A user-centric, pallet based UI for Koffice

In this document I will describe an interface that borrows from but does not follow the typical interface as popularized by the macintosh. The interface will retain enough gestures and visual elements from the old-style interface to allow users to rapidly assimilate the new design. The design I am proposing would necessitate a large number of changes to cascade through other applications within the KDE domain.

A few statements to guide the design:

- The classic menubar-along-the-top design allows faster movement of the cursor by allowing the user to "slap" it against the side of the workspace. This is a plus for usability but ignores that larger screens demand a larger initial movement of the cursor. Fixed-in-place menubars do not adapt to user preferences or or technological change, they ignore their relationship to the workspace's overall structure.
- Replicating a menubar on every window avoids large cursor movements but at the expense of considerable workspace usage and unnecessary visual complexity.
- Most applications share many of their menus and toolbars. This is a boon that should be encouraged.
- Users are most likely to be at their most productive if an application can share parts of its files seamlessly with other applications. Kparts is an integral and necessary though implicit part of my proposal.
- Users will be at their most productive if an application can share parts and abilities seamlessly with other applications.
- The separation between text menus and widget/tool pallets is artificial and unnecessary. There is no reason to force different behaviour on these to UI elements.
- Windows are treated as resize-able pallets.
- The use of microsoftian toolbars across the screen is a waste of space, unnecessary, and abetted by the use of inscrutable icons.
- Most "wizards" are an effort to compensate for bad design.
- Increasingly large screens allow for increasingly humane interfaces to be used.
- A user's actions should never be overridden by the software unless an agreement has been made via preferences.
- The software should always assume that the user is doing what she wants and not interfere (subject to the statement above). One implication of this is that an explicit "save" is unnecessary.
- Functions like “name” and “save” should be kept separate.

- The software should always allow the user to recover from mistakes or a change of mind. Previous file states must be retained.
- Dialogs that offer only a statement and an "OK" button are of little use and only distract the user from his task. Non-modal means of informing the user are always preferable.
- When possible, widgets and other control features should be kept to a minimum. The increased complexity may offer more flexibility but the added complexity is rarely worth it. When possible, widgets should be replaced with gestures or inherent behaviour.

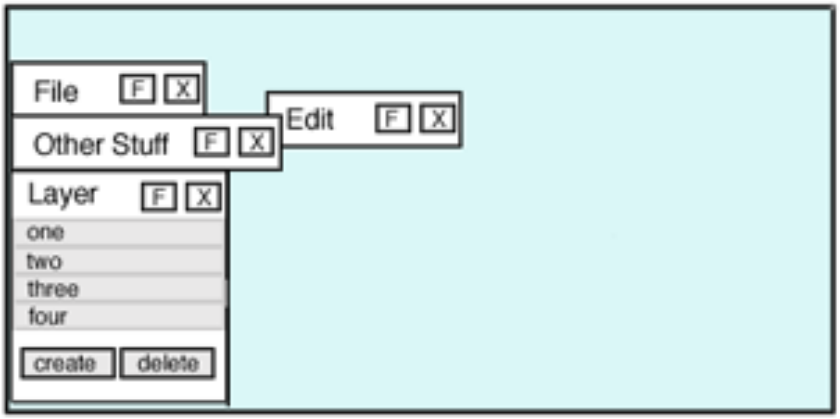
A statement about marketing.

Everybody seems to want to do things the same way. Everybody's UI looks like everybody else's. If Koffice is to escape the shadow of OpenOffice's long shadow then it must find a better way to get the work done and also find its own visual identity. OpenOffice's slavish emulation of Microsoft Office's UI is its greatest weakness. Yes, it allows for easier movement from MS apps but it also repeats the many errors that MS Office inflicts on its users. Many of the "features" of MSO are the product of bad UI. As much as possible, Koffice's abilities should be implicit through the flexibility and intelligence of tools rather than explicit through the addition of yet another feature.

KDE's compact, well-factored codebase gives the opportunity for KDE to offer a much more modern, user-centric and humane interface than its rivals.

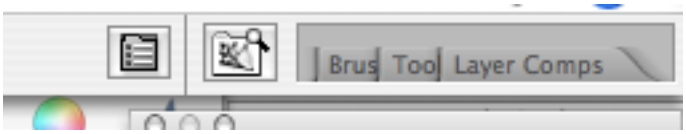
Menubars and pallets

- A menubar menu can be viewed as a special type of pallet: it uses all text, is fixed in location and is normally in a closed state. If a menubar is not treated as a special case then the menu bar menus can be treated as a set of discrete elements. Discrete elements should be moveable and placeable by the user.
- The above implies that there must be a pallet-of-pallets. This in turn implies that there is one pallet that cannot be removed from the screen. To reduce confusion, user must have a single point of "entry" that is always available. The pallet-of-pallets is it.



Example of pallet layout. Pallet-of-pallets not shown.

- A general proof of the above statements can be seen in two common UI practices. One is, for example, Adobe's use of a pallet "bin" that is adjacent with the menu bar. A second is the common "Window" menu item that activates or hides specific pallets.



Adobe's pallet "bin."

- Each pallet should be tightly focused by function; no "grab-bag" pallets allowed!
- Tabs within a pallet are exceptable when pallet size becomes problematic. It is usually better to split the pallet.
- Pallets should be atomic and not provide for large changes of their individual UI. If users are frequently requesting the ability to change pallet choices or structure it is a good indication that the pallet(s) is poorly designed.
- Features or preferences that are infrequently used can be contained in a pallet-attached sub-menu or independently expandable sections. If the use of such UI features by the designers becomes too frequent then it is an indication that the pallet(s) need redesigning.

Movements and magnetism

- Since pallets can be moved by the user they can, of course, also be arranged by the user.
- A pallet should be grab-able from any surface that is not spatially close to an active surface such as a widget. The user should be able to grab and move a pallet at any time without restriction.
- Pallets should be "magnetic" to ease their arrangement.
- Side by side pallets should arrange so that the tops are aligned. Pallets that are connected above and below should arrange so that their edges align as well.
- If a pallet is minimized to its name only, then other pallets should move upward to maintain connectedness. Pallets should also be attracted to the edge of the workspace.
- Pallets can be dragged so that most of their surface of outside the visible workspace.
- If a pallet is tugged with a short, sharp gesture then it should disconnect from its magnetic siblings, if any.
- If the disconnected pallet is in the middle of a group then the other members should shuffle to reconnect. Shuffle direction should be towards the edges of the workspace.
- If a pallet is grabbed and "thrown" it should move to the workspace edge to which it has been aimed. If other pallets are present along the same workspace edge then the newly placed pallet should connect with the others (along the edge or, if full, along the side of the pallet group).

Revelation of UI and its hiding

- Double clicking the name bar should hide or reveal the pallet (except the name bar which should always be visible).
- All pallets except the pallet-of-pallets should have a "destroy" widget placed in the pallet's name bar.
- Destruction of a window-pallet triggers a save of any contained work.
- Pallets or windows that have been minimized should stay in place. No elements should move on their own, except for the previously mentioned “shuffling.”

What comes first and does it always?

- Since windows are pallets by another name, pallets and windows share precedence and focus rules; they share the same stack of precedence.
- A widget in the name bar to selectively allow pallets to float above other elements should be included on all pallets and windows except the pallet-of-pallets.
- If a pallet that is floated is magnetically connected to other pallets then the connected pallets float too. If one of the connected pallets is removed from the group then its float status should remain the same.
- All pallets should allow minimization by double clicking on the name bars.
- A pallet or window that has been floated to the top can lose focus despite being on top.
- The pallet-of-pallets is the one exception to the above rule. It should always be available on top but otherwise follow focus rules.
- Re-selection of a pallet from the pallet-of-pallets should move that pallet to the front and give it the focus.
- A submenu should be provided on the pallet-of-pallets for open applications to move associated pallets to the top.
- Use of the float widget, minimization or maximization, or movement of a pallet should not change focus.

non-modalism saves history

- The insistence of modern UIs for explicit document saves is unnecessary.
- Saves of a document should be automatic.
- Since pallets are persistent there is no need for an “OK” button. A manipulation of the pallet’s controls is accepted without need for confirmation.

- Undo should be available back to at least to the most recent opening of the document.
- Undos to the previous state of closed documents should also be available.
- A subversion-like system should be present but hidden to facilitate the above undos and to allow easier shared creation of documents.

Conclusion

I suspect that my proposal is more far reaching than what you hoped for. However, with consideration, you will find that it is consistent, modern, and disposes much of the cruft once necessitated by much less powerful computers than we have today.

The proposal does not provide specific recommendations or images of specific pallets. I chose to not do so because of the large amount of redesign and re-factoring of features that would be necessary. Authors of individual software would need to consider the current and future plans of their project. The UI is one that would, admittedly, require more time and version releases than you may wish to contemplate. However, it is a journey that will pay considerable dividends in the future and should be evaluated on that basis.

The pallet specific UI will allow KDE to differentiate itself from competitors while providing a superior interface for users.