localazy.com

Floating Windows on Android: Tips & Tricks

5-7 minutes

Learn how to use floating windows in your Android apps. The last lesson teaches you a few tips and tricks I learned the hard way.

Have you ever wondered how to make those floating windows used by Facebook Heads and other apps? Have you ever wanted to use the same technology in your app? It's easy, and I will guide you through the whole process.

I'm the author of Floating Apps; the first app of its kind on Google Play and the most popular one with over 8 million downloads. After 6 years of the development of the app, I know a bit about it. It's

sometimes tricky, and I spent months reading documentation and Android source code and experimenting. I received feedback from tens of thousands of users and see various issues on different phones with different Android versions.

Here's what I learned along the way.

Before reading this article, it's recommended to go through Floating Windows on Android 9: Shortcomings.

In this article, I show you some tips & tricks I used in Floating Apps.

Minimize 🔗

In Floating Apps, it's possible to minimize windows. How do I achieve this effect?

Well, I move the window outside of the visible area of the screen. This is the safest thing you can do. Changing the window's size, removing the view from `WindowManager`, or any other similar action could break your app. For example, components rendering

content directly to the video memory such as `SurfaceView`, `VideoView`, etc. don't like it.

When the window is moved outside of the screen area, I inject a new view - the bubble. It's all done together with smooth animations, so it actually looks like the window is minimized to the bubble. But it's all just a smart effect.

Maximize 🔗

In Floating Apps, there is heavy math behind the window's size as there are different modes (with the title bar, without it, with minimal bar, etc.), half-screen size, etc.

But maximizing the window is not the case 😃. You can use `MATCH_PARENT` for `LayoutParams`, and it automatically sets the maximal available size. Simple.

```
params.gravity = Gravity.TOP or Gravity.LEFT
params.width =
ViewGroup.LayoutParams.MATCH_PARENT
```

```
params.height =
ViewGroup.LayoutParams.MATCH_PARENT
params.x = 0
params.y = 0
```

Don’t forget to remember the original position and size, and disable moving the maximized window.

Also, be sure that your layout is flexible enough to work correctly with the maximized window as well as with smaller ones.

Resizing Windows 🔗

This one is tricky. In Floating Apps, some windows can be resized, and such windows have a small handle in the right bottom corner.

Resizing the window is very similar to moving it, and you can use the same `DraggableTouchListener` as we introduced in the **Moving Window** article. Just change `x` and `y` for `width` and `height`.

I experimented with changing the window size directly, but for

windows with a complex layout, it's slow.

So my final version is: When the resize handle is touched, a new semi-transparent floating view is injected above the original window with the same size and position and it's resized instead of it. When resizing is finished, the new size is applied to the original window.

Transparency 🔗

`WindowManager.LayoutParams` comes with `alpha`, so this one is as simple as:

```
params.alpha = 0.5f
```

Don't allow the user to make the window completely invisible as it could have undesired effects. I bet you can imagine it.

FUN TIP: Completely invisible floating window is a nice prank! 😉.

Screen Rotation 🔗

In the foreground service, register the broadcast receiver to listen to

`Intent.ACTION_CONFIGURATION_CHANGED`, and you get notified when the screen is rotated.

In Floating Apps, when the screen orientation is changed, I keep the same size of the window and recalculate its position using the percentual calculation:

```
newX = oldX / oldScreenWidth * newScreenWidth
```

The window seems to be still in the same position relatively.

Bubble Physics 🔗

In Floating Apps, the bubbles for minimized windows are just specifically layouted views and nothing more. There is almost no difference from floating windows, as we discussed them in this series.

However, the bubbles have nice psychics when they’re moved around the screen. All the magic is created with the Facebook Rebound library - a java library that models spring dynamics.

Ooh, I just noticed that the library is archived. Pity. It’s very nice.

And We Are Done! 🔗

This is the last article in the series about the floating technology. I taught you almost everything I learned during the last few years.

Enjoy it and feel free to share your apps with me.

Also, I will be glad if you decide to localize your apps with Localazy. I put the very same love into it as I did with Floating Apps.

Thanks for reading! Follow me (@vaclavhodek) and Localazy (@localazy) on Twitter, or like Localazy on Facebook for more interesting information about Android development.

The Series 🔗

This article is part of the **Floating Windows on Android** series.

- Floating Windows on Android 1: Jetpack Compose & Room
- Floating Windows on Android 2: Foreground Service
- Floating Windows on Android 3: Permissions
- Floating Windows on Android 4: Floating Window

- Floating Windows on Android 5: Moving Window
- Floating Windows on Android 6: Keyboard Input
- Floating Windows on Android 7: Boot Receiver
- Floating Windows on Android 8: The Final App
- Floating Windows on Android 9: Shortcomings
- Floating Windows on Android 10: Tips & Tricks