jugend forscht

Mathematik/Informatik

Tobias Kaiser

Entwicklung eines effizienten und neuartigen X-Fenstermanagers

8. Februar 2010

Inhaltsverzeichnis

1 Einleitung **2**
1.1 Aufgabe eines Fenstermanagers 2
1.2 Das X Window System 3

2 Bekannte X-Fenstermanager **3**

3 Konzept **4**
3.1 Arbeitsflächen 4
3.2 Nicht überlappende Fenster 5
3.3 Überlappende Fenster 5

4 Umsetzung **6**
4.1 Wie programmiert man einen Fenstermanager? 6
4.2 Entwicklungswerkzeuge 6
4.3 Beschreibungssprache für nicht überlappende Fenster 7
4.4 Fensterrahmen 7
4.5 Bedienungs- und Konfigurationssprache 8

5 Ergebnis **8**

6 Diskussion **9**

Literatur **10**

1 Einleitung

Da ich mich sowieso für Informatik und Programmierung von Computern interessiere, und ich alternative Bedienkonzepte für Computer spannend finde, geht es in dieser Arbeit um eine Möglichkeit, das Arbeiten am Computer zu erleichtern.

WIMP ist eine Abkürzung für Window (Fenster), Icon (Symbole), Menu (Menü) und Pointer (Zeiger): die Grundsteine der meisten heute verwendeten grafischen Benutzeroberflächen. [1] Diese Elemente ermöglichen einen einfachen, intuitiven Umgang mit dem Computer. Doch sind sie auch die effektivste, produktivste, schnellste Möglichkeit, mit einem Computer zu arbeiten?

Indem ich alternative Bedienkonzepte ausprobierte, lernte ich andere, in meinen Augen effizientere, also produktivere, und fortgeschrittenere Möglichkeiten kennen, mit einem Computer zu arbeiten, als nach dem WIMP-Konzept.

Menüs und intuitive Bedienung über Symbole fallen bei diesen ungewöhnlichen Konzepten im Großen und Ganzen weg, die Maus tritt in den Hintergrund und die Fenster haben keinen gewöhnlichen Rahmen mehr.

1.1 Aufgabe eines Fenstermanagers

Ein Fenstermanager (engl. *window manager*, abgekürzt oft WM) ist allgemein ein Computerprogramm, welches Fenster auf einem Bildschirm (oder mehreren Bildschirmen) verwaltet. Er bestimmt die Anordnung der Fenster und stellt beispielsweise Funktionen zum Verschieben, Vergrößern, Verkleinern, Minimieren, Maximieren und Auswählen der Fenster zur Verfügung. Oft stellt der Fenstermanager auch die Fensterränder und die Titelzeile dar.

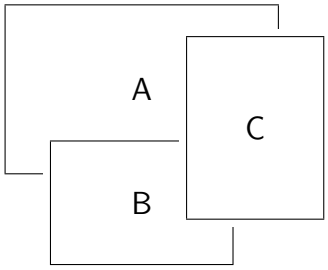

Abb. 1: Beispielhafte Fensteranordnung eines gewöhnlichen Fenstermanagers. Die Rechtecke stellen die Fenster dar.

Unter Microsoft Windows wird die Aufgabe des Fenstermanagers vom Client/Server-Laufzeit-Subsystem (csrss.exe) übernommen, welcher zu den Grundbestandteilen des Betriebssystems gehört. [2]

Es ist unter Microsoft Windows recht schwierig, die Bedienung und die grafische Ausgabe des Computers im Kern zu ändern, da der Fenstermanager nicht auswechselbar konzepiert wurde.

Unter Linux und anderen unixoiden Systemen ist das X Window System das üblicherweise verwendete System zur Ein- und Ausgabe für grafischen Benutzeroberflächen. Der von mir programmierte Fenstermanager ist für das X Window System programmiert, da ich selbst Linux und das X Window System verwende und weil das X Window System transparent aufgebaut und dokumentiert ist, was es einfacher macht, einen Fenstermanager für dieses System zu entwickeln.

1.2 Das X Window System

Mit der Entwicklung des X Window Systems wurde 1984 am Massachusetts Institute of Technology in den USA begonnen. Seit der Veröffentlichung der Version 11 im Jahr 1987 wurde der Kern des Systems nicht mehr bedeutend verändert.

Der X-Server ist ein Programm, das auf dem Computer läuft, an dem die Ein- und Ausgabegeräte angeschlossen sind.

Zum X-Server können sich dann die Anwendungsprogramme, zum Beispiel ein Internetbrowser oder ein Textverarbeitungsprogramm als X-Clients verbinden. Die Clients können Fenster öffnen, in ihnen zeichnen und Eingaben des Benutzers vom X-Server entgegennehmen.

Der Fenstermanager verbindet sich auch als Client zu einem X-Server. Es kann sich aber zu jedem X-Server nur ein X-Fenstermanager verbinden. [3]

2 Bekannte X-Fenstermanager

Ein X-Fenstermanager wird dann notwendig, wenn man komfortabel mit dem X Window System arbeiten möchte. Daher benutzen alle Desktop-Linux-Anwender, die X verwenden, und die meisten tun das, einen X-Fenstermanager.

Die großen Linux-Desktopumgebungen haben ihre eigenen X-Fenstermanager. Diese Fenstermanager orientieren sich in der Regel an den Bedienkonzepten von Microsoft Windows. Beispiele für solche Fenstermanager bekannter Desktopumgebungen sind Metacity [4] aus GNOME und KWin [5] aus KDE. KDE und GNOME sind die beiden wichtigsten Desktopumgebungen für Linux und ähnliche Betriebssysteme.

Diese Fenstermanager lassen sich intuitiv bedienen: Es gibt Fensterrahmen, mit der Maus lassen sich die Fenster beliebig überlappend positionieren und mit einem Kreuz-Symbol im Fensterrahmen kann man beispielsweise das Fenster schließen. Dies hat den Vorteil, dass man nichts über den Fenstermanager wissen muss um ihn zu bedienen, auf der anderen Seite hat man durch den Fenstermanager wesentliche Einschränkungen im Umgang mit Fenstern.

Neben diesen einfach zu bedienenden, für normale Anwender gedachte Fenstermanagern, gibt es auch Fenstermanager, die auf anderen Bedienkonzepten aufbauen.

Ein alternativer, heute von wenigen Leuten verwendeter Ansatz zur Fensterverwaltung ist das nicht überlappende, bildschirmausfüllende Anordnen der Fenstern in so genannten Kacheln (engl. *tiles*). Diese Fenstermanager nennt man dann *tiled window manager*. Es gibt eine ganze Reihe dieser Fenstermanager für X, zum Beispiel Ion [6], Ratpoison [7] oder Xmonad [8].

Ich habe viele solche Fenstermanager lange verwendet, doch kein Fenstermanager machte genau das, was ich wollte. An Xmonad oder Ratpoison gefiel mir vor allem nicht, dass man die Fenster nicht beliebig nicht überlappend anordnen

konnte. An Ion und Xmonad gefiel mir nicht, dass sie ihre Konfiguration in vielen unübersichtlichen Dateien lagern.

Ich beschloss also, einen eigenen Fenstermanager zu schreiben, der meinen Vorstellungen entspricht.

3 Konzept

Bevor ich angefing, Programmcode zu schreiben, habe ich mir überlegt, was der Fenstermanager überhaupt können sollte.

Mein Fenstermanager soll folgende Merkmale haben:

- Der Fenstermanager soll vollständig über selbst konfigurierbare Tastenkombinationen steuerbar sein. Dies ermöglicht eine schnelle Bedienung.
- Der Platz auf dem Bildschirm soll von den Programmen möglichst effizient genutzt werden. Dadurch bekommt der Benutzer mehr Platz für seine wirkliche Arbeit, die im Grunde genommen überflüssigen Fensterrahmen werden nicht benötigt.
- Der Fenstermanager soll mit *einer einizgen* Konfigurationsdatei auskommen. Bei Xmonad und Ion hat mich die große Anzahl von unübersichtlichen Konfigurationsdateien gestört.
- Das Programm sollte von möglichst wenigen Programmbibliotheken abhängig sein. So ist es weniger Aufwand, das Programm auf anderen Systemen zu installieren.

3.1 Arbeitsflächen

Eine unter X-Fenstermanagern verbreitete Funktion ist die Bereitstellung mehrerer Arbeitsflächen für einen Bildschirm.

Jede Arbeitsfläche ist eine Art *virtueller Bildschirm*, auf dem sich mehrere Fenster befinden können. Es wird immer nur eine Arbeitsfläche auf einem Bildschirm dargestellt. Zwischen den Arbeitsflächen kann typischerweise mit Tastenkombinationen gewechselt werden.

Mit mehreren Arbeitsflächen verliert der Benutzer auch bei der gleichzeitigen Verwendung von vielen Programmen nicht den Überblick, da Mengen an übereinandergetapelten Fenstern und unübersichtlich viele minimierte Fenster nicht mehr notwendig sind, stattdessen kann man die Anwendungen auf verschiedene, aufgeräumte Arbeitsflächen verteilen.

Fenster lassen sich von einer Arbeitsfläche auf eine andere verschieben.

Ich habe nicht vor, meinem Fenstermanager Minimieren von Fenstern beizubringen, da durch das Minimieren eine Taskleiste zum Anzeigen der laufenden Fenster unverzichtbar wäre, diese Taskleiste aber wertvollen Bildschirmplatz verbrauchen würde.

3.2 Nicht überlappende Fenster

Die Kernfunktion des Fenstermanagers ist das *bildschirmfüllende, nicht überlappende* Anordnen von Fenstern.

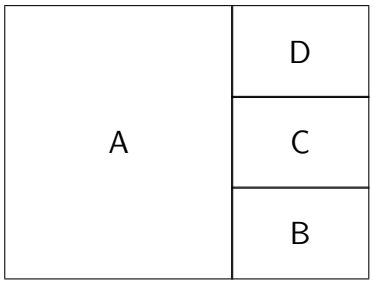

Abb. 2: Eine einfache Fensteranordnung

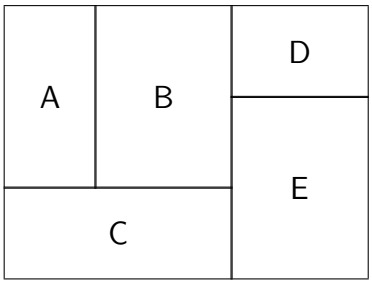

Abb. 3: Eine komplexe Fensteranordnung

Es gibt verschiedene Möglichkeiten, Fenster so anzuordnen.

Der minimalistische Fenstermanager Dwm, der Fenster nicht überlappend verwalten kann, teilt den Bildschirm in zwei vertikal getrennte Teile ein: ein Teil, in dem sich das wichtigste Fenster befindet und ein Teil, in dem sich alle anderen Fenster übereinander angenordnet befinden. [9]

Dieser Ansatz hat den Nachteil, dass man nicht beliebige Anordnungen vornehmen kann. Während die Anordnung in Abbildung 2 von Dwm dargestellt werden kann, kann Dwm keine komplexeren Anordnungen wie in Abbildung 3 vornehmen.

Der Fenstermanager Ion kann dagegen auch die Anordnung aus Abbildung 3 darstellen, da er beliebige nicht überlappende Fensteranordnungen erlaubt.

Mein Fenstermanager kann wie Ion jede mögliche bildschirmfüllende Anordnung von nicht überlappenden Fenster darstellen.

3.3 Überlappende Fenster

Obwohl die Hauptaufgabe des Fenstermanagers im Verwalten von nicht überlappenden Fenstern liegt, möchte ich dem Benutzer auch ermöglichen, über der Ebene von nicht überlappenden Fenstern eine Ebene von frei platzierbaren, überlappenden Fenstern zu haben. Im Englischen wird diese zweite Ebene als *floating layer* bezeichnet.

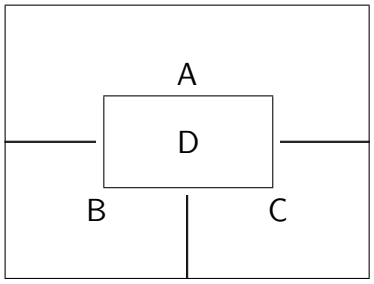

Abb. 4: Gemischte Anordnung

Dieser Mechanismus ermöglicht es, Fenster wie von normalen Fenstermanagern gewohnt anzuordnen.

Dies ist für viele Fenster, die nur kurz auftauchen, wie zum Beispiel Assistenten zum Einrichten einer Software, Passworteingaben oder Fehlermeldungen eine nützliche Sache, da man nicht jedem Dialogfenster, das nur kurz auftaucht,

einen eigenen Bereich auf dem Bildschirm zuordnen möchte. Stattdessen erscheinen diese Fenster dann wie Fenster D in Abbildung 4.

4 Umsetzung

Der nächste Schritt in der Entwicklung des Fenstermanagers war die Programmierung des Fenstermanagers nach dem geplanten Konzept.

Beim Programmieren entstanden weitere Fragen und im Konzept noch nicht behandelte Probleme tauchten auf, die ich noch beschreiben werde.

4.1 Wie programmiert man einen Fenstermanager?

Fenstermanager für X werden normalerweise ganz oder zumindest teilweise in C programmiert, da die klassische Programmierschnittstelle Xlib des X Window System für C geschrieben ist.

Es gibt neben der Xlib auch noch eine zweite neuere, auch für die Verwendung mit C geschriebene Programmierschnittstelle namens XCB, mit der man auch Fenstermanager programmieren kann.

Ich habe mich für die Verwendung der Xlib entschieden, da sie verbreiteter und besser dokumentiert ist.

Eine verständliche und ausführliche Xlib-Dokumentation, mit deren Hilfe ich meinen Fenstermanager geschrieben habe, ist unter [10] zu finden.

Außerdem habe ich mir den Quellcode des Fenstermanagers Dwm angeschaut, um zu verstehen, wie er funktioniert. Dwm ist hierfür besonders gut geeignet, da er nur aus etwa 2000 Zeilen C-Programmcode besteht. [9] Die Analyse seines Quelltexts hat mir sehr geholfen, um zu verstehen, wie ein X-Fenstermanager im Detail funktioniert.

4.2 Entwicklungswerkzeuge

Ich schreibe den Code mit einem Texteditor und kompiliere ihn mit dem GNU C Compiler und einem Makefile.

Natürlich muss ich beim Entwickeln den Fenstermanager oft ausprobieren, um Fehler zu erkennen und beseitigen. Ich habe zwei Möglichkeiten gefunden, einen X-Fenstermanager zu testen:

1. Man kann unter Linux ohne das X Window System den Fenstermanager in einem Textterminal (ähnlich der DOS-Eingabeaufforderung) entwickeln und das X Window System zu jedem Test starten.

2. Ich habe eine andere Möglichkeit gefunden, im normalen X Window System zu programmieren und dann zum Testen ein Fenster zu öffnen, in dem der Fenstermanager läuft. Das geht mit dem Programm Xephyr, welches sich als X-Client in Form eines Fensters zu dem laufenden X-Server verbindet

und selber als zweiter X-Server arbeitet. [11] Xephyr ist praktisch ein „Bildschirm im Bildschirm“, mit dem sich die Entwicklung eines Fenstermanagers beschleunigen lässt.

4.3 Beschreibungssprache für nicht überlappende Fenster

Ich habe eine Beschreibungssprache für nicht überlappende, bildschirmfüllende Anordnungen entwickelt, mit der man solche Anordnungen in wenigen Zeichen für den Computer eindeutig konstruierbar ausdrücken kann.

Sie ist wie die Umgekehrte Polnische Notation (Postfixnotation) in der Mathematik oder die Programmiersprache forth stapelorientiert. [12] Man kann Fensterrahmen auf den Stapel legen (Syntax: `(`*`Fenstername`*`)`). Es gibt zwei Aktionen/Befehle in der Sprache, die die beiden obersten Elemente des Stapels durch ein Element ersetzen: Die Anordnung der Fenster horizontal und die Anordnung der Fenster vertikal nebeneinander. Dieser Stapelaktion wird das Größenverhältnis des oberen zum unteren Teil bzw. des linken zum rechten Teil als Argument gegeben. Die Syntax für die Anordnung nebeneinander lautet `v[`*`Verhältnis`*`]`, für die Anordnung übereinander `h[`*`Verhältnis`*`]`. Das Verhältnis wird in Form von zwei ganzen Zahlen angegeben, z. B. `1:5` oder `7:3`.

Wenn alle Kommandos abgearbeitet sind, liegt genau eine Anordnung auf dem Stapel, die auf dem Bildschirm dargestellt wird.

Die einfachste nicht überlappende und bildschirmfüllende Anordnung ist ein einziges, „maximiertes“ Fenster. Eine gesonderte Maximieren-Funktion wird überflüssig, weil eine solche Anordnung den gleichen Zweck erfüllt. In dieser Sprache drückt man diese Anordnung als `(Fenster)` aus.

Die in Abbildung 3 (S. 5) dargestellte Anordnung kann man beispielsweise in dieser Sprache wie folgt beschreiben: `(A)(B)v[2:3](C)h[2:1](D)(E)h[1:2]v[5:3]`, für die Anordnung in Abbildung 2 lautet der Beschreibungstext `(A)(B)(C)(D)h[1:1]h[1:2]v[5:3]`.

Eine Funktion wandelt so eine Zeichenkette in die Positionen der Fenster um. Durch die Änderung des Beschreibungstextes mit einer Funktion lassen sich Transformationen auf eine Fensteranordnung anwenden: Man kann so zum Beispiel einen Rahmen schließen oder den selektierten Fensterrahmen vertikal oder horizontal teilen.

4.4 Fensterrahmen

Fensterrahmen (auch als Fensterdekoration bezeichnet) sind die Ränder, die ein Fenstermanager um die Fenster zeichnet. Der Rahmen hat bei gewöhnlichen Fenstermanagern mehrere Funktionen: Die obere Seite des Rahmens enthält die Titelleiste und Bedienknöpfe. Am Rahmen lassen sich Fenster vergrößern, verkleinern und verschieben.

Mein Fenstermanager zeichnet keine solchen konventionellen Fensterrahmen, sondern lediglich einen 1 Pixel breiten Rand, der, je nach dem, ob das Fenster

gerade aktiv ist, weiß oder rot ist. Dies führt zu einer sehr effizienten Platzaufteilung.

Mein Fenstermanager verhält sich in dieser Hinsicht wie der Fenstermanager Xmonad in der Grundeinstellung. [8]

Normalerweise zeichnet ein Fenstermanager nur Fensterrahmen zu existierenden Fenstern. In meinem Fenstermanager nehmen die Fensterrahmen der Ebene aus nicht überlappenden Fenstern eine Sonderstellung ein: Sie werden auch schon dargestellt, wenn sie vom Benutzer angelegt, aber noch nicht mit einem Fenster gefüllt worden sind.

Ein solcher leerer Fensterrahmen wird mit einem Fenster gefüllt, wenn der leere Fensterrahmen ausgewählt ist und sich dann ein neues Fenster beim Fenstermanager anmeldet. Wenn sich jedoch schon ein Fenster in dem Fensterrahmen befindet, wird das Fenster automatisch der Ebene aus überlappenden Fenstern hinzugefügt. Dieser Mechanismus ist die Umsetzung des Entwurfs aus Abschnitt 3.3. Er stellt einen großen Unterschied meines Fenstermanagers zu anderen Fenstermanagern dar. Ion platziert auch gerne mehrere Fenster in einen Fensterrahmen; bei Xmonad und Dwm gibt es gar keine leeren Fensterrahmen.

4.5 Bedienungs- und Konfigurationssprache

Ich habe eine kleine Sprache programmiert, mit der man den Fenstermanager bedienen kann, z. B. Anwendungen öffnen kann oder Tastenkombinationen festlegen kann.

Eine bestimmte Konfigurationsdatei wird ausgeführt, wenn der Fenstermanager gestartet wird. In dieser Konfigurationsdatei stehen im Moment in erster Linie die Befehle, die die Tastenkombinationen für den Fenstermanager festlegen.

Neben Tastenkombinationen für Fenstermanager-interne Aufgaben lassen sich auch globale Tastenkombinationen erstellen, zum Beispiel zum Aufrufen eines Programms.

Tastenkombinationen sind nützlich, da sie, vorrausgesetzt man kennt sie, häufige Aufgaben, wie z. B. das Aufrufen eines Webbrowsers oder das Verfassen einer E-Mail beschleunigen. Statt in Menüs nach einem häufig verwendeten Programm suchen zu müssen, drückt man einfach zwei Tasten.

5 Ergebnis

Mein Fenstermanager tut, was er tun muss, nämlich Fenster verwalten.

Dauerhaft ist der Fenstermanager noch nicht praktisch einsetzbar, da er noch gelegentlich an mysteriösen Fehlern, die auf die maschinennahe C-Programmierung zurückzuführen sind, abstürzt.

Einige Funktionen des Fenstermanagers sind auch noch nicht funktionierend implementiert.

Daher werde ich wohl vorerst nicht aufhören können, Metacity, einen normalen Fenstermanager, zu benutzen – um den Fenstermanager dauerhaft einzusetzen, muss ich ihn wohl noch weiterentwickeln.

Es ist aber schon zu erkennen, dass die Arbeit mit meinem Fenstermanager die Bedienung des Computers grundlegend verändert. Wenn man die etwas kompliziertere Bedienung gelernt hat, kann man mit dem Fenstermanager wesentlich schneller und angenehmer arbeiten.

6 Diskussion

Natürlich gibt es noch viele Funktionen, die man in den Fenstermanager einbauen könnte.

- Optionale Fenster-Titelleisten wären manchmal nützlich, da einige wenige Programme, z. B. Webbrowser, möchten, dass interessante Informationen, beim Webbrowser der Titel der Website, in ihrer Titelleiste vom Fenstermanager dargestellt werden. In vielen Fällen sind diese Titel aber überflüssig, da sie keine nützliche Information enthalten.

- Der Fenstermanager unterstützt noch keine Panels. Panels sind der Startleiste unter Microsoft Windows ähnliche Programme, die z. B. Statussymbole, Menüs oder die laufenden Programme enthalten.

- Die Extended Window Manager Hints, die in [13] spezifiziert werden und die Kommunikation zwischen Fenstermanager und Fenster verbessern sollen, werden von meinem Fenstermanager kaum unterstützt. Mit diesen Spezifikationen wäre es zum Beispiel möglich, mit unabhängig von meinem Fenstermanager entwickelten Panels die Arbeitsfläche zu wechseln.

- Der Fenstermanager beherrscht noch nicht die selten verwendete Funktion, schon existierende Fenster bei seinem Aufruf zu erkennen und anzuordnen. Diese Funktion bräuchte man für den mehr oder weniger sinnvollen Wechsel des Fenstermanagers einer schon laufenden X-Sitzung.

- Ein umfangreiches System zum Einstellen von Fenstereigenschaften wie Rahmen- oder Hintergrundfarbe wäre praktisch.

- Man könnte die eigene Konfigurationssprache durch eine Skriptsprache ersetzen – diesen Weg sind Ion (mit der Sprache Lua) und Xmonad (mit der Sprache Haskell) gegangen. Man kann natürlich auch die eigene Konfigurationssprache weiterentwickeln, wie es der Fenstermanager Ratpoison gemacht hat.

An einem Fenstermanager lässt sich immer weiterprogrammieren und experimentieren: Er ist nie wirklich fertig, weil es immer noch weitere Funktionen gibt, die man einbauen könnte.

Literatur

[1] Eine kurze Geschichte des WIMP Interface. http://mark13.org/utf8/wimp/, 2008. [Online; Stand 23. Januar 2010].

[2] Wikipedia. Architecture of Windows NT, Abschnitt „User mode“ – Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Architecture_of_Windows_NT&oldid=319607949, 2009. [Online; Stand 18. Oktober 2009].

[3] Hans-Peter Oser. Architektur des X Window Systems. http://www.oser.org/~hp/bsyII/node63.html, 2007. [Online; Stand 18. Oktober 2009].

[4] The GNOME Projekt. Metacity. http://live.gnome.org/Metacity, 2009. [Online; Stand 20. Januar 2010].

[5] KDE Community. KWin. http://techbase.kde.org/index.php?title=Projects/KWin&oldid=39085, 2009. [Online; Stand 24. Januar 2010].

[6] Tuomo Valkonen. Ion. http://www.modeemi.cs.tut.fi/~tuomov/ion/, 2010. [Online; Stand 24. Januar 2010].

[7] Shawn Betts. ratpoison. http://www.nongnu.org/ratpoison/, 2009. [Online; Stand 24. Januar 2010].

[8] The xmonad developer team. xmonad. http://www.xmonad.org/, 2010. [Online; Stand 24. Januar 2010].

[9] suckless.org community. dwm dynamic window manager. http://dwm.suckless.org/, 2009. [Online; Stand 24. Januar 2010].

[10] Adrian Nye. Xlib Programming Manual. http://tronche.com/gui/x/xlib/, 1992. [Online; Stand 12. November 2009].

[11] Matthew Allum. Xephyr. http://www.freedesktop.org/wiki/Software/Xephyr, 2004. [Online; Stand 24. Januar 2010].

[12] Wikipedia. Stapelspeicher, Abschnitt Postfixnotation und stapelorientierte Sprachen – Wikipedia, Die freie Enzyklopädie. http://de.wikipedia.org/w/index.php?title=Stapelspeicher&oldid=69525643, 2010. [Online; Stand 24. Januar 2010].

[13] X Desktop Group. Extended window manager hints. http://standards.freedesktop.org/wm-spec/wm-spec-latest.html, 2003. [Online; Stand 24. Januar 2010].