RZTU

Rechenzentrum der
Technischen Universität Hamburg-Harburg

Einführung in die Benutzung des X–Window-Systems

unter dem Betriebssystem UNIX

Dokumentationsschlüssel: BEN 2

Version M

2. Auflage, Oktober 1993

Herausgeber	:	Technische Universität Hamburg-Harburg – Rechenzentrum – Denickestr. 17 21073 Hamburg
Verantwortlich	:	Peter Junglas, Tel.: 3193

Inhaltsverzeichnis

1 Konzeption des X-Window-Systems 3

1.1 Geschichtliches 3

1.2 Grund-Konzepte 3

1.3 Programmierung von X-Clients 4

1.4 Window-Manager 5

2 Benutzung der X-Window-Oberfläche 5

2.1 Vorbereitungen 5

2.2 Bedienung des twm-Window-Managers 6

2.3 Benutzung anderer Rechner 9

2.4 Startup von anderen Maschinen 10

3 Standard-X-Window-Programme 11

3.1 xterm 11

3.2 xedit, xclipboard 13

3.3 xman 15

3.4 xhost 15

3.5 xclock, oclock, xbiff, xcalc, xload 15

3.6 xwd, xwud, xpr, xdpr 17

3.7 xmag, bitmap 17

4 Anpassung der Oberfläche an eigene Vorstellungen 18

4.1 Fonts 19

4.2 Farben 22

4.3 Kommandozeilen-Optionen 23

4.4 Die Resourcen-Datenbank 23

4.5 Konfiguration des Window-Managers twm 26

4.6 Das X-Startup-File 28

4.7 Ändern der Tastenbelegung 29

5 Spezielle X-Clients **31**

5.1 Informationen über Clients und Server 31

5.2 Clients am RZTU 32

5.3 X-Schnittstellen einiger Programm-Pakete 35

6 Der Motif-Window-Manager (mwm) **39**

6.1 Benutzung des mwm 39

6.2 Konfiguration des mwm 40

6.2.1 Das .mwmrc-File 40

6.2.2 Resourcen für den mwm 41

7 Ausblick: Was bringt X11R5 ? **42**

8 Anhang A: Beispiel für ein .twmrc-File **43**

9 Anhang B: Beispiel für ein Startup-File **47**

10 Anhang C: Beispiel einer Key-Tabelle **48**

11 Anhang D: Das system.mwmrc-File **51**

12 Anhang E: Steuercodes des xedit und der Athena-Textwidgets **54**

Abbildungsverzeichnis

1 7

2 9

3 14

4 16

5 21

6 33

7 34

8 36

1 Konzeption des X-Window-Systems

1.1 Geschichtliches

Zu Beginn der 80er Jahre gab es am MIT eine ganze Reihe von Workstations, Minis und größeren Rechnern verschiedenster Hersteller und mit unterschiedlichen Betriebssystemen. Diese Vielfalt verhinderte, daß Resourcen von anderen Rechnern vom eigenen Arbeitsplatz aus genutzt werden konnten. Man setzte sich daher zum Ziel, ein System zu entwickeln, daß einen Zugriff auf Anwendungen auf anderen Rechnern in einer Fenster-Umgebung vom eigenen Graphik-Bildschirm aus erlaubte. Mit Unterstützung der Industrie entstand so daß X-Window-System. Es stieß bald auf allgemeines Interesse, so daß sich das MIT zur Gründung eines X-Consortiums entschloß, das die Einhaltung der Standards und die Weiterentwicklung von X überwachen sollte und dem inzwischen fast alle Computer-Hersteller angehören. Mit dem Aufkommen relativ preiswerter Workstations steht inzwischen die benötigte Hardware zur Verfügung, um X einzusetzen, so daß es sich nun zu einem Industrie-Standard entwickelt hat. Weite Verbreitung fand die Version 11, genannt X11, die inzwischen hauptsächlich in den Releasen 3 und 4 benutzt wird. Seit wenigen Monaten ist die 5. Release (X11R5) ausgeliefert worden und wird sicher bald allgemein vorhanden sein.

Der komplette Source-Code des X-Window-Systems wird vom MIT incl. ausführlicher Dokumentation und Beispiel-Programmen vertrieben und ist auf vielen File-Servern erhältlich.

1.2 Grund-Konzepte

Hardware-Voraussetzung für den Einsatz des X-Window-Systems ist ein Bildschirm, der Pixelgraphik darstellen kann, sowie eine Tastatur und eine Maus als Eingabegeräte. Auf der Maschine, die diese Geräte ansteuert, muß ein spezielles Programm laufen, der sogenannte X-Server. Er ist für die gesamte Ansteuerung der Ein- und Ausgaben zuständig, andere Programme können nur indirekt über ihn den Bildschirm ansprechen oder Eingaben von Maus und Keyboard bekommen. Der X-Server wartet auf Anforderungen von Anwendungs-Programmen (den sog. X-Clients) in einer speziellen "Sprache", dem X-Protokoll, und gibt Eingaben über Maus oder Keyboard gemäß dem X-Protokoll an andere Programme weiter. Beispiele von solchen Anforderungen wären etwa "Erzeuge ein Fenster" oder "Schreibe 'Juhu' in das Fenster 42", in umgekehrter Richtung z.B. "Maus wurde in Fenster 7 bewegt". Diese Kommunikation läuft asynchron, d.h. der jeweilige Sender wartet nicht auf den Empfänger; er kann i.a. nicht einmal feststellen, ob seine Nachricht angekommen ist. (Dies sicherzustellen ist Aufgabe "tieferer" Schichten der Kommunikations-Protokolle, wie etwa TCP/IP, auf deren Vorhandensein das X-Protokoll aufbaut.)

Oft stellt man sich unter einem Server eine entfernte Maschine vor, der X-Server aber läuft auf der eigenen, während die Clients lokal oder auf einem anderen Rechner laufen können. Dieser Bezeichnung liegt folgende Vorstellung zugrunde: Der X-Server stellt anderen die von ihm verwaltete Hardware (Bildschirm, Tastatur und Maus) zur Verfügung; die Clients

benutzen diese Dienste, sie sind "Kunden" des X-Servers.

Diese Entkopplung von Anwendungs-Programmen und Ein-/Ausgabe-Hardware garantiert zwei der wesentlichen Zielsetzungen: Hersteller-Unabhängigkeit und Netzwerk-Transparenz. Dem X-Server ist es egal, auf welchem Rechner und unter welchem Betriebssystem die Clients laufen, solange sie sich des X-Protokolls bedienen. Für Anwendungs-Programmierer bedeutet dies, daß sie Graphikausgabe und Mausbenutzung hardware-unabhängig programmieren können. Workstation-Hersteller auf der anderen Seite garantieren durch Bereitstellen eines (hardware-abhängigen !) X-Servers, daß eine große Anzahl von Anwendungen benutzt werden kann.

1.3 Programmierung von X-Clients

Um Anwendungen zu schreiben, die das X-Window-System benutzen, gibt es eine Menge von Möglichkeiten zunehmenden Abstraktionsgrads:

Die elementare Methode (die nie verwendet wird) ist, über eine Netzwerk-Schnittstelle (Socket, Stream) Kontakt mit einem X-Server aufzunehmen und direkt Daten im Format des X-Protokolls zu senden.

Die nächste Ebene darüber stellt die Xlib-Library dar. Sie enthält etwa 300 C-Funktionen, über die alle Funktionen des X-Servers aufgerufen werden können. Sie enthält z.B. Funktionen zur Kontaktaufnahme mit einem X-Server, zum Öffnen eines Fensters, zum Zeichnen von Graphik oder zum Empfangen von Eingaben. Es sind aber keine "höheren" Funktionen wie Menüs oder Scrollbars vorhanden. Dies würde der Philosophie der X-Window-Implementierer zuwiderlaufen, keinerlei geschmacksabhängige Elemente (wie sehen Scrollbars, Menüs etc. aus?) in die Basis-Library einzubauen.

Die nächsthöhere Ebene der X-Window-Programmierung baut auf den Ideen der objektorientierten Programmierung auf: Man arbeitet mit ganzen Hierarchien von fertigen oder selbstdefinierten Objekten (bei X "Widgets" genannt) wie z.B. Scrollbars, Menüleisten, Knöpfen, Auswahlfenstern usw., die "selbst" dafür sorgen, daß sie bei Bedarf neu gezeichnet werden, oder die ihre Unterfenster selbst sinnvoll anordnen usw. Da aber alle X-Libraries in C geschrieben sind, einer Sprache, die keine Objekte kennt, wurden in einem ersten Schritt Funktionen zur Verfügung gestellt, die - bei richtiger Anwendung - objektorientierte Konstruktionen ermöglichen. Diese Bibliothek heißt "Toolkit Intrinsics Library", libXt.a. Sie bietet die Grundfunktionen objektorientierter Programmierung – unter besonderer Berücksichtigung der Erfordernisse des X-Window-Systems –, enthält selbst aber keine Objekt- (Widget-) Definitionen. Dazu kommt eine weitere Bibliothek, die "Athena Widgets Library", die einige grundlegende Widgets zur Verfügung stellt.

Weitere Widget-Pakete, die jeweils ein spezielles Aussehen der Fenster und ihrer Teile vorgeben, werden von verschiedenen Herstellern teils umsonst, teils gegen Gebühr vertrieben. Dazu gehören das OSF/Motif-Toolkit, das dreidimensionale Knöpfe enthält, und das Open-Look-Toolkit, das die Basis für die SUN-typische Oberfläche ist.

Auch mit diesen Toolkits ist das Programmieren von benutzerfreundlichen Oberflächen immer noch eine aufwendige Arbeit. Daher wurden Systeme entwickelt, die helfen sollen, die Benutzer-Schnittstelle eines Programms unabhängig von dessen sonstiger Funktion graphisch zu erstellen. Eine weitere Entwicklung sind Bibliotheken, die erlauben, dieselbe Oberfläche in einer X-Window-Umgebung, für den Presentation-Manager von OS/2 und für MS-Windows 3.0 zu verwenden.

1.4 Window-Manager

Um mit einem Window-System arbeiten zu können, muß es möglich sein, Fenster zu verschieben, in der Größe zu verändern, ggf. momentan beiseite zu legen (als Icons o.ä.) usw. Da sich solche Funktionen nicht ohne Festlegungen von Stilfragen implementieren lassen (Rahmen, Knöpfe, Menüs, ...), wurden sie nicht im X-Server implementiert. Stattdessen werden spezielle X-Clients, sogenannte Window-Manager, verwendet, die diese Aufgaben übernehmen. Wie normale Clients kommunizieren sie mit dem X-Server, allerdings nehmen sie die Fenster anderer Anwendungen unter ihre Kontrolle (bzgl. Größe, Lage usw., nicht bzgl. des Inhalts). Ein Window-Manager legt damit auch fest, was für ein Rahmen um ein Fenster gezeichnet wird, welche Knöpfe und Menüs es gibt usw., d.h. er implementiert einen bestimmten Stil (neuerdings "Look and Feel" genannt). Zum X-Window-System gehört der twm (Tom's Window Manager), der ein relativ "neutrales" Erscheinungsbild liefert. Da er umsonst und im Verbrauch von Resourcen einigermaßen bescheiden, ist er bei uns (und an vielen anderen Orten) noch der Standard. Allmählich breiten sich allerdings zwei weitere Window-Manager aus: der mwm (Motif Window Manager), der zum Motif-Toolkit paßt, und der olwm (Open Look Window Manager), der auf Suns verwendet wird.

Ein weiterer spezieller Client ist der Session-Manager, der - meist über Menüs - andere X-Clients startet. Allerdings wird diese Funktion von den oben angeführten Window-Managern (twm, mwm, olwm) mit übernommen.

2 Benutzung der X-Window-Oberfläche

2.1 Vorbereitungen

Um X auf den Apollos oder den Silicons zu starten, sind einige Vorbereitungen zu treffen, die bei neuen Benutzern z.Z. schon vom Rechenzentrum erledigt wurden:

1. Man benötigt folgende drei Files im eigenen Home-Directory: **.xinitrc**, **.twmrc**, **.Xdefaults**. Falls sie noch nicht existieren, kann man Beispiele dafür aus dem Directory **/usr/tuhh/skel** (auf Apollos oder Silicons) kopieren.
2. Je nachdem, ob man immer mit X arbeiten möchte oder ob man auch das andere Windowsystem der Workstation (Domain Display-Manager auf den Apollos, NeWS

auf den Silicons) benutzen will, kann man sich entweder im .login-File oder in einem extra File ein X-Startkommando einbauen. Im Kurs wird dafür ein spezielles File .login.X11 benutzt, das nach einer Abfrage beim Einloggen gestartet wird.

3. Der Such-Pfad, der normalerweise im File **.cshrc** gesetzt wird, muß das Directory **/usr/bin/X11** enthalten, damit die X-Programme (Clients) gefunden werden.

Nachdem man so das X-Window-System gestartet hat, erscheinen auf dem Bildschirm einige Fenster: in der linken oberen Ecke ein Terminalfenster, rechts oben eine Uhr und darunter ein kleiner Briefkasten. Außerdem ist darunter ein Feld mit dem Namen des Terminalfensters zu sehen, das wir im Moment noch ignorieren. Die drei anderen Fenster gehören zu drei verschiedenen X-Clients, nämlich **xterm**, **xclock** und **xbiff**. Mit den Funktionen und Eigenschaften von einigen wichtigen Clients werden wir uns später ausführlich beschäftigen; für den Augenblick reiche uns ihre unmittelbare Funktion: xclock zeigt die Zeit an, xbiff wird schwarz, wenn Mail ankommt, und xterm stellt ein normales Terminal zur Verfügung (genauer: ein VT100). Außerdem ist noch ein weiteres wichtiges Fenster zu sehen: das - graugemusterte - Hintergrund- oder Root-Fenster.

2.2 Bedienung des twm-Window-Managers

Ein weiterer wichtiger Client, der ebenfalls automatisch gestartet wurde, ist der Window-Manager, und zwar hier der twm. Er ist u.a. zuständig für die "Fenster-Verzierung" um das xterm-Fenster, die wir uns jetzt genauer ansehen wollen:(s. Abb. 1) Am oberen Rand befindet sich die Titelzeile mit dem Namen des Fensters (hier gleich dem Namen der Maschine). Wenn wir die Maus in das xterm-Fenster bewegen, ändert die Titelzeile ihre Farbe, um anzuzeigen, das dieses Fenster aktiv ist. Gleichzeitig wird der sonst hellgraue Rahmen um das ganze Fenster schwarz. Nur wenn ein Fenster aktiv ist, werden Tastatureingaben an den entsprechenden Client weitergereicht. Auf diese Weise ist garantiert, daß bei vielen Fenstern auf dem Bildschirm genau eins mit der Tastatur "verbunden" ist. Das aktive Fenster muß dabei nicht ganz zu sehen sein; es reicht, wenn sich die Maus irgendwo in diesem Fenster befindet. Die Titelzeile dient auch zum Verschieben eines Fensters. Dazu bewegt man die Maus in diese Zeile, drückt die linke Maustaste und hält sie gedrückt. Es erscheint ein Rahmen um das Fenster und ein Gitternetz. Dieser Rahmen folgt der Mausbewegung, solange man die Maustaste gedrückt läßt; beim Loslassen springt das ganze Fenster an die so markierte neue Position. Hat man sich während des Verschiebens entschlossen, das Fenster doch an der alten Position zu lassen, drückt man eine weitere Maustaste und läßt dann alle los, der Vorgang wird dann abgebrochen.

Das kleine Kästchen in der rechten oberen Ecke dient zum Vergrößern und Verkleinern des Fensters. Beim Drücken der linken Maustaste in diesem Kästchen erscheint wieder ein Rahmen mit Gitternetz, der sich diesmal beim Bewegen der Maus entsprechend vergrößert und verkleinert. Dabei muß man sich zum Verkleinern in einer Richtung (horizontal oder vertikal) zunächst in die entgegengesetzte Richtung bewegen, d.h. vergrößern, bis der Rahmen folgt. Dieses Verhalten ist nützlich, wenn man die Größe nur in einer Dimension verändern will. Bewegt man sich beim Verkleinern über die Grenzen des ursprünglichen Fensters hinaus,

Abbildung 1:

klappt der Rahmen um und man vergrößert jetzt in dieser Richtung. Auf diese Weise ist es möglich, nicht nur nach rechts und oben, sondern in jede Richtung zu vergrößern. Durch Betätigen einer zusätzlichen Maustaste kann man wieder den Vorgang abbrechen.

Um weitere Funktionen des Window-Managers am Beispiel ausprobieren zu können, starten wir ein weiteres xterm-Fenster, indem wir im vorhandenen Fenster den Befehl "**xterm &**" eingeben. Kurz darauf erscheint ein weiteres Terminalfenster, das das alte wahrscheinlich größtenteils überdeckt. Um ein (teilweise) verdecktes Fenster ganz nach vorne oder ganz nach hinten zu bringen, ohne es zu verschieben (insbesondere bei vollem Schirm sehr nützlich), gibt es mehrere Möglichkeiten: Klicken mit der linken Maustaste auf den seitlichen oder unteren Rand eines Fensters bringt es nach oben, mit der rechten Maustaste nach unten. In der Titelzeile bewirkt ein Klicken mit der mittleren Maustaste, daß das Fenster ganz nach hinten geht, wenn es vorne ist, sonst kommt es nach vorne.

Wenn die Zahl der Fenster, mit denen man arbeitet, größer wird, möchte man oft einige zeitweilig loswerden, ohne die Clients abzubrechen. Dazu gibt es die Möglichkeit, Fenster in kleine Bildchen, sogenannte Icons, zu verwandeln. Ein solcher Client arbeitet ganz normal weiter, allerdings kann er keine Eingaben mehr bekommen. Dazu müßte man ihn erst wieder zu voller Größe erwecken. Zum "Iconisieren" dient der kleine Knopf in der linken oberen Ecke: Klickt man ihn mit der linken Maustaste an, verschwindet das Fenster. Dafür wird in der Liste am rechten Rand (dem Icon-Manager) der Eintrag des entsprechenden Fensters durch ein "X" markiert. Durch Klicken auf diesen Eintrag mit der linken Maustaste bringt man das Fenster wieder auf den Bildschirm. Wahlweise (nach Ändern von Konfigurationsdateien, s.u., Kap. 4) kann man statt des Icon-Managers auch für jedes iconisierte Fenster ein eigenes Bildchen bekommen, das Icon, das man ebenfalls durch Klicken mit der linken Maustaste wieder zu voller Größe erweckt.

Übrigens kann man ein beliebiges Fenster auch zu voller Bildschirmgröße aufblasen - und wieder auf altes Format zurück, und zwar durch gleichzeitiges Drücken der Shift- und der linken Maustaste.

Vielleicht ist inzwischen schon aufgefallen, daß es zwei Clients gibt, die weder eine "Dekoration" des Window-Managers haben (d.h. Rahmen und Titelzeile) noch im Icon-Manager auftauchen, nämlich die Uhr und der Briefkasten. Dies ist so eingestellt, weil man diese Clients in der Regel auch nicht verändern möchte. Will man dies aber trotzdem tun, kann man sich des twm-Menüs bedienen:

Drückt man mit der linken Maustaste ins Root-Fenster (den Hintergrund) und hält die Taste gedrückt, so erscheint ein Menü mit elf Einträgen in vier Gruppen, und der Mauszeiger verwandelt sich in einen horizontalen Pfeil (s. Abb. 2). Die erste Gruppe enthält sechs Einträge, mit denen man Window-Manager-Funktionen für ein Fenster auswählen kann. Dazu bewegt man den Mauszeiger (bei gedrückter linker Taste) nach unten in den gewünschten Eintrag, der dabei farblich gekennzeichnet wird. Läßt man nun die Maustaste los, verwandelt sich der Zeiger wiederum, diesmal in einen kleinen Kreis. Nun kann man (natürlich wieder durch Anklicken mit der linken Maustaste) ein Fenster auswählen, um die gewünschte Funktion damit auszuführen. Auf diese Weise läßt sich jedes Fenster iconisieren ("Iconify"), vergrößern oder verkleinern ("Resize"), verschieben ("Move"), nach vorne oder hinten bringen ("Rai-

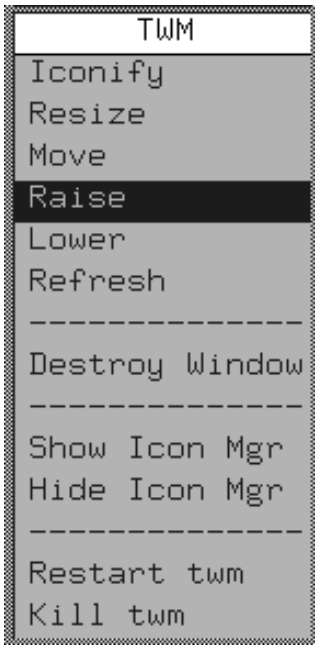

Abbildung 2:

se", "Lower") oder sein Inhalt neu aufbauen ("Refresh"). Auf die gleiche Weise kann man auch ein Fenster samt dazugehörigem Client vernichten ("Destroy Window"). Die weiteren Menüpunkte dienen dazu, das Fenster des Icon-Managers verschwinden zu lassen oder es wieder hervorzuholen ("Show/Hide Icon Mgr"), den Window-Manager (z.B. nach Neukonfigurierung, s.u., Kap.4) neu zu starten oder ganz zu beenden. Letzteres ist in der Regel nicht nötig, da der twm - anders als etwa der mwm - mit dem Login-Fenster zusammen verschwindet.

Auf der mittleren und rechten Maustaste im Root-Fenster liegen zwei weitere Menüs, mit denen man in gleicher Weise (Maustaste festhalten und herunterziehen) weitere X-Clients starten oder sich auf andere Rechner einloggen kann. Einzelheiten über diese und weitere Clients folgen im Kapitel 3.

2.3 Benutzung anderer Rechner

Eine wichtige Eigenschaft des X-Window-Systems ist die Netzwerk-Transparenz, d.h. es spielt keine Rolle, auf welcher Maschine ein Client rechnet. Allerdings sind dafür zunächst noch Vorbereitungen zu treffen. Als Beispiel gehen wir im folgenden davon aus, daß wir vor einer Apollo, etwa der ap07 sitzen, auf der X gestartet wurde, und uns in einem xterm-Fenster mit telnet auf die C1 eingelogged haben.

Zunächst muß man dem X-Server mitteilen, daß man Clients von der Convex aus starten möchte. Dazu setzen wir auf der ap07 (entweder in einem zweiten xterm-Fenster oder vor dem telnet) das Kommando

xhost tuhhco.rz

ab. Dies ist nötig, da der Server sonst Anfragen von Clients anderer Maschinen ignoriert. Schließlich möchte man nicht, daß einem jeder etwas auf den Schirm schicken kann. Allerdings ist dieses Autorisierungsverfahren sehr primitiv: Nach dem xhost-Befehl können prinzipiell alle Benutzer von der Convex Kontakt mit dem Server der ap07 aufnehmen.

Als nächstes muß man dafür sorgen, daß die Ausgaben von X-Programmen der Convex zum X-Server der Apollo umgelenkt werden. Dazu setzen wir die Environment-Variable DISPLAY wie folgt:

setenv DISPLAY ap07.cip3a:0.0

Startet man jetzt einen X-Client auf der Convex, weiß dieser, daß er seine X-Anforderungen an den Bildschirm 0 des X-Servers 0 auf der Maschine "ap07.cip3a" schicken soll. Normalerweise hat man auf einer Workstation nur einen X-Server laufen, der auch nur einen Bildschirm bedient. In diesem Fall kann man die Endung ".0" weglassen, ":0" muß aber immer vorhanden sein.

Nun kann auf der Convex irgendein Client gestartet werden, etwa

xcalc &

und schon erscheint der Taschenrechner auf dem Bildschirm. Die eigentlichen Berechnungen erfolgen jetzt auf der Convex, die Ein- und Ausgaben erledigt der X-Server auf der Apollo.

2.4 Startup von anderen Maschinen

Im obigen Beipiel gingen wir davon aus, daß auf der Apollo schon ein X-Server läuft und daß man problemlos weitere Clients (etwa ein xterm) starten kann. Es gibt nun andere Maschinen, auf denen diese Voraussetzungen nicht erfüllt sind. In diesem Fall sind einige weitere Vorkehrungen zu treffen:

Auf den Apollos (unter Domain OS 10.3) und den Silicons (unter IRIX System V.3) wird der X-Server automatisch gestartet (meist neben dem eigenen Window-System der Maschine). Auf anderen Systemen oder älteren Betriebssystem-Versionen muß ein X-Server - falls vorhanden - explizit gestartet werden. Dafür gibt es das Programm xinit, das auch gleich ein xterm als Client dazutut, damit man weitere Kommandos absetzen kann. Das Verhalten von **xinit** kann über Konfigurationsdateien angepaßt werden. Näheres dazu entnehme man bei Bedarf dem Manual (**man xinit**).

Ein anderes Problem entsteht auf Rechnern, die außer dem X-Server keine weiteren Programme starten können, etwa auf PCs unter DOS, die nur ein Programm auf einmal abarbeiten können - eben den X-Server -, oder auf X-Terminals, speziellen Rechnern, die über die nötige Hardware für X verfügen und auf denen gar kein Betriebssystem läuft, sondern überhaupt nur der X-Server. Die Frage ist: Wie startet man hier ein erstes xterm auf einem anderen Rechner? (Von diesem ausgehend könnte man dann ja auch andere Clients, ggf. auch auf anderen Maschinen starten.) Dazu sind verschiedene Lösungen entwickelt worden; die drei wichtigsten möchte ich kurz vorstellen:

Eine Möglichkeit ist es, den X-Server zusätzlich zu seinen normalen Funktionen noch mit der Fähigkeit auszustatten, eine Telnet-Sitzung ohne X zu starten. Meist geht man hier in einen speziellen Setup-Mode, bekommt eine primitive Terminalemulation vorgesetzt und loggt sich damit auf irgendeinem Rechner ein. Hier kann man dann, nach den in Abschnitt 2.3 beschriebenen Vorbereitungen, Clients - etwa ein xterm - starten, die telnet-Sitzung beenden und dann ganz unter X weitermachen.

Ganz innerhalb des X-Window-Systems bleibt ein anderes Verfahren, das auf einem speziellen Client beruht, dem X-Display-Manager, kurz XDM, der extra für diesen Zweck entwickelt wurde. Der XDM kann auf irgendeinem Rechner im Netz laufen und wartet, bis sich ein X-Terminal oder ein anderer X-Server bei ihm anmeldet (mit einem speziellen Protokoll). Auf dessen Bildschirm schickt er nun ein Fenster mit einer Login-Aufforderung, in dem man sich wie üblich anmelden kann. Anschließend wird ein Startup-File des Benutzers durchlaufen (~/.xsession), mit dem spezielle Clients gestartet werden, darunter normalerweise auch ein xterm. Ist kein solches File vorhanden, wird aber zumindest ein xterm erzeugt, von dem aus man weitermachen kann.

Speziell bei X-Terminals gibt es noch ein weiteres Verfahren: Die Prozessoren dieser Terminals sind meist problemlos in der Lage, außer dem X-Server noch einige Clients zu unterhalten. Daher wurden spezielle abgemagerte xterm-Versionen für sie geschrieben (oft “xtelnet” genannt), die das oben beschriebene Telnet-Verfahren in ein normales X-Fenster integrieren. Mittlerweile werden sogar schon ganze Window-Manager in solche X-Terminals integriert.

3 Standard-X-Window-Programme

In diesem Kapitel sollen die wichtigsten Clients vorgestellt werden, die zum Standard-Umfang von X-Windows gehören.

3.1 xterm

Der xterm-Client erzeugt ein Terminal vom Typ DEC VT102 in einem eigenen Fenster (s. Abb. 1). Dies ist sicherlich eine der am häufigsten benutzten Anwendungen. Von einem xterm aus kann man weitere Clients starten und ganz normale (Shell-)Kommandos absetzen. Beim Übergang von X11R3 zu X11R4 haben einige Änderungen am xterm stattgefunden.

Hier wird das Release-4-xterm beschrieben, wie es z.Zt. auf allen Maschinen außer den Apollos verfügbar ist. Der Umgang mit dem veralteten Release-3-xterm (z.Zt. auf den Apollos) sollte dann auch keine Schwierigkeiten machen.

Das xterm enthält vier Menüs, von denen man die ersten drei erreicht, indem man (bei aktivem xterm - Fenster) die Control-Taste und eine der Maustasten drückt und festhält. Die wichtigsten Menüpunkte werden i.f. kurz vorgestellt, alles weitere kann man der Manual-Seite (**man xterm**) entnehmen. Die Menüs sind durch Linien in mehrere Teile geteilt. Im obersten Teil stehen jeweils Optionen, die man mit der Maus an- und ausschalten kann. Eine ausgewählte Option wird mit einem Häkchen markiert.

Mit Control und der linken Maustaste erhält man das "Main Options"-Menü. Interessant ist hier der "Log to File"-Eintrag: Wird dieser Menüpunkt angewählt, werden alle Ein- und Ausgaben in diesem Fenster in eine Datei namens XtermLog.nnnnn geschrieben, wobei nnnnn die Prozeß-ID des xterms ist.

Mit <CONTROL>-"mittlere Maustaste" erhält man das "VT-Options"-Menü, das zunächst eine ganze Anzahl von Optionen enthält, die direkt die Eigenschaften des VT102 beeinflussen. Mit "Enable Scrollbar" kann man einen Scroll-Balken an- und ausschalten, der am linken Rand erscheint und mit dem man sich frühere Schirminhalte ansehen kann. Ein grauer Schieber zeigt durch seine Länge und Position an, wie groß der Bildschirm-Ausschnitt ist und welchen Teil man gerade sieht. Befindet sich der Mauszeiger im Scroll-Balken, kann man mit der linken Maustaste nach unten blättern, mit der rechten nach oben, und mit der mittleren direkt den gewünschten Ausschnitt angeben. Die unteren drei Menü-Einträge dienen zum Umschalten auf die Tektronix-Emulation: Wahlweise kann man beide Fenster (VT102 und Tektronix 4014) gleichzeitig darstellen - allerdings ist nur eines aktiv - oder nur eines. "Show Tek Window" bringt das Tektronix-Fenster auf den Bildschirm, "Switch to Tek Mode" macht es aktiv, und "Hide VT Window" verbirgt das VT102-Fenster, falls das Tektronix-Fenster sichtbar ist. Ist das nicht der Fall, ist der letzte Menü-Eintrag gesperrt, d.h. hellgrau dargestellt.

<CONTROL>-"rechte Maustaste" läßt das "VT Fonts"-Menü erscheinen. Hier kann man einen von mehreren Fonts verschiedener Größe ("Tiny", "Small", "Medium", "Large") auswählen oder zum Ausgangsfont zurückgehen ("Default"). Das xterm-Fenster paßt sich dabei in der Größe an. Der tiny-Font ist unlesbar klein; er dient dazu, das xterm quasi als "aktives Icon" zu benutzen. Die beiden anderen Menüpunkte werden zur Auswahl weiterer Fonts verwendet; sie werden in Kapitel 4 besprochen.

Im Tektronix-Fenster erhält man mit <CONTROL>-"mittlere Maustaste" das "Tek Options"-Menü. Hier kann man verschiedene Fonts auswählen, mit "PAGE" den Bildschirm löschen, mit "COPY" die bisher ausgeführten Tektronix-Kommandos in ein File namens COPY "datum" schreiben und wieder zum VT102 zurückschalten.

Eine weitere Möglichkeit, die auch viele andere Clients erlauben, ist die Copy-und Paste-Funktion: Mit gedrückter linker Maustaste kann man einen beliebigen Text markieren, er wird dann invers dargestellt. Zweimaliges Klicken mit links wählt ein ganzes Wort aus, dreimaliges eine Zeile. Mit der rechten Maustaste kann man den ausgewählten Bereich noch

vergrößern. Durch Klicken mit der mittleren Maustaste wird der ausgewählte Text kopiert, und zwar beim xterm immer direkt in die Kommandozeile.

Außer mit den Menüs läßt sich das xterm auch durch Kommandozeilen-Optionen konfigurieren. So kann man z.B. mit -l bzw. +l das Mitprotokollieren ein- oder ausschalten (letzteres ist der Default), mit -sb den Scrollbalken einschalten und mit "-sl nummer" die Anzahl der Zeilen angeben, die gespeichert werden (Default: 64). Mit der Option "-e kommando [argumente]" als letzter Option in einem Aufruf wird ein xterm erzeugt, in dem gleich das angegebene Kommando mit eventuellen Parametern ausgeführt wird. Nach Beendigung des Kommandos verschwindet das xterm-Fenster wieder. Normalerweise wird sonst in einem xterm eine Shell (bei uns die csh) gestartet. Nach Beenden dieser Shell (mit logout, exit, Control-D) oder mit dem Quit-Eintrag im "Main Options"-Menü verschwindet das xterm-Fenster.

Weitere Optionen und Konfigurations-Möglichkeiten werden im vierten Kapitel vorgestellt.

3.2 xedit, xclipboard

Der xedit ist ein einfacher Text-Editor, der die wichtigsten Editier-Funktionen enthält. Er wird aufgerufen mit "**xedit** *[filename]* **&**", wobei der Filename optional ist. Es erscheint ein mehrfach geteiltes Fenster mit drei Textbereichen (s. Abb. 3). Um Texte zu ändern, muß sich der Mauszeiger im entsprechenden Feld befinden. Die Größe der einzelnen Bereiche läßt sich durch Schieber (dargestellt als schwarze Quadrate) mit der Maus verändern. Die obere Zeile enthält drei "Knöpfe" mit den Bezeichnungen "Quit", "Save" und "Load", die durch Anklicken mit der linken Maustaste bedient werden. Daneben steht in einem Textfenster der Filename, falls er vorgegeben wurde. Hier kann man einen neuen Namen eintragen, um den Text in ein anderes File zu speichern oder um ein neues File zu laden. Das nächste Textfeld dient zur Ausgabe von Meldungen des xedit. Schließlich befindet sich im größten Feld der zu editierende Text. In jedem Feld steht ein Textcursor, der durch ein "^" dargestellt wird.

Die wichtigsten Funktionen werden durch Tasten-Kombinationen wie beim Emacs aufgerufen, z.B. setzt man mit <Control-a> den Textcursor an den Anfang einer Zeile, mit <Control-e> ans Ende, mit <Control-k> löscht man vom Textcursor bis ans Ende der Zeile. Suchen und Ersetzen kann man mit <Control-s> bzw. <Control-r>. Weitere Funktionen sind mit der Maus zu erreichen: Der Textcursor läßt sich durch Klicken mit der linken Maustaste positionieren. Außerdem steht die im xterm-Abschnitt beschriebene Copy- and Paste-Funktion zur Verfügung. Zusätzlich ist es möglich, markierten Text mit <Control-w> zu löschen. Weitere Funktionen findet man im Anhang E, welche, da es sich bei den Textfeldern des xedit um Athena-Widgets handelt, auch in anderen X-Programmen, die diese Widgets verwenden, benutzt werden können.

Das xclipboard ist ein kleiner Notizblock (s. Abb. 3): Es zeigt einen Textbereich an, in dem wie beim xedit editiert oder mit der Maus Text zwischen Clients ausgetauscht werden kann. Mit einem "New"-Knopf können weitere Notizzettel erzeugt werden, zwischen denen man mit "Next" und "Previous" hin- und herschalten kann. Schließlich kann man mit "Delete"

Abbildung 3:

überflüssige Seiten wieder löschen.

Sowohl beim xedit als auch beim xclipboard wird automatisch ein Scrollbalken eingeblendet, wenn der Text im vorgegebenen Bereich nicht mehr dargestellt werden kann.

3.3 xman

Der xman ist ein Client, mit dem man komfortabel Manual-Seiten lesen kann. Defaultmäßig erscheint nach "**xman &**" ein kleines Fenster, mit dem man eine "Help"-Funktion aufrufen oder den eigentlichen "Leser" starten kann. (Dieses Startfenster kann man auch unterdrücken, s. Kap. 4). Der "Leser" zeigt zunächst ein großes Textfenster mit Hilfstext und zwei Knöpfen, "Options" und "Sections" (s. Abb. 4). Dahinter verbergen sich zwei Menüs, die wie üblich mit der linken Maustaste aufgeklappt werden. Unter "Sections" kann man ein Kapitel des UNIX-Handbuches auswählen (meist braucht man Nr. 1, "User Commands"), mit "Options" kann man sich u.a. ein Inhaltsverzeichnis anzeigen lassen, das alle Einträge in diesem Kapitel enthält und in dem man den gesuchten Begriff durch Anklicken auswählen oder zur Manual-Seite zurückgehen kann. Außerdem ist es möglich, beide Seiten gleichzeitig zu sehen, das Fenster wird dazu geteilt ("Show Both Screens"). Wie beim xedit kann man das Größenverhältnis der beiden Fenster zueinander wieder mit der Maus verändern.

3.4 xhost

Das xhost-Kommando ist oben schon angesprochen worden, es steuert die Zugangsberechtigung von Rechnern zum X-Server. Mit **xhost** erhält man eine Liste aller zugelassenen Maschinen, mit **xhost** *hostname* kann man einen weiteren Rechner zulassen, **xhost** *-hostname* streicht ihn wieder aus der Liste.

3.5 xclock, oclock, xbiff, xcalc, xload

Die folgenden fünf kleinen Clients sind teils nützlich, teils Spielerei:

xclock [-digital] bringt eine Analog- oder Digitaluhr auf den Schirm (s. Abb. 1).

oclock ist eine ovale Uhr (ab Release 4) (s. Abb. 3).

xbiff ist ein kleiner Briefkasten, der anzeigt, wenn Mail ankommt (s. Abb. 1).

xcalc ist ein mausbedienbarer Taschenrechner, wahlweise ein TI-30 (default) oder ein HP-10C (Option -rpn). Tip: Beenden mit der rechten Maustaste auf der AC- bzw. ON-Taste (s. Abb. 4).

xload zeigt graphisch die Auslastung ("Load") einer Maschine an (s. Abb. 3).

Abbildung 4:

3.6 xwd, xwud, xpr, xdpr

Die nächsten vier Clients dienen dazu, Fensterinhalte auf Platte oder Drucker zu bekommen.

Mit **xwd** ("X Window Dump") kann der Inhalt eines Fensters in einem speziellen X-spezifischen Format in ein File geschrieben werden. Nach dem Aufruf "**xwd -out** *filename*" erscheint ein Kreuz als Cursor. Dieses bewegt man in das gewünschte Fenster und klickt mit der linken Maustaste. Zur Bestätigung ertönt die Glocke zunächst einmal, nach Beendigung des Dump-Vorgangs noch zweimal. Wird das ausgewählte Fenster teilweise überlappt, wird der entsprechende Ausschnitt des darüberliegenden Fensters mitgedumpt, man muß also das gewünschte Fenster ganz nach vorne bringen. Durch Auswahl des Hintergrund-("Root"-)Fensters kann der gesamte Bildschirm gedumpt werden.

Um eine Datei im Dump-Format wieder als Fenster sichtbar zu machen, benutzt man das xwud-("X Window UnDump"-)Programm in der Form "**xwud -in** *filename* **&**". Damit wird das Fenster wieder angezeigt, wobei man speziell für Farbbilder noch einige Optionen hat, um die Farben anzupassen.

Möchte man ein solches Dump-File ausdrucken, kann man das xpr-Programm verwenden. Ein typischer Aufruf ist etwa

xpr -device pjet -header "Mein Dump" file.dump | lpr -Ppaintjet -b

xpr verwandelt das File file.dump in das für den HP Paintjet spezifische Format, versieht das Bild mit der angegebenen Überschrift und gibt es auf dem Standard-Output aus, von wo es via Pipe zum Paintjet weitergeschickt wird. Weitere unterstützte Formate (devices) sind ljet (HP Laserjet), pjetxl (HP Paintjet XL) und ps (PostScript). Es gibt noch eine Reihe von Optionen, um die Lage des Bildes auf dem Papier zu beeinflussen.

Da man oft xwd, xpr und lpr zusammen benutzt, um eine Hardcopy von einem Fenster zu machen, wurde xdpr (X Dump and Print) geschrieben, das alles zusammenfaßt. Nach

xdpr -device pjet -Ppaintjet -header "Mein kleiner Dump"

wählt man wie bei xwd ein Fenster aus, das dann gedumpt, gewandelt und gedruckt wird.

Leider sind die Treiber für den Paintjet und auch für PostScript nicht gut und nutzen die Farbmöglichkeiten nicht richtig aus. Bessere Hardcopies erzeugt das Programm xgrab (s.u.).

3.7 xmag, bitmap

Mit xmag kann man sich Teile des Schirms vergrößert ansehen, wobei die Vergrößerung mit einer Option -mag eingestellt wird (s. Abb. 3). Der Default-Wert ist fünf. Außerdem

kann man mit der Option -source NxM die Größe des Ausschnitts festlegen. Hier ist die Grundeinstellung 64x64. Nach dem Aufruf erscheint ein Fadenkreuz, das mit einem kleinen Rahmen umgeben ist. Dieses verschiebt man zur gewünschten Stelle - der Rahmen gibt dabei den Ausschnitt an - und klickt mit der linken Maustaste. Es erscheint ein neues Fenster mit entsprechend vergrößertem Bild. Klickt man in diesem ein (vergrößertes) Pixel mit der linken Maustaste an, wird seine Position und der Farbwert in der obersten oder untersten Zeile des Fensters angezeigt. Klickt man mit einer anderen Maustaste, verschwindet das Fenster, das Fadenkreuz kommt wieder, und man kann eine neue Stelle auswählen. Mit "q", "Q" oder "Control-c" im Fenster beendet man das Programm.

An verschiedenen Stellen werden von X-Clients einfache Bildchen, sogenannte Bitmaps, benutzt, z.B. für Füllmuster, Icons oder Cursor. Bitmaps sind im Prinzip schwarz-weiß, allerdings kann man sie durch Setzen von Hinter- und Vordergrundfarbe zweifarbig machen. Mit dem Programm "bitmap" kann man eigene Bitmaps einfach erstellen.

Nach dem Aufruf "**bitmap** *filename [BxH]* **&**" erscheint ein Fenster, das drei Bereiche enthält (s. Abb. 4): Der eigentliche Zeichenbereich hat Breite B und Höhe H (Default: 16x16), hier kann man Pixel - dargestellt als schwarze Quadrate - mit den drei Maustasten setzen, invertieren und löschen. Unten rechts wird die Bitmap in Originalgröße normal und invertiert angezeigt. Am rechten Rand befinden sich Knöpfe, mit denen man diverse Hilfsfunktionen aufrufen kann, z.B. um Linien oder Kreise zu zeichnen oder um Rechtecke zu löschen, zu füllen oder zu verschieben. Möchte man eine Bitmap für einen eigenen Cursor erzeugen, muß man zusätzlich noch angeben, welcher Punkt durch diesen Cursor ausgewählt wird (der sogenannte "Hot Spot"). Schließlich kann man sein Werk abspeichern und den bitmap-Editor verlassen.

Die bitmaps werden als Fragmente von C-Code abgespeichert und können so bequem in eigene Programme eingebunden werden. Einige Clients lesen dieses Bitmap-Format direkt, z.B. kann man mit "xsetroot -bitmap name" seine Creation auf dem Root-Fenster sichtbar machen.

4 Anpassung der Oberfläche an eigene Vorstellungen

Beim Design des X-Window-Systems wurde von Anfang an auf Flexibilität Wert gelegt. Die Zahl der Parameter, die man beeinflussen kann, ist schon für die wichtigsten Standard-Anwendungen incl. Window-Manager kaum überschaubar. Es gibt Clients, deren äußeres Erscheinungsbild sich vollständig eigenen Vorstellungen anpassen läßt, andere dagegen - vor allem X-Window-Oberflächen für ältere Pakete - sind ganz festgelegt. In diesem Abschnitt sollen vor allem solche Parameter, die bei vielen Clients auftauchen, beschrieben werden, dazu kommen die wichtigsten Konfigurationsmöglichkeiten des twm und einige allgemeine Einstellungen.

4.1 Fonts

Fast für alle Clients ist es möglich, den oder die Zeichensätze ("Fonts") für die Darstellung von Texten auszusuchen. Dabei hat man eine große Auswahl, denn zum Standard-Umfang von X11R4 gehören 474 Fonts. Damit diese - und zukünftige weitere - eindeutig bezeichnet werden können, hat man sich ein umfassendes, aber kompliziertes Benennungsschema ausgedacht. Jeder Font wird durch 14 Parameter gekennzeichnet, nämlich:

Name	Bedeutung	Beispiele
foundry	Firma, von der der Font kommt	("adobe", "sony")
family	Font-Familie, der "eigentliche" Stil	("helvetica", "times")
weight	Stärke	("bold", "medium")
slant	Neigung	("roman", "italic")
set width	Breite	("condensed")
style	nähere Stilbezeichnung	("nil", "sans serif")
pixel size	Größe in Pixel	(6, 12, 33)
point size	Größe in Points	(60, 90, 240)
resolution x/y	Anzahl der Points pro Inch in x/y-Richtung	(75, 100)
spacing	fest oder proportional	(p, m, c)
average width	mittlere Zeichenbreite in pixel/10	(44, 119, 206)
registry	Zeichensatz-Standard	(iso8859, sunolcursor)
encoding	Zeichensatz-Kennzeichen	(0, 1, dectech)

Die genaue Bezeichnung eines Fonts setzt sich aus Werten für alle Parameter zusammen, jeweils durch "-" getrennt. Damit entstehen so schöne Fontnamen wie

```
-adobe-helvetica-bold-r-normal--12-120-75-75-p-70-iso8859-1
-b&h-lucida-medium-i-normal-sans-20-140-100-100-p-114-iso8859-1
-sony-fixed-medium-r-normal--16-120-100-100-c-80-jisx0201.1976-0
```

Zum Glück sind die meisten Bestandteile eines Namens in der Regel entbehrlich. Oft kommt man schon mit der Auswahl der Parameter family, weight, slant und size (pixel oder point) zu einem eindeutig bestimmten Font. Um sich die Angabe der weiteren Parameter zu sparen, kann man, ähnlich wie in der Shell, das Zeichen "*" als Abkürzung für irgendwelche Teile des Namens verwenden. Obige Beispiele wären etwa durch folgende Angaben eindeutig festgelegt:

```
-*-helvetica-bold-r-*-12-*
*lucida-medium-i-*-20*
-sony-*-24-*-75-*-jisx0201.1976-*
```

Sollte ein solcher abgekürzter Name nicht eindeutig sein, wird automatisch der erste Font in der Liste aller Fonts ausgewählt, auf den das Textmuster paßt.

Eine zweite Vereinfachung zur Bezeichnung von Fonts bieten die Alias-Namen: Es ist möglich, einem Font einen oder mehrere weitere, beliebige Namen zu geben. Vordefinierte Aliasnamen sind z.B.

```
fixed    : -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
7x13     : -misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
```

Leider können solche Namen nur vom System-Verwalter festgelegt werdem, nicht von jeden Benutzer. Daher gibt es meistens nur eine Anzahl von Standard-Abkürzungen, die schon im X11R4 eingebaut sind.

Um einen Überblick über die vorhandenen Fonts zu bekommen, sind die folgenden drei Clients sehr nützlich:

xlsfonts rattert die gesamte Liste aller Namen - einschließlich alias-Namen - herunter, die man sich dann, etwa mit less, ansehen kann oder aus der man sich mit egrep die gesuchten Namen herauspickt.

Einen guten Überblick über das Aussehen der Fonts bietet xfontsel. Nach dem Start erscheint ein Fenster, in dem man zu jedem der 14 Parameter mit der linken Maustaste ein Menü aller Möglichkeiten herausklappen und Werte auswählen kann (s. Abb. 5). Groß-, Kleinbuchstaben und Ziffern dieses Fonts werden in einem Fenster dargestellt. Außerdem wird angezeigt, wieviele Fonts auf das gerade ausgewählte Muster (das evtl. einige "*" enthält) passen. Der Name des Fonts, der in einer eigenen Zeile steht, kann nicht wie üblich mit der linken Maustaste übernommen werden. Stattdessen gibt es einen Knopf "select" mit dieser Funktion: Wurde er gedrückt, kann der Name wie üblich mit der mittleren Maustaste an anderer Stelle eingefügt werden.

Möchte man sich schließlich einen bestimmten Font ganz genau ansehen, ruft man mit "xfd -fn fontname &" den "X-Font-Displayer" auf, der alle Zeichen aus einem Font in einem Fenster anzeigt (s. Abb. 5). Sind es zu viele (etwa beim chinesischen Font), kann man zwischen mehreren Seiten blättern. Außerdem erhält man durch Klicken auf einen Buchstaben Größenangaben über ihn.

Beim xterm kann man über das dritte Menü (<Control-rechte Maustaste>) zwischen mehreren Fonts wählen (s. Abb. 5). Die meisten Fonts lassen sich allerdings nur vor dem Start des xterm angeben (s.u.). Einen einzigen kann man jedoch auch noch zur Laufzeit ändern. Dazu markiert man zunächst irgendeinen Fontnamen (mit der linken Maustaste etwa aus dem Clipboard oder mit "select" im xfontsel) und wählt dann im Font-Menü den Punkt Selection aus. Der Font des xterm wird sofort durch den neuen ersetzt, wobei das xterm ggf. seine Größe ändert. Nun kann man zwischen den vorgegebenen Fonts oder diesem neuen durch Auswahl von "Default" bis "Large" oder "Escape Sequence" im Font-Menü hin- und herwechseln. Natürlich kann man jederzeit auf die beschriebene Weise mit dem Selection-Feld den neuen Font durch einen anderen ersetzen.

Abbildung 5:

4.2 Farben

Hat man einen Farbschirm zur Verfügung, kann man für fast jedes Fenster - incl. Rahmen, Cursor, Root - Vorder- und Hintergrundfarben festlegen. Farben werden in X11R4 im RGB-Modell bezeichnet, d.h. man wählt je einen Wert für den Rot-, Grün- und Blauanteil einer Farbe aus. Diese Werte müssen als Hexadezimalzahlen in der Form #RGB angegeben werden, wobei R, G und B jeweils für die gleiche Anzahl von Ziffern (0-9, a-f/A-F) stehen. Für fast alle gängigen Workstations ist die Hardware auf RGB-Werte jeweils im Bereich von 0 bis FF ausgelegt, mehr als zwei Ziffern machen also in der Regel keinen Sinn. Somit erhält man etwa folgende Farben:

#000000	schwarz
#808080	mittelgrau
#ffffff	weiß
#0000ff	blau
#ffff00	gelb
#d2691e	schokoladenbraun

Natürlich wäre es schön, Farben direkt über Namen ansprechen zu können. Auch dafür gibt es wieder ein Alias-File mit Zuordnungen von Namen und RGB-Werten, das aber auch nur vom System-Verwalter geändert werden kann. Neben den gängigen (englischen!) Farbbezeichnungen enthält das Standard-Alias-File so klangvolle Namen wie PeachPuff oder MidnightBlue, insgesamt 738 Einträge.

Leider gibt es im Standard-Umfang kein Programm, um sich einfach mal eben zu RGB-Werten die entsprechende Farbe anzusehen. Daher wurde am RZTU als kleine Übung im X-Programmieren ein solches Tool geschrieben. Es heißt xpalette und erlaubt, durch Mausklicks RGB-Werte einzustellen, die dann dezimal und hexadezimal zusammen mit einem Feld der entsprechenden Farbe angezeigt werden.

Wie viele Farben sich gleichzeitig auf einem Bildschirm unter X darstellen lassen, hängt natürlich in erster Linie von der Hardware ab. Da das X-Window-System möglichst hardwareunabhängig sein soll, werden verschiedene Methoden zur Farbdarstellung ermöglicht, die je nach Art des Bildschirms verwendet werden können. Für Farbschirme, die maximal 256 Farben auf einmal darstellen können - wie die meisten an der TU verwendeten -, verwaltet der Server Farbtabellen mit 256 Einträgen, in denen zu jeder Nummer 0 ... 255 ein RGB-Wert eingetragen ist. Es gibt eine Standard-Farbtabelle, die von solchen Clients benutzt wird, die nur wenige Farben benötigen. In dieser Tabelle reservieren sie sich eine kleine Anzahl von Plätzen für ihre eigenen Bedürfnisse, z.B. Vorder- und Hintergrundfarbe. Meistens ist für diese Zwecke ausreichend Platz vorhanden. Es gibt aber Programme, die eine große Zahl von Farben brauchen - etwa zur Darstellung von fließenden Farbverläufen. Solche Clients können sich eine eigene Farbtabelle im Server anlegen und mit den benötigten Farben besetzen. Das Umschalten zwischen den Farbtabellen geschieht dann automatisch beim Aktivieren der entsprechenden Fenster, etwa wenn man mit der Maus vom Root-Fenster (mit der Standard-Farbtabelle) in das Fenster einer solchen Anwendung geht. Die Farben in der aktuellen Farbtafel kann man sich ansehen mit xshowcmap.

4.3 Kommandozeilen-Optionen

X-Clients erlauben in der Regel eine Vielzahl von Kommandozeilen-Optionen, wobei die meisten - genauer gesagt: zumindest alle mit dem X-Toolkit geschriebenen - eine Menge von Standard-Optionen kennen. Die wichtigsten dieser Optionen sollen i.f. kurz vorgestellt werden.

-display : Falls man keine DISPLAY-Umgebungsvariable gesetzt hat (s.o.) oder einen anderen Server (Bildschirm) ansprechen will, kann man mit dieser Option den X-Server für den Client angeben. Bsp.: -display ap03:0

-geometry : Mit dieser Option kann man Größe und Lage des Hauptfensters eines Clients in der Form "BreitexHöhe+Xoff+Yoff" festlegen. Dabei bezeichnen Breite und Höhe die Größe, Xoff und Yoff den Abstand einer Ecke vom Rand, normalerweise in Pixeln. Das (obligatorische) Vorzeichen von Xoff (bzw. Yoff) gibt an, ob von oben oder unten (bzw. links oder rechts) aus gemessen wird; der Abstand bezieht sich dann auf die nächstgelegene Ecke des Fensters.
Bsp.: -geometry 120×210+1-1 Das Fenster hat die Größe 120x210, es befindet sich in der linken unteren Ecke.
Einige Clients (z.B. das xterm) beziehen die Größenangaben auf Zeichen, nicht auf Pixel.

-foreground (-fg),**-background** (-bg): Diese Optionen legen die Vorder- und Hintergrundfarbe fest.
Bsp.: -fg red -bg #ffff80 rote Schrift auf hellgelbem Untergrund.

-font (-fn): Festlegung des (oder des Standard-)Fonts.
Bsp.: -fn 6x13

-title : Diese Option erlaubt, einen Namen für die Titelzeile anzugeben. Das ist insbesondere bei mehreren xterms auf einmal sehr nützlich. Bsp.: -title "telnet auf der AP02"

-name : Ändert man nicht den Titel, sondern den Namen eines Clients, bekommt auch das Icon diesen Namen. Allerdings hat ein neuer Name auch Auswirkungen auf andere Voreinstellungen (s.u.).

4.4 Die Resourcen-Datenbank

X-Clients sind in der Regel auf vielfältige Weise konfigurierbar: Außer den in 4.3 erwähnten Größen kann man z.B. festlegen, ob ein Scrollbalken angezeigt wird, welche Fonts an welchen Stellen benutzt werden, manchmal sogar, welche Texte in den Knöpfen ("Buttons") stehen

und auf welche Tasten- oder Mausdrücke sie wie reagieren. Um alle diese Daten ("Resourcen" genannt) zu verwalten, wird vom X-Server eine Datenbank angelegt, auf die man mit dem Programm xrdb ("X Resource Data Base") zugreifen kann. Die gewünschten Resource-Werte schreibt man in ein File (meistens ~/.Xdefaults genannt) und lädt sie mit "xrdb -load filename". Weitere Werte aus einem anderen File kann man dazuladen mit "xrdb -merge filename", die aktuellen Werte werden mit "xrdb -query" angezeigt.

Ein Resource-Eintrag im .Xdefaults-File steht in einer Zeile und hat die Form

Resource-Name: Wert

wobei nach dem ":" beliebig viele Blanks und Tabs folgen können. Der Wert einer Resource ist ein Text-String, der allerdings auch als numerischer oder boolescher Wert interpretiert werden kann. Beispiele:

```
XTerm*font:                 10x20
xedit*geometry:             550x700-285+1
xclock.clock.foreground:    Blue
```

Reicht eine Zeile nicht aus, kann man mit "\" als letztem Zeichen die Zeile verlängern. Kommentarzeilen beginnen mit "!". Ist in einer Zeile ein Fehler (Resource-Name falsch, Doppelpunkt fehlt ...), wird sie einfach übergangen.

Um das Format des Resource-Namens verstehen zu können, muß man etwas über den internen Aufbau von X-Clients wissen: Die meisten Clients sind aus vorgefertigten "Objekten" (in X "Widgets" genannt) wie Dialogboxen oder Knöpfen zusammengesetzt, die hierarchisch aufgebaut sind (z.B. ein Knopf in einem Textfeld in einem Unterfenster ...). Außerdem sind mehrere Objekte (manchmal auch nur eins) zu einer Klasse zusammengefaßt, über die sie alle gleichzeitig angesprochen werden können. Die Klassennamen beginnen zur Unterscheidung mit einem oder zwei Großbuchstaben, während die einzelnen Objekte ("Instanzen" der Klasse genannt) mit Kleinbuchstaben beginnen. Als Beispiel betrachten wir die Widget-Hierarchie des xclipboard. In jeder Zeile steht ein Widget mit vorangestelltem Klassennamen, die Einrückungen sollen die Hierarchiestufen verdeutlichen:

XClipboard xclipboard

Form form

Command quit
Command delete
Command new
Command next
Command prev
Text text

Ein Resource-Name hat nun die Form

objekt[.subobjekt...].attribut

wobei statt der (Sub-)Objekt- und Attributnamen auch die Klassennamen stehen können, wenn man alle Objekte dieser Klasse meint. Dabei haben speziellere Angaben Vorrang vor allgemeinen. Sind etwa folgende Resourcen für das xclipboard gesetzt:

```
XClipboard.Form.Command.background:   LightSeaGreen
XClipboard.Form.quit.background:      red
```

dann haben alle Command-Knöpfe außer dem roten "Quit" -Button einen grünen Hintergrund.

Ein Beispiel für den Einsatz eines Klassennamens für ein Attribut zeigt

xterm.vt100.Foreground: blue

Durch Angabe der Klasse Foreground werden die Werte für alle Instanzen dieser Klasse gesetzt, in diesem Fall für foreground (Textfarbe), cursorColor (Farbe des Textcursor) und pointerColor (Farbe des Mauszeigers).

Statt des "." als Trennzeichen ist auch das Zeichen "∗" möglich. Es steht für keine oder mehrere beliebige Zwischenstufen in der Hierarchie. So steht "xterm∗Foreground" u.a. für "xterm.vt100.Foreground" und für "xterm.tek4014.Foreground", aber auch die Farbe des Scrollbalkens und der Menüs wird davon gesetzt. Die Benutzung des "∗" ist nützlich, wenn man Resourcen für ganze Teilbäume der Widget-Hierarchie setzen möchte oder einfach nicht weiß, wie der ganze Baum aussieht. Wieder gilt: Spezielle Angaben haben Vorrang vor allgemeinen. Z.B. könnte man mit

∗Background: #cdfff0

den Hintergrund aller Clients in beruhigendes Grün tauchen und sich für einige ausgewählte Anwendungen eine andere Farbe wählen.

Die Möglichkeit, auch für den Clientnamen eine Klassenbezeichnung angeben zu können, ist besonders in Zusammenhang mit der Option "-name" interessant: Sie bewirkt, daß der Instanzname sich ändert, während der Klassenname gleich bleibt. Man kann sich z.B. spezielle Resourcen setzen wie etwa "meinXterm∗attribut: 42", die für ein "normales" xterm nicht gelten. Ruft man es aber mit der Option "-name meinXterm" auf, so werden sie benutzt, während die mit "xterm∗..." gesetzten Werte unbeachtet bleiben. Alle Resourcen, die mit XTerm∗..., also dem Klassennamen, gesetzt wurden, gelten für beide xterms.

Bei den meisten Clients ist es möglich, alle über Kommandozeilen-Parameter änderbaren Größen auch über Resourcen zu setzen. Auf diese Weise spart man sich endlos lange Aufrufe. Möchte man im Einzelfall doch einen anderen Wert, hat die Option Vorrang vor dem Resourcewert.

Die Konfigurations-Möglichkeiten mancher Clients gehen weit über die hier gezeigten Beispiele hinaus; sie reichen bis zur völligen Umgestaltung des Aussehens und der Tasten-Kommandos. Die Manualseiten geben jeweils erschöpfend Auskunft.

4.5 Konfiguration des Window-Managers twm

Da der twm eigentlich auch ein normaler X-Client ist, könnte man annehmen, daß er ebenfalls über Resources konfiguriert wird. Die Entwickler des twm entschieden sich aber - wohl wegen der großen Anzahl der Parameter -, daß der twm beim Start sein eigenes File liest, das normalerweise ~/.twmrc heißt. In diesem File werden Variablenwerte gesetzt, Zuordnungen von Maus- und Tastendrücken zu Aktionen definiert und die Menüs beschrieben. Außerdem können Kommentarzeilen eingefügt werden, sie beginnen mit "#". Statt systematisch alle Möglichkeiten zu erklären, wird i.f. als Beispiel das .twmrc-File beschrieben, das neue Benutzer auf den Apollos mitbekommen. Es ist im Anhang A komplett abgedruckt. Weitere Einzelheiten kann man dem Manual zum twm entnehmen.

Die Variable "Color" legt in einer Liste Farben für verschiedenen Zwecke fest, im Beispiel für den Titelbalken, die Schrift im Titel, die Menüs und den Icon-Manager. Diese Werte werden bei Schwarz-Weiß-Monitoren ignoriert. Dann wird die Randbreite ("BorderWidth") auf 3 gesetzt und einige Fonts ausgewählt. Die Angabe von "RandomPlacement" bewirkt, daß neue Fenster, die keine Geometrie-Angaben haben (über Resourcen oder Kommandozeilen-Optionen), vom twm selbst "irgendwohin" gesetzt werden. Defaultmäßig müssen solche Fenster beim Erscheinen vom Benutzer explizit positioniert werden.

Die nächsten Einträge haben mit der Icon-Verwaltung zu tun. Beim twm gibt es zwei verschiedene Formen der Iconisierung: Ein Fenster verwandelt sich in ein kleineres (eben das Icon), das irgendwo auf dem Bildschirm erscheint, oder man benutzt den Icon-Manager, ein eigenes Fenster, in dem Clients eingetragen sind und beim Iconisieren durch ein "X" markiert werden. Die Zeile "ShowIconManager" sorgt dafür, daß der Icon-Manager angezeigt wird; wegen der Angabe "SortIconManager" erscheinen die Clients in alphabetischer Reihenfolge. Die nächsten beiden Angaben legen Lage und Größe sowie den Font des Icon-Managers fest.

Einige Standard-Clients (wie die Uhr und der Briefkasten) bleiben normalerweise unverändert am Bildschirm stehen, sie sollen nicht iconisiert werden. Daher ist für sie auch kein Eintrag im Icon-Manager nötig. Er wird unterdrückt durch Eintrag in der Liste zur Variablen "IconManagerDontShow". Außerdem wollen wir für die normalen Clients keine Extra-Icons, da ja der Icon-Manager verwendet wird. Dazu dient die Zeile "IconifyByUnmapping". Damit haben wir jedoch ein Problem erzeugt: Sollte etwa eine oclock doch iconisiert werden (z.B. über ein Menü), wird für sie kein Icon erzeugt, sie ist aber auch im Icon-Manager nicht vorhanden, mithin gar nicht mehr. Obwohl das Programm also noch läuft, wie man mit ps feststellen kann, ist es nicht mehr auf den Schirm zu bekommen. Man kann es nur noch mit kill abbrechen. Um ein solches Verhalten bei versehentlichem Iconisieren zu verhindern, wird für die Variable "DontIconifyByUnmapping" eine Liste von Clients angegeben, für die - entgegen der vorigen Einstellung - eben doch Icons angezeigt werden sollen. Dies sind natürlich gerade die Clients, die nicht im Icon-Manager angezeigt werden sollen. Außerdem

ist für sie eine Titelzeile überflüssig, ebenso wie für das Fenster des Icon-Managers, den wir ja schon explizit plaziert haben, daher tragen sind sie in der Liste der Variable "NoTitle" vermerkt.

Schließlich sollen die Icons (wenn schon mal welche auftauchen) nicht beliebig über den Schirm verstreut werden, sondern sich unter dem Icon-Manager sammeln. Dafür sorgt der Wert der Variablen "IconRegion" (Details: s. man).

Die nächsten Zeilen legen fest, welche Aktionen bei welchen Tastendrücken erfolgen sollen. Dazu gibt es eine Reihe vordefinierter Aktionen des twm, z.B.:

f.menu	zeigt ein (weiter unten im .twmrc-File definiertes) Menü an
f.move	bewegt ein Fenster
f.fullzoom	vergrößert ein Fenster auf Bildschirmformat bzw. bringt es auf Originalgröße zurück
f.function	ruft eine selbst definierte Funktion auf
f.title	zeigt eine Titelzeile in einem Menü an
f.exec	ruft ein UNIX-Kommando auf
!	Abkürzung für f.exec

Zunächst wird mit der Variablen "DefaultFunction" festgelegt, welche Funktion ausgeführt wird bei undefinierten Tastendrücken. Im Beispiel-File wird dann ein Menü namens "TWM" angezeigt.

Danach folgt eine Liste von expliziten Tastendefinitionen der Syntax

BUTTON = [KEY] : CONTEXT : FUNCTION

Button ist die zu definierende Taste, dabei bezeichnet Button1 (B.2, B.3) die linke (mittlere, rechte) Maustaste (falls keine Tasten umdefiniert wurden, s. Kap. 4.7). Dann folgt evtl. eine weitere "modifizierende" Taste, z.B. "Shift" oder "Control", die "gleichzeitig", d.h. am besten vorher, gedrückt werden muß. Das CONTEXT-Feld gibt an, wo sich der Mauszeiger während des Tastendrucks befinden soll, etwa am Rand eines Fensters ("frame"), mitten in einem Fenster ("window") oder im Root-Fenster ("root"). Schließlich folgt der Name der Funktion, die ausgeführt werden soll. Beispielsweise bewirkt

Button2 = : frame : f.move ,

daß man ein Fenster verschieben kann, wenn man mit der mittleren Maustaste auf seinen Rand drückt, wegen

Button3 = : root : f.menu "Logins"

erscheint bei Drücken der rechten Maustaste im Root-Fenster das "Logins"-Menü, und aufgrund der Zeile

Button1 = s : window : f.fullzoom

läßt sich ein Fenster durch <Shift><linke Maustaste> auf volle Größe bringen (und zurück).

Im betrachteten .twmrc-File folgt dann ein (triviales) Beispiel für eine selbst-definierte Funktion namens "**beep-beep**".

Schließlich werden mit

menü "name"
{
Menü-Eintrag Aktion
... ...
}

die drei Menüs "TWM", "Utilities" und "Logins" definiert.

4.6 Das X-Startup-File

Beim Beginn einer X-Sitzung wird i.a. ein Benutzer-File ausgeführt, das einige Voreinstellungen setzt und die ersten Clients startet. Dieses File heißt meistens .xinitrc oder .xsession (beim XDM). I.f. soll ein Beispiel-Startup-File besprochen werden; es ist in Anhang B abgedruckt.

Zunächst wird mit xrdb die X-Resource-Datenbank geladen. Danach wird das Root-Fenster mit einer eigenen Bitmap "tapeziert". Mit dem Programm xsetroot kann man Farbe und Muster des Hintergrundfensters festlegen und bestimmen, welche Form der Cursor dort haben soll.

Weitere allgemeine Parameter können mit dem Programm xset gesetzt werden, z.B.:

Lautstärke des Warntons in %, Tonhöhe in Hz, Dauer in ms :
xset b 50 700 100
Tastatur-Klick: Lautstärke in % oder "off":
xset c off
Mausbeschleunigungsfaktor und -Empfindlichkeit:
xset m 10 2
Tasten-Wiederholung on/off:
xset r on
Anzeige aller aktuellen xset-Parameter:
xset q

Danach wird mit xhost Clients von der SG03 erlaubt, den eigenen X-Server anzusprechen. Wie oben schon erwähnt, ist es nun aber jedem Benutzer dieser Maschinen erlaubt, Clients auf diesen Bildschirm zu bringen, was natürlich ein Sicherheitsrisiko darstellt. Seit X11R4 gibt es daher ein Autorisierungsschema, das den XDM (X Display Manager) benutzt und userbezogene Autorisierung gestattet:

Beim Einloggen legt der xdm im Homedirectory ein File .Xauthority an, das einen Schlüssel enthält. Von nun an können Maschinen, die nicht mit xhost eingetragen wurden, dennoch den X-Server ansprechen, falls sie Zugriff auf diesen Schlüssel (d.h. auf das File .Xauthority) haben. Dies ist automatisch gewährleistet, falls man auf einem anderen Rechner über NFS denselben Homebereich benutzt. Ansonsten muß man den Schlüssel in das eigene Homeverzeichnis auf der anderen Maschine übertragen (mit ftp, rsh, ...) und kann dann von dort Kontakt mit dem X-Server aufnehmen. Um sich das Übertragen zu vereinfachen, kann man das xauth-Programm benutzen und etwa folgende Zeile statt der xhost-Autorisierung ins .xsession einbauen:

xauth extract - $DISPLAY | rsh sg03.cip3s xauth merge -

(s. man xauth). Um mit Rechnern zu arbeiten, auf denen diese Autorisierungsfunktionen noch nicht installiert sind (bei uns z.B. auf den Apollos mit X11R3), muß man diese allerdings nach wie vor mit xhost zulassen.

Schließlich werden im Startup-File die Lieblings-Clients und der Window-Manager gestartet. Loggt man sich über den xdm ein, wird automatisch noch ein spezielles xterm gestartet. Loggt man sich an diesem xterm aus, werden alle anderen Clients abgebrochen, man muß sie also nicht einzeln beenden.

4.7 Ändern der Tastenbelegung

Natürlich ist es in X auch möglich, die Tastaturbelegung zu ändern. Ein Problem ist allerdings, daß verschiedene Tastaturen unterschiedliche Codes erzeugen. Um diese Hardwareabhängigkeit so weit wie möglich zu umgehen, wird eine Taste unter X auf zwei verschiedene Weisen benannt: Zunächst hat jede Taste einen Keycode. Dies ist einfach eine Zahl, die die Hardware beim Druck der Taste sendet. Außerdem wird sie noch durch einen Symbolwert ("keysym") gekennzeichnet. Dieser Wert ist eine hardware-unabhängige "Funktionsbeschreibung" der Taste. Z.B. hat bei meinem X-Terminal die Taste 'a' den Keycode 28 und den Symbolwert 'a', die Taste "Shift" den Keycode 18 und den Symbolwert 'Shift_L'. In der Regel macht der Name eines Symbolwerts deutlich, welche Taste damit gemeint ist; nur dieser Wert spielt in Anwendungen eine Rolle. Allerdings ist es möglich, die Zuordnung zwischen Keycode und Symbolwert beliebig zu ändern; das bedeutet ja gerade eine Tasten-Umdefinition.

Neben normalen Tasten gibt es noch die "Modifier"-Tasten. Diese ändern, zusammen mit anderen gedrückt, deren Bedeutung. Solche Modifier werden etwa mit "shift", "lock" oder "control" bezeichnet. Wiederum sind es aber nicht notwendigerweise (wenn auch im Re-

gelfall) die so bezeichneten Tasten auf der Tastatur, sondern jede beliebige Taste kann die Funktion "shift" übernehmen.

Eine dritte Kategorie sind die Maustasten. Sie werden einfach von links nach rechts mit 1, 2, 3 durchnumeriert.

Leider gibt es keinen Standard-Client, mit dem man sich Keycode und Symbolwert einer Taste direkt ansehen kann. Auf den Apollos befindet sich allerdings das Programm "xev", das alle X-Aktionen mitprotokolliert, darunter auch Tastendrücke. Nach "xev" erscheint ein Fenster mit einem kleinen Quadrat. Bewegt man die Maus dort hinein und drückt z.B. die Taste 'a', so erscheinen (unter anderem) folgende Zeilen:

```
KeyPress event, serial 16, synthetic NO, window 0x4800001,
root 0x28, subw 0x4800002, time 1481084, (35,43), root:(197,205),
state 0x0, keycode 28 (keysym 0x61, a), same_screen YES,
XLookupString gives 1 characters: "a"
```

Hieraus kann man (nach etwas Suchen) die oben angegeben Werte für 'a' ablesen.

Alle Aktionen, die mit der Tastenbelegung zu tun haben, werden mit dem Programm "xmodmap" durchgeführt. Bevor man sich aber ans Ändern macht, sollte man die ursprüngliche Tastentabelle, also die Zuordnung von Keycodes zu Symbolwerten, abspeichern. Das geschieht mit

xmodmap -pk > keytable

Ein vollständiges Beispiel einer solche Tabelle ist in Anhang C abgedruckt.

Nun wollen wir uns drei Beispiele für Tasten-Umdefinitionen ansehen. Als erstes sollen die 1. und 3. Maustaste vertauscht werden ("Linkshänder-Maus"). Mit "xmodmap -pp" kann man sich die Belegung der Maustasten anzeigen lassen. Die Umordnung erfolgt durch

xmodmap -e 'pointer = 3 2 1'

Als nächstes soll die Caps-Lock-Taste, die ich immer versehentlich drücke, ohne sie je zu brauchen, funktionslos werden. Mit "xmodmap" sehen wir uns die Zuordnung von Symbolwerten und Modifiern an. Tatsächlich hat die "CapsLock"-Taste die "lock"-Funktion. Sie kann davon entbunden werden mit:

xmodmap -e 'remove lock = Caps_Lock'

Das letzte Beispiel stammt aus der Arbeit mit dem Emacs: Mir fiel auf, daß ich bei F1 und Shift-F1 immer die gleiche Funktion bekomme. Das liegt daran, daß zum Keycode für den Keysym "F1" nur ein Eintrag steht. Ein zweiter Eintrag wird - falls vorhanden - für die "geshiftete" Funktion verwendet. Da der keysym "F17" bei mir nicht verwendet wird (ich habe nur 12 Funktionstasten an meinem Terminal), lege ich ihn auf <Shift-F1>:

xmodmap -e 'keysym F1 = F1 F17'

Nun erzeugen F1 und <Shift-F1> unterschiedliche Escape-Sequenzen, die man im Emacs verschiedenen Funktionen zuordnen kann.

Sollen die mühsam gebastelten Definitionen auch nach dem nächsten Einloggen gelten, muß man die xmodmap-Befehle ins .xsession-File (oder ein analoges Startup-File) schreiben. Zur Vereinfachung kann man die eigentlichen Kommandos für xmodmap in ein File (z.B. namens .xmodmaprc) schreiben, das in unserem Beispiel folgenden Inhalt hätte:

pointer = 3 2 1
remove lock = Caps_Lock
keysym F1 = F1 F17

und es mit "xmodmap .xmodmaprc" laden.

5 Spezielle X-Clients

5.1 Informationen über Clients und Server

In diesem Abschnitt werden Clients aus dem Standard-Umfang von X11R4 vorgestellt, die verschiedene Informationen über den Server und laufende Clients liefern.

appres: Um festzustellen, welche Resource-Werte gerade für eine Applikation gültig sind, wobei auch evtl. vorhandene systemweite Defaults mit angezeigt werden, dient das Programm appres. Mit "appres XTerm" sieht man so nicht nur, welche Werte man - etwa mit xrdb - selbst geladen hat, sondern auch eine große Zahl weiterer Werte, die aus dem File /usr/lib/X11/app-defaults/XTerm stammen.

xwininfo: Nach "xwininfo" verwandelt sich der Cursor in ein Fadenkreuz, mit dem man wie üblich mit Mausklick links ein Fenster auswählen kann. Über dieses Fenster werden dann u.a. Geometrie-Daten wie Größe des Fensters und Lage auf dem Bildschirm ausgegeben. Außerdem erhält man so die Window-ID, eine eindeutige Kennzeichnung für jedes dargestellte Fenster, die für einige andere Kommandos benötigt wird.

xlswins: Die Fenster auf dem Schirm und auch die Unterfenster einer Anwendung sind hierarchisch geordnet. Diese Hierarchie kann man sich für ein bestimmtes Fenster (und seine Unterfenster) ansehen mit "xlswins Window-ID". Mit der Option -l erhält man auch Geometrie-Angaben der Unterfenster. Gibt man keine Window-ID an, wird das Root-Fenster als Startpunkt gewählt, d.h. man bekommt eine Liste aller Fenster.

xlsclients: Mit xlsclients bekommt man eine Liste aller Clients auf dem Schirm incl. ihrer Optionen und der Herkunftsmaschine. Mit der Option -l werden Zusatz-Informationen, z.B.

Klasse und Instanz eines Clients, angezeigt.

xdpyinfo: Informationen über den eigenen X-Server und das Display erhält man mit xdpyinfo. Unter der Vielzahl der angezeigten Eigenschaften findet man u.a. den Namen des Displays, die Größe in Pixeln und Millimetern sowie die Auflösing in "dots/inch" und die Zahl der Farben.

5.2 Clients am RZTU

Eine große Anzahl von X-Programmen für jede sinnvolle oder unsinnige Anwendung ist frei verfügbar und kann von FTP-Servern geladen werden. Eine weitere Quelle sind die News[1], z.B. die Gruppe comp.x.sources. I.f. sollen einige Beispiele, die am RZTU vorhanden sind, vorgestellt werden.

xgrab (C1): ein Programm zum Erzeugen von Bildschirm-Dumps. Man kann wahlweise ein Fenster oder einen beliebigen rechteckigen Bildschirm-Ausschnitt in verschiedenen Formaten (darunter das X-Window-Dump-Format und Farb-Postscript) in ein File schreiben oder gleich zu einem Drucker schicken. Die Bedienung erfolgt über ein eigenes Fenster mit vielen Knöpfen (s. Abb. 6).

xcalendar (C1): ein komfortabler Terminkalendar. Man erhält einen Kalendar des aktuellen Monats (kann aber auch mit Pfeiltasten frühere oder spätere Monate anzeigen lassen). Klickt man auf einen Tag, erscheint ein Fenster, in dem man seine Termine eintragen und speichern kann. Tage mit Terminen werden im Kalendar-Blatt gekennzeichnet (s. Abb. 7).

xrn: komfortabler News-Leser. Für alle, die News lesen, ist der xrn ein wichtiges Werkzeug. So ziemlich alles, was man mit News machen kann, ist hier möglich. Da die "Tasten-Bezeichnungen" selbst-erklärend sind, erübrigt sich das Lernen von Abkürzungen wie etwa beim rn oder nn (s. Abb. 7).

xarchie: Archiv-Server-Datenbasis-Befrager. Die Zahl der Programme, die von irgendwelchen Fileservern im Internet abgerufen werden können, ist kaum überschaubar. Daher hat man spezielle Server eingerichtet - sogenannte "Archie"-Server -, die Informationen über den Inhalt vieler FTP-Server speichern und die etwa nach Suchbegriffen gefragt werden können. Eine bequeme Möglichkeit, diese Server zu befragen, bietet xarchie. Nach dem Aufruf erscheint ein Fenster mit vielen Feldern (s. Abb. 6). Um zu sehen, wo man die Sourcen für den xrn herbekommt, trägt man "xrn" im Feld "Search Term" ein und fragt den Archie-Server dann mit "Query" (obere Knopfleiste). Nach einiger Zeit erscheint im linken Feld eine Reihe von Rechnern, auf denen etwas gefunden wurde. Durch Anklicken kann man sich nähere Informationen von einem dieser Rechner holen, am besten einem nahegelegenen. Möchte man die zugehörigen Files haben, braucht man nur noch "Ftp" anzuklicken, dann werden sie übertragen. Man sollte allerdings die Netzbelastung bedenken und am besten spät abends Daten übertragen. Außerdem kann man mit "Nice Level" seine Priorität herabsetzen, wenn man es nicht ganz so eilig hat.

[1] Vergleiche hierzu die Informationsschrift des RZTU: NET 6 – Das News-System

Abbildung 6:

Abbildung 7:

Natürlich gibt es auch viele Spiele unter X, z.B. Mahjongg (xmj) oder Tetris (xtetris) (s. Abb. 8).

5.3 X-Schnittstellen einiger Programm-Pakete

Viele Programme mit Graphik-Ausgabe, die ursprünglich für spezielle Hardware (Tektronix-, Apollo-, Sun-Bildschirme usw.) geschrieben wurden, verwenden inzwischen X-Routinen zur Graphik-Darstellung und sind damit unabhängig von spezifischer Hardware. Einige der wichtigsten Anwendungs-Pakete auf der C2 mit X-Schnittstellen sollen i.f. kurz vorgestellt werden. Dabei geht es nicht um eine Einführung in das Arbeiten mit diesen Programmen, sondern es soll nur kurz an einfachen Beispielen gezeigt werden, wie solche X-Umgebungen aussehen. Auch wenn zum genauen Verständnis meistens ein Studium der Handbücher erforderlich ist, kann man doch durch Experimentieren einen ersten Eindruck bekommen.

Jeder der folgenden Abschnitte beginnt mit einer kurzen Vorstellung des Programms. Danach werden Beispielfiles angegeben, die ins aktuelle Verzeichnis kopiert werden sollten. Anschließend werden die Kommandos bzw. die Menü-Eintrage (in Großbuchstaben) aufgelistet, die man eingeben muß, um die Beispiele zu reproduzieren. Schließlich werden die zum Verlassen des Programms nötigen Aktionen beschrieben.

Unigraph: Programm zur interaktiven graphischen Darstellung von wissenschaftlichen Daten etwa als Linien-, Torten- oder Balkengraphik, in zwei- oder dreidimensionaler Darstellung usw.

Files in /usr/docall/uniras/unigraph:

engine.com, engine.dat, enginereg.dat

Beispiel-Sequenz:

source /usr/local/bin/uni.login
unigraph -d lx11 -m -exit &
OPTIONS
EXECUTE
engine

verlassen mit

EXIT
YES

Uniedit: direktes Erzeugen von Graphiken, Nachbearbeiten von Bildern aus Unigraph

Abbildung 8:

Files in /usr/docall/uniras/uniedit:

bicycle.cell

Beispiel-Sequenz:

source /usr/local/bin/uni.login
uniedit -d lx11 -m -f -exit &
EDIT
bicycle

verlassen mit

TOP
EXIT
YES

Mentat: Pre- und Post-Prozessing für das Finite-Elemente-Programm Marc

Files in /usr/docall/marc/mentat:

ses8a_proc
f08a01_data

Beispiel-Sequenz:

mentat
device scope xwindows return
<RET>
PROCEDURE
OPEN
INPUT
ses8a_proc

verlssen mit

STOP
YES

Phoenics: Berechnung von numerischen Problemen aus der Strömungsmechanik sowie Vor- und Nachbearbeitung der Daten

Files in /usr/tuhh/info/tu_beispiele/phoenics

∗ (alle, zusammen etwa 170 k)

Beispiel-Sequenz:

make_link

runpho

5

phi

<RET>

<RET>

<RET>

con h1 z 1 fill

1

clear

vec z 1 sh

verlassen mit

quit

Bemerkung: Dies ist ein gutes Beispiel für eine schlechte X-Anbindung (allerdings immer noch besser als gar keine): Wird das Ausgabe-Fenster überdeckt, in ein Icon verwandelt oder irgendwie manipuliert, wird es nicht neu aufgebaut, sondern bleibt (ganz oder teilweise) schwarz.

Maple5: algebraische Formel-Manipulationen, Graphik-Teil zur Darstellung von Funktionen (s. Abb. 8).

Beispiel-Sequenz:

maple5

```
interface(plotdevice=x11);
plot3d(cos(sqrt(x*x+y*y))*exp(-0.2*(x*x+y*y)), x=-3..3, y=-3..3);
```

verlassen mit

Close

quit;

6 Der Motif-Window-Manager (mwm)

6.1 Benutzung des mwm

Neben dem Standard-Window-Manager von X11R4, dem twm, hat inzwischen auch der Window-Manager von Motif, der mwm, weite Verbreitung gefunden. Er hat im wesentlichen die gleichen Funktionen wie der twm, aber eine andere Optik (s. Abb. 8): Das Aussehen ähnelt dem von MS Windows mit 3D-Effekten wie Schatten um die Rahmen und Knöpfen, die sich "hineindrücken" lassen. Die Standard-Einstellungen unterscheiden sich bei verschiedenen Maschinen - ich werde i.f. die HP-UX-Konfiguration beschreiben -, wie beim twm kann man aber alles selbst konfigurieren.

Die Fenster sind wieder von Rahmen umgeben, die eine von zwei verschiedenen Farben haben (oft olivgrün und weiß), wobei die auffälligere Farbe das aktive Fenster markiert, also das, das Tastatur-Eingaben entgegennimmt. Normalerweise bleibt ein Fenster aktiv, auch wenn der Cursor sich woanders befindet; erst durch Klicken in einem anderen Fenster wird dieses aktiv ("erhält den Fokus"). Allerdings läßt sich wie beim twm einstellen, ob der Fokus dem Cursor folgen soll.

Die "Dekoration" eines Fensters fällt beim mwm etwas üppiger aus als beim twm: Jedes Fenster hat eine Titelzeile, in der der Name des Clients steht (genauer: sein Titel, s.o.). Die Titelzeile dient wieder als "Griff" zum Verschieben des Fensters, er wird wieder mit der linken Maustaste bedient. Beim Drücken der Maustaste werden kleine dunkle und helle Ränder vertauscht, die die Titelzeile umgeben. Dadurch entsteht der Eindruck, die Zeile wurde in den Bildschirm "hineingedrückt". Rechts neben dem Titel befindet sich ein kleiner, mit einem Punkt markierter Knopf. Er dient zum Iconisieren des Fensters. Das zugehörige Icon taucht am unteren Rand des Bildschirms auf; anders als beim twm sind alle Icons gleich groß. Durch zweimaliges schnelles Anklicken verwandelt man das Icon wieder in seine ursprüngliche Gestalt. Ganz rechts befindet sich ein mit einem Quadrat gekennzeichneter Knopf, mit dem man das Fenster auf volle (Bildschirm-)Größe bringen kann - und wieder zurück. Das ganze Fenster wird von einem relativ breiten Rahmen mit abgeteilten Ecken umgeben. Bewegt man den Cursor auf eine Ecke oder ein Seitenteil, verwandelt er sich in einen Winkel oder einen Balken. Man kann nun die Größe des Fensters in der entsprechenden Richtung verändern.

Schließlich befindet sich in der linken oberen Ecke ein Knopf mit einem "Schlitz". Hier kann man durch Anklicken ein Menü, das Window-Menü, aufklappen, mit dem man alle eben beschriebenen Funktionen explizit aufrufen kann. Das Window-Menü erscheint auch, wenn man ein Icon einmal anklickt. Die Standard-Eintrage sind:

"Restore"	auf "Normalgröße" bringen
"Move"	Bewegen
"Size"	Verändern der Größe
"Minimize"	Iconisieren
"Maximize"	auf volle Größe bringen
"Lower"	Fenster ganz nach hinten setzen
"Close"	Client beenden

In jeder Menüzeile ist ein Buchstabe unterstrichen. Ist das Menü sichtbar, kann man mit diesem Buchstaben die entsprechende Menü-Funktion von der Tastatur aus auswählen. Zusätzlich sind Abkürzungen angegeben, z.B. "Alt + F7" für "Move", mit denen die Funktion jederzeit (auch ohne daß das Menü sichtbar ist) aufgerufen werden kann. "Alt" meint dabei die Modifier-Funktion "Meta", die in der Regel auf der "Alt"-Taste liegt. Der mwm kann Menüs verwalten, mit denen man ähnliche Funktionen wie beim twm ausführen kann. Standardmäßig ist ein Root-Menü vorhanden, das man mit der linken Maustaste aus dem Root-Fenster hervorzaubert. Es enthält meistens folgende Einträge:

"New Window"	neues Fenster (meist ein xterm)
"Shuffle Up"	Überlappungsreihenfolge der Fenster zyklisch
"Shuffle Down"	verändern
"Refresh"	Bildschirm neu aufbauen
"Restart"	Mwm neu starten (neue Konfiguration einlesen)
"Exit"	Mwm beenden

Im Gegensatz zum twm, der sich mit seinem letzten Client verabschiedet, muß der mwm explizit beendet werden. Tut man das nicht, sondern schaltet z.B. einfach sein X-Terminal aus, bleiben im System "Leichen" übrig. Man sollte also hin und wieder mit ps alte Prozesse aufspüren und abbrechen.

6.2 Konfiguration des mwm

Der mwm wird sowohl durch normale Resourcen gesteuert als auch durch ein eigenes Files namens .mwmrc, das die Menü-Beschreibungen und Tasten-Zuordnungen enthält.

6.2.1 Das .mwmrc-File

Die Definition von Menüs und die Zuordnung von Tasten- und Mausdrücken zu mwm-Aktionen geschieht im .mwmrc-File in sehr ähnlicher Weise wie im entsprechenden Abschnitt des .twmrc-Files. Als Beispiel ist das Standard-File, das normalerweise /usr/lib/X11/system.mwmrc heißt, im Anhang D abgedruckt.

Zunächst wird das Root-Menü definiert, dann das Window-Menü, in dem neben dem Menü-Eintrag und der Funktion auch die Abkürzungen angegeben werden, die durch Unterstreichung im Menü bzw. explizit angezeigt werden.

Danach werden Tasten-Zuordnungen beschrieben. Dies geschieht fast ganz genauso wie beim twm, nur die Tastenbezeichnungen lauten hier anders; ihre Syntax ist:

MODIFIER<Key>TASTE

z.B. "Shift<Key>Escape" für gleichzeitiges Drücken von Shift- und Escape-Taste. Mehrere Tastendefinitionen werden zu einer Tabelle namens "DefaultKeyBindings" zusammengefaßt.

Ganz analog werden schließlich Maustasten belegt; die Syntax wird aus den Beispielen im Anhang ersichtlich. Allerdings werden hier mehrere Tabellen definiert: "DefaultButtonBindings", "ExplicitButtonBindings" und "PointerButtonBindings". Normalerweise gelten die "DefaultButtonBindings", die anderen kann man über Resourcen benutzen (s.u.).

6.2.2 Resourcen für den mwm

Aus der großen Vielzahl der Resourcen für den mwm sollen hier nur einige wenige vorgestellt werden, alle weiteren entnehme man dem Manual.

Wer sich daran stört, daß ein Fenster erst durch Klicken aktiviert wird, kann dies ändern mit

Mwm*KeyboardFocusPolicy: pointer

Danach folgt der Fokus wieder dem Mauszeiger. Der Default-Wert lautet "explicit".

Möchte man seine Icons - ahnlich zum Icon-Manager des twm - schön zusammengefaßt haben, empfiehlt sich die Verwendung der Icon-Box:

Mwm*useIconBox: true

Die Icons werden dann in einem eigenen Fenster gesammelt, dessen Größe und Lage natürlich auch wieder über Resourcen gesteuert werden können. Reicht das Fenster für die Icons nicht aus, werden automatisch Scroll-Balken eingeblendet, mit denen man an die verdeckten Icons kommt. In der Icon-Box ist für jeden Client auf dem Schirm ein Icon vorhanden, nicht nur für die iconisierten. Letztere sind etwas dicker, damit man sieht, welche Clients gerade "zu Hause" sind. Hat man einige Clients beendet, bleiben in der Icon-Box Lücken an den entsprechenden Stellen. Um diese wieder aufzufüllen, hat die Icon-Box in ihrem Window-Menü einen speziellen Eintrag "PackIcons".

Möchte man nicht die Default-Tastenbelegungen benutzen, kann man andere, die im .mwmrc-File definiert wurden, aktivieren, z.B.:

Mwm*buttonBindings: ExplicitButtonBindings

Soll die Uhr keine Verzierung haben, kann man sie davon befreien durch

Mwm*Xclock*clientDecoration: +none

Man kann dann aber die Window-Menü-Funktionen immer noch aufrufen, z.B. "Move" mit <Alt-F7>.

Natürlich kann man die Farben verändern, die Dicke von Rändern usw. Mit dem Ausprobieren aller Möglichkeiten kann man leicht einige Tage verbringen.

7 Ausblick: Was bringt X11R5 ?

Am 5.9.1991 wurde das nächste Release, X11R5, des X-Window-Systems freigegeben. Damit wurde der Standard erweitert, einige neue Konzepte eingeführt und Clients verbessert oder neu entwickelt.

Die wichtigste Änderung ist wohl die Einführung von PEX, einer Erweiterung des X-Protokolls, mit der dreidimensionale Objekte in hierarchischer Weise (auf der Basis von PHIGS) beschrieben werden können. Damit müssen komplizierte 3D-Bilder nicht mehr als Bitmaps zwischen Client und Server übertragen werden, sondern können an Hand von abstrakten Beschreibungen vom Server erstellt werden. Dieses Verfahren verringert nicht nur die Netz-Belastung, sondern erlaubt vor allem die Benutzung spezieller Hardware zur 3D-Darstellung, wie sie in Graphik-Workstations meist vorhanden ist.

Ebenfalls neu ist die Möglichkeit skalierbarer Fonts und die Einführung von Font-Servern. Damit soll - neben verbesserten Darstellungs-Möglichkeiten - vor allem der für Fonts nötige Speicherplatz reduziert werden.

Auch bei der Farbdarstellung wurde manches verbessert: Neben dem RGB-Farbraum werden nun auch andere Farbräume unterstützt, weitere können einfach eingebunden werden. Besonders interessant ist die Möglichkeit geräteunabhängiger Farb-Darstellung: Die Eigenschaften des Farbschirms werden im Server abgelegt, der daraufhin Farben so umrechnen kann, daß sie auf allen Schirmen gleich aussehen.

Natürlich wurde auch an der Programmier-Schnittstelle Xlib und den Athena Widgets einiges getan. Sichtbarstes Zeichen dafür ist das Erscheinungsbild einiger Clients, wie z.B. von editres, mit dem man sich die Widget-Struktur eines Clients als Baum ansehen kann. Wesentlich verbessert wurden auch bitmap und xmag, die jetzt zusammenarbeiten können.

Und während die meisten noch mit der Umstellung auf X11R5 beschäftigt sind (auf den Apollos läuft sogar noch X11R3!), munkelt man schon, was wohl X11R6 bringen wird ...

8 Anhang A: Beispiel für ein .twmrc-File

```
#************************************************************************
#    Beispiel -  .twmrc
#    wird ausgefuehrt beim Aufruf von twm
#************************************************************************

#
# Farbvoreinstellungen fuer
#   Titelbalken (2x wegen Schachbrett-Muster), Schrift im Titelbalken (2x),
#   Menues (2x), Icon-Manager (3x)
#

Color
{
    BorderTileForeground "yellow"
    BorderTileBackground "blue"
    TitleForeground "yellow"
    TitleBackground "blue"
    MenuBackground "pale green"
    MenuForeground "blue"
    IconManagerBackground "PeachPuff1"
    IconManagerForeground "black"
    IconManagerHighlight "red"
}

#
# Randbreite (default: 2)
#

BorderWidth  3

#
# Fontvoreinstellungen fuer
#   Titel, Menue, Resize-Box
#

TitleFont  "9x15"
MenuFont  "9x15"
ResizeFont  "8x13"

#
# Windows ohne Geometrieangaben selbstaendig setzen
#     (ohne eine Benutzer-Plazierung anzufordern)
#
```

```
RandomPlacement

#
# Icon-Verwaltung:
#   1. Der Icon-Manager wird verwendet, andere Icons werden nicht benutzt.
#   2. Die folgenden Programme werden vom Icon-Manager nicht angezeigt:
#        oclock, xbiff, xclock, xload, xpostage .
#      Sie erhalten auch keine Dekoration und koennen somit nicht auf
#      direktem Weg iconisiert werden.
#   3. Iconisierung ist moeglich durch eine Menuefunktion (s.u.). Die
#      Programme unter Punkt 2 verwenden dann einzelne Icons.
#

ShowIconManager
SortIconManager

IconManagerGeometry "=120x10-1+230"
IconManagerFont "vtsingle"
IconManagerDontShow
{
    "oclock"
    "xbiff"
    "xclock"
    "xload"
    "xpostage"
}

IconifyByUnmapping

DontIconifyByUnmapping
{
    "oclock"
    "xbiff"
    "xclock"
    "xload"
    "xpostage"
}

NoTitle
{
    "TWM"
    "oclock"
    "xbiff"
    "xclock"
    "xload"
    "xpostage"
}
```

```
#
# Bereich fuer die (normalerweise nicht auftretenden !) Icons
#

IconRegion    "200x200-5+380" North East 10 10

#
# Reaktion auf Maus- oder Tastendruck ohne explizite Definition:
#    Standard-Menue
#

DefaultFunction f.menu "TWM"

#
# Tasten- und Mausklick-Belegungen
#
# Format:
#    BUTTON = [KEY] : CONTEXT : FUNCTION
#
# Bedeutung:
#    Wenn BUTTON (und ggf. KEY) in der Umgebung CONTEXT (root, icon,
#    am Rand, ...) gedrueckt sind, wird FUNCTION ausgefuehrt.
#

Button1 =      : root   : f.menu "TWM"
Button2 =      : root   : f.menu "Utilities"
Button3 =      : root   : f.menu "Logins"

Button1 =      : frame  : f.raise
Button2 =      : frame  : f.move
Button3 =      : frame  : f.lower

Button1 =      : icon   : f.iconify
Button2 =      : icon   : f.move
Button3 =      : icon   : f.raiselower

Button1 =   s  : window : f.fullzoom

Button3 =   c  : root   : f.function "beep-beep"


#
# selbstdefinierte Funktionen: ein einfaches Beispiel
#

Function "beep-beep"
```

```
{
    f.beep
    f.beep
    f.beep
    f.beep
    f.beep
}


#
# Menue-Definitionen
#


# linke Maustaste:

menu "TWM"
{
"TWM"         f.title
"Iconify"  f.iconify
"Resize"  f.resize
"Move"          f.move
"Raise"          f.raise
"Lower"          f.lower
"Refresh"  f.winrefresh
"--------------"         f.nop
"Destroy Window"  f.destroy
"--------------"         f.nop
"Show Icon Mgr"  f.showiconmgr
"Hide Icon Mgr"  f.hideiconmgr
"--------------"         f.nop
"Restart twm"            f.restart
"Kill twm"  f.quit
}

# mittlere Maustaste:
#   (! ist Abkuerzung fuer f.exec)

menu "Utilities"
{
"X-Utilities"   f.title
"xedit"         ! "xedit &"
"xterm"         ! "xterm -n 'Neues Fenster' &"
"xtermApollo"   ! "xterm -n 'Apollo Fenster' -fn 9x15 -bw 4 -sl 1000 &"
"xclipboard"    ! "xclipboard &"
"xman"          ! "xman &"
"xcalc"         ! "xcalc &"
```

```
}


# rechte Maustaste:

menu "Logins"
{
"Other Logins"  f.title
"tuhhco"        ! "xterm -n telnet@tuhhco -e telnet tuhhco.rz.tu-harburg.de &"
"c2"            ! "xterm -n telnet@c2 -e telnet c22.rz.tu-harburg.de &"
}


#***************************************************************************
#    Ende von .twmrc
#***************************************************************************
```

9 Anhang B: Beispiel für ein Startup-File

```
#***************************************************************************
#    Beispiel - .xsession
#    wird beim Einloggen ueber den xdm ausgefuehrt.
#***************************************************************************

#
# X-Resource-Datenbank laden
#
xrdb   -load ~/.Xdefaults
#
# Rootfenster verschoenern
#
xsetroot -fg orange -bg MidnightBlue -bitmap ~/x/bitmaps/smile
#
# Keyclick, Warnton und Mausparameter setzen
#
xset c 50
xset b 50
xset m 6 2
#
xhost tuhhco.rz c22.rz sg03.cip3s
#
#
# weitere Clients starten: Terminal, Uhr und Postkasten
```

```
#
xterm  -geometry 80x28+5+5 -n ‘hostname‘ -fn 10x20 -i &
oclock &
xbiff &
#
# Window-Manager starten
#
twm &

#*********************************************************************
#    Ende von .xsession
#*********************************************************************
```

10 Anhang C: Beispiel einer Key-Tabelle

```
There are 2 KeySyms per KeyCode; KeyCodes range from 7 to 132.

    KeyCode Keysym (Keysym) ...
    Value   Value   (Name)  ...

      7
      8     0xff1b (Escape)
      9
     10
     11
     12
     13     0xff09 (Tab)
     14     0x0060 (grave) 0x007e (asciitilde)
     15     0xffbf (F2)
     16     0xffbe (F1)
     17     0xffe5 (Caps_Lock)
     18     0xffe1 (Shift_L)
     19
     20     0xffe3 (Control_L)
     21     0x0051 (Q)
     22     0x0031 (1) 0x0021 (exclam)
     23     0xffc0 (F3)
     24
     25     0xffe9 (Alt_L) 0xffe7 (Meta_L)
     26     0x005a (Z)
     27     0x0053 (S)
     28     0x0041 (A)
     29     0x0057 (W)
     30     0x0032 (2) 0x0040 (at)
```

```
31     0xffc1 (F4)
32
33     0x0043 (C)
34     0x0058 (X)
35     0x0044 (D)
36     0x0045 (E)
37     0x0034 (4) 0x0024 (dollar)
38     0x0033 (3) 0x0023 (numbersign)
39     0xffc2 (F5)
40
41     0x0020 (space)
42     0x0056 (V)
43     0x0046 (F)
44     0x0054 (T)
45     0x0052 (R)
46     0x0035 (5) 0x0025 (percent)
47     0xffc3 (F6)
48
49     0x004e (N)
50     0x0042 (B)
51     0x0048 (H)
52     0x0047 (G)
53     0x0059 (Y)
54     0x0036 (6) 0x005e (asciicircum)
55     0xffc4 (F7)
56
57     0x0000 (NoSymbol) 0xffe8 (Meta_R)
58     0x004d (M)
59     0x004a (J)
60     0x0055 (U)
61     0x0037 (7) 0x0026 (ampersand)
62     0x0038 (8) 0x002a (asterisk)
63     0xffc5 (F8)
64
65     0x002c (comma) 0x003c (less)
66     0x004b (K)
67     0x0049 (I)
68     0x004f (O)
69     0x0030 (0) 0x0029 (parenright)
70     0x0039 (9) 0x0028 (parenleft)
71     0xffc6 (F9)
72
73     0x002e (period) 0x003e (greater)
74     0x002f (slash) 0x003f (question)
75     0x004c (L)
76     0x003b (semicolon) 0x003a (colon)
77     0x0050 (P)
```

```
 78     0x002d (minus) 0x005f (underscore)
 79     0xffc7 (F10)
 80
 81
 82     0x0027 (apostrophe) 0x0022 (quotedbl)
 83
 84     0x005b (bracketleft) 0x007b (braceleft)
 85     0x003d (equal) 0x002b (plus)
 86     0xffc8 (F11)
 87     0xff0a (Linefeed)
 88
 89
 90     0xff0d (Return)
 91     0x005d (bracketright) 0x007d (braceright)
 92     0x005c (backslash) 0x007c (bar)
 93
 94     0xffc9 (F12)
 95     0xff6b (Break)
 96     0xff54 (Down)
 97     0xff51 (Left)
 98
 99     0xff52 (Up)
100     0xffff (Delete)
101     0xff57 (End)
102     0xff08 (BackSpace)
103     0xff63 (Insert)
104
105     0xffb1 (KP_1)
106     0xff53 (Right)
107     0xffb4 (KP_4)
108     0xffb7 (KP_7)
109     0xff56 (Next)
110     0xff50 (Home)
111     0xff55 (Prior)
112     0xffb0 (KP_0)
113     0xffae (KP_Decimal)
114     0xffb2 (KP_2)
115     0xffb5 (KP_5)
116     0xffb6 (KP_6)
117     0xffb8 (KP_8)
118     0xff7f (Num_Lock)
119     0xffaf (KP_Divide)
120
121     0xff8d (KP_Enter)
122     0xffb3 (KP_3)
123
124     0xffab (KP_Add)
```

```
    125     0xffb9 (KP_9)
    126     0xffaa (KP_Multiply)
    127
    128
    129
    130
    131
    132     0xffad (KP_Subtract)
```

11 Anhang D: Das system.mwmrc-File

```
#
#  DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc and .mwmrc)
#


#
# menu pane descriptions
#

# Root Menu Description
Menu RootMenu
{
"Root Menu" f.title
"New Window" f.exec "xterm &"
"Shuffle Up" f.circle_up
"Shuffle Down" f.circle_down
"Refresh" f.refresh
no-label f.separator
"Restart..." f.restart
}


# Default Window Menu Description

Menu DefaultWindowMenu
{
Restore _R Alt<Key>F5 f.normalize
Move _M Alt<Key>F7 f.move
Size _S Alt<Key>F8 f.resize
Minimize _n Alt<Key>F9 f.minimize
Maximize _x Alt<Key>F10 f.maximize
Lower _L Alt<Key>F3 f.lower
```

```
no-label f.separator
Close _C Alt<Key>F4 f.kill
}


#
# key binding descriptions
#

Keys DefaultKeyBindings
{
Shift<Key>Escape window|icon f.post_wmenu
Meta<Key>space window|icon f.post_wmenu
Meta<Key>Tab root|icon|window f.next_key
Meta Shift<Key>Tab root|icon|window f.prev_key
Meta<Key>Escape root|icon|window f.next_key
Meta Shift<Key>Escape root|icon|window f.prev_key
Meta Shift Ctrl<Key>exclam root|icon|window f.set_behavior
Meta<Key>F6 window f.next_key transient
Meta Shift<Key>F6 window f.prev_key transient
    <Key>F4 icon f.post_wmenu
}

#
# button binding descriptions
#

Buttons DefaultButtonBindings
{
<Btn1Down>icon|frame f.raise
<Btn3Down>icon f.post_wmenu
<Btn1Down>root f.menu RootMenu
}

Buttons ExplicitButtonBindings
{
<Btn1Down>frame|icon f.raise
<Btn3Down>frame|icon f.post_wmenu
<Btn1Down>root f.menu RootMenu
Meta<Btn1Down>window|icon f.lower
! Meta<Btn2Down>window|icon f.resize
! Meta<Btn3Down>window|icon f.move

}

Buttons PointerButtonBindings
{
```

```
<Btn1Down>frame|icon f.raise
<Btn3Down>frame|icon f.post_wmenu
<Btn1Down>root f.menu RootMenu
<Btn1Down>window f.raise
Meta<Btn1Down>window|icon f.lower
! Meta<Btn2Down>window|icon f.resize
! Meta<Btn3Down>window|icon f.move
}

#
#  END OF mwm RESOURCE DESCRIPTION FILE
#
#*****************************************************************************
#    Ende von system.mwmrc
#*****************************************************************************
```

12 Anhang E: Steuercodes des xedit und der Athena-Textwidgets

Taste	Funktion
Control-a	Cursor an den Anfang der Zeile setzen
Control-b	Cursor ein Zeichen nach links
Control-d	Zeichen rechts vom Cursor löschen
Control-e	Cursor an das Ende der Zeile setzen
Control-f	Cursor ein Zeichen nach rechts
Control-g	Vielfachoperation abbrechen
Control-h	Zeichen links vom Cursor löschen
Control-j	neue Zeile mit Einrückung
Control-k	Ab Cursor bis zum Ende der Zeile löschen (kill)
Control-l	Textfelder neuzeichnen
Control-m	Neue Zeile
Control-n	Cursor eine Zeile tiefer
Control-o	Rechts vom Cursor eine neue Zeile einfügen
Control-p	Cursor eine Zeile höher
Control-r	Suchen/Ersetzen rückwärts
Control-s	Suchen/Ersetzen vorwärts
Control-t	Zeichen rechts und links vom Cursor vertauschen
Control-u	nächste Aktion 4 mal ausführen
Control-v	Cursor an den Anfang der nächsten Seite
Control-w	Ausgewählten Bereich löschen (kill)
Control-y	Gelöschten Bereich einfügen (unkill)
Control-z	Fenster eine Zeile nach oben rollen
Meta-b	Cursor ein Wort nach links
Meta-f	Cursor ein Wort nach rechts
Meta-i	Datei einfügen
Meta-k	Bis zum Ende des Abschnitts löschen (kill)
Meta-q	Abschnitt formatieren
Meta-v	Seite zurückblättern
Meta-y	Momentan selektierten Bereich einfügen
Meta-z	Fenster eine Zeile nach unten rollen
Meta-d	Wort rechts vom Cursor löschen
Meta-D	Wort rechts vom Cursor löschen (kill)
Meta-h	Wort links vom Cursor löschen
Meta-H	Wort links vom Cursor löschen (kill)
Meta-<	Cursor an den Anfang der Datei
Meta->	Cursor an das Ende der Datei
Meta-[	Cursor an den Anfang des Abschnittes
Meta-]	Cursor an das Ende des Abschnittes
Meta-Delete	Wort links vom Cursor löschen
Meta-Shift-Delete	Wort links vom Cursor löschen (kill)
Meta-Backspace	Wort links vom Cursor löschen
Meta-Shift-Backspace	Wort links vom Cursor löschen (kill)