# Resources Android App



**Aims:**

To introduce developing an App for Android with image and sound resources

### Objectives:

* Importing image and sound resources
* Using background images and image buttons
* The MediaPlayer class
* The SoundPool class

Let’s review lecture

* How are resources imported into your Android Studio project?
* What’s the difference between MediaPlayer and SoundPool?
* Describe the steps in enabling an image button to display and respond to click events

## 1. Introduction

Here we’ll use Android Studio to develop an application that includes images and sounds.

## 2. Starting a new project

Let’s get started…

Open Android Studio and ‘Start a new Android Studio project’.

Choose an ‘Empty Activity’. Click [Next].

In the ‘Configure your new project’ dialog set your Application’s…

* name to ‘ResourcesApp’ or similar
* package name to com.your\_initials.resourcesapp
* ‘Save location’ to a suitable directory on your OneDrive
* ‘Language’ to **Java** **(NOT Kotlin!)**
* ‘Minimum API level’ to suggested default *…e.g. API 16: Android 4.1 (Jelly Bean)*

Click [Finish].

Allow a minute or so to let Gradle complete executing tasks. You will be presented with the Android Studio Integrated Development Environment.

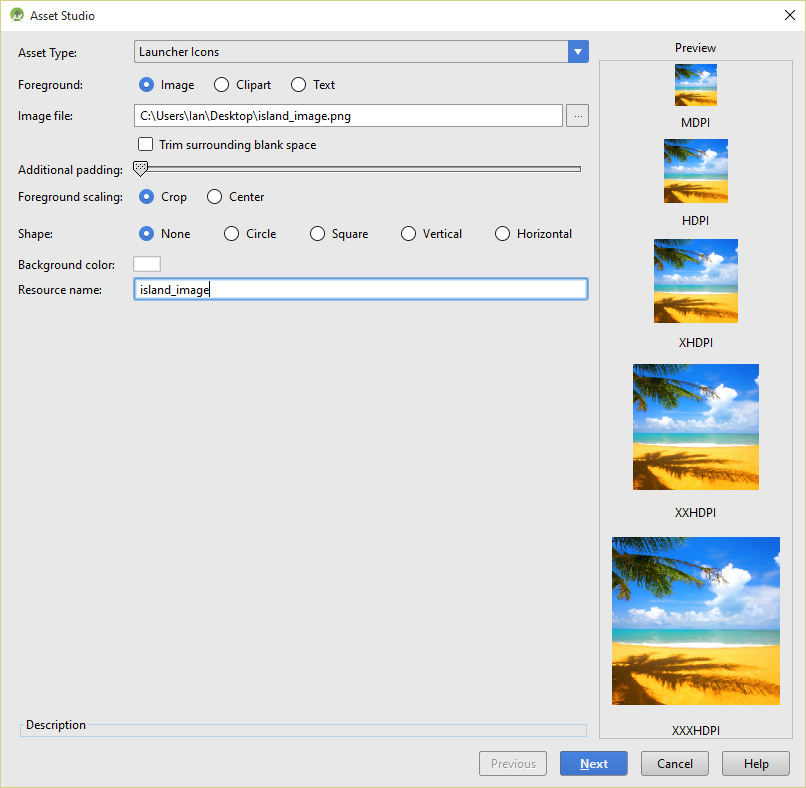
## Adding an image resource

Download a jpg or png suitable for a background image.



Via the resources directory, you can drag raw images into the res/drawable directory. If you can’t drag and drop, try moving the image into the project’s res/drawable directory via the file system.

Alternatively if you wish the image to be used with a range of fixed sizes and some padding options you can create a New->Image Asset. Hence, use the resulting dialog box to select your newly downloaded image file and associate with it a new resource name, e.g. ‘island\_image’.



To set the background to your image resources you could try editing the activity\_main.xml file directly.

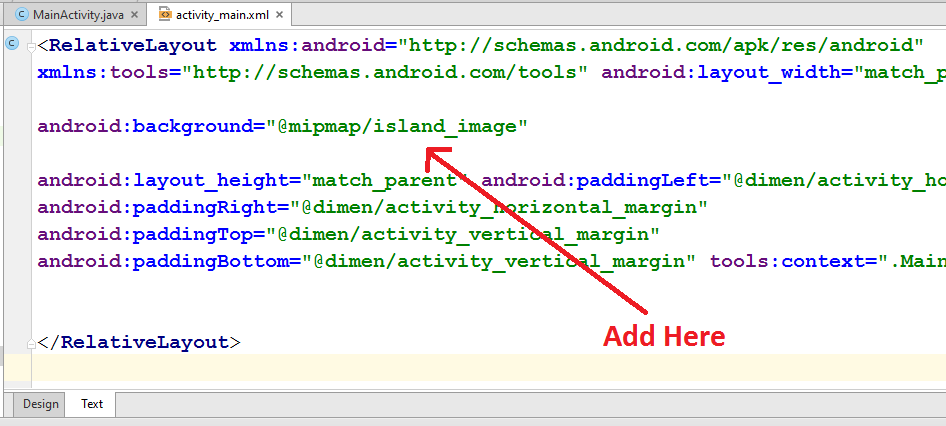
Add this following line of XML

android:background="@drawable/island\_image"

or if your image is in a mipmap directory

android:background="@mipmap/island\_image"

Where instead of ‘island\_image’ you have the name of your image resource.



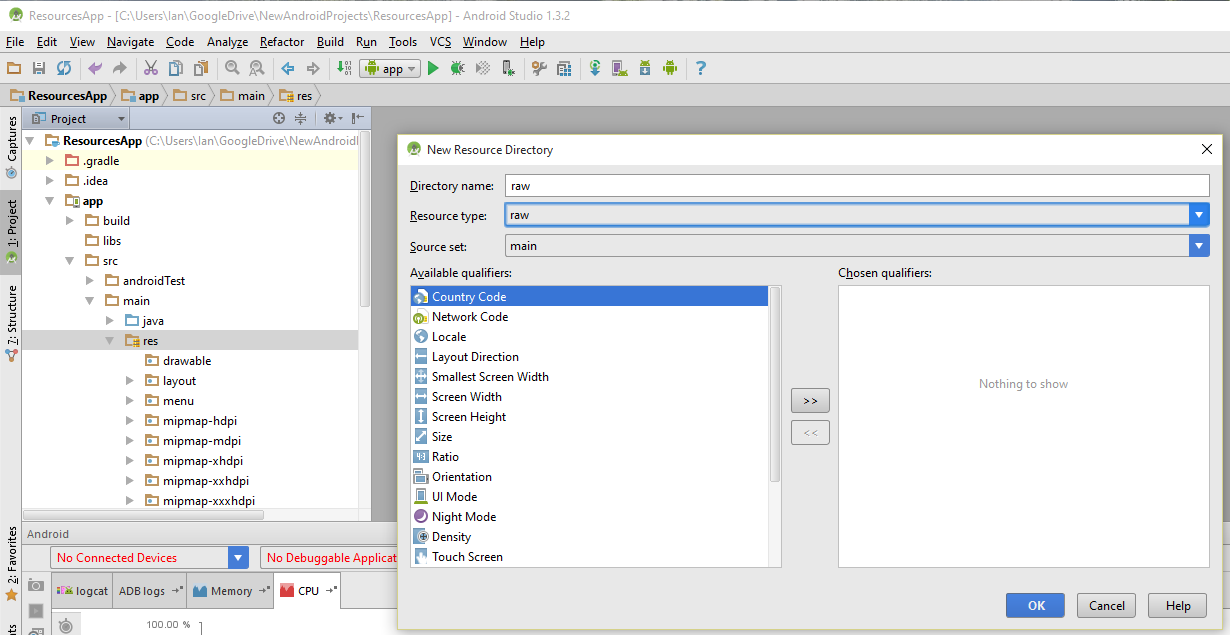
Save and Run As Android application.



## Adding a sound resource

Download or copy from C:\Windows\Media a suitable short wav file for a notification sound.

In the Project Explorer right click on ‘res’ to create a new Android Resource Directory with Directory Name ‘raw’ and Resource Type ‘raw’.



Copy and paste your sound file into this folder.

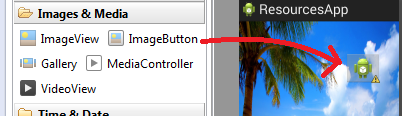
## 

If you can’t drag and drop, try moving sounds into the raw project directory you created via the file system. If you are prompted for a file type, just click cancel.

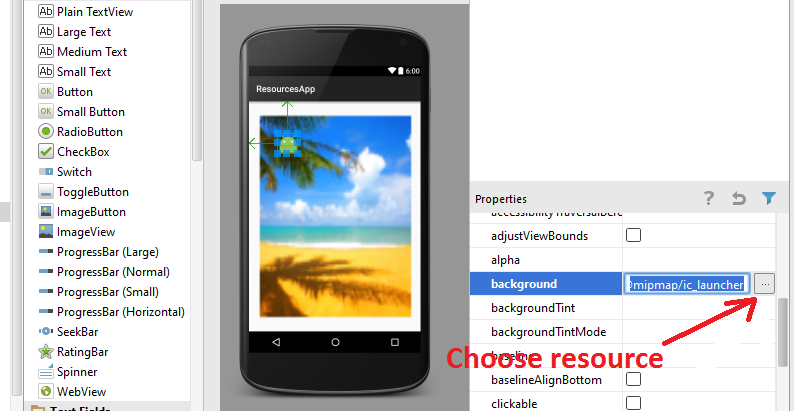
## Playing the sound with MediaPlayer

To play the sound let’s add an image button and a listener to handle its click event.

From the Graphics Layout editor drag and drop and image button onto your design area.



Via the Properties you can set the background image for your button. You’ll be presented with a ‘Resource Chooser’ dialog. For now choose the ‘ic\_launcher’ option, though you’ll be able to choose from your own imported images for the button.



To add interaction you’ll edit MainActivity.java

Add to as fields of the MainActivity class

ImageButton b1;

And

MediaPlayer mp1;

The MediaPlayer class can be used to control playback of audio/video files and streams.

Add as a field a listener for handling the image button click event

OnClickListener b1Handler = **new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

mp1.start();

}};

Where bHandler is an instance of an anonymous class that is a realisation of the OnClickListener interface. We use the concrete implementation of onClick() to start the media player.

To the onCreate() method add the following

b1= (ImageButton) findViewById(R.id.*imageButton*);

b1.setOnClickListener(b1Handler);

This binds our button reference to its XML widget and associates it with the listener object.

*(make sure you saved activity\_main.xml or you won’t see the imagebutton id)*

Also add the following, where instead of ‘chimes’ use the id for your wav file.

mp1=MediaPlayer.*create*(**this**, R.raw.*chimes*);

This binds the media player reference with a media player object to play our wav file.

Save and Run As an Android Application.

## Playing the sound with SoundPool

For short sounds like effects in a game you’ll get better performance with sound pool. Note SoundPool has been replaced by SoundPool.Builder for API 21 and above.

The following code is an example how to use SoundPool.builder

**package** com.example.ian.mysound;  
  
**import** android.content.DialogInterface;  
**import** android.media.AudioAttributes;  
**import** android.media.AudioManager;  
**import** android.media.SoundPool;  
**import** android.os.Build;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 SoundPool **mySounds**;  
 **int s1**;  
 **int s2**;  
 AudioAttributes **audioAttributes**;  
 View.OnClickListener **h** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **mySounds**.play (**s1**, 1.0f, 1.0f, 1, 0, 1);  
 **mySounds**.play (**s2**, 1.0f, 1.0f, 1, 0, 1);  
 }  
 };  
 Button **b1**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***LOLLIPOP***) {  
 *//code greater or equal to API 21 (lollipop)* **audioAttributes** = **new** AudioAttributes.Builder()  
 .setContentType(AudioAttributes.***CONTENT\_TYPE\_MUSIC***)  
 .setUsage(AudioAttributes.***USAGE\_GAME***)  
 .build();  
 **mySounds** = **new** SoundPool.Builder()  
 .setMaxStreams(2)  
 .setAudioAttributes(**audioAttributes**)  
 .build();  
 }  
 **else** {  
 **mySounds** = **new** SoundPool(2, AudioManager.***STREAM\_MUSIC***, 1);  
 }  
 **s1** = **mySounds**.load(**this**, R.raw.***cat***, 1);  
 **s2** = **mySounds**.load(**this**, R.raw.***dog***, 1);  
  
 **b1**= (Button) findViewById(R.id.***button***);  
 **b1**.setOnClickListener(**h**);  
 }  
}

Note the arguments are …

*mysounds.play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate);*

## EXERCISE

Create a small drum set of image buttons and associated sounds by incorporating some more short wav files and smallish images into your app. Surf the web for sound and image resources.

## Complete Code Example

**package** com.example.ian.resourcesapp;  
  
**import** android.media.MediaPlayer;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.ImageButton;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 ImageButton **b1**;  
 MediaPlayer **mp1**;  
 View.OnClickListener **b1Handler** = **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View arg0) {  
 **mp1**.start();  
 }};  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **b1**= (ImageButton) findViewById(R.id.***imageButton***);  
  
 **b1**.setOnClickListener(**b1Handler**);  
 **mp1**=MediaPlayer.*create*(**this**, R.raw.***chimes***);  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 *// Inflate the menu; this adds items to the action bar if it is present.* getMenuInflater().inflate(R.menu.***menu\_main***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 *// Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.* **int** id = item.getItemId();  
  
 *//noinspection SimplifiableIfStatement* **if** (id == R.id.***action\_settings***) {  
 **return true**;  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
}