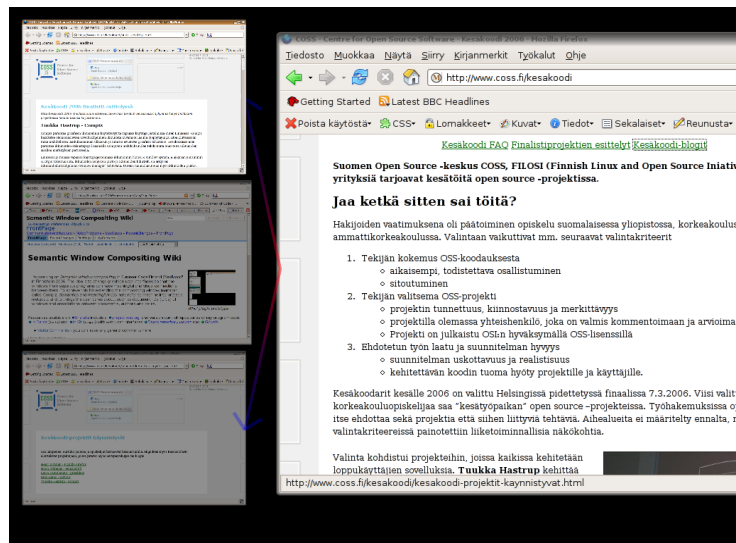


Project report on Semantic Window Compositing

1 Introduction

Semantic Window Compositing was one of the five projects accepted to the Summer Code Finland 2006 program funded by IBM, Nokia, Teknologiakeskus Hermia, Ericsson, and Novell.

The project implemented a new user interface concept in the X Window System as a plug-in for the Compiz compositing window manager and an accompanying extension for the Firefox web browser. Although the project did not fulfill all its initial goals, the concept is now demonstrable in working code, and I learned a lot about developing for the big platforms Firefox and the X Window System.



Picture 1: The user interface

Window compositing here refers to the technology used to create new visualizations of ordinary application windows. *Semantics* and *meaningfulness* here refer to machine-interpretable metadata such as document IDs (URIs) of windows, and to things the user cares about, such as associations between documents, authors and dates. These are in opposition to graphical data whose meaning is opaque to software, and the arbitrary conventions of current desktop systems including files and applications.

1.1 X Window System

X Window System is the underlying platform for free desktop environments such as Gnome and KDE. It has been the *de facto* standard for at least ten years and there are no comparable competitors. Fortunately it continues to evolve even today.

This project was made feasible by Xgl, which is an implementation for a server part of the X Window System using OpenGL for graphics card -accelerated, potentially three-dimensional scenes. The Xgl software has been made a part of the most often-used X Window System implementation, the X.Org. It has been

deployed as an option to the users at least in SUSE Linux Enterprise Desktop 10.

Xgl is one of the possible directions of future X.Org development. Currently however, it works best only with graphics adapters from ATI, nVidia and Intel. Even if Xgl isn't widely deployed, the window compositing technologies required by Compiz are also distributed in the traditional X.Org at the time of writing.

1.2 Compiz

Compiz takes advantage of the features provided by Xgl and implements the user-visible functionality. It functions as a compositing window manager via a recent X Window System extension called XComposite, which permits it to control how application programs are rendered on screen without needing any kind of support from them.

Compiz is cutting-edge technology. It has concentrated on parity with competitors and visual impression, although it has potential for major desktop usability and architectural advancements. Compiz is built from plug-ins, which facilitates the implementation of new features like the ones implemented in this project.

Compiz has been licensed with both MIT and GPL licenses and is available from the Freedesktop.org server.

1.3 Firefox

Firefox is one of the established browsers in the market and a flagship of the free software community. It is searching for ways to stand out in the next browser generation from competitors. Browser tab interface has been a major advantage but is becoming commodity. It also has its limitations in the case of more than about ten tabs per window.

Firefox was chosen as the first application extended because web browsing is a key task for users and it could benefit from new user interface concepts. In a way, tabbed browsing is treating the symptoms of problematic window management by creating a new, restricted and limited kind of "windows". Treating the cause means to improve for all applications the window management use case of numerous simultaneously open windows.

1.4 Semantic Desktop research

Using RDF in application programs has become an interest of research a couple of years ago. This Semantic Web -related research area has been given the name Semantic Desktop. The implemented software applications, however, have been based on the needs of the scientific research instead of the Free Software community. They have, for example, been implemented in Java and are otherwise unfitting to the environment. Characteristically, a recently published article on the topic explains how a Semantic Desktop application would typically be full-screen (Sauermaun *et al*, 2005).

However this project held the view that the issue is just that implementing the ideas of the Semantic Desktop requires changes that span the whole screen. These changes could also be made in a way that the existing applications and

programs following the X Window System architecture can form a part of it. Thus, the most significant benefit of the direction explored is precisely in advancing an existing system architecture that is suitable for Free Software. In the future, the view components described in the article could be implemented as separate X programs that communicate in agreed manner not unlike the current window management standard ICCCM.

The project was not scientific but applied. The topic was defined so that the results after the summer could be generally deployed. However, the topic had been discussed also with some published Semantic Web researchers, and they were interested in the goals of this work. Via the Fenfire project, this project was related to researchers both from the University of Jyväskylä and from the international hypertext and Semantic Desktop community. It is possible that the work could also form a platform for future Semantic Desktop research. Depending on the interests of the organizers, the project could also have included scientific work or its preparation.

2 Idea and motivation

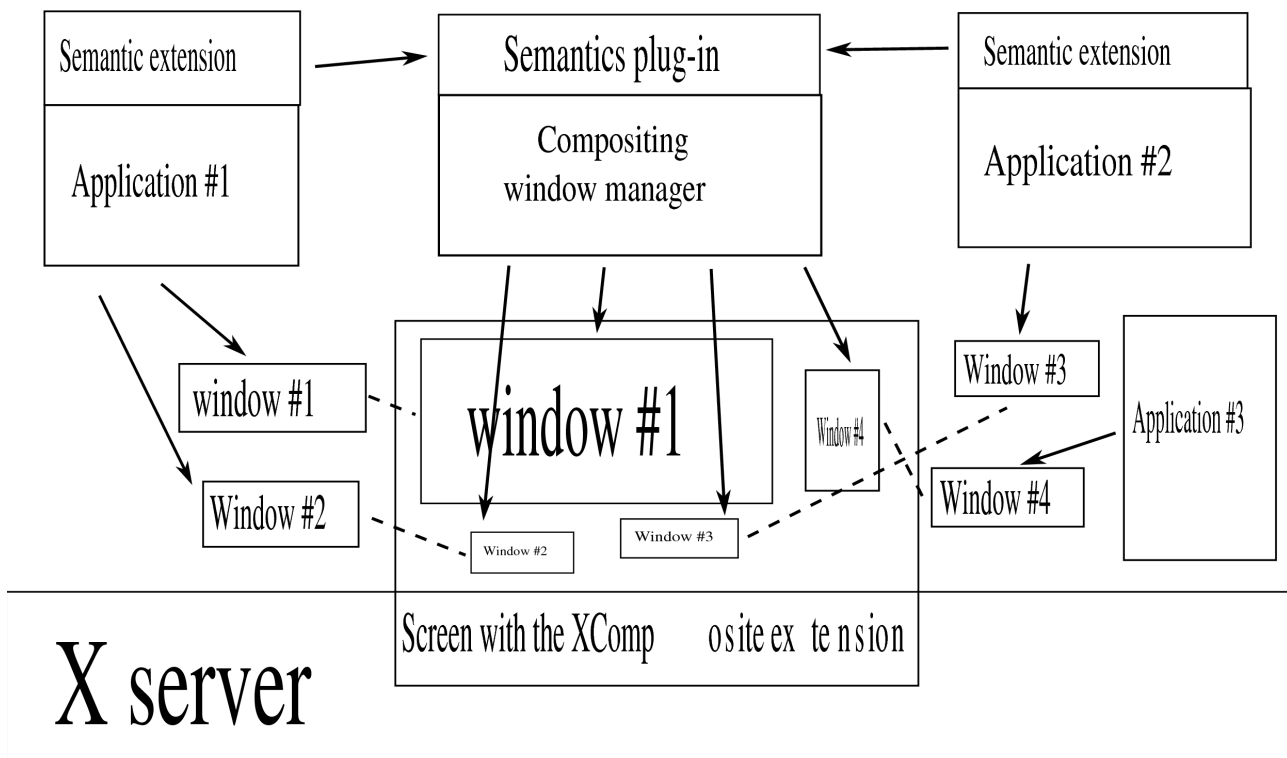
The idea of the project was to encourage new implementations of window management ideas and research by performing one concrete exploration: changing graphical user interfaces so that the windows from separate programs can have meaningful and visual connections between them. Compiz as a compositing window manager and with its plug-in system makes this possible with evolutionary changes to the current systems.

Window compositing is a major technical leap in windowing systems provided by the use of modern graphics cards. We can now treat any window as if it was an image or a texture. This enables zooming, animating, transparency, perspectives etc. and Compiz has acquired a strong end-user community who appreciates a visually modern windowing environment. This project needed the possibility to transition between separate window layouts with scaling and to visualize associations between windows.

The ultimate target aimed at by the project was turning new technologies into increased usability. It is no longer necessary to follow the historically feasible model of simulating in the windowing system an office desk of the 1970s with paper documents stacked on it. In the long run, “files” might get replaced by a connected network of information and “applications” with various views and tools on information. People are interested in the real world and its relations.

While the graphical pixel data is enough to present a user interface to a human, semantic data is needed for the window manager to be able to operate based on what the human sees as the (content of the) windows. The semantic web technologies provide for this in a de-centralized architecture: URI “addresses” (Uniform Resource Identifiers) can be formed for meaningful concepts and communicated between programs, and RDF statement sets can express metadata and connections between concepts.

In summary, the strengths of this approach are: evolutionary change, decentralized architecture, deployment of underutilized hardware, and an existing end-user community.



Picture 2: Structure of the system

3 Plan

The functionality to be implemented was divided into three parts:

1. Compiz plug-in
2. Extensions to applications
3. In the plug-in, interpretation of the data provided by the extensions

The plug-in was deemed the most important part of the work because of its central role. It was first developed to the stage where it can composite the windows of an application next to each other. More semantic data is gathered from the application programs by using a simple but extensible and open communications protocol which was developed next. The communication can happen via X as implemented, or via a TCP/IP connection, and the content of the messages can be for example (along the RDF model, see <http://www.w3.org/RDF>):

```
<window number 1> <displays address> <http://coss.fi/kesakoodi>
<window number 3> <displays address> <http://coss.fi/kesakoodi/faq.html>
<window number 2> <was opened as a link from> <window number 1>
```

The windows can belong to separate application programs, and when the plug-in is active, it should understand to composite – when window number 1 is in focus

– next to it windows number 2 and 3 in smaller size.

To communicate these messages, the plan was to implement some small extensions in popular application programs such as Firefox, and at least in two application programs. Firefox was deemed a good example application, as a lot of data and documents handled on a workstation are somehow connected to the WWW. In addition, users often have a lot of WWW pages open at a time, which can make it difficult to find the right one from all the windows and tabs.

There are many simple ways to interpret and present the data in the plug-in. The plan was to implement at least one functionality in which the association type <was opened as a link from> is presented by an arrow that connects the windows. Further, the project examined the possibility of aligning the tail of the arrow with the link itself whenever it is visible. Tabbed browsing is a popular feature in Firefox, and this provided comparable functionality adding the possibility to see the associations between pages, and more than one page can be visible at a time.

In addition to other work during the project, the plug-in interface of Compiz was documented for other developers.

4 Background events

I felt the upstream project, its developers and more generally the whole open-source community was significantly less active and thus less present during the summer months. However in the large, Compiz is achieving deployments and paving the way for the compositing window manager technology in general.

- Compiz upstream at Freedesktop.org moved from CVS to Git on 2006-06-29
- Compiz web and wiki site started on 2006-11-08

Compiz support on normal X servers in addition to Xgl:

- X.Org X11R7.1 released with AIGLX and GLX_EXT_texture_from_pixmap on 2006-05-22
- Compiz upstream supports AIGLX on 2006-09-20
- NVIDIA beta driver with GLX_EXT_texture_from_pixmap support on 2006-09-21

Versions of Compiz:

- Compiz released as a preview in SUSE Linux 10.1 on 2006-05-11
- Compiz released for production in SUSE Linux Enterprise Desktop on 2006-07-17
- Community fork Beryl
 - Development release of Beryl 0.1.0 on 2006-09-28
- Compiz entered Debian unstable on 2006-09-29 and is targeted for next stable release
- Stable release of Compiz 0.2.0 on 2006-10-02

5 Schedule

Plan for June was:

- Starting the project
- Building a development environment
- Preliminary plug-in to Compiz
- A prototype of the communication system
- Familiarization with Firefox extensions
- Documentation on the Compiz plug-in interface

June went mostly as expected, although I decided to concentrate on Compiz first and started with Firefox in the beginning of July.

Plan for July was:

- Communication system
- Preliminary extension to Firefox and some other software (GPDF, X-Chat, Gnome Terminal, Gnowsis?)
- Exploration of the possibilities of semantics presentation

In July I accomplished the goals, but because of unexpected difficulties with both Compiz and Firefox didn't really meet the intended depth and quality. Thus I had to continue on these tasks in August. I had enough to do trying to hack the Firefox extension, so I didn't work on an extension to any other application yet.

Plan for August was:

- More implementation of semantics presentation
- Merging the plug-in to Compiz CVS
- Publishing the Firefox extension
- Final user manual

Unlike planned, I continued work on the basic features still in August. The code was not of releasable quality so the release plans were not topical. Instead, I worked to turn the code into alpha-testable form so that interested people can experiment with the code by following the development and installation guide.

The work extended into autumn as I wanted to bring the project into a proper conclusion while I continued my computer science studies and began technology business studies. The work became easier with a more active community and better releases.

I prepared a presentation and a demonstration for the Openmind 2006 seminar in Tampere, Finland. A difficult part was creating a video file of the user interface in software, without analog video capture devices. The existing record software was able to capture the screen otherwise, but surprisingly not the new user interface. I had to create a new Compiz plug-in which captures each frame via OpenGL into an image file. These images can then be assembled and encoded into a video. A result was shown in the seminar and is available for download from the project web site.

In November I finally managed to implement input redirection faking in the Compiz plug-in. Input redirection is a feature missing from X.Org that complements window compositing. Without it, keyboard and mouse input to transformed windows is interpreted based on untransformed, wrong locations. At planning stage I didn't realize it was missing and later I was hoping it would get implemented soon or (as it became clear implementing it is complicated) that I could give a fair idea of the system even without input. However, needing to ask people to imagine it was a major disadvantage and I tried to implement faking several times. The current faking implementation moves the untransformed window so that the location under the mouse pointer matches the transformed location. This makes input possible until a cleaner solution lands in X.Org.

6 Results

The results of the project provide a Compiz plug-in and a Firefox extension as planned, although they are not of production quality yet. The Compiz plug-in is usable but has bugs and lacks in polish and generality. The Firefox extension is a minimal implementation of the idea serving as a live demonstration of the concept. Together they provide a top-to-bottom vertical coverage from the meaningful concept of a web page, communicated as the URI by the semantic extension via window metadata properties to the semantic plug-in of Compiz, and visualized as connecting lines between windows showing pages from the same web site.

Results of the documentation goal are a significant part of the limited documentation available about Compiz. There is an overview document of Compiz technologies, implementation and interfaces. There are comments for the Compiz application programming interface (API) although not extensively.

6.1 Compiz plug-in Whirl

The Compiz plug-in Whirl code is based on the core scale plug-in which provides a window switching utility by tiling all the windows so that they don't overlap and letting the user click a window to return to the normal window arrangement with the selected window made active.

As the project builds on Compiz, it is possible to switch between the contemporary desktop and the new user interface with a key press and see the same application windows seamlessly animate to new positions. The active window becomes the focus of the new user interface, shown large on the right side of the screen and scaled to fit if larger than the allocated space. Other windows of the same application and semantically connected windows form the context and are tiled on the left side of the screen. No windows overlap so it is easy to work on the focused window and see all the related windows outright.

In the new user interface, switching the active window is implemented in clicking one of the windows in the context. This initiates a transition to a new layout where the newly selected window becomes the focus. As the context switches to correspond to the new focus, other windows appropriately fade in and out.

Input redirection faking makes it possible for the window in focus to receive keyboard and mouse input. For example, it's possible to browse the web, open links in new windows and see them pop up in the context on the left. Along with this, it made sense to make the desktop and edge panels visible, as it is now possible to interact with them.

6.2 Firefox extension Window Metadata

The Firefox extension Window Metadata adds basic semantic information to the properties of the browser windows. These properties are accessible to the Compiz plug-in and any other programs on the desktop session. Currently the information consists of the address and title of the web page loaded in the window.

As there weren't applicable examples available of Firefox extensions that contain native code, this extension also servers as one. The extension implements a native component that provides the service of setting window properties. The rest of the extension is then implementable in Javascript as usual.

7 Experiences

Work on the project was more difficult than estimated in the interfaces of Compiz and Firefox. In case of Compiz there were some good examples but extracting meaning from the undocumented code was problematic. Firefox has some documentation but surprisingly there weren't significant examples of native components as part of an extension. The vast scope of Firefox shows in the size of the interfaces.

It is clear that Compiz is programmed as a low-level project, closer to the X server and graphics drivers than application programs I suppose. That is not the impression I initially got or what I would expect. My experience is that a higher-level approach would be more appropriate: window management is a notoriously complex and delicate task to get right because of the variety of clients and exceptional cases in the code, and it's not handling individual pixels, complex 3D models or even hundreds of windows. The plug-in system introduces additional complexity but perhaps it can also be used to provide a higher-level interface for programmers.

Having programmed mainly in Java and Python lately, the development work in this project implied going back to basics in the extreme: crashes in kernel graphics drivers, exceptions without back traces in Firefox, having to run the debugger from another machine, the usual problems of using C. It was good experience to during the course of the project remember the usefulness of enabling and then analyzing core dumps.

8 Conclusions

I think the first Summer Code Finland was a success in how it gave professional-level employment conditions and resourcing for five diverse Free Software projects that all would have left a hole for a time if unimplemented. It gives an example on how to fairly hire students and direct the work so as to not demotivate existing community actors.

In hindsight it is clear to me that a Summer Code project deviates from free-time Free Software projects in some essential points that I overlooked, the main issue being that of producing results in a given time period. Of course there was a plan with a schedule, but the other side was completely missing: risk analysis! I think the ignorance towards risks stems from the typical, informal Free Software project. I do not mean risks should be avoided. I mean risks should be thought about as a part of continuous planning, to put the current situation and plans into a context.

There's also a point where Summer Code Finland is the opposite of Google's Summer of Code: the mentoring is unofficial, not explicitly part of the program. This is good because applicants can pursue an idea irrespective of whether a suitable mentoring organization participates. It should just be kept in mind that lack of formal mentoring means less pressure to social communication from that direction. I think one thing with space for improvement is the official forms of communication.

The official form of communication was the public project log. It didn't work for two-way communication at all. In such a case it's impossible to improve the log keeping. While I think writing weekly log entries makes sense, there should be some scheme to get comments. Perhaps together with this or independently, a weekly IRC meeting could be scheduled to ensure discussions between the participants and organizers – and mentors if they have the time.

All in all, I'm happy of the full possibility to work independently and make the project succeed or fail depending on my own decisions and actions. It was a highly challenging experience and my high goals made it more challenging. I hope the project is both a useful data point for subsequent applicants and one inspiration for future windowing system work.

9 Future work

One obvious possibility for future work is developing the Compiz plug-in into production quality with respect to input redirection faking or the future real implementation, generalizing it to work on non-hardcoded types of semantic information and enhancing the appearances.

One short-term task is making a better demonstration video of the user interface now that it has input redirection faking.

I have planned scientific theses for a part of my studies: one on the motivation and evaluation for more radical usability ideas, and one on the exploration of the architectural ideas born from this project.

The communication system between Compiz and the applications is currently very simplistic. There would need to be a two-way messaging system for the further development of the idea of visual desktop components.

One clear need in perspective of my struggles is the possibility to program window compositing in a higher-level language, whether it's a compositing window manager written in such a language or a plug-in environment for Compiz.

10 Acknowledgments


The ideas of this project are based on the work of the scientific hypertext and semantic desktop research communities.

I would like to thank my mentor David Reveman of the Compiz project for his work on the underlying technologies and Compiz, and for his encouraging comments on my unclear plans.

This project owes its existence to the organizers and sponsors of Summer Code Finland 2006. Thank you for the opportunity!

11 Further information

1. Progress reporting in the project weblog at <http://iki.fi/Tuukka/2006/summercode/blog>
2. Project documentation in the project wiki at <http://iki.fi/Tuukka/2006/summercode/wiki>
3. Produced code accessible with instructions at <http://iki.fi/Tuukka/2006/summercode/wiki/CheckingMyCode>
4. Project presentation at Openmind 2006 seminar: slides and demo video at <http://tuukka.iki.fi/2006/summercode>
5. Leo Sauermann, Ansgar Bernardi, Andreas Dengel. Overview and Outlook on the Semantic Desktop. In Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference. <http://www.dfki.uni-kl.de/~sauermann/papers/Sauermann+2005d.pdf>
6. Compiz web and wiki site at <http://go-compiz.org/>



Semantic Window Compositing
a Summer Code Finland 2006
project

Tuukka Hastrup <Tuukka@iki.fi>
Openbyte 2006-10-24