

# Floating widgets - Android

Floating widgets are the views that float over the screen. This view will remain floating on the screen no matter whichever screen you are in. They are very convenient for multitasking as a user can work on other application and control your application for the same time. That means if you are in the calculator application and a widget from the music player is floating over the screen, you can control your music at the same time.

Floating widget is nothing but the view that is drawn over other applications. Android system allows applications to draw over other application if the application has **android.permission.SYSTEM\_ALERT\_WINDOW** permission. We are going to use the background service to add the floating widget into the view hierarchy of the current screen. So, this floating view is always on top of application windows.

To drag the view across the screen we are going to override **OnTouchListener()** to listen to drag events and change the position of the view has in the screen.

Here are the steps to create the Floating Widgets:

1. Add **android.permission.SYSTEM\_ALERT\_WINDOW** permission to the **AndroidManifest.xml** file. This permission allows an app to create windows, shown on top of all other apps. We'll also add a service named **FloatingViewService** that will be created shortly.
2. Create a new layout file for the floating view. This layout will contain two main views. An example to this kind of layout file can be found at Appendix A.

**Collapsed view:** The floating widget will remain collapsed when the view is launched. When the user clicks on this view, an expanded view will open.

**Expanded View:** This view will contain buttons to play music, change the song to next/previous and open the application.

3. Now create a new service for that controls the logic of the widget. Whenever you want to display a floating view, start the service using **startService()** command. Just extend the Service class and override the **onCreate()** of the service, and there do the following:

- A. Inflate the xml layout file you created before.
- B. Create WindowManager.LayoutParams object with height and width WRAP\_CONTENT, type WindowManager.LayoutParams.TYPE\_APPLICATION\_OVERLAY for build version 26 and above and type WindowManager.LayoutParams.TYPE\_PHONE for version below 26 - this gives the ability to float over other apps, flag WindowManager.LayoutParams.FLAG\_NOT\_FOCUSABLE - so he wouldn't get the clicks and the last parameter pass PixelFormat.TRANSLUCENT for the transparency.
- C. Get the WindowManager instance with getSystemService(WINDOW\_SERVICE) and add the view you inflated in stage A with the params from stage B. Don't forget to remove the view from the window in your service onDestroy method.
- D. Here you can get a reference to the Layout's buttons and add OnClickListener to them and perform the necessary tasks for each click, note that if your layout contains the sub layouts for expanded and collapsed views first get a reference to the before getting a reference to the buttons.
- E. To drag the floating view along with the user's touch, we have to override **OnTouchListener() of the root layout we inflated before**. Whenever the user touches the root of the view, we will record the initial **x and y coordinates**, and when the user moves the finger, the application will calculate the **new X and Y** coordinate and move the float view by setting the layout params x and y and update the window manager instance with the updateViewLayout() function passing the inflated view and the new layout params. Save the original view x and y in action down and the initial touch x and y also, in ACTION\_MOVE calculate the new coordinates with the initial x and y plus the difference(the new touch - the original touch).
- F. Now we want to expand and collapse the menu. When user clicks on the image view of the collapsed layout, visibility of the collapsed layout should be changed to **View.GONE** and expanded view should become visible. To achieve this we need to implement OnClickListener() to the imageview of the collapsed layout. But, as we implemented OnTouchListener() to the root view, OnClickListener() won't work. So, to detect clicks we will detect clicks in **MotionEvent.ACTION\_UP** in the **OnTouchListener()**, by calculating the distance between the initial touch and the ACTION\_UP place, if the difference is less than 10 it means the we almost didn't move at all and it's probably a click.

The full Service code can be found in Appendix B

4. Now one final step is remaining is adding floating view by starting the **service**. For that, we need to check if the application has **android.permission.SYSTEM\_ALERT\_WINDOW** permission or not? For android version  $\leq$  API22, this permission is granted by default when declaring it in the manifest as we did before(step 1). But for the android versions running API  $>$  22, we need to check for the permission runtime. If the permission is not available, we will open permission management screen to allow the user to grant permission using **Settings.ACTION\_MANAGE\_OVERLAY\_PERMISSION** intent action. This will open below screen facilitate user to grant android.permission.SYSTEM\_ALERT\_WINDOW permission. You can use this code to do the work:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && !Settings.canDrawOverlays(this)) {  
    //If the draw over permission is not available open the settings screen  
    Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION, Uri.parse("package:" +  
        getPackageName()));  
    startActivity(intent);  
} else {  
    startService(new Intent(MainActivity.this, FloatingViewService.class));  
    finish();  
}
```

## Appendix A - Floating widget layout file

```
<FrameLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content">  
  
    <!--Root container-->  
    <RelativeLayout  
        android:id="@+id/root_container"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        tools:ignore="UselessParent">  
  
        <!--View while view is collapsed-->  
        <RelativeLayout  
            android:id="@+id/collapse_view"
```

```

        android:layout_width="wrap_content"
        android:visibility="visible"
        android:layout_height="wrap_content"
        android:orientation="vertical">

<!--Icon of floating widget -->
<ImageView
    android:id="@+id/collapsed_iv"
    android:layout_width="60dp"
    android:layout_height="60dp"
    android:layout_marginTop="8dp"
    android:src="@drawable/ic_android_circle"
    tools:ignore="ContentDescription"/>

<!--Close button-->
<ImageView
    android:id="@+id/close_btn"
    android:layout_width="20dp"
    android:layout_height="20dp"
    android:layout_marginLeft="40dp"
    android:src="@drawable/ic_close"
    tools:ignore="ContentDescription"/>
</RelativeLayout>

<!--View while view is expanded-->
<LinearLayout
    android:id="@+id/expanded_container"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#F8BBD0"
    android:visibility="gone"
    android:orientation="horizontal"
    android:padding="8dp">

<!--Album image for the song currently playing.-->
<ImageView
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:src="@drawable/music_player"
    tools:ignore="ContentDescription"/>

<!--Previous button-->
<ImageView

```

```

        android:id="@+id/prev_btn"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="20dp"
        android:src="@mipmap/ic_previous"
        tools:ignore="ContentDescription"/>

<!--Play button-->
<ImageView
    android:id="@+id/play_btn"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="10dp"
    android:src="@mipmap/ic_play"
    tools:ignore="ContentDescription"/>

<!--Next button-->
<ImageView
    android:id="@+id/next_btn"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_gravity="center_vertical"
    android:layout_marginLeft="10dp"
    android:src="@mipmap/ic_play_next"
    tools:ignore="ContentDescription"/>

<RelativeLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/close_button"
        android:layout_width="20dp"
        android:layout_height="20dp"
        android:src="@drawable/ic_close"/>

    <ImageView
        android:id="@+id/open_button"
        android:layout_width="20dp"
        android:layout_height="20dp"

```

```

        android:layout_alignParentBottom="true"
        android:src="@drawable/ic_open"/>
    </RelativeLayout>
</LinearLayout>
</RelativeLayout>
</FrameLayout>

```

## Appendix B - Floating widget service file

```

public class FloatingViewService extends Service {
    private WindowManager mWindowManager;
    private View mFloatingView;

    public FloatingViewService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        //Inflate the floating view layout we created
        mFloatingView = LayoutInflater.from(this).inflate(R.layout.layout_floating_widget, null);

        //Add the view to the window.
        final WindowManager.LayoutParams params = new WindowManager.LayoutParams(
            WindowManager.LayoutParams.WRAP_CONTENT,
            WindowManager.LayoutParams.WRAP_CONTENT,
            WindowManager.LayoutParams.TYPE_PHONE,
            WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE,
            PixelFormat.TRANSLUCENT);

        //Specify the view position
        params.gravity = Gravity.TOP | Gravity.LEFT;    //Initially view will be added to top-left
corner
        params.x = 0;
        params.y = 100;

        //Add the view to the window
    }
}

```

```

mWindowManager = (WindowManager) getSystemService(WINDOW_SERVICE);
mWindowManager.addView(mFloatingView, params);

//The root element of the collapsed view layout
final View collapsedView = mFloatingView.findViewById(R.id.collapse_view);
//The root element of the expanded view layout
final View expandedView = mFloatingView.findViewById(R.id.expanded_container);

//Set the close button
ImageView closeButtonCollapsed = (ImageView)
mFloatingView.findViewById(R.id.close_btn);
closeButtonCollapsed.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //close the service and remove the from from the window
        stopSelf();
    }
});

//Set the view while floating view is expanded.
//Set the play button.
ImageView playButton = (ImageView) mFloatingView.findViewById(R.id.play_btn);
playButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(FloatingViewService.this, "Playing the song.",
Toast.LENGTH_LONG).show();
    }
});

//Set the next button.
ImageView nextButton = (ImageView) mFloatingView.findViewById(R.id.next_btn);
nextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(FloatingViewService.this, "Playing next song.",
Toast.LENGTH_LONG).show();
    }
});

//Set the pause button.

```

```

    ImageView prevButton = (ImageView) mFloatingView.findViewById(R.id.prev_btn);
    prevButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(FloatingViewService.this, "Playing previous song.",
                Toast.LENGTH_LONG).show();
        }
    });

    //Set the close button
    ImageView closeButton = (ImageView) mFloatingView.findViewById(R.id.close_button);
    closeButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            collapsedView.setVisibility(View.VISIBLE);
            expandedView.setVisibility(View.GONE);
        }
    });

    //Open the application on thi button click
    ImageView openButton = (ImageView) mFloatingView.findViewById(R.id.open_button);
    openButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //Open the application click.
            Intent intent = new Intent(FloatingViewService.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);

            //close the service and remove view from the view hierarchy
            stopSelf();
        }
    });

    //Drag and move floating view using user's touch action.
    mFloatingView.findViewById(R.id.root_container).setOnTouchListener(new
    View.OnTouchListener() {
        private int initialX;
        private int initialY;
        private float initialTouchX;

```



```

private float initialTouchY;

@Override
public boolean onTouch(View v, MotionEvent event) {
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:

            //remember the initial position.
            initialX = params.x;
            initialY = params.y;

            //get the touch location
            initialTouchX = event.getRawX();
            initialTouchY = event.getRawY();
            return true;
        case MotionEvent.ACTION_UP:
            int Xdiff = (int) (event.getRawX() - initialTouchX);
            int Ydiff = (int) (event.getRawY() - initialTouchY);

            //The check for Xdiff <10 && YDiff< 10 because sometime elements moves a little
            while clicking.
            //So that is click event.
            if (Xdiff < 10 && Ydiff < 10) {
                if (isViewCollapsed()) {
                    //When user clicks on the image view of the collapsed layout,
                    //visibility of the collapsed layout will be changed to "View.GONE"
                    //and expanded view will become visible.
                    collapsedView.setVisibility(View.GONE);
                    expandedView.setVisibility(View.VISIBLE);
                }
            }
            return true;
        case MotionEvent.ACTION_MOVE:
            //Calculate the X and Y coordinates of the view.
            params.x = initialX + (int) (event.getRawX() - initialTouchX);
            params.y = initialY + (int) (event.getRawY() - initialTouchY);

            //Update the layout with new X & Y coordinate
            mWindowManager.updateViewLayout(mFloatingView, params);
            return true;
    }
}

```

```

        }
        return false;
    }
    });
}

/**
 * Detect if the floating view is collapsed or expanded.
 *
 * @return true if the floating view is collapsed.
 */
private boolean isViewCollapsed() {
    return mFloatingView == null ||
mFloatingView.findViewById(R.id.collapse_view).getVisibility() == View.VISIBLE;
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (mFloatingView != null) mWindowManager.removeView(mFloatingView);
}

```

## Reference

[Android Floating Widget like Facebook Chat Head](#) from androidhive