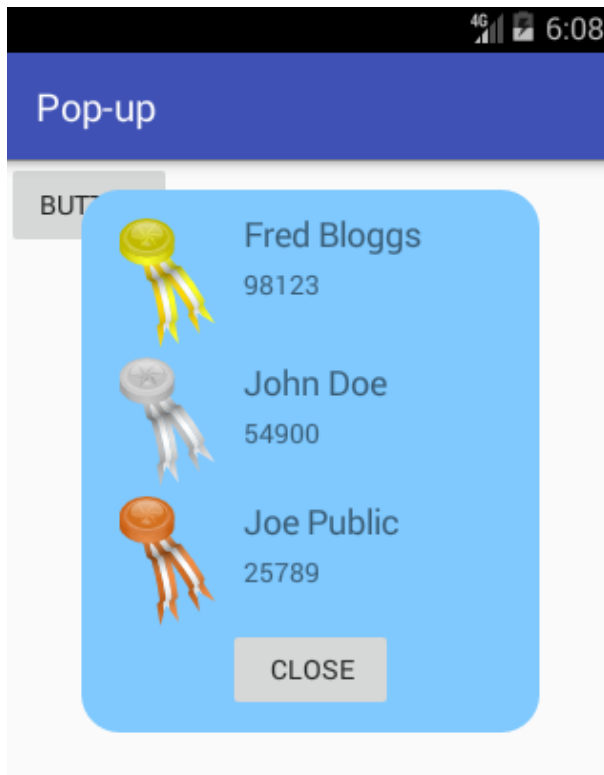



[Home](#) [Index](#) [Shop](#) [Advertising](#) [About](#) [Contact](#)

## Pop-up Window in Android

The main screen for an `Activity` is not the only way to interact with the user of an app. A brief message or dialog can be displayed in appropriate circumstances to ask a specific question, get specific input, or show a brief message. Android has build in support for such small focused interactions that require immediate attention.



 [↓markdown↓ CMS](#) is fast and simple. Build websites quickly and publish easily. For beginner to expert.

---

Articles on:

[Android Programming and Android Practice Projects](#), [HTML](#), [VPS](#), [Computing](#), [IT](#), [Computer History](#), [↓markdown↓ CMS](#), [C#](#)

(This Android pop-up tutorial assumes that [Android Studio is installed](#), a [basic App can be created](#) and run, and the code in this article can be [correctly copied](#) into Android Studio. The example can be changed to meet other requirements. When entering code in Studio add import statements when prompted by pressing *Alt-Enter*.)



## How to Display A Smaller Window on Top of an Activity

There are different ways in Android to pop-up a brief window over the current Activity screen, these include:

- The Toast class – to display brief informational only message.
- The Dialog class, managed by a DialogFragment – to support a flexible input that can display information and can ask for inputs or choices. There are several built in sub-classes including:
  - AlertDialog – supports buttons (from zero to three), list selections, check boxes and radio buttons (see the article [About Box in Android App Using AlertBuilder](#)).
  - ProgressDialog – supports a progress bar or wheel and supports buttons because it extends AlertDialog.
  - DatePickerDialog – supports selecting a date.
  - TimePickerDialog – supports a time selection.
- The PopupWindow class – allows a View to float on top of an existing activity. Suitable for custom informational messages.

These classes build their user interfaces through class methods with support from custom layouts when required. For this article the `PopupWindow` class is used. This Android pop-up tutorial includes example code. What is the PopupWindow? The [Android developer PopupWindow documention](#) gives this class overview:

*"This class represents a popup window that can be used to display an arbitrary view. The popup window is a floating container that appears on top of the current activity."*

It can be used to display extra information without needing to code another Activity. An example would be to show results at the end of a game (or end of a level), which will be done in this tutorial. The steps for this

[Programming, Using Windows for Programming](#)

---

Free Android Projects and Samples:

[Android Examples](#), [Android List Examples](#), [Android UI Examples](#)

---

Maintain your PC with [Piriform](#)

[Shop at Tek Eye for PC utilities](#)

---

example project and code are:

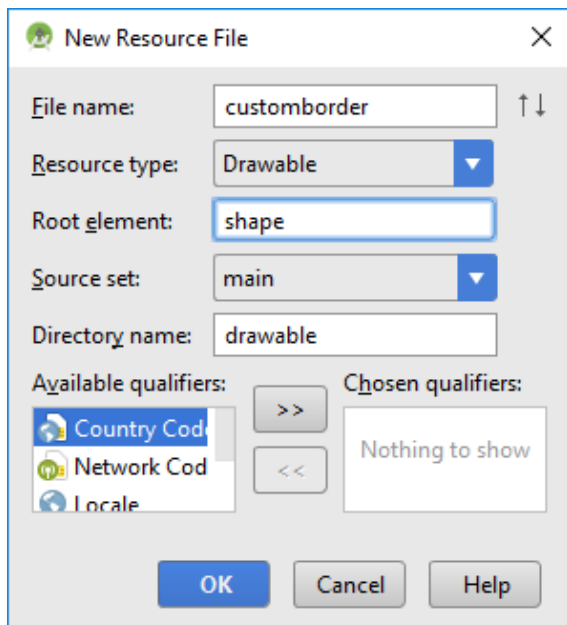
- Create a new project in Android Studio.
- Design the pop-up window.
- Add the code to load and populate the pop-up window.
- Call the code.

## Design the Layout for the Pop-Up

For this example a pop-up is going to display a game's result as gold, silver and bronze medal positions. The pop-up will have a graphic, shown in a `ImageView`, for each of the medals; a `TextView` for the winner's name, and one for the score. The images for the medals came from the [Open Clip Art Library](#) by user [momoko](#). They have been resized for the Android project and are available in [medals\\_png.zip](#). Ready to download and add into the app project.



Start a new app project in Android Studio. Here it was called *Pop-up* and uses an *Empty Activity* with all other settings left at their default values. The pop-up window will be defined in a new layout XML file in the project's layout folder, called *winners.xml*. To add some interest to the pop-up screen it will have a blue background and rounded corners, this is done using a shape drawable (see [Add a Border to an Android Layout](#)). The shape drawable for the custom background definition is a drawable resource XML file in the folder **res/drawable**. With the *app* or *res* folder highlighted in the *Project* explorer use the *File* or context (often called right-click) menu. Select the *New* then *Android resource file* option. Call the new resource *customborder.xml*:



Add this code for the pop-up window background shape definition. The `android:shape` is a rectangle:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">
  <corners android:radius="20dp"/>
  <padding android:left="10dp" android:right="10dp" android:top="10dp" android:bottom="10dp"/>
  <solid android:color="#7FC9FF"/>
</shape>
```

Add the images for the medals to the project's `res/drawable` folders (see [medals\\_png.zip](#)). Now create the `winners.xml` layout file, again use the `New` menu and select `XML` then `Layout XML File`. A `RelativeLayout` is used for the `Root Tag`. Open the file, add the `ImageViews` for the medals and `TextViews` for winners names and scores. Here is the code used for the layout used in this example:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/popup_element"
  android:layout_width="240dp"
  android:layout_height="285dp">
```

```
android:background="@drawable/customborder">
<ImageView android:id="@+id/goldMedal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="5dp"
    android:src="@drawable/gold_medal" />
<TextView android:id="@+id/goldName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/goldMedal"
    android:layout_marginLeft="15dp"
    android:layout_toEndOf="@+id/goldMedal"
    android:layout_toRightOf="@+id/goldMedal"
    android:text="Gold Name"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<TextView android:id="@+id/goldScore"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/goldName"
    android:layout_below="@+id/goldName"
    android:layout_marginTop="5dp"
    android:text="Gold Score"
    android:textAppearance="?android:attr/textAppearanceSmall" />
<ImageView android:id="@+id/silverMedal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/goldMedal"
    android:layout_below="@+id/goldMedal"
    android:layout_marginTop="5dp"
    android:src="@drawable/silver_medal" />
<TextView android:id="@+id/silverName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/goldName"
```

```
        android:layout_alignTop="@+id/silverMedal"
        android:text="Silver Name"
        android:textAppearance="?android:attr/textAppearanceMedium" />
<TextView android:id="@+id/silverScore"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/silverName"
    android:layout_below="@+id/silverName"
    android:layout_marginTop="5dp"
    android:text="Silver Score"
    android:textAppearance="?android:attr/textAppearanceSmall" />
<ImageView android:id="@+id/bronzeMedal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/silverMedal"
    android:layout_below="@+id/silverMedal"
    android:layout_marginTop="5dp"
    android:src="@drawable/bronze_medal" />
<TextView android:id="@+id/bronzeName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/silverName"
    android:layout_alignTop="@+id/bronzeMedal"
    android:text="Bronze Name"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<TextView android:id="@+id/bronzeScore"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/bronzeName"
    android:layout_marginTop="5dp"
    android:layout_alignLeft="@+id/bronzeName"
    android:text="Bronze Score"
    android:textAppearance="?android:attr/textAppearanceSmall" />
<Button android:id="@+id/close"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/bronzeMedal"
```

```
        android:layout_centerHorizontal="true"
        android:text="Close" />
</RelativeLayout>
```

## Code to Load a Pop-up Window

The code to display the pop-up window will be wrapped in a function called *showWinners()*. It can be called from a button press or other action. The first code in the function *inflates* the pop-up window layout using a

`LayoutInflater`:

```
//We need to get the instance of the LayoutInflater, use the context of this activity
LayoutInflater inflater = (LayoutInflater) MainActivity.this.getSystemService(Context.LAYOUT_INFL
//Inflate the view from a predefined XML layout (no need for root id, using entire layout)
View layout = inflater.inflate(R.layout.winners,null);
```

The inflated layout is used to get the id of the TextViews. The ids are used to set the names of the winners and their scores, e.g. here we assume that the name of the gold medal winner is stored in the string *goldWinner*:

```
((TextView)layout.findViewById(R.id.goldName)).setText(goldWinner);
```

The pop-up window is created by passing in the layout and required size. We defined a size in the XML which is used here, scaled by the screens density to support different devices:

```
//Get the devices screen density to calculate correct pixel sizes
float density=MainActivity.this.getResources().getDisplayMetrics().density;
// create a focusable PopupWindow with the given layout and correct size
final PopupWindow pw = new PopupWindow(layout, (int)density*240, (int)density*285, true);
```

The button on the pop-up is given an `OnClickListener` so that the pop-up can be closed (if not familiar with coding event handlers see the article [Different Ways to Code Android Event Listeners](#)).

```
//Button to close the pop-up
((Button) layout.findViewById(R.id.close)).setOnClickListener(new View.OnClickListener() {
```

```
    public void onClick(View v) {  
        pw.dismiss();  
    }  
});
```

A touch event outside of the pop-up will also close it. The trick here is to call *setBackgroundDrawable* with a transparent drawable. This allows for screen presses outside of the pop-up to also close the pop-up:

```
//Set up touch closing outside of pop-up  
pw.setBackgroundDrawable(new ColorDrawable(android.graphics.Color.TRANSPARENT));  
pw.setTouchInterceptor(new OnTouchListener() {  
    public boolean onTouch(View v, MotionEvent event) {  
        if(event.getAction() == MotionEvent.ACTION_OUTSIDE) {  
            pw.dismiss();  
            return true;  
        }  
        return false;  
    }  
});  
pw.setOutsideTouchable(true);
```

Finally display the pop-up:

```
// display the pop-up in the center  
pw.showAtLocation(layout, Gravity.CENTER, 0, 0);
```

## Calling the Pop-up Function

The following listing is for a simple Activity with a button that has an *OnClickListener* to call the *showWinners()* function that is described above. The names of the medal winners and there scores are simple variables, in a real app game they would come from the game results. This is the complete code for the example in action shown at the beginning of the article:

```
package com.example.pop_up;
```



```
import android.content.Context;
import android.graphics.drawable.ColorDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.PopupWindow;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    String goldWinner="Fred Bloggs";
    int goldScore=98123;
    String silverWinner="John Doe";
    int silverScore=54900;
    String bronzeWinner="Joe Public";
    int bronzeScore=25789;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //attach an instance of HandleClick to the Button
        findViewById(R.id.button).setOnClickListener(new HandleClick());
    }
    private class HandleClick implements View.OnClickListener {
        public void onClick(View arg0) {
            showWinners();
        }
    }
    private void showWinners(){
        //We need to get the instance of the LayoutInflater, use the context of this activity
        LayoutInflater inflater = (LayoutInflater) MainActivity.this.getSystemService(Context.LAY
        //Inflate the view from a predefined XML layout (no need for root id, using entire layout
        View layout = inflater.inflate(R.layout.winners,null);
        //load results
```

```
((TextView)layout.findViewById(R.id.goldName)).setText(goldWinner);
((TextView)layout.findViewById(R.id.goldScore)).setText(Integer.toString(goldScore));
((TextView)layout.findViewById(R.id.silverName)).setText(silverWinner);
((TextView)layout.findViewById(R.id.silverScore)).setText(Integer.toString(silverScore));
((TextView)layout.findViewById(R.id.bronzeName)).setText(bronzeWinner);
((TextView)layout.findViewById(R.id.bronzeScore)).setText(Integer.toString(bronzeScore));
//Get the devices screen density to calculate correct pixel sizes
float density=MainActivity.this.getResources().getDisplayMetrics().density;
// create a focusable PopupWindow with the given layout and correct size
final PopupWindow pw = new PopupWindow(layout, (int)density*240, (int)density*285, true);
//Button to close the pop-up
((Button) layout.findViewById(R.id.close)).setOnClickListener(new View.OnClickListener()
    public void onClick(View v) {
        pw.dismiss();
    }
});
//Set up touch closing outside of pop-up
pw.setBackgroundDrawable(new ColorDrawable(android.graphics.Color.TRANSPARENT));
pw.setTouchInterceptor(new View.OnTouchListener() {
    public boolean onTouch(View v, MotionEvent event) {
        if(event.getAction() == MotionEvent.ACTION_OUTSIDE) {
            pw.dismiss();
            return true;
        }
        return false;
    }
});
pw.setOutsideTouchable(true);
// display the pop-up in the center
pw.showAtLocation(layout, Gravity.CENTER, 0, 0);
}
```

Bookmark this page in your browser so that you can use the code as a template for future projects that require a pop-up.

## See Also

- Download the project code for this pop-up example, available in [android-pop-up.zip](#) ready for importing into Android Studio
- For more android example projects see the [Android Example Projects](#) page.
- The medal images are also available via the [Free Launcher Icons, Menu Icons and Android Graphics](#) page.
- View the Tek Eye full [Index](#) for other articles.

Author: Daniel S. Fowler Published: 2013-02-05 Updated: 2017-07-23



---

[dan@tekeye.uk](#)

[Index](#)

[About](#)

[Contact](#)

[Policies](#)

© 2011-2020 Daniel S.  
Fowler

---