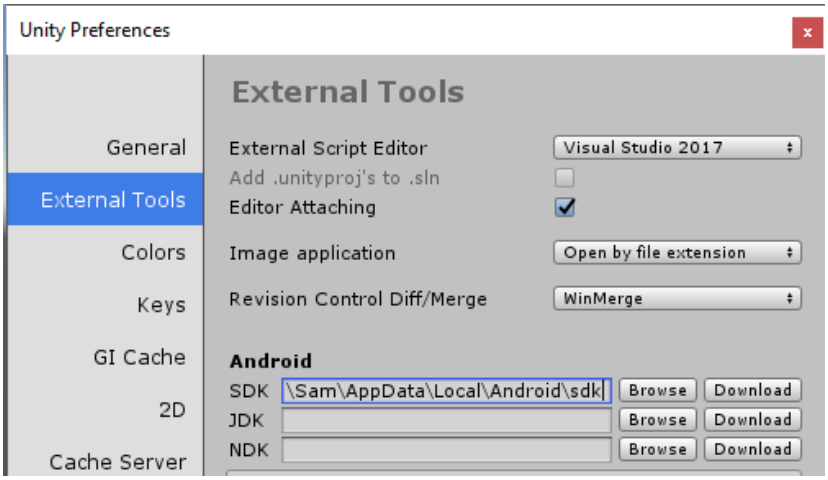
The *Settings* window will open. It should open to the *Android SDK* settings.

The path we need is in the *Android SDK Location* field:

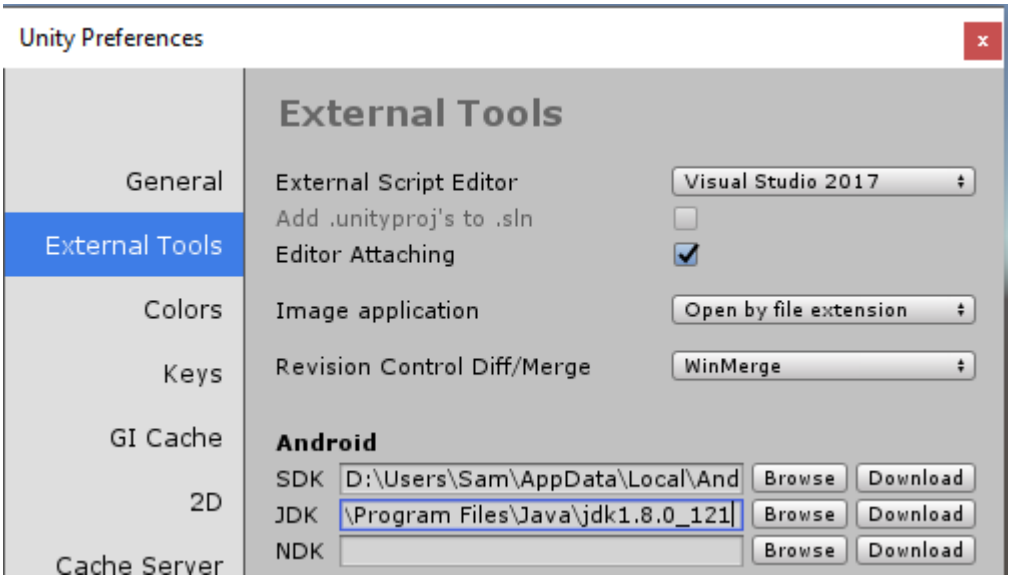


Copy this path and return to *Unity*.

c. Insert the *Android SDK Location* copied from *Android Studio* into the *SDK* filed in the *Unity* settings.



d. Find the location where you installed the *Java JDK,* and enter that path in the *JDK* field.

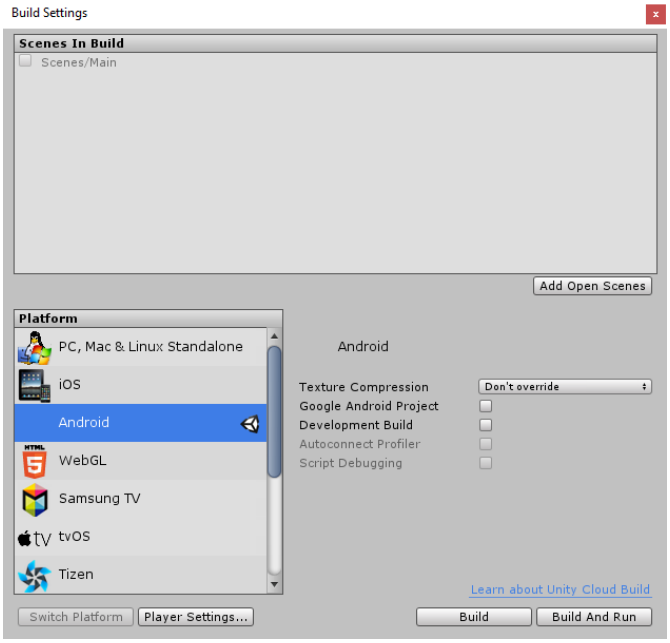


We do not need to provide an NDK path.

e. Close the preferences window.

3. Go to *File -> Build Settings...*

4. Under *Platforms* select *Android* and then press *Switch Platform*. Depending on how many assets are in your game this may take some time.

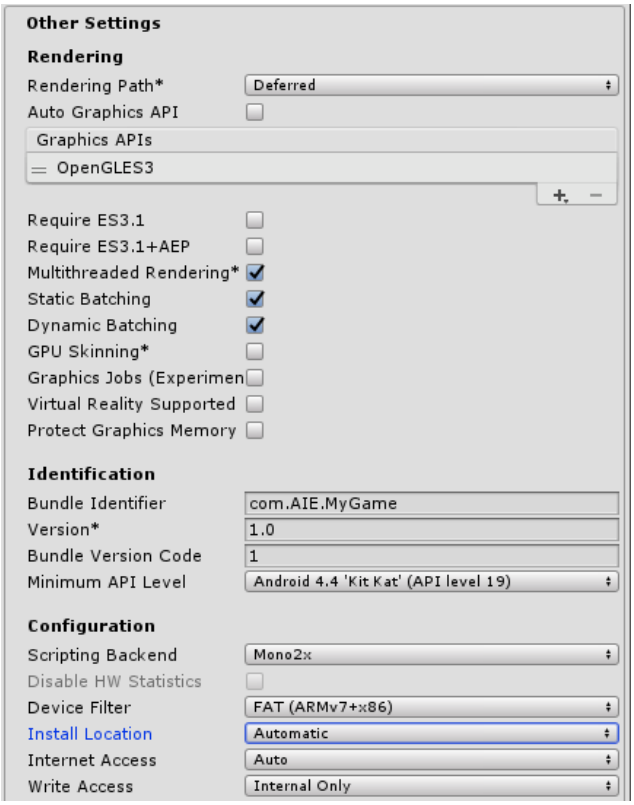


5. When this is finished click "*Player Settings...*". This will appear in the Inspector tab.

6. In the Player Settings Inspector, ensure that the Android tab is selected, and then open the *Other Settings* section and set the following:

   Bundle Identifier: com.AIE.MyGame

   Install Location: Automatic



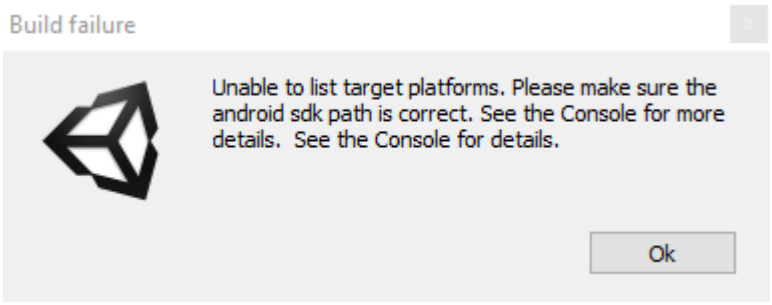
7. If it is not already running, start your Android Virtual Device using the Android Studio. (From the *Tools* menu select *AVD Manager,* then press the *Play* button for your virtual device).

8. Back in *Unity*, in the Build Settings, click *Build and Run*.

    a. When prompted, select a location to save the APK file which will be generated.
    b. Then, wait for the game to appear on your Android Virtual Device.

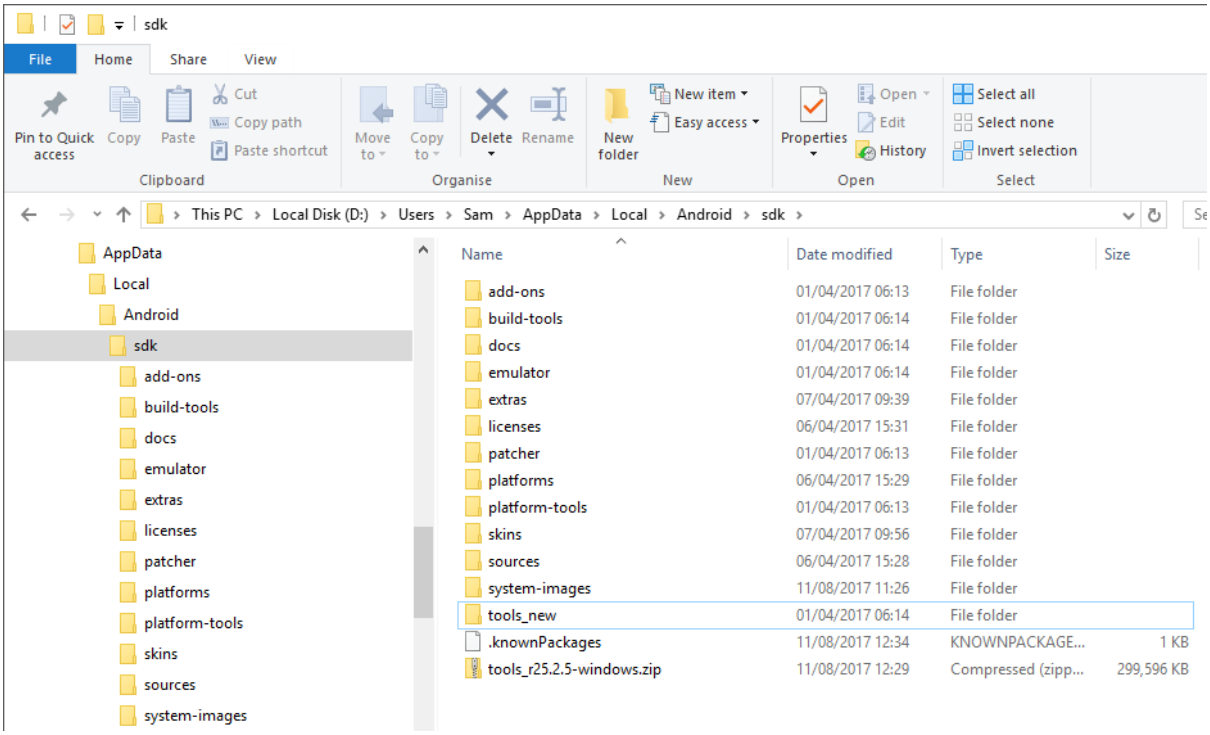
Troubleshooting:

It's possible that the build process may fail with the following error if your version of *Unity 3D* is incompatible with the latest *Android SDK*.

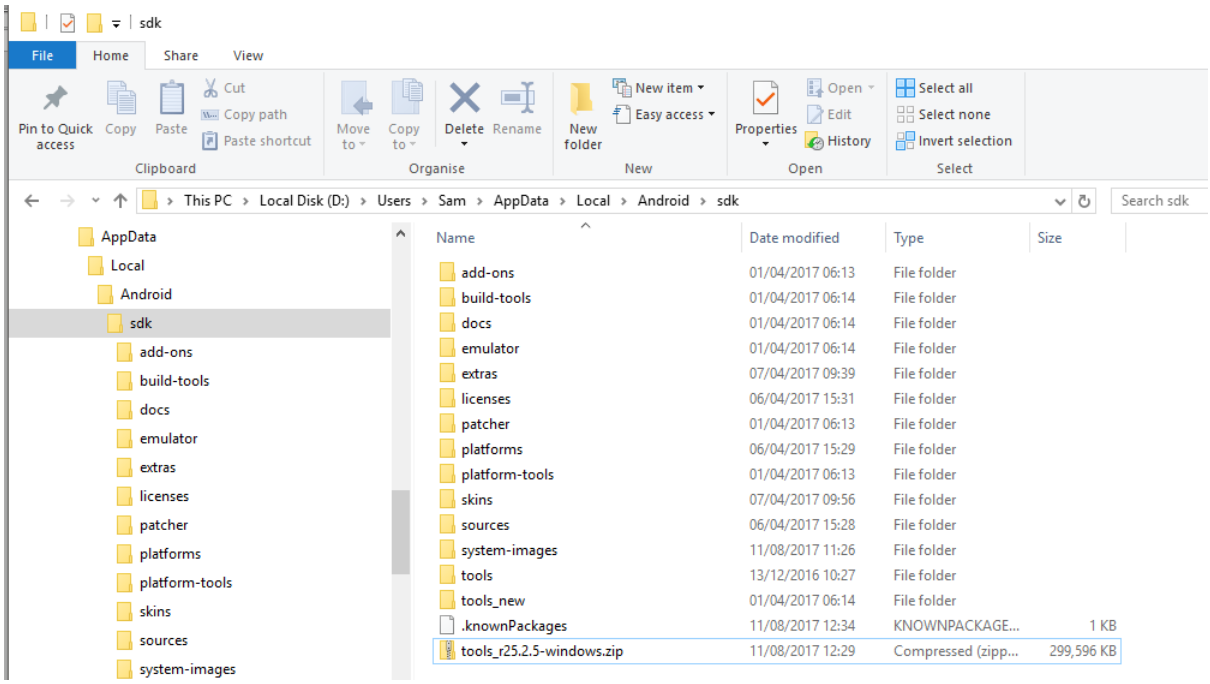


In this case you will need to download an old version of the *Android SDK* to use with *Unity*.

1. Download this version of the *SDK* from *Google*: http://dl-ssl.google.com/android/repository/tools_r25.2.5-windows.zip
2. Open the folder where *Android Studio* installed the *Android SDK*.
   (You can find this path by opening the *Android SDK Manager* from within *Android Studio*, as described previously.)
3. Rename the *tools* folder to "*tools_new*"

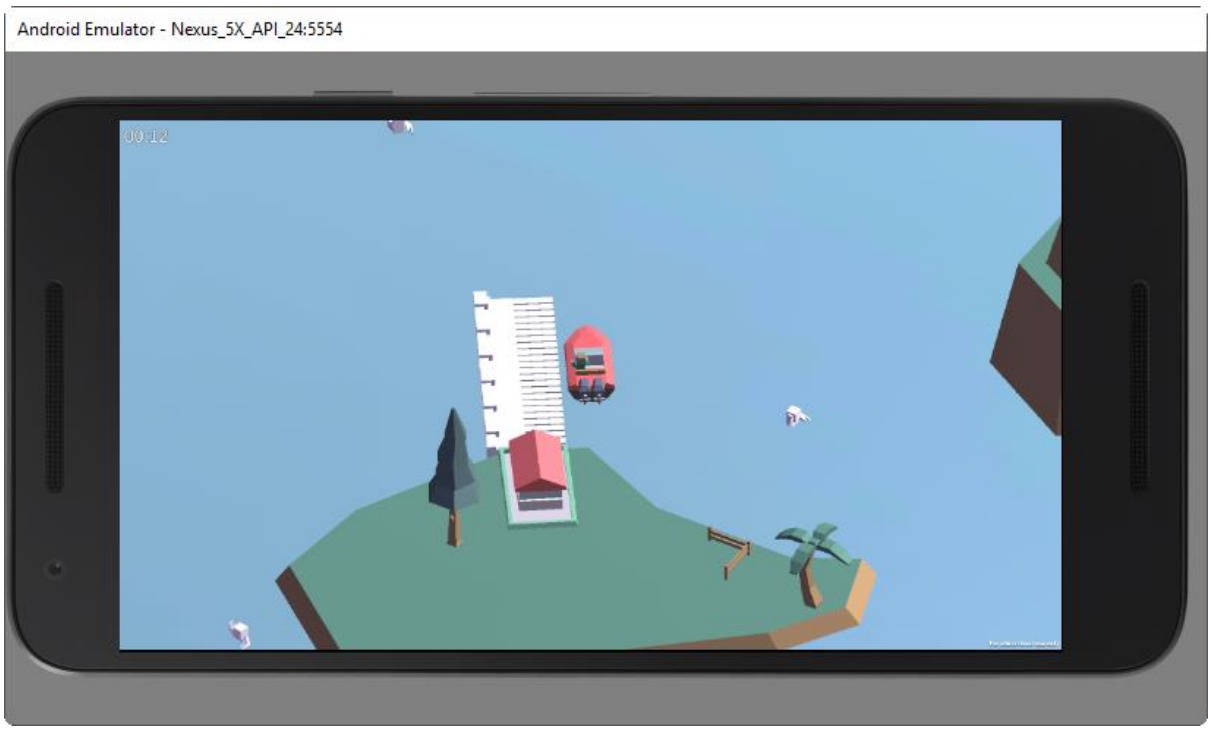


4. Extract the downloaded .zip file, ensuring that the *tools* folder is extracted to this *sdk* folder. You should now have a *tools* folder, and a *tools_new* folder.



5. Return to *Unity* and click the *Build and Run* button again.

At the end of this tutorial you should have your project running on a virtual *Android* device.



(If you have an Android phone connected to your computer via USB, it is best to disconnect it before pressing *Build and Run*, or else Unity may try to install the game to your phone instead.)