GUI and Functionality Design Competition for KOffice 2

by Martin Pfeiffer, hubipete@gmx.net

Foreword

KOffice is a relative small office suite and is not as powerful as Microsoft Office or Open Office. But this makes KOffice very fast and handy for daily usage which don't need special features. As KOffice won't gain as much features as the big suites till the 2.0 release it is important that KOffice provides an innovative and perfect user experience. Therefore I would like to see my sometimes adventurous ideas become true in KOffice 2.
I have analysed current problems and I took also Microsoft Office's new GUI concept into mind. The resulted concept can be seen as one complete solution but I paid attention that the workflow doesn't suffer if one feature is decided to be not realised.
A last thing: forgive my bad english I hope you'll understand all explanations anyway.
Have fun...

General Concepts

This chapter is about the general concepts and ideas I had in mind while designing the new GUI. They all make the GUI more efficient, handy and faster.

Back to the roots

I started with a return to the origines of the wordprocessor: the desk with a sheet of paper and a pen. We always put the sheet of paper in the center, not on the right site nor on the left side so a natural GUI shood display the edited document in the center and by default in a size that the user sees the whole page. This means we try to rebuild or model our daily office live in the computer and therefore we want to have a GUI that is as "natural" as possible.

Reunion of property and object

When you change the size of an image there are several ways to do it: Directly into the GUI with the mouse and in a dialogue using some spinboxes. When editing the size in the GUI you have no feedback on the exact value but in the dialogue you have no feedback on the look of your change. Why don't we display these values near the image while resizing it?
The reason is that old computers weren't able to handle those things in realtime. This lead to the separation of property and object and the creation of the eval dialogues.
Today the computers are able to display complex graphics so it is also possible to display context informations for a selected object. So lets bring the most useful properties to the surface and stop hiding them into compley dialogues.
This leads me to an other thing I learnd from the old natural desktop: If tools are too far away we don't use them. And three clicks to see the image's coordinates are way to much. This three clicks include the assumption that you know where to find them.
Learned mantra: Show as much as you can directly in the GUI and avoid long "distances".

Reuse

In the software development sector reuse of existing code is a common paradigm. In actual office suites this aspect is underrated, KOffice4 should change this.
Do you know those annoying tasks during the creation of a document? When you know exactly that you had the slide's layout 2 weeks ago in an other presentation but can't use it again. Therefore KOffice should hold those layouts so that we can easily and fast reuse them.

Separation of layout, content and structure

A mantra realised in HTML and CSS is the separation of layout, content and structure. CSS takes care of all aspects related to the layout of a website. (X)HTML defines the structure and the content is often generated dynamically via CGI etc.
Such a separation would be useful for an office document, too. Especially with reuse in mind those components could be handeled separatelly. While writing a thesis the layout is not important but the content thus the text. As your thesis needs to be well structured, there should be tools to quick outline a structure or organize existing content in a certain manner. But we should not forget that also structure and layout are related closely for example a table of contents has a different layout than pure text.

Layout

The document's layout is very important and there is already a lot of functionality built in current office applications also in KOffice. A layout can be very complex and this is the case where it must be easy to create it. Easy and fast this are the main points for layouting. I already mentioned the reuse of existing components in office suites. So if you need a common layout KOffice should it make very easy to adept it for a new text. Not only layouts the user created should be available, but also there should be a lot of patterns to use.
Next point is the speed which is generally dependend on how fast and efficient you can interact with the GUI. Fast means very few clicks to find and change or apply a feature to the document.
One thing I neglected so far concerning the layout is layout management. Imagine you are writing a 100 pages document for your thesis and you need common layout for it. What is needed is functionality to manage a general layout.

Sharing features and integration

Sharing features between applications is useful for the developers because they don't have the wheel invented twice. But did you thought of the advantages for the users: the features are the same so the usage is the same, no need to learn something again. Properties applied once don't get lost when transferring an object between applications etc...

A second point linkes closely to shared features is the integration of office suites. Working for a big thesis often implies a presentation, a text document, charts, flowcharts and drawings. All these are handled by different applications. It is annoying to change continuosly the window and the save-open-insert-...-method costs a lot of time. Ergo an integration of all components under one hood is desirable and makes feature sharing even easier.

Conclusion

All these thoughts and ideas have one common goal: Reduce the users effort while creating a document. The user should only enter the data e.g. the text and define a layout and a structure. He should concentrate on the things that matter.

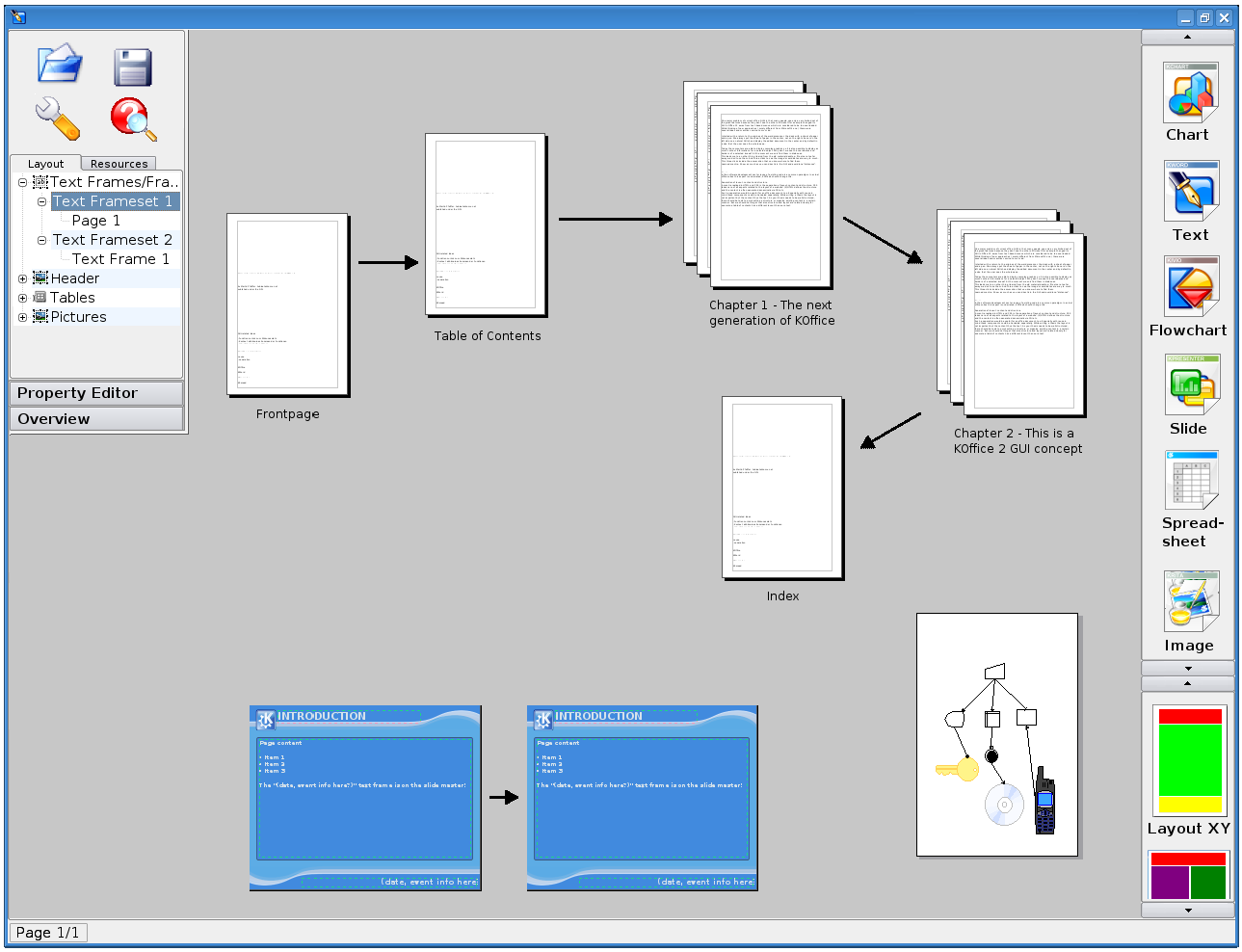
KOffice 2

The new GUI

The general design

The most important aspect of my new design for KOffice 2 is the unification of all components into one big application. This is how it is presented to the user who only has to start one application to access all components. This is a very radical step but it makes KOffice a real integrated office suite.
Bottom-up this decision I want to introduce now some new workflows, GUI elements and features. Let's start with the first impression the user gains after starting KOffice. He will be watching an empty surface which is based on a material desktop. On this surface the user will later organize and browse the documents he's working on. By drag and drop the user will be able to move the documents around and group them. Via arrows between documents of the same type they can be put together. Droping a chart on a text document will insert it. But most frequently a document will be opened in order to edit it that means the user has to double click on a thumbnail "laying" on the "desktop".
This workflow shows that I want to facilitate the creation of big documents. KOffice would provide the functionality to manage larger projects. And as a consequence a single document e.g. a chart or a spreadsheet is not longer bound to a special application nor is it in focus all the time. It becomes just a part of a larger project but this does not mean that KOffice could only create large projects the common single-document-usage is still possible. The user's benefit of these changes is obvious: All those create-save-open-...-orgies are a thing of the past, all this window changing and complicated inserting of document into document is omitted.
The document itsself becomes a composition of various possible ingredients like text frames, pictures, charts, spreadsheets, slides of a presentation. This are all potential parts of one page of a document and therefore it is good to imagine that such a document is made of several layers for every type of ingredient. To edit the ingredient the users clicks the object on the canvas and this invokes the functionality for the particular type of object. There is no change in the GUI in general. But there is also the possibility to edit the inserted object directly as it appears as document on the "desktop".
Let's face the most common case that the user edits a special document. There are three things the user want to do: Enter his own content, insert objects to the document and specify a layout.
The only point of the three I listed above that can't be simplified is the input of content. But the other two can be simplified.
Currently you have to click around a lot to insert an image for example. With the new sidebar widget this is done within a second by drag and drop.
The next point is much more complicated: the layout. We find a lot of layout functionality in current office suites but I think it is still not easy enough. KOffice provides the concept of frames which is nice but not mellowed and exhausted yet. Layouting is especially important for text documents as they are printed so I will concentrate on them.
What does the user need to layout fast and efficient? First of all he needs a lot of patterns he can use. With patterns I don't mean only complete layouts but things like different headers, different footers, different page numbers, different content layouts etc... . These parts could be made easily combinable and we had a nice layout construction kit. The user would choose header A and footer B drag them from the sidebar onto the document and his work would be nearly done.
Using patterns to layout is an easy and fast way but not as flexible and often there is the need of adapt something. Thus KOffice should provide a layout management system which highlights the single objects and lets the user define its own layout by draging the objects around. This layout system is something we already have, using frames the user can create nearly every layout. But this system can be enhanced.
Problems occur when the user has created a longer document and then changes the layout a bit what causes a translation of some objects. So the fine adjusted layout is destroyed very quickly. Thus I of gluing which means to glue several frames together in order to create a union that does not change when the layout changes elsewhere. This gluing can be made very complex with roles that determine how to resize a special object.
Furthermore the user should not know that he is dealing with frames. He only should know that he can drag everything where he wants. So at the beginning he is acting on a plain surface he puts objects on. These objects embed into the surface and suppress the old area. Then he defines the glue points this means the points where the to areas are hold together. Moving the vertices of an area changes its form. There should be the possibility to define round areas etc. If this areas are treated as frames internally is an other question.
A further aspect I want to be known is the linkage between the single applications of KOffice. I dream of an environment where I see live changes, this means a slide which is inserted into a text document changes live when the slide itself is edited by the user. I think the combination and the team work of all components would make KOffice really powerful. An other thinkable feature is a marker function for text documents. The user would have a tool to highlight paragraphs like a text marker in the reality. This digital highlighting would

result into a list of headwords that can be used to fill a presentation's slide. The user does not have to copy and paste so much.
The last point I want to mention here is that I propose to ban the large spreadsheets. Who needs these thousands of table cells? I made the experience it is more useful to have some small sheets with each one containing a table than to have all tables on one sheet. So a spreadsheet will grow with the needed space. This means the user can define how many rows and columns he needs. This makes the sheets very handy.



A mockup of the „desktop“ with some documents open

A menu of icons

One major problem of current office GUI's is that many people use only a very limited set of the given features because they don't see the others. Microsoft for example changed its GUI in Office 12 away from text-based menus which are considered to be too overloaded. This text-based menus are a relict of the old days when computers weren't able to render complex graphics. So today they are out-dated.
In my mockup there is no textbased menu and there are very few icons. I think icons are as difficult to use as a text-based menu. You always have to learn what an icon means when you use an application for the first time. So I banned them all except the "Open", "Save", "Help" and "Settings" icon which are so well-known that everybody can use them immediately. They represent functionality which can't be placed elsewhere.

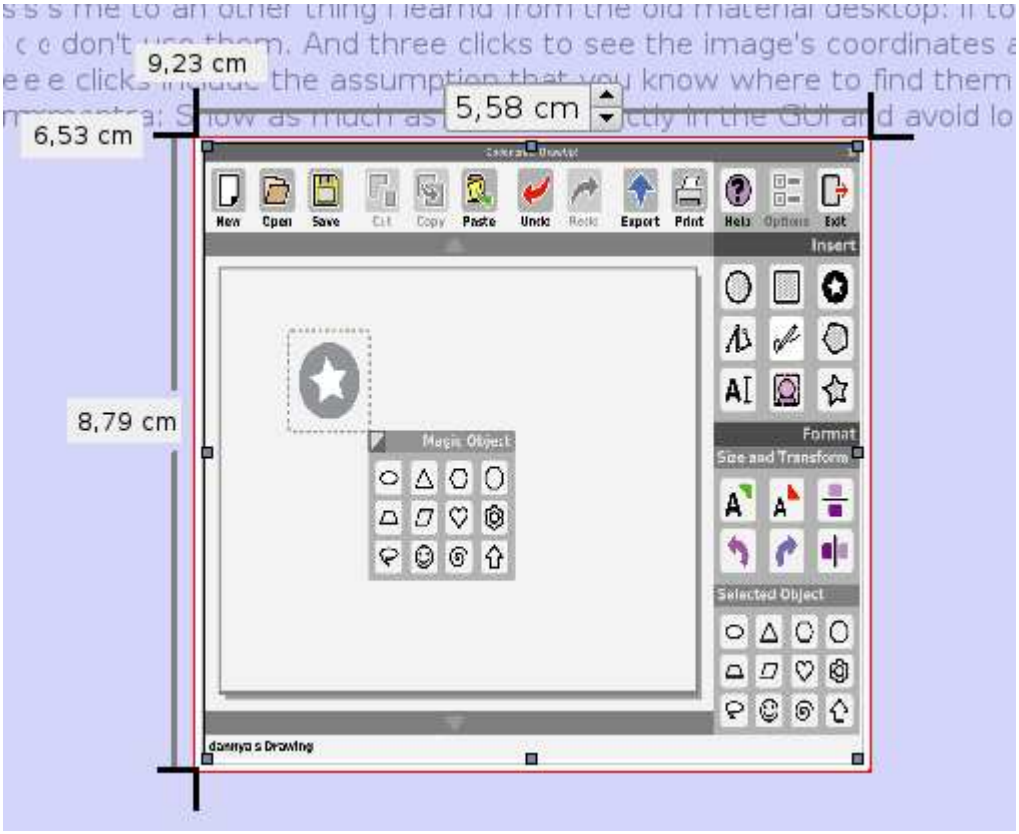
The visual scrollbar

Though there is the possibility to split long documents into several small parts a document may be longer than one page. Therefore we need the scrollbars to walk the document.

We all know the common scrollbar which is part of nearly every application GUI. But in my opinion it is antiquated in matters of the visual appearance.
When computers had not as much power as today it was only reasonable to only display a "button" which is moved up and down. But today there a ways more possibilities and we know an interesting way of navigation: In KPDF there is a sidebar which displays a thumbnail for every page of the opened document. Combining this widget with a normal scrollbar results in the visual scrollbar.
This one has several advantages:
You see the next page before you "open" it, so you know where you navigate to. You can click a page in order to bring it on the screen, which means that you can very fast switch in a range of 5 consecutive pages. There are some nice effects possible and eyecandy is increased.
In my mockup the visual scrollbar is visible although it is thought to be invisible as long as the mouse is on the document. There are arrow buttons to go up and down by click but similar to the Xbars moving the mouse fast up will also scroll the document.
I think of the visual scrollbar as the logical advancement of the "old" common scrollbar.

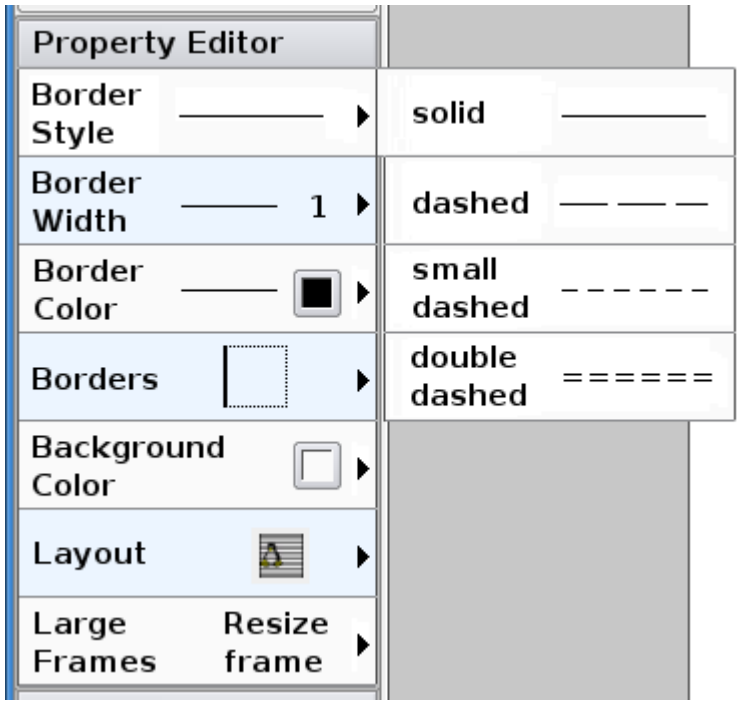
Local property changing

A problem that every office suite has to face is to find a way to provide access toall properties of a selected object. This was solved by separating the object and its properties and putting the latter ones into dialogues. This caused several discomforts: You have to find the location where you can change the particular property. You always have to navigate to the place to change it even if you learned the location where to find the property. While changing the property in a dialogue you have no visual feedback what your changes caused. Closing the dialogue to see the changes or press the "Apply"-button is circuitous.
This was also something Microsoft remarked and thus changed its GUI for Office 12.
My solution is splitted into two parts: Displaying properties near the object and the property manager widget.
Following the paradigms I wrote above it would be the best to reduce the way for the mouse and embed the functionality in the GUI. So the logical solution is to display the icons near the selected object directly on the document's surface. When changing a property the user gets promptly a visual feedback and the way for the mouse was reduced as much as possible.



The property manager

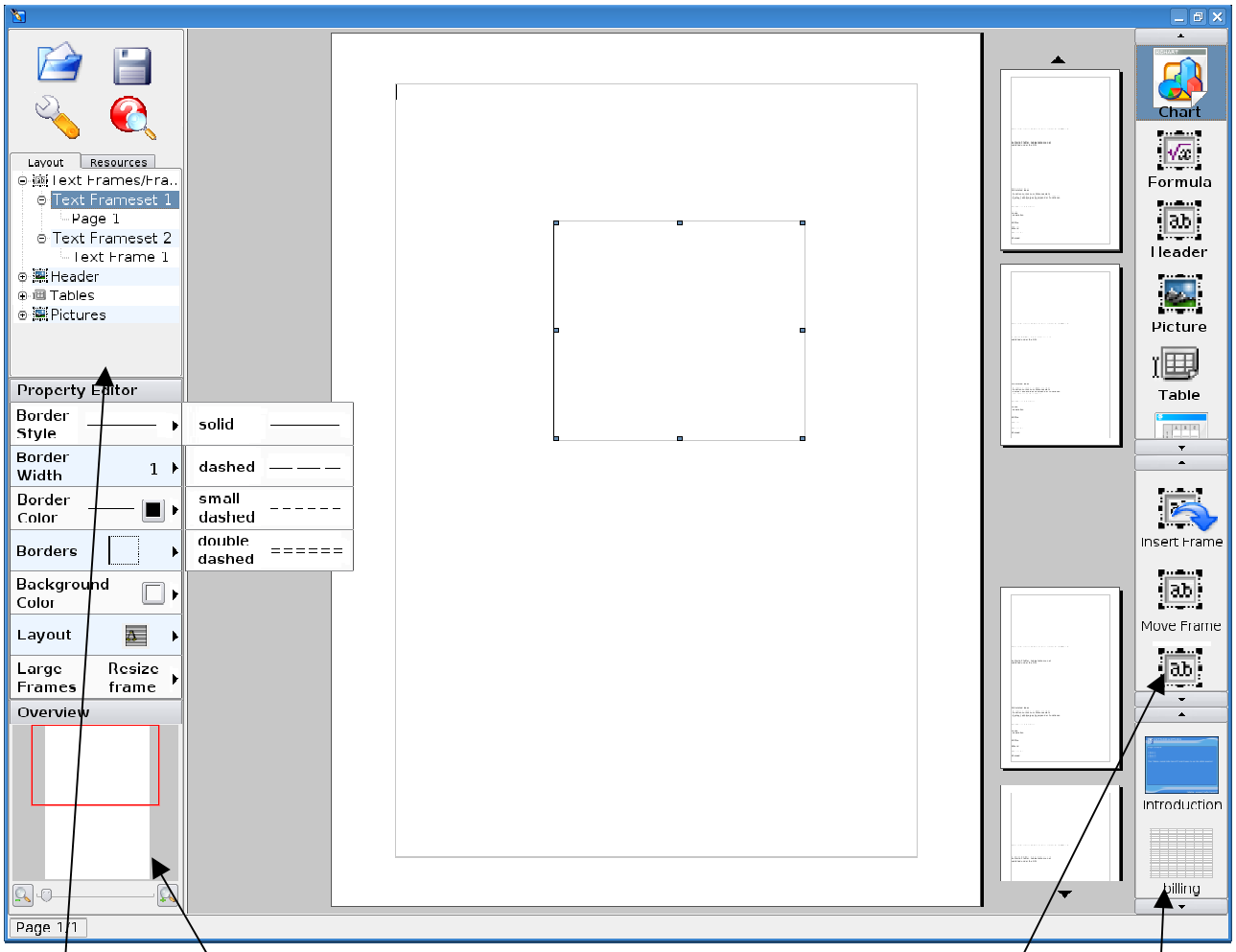
But not every property can be displayed as icon beside the selected object. This would destroy the favourable aspect of a slick assortment of options.
But the solution is already invented so I only altered it a bit. The property manager is a widget based on the one e.g. QtDesigner is using. I made the rows larger and added meaningful icons. Now it is possible to display all options that are not directly put near the object in a table. The advantages against a dialogue are obvious: You know the place on the screen were you have to look for the possibilities. You have a fast overview of all things you can do. And for both not a single click is necessary.



Xbars - Extending scrollable bars

The last new widget I want to introduce are the Xbars. Xbars stands for extending scrollable bars and these are my solution for a container widget.

In the current version of KWord you need to navigate in the menu and do 2 clicks to insert a simple image. A lot of action for a simple action so wouldn't it be cool to see all the time the things you can insert, to insert an object with a single click or by drag and drop? It is certainly cool and the extending scrollable bar is my solution. You might think that it is a simple container like all the others but you're wrong.
The advantage of the Xbar over the other containers is its self-organising ability. Docked to one side of the application's window the Xbar has to share the space with other Xbars. So every docked Xbar is decreased in size and finally all bars are visible. But when entering the bar with the mouse it extends itself so that most of it's children are visible. Leaving the bar makes it shrink back to its standard size. This behaviour is a nice advantage since you don't need size the containers one by one to fit the needs. Moreover the Xbars provide enough space for a nice big icon with describing text. So they are perfectly suited to contain the objects which can be inserted.



KOffice editing a document

A simple treeview that list all elements

A common overview widget with zoom ability

The undo-history

Other documents that are open – ready to include

Internal structures

This section consists of some thoughts about KOffice's internal document treatment and application structures.

Making KOffice one big application has not only usage advantages but also regarding the source code there can be improved and changed a lot. The keyword is "unify" so a lot of parts are put together.

Document handling...

KOffice has quiet nicely adapted the OASIS format. KSpread, KWord and KPresenter support this format natively, which means that all the contents are saved to XML. Internally all three hold a set of classes describing the document. But why not holding all document informations directly in XML already during runtime? There are appropriate classes for dealing with XML documents, like QT's QDomDocument. So a KWord document would be represented internally as a QDomDocument whos child nodes are changed. Saving would be very easy and the way KOffice deals with documents internally would be unified.

...and canvas handling

Apart from holding the document's information the KOffice apps also have to display the edited document. Therefore KSpread, KPresenter and KWord have all three different canvas implementations. But especially between the requirements of the KWord's and KPresenter's canvas there are many similiarities. This means that code sharing in a lib is possible. Beside canvas elementes like shapes, arrows and included pictures for example KSpread could share table functionalities. This means that there is no need for 3 different canvas implementations.
To continue the thought: What does a canvas do? It translates the document information hold in the application into a viewable representation on the screen. There are multiple ways to realise a canvas: OpenGL, SVG and the commonly used QPixmap-based implementation.

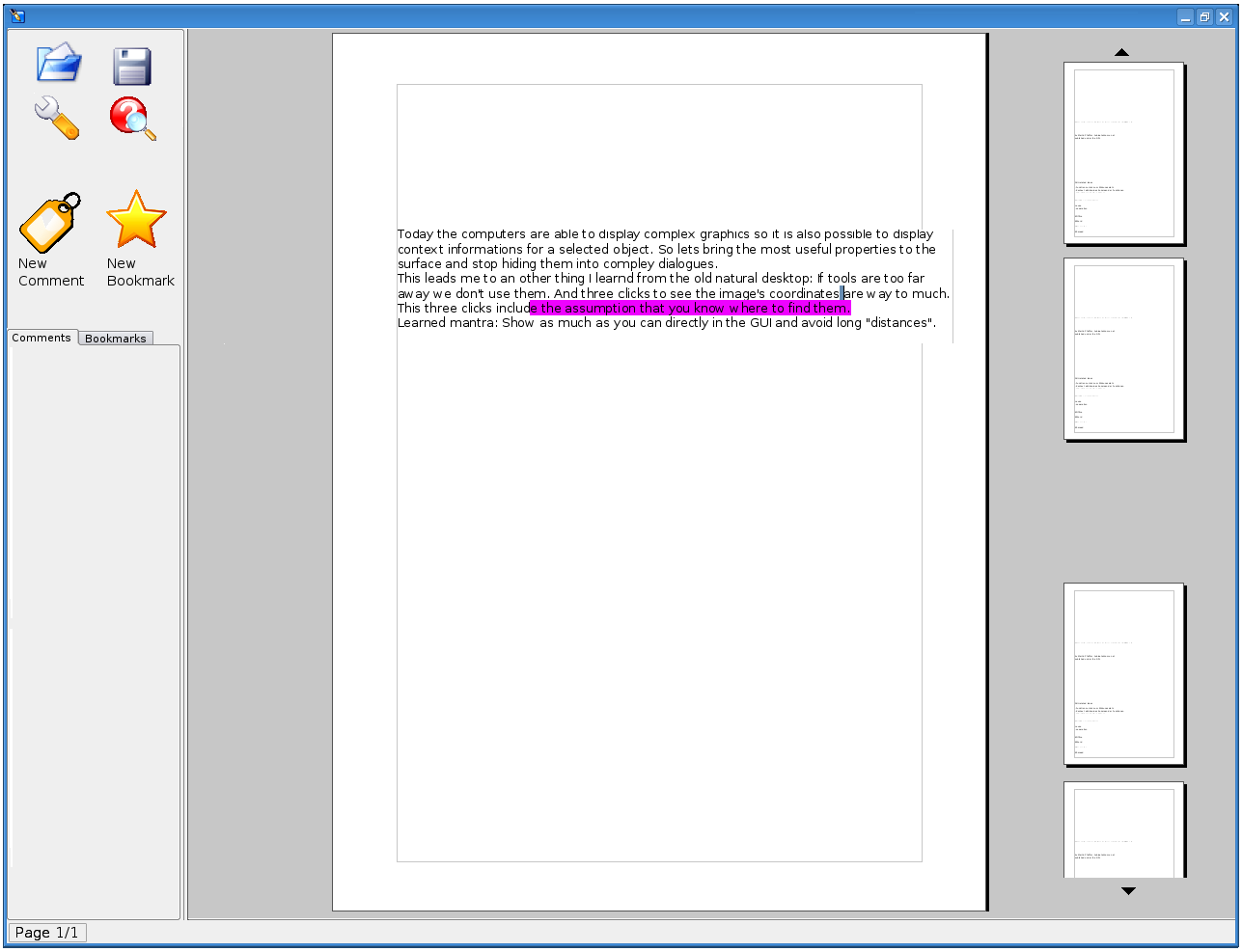
Now we bring both thoughts together and we get a nice application model. At top there is the KOffice application which is the main application, doing all load, save and config things. KOffice application holds multiple KOffice documents. These are XML based in the cases of KWord, KPresenter and KSpread. Then there are multiple viewers for the existing documents. These do not more than simply "translate" XML into canvas elements. Imagine the possibility to implement various viewers e.g. a "normal" QPixmap-based one, an OpenGL based and one which uses SVG and KSvg or QT's Svg implementation to display contents.
In fact such a code model would realise real document-model-concepts and would allow a high code sharing. This is something (the code sharing) KOffice misses at the moment.

Single ideas and improvements

KWord

Review mode

Often you don't have to create a document but you must read it and perhaps you would like to mark or comment special sections. Thus you need a marker-function and a good comment and bookmarking ability. KOffice could support a reviewing mode which does not allow any editing but provides a bookmark and comment manager along with the ability to highlight text like you do it with a peace of paper and a marker.



KPresenter

File card support

Often the presentator uses some file cards for his notes which are useful during the presentation. Therefore KPresenter has already a notes window. But it is not nicely integrated and not as useful as it could be.
I propose to implement first an incomplex way to "export" those notes to a KWord document. And as a supplementary feature I would like to see some nice preformated printing options, so that these cards can be printed directly out of KPresenter and then cuted with a pair of scissors.

Animation recorder

There are applications called makro recorder which provide a way to create a makro by clicking the functions in the right progression. For KPresenter I would like to realize the animations like this. In addition to the prefabricated animations the user would have the possibility to define a sequence of things that would be "played" as an animation during the presentation. Thinkable are for example moves of objects along a predefined way.

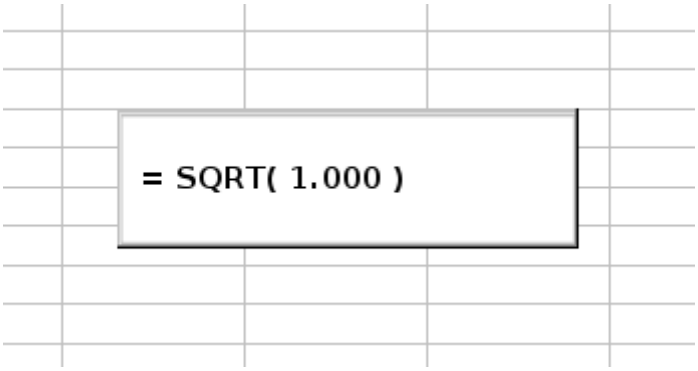
Headword patterns

For a visually pleasant presentation it is important to produce nice graphical illustrations of a list of headwords. For example put a headword in every corner and link them with arrows. As these layouts are relatively heavy in creation I would like to see patterns for them build in KPresenter. Along with the general layout they could be inserted into a slide.

KSpread

Zoom on enter

It is very useful to zoom a cell of the table on entering information to it. When formulas become complicated means long it is better to see them always complete. This is what zoom on enter guaranties. The cell is expanded and highlighted in a way you might think it is floating over the table.



Kivio and Karbon 14

I also thought about Kivio and Karbon14 as parts of KOffice that are quite useful. And these two applications are my perfect example of a wise merge.
When I wanted to create a Kivio flowchart I had really fast some problems:
I wanted to change the shapes to let them fit my needs. I wanted to draw a bit more than just shapes and the connectors between them. This was not possible but Kivio has some nice collections of shapes that would have speeded up my work.
So I finally took Karbon14 to create the shapes I needed and to draw the rest.
You surely understand conclusion: Merge Kivio and Karbon14. Karbon14 brings the vector support and a lot painting tools. Kivio contributes the functionality to create flowcharts.
Imagine how powerful such a combination would be:
The synthesised application would be a real 2D CAD application. With a shape collection including shapes of functional components or even assembly units and a drawing capability there are no borders.
Back to reality you might think that it is impossible to do technical outlines with such a synthesised application. You are right one thing is really badly missing, already in today's releases: a common and powerful dimension system.
This is something which would really make life easier. Entering just "4 cm" and the application turns this value into the exact length of pixels referring to a given reference value would be very comfortable. This is very clear because normally you don't have a pixel value as a length given on a master. The more so as nobody thinks in real lengths so the application should deal with all ratios.
Defining different types of lines with each one having different properties is the next step. This particular feature eliminates all problems with lines widths changing during a resize and so on.

KChart

As drag and drop is a fast and easy way to create things I would like to see it in KChart as well. The workflow for creating a chart would be like this:
Open or create the table with the information using KSpread. Drag the first data value this means the first cell into the new chart which is visible beside the spreadsheet. The value becomes visible as a part of the chart immediately when the draging mouse cursor enters the chart window. The chart's part can be for

example a piece of a pie chart. By moving it on the window the user can define the value's position within the chart.
Further layouting can be done by draging the wanted parts e.g. legends, headers etc on the chart.