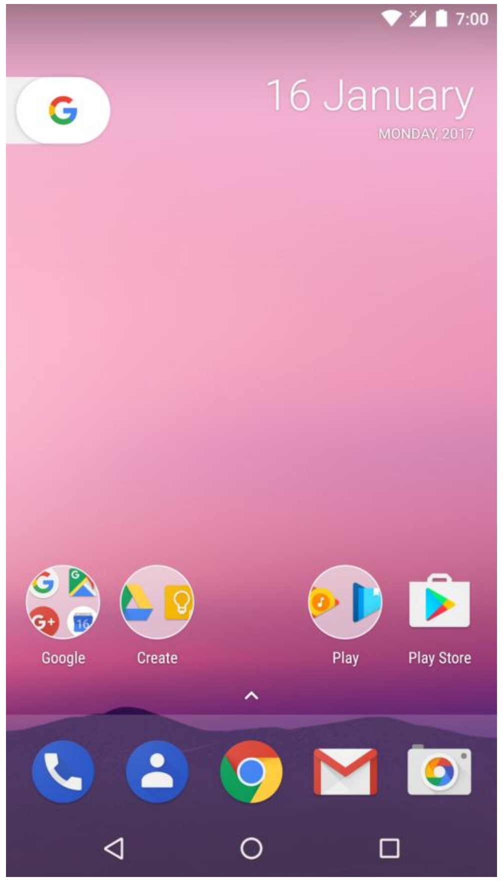
Advanced Programming 2
Recitation 12 – Android Part I

Roi Yehoshua
2017

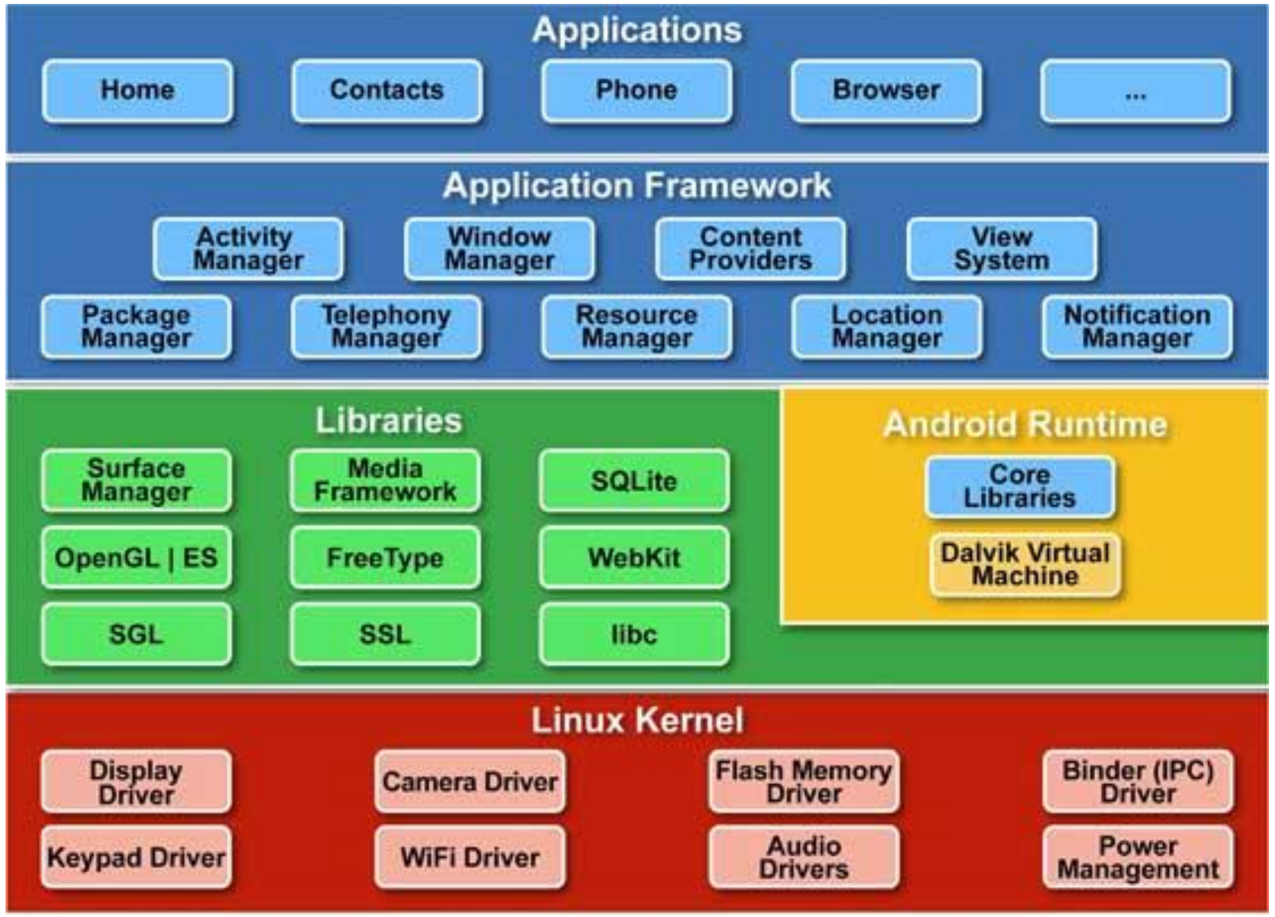
Android

Android

- Android is a mobile operating system developed by Google
- Based on the Linux kernel
- Created as an open-source project by Android Inc in 2003 and purchased by Google in 2005
- Dominates the smartphone market with a share of 86.8%
- The official site for Android developers
  - http://developer.android.com
  - Contains documentation and tutorials



Android Architecture



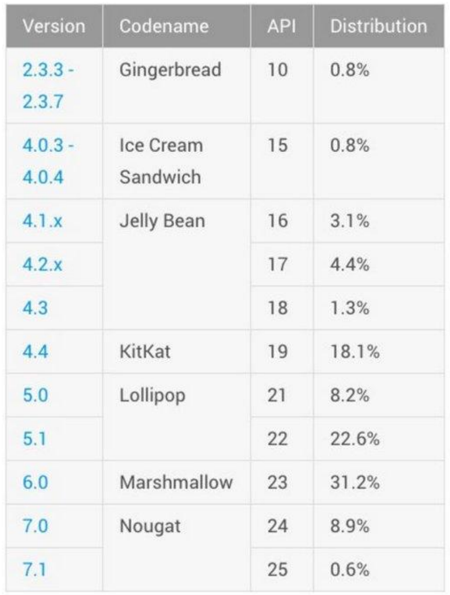
Android SDK

- The **Android Software Development Kit (SDK)** provides the tools and APIs necessary to develop Android applications
- The Android SDK supports most of the Java platform Standard Edition
  - except for the AWT and Swing libraries
- The Android SDK includes a comprehensive set of development tools, which include a debugger, libraries, a handset emulator , documentation, sample code, and tutorials

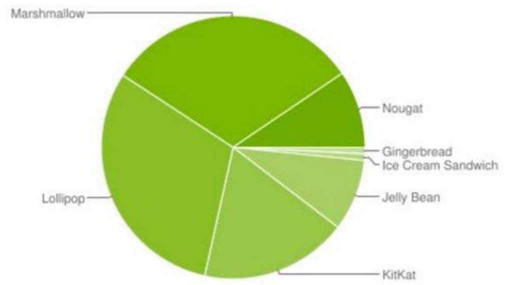
Android API Level

- Each major release of Android version is named after something sweet
- **API Level** is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform
  - It lets applications describe the framework API revision that they require
  - You can have a new Android version with the same API release as the previous version

<u>Android version distribution as of June 2017</u>

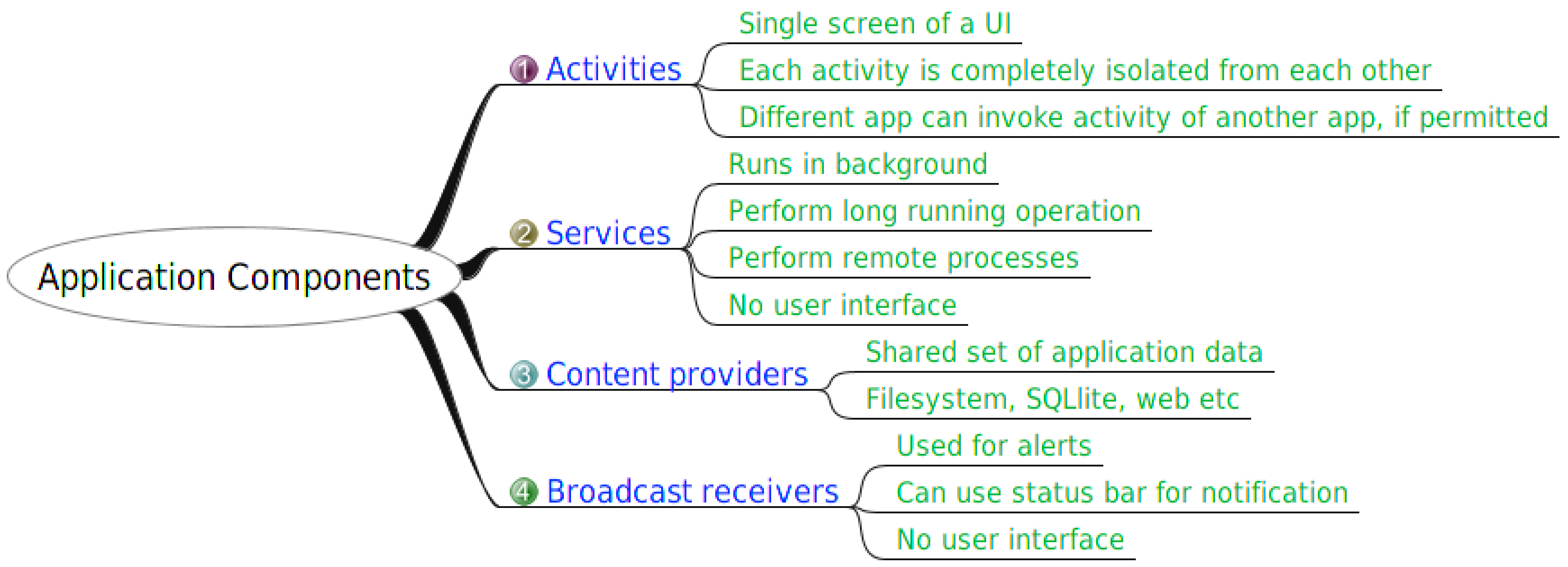


Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.1%
4.2.x		17	4.4%
4.3		18	1.3%
4.4	KitKat	19	18.1%
5.0	Lollipop	21	8.2%
5.1		22	22.6%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	8.9%
7.1		25	0.6%



Android Application Components

- App components are the essential building blocks of an Android app
- Each component is an entry point through which the system can enter your app
- There are four different types of app components:

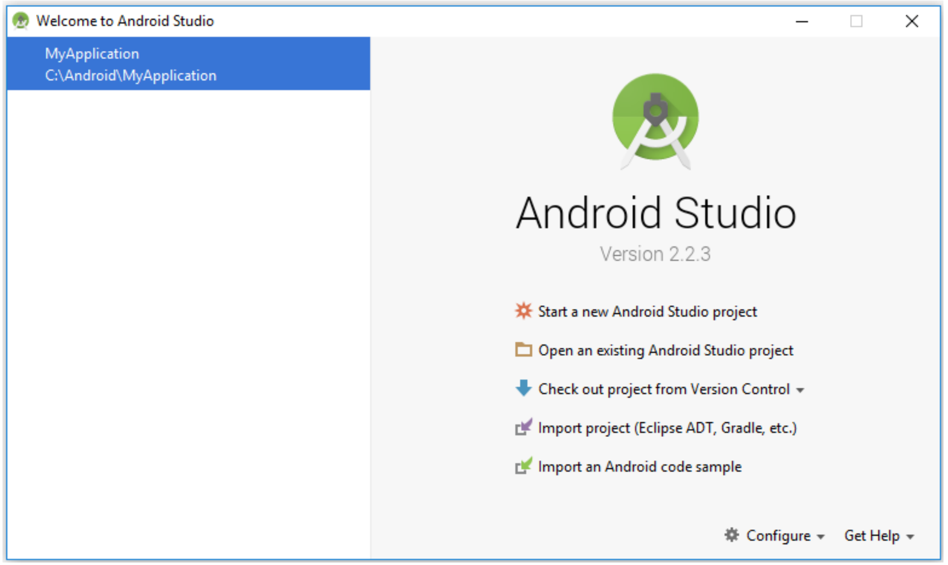


Application Components
1 Activities
Single screen of a UI
Each activity is completely isolated from each other
Different app can invoke activity of another app, if permitted
2 Services
Runs in background
Perform long running operation
Perform remote processes
No user interface
3 Content providers
Shared set of application data
Filesystem, SQLlite, web etc
4 Broadcast receivers
Used for alerts
Can use status bar for notification
No user interface

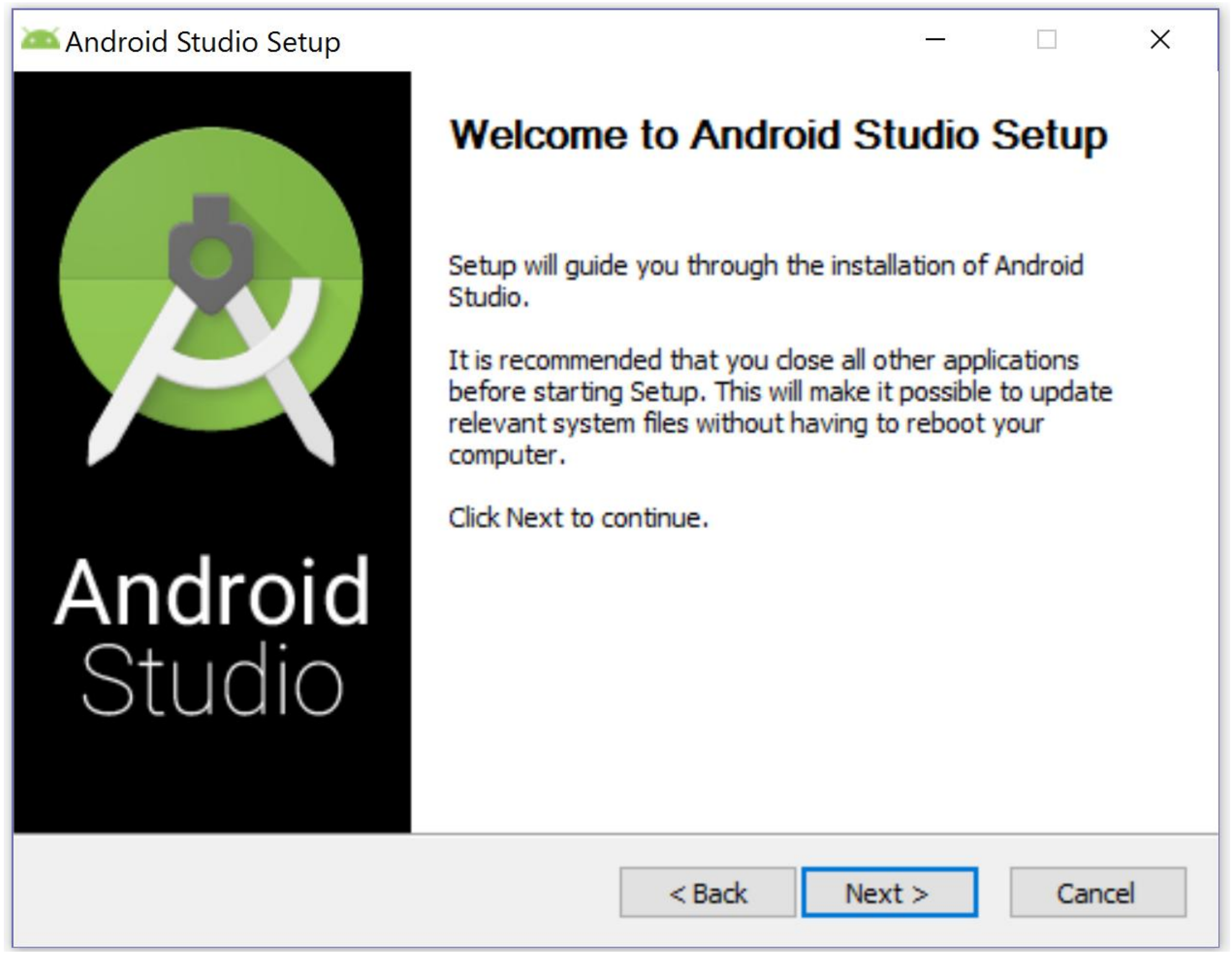
Android Studio

Android Studio

- The Official IDE for Android
- Supports code editing, debugging, fast and rich emulator, performance tooling, flexible build system, and GitHub integration
- Download https://developer.android.com/studio/index.html#downloads
  - There are versions for Windows, Mac and Linux


Welcome to Android Studio
MyApplication
C:\Android\MyApplication
Android Studio
Version 2.2.3
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
Get Help

Android Studio Setup



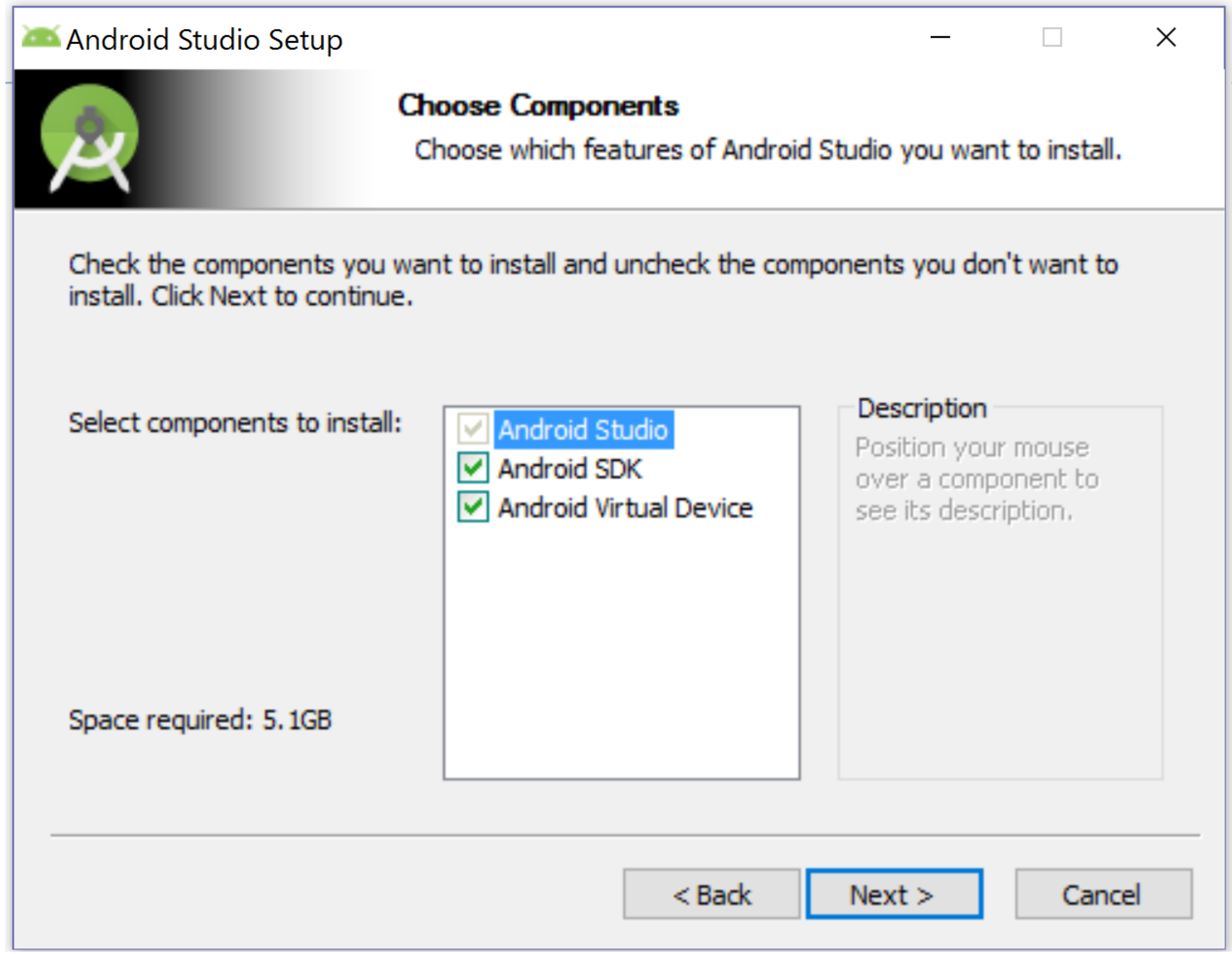
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Android Studio Setup



Android Studio Setup

Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

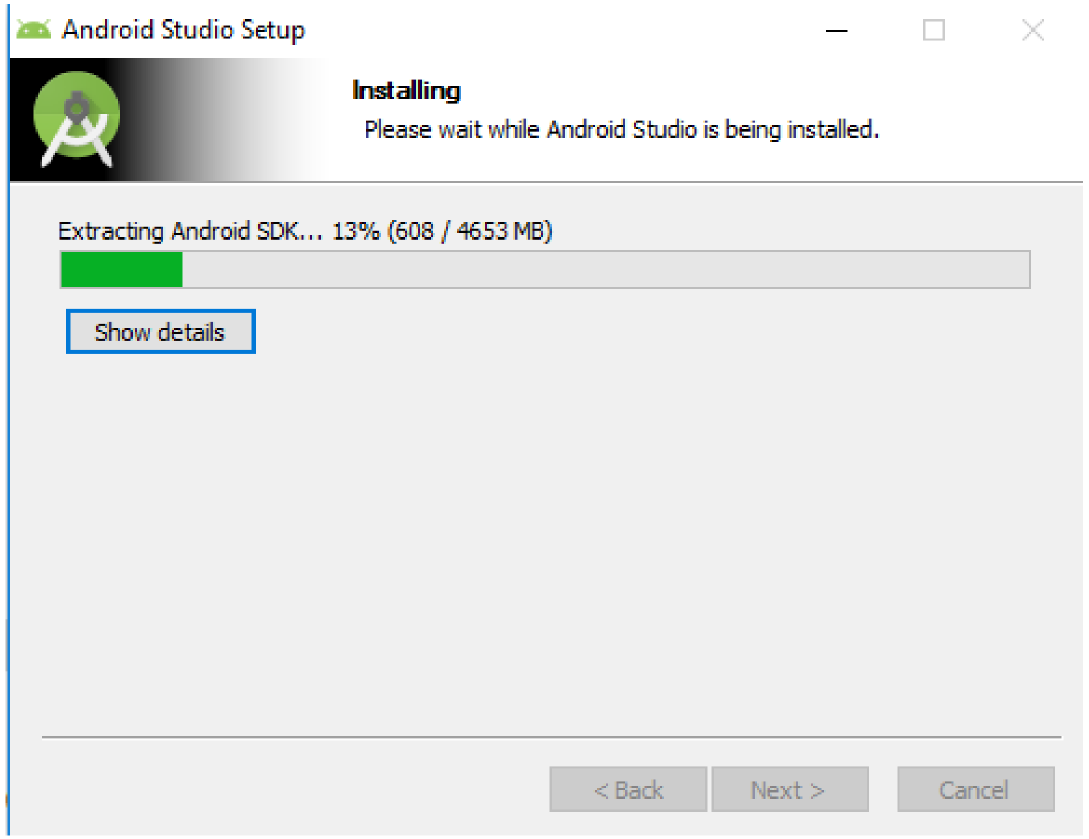
- [✓] Android Studio
- [✓] Android SDK
- [✓] Android Virtual Device

Description

Position your mouse over a component to see its description.

Space required: 5.1GB

Android Studio Setup

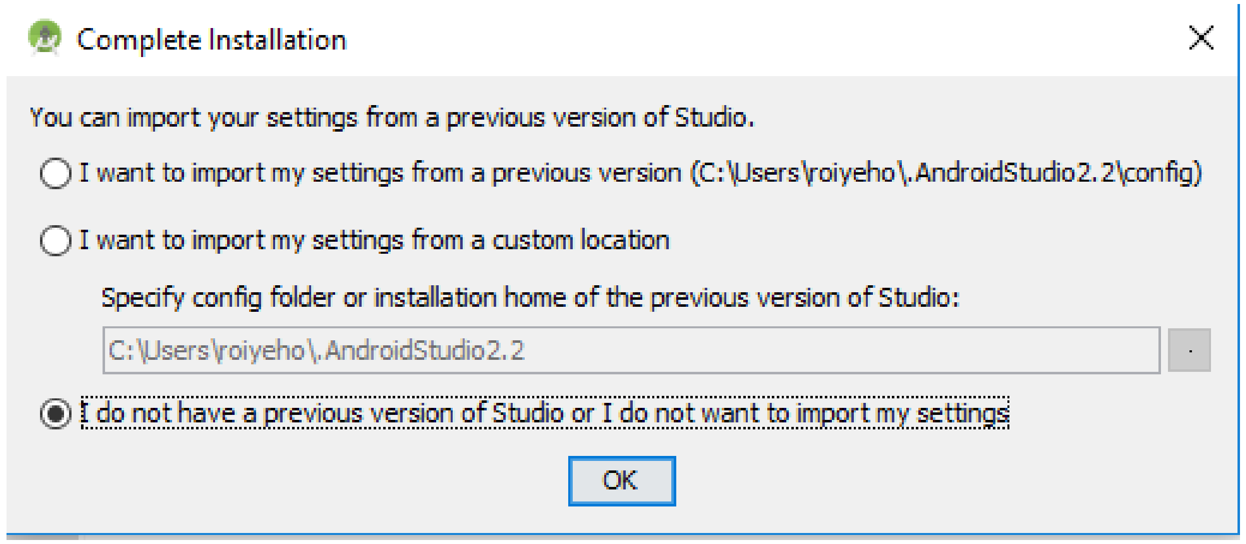

Android Studio Setup

Installing

Please wait while Android Studio is being installed.

Extracting Android SDK... 13% (608 / 4653 MB)

Android Studio Setup



Complete Installation

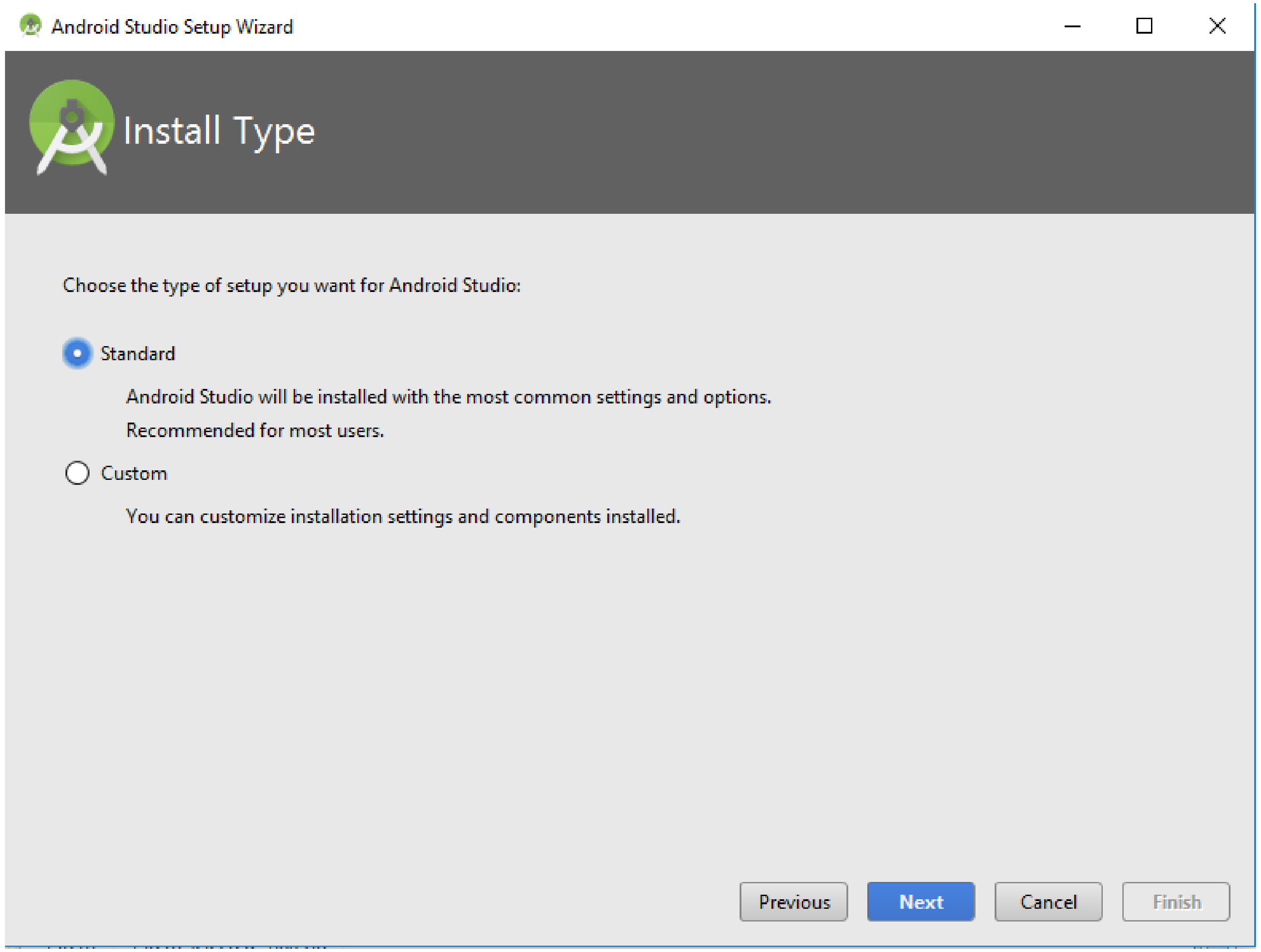
You can import your settings from a previous version of Studio.

- ○ I want to import my settings from a previous version (C:\Users\roiyeho\.AndroidStudio2.2\config)
- ○ I want to import my settings from a custom location

  Specify config folder or installation home of the previous version of Studio:

  C:\Users\roiyeho\.AndroidStudio2.2
- ● I do not have a previous version of Studio or I do not want to import my settings

Android Studio Setup


Android Studio Setup Wizard

Install Type

Choose the type of setup you want for Android Studio:

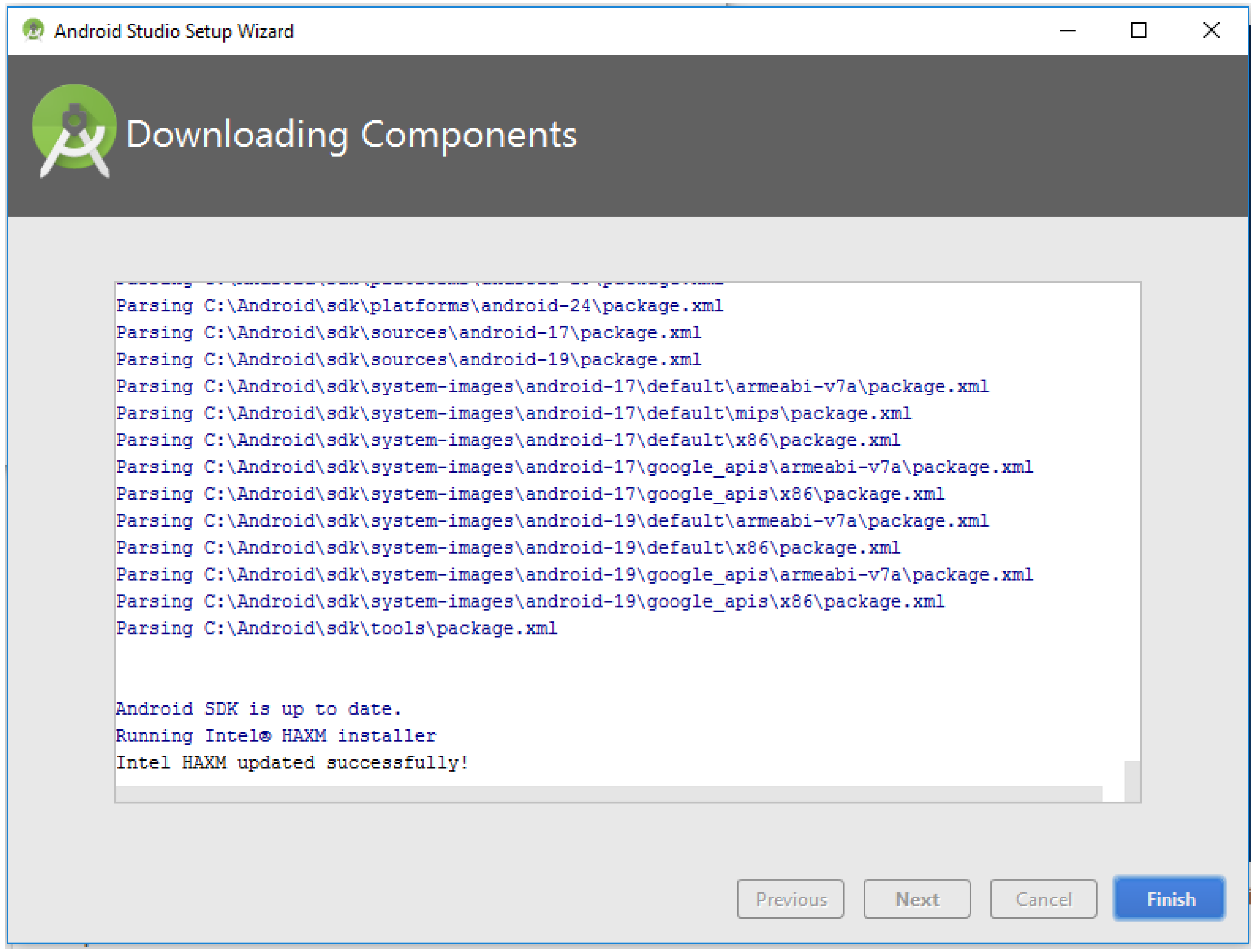
[✓ Standard]

Android Studio will be installed with the most common settings and options.
Recommended for most users.

Custom

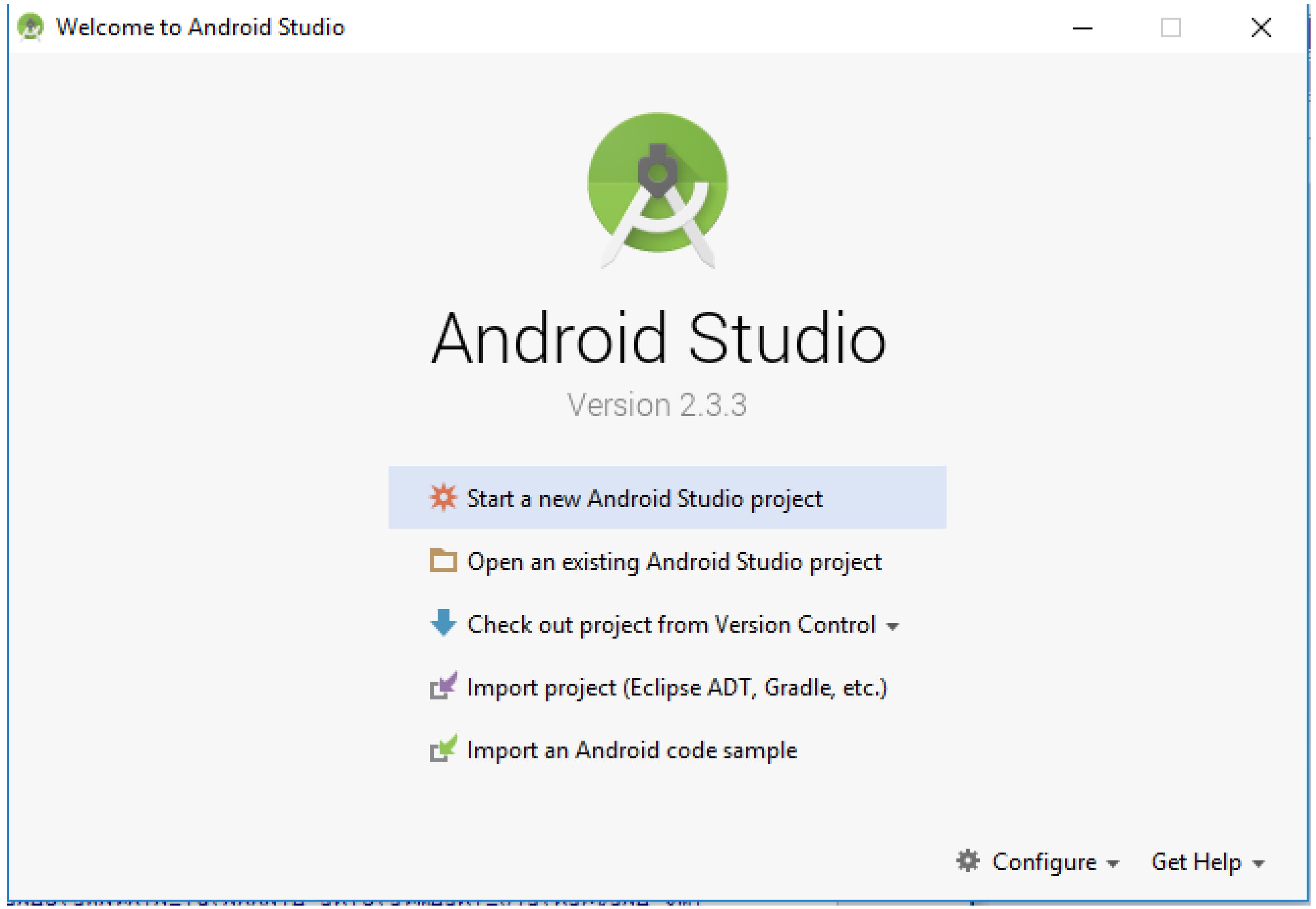
You can customize installation settings and components installed.

Android Studio Setup

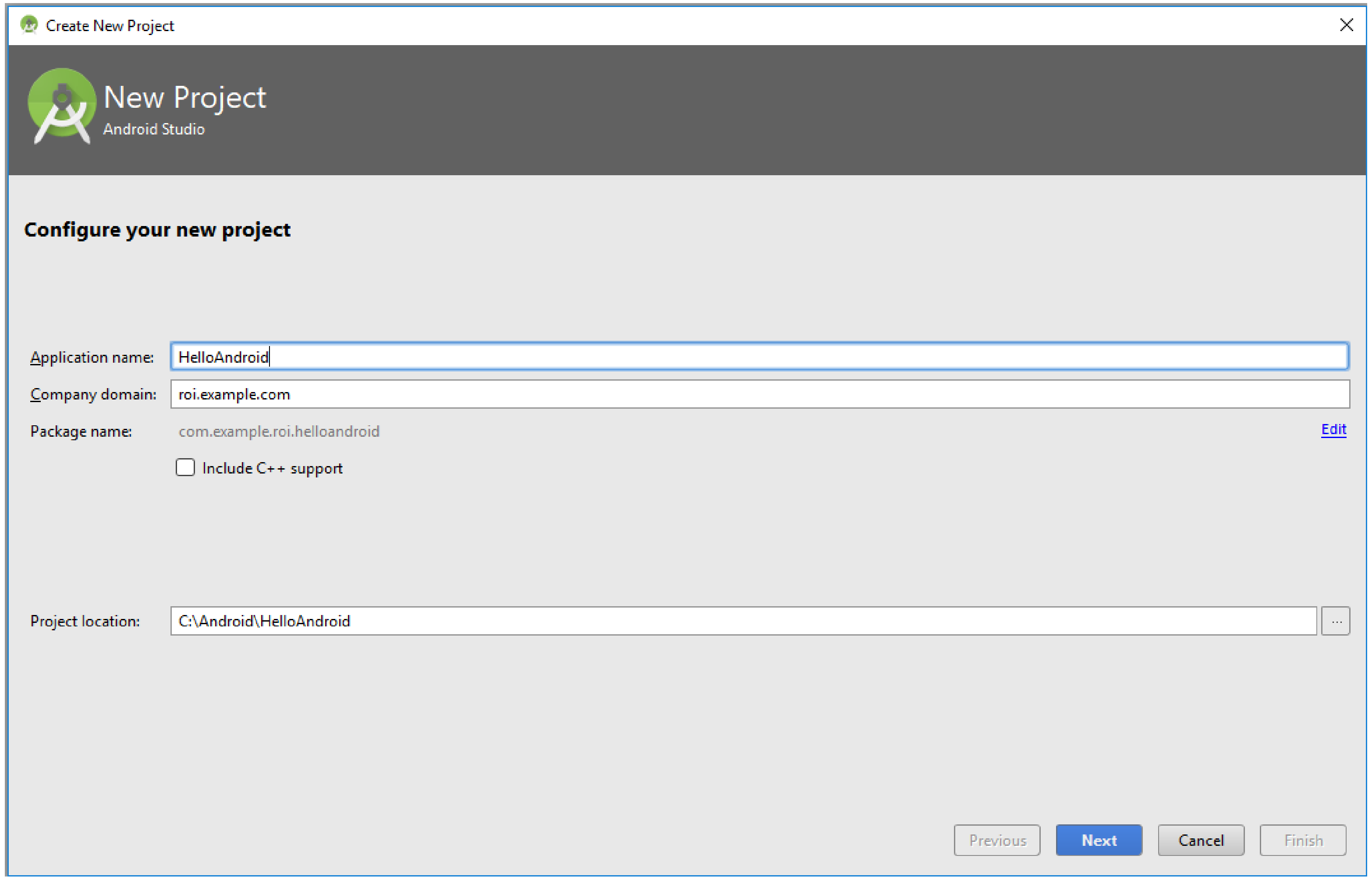


```
Parsing C:\Android\sdk\platforms\android-24\package.xml
Parsing C:\Android\sdk\sources\android-17\package.xml
Parsing C:\Android\sdk\sources\android-19\package.xml
Parsing C:\Android\sdk\system-images\android-17\default\armeabi-v7a\package.xml
Parsing C:\Android\sdk\system-images\android-17\default\mips\package.xml
Parsing C:\Android\sdk\system-images\android-17\default\x86\package.xml
Parsing C:\Android\sdk\system-images\android-17\google_apis\armeabi-v7a\package.xml
Parsing C:\Android\sdk\system-images\android-17\google_apis\x86\package.xml
Parsing C:\Android\sdk\system-images\android-19\default\armeabi-v7a\package.xml
Parsing C:\Android\sdk\system-images\android-19\default\x86\package.xml
Parsing C:\Android\sdk\system-images\android-19\google_apis\armeabi-v7a\package.xml
Parsing C:\Android\sdk\system-images\android-19\google_apis\x86\package.xml
Parsing C:\Android\sdk\tools\package.xml


Android SDK is up to date.
Running Intel® HAXM installer
Intel HAXM updated successfully!
```

Launch Android Studio


Welcome to Android Studio
Android Studio
Version 2.3.3
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
Get Help

Create a New Project


Create New Project

Configure your new project

Application name: HelloAndroid

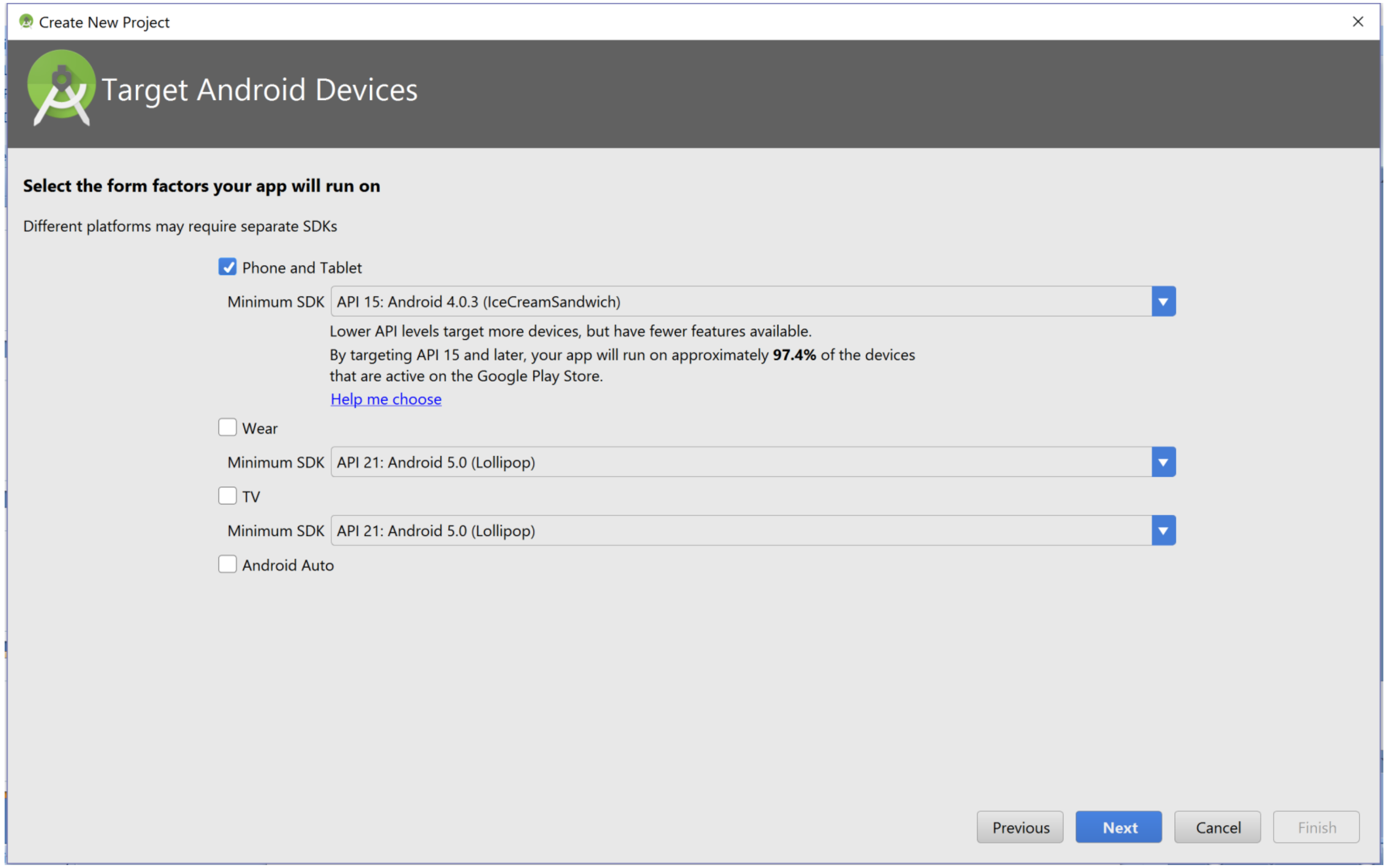
Company domain: roi.example.com

Package name: com.example.roi.helloandroid

Include C++ support [ ]

Project location: C:\Android\HelloAndroid

Create a New Project


Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.
Help me choose

[ ] Wear

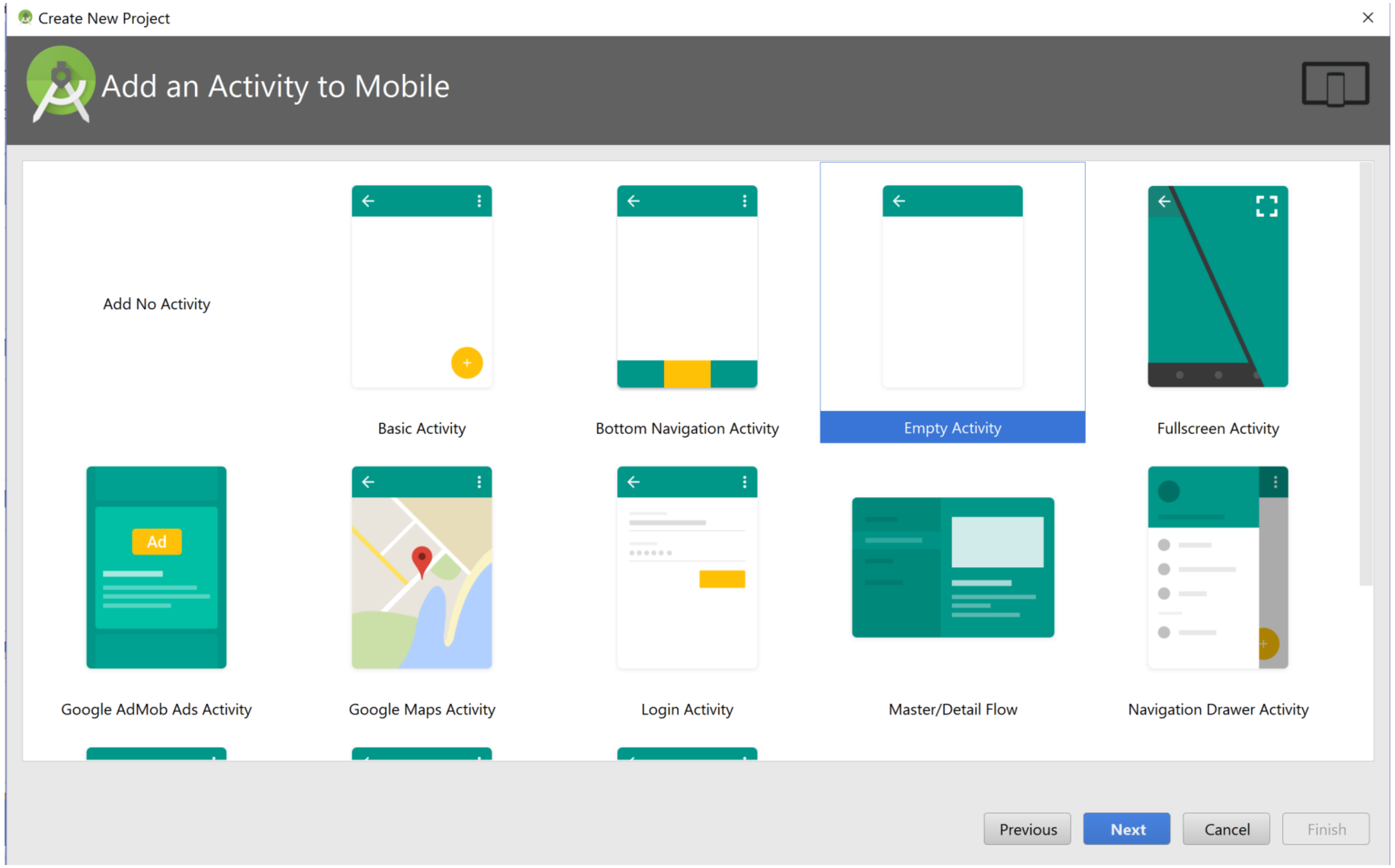
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV

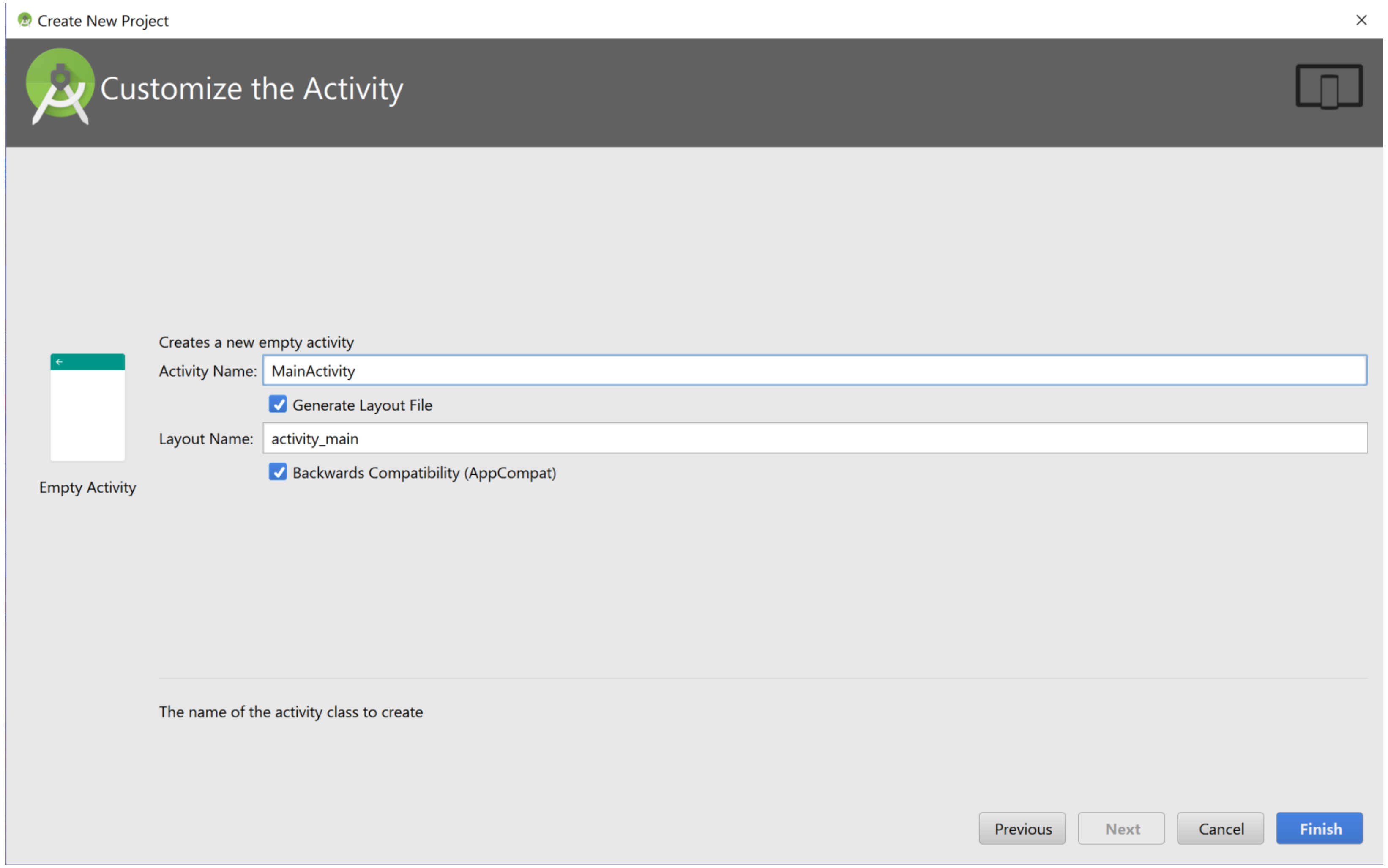
Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Android Auto

Create a New Project



Create a New Project


Create New Project

Customize the Activity

Creates a new empty activity

Activity Name: MainActivity

[✓] Generate Layout File

Layout Name: activity_main

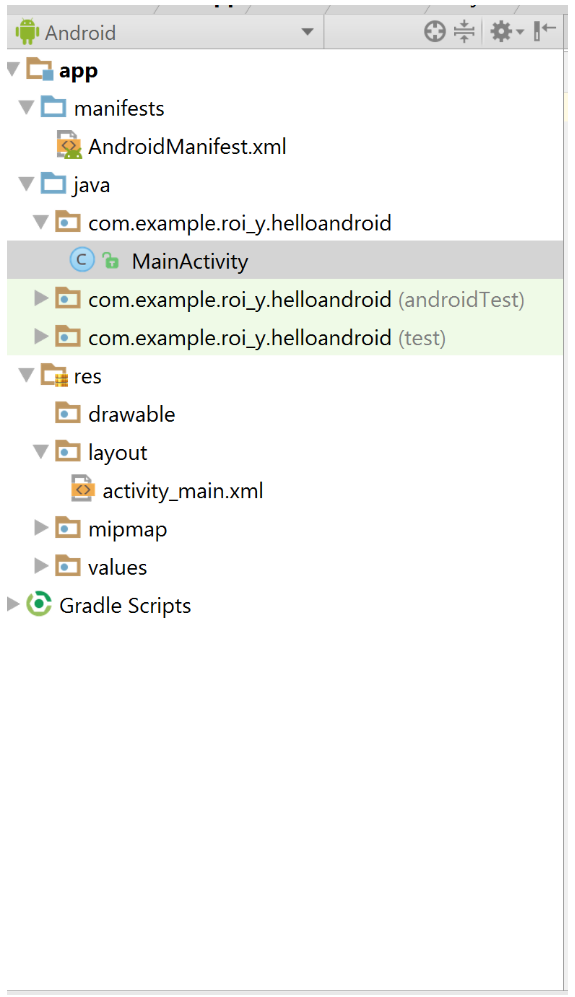
[✓] Backwards Compatibility (AppCompat)

Empty Activity

The name of the activity class to create

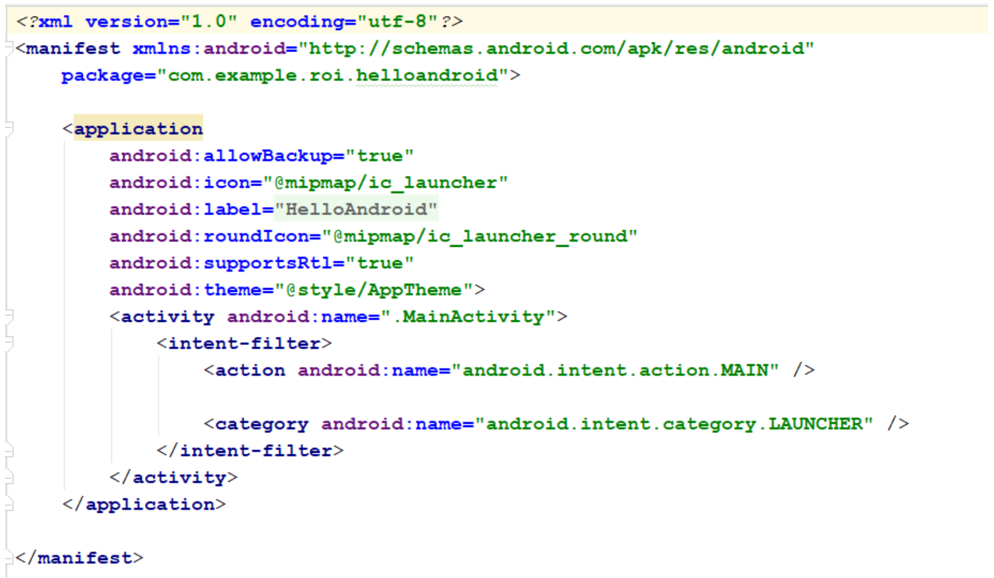
Project Structure

- Within each Android app module, files are shown in the following groups:
  - **manifests** - contains the AndroidManifest.xml file
  - **java** - contains the Java source code files, separated by package names,
  - **res** - contains all non-code resources, such as XML layouts, UI strings, and bitmap images, divided into corresponding sub-directories


Android
app
manifests
AndroidManifest.xml
java
com.example.roi_y.helloandroid
MainActivity
com.example.roi_y.helloandroid (androidTest)
com.example.roi_y.helloandroid (test)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts

App Manifest

- AndroidManifest.xml provides essential information about your app:
  - The Java package name for the application
  - The minimum level of the Android API that the application requires
  - The components of the application, including the classes that implement them and the processes that host them
  - The permissions that the application must have in order to interact with other processes
  - The permissions that others are required to have in order to interact with the application's components

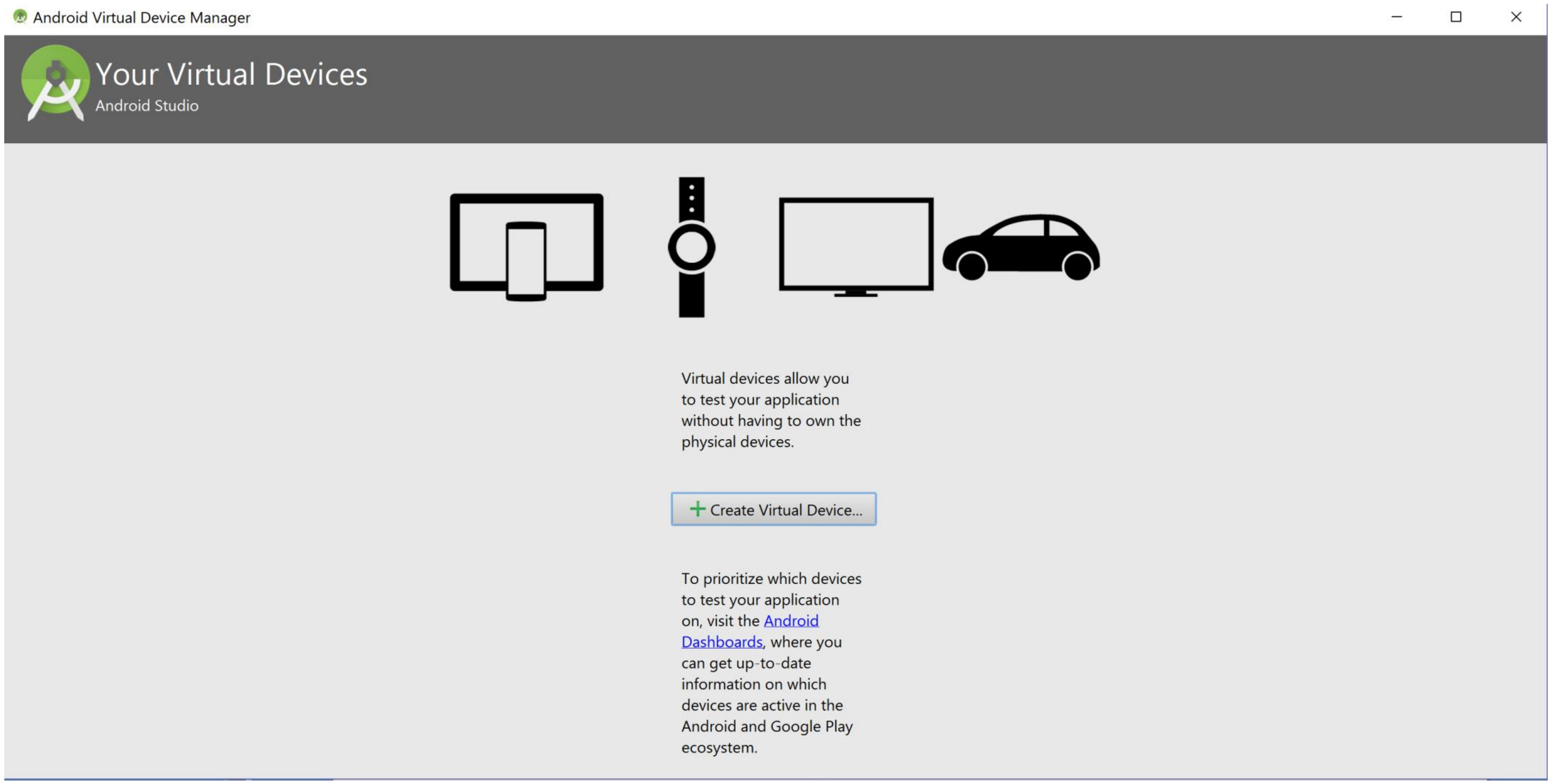


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.roi.helloandroid">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloAndroid"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Android Virtual Device (AVD)

- Before you run your app on an emulator, you need to create an Android Virtual Device (AVD)
- An AVD definition specifies the characteristics of the Android device that you want to simulate
- Create an AVD Definition as follows:
  - Launch the Android Virtual Device Manager by selecting **Tools > Android > AVD Manager**, or by clicking the AVD Manager icon in the toolbar.
  - In the **Your Virtual Devices** screen, click **Create Virtual Device**.
  - In the **Select Hardware** screen, select a phone device, such as Pixel, and then click **Next**.
  - In the **System Image** screen, click **Download** for one of the recommended system images. Agree to the terms to complete the download.
  - After the download is complete, select the system image from the list and click **Next**.
  - On the next screen, leave all the configuration settings as they are and click **Finish**.
  - Back in the **Your Virtual Devices** screen, select the device you just created and click **Launch this AVD in the emulator**.

Android Virtual Device (AVD)


Android Virtual Device Manager

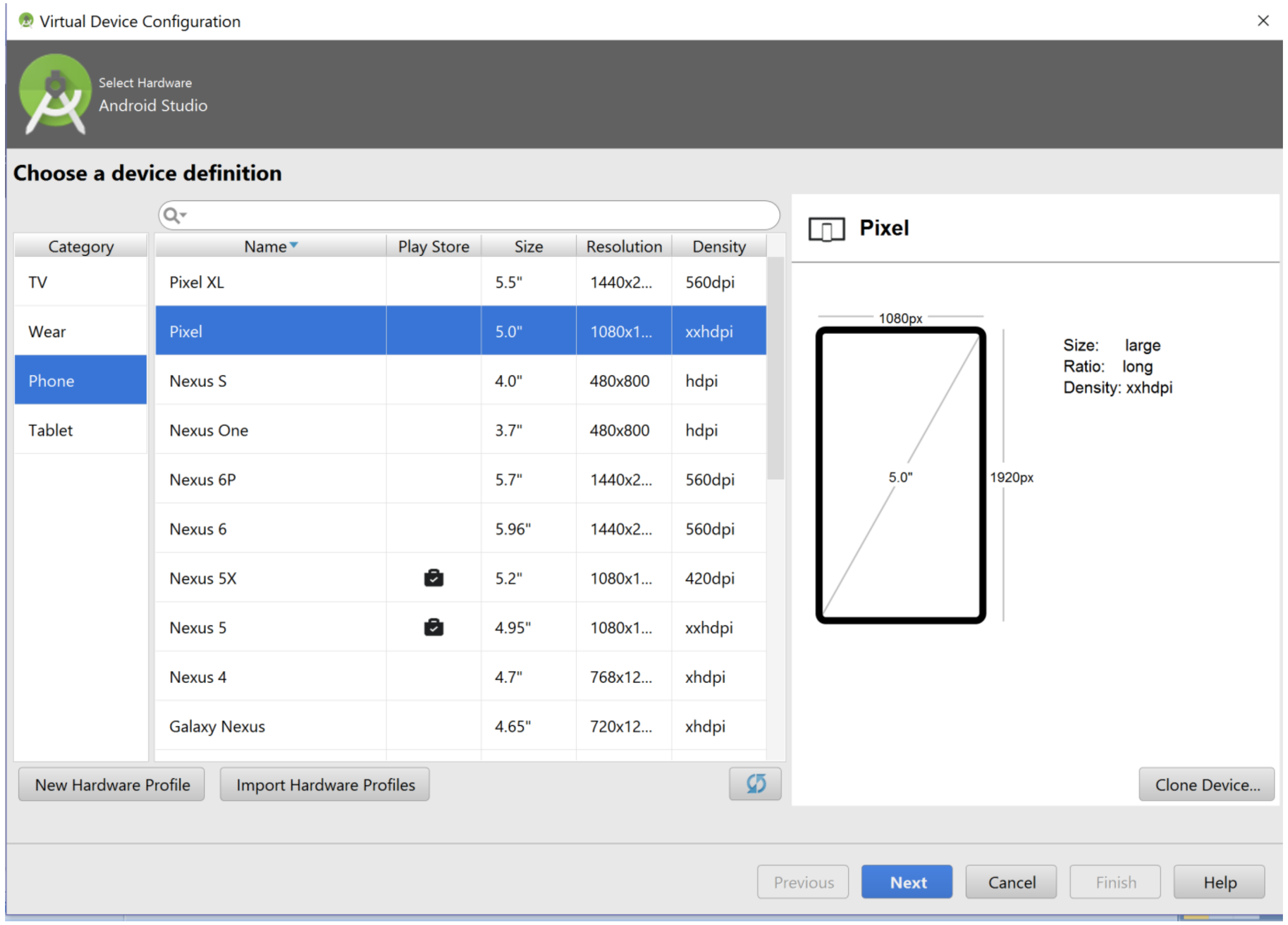
Your Virtual Devices

Android Studio

Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

Android Virtual Device (AVD)



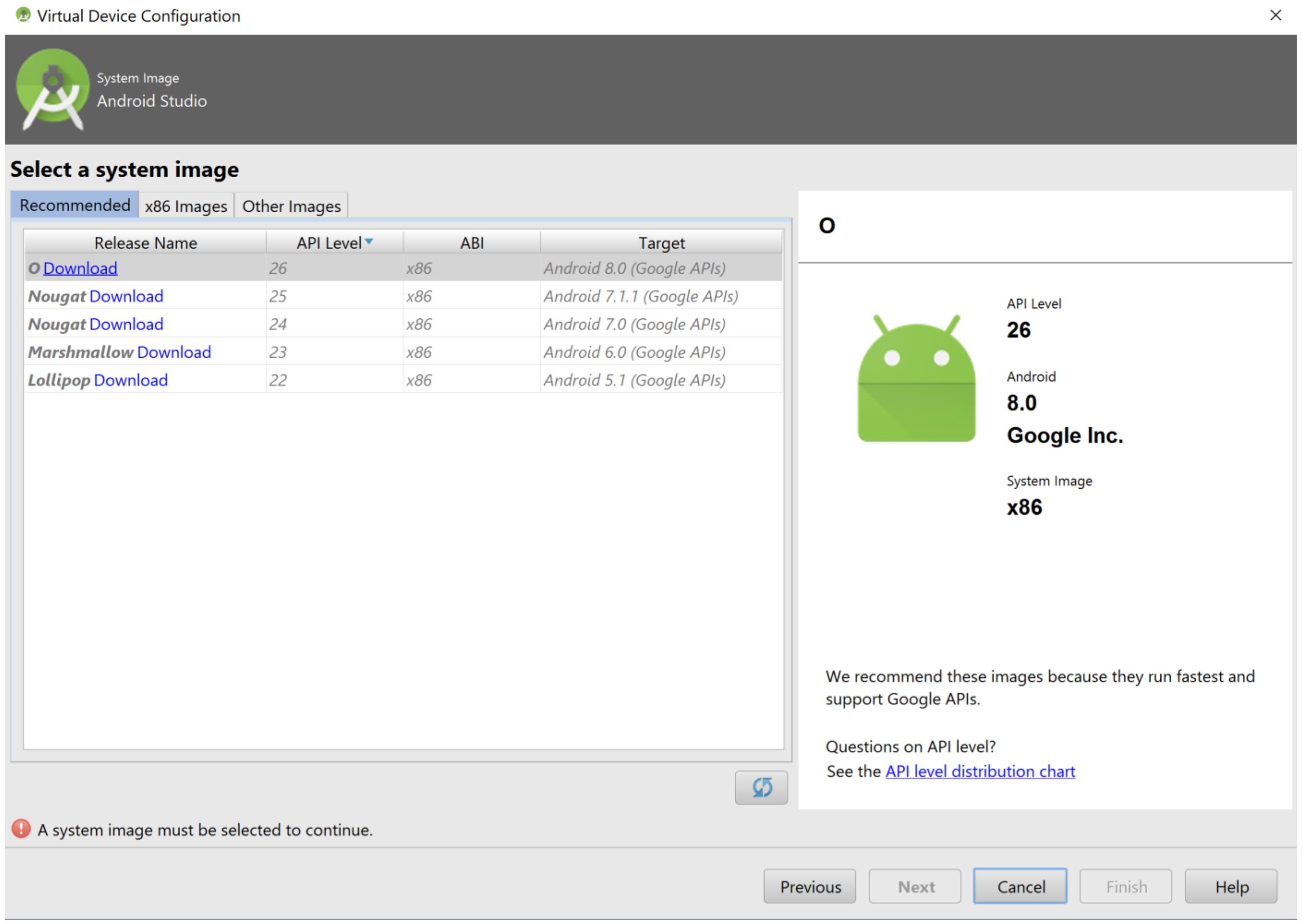
Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Pixel XL		5.5"	1440x2...	560dpi
Wear	Pixel		5.0"	1080x1...	xxhdpi
Phone	Nexus S		4.0"	480x800	hdpi
Tablet	Nexus One		3.7"	480x800	hdpi
	Nexus 6P		5.7"	1440x2...	560dpi
	Nexus 6		5.96"	1440x2...	560dpi
	Nexus 5X	●	5.2"	1080x1...	420dpi
	Nexus 5	●	4.95"	1080x1...	xxhdpi
	Nexus 4		4.7"	768x12...	xhdpi
	Galaxy Nexus		4.65"	720x12...	xhdpi

Pixel

Size: large
Ratio: long
Density: xxhdpi

Android Virtual Device (AVD)


Virtual Device Configuration

System Image
Android Studio

Select a system image

Release Name	API Level	ABI	Target
O Download	*26*	*x86*	*Android 8.0 (Google APIs)*
Nougat Download	*25*	*x86*	*Android 7.1.1 (Google APIs)*
Nougat Download	*24*	*x86*	*Android 7.0 (Google APIs)*
Marshmallow Download	*23*	*x86*	*Android 6.0 (Google APIs)*
Lollipop Download	*22*	*x86*	*Android 5.1 (Google APIs)*

O

API Level
26

Android
8.0
Google Inc.

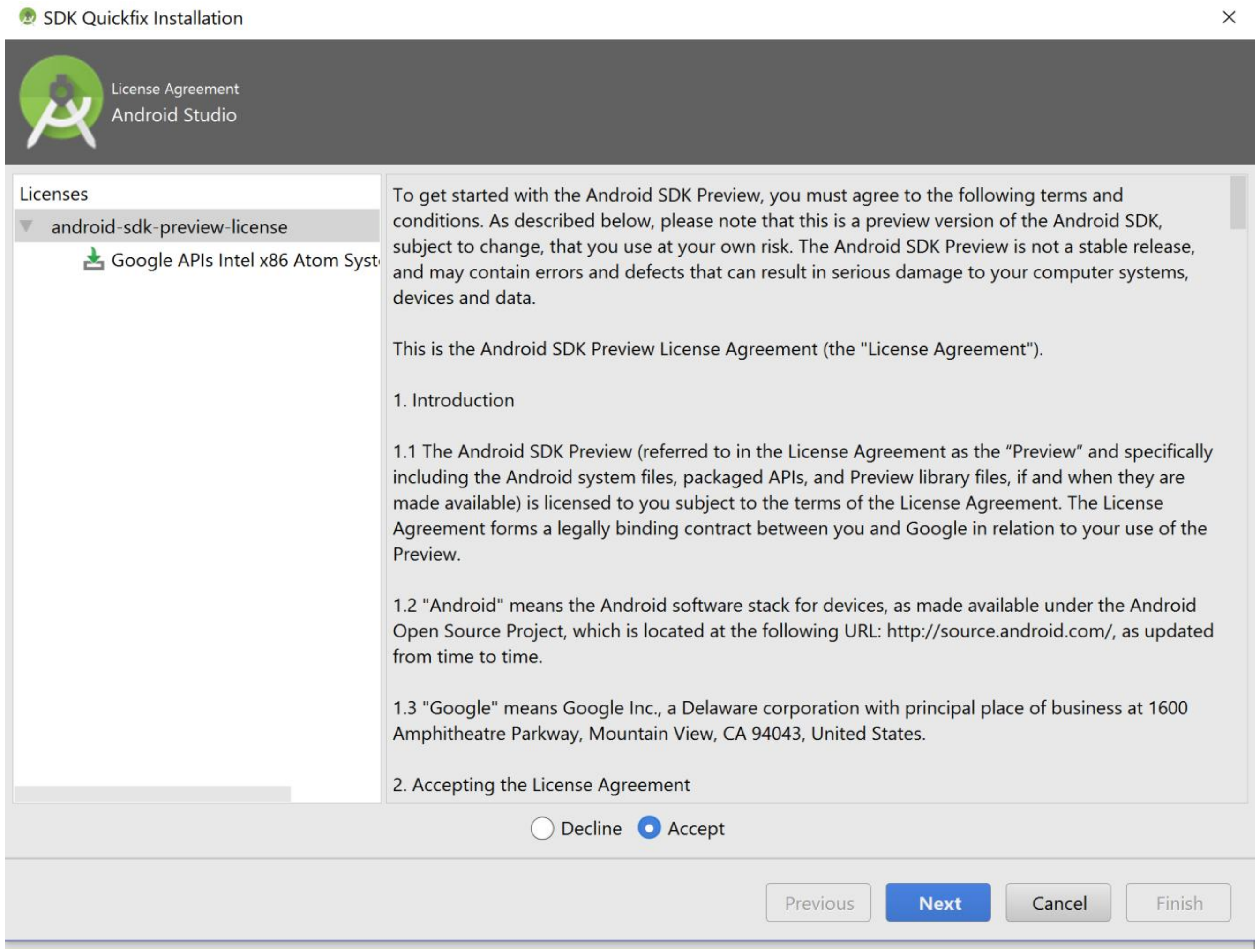
System Image
x86

We recommend these images because they run fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

A system image must be selected to continue.

Android Virtual Device (AVD)


SDK Quickfix Installation
License Agreement
Android Studio
Licenses
android-sdk-preview-license
Google APIs Intel x86 Atom Syst
Decline
Accept
Previous
Next
Cancel
Finish

To get started with the Android SDK Preview, you must agree to the following terms and conditions. As described below, please note that this is a preview version of the Android SDK, subject to change, that you use at your own risk. The Android SDK Preview is not a stable release, and may contain errors and defects that can result in serious damage to your computer systems, devices and data.

This is the Android SDK Preview License Agreement (the "License Agreement").

1. Introduction

1.1 The Android SDK Preview (referred to in the License Agreement as the “Preview” and specifically including the Android system files, packaged APIs, and Preview library files, if and when they are made available) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the Preview.

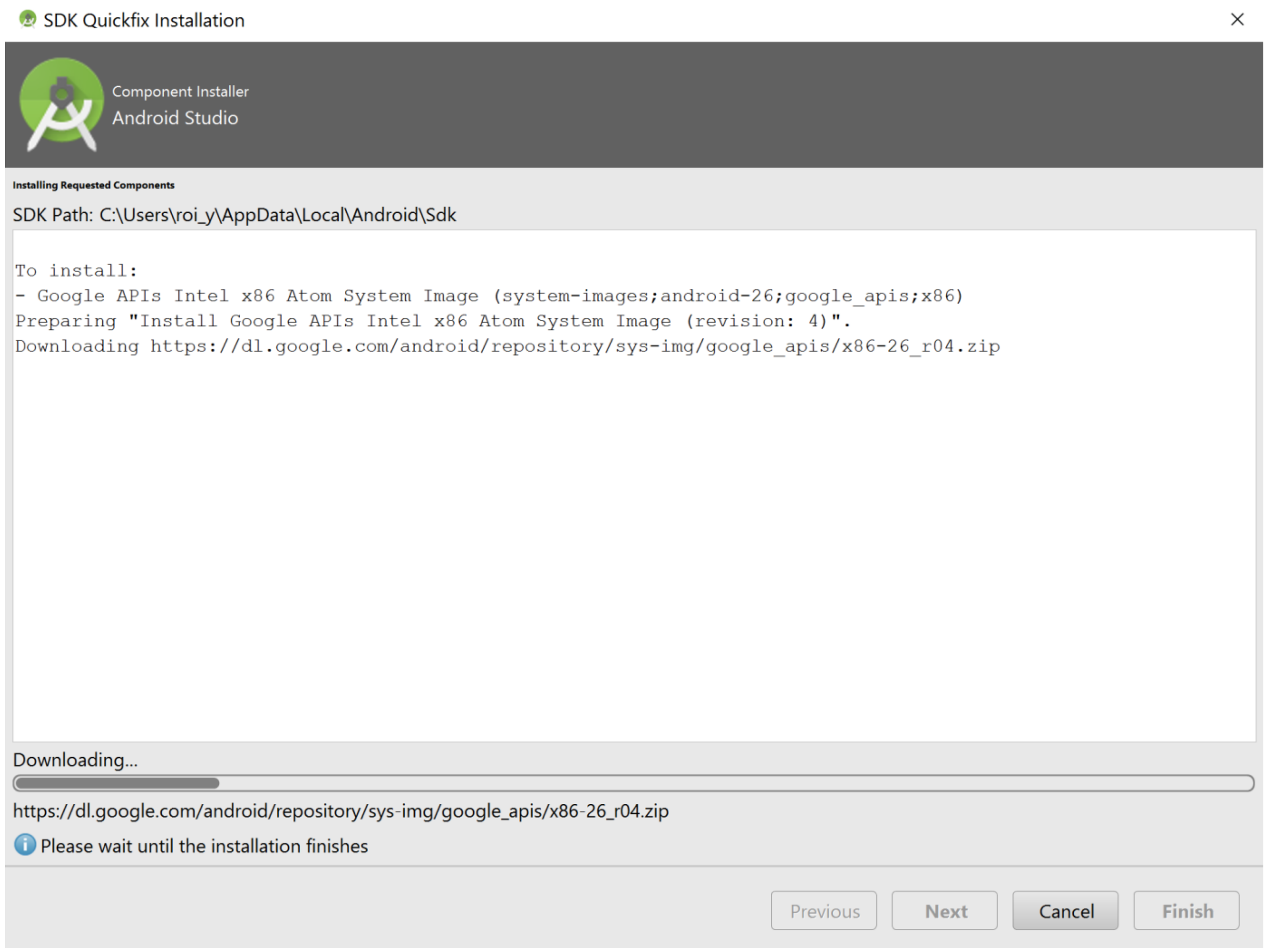
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

1.3 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

2. Accepting the License Agreement

Decline [✓ Accept]

Android Virtual Device (AVD)



Installing Requested Components

SDK Path: C:\Users\roi_y\AppData\Local\Android\Sdk

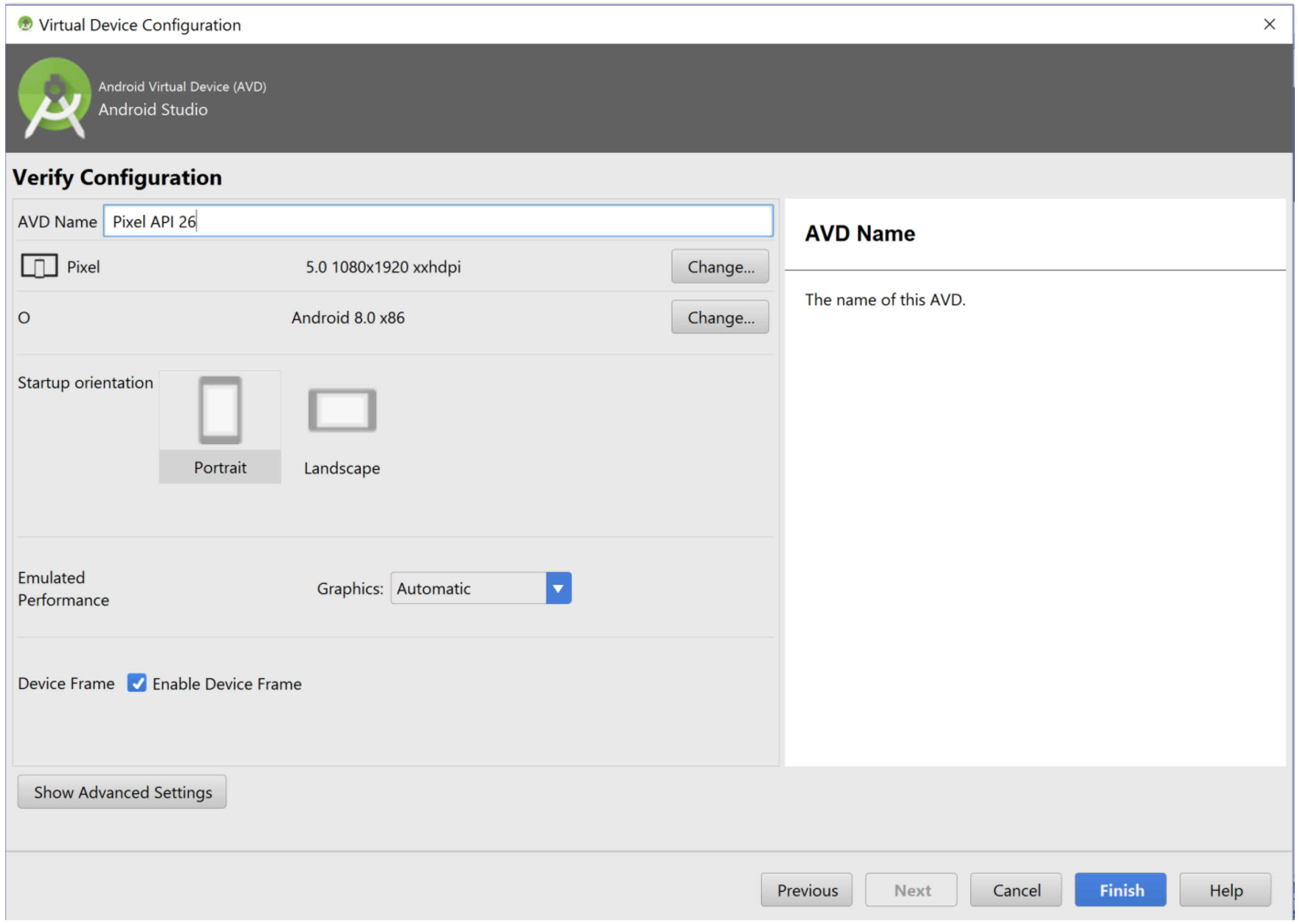
```
To install:
- Google APIs Intel x86 Atom System Image (system-images;android-26;google_apis;x86)
Preparing "Install Google APIs Intel x86 Atom System Image (revision: 4)".
Downloading https://dl.google.com/android/repository/sys-img/google_apis/x86-26_r04.zip
```

Downloading...

https://dl.google.com/android/repository/sys-img/google_apis/x86-26_r04.zip

Please wait until the installation finishes

Android Virtual Device (AVD)


Virtual Device Configuration

Android Virtual Device (AVD)
Android Studio

Verify Configuration

AVD Name: Pixel API 26

Pixel — 5.0 1080x1920 xxhdpi

O — Android 8.0 x86

Startup orientation: Portrait / Landscape

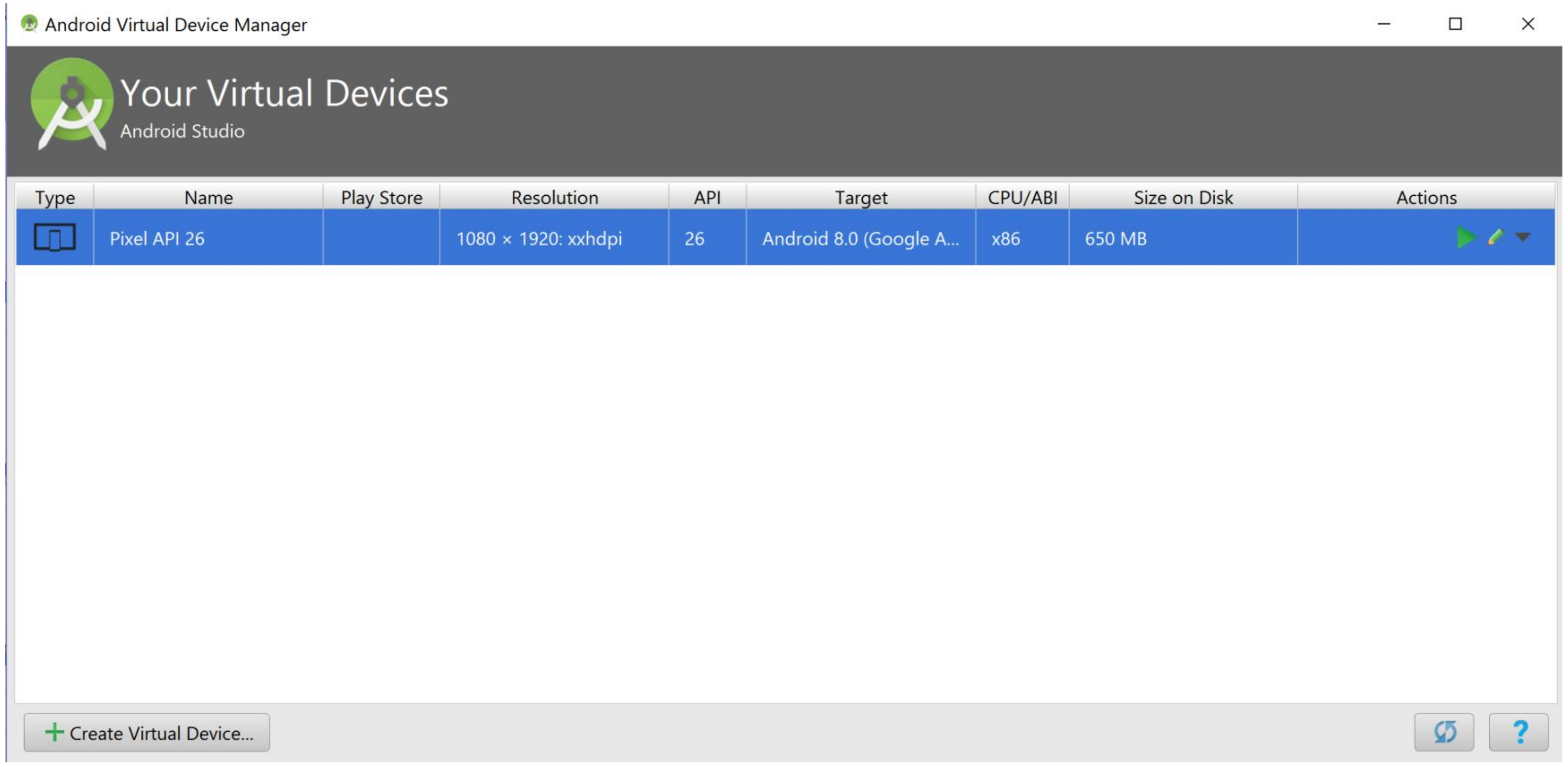
Emulated Performance — Graphics: Automatic

Device Frame: ☑ Enable Device Frame

AVD Name

The name of this AVD.

Run the Emulator

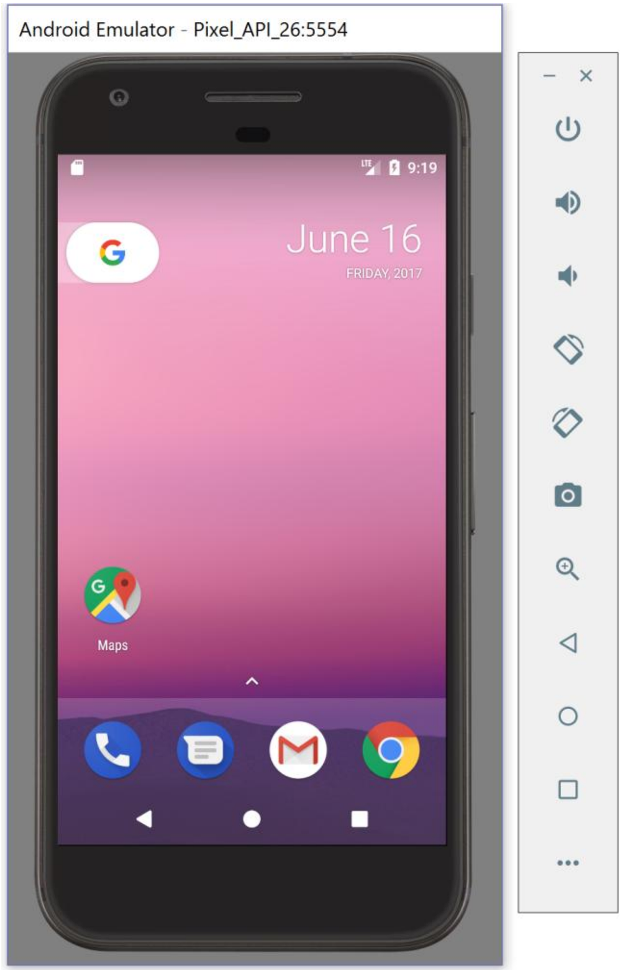


Android Virtual Device Manager

Your Virtual Devices
Android Studio

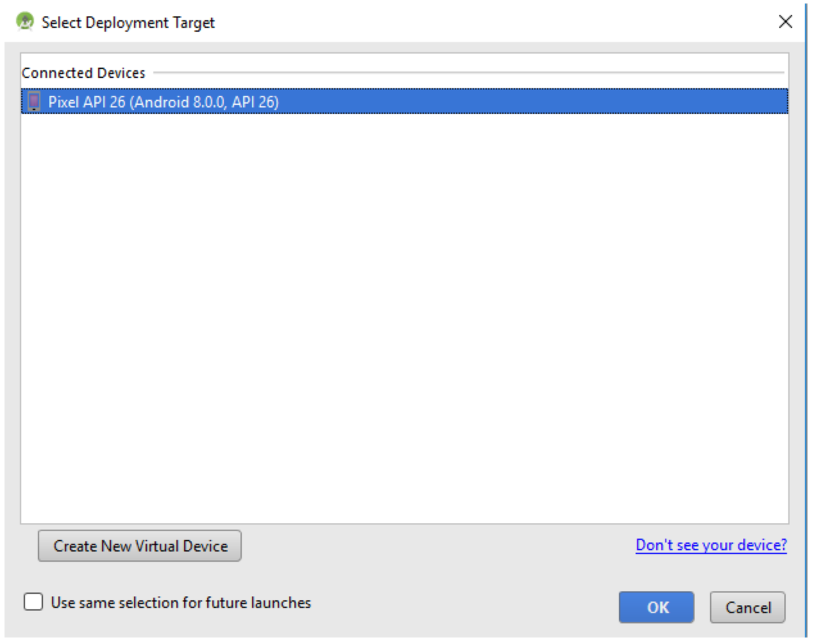
Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel API 26		1080 × 1920: xxhdpi	26	Android 8.0 (Google A...	x86	650 MB	

Run the Emulator

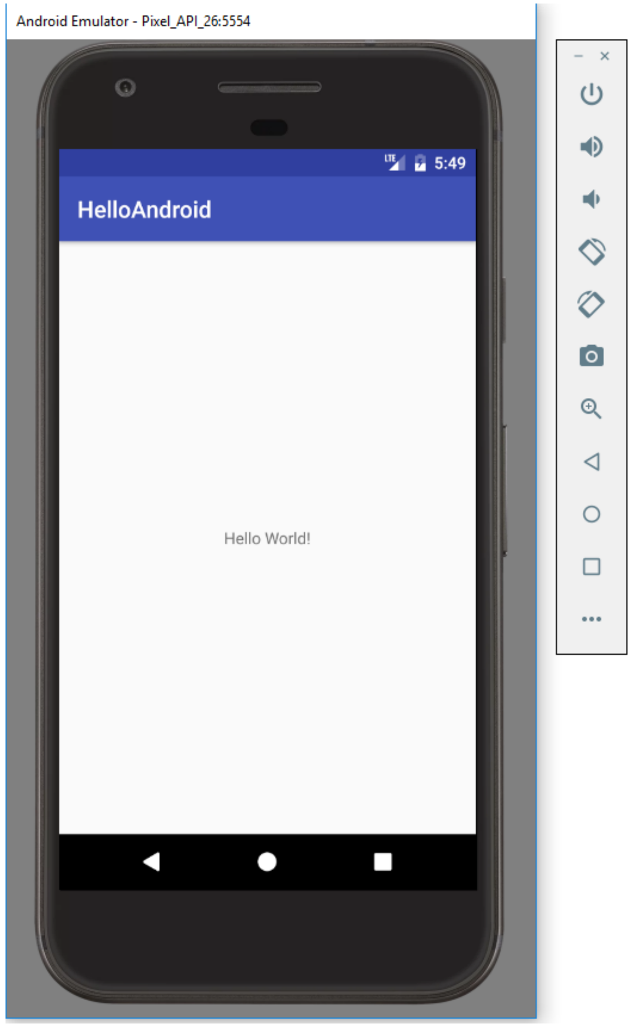


Run Your App on the Emulator

- Once the emulator is booted up, click the **app** module in the **Project** window and then select **Run > Run** (or click **Run** in the toolbar)
- In the **Select Deployment Target** window, select the emulator and click **OK**
- Android Studio installs the app on the emulator and starts it
- That's "hello world" running on the emulator!


Select Deployment Target
Connected Devices
Pixel API 26 (Android 8.0.0, API 26)
Create New Virtual Device
Don't see your device?
Use same selection for future launches
OK
Cancel

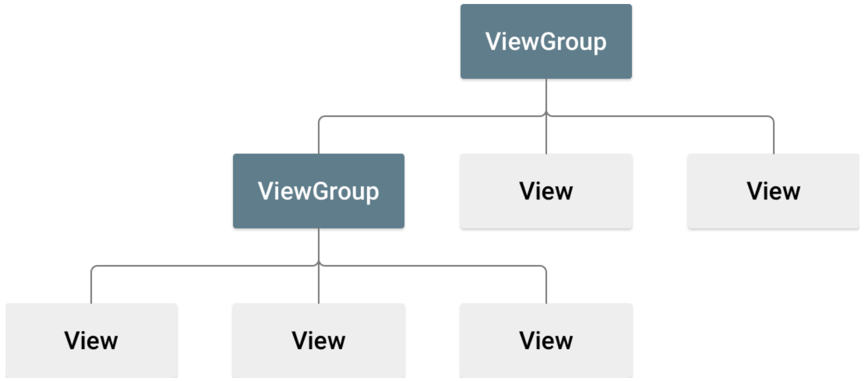
Run Your App on the Emulator



Building User Interface

Build a Simple User Interface

- The UI for an Android app is built using a hierarchy of layouts and widgets
  - Layouts (**ViewGroup** objects) are invisible containers that control how its child views are positioned on the screen
  - Widgets (**View** objects) are UI components such as buttons and text boxes

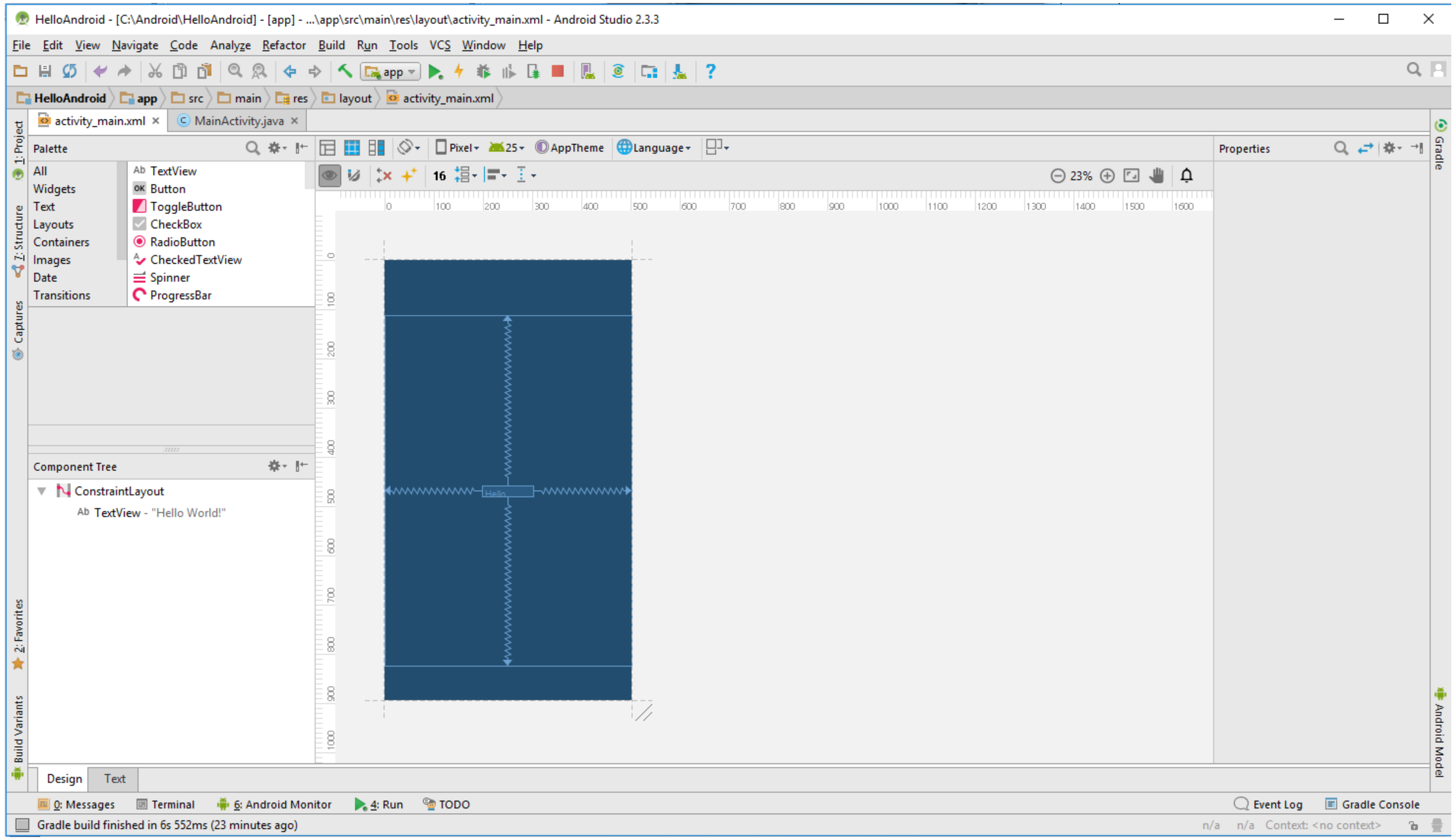


- The UI is defined in XML files stored in the **res/layout** folder
- You can use the Layout Editor to build the layout by drag-and-dropping views

Open the Layout Editor

- In Android Studio's Project window, open **app > res > layout > activity_main.xml**
- To get started, set up your workspace as follows:
  - To make more room for the Layout Editor, hide the **Project** window by clicking **Project** on the left side of Android Studio
  - Click **Show Blueprint** so only the blueprint layout is visible
  - Make sure Show Constraints is on
  - Click **Default Margins** in the toolbar and select **16**
  - Click **Device in Editor** in the toolbar and select **Pixel XL**

Layout Editor



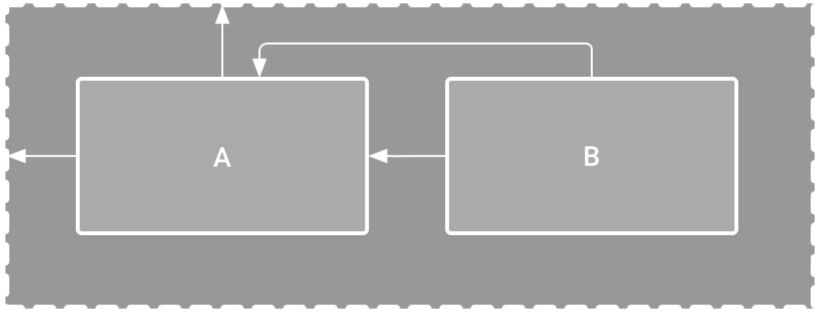
HelloAndroid - [C:\Android\HelloAndroid] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 2.3.3

Gradle build finished in 6s 552ms (23 minutes ago)

n/a n/a Context: <no context>

ConstraintLayout

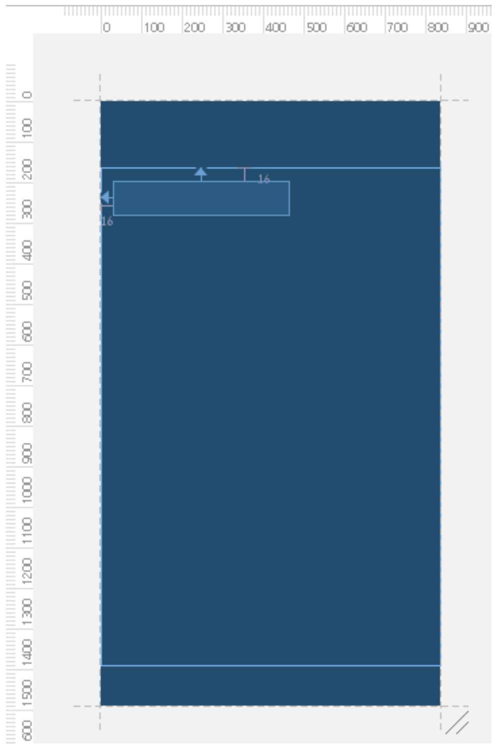
- **ConstraintLayout** is a layout that defines the position for each view based on constraints to sibling views and the parent layout
- It allows you to create both simple and complex layouts with a flat view hieararchy, i.e., it avoids the need for nested layouts, which can increase the time required to draw the UI
- For example, you can declare the following layout:
  - View A appears 16dp from the top of the parent layout
  - View A appears 16dp from the left of the parent layout
  - View B appears 16dp to the right of view A
  - View B is aligned to the top of view A



- **dp** is a device-independent pixel - a virtual pixel unit that you should use to express layout dimensions or position in a density-independent way
  - px = dp * (dpi / 160)

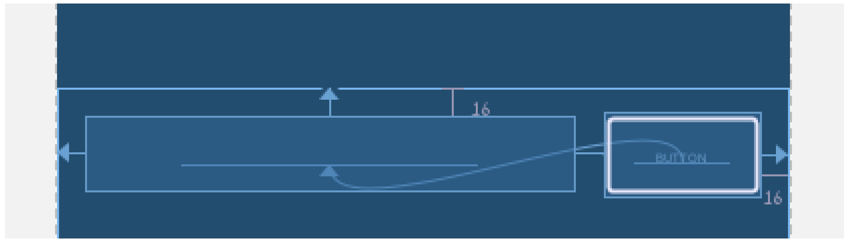
Add a TextBox

- First, you need to remove what's already in the layout
  - So click **TextView** in the **Component Tree** window, and then press Delete
- From the **Palette** window on the left, click **Text** in the left pane, and then drag **Plain Text** into the design editor and drop it near the top of the layout
  - This is an EditText widget that accepts plain text input
- Click the view in the design editor
- You can now see the resizing handles on each corner (squares), and the constraint anchors on each side (circles)
- Click-and-hold the anchor on the top side, and then drag it up until it snaps to the top of the layout and release
  - That's a **constraint** - it specifies the view should be 16dp from the top of the layout (because you set the default margins to 16dp).
- Similarly, create a constraint from the left side of the view to the left side of the layout



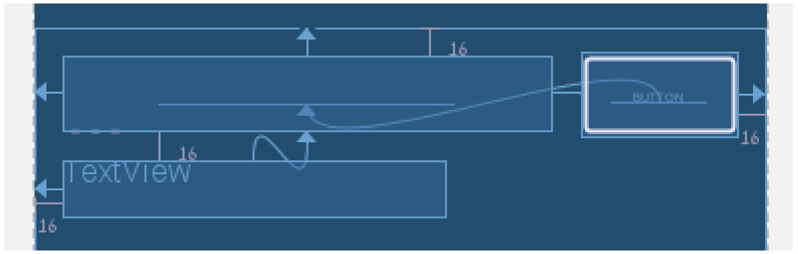
Add a Button

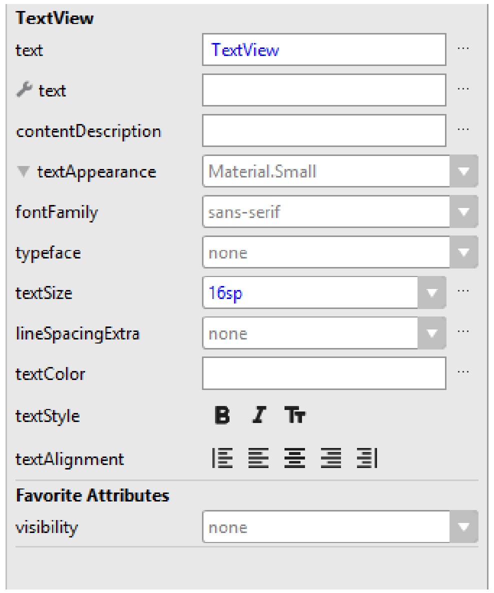
- From the **Palette** window, click **Widgets** in the left pane, and then drag **Button** into the design editor and drop it near the right side
- Add a constraint from the right side of the button to the right side of the layout
- Add a constraint from the right side of the text box to the left side of the button
- To constrain the views in a horizontal alignment, you need to create a constraint between the text baselines
  - So click the button, and then click **Baseline Constraint** below the button
- Click-and-hold the baseline anchor inside the button and then drag it to the baseline anchor that appears in the text box



Add a TextView

- From the **Pallete** window, drag a **TextView** into the layout and place it below the text
- Anchor its left to the left side of the layout
- Anchor its top to the bottom side of the text box
- In the Properties window change its id to textView
- Expand **textAppearance** change the **textSize** to 16sp




TextView

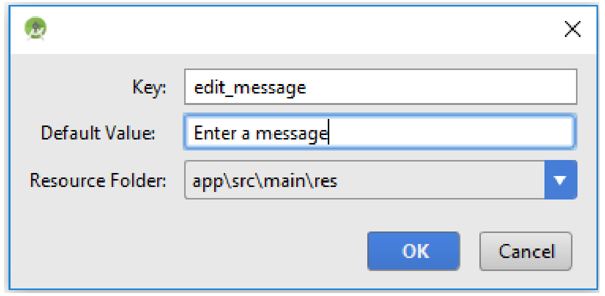
text	TextView
text	
contentDescription	
▼ textAppearance	Material.Small
fontFamily	sans-serif
typeface	none
textSize	16sp
lineSpacingExtra	none
textColor	
textStyle	B I Tт
textAlignment	

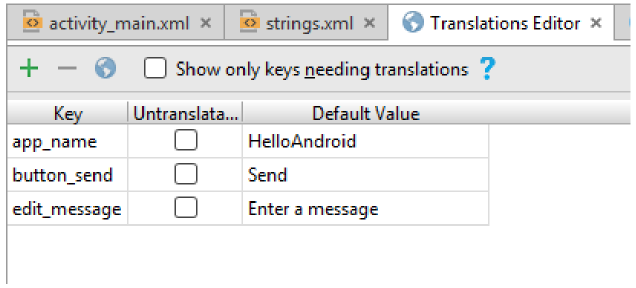
Favorite Attributes

visibility	none

Change the UI Strings

- Click **Show Design** in the toolbar to preview the UI
- Notice that the text input is pre-filled with "Name" and the button is labeled "Button." So now you'll change these strings.
- Open the **Project** window and then select **res > values > strings.xml**.
  - This is a string resources file where you should specify all your UI strings
- Click **Open editor** at the top of the editor window.
  - This opens the **Translations Editor**, which provides a simple interface for adding and editing your default strings
- Click **Add Key** to create a new string as the "hint text" for the text box
  - Enter "edit_message" for the key name
  - Enter "Enter a message" for the value
  - Click **OK**
- Add another key named "button_send" with a value of "Send.“

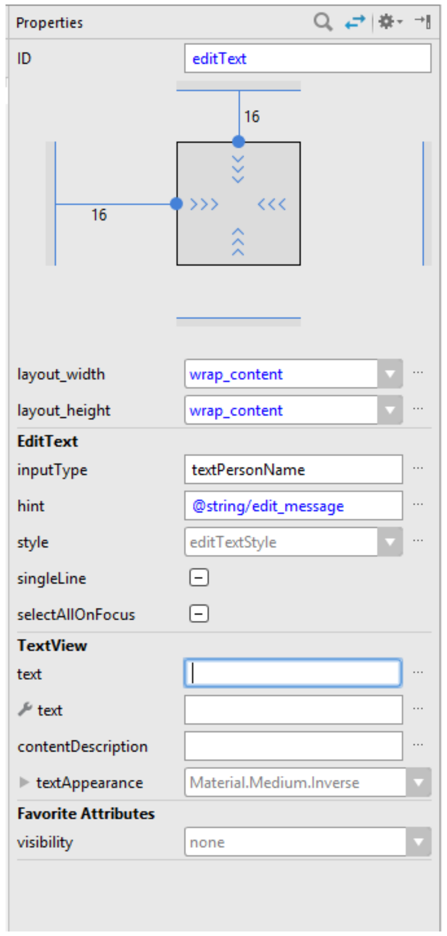

Key: edit_message
Default Value: Enter a message
Resource Folder: app\src\main\res



Key	Untranslata...	Default Value
app_name	☐	HelloAndroid
button_send	☐	Send
edit_message	☐	Enter a message

Change the UI Strings

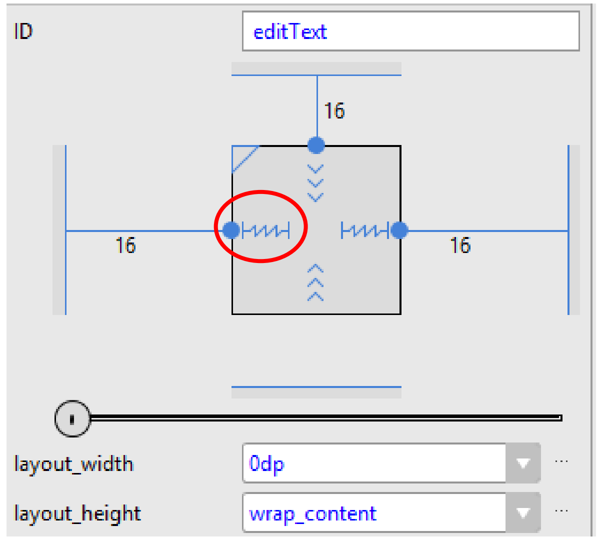
- Now you can set these strings for each view
- Return to the layout file by clicking **activity_main.xml**
- Add the strings as follows:
  - Click the text box in the layout and, if the **Properties** window isn't already visible on the right, click **Properties** on the right sidebar
  - Locate the **hint** property and then click **Pick a Resource** to the right of the text box
  - In the dialog that appears, double-click on **edit_message**
  - Also delete the **text** property (currently set to "Name")
- Click the button in the layout, locate the **text** property, click **Pick a Resource**, and then select **button_send**.
- Click the text view in the layout and delete its **text** property



ID: editText
layout_width: wrap_content
layout_height: wrap_content
EditText
inputType: textPersonName
hint: @string/edit_message
style: editTextStyle
singleLine
selectAllOnFocus
TextView
text
text
contentDescription
textAppearance: Material.Medium.Inverse
Favorite Attributes
visibility: none

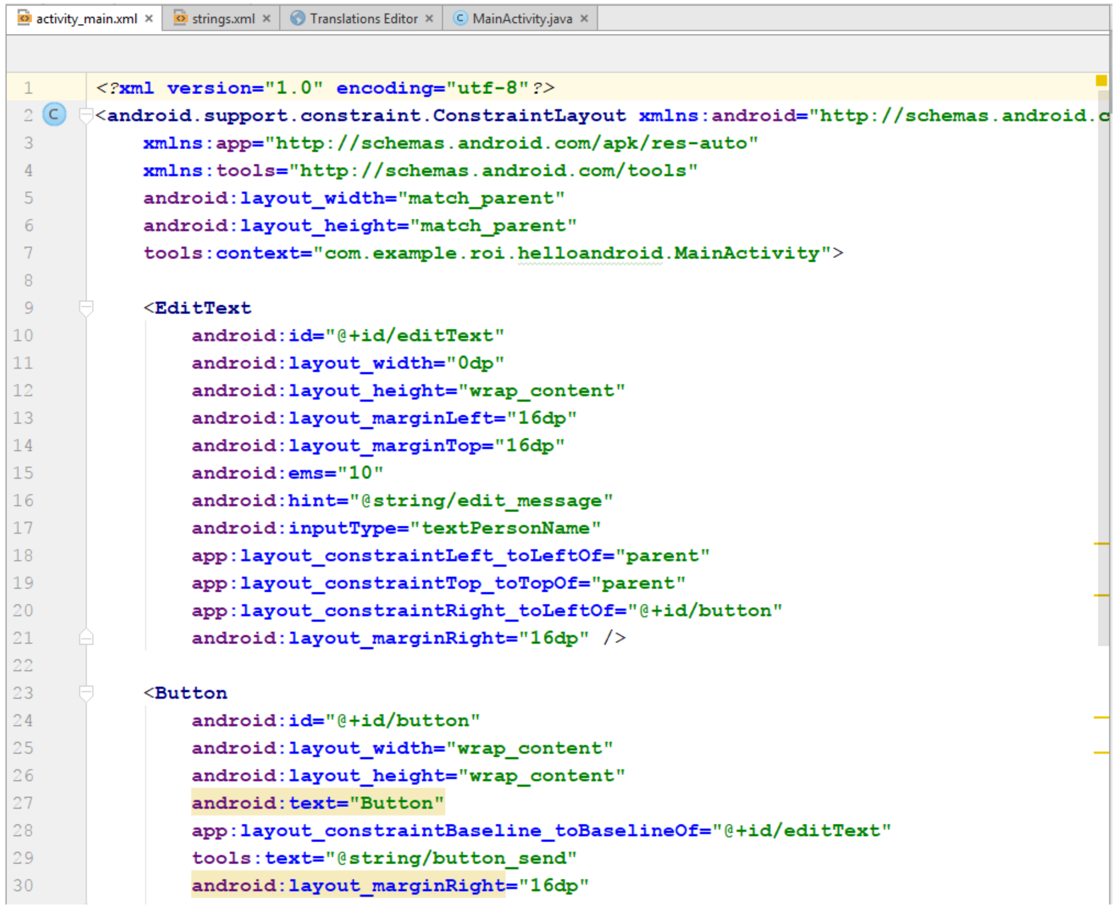
Make the Text Box Size Flexible

- To create a layout that's responsive to different screen sizes, you'll now make the text box stretch to fill all remaining horizontal space (after accounting for the button and margins)
- Open the **Properties** window for the text box and then click the width indicator until set to **Match Constraints**
  - “Match constraints" means that the width is now determined by the horizontal constraints and margins. Therefore, the text box stretches to fill the horizontal space.


ID: editText
layout_width: 0dp
layout_height: wrap_content

XML Layout File

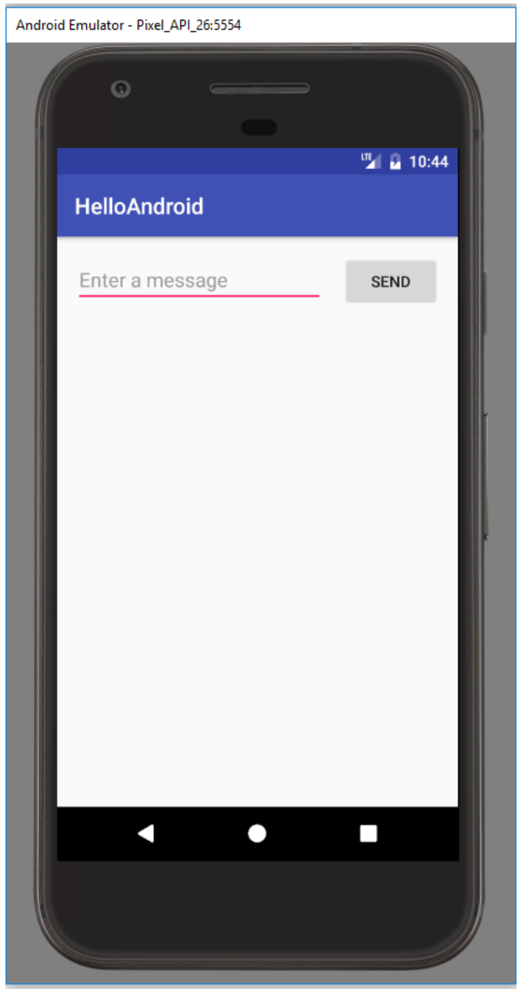
- Click the Text tab to see the final XML layout code



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.c
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.roi.helloandroid.MainActivity">

    <EditText
        android:id="@+id/editText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:hint="@string/edit_message"
        android:inputType="textPersonName"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/button"
        android:layout_marginRight="16dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBaseline_toBaselineOf="@+id/editText"
        tools:text="@string/button_send"
        android:layout_marginRight="16dp"
```

Run the App

- If your app is already installed on the device, click Apply Changes ⚡ from the toolbar
- Or click **Run** to install and run the app
- Next we will show the message entered in the text box on the text view when the button is tapped



Main Activity

- Open the file **app > java > com.example.myfirstapp > MainActivity.java**

```
package com.example.roi.helloandroid;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Base class for activities that use action bar features

onCreate() is where you initialize your activity

Sets the activity content from a layout resource

Handle Events

- In **MainActivity.java**, add the sendMessage() method as shown below:

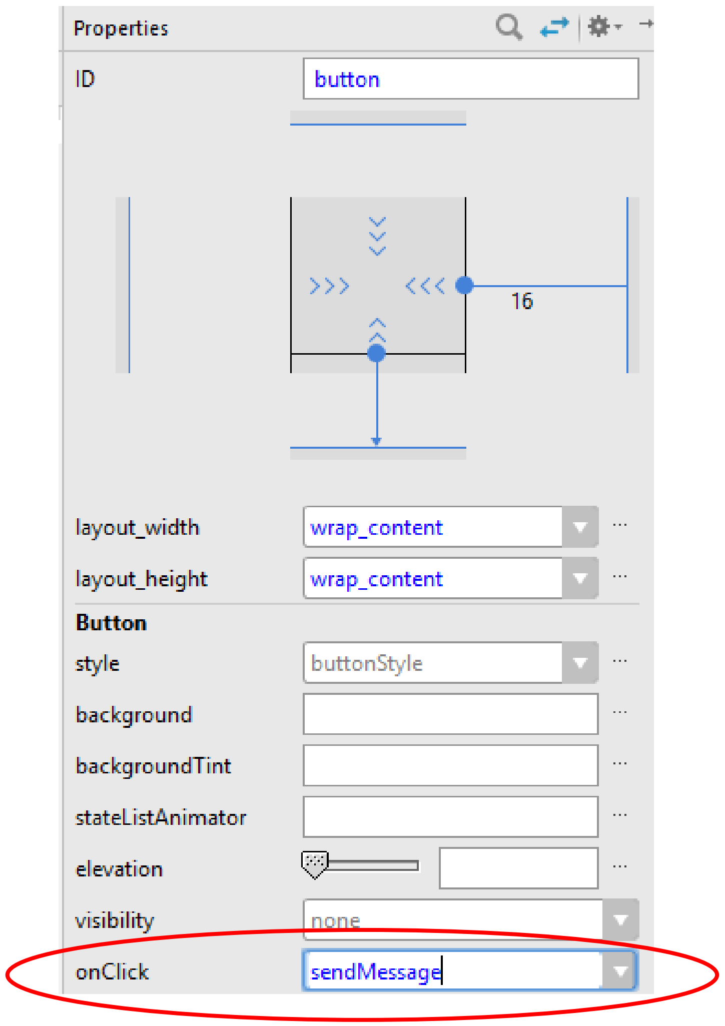
```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    // Called when user taps the Send button
    public void sendMessage(View view) {
        EditText editText = (EditText)findViewById(R.id.editText);
        TextView textView = (TextView)findViewById(R.id.textView);

        textView.setText("Message: " + editText.getText().toString());
    }
}
```

findViewById() finds a view that is identified by the android:id XML attribute

R.java is a dynamically generated class that identifies all assets (from strings to layouts), for usage in java classes

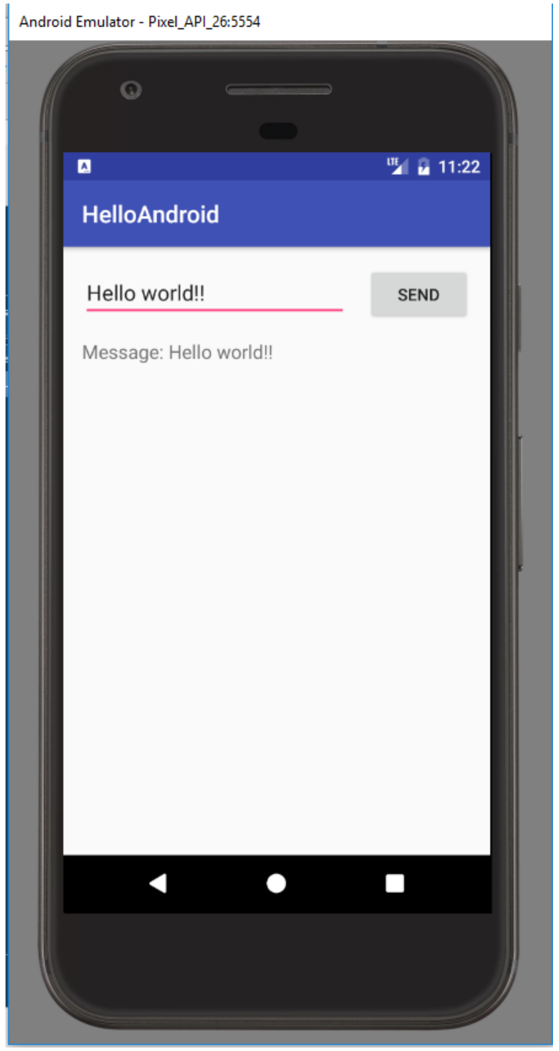
Handle Events

- Now return to the **activity_main.xml** file to call this method from the button:
  - Click to select the button in the Layout Editor.
  - In the **Properties** window, locate the **onClick** property and select **sendMessage [MainActivity]** from the drop-down list.



ID	button
layout_width	wrap_content
layout_height	wrap_content
style	buttonStyle
background	
backgroundTint	
stateListAnimator	
elevation	
visibility	none
onClick	sendMessage

Run the App

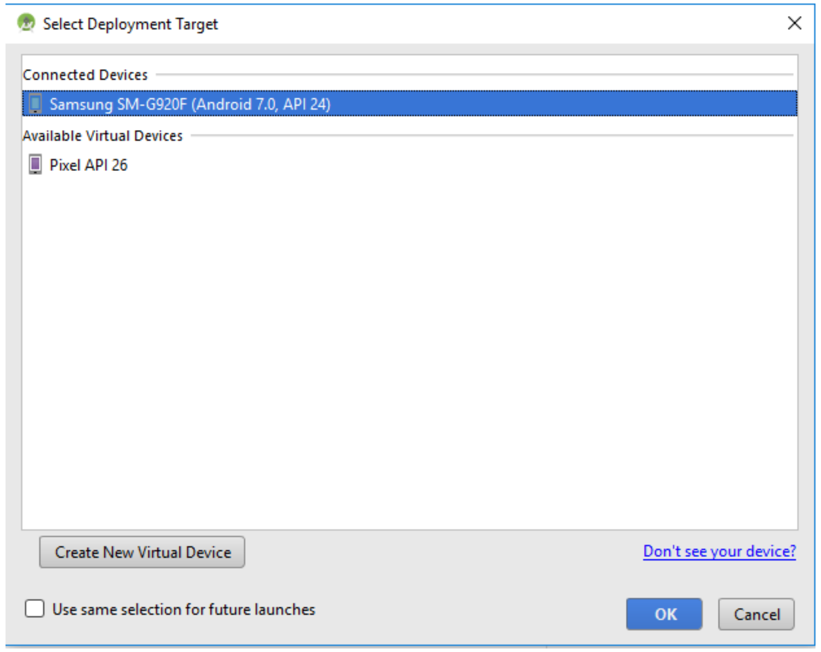


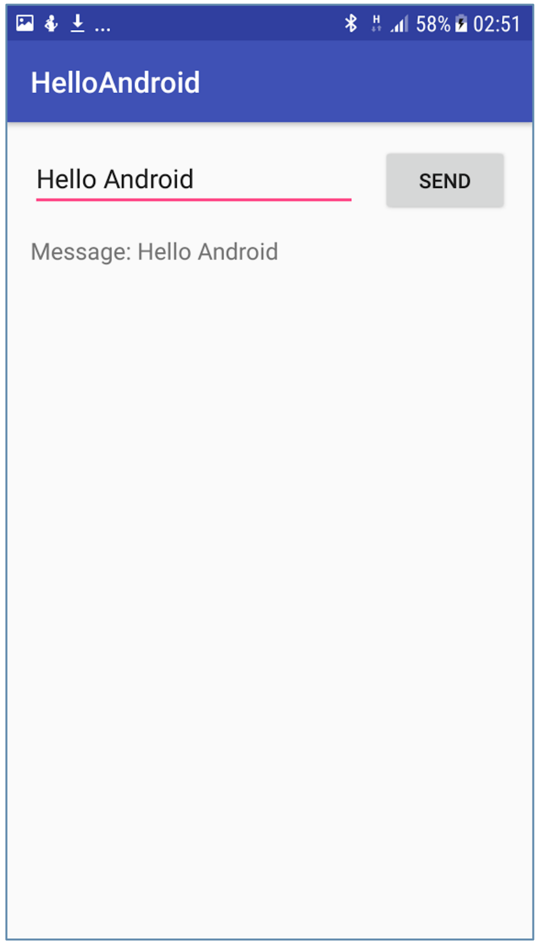
Message: Hello world!!

Run Your App on a Real Device

- Set up your device as follows:
  - Connect your device to your development machine with a USB cable.
  - If you're developing on Windows, you might need to install the appropriate USB driver for your device. For help installing drivers, see the OEM USB Drivers document.
  - Enable **USB debugging** on your device by going to **Settings > Developer options**.
  - **Note:** On Android 4.2 and newer, **Developer options** is hidden by default. To make it available, go to **Settings > About phone** and tap **Build number** seven times. Return to the previous screen to find **Developer options**.
- Run the app from Android Studio as follows:
  - In Android Studio, click the **app** module in the **Project** window and then select **Run > Run** (or click **Run** in the toolbar).
  - In the **Select Deployment Target** window, select your device, and click **OK**.
  - Android Studio installs the app on your connected device and starts it.

Run the App on a Real Device


Select Deployment Target
Connected Devices
Samsung SM-G920F (Android 7.0, API 24)
Available Virtual Devices
Pixel API 26
Create New Virtual Device
Don't see your device?
Use same selection for future launches
OK
Cancel


58% 02:51
HelloAndroid
Hello Android
SEND
Message: Hello Android