Making Plasma Mobile highly configurable

The introduction of principal concepts, in particular the "Swipe to Resize Panels"

by Alex L.

Introduction

Plasma Desktop is an highly configurable desktop environment. It provides by default a simple configuration with a bottom *panel*, a *task manager*, a *start menu* and a *sys-tray* area. Thanks to this default configuration it's easy for Windows user to switch to Plasma. But user can configure Plasma a lot: for example he could add a panel on the top with *app-menu* like OS X. Or he could add a left-side panel with an *icon-only task manager* like Ubuntu Unity's one.

The mobile world have develop its own interaction language: Android have a well-known UI/UX and Google stress a lot its importance with Material Design. But there are also other actors: Jolla Sailfish OS and Windows Phone for example provide their own original UI/UX.

Plasma Mobile should have the same role of Plasma Desktop in PC's world: it should have by default a well-know behavior, so who knows Android does not have problems using Plasma Mobile as Windows users with default Plasma Desktop. But Plasma Mobile should also been highly configurable so that the user can recreate (partially or totally) Sailfish OS or Windows Phone experience on his/her device.

A first problem: integration between system and apps

In PCs' world, applications live in their windows and their system integration consists in communications between them and the system (i.e. notifications protocol). In mobile world things aren't so easy: system UI/UX necessarily affects apps UI/UX and vice versa. How Android solve this? It provide two always visible bars: notifications/status bar on top and system buttons on bottom. It's also a standard that Android's apps reserve to their selves the left and the right edges of the screen. How Plasma could

combine highly customization of UX with a contract with apps? Assigning bottom and top screen edges to system and left and right to apps is too much restrictive.

The solution: the S2RP and controls exported to plasmoids

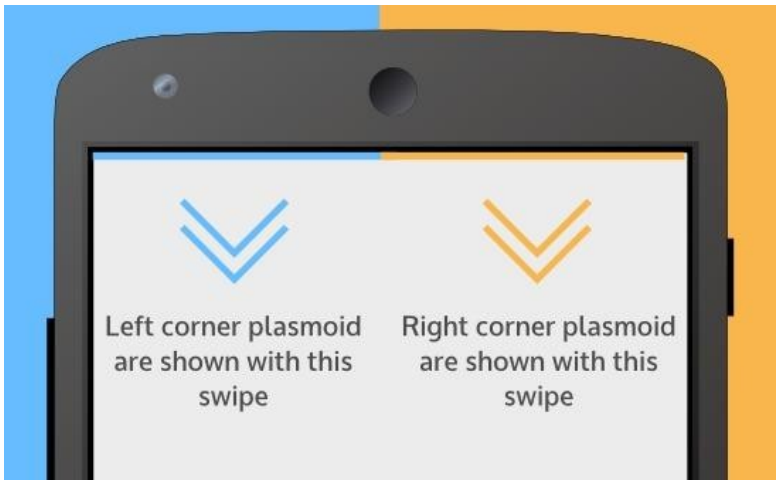
My proposal is to reserve all screen edges to the system, in particular to Plasma's panels. It's time to introduce the “swipe to resize panel” (S2RP) options for panels: the idea came to me trying to combine Android notifications bar with a Plasma panel. In fact a panel with S2RP enabled behaves like Android notifications bar: it's resized with a swipe. See the image below.

As you can see in the following mock-up, the small icons in the panel are resized together with the panel, so the user can switch between the tabs containing notifications, Internet and volume controls, etc.

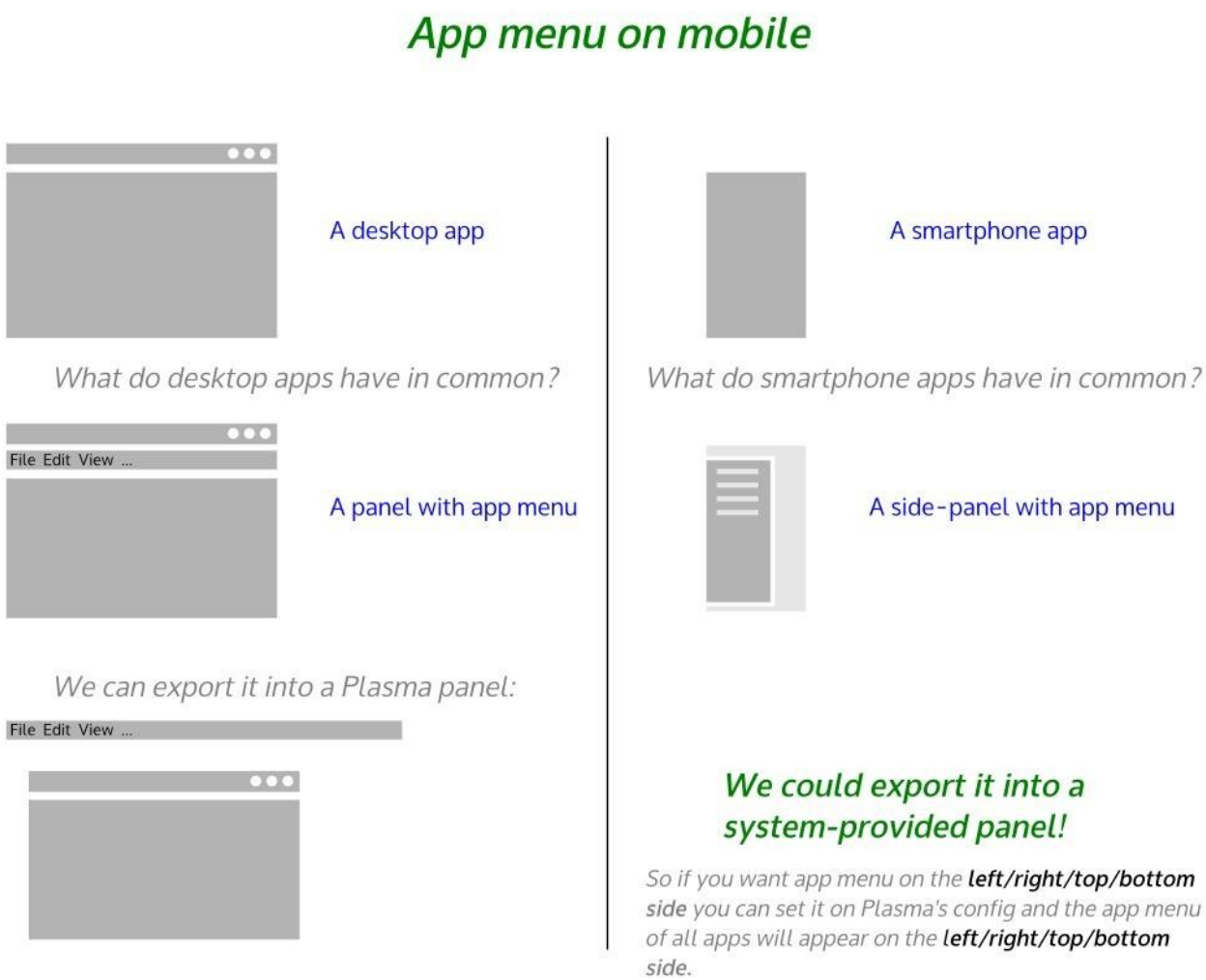
But the S2RP is not reserved to notifications panel: it could be use also for task manager as you can see in the mock-up below:

But the behavior of a panel with **task manager** plasmoid is different from the notifications one: the user resize the panel with a swipe but when the user release the his finger the panel stay at the size reached, so the user can tap on the icons, that were too much small when the panel was at its original size. Additionally, if the swipe exceeds a certain dimension, the panel show running apps preview, as you can see in the end of the mock-up. Finally the panel can be resized to its original dimension with a swipe down. In the mock-up the KDE icon show start-menu/app-drawer and Plasma icon show Plasma home (like Android's home button).

Additionally, if the user put a plasmoid in the left corner of the panel and an other plasmoid in the right corner, what the panel show when resized depends on where the swipe is made: if the swipe start in the left half of the panel plasmoid in left corner are shown, if the swipe start in the right half of the panel plasmoid in right corner are shown.



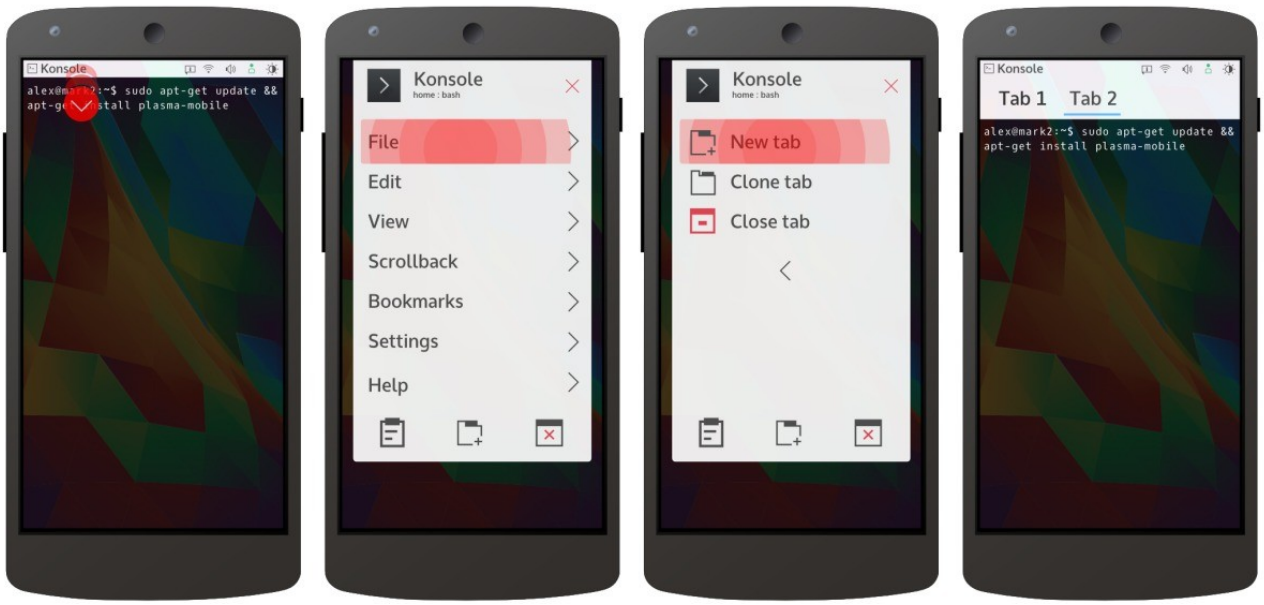
But with this concept, how can apps, for example, draw a side app-menu like Android's one? All screen edges are reserved to system. So we need to export something from app to system. The following mock-up show how this is made on Plasma Desktop and how it could be made on Plasma Mobile.



Example:

In this example left-side and right-side panels have “auto-hide” option enabled: they are hidden but a swipe from the edge of the screen show them.

Additionally, this could be a good way to provide a touch-oriented app-menu for legacy desktop app. In the following mock-up you can see Konsole and its classical app-menu with a touch-oriented aspect. The app-menu is a plasmoid added in the left corner of the top panel and is shown with a swipe in the left half of the panel.



In the mock-up there is also a close button on top and actions shortcuts in the bottom of the expanded panel.

This approach could be convergence-ready: if the user plug his smartphone to monitor and mouse, the app-menu plasmoid could be expanded on the panel to appear like classical desktop app-menu (like OS X's one), similar to menu of responsible web pages.

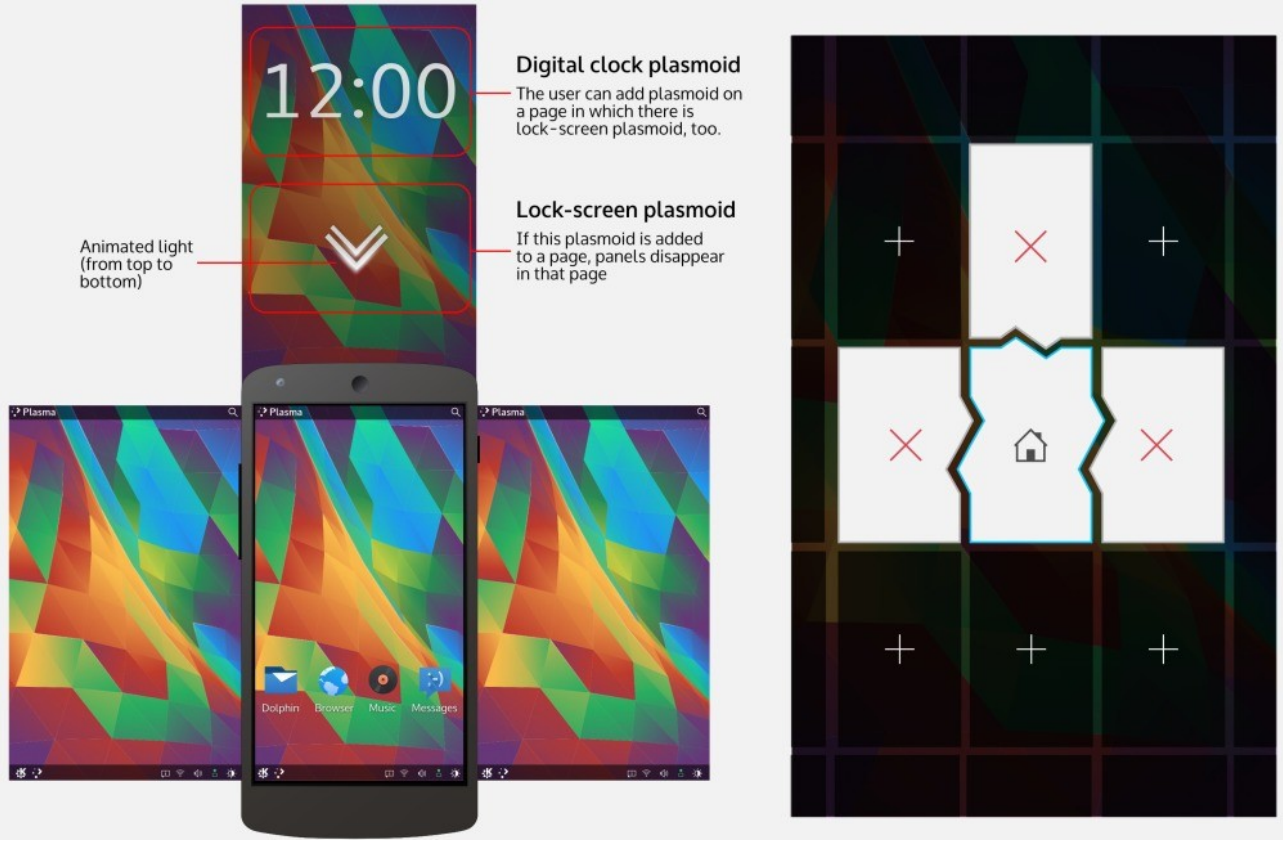
Conclusion: the swipes that start from the screen edges are reserved to Plasma panels and other swipes are reserved to the current app (for example to switch between tabs).

Plasma Mobile pages

Basically in mobile world there are three approaches to "home": horizontal pages switchable with horizontal swipes (Android approach), vertical scroll of elements (Windows Phone approach) and complex pages switchable with four-directions swipes (Sailfish approach).

My proposal for Plasma Mobile is to combine these three approach: a grid of pages switchable with swipes but customizable by the user and an optional container plasmoid (in which user can add plasmoids) that provides vertical scroll.

In the following mock-up you can see a simple configuration of the pages: three pages similar to Android home and a lock-screen page. Aside you can see the how the user can add/remove/move pages and able/disable links between them.



The user could also link a page to an app that will be start when the user switch to its page: for example a dialer page, or a camera page aside the lock-screen. Obviously when an app are started the swipe is reserved to it and the user can't come back with a swipe as in standard pages.

Vertical-scroll container plasmoid

As I said, the vertical scroll is provided by an optional container plasmoid: the user can add a vertical-scroll container plasmoid and the he can add in it all plasmoids he wants.

Example configurations