Yak

COLLABORATORS			
	TITLE : Yak		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		September 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Yak

1.1 Yak v1.60 Anleitung

```
                          Yak Version 1.60
     von Gaël Marziou und Philippe Bastiani
      veröffentlicht am 25. September 1994

INHALTSVERZEICHNIS
```

1.2 Copyright

Yak darf frei weitergegeben werden. Der Quellcode liegt dem Programmpaket bei und darf nach Belieben für den persönlichen Gebrauch verändert werden. Eine solche veränderte Version darf allerdings NICHT weitergegeben werden. Wenn Sie Veränderungen vornehmen, die auch andere interessieren könnten, schicken Sie mir bitte die Änderungsvorschläge, so daß ich sie in zukünftigen Versionen berücksichtigen kann.

Da Yak kostenlos ist, gibt es KEINE GARANTIE für den einwandfreien Betrieb. Der Autor ist nicht verantwortlich für irgendeinen Verlust oder Schaden, der durch die Benutzung von Yak entsteht. Sie benutzen Yak auf eigene Gefahr.

Yak darf nicht verkauft werden; erlaubt ist lediglich eine Kopiergebühr von nicht mehr als 5 DM. Außerdem darf Yak nicht ohne vorherige Zustimmung seitens des Autors zusammen mit einem kommerziellen Produkt vertrieben werden. Yak darf ausschließlich weitergegeben werden, wenn die gesamte Anleitung vorhanden ist und nicht verändert wurde; außerdem sollte der Quellcode dabei sein. Fred Fish hat ausdrücklich die Erlaubnis, Yak in seine hervorragende Public-Domain-Serie zu übernehmen.

Obwohl Yak "freeware" ist, bin ich kleinen Aufmerksamkeiten (z.B. Geld oder selbstgeschriebene Programme) gegenüber nicht abgeneigt.

Siehe auch:
Adresse des Autors
.

1.3 Wichtig, wenn Sie bereits Yak benutzen

Das Format der Voreinsteller-Datei für Yak hat sich erneut geändert
insofern, als es in die zwei Dateien "S:Yak.prefs" und "S:Yak.hotkeys" geteilt wurde. Das beigelegte Programm "Convert" dient zur Umwandlung einer alten 1.3/1.4-Voreinsteller-Datei in die äquivalenten 1.5-Dateien. Es

gibt allerdings zwei kleine Mängel bei der Umwandlung: Bei den Tastenbefehlen müssen Sie die Eintragungen für "DOS-Befehl", sowie die Format-Zeichenfolge für das Datum erneut eingeben (siehe Abschnitt
Tastenbefehle
für
detaillierte Informationen darüber); der zweite Mangel ist der, daß ein spezieller Tastenbefehl für das "Yak Einsteller"-Fenster erzeugt wird, der im Grunde genommen überflüssig ist, da hierfür das CX_POPKEY-Merkmal benutzt wird (s.u.).

Es gibt einige wichtige Unterschiede zwischen Yak V1.4 und 1.5. Das wichtigste zuerst: Yak benötigt jetzt einen Stack von 4500 Bytes! Vergessen Sie also nicht, gegebenenfalls in Yaks Piktogramm die Stack-Größe anzupassen.

Zweitens hat Yak jetzt nur noch dann ein Applikationpiktogramm, wenn Sie ausdrücklich das Merkmal "APPICON=TRUE" eintragen. Dieses Merkmal ersetzt das "NOICON"-Merkmal älterer Versionen.

Drittens können Sie über das Merkmal "CX_POPKEY" angeben, welcher Tastenbefehl das "Yak Einsteller"-Fenster öffnet. Dieser Tastenbefehl funktioniert auch, wenn über Tastenbefehls-Definitionen ein anderer eingestellt ist. In diesem Fall existieren dann mehrere Tastenbefehle, die Yaks Fenster öffnen.

Viertens ist Yak nun lokalisiert und unterstützt jetzt Englisch (als eingebaute Sprache), sowie Französisch, Italienisch, Schwedisch, Niederländisch, Dänisch und Deutsch über die mitgelieferten Kataloge. Siehe auch den Abschnitt
Lokalisation
.

Natürlich gibt’s noch eine Menge anderer Änderungen und Neuheiten. Daher sollten Sie in dieser Anleitung etwas herumschmökern, um mit der Benutzung von Yak vertraut zu werden.

1.4 Einleitung

Yak hat eigentlich nichts mit dem gleichnamigen Rindvieh zu tun ↩
(bis auf
das Piktogramm :-), sondern ist eine Abkürzung für "Yet Another Kommodity", wobei sich der Rechtschreibfehler natürlich nur zufällig eingeschlichen hat (siehe auch
Witze
). Yak ist ein Maus- und Fenster-Manipulator der Gattung DMouse, MightyMouse usw.

Warum also noch ein weiteres Programm?! Keines der anderen (und ich habe wirklich (fast) alle anderen ausprobiert) hat mir so richtig gefallen. Yak hat folgende Eigenschaften:

- AutoPoint (sunmouse, automatische Fenster-Aktivierung), wobei nur aktiviert wird, wenn die Maus anhält
  Sie können auch angeben, welche Schirme sich so aktivieren lassen
  (Kompatibel mit Programmen der Art "Popup-Menü")

- AutoPop von Fenstern, d.h. Fenster kommen nach vorn, wenn sie

(automatisch) aktiviert werden

- KeyActivate: Durch Tastendruck wird ein Fenster aktiviert

- Fenster durch Menütaste der Maus aktivieren

- Fenster nach vorn oder hinten klicken. Sie können angeben, für welche Fenster auf welchen Schirmen das geht

- Zyklisches Vertauschen der Schirme mit der Maus

- Bildschirmschoner und Ausblenden des Mauszeigers (auch durch Tastenbefehl)

- Erweiterbares Tastenbefehls-System mit folgenden Befehlen:

  Ausführen eines DOS-Befehls
  Einfügen von Text (per eingebautem Tastenbefehl)
  Einfügen des Datums (ggf. im lokalisierten Format)
  Schließen/"Zippen"/Verkleinern/Vergrößern von Fenstern
  Zyklisches Vertauschen von Schirmen und Fenstern
  Workbench aktivieren
  Den vordersten Schirm zentrieren
  Bildschirmschoner
  Palette auf vorderstem Schirm öffnen (benötigt reqtools)
  Standard Öffentlichen Schirm setzen

- Tastenklicken mit einstellbarer Lautstärke

- Klick-Unterdrückung bei Laufwerken

- Benutzung von '*' als Jokerzeichen (wie bei StarBurst)

- Optionales Applikationspiktogramm für Voreinsteller-Fenster

- Schönes Einsteller-Fenster à la gadtools

Kommt Ihnen das vertraut vor? Yak ist eine Verbindung der Standard-Commodities AutoPoint, ClockToFront, Blanker und IHelp/FKey (auf der "Extras"-Diskette), gemischt mit KCommodity und DMouse.

Und Yak ist ziemlich klein: Es verbraucht nur ca. 29K auf der Diskette und etwa genau soviel Arbeitsspeicher. Es erreicht diese Speicher-Effizienz durch "overlays", d.h. der Programmteil für das Einsteller-Fenster wird nur bei Bedarf geladen. Dann allerdings braucht Yak mehr RAM (etwa 45K).

1.5 Witze

Vielleicht sind diese Witze alt, aber insbesondere der erste dürfte kaum an Aktualität verlieren... ;)

Frage: Wie wechseln die Software-Klempner von Mikrosaft eine defekte Glühbirne aus?
Antwort: Gar nicht. Sie definieren Dunkelheit als Industriestandard.

Meschen sind in gewisser Hinsicht mit dem Computer vergleichbar:
Alle Menschen haben einen hohen IQ --- nur manche machen nichts draus. Sie laufen unter MS-DOS.

1.6 Beschränkungen

Yaks Bildschirmschoner schaltet nur den Schirm dunkel. Seit der Einführung von OS 2.0 gibt es ansprechende Bildschirmschoner in Hülle und Fülle wie z.B. Spliner, ASwarm, FracBlank u.a. Ich glaube, die meisten haben bereits ihren Lieblings-Bildschirmschoner, darum habe ich keinen in Yak eingebaut.

Es gibt keinen Mausbeschleuniger. Der Standardbeschleuniger, der über den Input-Voreinsteller veränderbar ist, ist meiner Meinung nach völlig ausreichend. Falls ich allerdings genug Anfragen diesbezüglich bekomme, werde ich eine schnellere Beschleunigung einbauen (allerdings hat bisher so gut wie keiner danach gefragt).

1.7 Yak starten

Yak läßt sich am einfachsten vom "SYS:WBStartup"-Verzeichnis aus ↩
starten,
jedoch ist auch ein Aufruf mittels 'run'-Befehl von der Shell aus möglich (was allerdings nicht empfehlenswert ist, da das bedeutend mehr Speicher verbraucht). Alle Yak-Einstellungen werden in zwei Dateien gespeichert, nämlich der Voreinsteller-Datei "S:Yak.prefs" sowie der TastenbefehlsDatei "S:Yak.hotkeys". Am besten starten Sie das Installations-Skript, wenn Sie Yak zum ersten Mal benutzen. Sie können anschließend Yak so konfigurieren, wie Sie es gerne wollen, und diese Einstellungen für spätere Aufrufe speichern.

Übrigens: Yak benötigt einen 4500 Byte großen Stack, der in Yaks Piktogramm eingetragen sein sollte. Falls Sie Yak von der Shell aus starten, benutzen Sie folgende Befehle:

```
stack 4500
run >NIL: yak
```

Die einzigen Merkmale, die Yak unterstützt, sind die Standard-Merkmale für die Workbench (z.B. DONOTWAIT), für Commodities (z.B. CX_POPUP) und für Applikations-Piktogramme, sowie das LANGUAGE-Merkmal. Hier eine Übersicht:

Merkmal	Kategorie	Beschreibung	Normal
CX_POPKEY	Tastenbefehl	Taste für Einsteller-Fenster	RCommand Help
CX_PRIORITY	ganze Zahl	Commodity-Priorität für Yak	0
CX_POPUP	TRUE/FALSE	Einsteller-Fenster bei Start	FALSE
APPICON	TRUE/FALSE	Wenn Ja: mit Appl.-Piktogramm	FALSE
ICONNAME	Zeichenfolge	Name des Appl.-Piktogramms	"Yak!"
ICONXPOS	ganze Zahl	x-Position des Appl.-Piktogramms	sucht WB aus

```
ICONYPOS     ganze Zahl    y-Position des Appl.-Piktogramms   sucht WB aus

LANGUAGE     Zeichenfolge  Name der zu benutzenden Sprache  nicht vorhanden
```

Sie sollten zusätzlich das Merkmal DONOTWAIT setzen, wenn Sie Yak aus dem "SYS:WBStartup"-Verzeichnis starten wollen.

Die Erzeugung eines Applikations-Piktogramms ist optional und normalerweise abgeschaltet. Wenn Sie es benutzen möchten, geben Sie APPICON=TRUE an; In diesem Fall legt Yak ein Applikations-Piktogramm auf die Workbench. Wenn Sie es doppelt anklicken, öffnet sich das Einsteller-Fenster. Wie dieses Piktogramm aussieht, richtet sich nach dem Piktogramm, von dem aus Yak gestartet wurde. Auf diese Weise können Sie das Applikations-Piktogramm an Farbe und Auflösung Ihrer Workbench anpassen, indem Sie einfach Yaks Piktogramm entsprechend verändern.

Das CX_PRIORITY-Merkmal könnte hilfreich sein, wenn Sie die Zusammenarbeit von Yak mit anderen Commodities verbessern wollen. Für ein Beispiel schauen Sie sich am besten die Hinweise zum
RMB aktivieren
-Schalter an.

Das LANGUAGE-Merkmal habe ich für solche hinzugefügt, die Yak in einer anderen Sprache benutzen wollen als die Workbench. Wenn Sie also z.B. eine englische Workbench gewohnt sind, Yak jedoch auf deutsch benutzen möchten, brauchen Sie einfach nur

```
LANGUAGE=deutsch
```

zu setzen. Wenn Sie allerdings für beides mit derselben Sprache zufrieden sind, können Sie dieses Merkmal getrost vergessen.

1.8 Yaks Einstellungen ändern

Yaks Einstellungen lassen sich alle in seinem Einsteller- ↩
Fenster ändern.
Sobald Yak läuft, können Sie dieses Fenster wie folgt öffnen:

Drücken Sie RCommand Help (d.h. die rechte Amiga-Taste zusammen mit der Help-Taste)
Dieser Tastenbefehl läßt sich (über das
CX_POPKEY
-Merkmal) konfi-
gurieren

Doppelklicken Sie auf Yaks Applikations-Piktogramm (falls sichtbar)

Starten Sie Yak ein zweites Mal

Benutzen Sie "Exchange" (von der "Extras"-Diskette)

Das dann geöffnete Fenster enthält viele Gadgets (Symbole), die in Klassen eingeteilt sind, nämlich folgende:

Schalter

Muster

Schoner

Verschiedenes

Tastenbefehle

Beachten Sie, daß Sie in einem Texteingabefeld die RETURN-, ↩
ENTER- oder
TABULATOR-Taste drücken müssen, damit die Eingabe registriert wird. Einfach neben das Feld zu klicken, führt nicht zu einer Übernahme der gemachten Eingabe. (Die TAB-Taste aktiviert automatisch das nächste Texteingabefeld.)

Ebenfalls gestattet Intuition normalerweise nicht das Ausschneiden und Einfügen von einem Texteingabefeld in ein anderes (das gilt insbesondere für das Tastenbefehls-Fenster :(Wenn Sie gerne eine solche Möglichkeit hätten, sollten Sie sich das Programm "NewEdit" von Uwe Roehm besorgen, das Ausschneiden in und Einfügen aus dem Zwischenspeicher erlaubt (Sie finden das Programm auf einer Fish-Disk und auf diversen ftp-sites).

Außerdem ist noch anzumerken, daß die Symbole "Verbergen" und "Entfernen" die Standard-Operationen bei Commodities ausführen, nämlich das Einsteller-Fenster zu schließen (Yak läuft weiter) bzw. Yak ganz zu entfernen. Das Schließ-Symbol hat den gleichen Effekt wie "Verbergen"; Yak wird dadurch nicht abgebrochen.

Das Einsteller-Fenster besitzt zusätzlich noch ein Menü, u.a. mit den Menüpunkten "Verbergen" und "Entfernen" (die völlig mit den entsprechenden Gadgets übereinstimmen). Des weiteren gibt es im Menü die Punkte "Laden" und "Speichern". Alle Änderungen an Yaks Einstellungen sind futsch, wenn Sie sie nicht durch "Speichern" dauerhaft festlegen. Um die zuletzt gespeicherten Einstellungen zurückzuerhalten, können Sie den Menüpunkt "Laden" auswählen.

1.9 Schalter

Es gibt folgende Schalter:

Auto-Aktivierung

Autom. nach vorn

Tasten-Aktivierung

Klick nach vorne

Klick nach hinten

MMB-Aktivierung

RMB-Aktivierung

Schirm-Zyklus

Schirm aktivieren

MMB -> Shift links

1.10 Auto-Aktivierung

Aktiviere das Fenster unter der Maus. Der Effekt ist fast der gleiche wie
beim "AutoPoint"-Commodity von Commodore, außer daß Yak ein Fenster nur dann aktiviert, wenn die Maus anhält. Auto-Aktivierung ist mit Programmen der Art "Popup-Menü" (wie z.B. dem hervorragenden "MagicMenu") kompatibel.

Beachten Sie, daß die Funktionen "Auto-Aktivierung" und "Autom. nach vorn" nur dann ausgeführt werden, wenn KEIN Qualifier (Maus oder Tastatur) gedrückt ist. Dadurch werden nicht nur Schwierigkeiten mit anderen Programmen vermieden, sondern Sie haben so eine Möglichkeit, die beiden Funktionen "Auto-Aktivierung" und "Autom. nach vorn" bei Bedarf zu unterdrücken.

Siehe auch:

Autom. nach vorn

Auto-aktivierb. Schirme

1.11 Autom. nach vorn

Funktioniert nur, wenn Auto-Aktivierung eingeschaltet ist. In diesem Fall
aktiviert Yak das Fenster und bringt es auch automatisch nach vorne. Ist die Maus allerdings über einem Fenster, das einen Requester anzeigt (keine AutoRequester, die in einem extra Fenster angezeigt werden, wie z.B. Workbench-Dialogfenster), ist "Autom. nach vorn" ohne Wirkung.

Siehe auch:

Auto-Aktivierung

Auto-aktivierb. Schirme

Popup-Fenster

1.12 RMB-Aktivierung

Ist dieser Schalter an, wird das Fenster unter der Maus auch durch Drücken der rechten Maustaste aktiviert --- unabhängig davon, ob Auto-Aktivierung an ist oder nicht. Dadurch bekommen Sie immer das richtige Menü, ohne auf Auto-Aktivierung des Fensters warten zu müssen oder es durch Anklicken zu aktivieren.

Wenn Sie z.B. das Vorder-/Hintergrund-Symbol eines Bildschirms benutzen, ist der neue vorderste Schirm nicht aktiviert, jedoch bekommen Sie (bei eingeschalteter RMB/MMB-Aktivierung) mit der Menütaste das richtige Menü.

Hinweise:
1) Falls sich kein Fenster unter der Maus befindet, wird das erste Fenster auf dem Schirm aktiviert.

2) Für eine korrekte Zusammenarbeit mit Programmen der Art "PopMenü", muß Yaks CX_PRIORITY u.U. höher sein als die CX_PRIORITY des "PopMenü"-Programms.

1.13 MMB-Aktivierung

Ist dieser Schalter an, wird das Fenster unter der Maus auch durch Drücken der mittleren Maustaste aktiviert --- unabhängig davon, ob Auto-Aktivierung an ist oder nicht. Dadurch bekommen Sie immer das richtige Menü, ohne auf Auto-Aktivierung des Fensters warten zu müssen oder es durch Anklicken zu aktivieren.

Wenn Sie z.B. das Vorder-/Hintergrund-Symbol eines Bildschirms benutzen, ist der neue vorderste Schirm nicht aktiviert, jedoch bekommen Sie (bei eingeschalteter MMB-Aktivierung) mit der Menütaste das richtige Menü.

Hinweis: Falls sich kein Fenster unter der Maus befindet, wird das erste Fenster auf dem Schirm aktiviert.

1.14 Tasten-Aktivierung

Diese Funktion aktiviert auf Tastendruck (Tastatur) das Fenster unter der Maus. Sie brauchen nur entweder Auto-Aktivierung oder Tasten-Aktivierung.

1.15 Klick nach vorne

Hiermit können Sie durch Doppelklick auf ein Fenster dieses in ↩ den Vorder-
grund (d.h. vor alle anderen) holen.

Wie auch bei den Funktionen "Auto-Aktivierung" und "Autom. nach vorn" geht diese Funktion nur, wenn kein Tastatur-Qualifier gedrückt ist.

Wenn
Schirm-Zyklus
aktiviert ist, funktioniert "Klick nach vorne" auch bei Schirmen.

Siehe auch:
Klickbare Schirme

Klickbare Fenster

1.16 Klick nach hinten

Hiermit können Sie ein Fenster in den Hintergrund (d.h. hinter alle anderen Fenster) bringen, indem Sie die linke Maustaste gedrückt halten und dann die rechte Maustaste betätigen.

Wie auch bei den Funktionen "Auto-Aktivierung" und "Autom. nach vorn" geht diese Funktion nur, wenn kein Tastatur-Qualifier gedrückt ist.

Wenn
Schirm-Zyklus
aktiviert ist, funktioniert "Klick nach hinten" auch bei
Schirmen.

Siehe auch:
Klickbare Schirme

1.17 Schirm-Zyklus

Mit derselben Maustasten-Kombination wie bei "Klick nach hinten" ("Klick nach vorne") wird ein ganzer Schirm hinter (vor) alle anderen Schirme gebracht unter der Bedingung, daß

- das Fenster unter der Maus ein fixiertes Hintergrund-Fenster ist (wie das Workbench-Hauptfenster)
- oder
- es nur ein Fenster auf dem Schirm gibt.

Siehe auch:
Klick nach hinten

Klick nach vorne

1.18 Schirm aktivieren

Ist dieser Schalter an, aktiviert Yak automatisch einen Schirm, der durch Tastenbefehle umgeordnet wird (z.B. LCommand m bzw. LCommand n). Dieser Schalter ist nötig, um Konflikte mit anderen Programmen zu vermeiden. Die Funktion ist dem Programm "WindX" von Steve Tibbet ähnlich, aber nicht damit identisch.

'Aktivieren eines Schirms' soll bedeuten, daß auf dem neuen vordersten Schirm das Fenster aktiviert wird, das sich unter dem Mauszeiger befindet. Dieses Fenster kann sein:

- das Fenster, das aktiv war, als Sie das letzte Mal mit Yaks Tastenbefehlen für "Schirm nach vorn/hinten" auf diesem Schirm waren.

- das Fenster unter dem Mauszeiger, wenn Sie vorher noch nie auf diesem

Schirm waren

- das erste Fenster dieses Schirms, wenn beide obigen Bedingungen nicht zutreffen

1.19 AmigaDOS *

Wenn angeschaltet, kann ein '*' unter AmigaDOS als Ersatz für '#?' verwendet werden (wie unter MS-DOS und UNIX). (Das StarBurst-Programm tut genau dies).

1.20 Kein Laufwerk-Klicken

Wenn angeschaltet, unterbindet Yak das (nervende) Klicken von leeren Disketten-Laufwerken.

1.21 Muster

Die Muster-Eingabefelder erlauben Ihnen, zu bestimmen, welche ↩ Schirme und
Fenster durch Yaks verschiedene Eigenschaften beeinflußt werden. Alle Muster sind einschließend, d.h. ein Muster muß auf einen Schirm oder ein Fenster passen, damit die betreffende Funktion angewendet wird. Als Muster können Sie die Standard-
AmigaDOS-Muster
verwenden.

Folgende Muster stehen zur Verfügung:

Auto-aktivierb. Schirme

Klickbare Schirme

PopUp-Fenster

Klickbare Fenster

1.22 Auto-aktivierb. Schirme

Auto-Aktivierung
funktioniert für alle Schirme, auf deren
Standard-Titel
dieses Muster paßt.

1.23 Klickbare Schirme

```
                Klick nach vorne
                 und
                Klick nach hinten
                 funktionieren  auf allen Schirmen,
auf deren
                Standard-Titel
                 Titel dieses Muster paßt.
```

1.24 Standard-Titel eines Schirms

```
                Der Standard-Titel eines Schirms  ändert sich nicht.  Programme  ↩
                   können zwar
über Fenster den Titel, nicht aber den Standard-Titel eines Schirms manipu-
lieren.
Beispielsweise gibt es viele Programme,  die den Workbench-Titel verändern,
wenn eines  der zu diesem Programm gehörenden Fenster  aktiv ist,  aber der
Standard-Titel des Workbench-Schirms bleibt davon unberührt.
Wenn Sie also ein Muster  für Schirm-Titel setzen,  sollten Sie den betref-
fenden Standard-Titel zugrunde legen.  (Dieser erscheint in der Titelzeile,
wenn kein Fenster auf dem Schirm aktiv ist.)
In fast allen Fällen ist es nicht schwer,  den Standard-Titel zu ermitteln.
In ganz hartnäckigen Situationen kann da z.B. ARTM recht hilfreich sein.

Siehe auch:
                AmigaDOS-Muster
```

1.25 PopUp-Fenster

```
                Autom. nach vorn
                 funktioniert  bei allen Fenstern,  auf deren Titel dieses
Muster paßt.
```

1.26 Klickbare Fenster

```
                Klick nach vorne
                 funktioniert  bei allen Fenstern,  auf deren Titel dieses
Muster paßt.
```

1.27 Verschiedenes

```
                In diesem Fenster können einige andere Eigenschaften verändert  ↩
                   werden.  Und
zwar:
```

Lautstärke

Auto-Aktiv-Verzögerung

AmigaDOS *

Schwarzer Rand

Kein Laufwerk-Klicken

1.28 Lautstärke

Hier wählen Sie die Lautstärke für Tasten-Klicke (das Geräusch, das erzeugt werden soll, wenn Sie eine Taste drücken). Wenn Sie Null eintragen, bedeutet das 'kein Klicken' (das ist zwar selbstverständlich, allerdings wird dann das audio.device gar nicht erst geöffnet).

Die maximale Lautstärke ist 64.

1.29 Auto-Aktiv-Verzögerung

Die Maus muß die angegebene Zeitspanne lang angehalten werden, bevor das Fenster unter der Maus automatisch aktiviert wird. Diese Zeitspanne darf zwischen 0 und 5 (jeweils einschließlich) liegen und gibt die Verzögerung in 10ms (1/100 Sekunden) an.

Dabei bedeutet eine null: sofortige Aktivierung. Klar, oder?

1.30 MMB -> Shift links

Yak simuliert beim Drücken der mittleren Maustaste (einer 3-Tasten-Maus) die linke Shift-(Umschalt-)Taste der Tastatur, wenn dieser Schalter aktiviert ist.
Genauer gesagt passiert das nur, wenn bei gedrückter mittlerer Maustaste die linke Maustaste ebenfalls gedrückt wird. So wird es einfacher, mehrere Piktogramme auf der Workbench bzw. mehrere Dateien in einem Dateiauswahlfenster anzuklicken, da dafür keine Eingabe über Tastatur mehr erforderlich ist.
Falls noch Funktionsbelegungen der mittleren Maustaste in anderem Zusammenhang existieren, bleiben diese hiervon unberührt.

1.31 Schwarzer Rand

Wenn dieser Schalter angeschaltet ist, erzeugt Yak auf allen Schirmen einen schwarzen Rand. Allerdings funktioniert das nur ab Kickstart 3.0+, da hierfür ein neues Flag der graphics.library benutzt wird.

1.32 Schoner

Zeiger ausblenden

Zeiger bei Tastendruck ausblenden

Abschaltzeiten

1.33 Zeiger ausblenden

Dieses Symbol gibt an, auf welche Art der Mauszeiger ausgeblendet werden soll. "Nicht" schaltet diese Funktion ab, "Durch Sprites" bedeutet Ausblenden durch Abschalten aller Sprites. Mit "Durch Copper" wird der Mauszeiger durch Manipulation der Copper-Liste ausgeblendet. Diese letzte Methode schaltet nur Sprite 0 (den Mauszeiger) ab, so daß Terminal-Programme, die ein Sprite für den Cursor verwenden, problemlos funktionieren. Allerdings ist diese Methode nicht ganz stabil (hin und wieder erscheint der Zeiger plötzlich wieder).

Ausblenden "Durch Copper" bereitet allerdings bei AGA-Maschinen Probleme, wenn als Maus-Zeiger ein HighRes-Sprite verwendet wird (benutzen Sie ggf. "Durch Sprites").

Siehe auch:
Probleme

1.34 Zeiger bei Tastendruck ausblenden

Beim Drücken einer Taste wird sofort der Mauszeiger ausgeblendet, wenn dieser Schalter angschaltet ist.

1.35 Abschaltzeiten

Mit diesen Eingabefeldern können Sie die Abschaltzeiten für Bildschirm und Maus einstellen.

Bildschirm

Maus

1.36 Bildschirm-Abschaltzeit

Wenn über diesen Zeitraum (in Sekunden) keine Benutzer-Eingabe (Maus oder Tastatur) erfolgt, wird der Bildschirm dunkel geschaltet. Setzen Sie den Wert auf 0, wenn Sie keine Abschaltung wünschen. Das Dunkel-Schalten wird durch einen 2-Farb-Bildschirm im Darstellungs-Modus des vordersten Schirms erzeugt.

1.37 Maus-Abschaltzeit

Wenn die Maus über diesen Zeitraum hinweg nicht bewegt wird, ↩
verschwindet
der Mauszeiger. Das geht natürlich nur, wenn
Zeiger ausblenden
nicht auf
"Nicht" eingestellt ist.

1.38 Tastenbefehle

Wenn Sie auf das "Tastenbefehle..."-Symbol klicken, öffnet sich ↩
ein neues
Fenster, in dem Sie Tastenbefehle erstellen, abändern und auch löschen können. Ein Tastenbefehl bewirkt, daß Yak jedesmal, wenn diese Taste gedrückt wird, eine bestimmte Aktion ausführt (wovon es eine ganze Reihe gibt).

Tastenbefehle werden mit Hilfe einer beschreibenden Zeichenfolge definiert, was eine recht flexible Art ist, Eingabe-Ereignisse zu beschreiben.

Sie können beliebig viele Tastenbefehle verwalten, da jeder Aktion mehrere Tastenbefehle zugeordnet werden können. (Für Haarspalter: es sind natürlich NICHT beliebig viele. Die Zahl ist begrenzt, da es nur endlich viele Tasten auf der Tastatur und der Maus gibt :)

Im Tastenbefehls-Fenster gibt es zwei Listen: Die linke zeigt die verfügbaren Aktionen, während in der rechten die zur Zeit für die ausgewählte Aktion definierten Tastenbefehle eingetragen sind.

Um einen neuen Tastenbefehl hinzuzufügen, wählen Sie zunächst (durch Anklicken in der linken Liste) diejenige Aktion aus, die ausgeführt werden soll. Dann klicken Sie auf das "Neu"-Symbol unter der Tastenbefehls-Liste. Das Texteingabefeld wird nun aktiv, und Sie können die Beschreibung für den Tastenbefehl eintippen.

Bei einigen Aktionen werden zusätzliche Symbole aktiv, wenn Sie den Tastenbefehl definieren. Das "Optionen"-Symbol (das Blättersymbol unterhalb der Aktionen-Liste) gibt an, was mit Bildschirmen passiert, wenn der Tastenbefehl gedrückt wird. Zur Zeit gibt es die Optionen "Kein Schirmwechsel", "Workbench nach vorn", sowie "Öffentl. Schirm vor" (der standardmäßige öffentliche Schirm kommt in den Vordergrund). Üblicherweise wird hiervon bei der "DOS-Befehl"-Aktion Gebrauch gemacht, damit automatisch der Schirm erscheint, auf dem sich das Fenster öffnet.

Das "Argument"-Eingabefeld wird bei bestimmten Aktions-Typen aktiv und erlaubt, eine Zeichenfolge mit dem Tastenbefehl zu verbinden (z.B. wird hier der auszuführende Befehl für die "DOS-Befehl"-Aktion eingetragen).

Einen bereits existierenden Tastenbefehl ändern Sie, indem Sie ihn anklikken und dann die betreffenden Eintragungen verändern. Natürlich wird mit "Löschen" der Tastenbefehl aus der Liste entfernt.

Wenn Sie mit den Einstellungen für die Tastenbefehle zufrieden sind, gelangen Sie durch Anklicken des "Zurück..."-Symbols wieder in das Hauptfenster. Sie können aber auch auf das Schließsymbol klicken, um Yak zu 'verbergen'. Vergessen Sie nicht, die gemachten Änderungen mittels des "Speichern"-Menüpunkts im Hauptfenster für spätere Yak-Aufrufe zu übernehmen.

Siehe auch:
Tastenbefehls-Aktionen

Tastenbefehls-Beschreibung

1.39 Tastenbefehls-Beschreibung

Hier können Sie die von Yak benutzten Tastenbefehle definieren.

Die augenblickliche Tastenbefehls-Definition wird in einem Eingabefeld angezeigt. Auf zwei einfache Arten können Sie einen Tastenbefehl definieren:

Automatische Definition

Manuelle Definition
Wenn Sie mit der Definition zufrieden sind, klicken Sie auf 'Ok', ↩
um diesen
neuen Tastenbefehl zu übernehmen.
Durch 'Abbruch' wird die Definition abgebrochen.

Siehe auch:
Probleme

1.40 Automatische Definition

Wählen Sie einfach die Klasse des Tastenbefehls mittels des linken ↩
Blätter-
symbols aus und drücken Sie anschließend die gewünschten (Maus-)Tasten, die den Tastenbefehl aktivieren sollen. Yak wandelt diese automatisch in die korrekte Zeichenfolge um.

Wenn Sie sich beim
Qualifier
vertippt haben, brauchen Sie die betreffende
Taste lediglich ein zweites Mal zu drücken. Sie wird dann aus der Beschreibung entfernt.

Mit dem 'Art'-Symbol (rechtes Blättersymbol) bestimmen Sie, ob der Befehl
beim Drücken und/oder Loslassen der Tastenkombination ausgeführt wird.

Siehe auch:
Mehr über Tastenbefehle

1.41 Manuelle Definition

Klicken Sie das Eingabefeld an und geben Sie die Beschreibung ↩
des Tasten-
befehls ein. Yak überprüft diese Eingabe und aktualisiert die Blättersymbo-
le entsprechend.

HINWEIS: 'NewPrefs'-, 'PointerPos'-, 'Timer'- und 'Event'-Klassen werden im
Moment noch nicht von Yak unterstützt.

WARNUNG: Die commodities.library ist nicht ganz fehlerfrei:

- eine falsche Beschreibung ist z.B.
  'NumericPad a'

- folgende Beschreibung funktioniert nicht:
  'NumericPad [' und 'NumericPad]' (mit deutscher Tastatur)

Siehe auch:
Mehr über Tastenbefehle

1.42 Tastenbefehls-Aktionen

Die vielen möglichen Aktionen sind:

DOS-Befehl

Fenster schließen

Fenster zip

Fenster minimal

Fenster maximal

Fenster-Zyklus

Palette öffnen

Schirm nach vorn

Schirm nach hinten

Workbench aktivieren

Schirm zentrieren

Bildschirmschoner

Text einfügen

Datum einfügen

Yak-Einsteller

Fenster-Rückzyklus

Standard-Öffentl. Schirm setzen

Menü-Tastenkürzel

1.43 DOS-Befehl

Argument: Der auszuführende Befehl

Diese Funktion führt den im "Argument"-Texteingabe-Feld angegebenen DOS-Befehl aus. Beachten Sie, daß Sie das "Optionen"-Blättersymbol benutzen können, um dabei automatisch die Workbench bzw. den standardmäßigen öffentlichen Schirm nach vorne zu holen (das bietet sich vor allem bei Programmen an, die ein Fenster öffnen).

Der Befehl wird asynchron ausgeführt, d.h. Sie brauchen dem Befehl kein 'Run' voranzustellen. Falls der Befehl eine Ausgabe erzeugt (oder Eingabe erwartet), wird automatisch ein Konsolen-Fenster geöffnet. Sie können natürlich die Ein-/Ausgabe umleiten (genau wie in der Shell).

Beispiele für solche Tastenbefehle:

* Öffnen einer Shell (= Befehlszeilenschnittstelle :-)
  Üblicherweise wird dies mit dem Tastenbefehl "lcommand esc" verbunden; bei mir wird folgender Befehl aufgerufen:

"NewShell CON:79/177/582/78/AmigaShell/CLOSE/ALT2/58/660/197"

  Beachten Sie das 'ALT'-Merkmal in der Fenster-Beschreibung, das nur recht spärlich dokumentiert ist (sprich: gar nicht erwähnt). Ich benutze übrigens zwei Tastenbefehle: einen, um eine normale Shell zu starten, und einen zweiten für eine CShell.

* Unbenutzten Speicher freigeben:

  SAS/C benutzt "shared libraries", die häufig wertvollen Chip-Speicher verbrauchen. Deshalb benutze ich einen Tastenbefehl, der den Befehl "avail >nil: flush" zum Freigeben dieses Speichers aufruft.

* Inhalt einer (in DF0:) eingelegten Diskette auflisten:

  Setzen Sie den Tastenbefehl auf "diskinserted" und das "Argument"-Ein-

gabefeld auf "Dir DF0:".

Siehe auch:
Probleme

Mehr über Tastenbefehle

1.44 Fenster schließen

Argument: keins

"Fenster schließen" schließt das zur Zeit aktive Fenster (äquivalent zum Anklicken des Schließ-Symbols).

1.45 Fenster zip

Argument: keins

'Zipt' das zur Zeit aktive Fenster (äquivalent zum Anklicken des Zoom-Symbols).

'Zippen' ist neudeutsch und bedeutet hier soviel wie Wechseln zwischen zwei Fenstergrößen :)

1.46 Fenster minimal

Argument: keins

"Fenster minimal" macht das zur Zeit aktive Fenster so klein wie möglich.

1.47 Fenster maximal

Argument: keins

"Fenster maximal" macht das zur Zeit aktive Fenster so groß wie möglich.

1.48 Fenster-Zyklus

Argument: keins

"Fenster-Zyklus" holt das hinterste Fenster nach vorn. Nützlich, um an tief "vergrabene" Fenster heranzukommen.

1.49 Fenster-Rückzyklus

Argument: keins

"Fenster-Rückzyklus" arbeitet wie "Fenster-Zyklus", nur rückwärts: Das vorderste Fenster wird in den Hintergrund gebracht.

1.50 Palette öffnen

Argument: keins

Öffnet eine Palette auf dem vordersten Schirm. Die Palette wird asynchron gestartet, und Sie können beliebig viele öffnen (bei ausreichend Speicher). Allerdings kann Yak nicht beendet werden, solange noch Paletten geöffnet sind. Das "Optionen"-Blättersymbol kann hierbei benutzt werden (um z.B. die Palette ausdrücklich auf der Workbench zu öffnen).

ACHTUNG: Auf Ihrem System muß reqtools.library installiert sein, damit "Palette öffnen" funktioniert.

WARNUNG: Schließen Sie immer erst das Paletten-Fenster, bevor der Schirm, auf dem sich die Palette befindet, geschlossen wird. Andernfalls würden Sie im harmlosesten Fall einen leeren, geöffneten Schirm behalten; im schlimmsten Fall könnte das System abstürzen.

1.51 Schirm nach vorn

Argument: keins

"Schirm nach vorn" holt den hintersten Schirm nach vorn.

Siehe auch:
Schirm aktivieren

1.52 Schirm nach hinten

Argument: keins

"Schirm nach hinten" bringt den vordersten Schirm nach hinten.

Siehe auch:
Schirm aktivieren

1.53 Workbench aktivieren

Argument: keins

Aktiviert ein Workbenchfenster (und holt gegebenenfalls den Workbench-Schirm nach vorn). So kommen Sie an die Workbench-Menüs, ohne vorher ein Workbenchfenster zum Aktivieren finden zu müssen (nützlich, wenn Sie z.B. ein Shell-Fenster benutzen, das den gesamten Schirm ausfüllt).

1.54 Schirm zentrieren

Argument: keins

Zentriert den vordersten Schirm horizontal.

1.55 Bildschirmschoner

Argument: keins

Schaltet den (eingebauten) Bildschirmschoner sofort ein.

Siehe auch:
Bildschirm-Abschaltzeit

1.56 Text einfügen

Argument: der einzufügende Text

Dieser Tastenbefehl fügt die im Argument angegebene Zeichenfolge in den Eingabestrom ein. Diese Zeichenkette wird vorher wie folgt bearbeitet:

Steuerzeichen wird umgewandelt in:

\n	Zeilenvorschub
\r	Zeilenvorschub
\t	Tabulator
\	das '\'-Zeichen selbst
<Tastenbef.>	der betreffende Tastenbefehl
\<	das '<'-Zeichen selbst

Wegen dieser Vorbearbeitung kann "Text einfügen" eine Menge nützlicher Aufgaben erfüllen. Z.B. benutze ich einen Tastenbefehl zum Einfügen meines Namens und des Datums. Dabei sieht das Argument so aus (ohne '"'):

"Martin W Scott, <lcommand d>"

In diesem Fall ist "lcommand d" ein weiterer Yak-Tastenbefehl, der bei mir die Aktion "Datum einfügen" bewirkt. Durch entsprechend kompliziertere Zeichenfolgen können Sie einfache Makros für andere Programme erzeugen.

VORSICHT: Solche eingefügten Tastenbefehle sind zwar sehr nützlich, sollten

aber nur mit äußerster Vorsicht verwendet werden. Vermeiden Sie insbesondere rekursive Definitionen wie z.B.

```
f1 insert text  "<f2>"
f2 insert text  "<f1>"
```

Wenn Sie nun f1 oder f2 drücken würden, bedeutete das eine Endlosschleife. Falls Sie leichtsinnigerweise so etwas tun, starten Sie am besten das Commodities-Exchange-Programm und deaktivieren Yak. Dann klicken Sie auf das "Anzeige sichtbar"-Symbol von Exchange und löschen/ändern den/die gefährlichen Tastenbefehl(e).

Sie sollten weiterhin beachten, daß Zeichenfolgen, die Tastenbefehle aufrufen (wie im "Datum einfügen"-Beispiel oben), unter Umständen nicht genau so arbeiten, wie Sie es vielleicht erwarten würden. Nehmen wir an, die "Argument"-Zeichenfolge ist "<lcommand d>\n". Tatsächlich erzeugt dies einen Zeilenvorschub und DANN das Datum, da zu dem Zeitpunkt, bei dem Yak den "lcommand d"-Tastenbefehl erhält, der Zeilenvorschub schon längst den Input Handler passiert hat und zum aktiven Fenster gesendet worden ist.

1.57 Datum einfügen

Argument: Zeichenfolge für Datums-Format

Fügt das Datum in den Lese-Strom ein (und wird damit zur Eingabe in das zur Zeit aktive Fenster). Wenn Sie AmigaDOS 2.1 oder höher laufen haben, können Sie das Format des einzufügenden Datums an Ihre Wünsche anpassen. Dieses Format wird im "Argument"-Eingabefeld angegeben. Siehe
Format des Datums
.
Wenn Sie zu den weniger Glücklichen gehören, die mit AmigaDOS 2.0 arbeiten, ist das Datum im Standard-"TT-MMM-JJ"-Format.

Beispiel: Das Format "%e. %B %Y" erzeugt z.B. "30. Oktober 1993".

1.58 Yak-Einsteller

Argument: keins

Zeigt Yaks Einsteller-Fenster. Wenn Ihr vorderster Schirm öffentlich ist, wird das Fenster auf diesem Schirm geöffnet, andernfalls wird Ihr normaler öffentliche Schirm nach vorne geholt, auf dem Yak dann das Fenster öffnet. Dieselbe Funktion wird auch vom CX_POPKEY-Tastenbefehl ausgeführt.

1.59 Standard-Öffentl. Schirm setzen

Argument: keins

Der zur Zeit aktive Schirm ist künftig der öffentliche Schirm der standardmäßig verwendet wird. Natürlich muß dieser Schirm ein öffenlicher Schirm

sein.

1.60 Menü-Tastenkürzel

Argument: Menü, Menüpunkt und optional Untermenüpunkt

Schickt ein Eingabeereignis zum Menü des momentan aktivierten Fensters, das eine Menü-Auswahl durch den Benutzer simuliert (falls der betreffende Menüpunkt existert). Das kann dann hilfreich sein, wenn Sie zusätzliche Tastenkürzel für Anwendungen haben möchten, die keine oder zuwenige besitzen.

Das Argument gibt die Nummer des Menüs, des Menüpunkts und optional des Menüunterpunkts an. Alle Nummern müssen durch Leerzeichen voneinander getrennt werden und beginnen mit 0 -- entsprechend den Regeln von Intuition (d.h., das erste Menü hat die Nummer 0 und nicht 1).

Beachten Sie, daß eine Trennlinie im Menü auch als Menüpunkt zählt. Falls also im Menü oberhalb des Menüpunkts solche Trennlinien vorhanden sind, müssen Sie diese mitzählen.

Sie brauchen aber nichts zu befürchten, wenn Sie sich bei der Angabe des Arguments irren, da Yak alles Nötige überprüft, bevor ein Menü-Ereignis simuliert wird, um die Menübehandlung Ihrer Anwendung nicht durcheinander zu bringen.

Hier einige Beispiele für Argumente:

Nehmen wir an, Sie haben beschlossen, daß die Workbench nicht genug Tastenkürzel besitzt.

Für den "letzte Meldung anzeigen"-Menüpunkt im "Workbench"-Menü: "0 4"

Für den "nur Dateien mit Piktogr."-Untermenüpunkt des "Inhalt anzeigen"-Untermenüs im "Fenster"-Menü: "1 7 0"

1.61 AmigaDOS-Muster

AmigaDOS-Muster werden bei Yak verwendet, um eine Eigenschaft ↩
von Yak nur
auf eine angegebene Liste von Schirmen/Fenstern wirken zu lassen bzw. diese Liste von der Eigenschaft auszuschließen. Diese Musterangaben können bei der Kompatibilität mit anderen Programmen hilfreich sein.

Die Musterprüfung unterscheidet zwischen Groß- und Kleinbuchstaben: "Amiga" ist etwas anderes als "AMIGA". Folgende Standard-AmigaDOS-Muster können Sie benutzen:

```
? Paßt auf ein einzelnes Zeichen
# Paßt auf den folgenden Ausdruck 0 oder mehrmals
(ab|cd) Paßt auf einen der durch '|' getrennten Einträge
~ Negiert den folgenden Ausdruck. Es paßt dann auf alle
  Zeichenfolgen, die NICHT mit diesem Ausdruck zusammenpassen
```

```
    (z.B. paßt ~(foo) auf alle Zeichenfolgen, die nicht genau
    gleich "foo" sind.
  [abc] Zeichenklasse: paßt auf eines der Zeichen in der Klasse
  a-z Zeichenbereich (nur innerhalb von Zeichenklassen)
  % Paßt immer auf kein Zeichen (nützlich bei "(foo|bar|%)").
  * Synonym für "#?", steht normalerweise allerdings nicht zur
    Verfügung. Benutzbar, wenn
                AmigaDOS *
                 angeschaltet ist.
```

Wenn Sie noch nicht mit Mustern vertraut sind, mag das alles für Sie recht entmutigend sein. Schlagen Sie in Ihrem System-Handbuch nach, um weitere Details zu erfahren. Normalerweise werden Sie eine von zwei grundlegenden Funktionsweisen benutzen wollen: Entweder eine endliche Liste der Namen, auf die eine Eigenschaft von Yak angewendet werden darf, oder eine endliche Liste derjenigen Namen, die davon ausgeschlossen werden sollen. Um eine Eigenschaft für alle Objekte zu ERLAUBEN (egal ob Schirm oder Fenster, je nachdem), benutzen Sie das "#?"-Muster (das paßt auf alles). Soll die Eigenschaft nur auf N Objekte mit den Namen "Name1" bis "NameN" angewendet werden, benutzen Sie

```
  (Name1|Name2| ... |NameN)
```

(Die "..." durch die übrigen Namen ersetzen!)
Um die Eigenschaft für diese Objekte zu unterdrücken, schreiben Sie ein ~ (Tilde) davor:

```
  ~(Name1|Name2| ... |NameN)
```

Ein Beispiel: Ich möchte nicht, daß "Autom. nach vorn" das Workbench-Fenster nach vorne holt, genauso wenig wie eines meiner Protext-Fenster, die alle die Zeichenfolge "Arnor" im Titel enthalten. Also schließe ich diese Fenster durch folgendes Muster aus:

```
  ~(Workbench|#?Arnor#?)
```

Beachten Sie, daß der zweite ’Name’ in Wirklichkeit ein Muster ist, daß auf alle Titel paßt, die den text "Arnor" enthalten.

Ein weiteres Beispiel: Ich möchte nicht, daß der "Directory Opus"-Schirm automatisch aktiviert wird. Da der Titel des Schirms nicht sichtbar ist, muß ich ARTM oder Xoper (o.ä.) benutzen, um den Namen des Schirms zu ermitteln. Ich finde heraus, daß der Schirm "DOPUS.1" heißt. Ich stelle mir vor, daß der Name entsprechend "DOPUS.2" ist, falls ich eine zweite Kopie laufen lasse, und beschließe daher, alle DOpus-Schirme wie folgt auszuschließen:

```
  ~(DOPUS#?)
```

Anmerkung 1: Um den Workbench-Schirm anzugeben, muß man einen Trick benutzen, da sich der Titel laufend ändert, je nachdem, welches Fenster aktiv ist. In den meisten Fällen ist ein Muster wie z.B. "#?Workbench#?" ausreichend; Allerdings setzen einige Applikationen den Workbench-Titel auf eine Zeichenfolge, die dieses Programm näher beschreibt.

Anmerkung 2: Schirme oder Fenster, die keinen Namen besitzen (NULL im be-

treffenden Eintrag) bleiben von den Mustern immer unberührt.

1.62 Format des Datums

Bei "Datum einfügen"-Tastenbefehlen müssen Sie eine Zeichenkette für das Format des Datums im Stil der locale.library angeben (außerdem brauchen Sie mindestens AmigaDOS 2.1). Unter locale.library können Sie folgende Optionen zur Formatierung benutzen:

```
  %a - abgekürzter Wochentag
  %A - Wochentag
  %b - abgekürzter Monatsname
  %B - Monatsname
  %c - wie "%a %b %d %H:%M:%S %Y"
  %C - wie "%a %b %e %T %Z %Y"
  %d - Tagesnummer (im aktuellen Monat) ggf. mit führender 0
  %D - wie "%m/%d/%y"
  %e - Tagesnummer (im akt. Monat) ggf. mit führendem Leerzeichen
  %h - abgekürzter Monatsname
  %H - Uhrzeit im 24-Stunden-Modus ggf. mit führender 0
  %I - Uhrzeit im 12-Stunden-Modus ggf. mit führender 0
  %j - Julianisches Datum
  %m - Monatsnummer ggf. mit führender 0
  %M - Minuten (in der aktuellen Stunde) ggf. mit führender 0
  %n - Zeilenvorschub
  %p - "AM"- oder "PM"-Zeichenfolgen
  %q - Uhrzeit im 24-Stunden-Modus
  %Q - Uhrzeit im 12-Stunden-Modus
  %r - wie "%I:%M:%S %p"
  %R - wie "%H:%M"
  %S - Sekunden (in der aktuellen Minute)
  %t - Tabulator-Zeichen
  %T - wie "%H:%M:%S"
  %U - Wochennummer, erster Tag der Woche = Sonntag
  %w - Wochentagsnummer
  %W - Wochennummer, erster Tag der Woche = Montag
  %x - wie "%m/%d/%y"
  %X - wie "%H:%M:%S"
  %y - Jahr, zweistellig, ggf. mit führender 0
  %Y - Jahr, vierstellig, ggf. mit führender 0
```

Diese Liste sollte wohl alle Ihre Bedürfnisse befriedigen; Sie können zusätzlich Ihren eigenen Text beliebig in die Formatierungs-Zeichenfolge einfügen. Einige Beispiele:

```
  "Die Zeit: %X"    erzeugt (z.B.)  "Die Zeit: 20:44:16"
  "Einen schönen %A!" erzeugt (z.B.)  "Einen schönen Montag!"
```

Wenn Sie mehr Details wissen wollen, schlagen Sie am besten in den AutoDocs zur locale.library nach (falls vorhanden).

1.63 Probleme

Ich weiß z.Zt. von einigen Problemen: Eine Shell, die durch einen "DOS-Befehl"-Tastenbefehl erzeugt wird, hat nicht die Stapelgröße und das aktuelle Verzeichnis, die beim Systemstart in der Startup-Sequence gesetzt werden. Der Pfad bleibt aber ERHALTEN. Ihre Shell-Startup-Datei sollte das aktuelle Verzeichnis und die benötigte Stapelgröße einstellen. Wenn nicht anders angegeben, ist das aktuelle Verzeichnis des gestarteten Prozesses die Systemstart-Diskette (SYS:).

Wenn Sie nicht mit Yaks "Zeiger-Ausblenden"-Funktion zufrieden sind, können Sie es mal mit dem "MouseBlank"-Commodity (WB 3.0) von Commodore versuchen, das in jedem Fall korrekt funktionieren sollte.

HINWEIS FÜR OneKeyII-BENUTZER: Deaktivieren Sie OneKeyII, wenn Sie einen Tastenbefehl eingeben.

HINWEIS FÜR AMOS-BENUTZER: Ich hasse AMOS (soweit zu meiner Meinung), teilweise auch, weil es so system-unfreundlich ist. Es klaut den gesamten Eingabestrom, so daß Maus-Ausblender u.ä. (wie z.B. in Yak) anspringen, da sie glauben, es gebe keine Eingabe. Außerdem wird ein ausgeblendeter Mauszeiger nicht wieder angezeigt, da es ja auch keine Mausbewegungen gibt. Da Yaks Methode der Maus-Ausblendung ziemlich schlecht ist, können Probleme auftauchen (die Maus verschwindet (ähm, natürlich der MausZEIGER:) und erscheint nicht wieder). Zwei mögliche Lösungen:

1) Benutzen Sie "Zeiger ausblenden: Durch Copper"
2) Setzen Sie die Maus-Abschaltzeit auf null. Durch Tastendruck verschwindet der Zeiger immer noch, nicht aber nach einem festen Zeitintervall.

Dann sollten die AMOS-Probleme mit dem verschwundenen Mauszeiger beseitigt sein.

1.64 Programm-Geschichte

(* = neue Eigenschaft)

v1.60 * Neuer 'MMB -> Shift links' - Schalter

- Fehler im Installations-Skript bei der Sprachauswahl beseitigt

- 'Zeiger ausblenden' funktioniert jetzt mit einer Picasso II - Grafikkarte besser, aber es gibt immer noch Probleme. Könnte mir vielleicht jemand mal so eine Karte geben, damit ich das Problem beheben kann? Das fände ich wirklich gut :-)

- Fehler beim Initialisieren des 'Abschaltzeiten'-Symbols im 'Schoner'-Fenster behoben

v1.59 - Enforcer-Hit beim Anklicken von "Verschiedenes" beseitigt

- Enforcer-Hit beim Öffnen einer Palette beseitigt

- Fehler beseitigt: 'Klick nach hinten'-Schalter war wirkungslos

- Fehler beseitigt: 'Schirm aktivieren'-Schalter war wirkungslos

- Fehler beseitigt: 'Kein Laufwerk-Klicken' und 'AmigaDOS *' waren gerade falsch herum

- Fehler beseitigt: 'Zeiger ausblenden' funktionierte nicht richtig

- Fehler beseitigt: automatische Tastenbefehls-Definition benutzte V38-Schlüsselwörter und war also unter V37 nicht benutzbar

- Fehler beseitigt: 'Klick nach vorn/hinten' haben das System eingefroren

* Yak erinnert sich jetzt daran, welches Fenster beim Schirm-Zyklus zuletzt auf einem Schirm aktiviert war

* Neuer Tastenbefehl "Menü-Tastenkürzel" um einen Menüpunkt anzuwählen

v1.58 - Fehler beim 'Fenster-Rückzyklus'-Tastenbefehl behoben

- Fehler im GUI für 'Auto-Aktiv-Verzögerung' und 'Lautstärke' behoben. Falsche Werte führten zum Absturz

- Fehler beim Zeiger-Ausblenden behoben, der auftrat, wenn Yak mit CLIExchange deaktiviert wurde

- Fehler beim Beenden von Yak behoben. (Hatte sich in Version 1.57 eingeschlichen)

- Fehler in 'Klick nach hinten' behoben

* Wenn die Alt-Taste gedrückt ist, wird der Mauszeiger jetzt nicht mehr ausgeblendet, so daß die Menübedienung per Tastatur leichter wird.

* Yak kann jetzt sowohl mit SAS C 6.51 als auch mit DICE kompiliert werden

* Schwarzer Rand für Schirme (nur ab Kickstart 3.0)

* Die grafische Benutzerschnittstelle hat ein neues Design von Reza Elghazi bekommen

* Der Paletten-Titel ist jetzt lokalisiert

* Muster für Schirme benutzen jetzt den Standard-Titel und nicht den aktuellen. Das vereinfacht das Erstellen der Muster für z.B. die Workbench oder den BrowserII-Schirm erheblich

* Neuer Tastenbefehl zum Setzen des standardmäßigen Öffentlichen Schirm

* Neue Methode, automatisch einen Tastenbefehl zu definieren, indem dieser einfach gedrückt wird. Von Philippe Bastiani.

* Yak öffnet jetzt nur noch eine Palette pro Schirm

* Neu ist ein dänischer Katalog, Installer-Text und Anleitung

* Es werden keine Overlays mehr benutzt

v1.57 - Das Einsteller-Fenster merkt sich beim Schließen die Position der 2 anderen Fenster

- Jetzt mit SAS C 6.50 compiliert, Programmgröße um 2 K verkleinert!

v1.56 - Das Einsteller-Fenster öffnet sich jetzt immer genau unterhalb der Titelleiste des Bildschirms - unabhängig von der Zeichensatz-größe

* Neuer Schalter "MMB-Aktivierung"

- "Fenster schließen" funktioniert jetzt auch für Shell-Fenster

* Jetzt mit schwedischem und niederländischem Katalog

* Das Einsteller-Fenster öffnet sich jetzt auf vorderstem Schirm, falls öffentlich, ansonsten auf dem normalen öffentlichen Schirm

- Fehler behoben, der beim Ermitteln des Fensters unter der Maus auftrat, wenn mehrere Schirme gleichzeitig sichtbar sind. Ein Dankeschön an Pierre Carette, der diesen Fehler gefunden und berichtigt hat.

* Neues Verhalten von "Klick nach vorne": Auch Schirme können nach vorne geklickt werden, wenn "Schirm-Zyklus" angeschaltet ist

* Neues Fenster "Verschiedenes", um Integration von neuen Funktionen zu vereinfachen

* "Auto-Aktiv-Verzögerung" kann nun vom Benutzer eingestellt werden

* Zwei neue Piktogramme für 8-farbige MagicWB-Umgebung

* Jetzt deutsche und italienische AmigaGuide-Anleitung

* Jetzt Simulation von Qualifieren mittels MMB (eine Art erweitertes MMBShift oder MouseShift)

v1.55 - Habe einen dummen Fehler korrigiert, der Yak 1.54 nicht unter Workbench laufen ließ.

- Aussehen des Einsteller-Fensters mit schmalen, hohen Zeichensätzen verbessert.

- Installer-Script überschreibt ein existierendes Yak-Piktogramm nicht mehr, um geänderte Merkmale nicht zu löschen.

```
 - Aminet-Archiv geändert: Quelltext in gesondertem Archiv, andere
   Piktogramme entfernt.

 * Neuer Tastenbefehl "Fenster-Rückzyklus".

 * Italienische(r|s) Katalog und Installer-Script.


v1.54  * Neues Merkmal LANGUAGE, um andere Sprache als die in locale.prefs
   gespeicherte zu benutzen.

 * "Kein Laufwerk-Klicken" für Disketten-Laufwerke hinzugefügt.

 * Deutsche(r|s) Katalog und Installer-Script.

 - Yaks Größe stark verkleinert (um mehr als 10k!) durch Benutzung
   von CATCOMP_BLOCK statt CATCOMP_ARRAY.


v1.53  - Habe einen dummen Fehler korrigiert, der unnötig ein Dialogfenster
   bei der Benutzung einer anderen Sprache als Französisch öffnete.

 * Jetzt Fehlermeldungen lokalisiert (französischer Katalog auf-
   datiert).

 - Fehler im Installer-Script bei der Installation des französischen
   Katalogs beseitigt.


v1.52  * Yak ist lokalisiert, französischer Katalog.

 * Jetzt mit Installations-Script.

 - Neue Versions-Numerierung für RCS-Archivierung.

 - Kleiner Fehler in NewMenus-Anzeige unter 3.0 behoben.

 - Kosmetische Änderungen an Anleitung und Schnittstelle.

 - Jetzt mit SAS C 6.3 kompiliert.


v1.5a  - Benötigter Stapel größer (Bedarf könnte über die bisherige Größe
   hinausgeschossen sein).

 - NOICON-Merkmal entfernt, statt dessen APPICON-Merkmal.

 - Am Tastenbefehls-System weiter herumfrisiert.


v1.5 * Erweiterbares Tastenbefehls-System hinzugefügt.

 * Aufpolierte grafische Benutzerschnittstelle.

 * Anleitung nun im AmigaGuide-Format (wirklich???)

 * "Datum einfügen" funktioniert jetzt auch unter 2.0
```

```
  - Neues Voreinsteller-Datei-Format, zusätzlich Tastenbefehls-Datei.
    Alte Voreinsteller-Dateien können nicht geladen werden.


 <frühere Versionen (v1.4 und darunter) weggelassen>
```

1.65 Yak compilieren

Yak ist zum Kompilieren unter SAS/C, ab v1.4a unter SAS/C Version 6.2. Der Code passiert den Compiler mit ANSI-Überprüfung ohne Probleme, so daß Benutzer anderer ANSI-Compiler wenig Probleme beim Wiederkompilieren haben--- lediglich die SAS-spezifischen Schlüsselwörter (wie __saveds) müssen u.U. geändert werden.

Yak benutzt "Overlays", das kann aber durch Neukompilation geändert werden, wenn Sie die Definition USE_OVERLAYS in SCOPTIONS entfernen.

Ich habe GTB 2.0b benutzt, um die Benutzerschnittstelle zu erzeugen. Der erzeugte Code (popup.c, popup.h) muß nur minimal verändert werden, insbesondere um NewLookMenus unter WB 3.0 und darüber zu benutzen; Änderungen sind durch Kommentare kenntlich gemacht, die mit vier Sternchen anfangen, z.B. /**** ADDED */.

1.66 Yak development team

Die Yak-Entwicklung ist Teamwork.

Programmierung

Gael Marziou
Philippe Bastiani
Reza Elghazi (GUI)

Übersetzungen, Vorschläge und beta-testen

```
Alex Galassi     : Dok, Installer-Script und Katalog (italienisch).
Ingolf Koch      : Dok, Installer-Script und Katalog (deutsch).
Christian Høj    : Dok, Installer-Script und Katalog (dänisch).
Patrick van Beem : Installer-Script und Katalog (niederländisch).
Johan Billing    : Katalog (schwedisch).
Peter Eriksson   : Installer-Script (schwedisch).
```

1.67 Danksagungen

Yak ist vollständig in C geschrieben und mit SAS/C 6.3 kompiliert. Danke, Reza Elghazi, für Deine Hilfe beim Übergang zu 6.2.

Die Graphische Benutzer-Schnittstelle des Einsteller-Fensters wurde mit dem exzellenten Programm GadToolsBox v2.0b von Jaba Development erstellt.

Yak benutzt die reqtools.library, Copyright Nico François. Herzlichen Dank auch an Steve Koren für SKsh, Matt Dillon für DMouse (er hat mir viele "Wie mache ich"-Fragen beantwortet) und an Kai Iske für KCommodity, von dem ich mir das Klickgeräusch bei Tastendruck "geborgt" habe.
Yak benutzt außerdem WB2CLI, ein sehr nützliches Link-Modul von Mike Sinz.

Danke auch an Heinz Wrobel für seine wunderbare Amiga-Portierung von RCS, an Pierre Carette und Sylvain Rougier für BrowserII, Martin Korndörfer für sein MagicMenu und an meine Freunde aus der französischen Mailing-Liste, die mir beim Lokalisieren von Yak ins Französische geholfen haben.

Der Teil "Mehr über Tastenbefehle" stammt aus der deutschen Anleitung zu ToolManager 2.1 von Stefan Becker. Danke, Stefan, für die Erlaubnis, das zu übernehmen.

Danke an Stefan Sticht für sein public domain MouseBlanker-Commodity - aus diesem Programm habe ich die Ausblendung der Maus via Copper gemopst.

Und natürlich ein herzliches Dankeschön an alle diejenigen, die mir Vorschläge und Fehlerberichte bezüglich Yak geschrieben haben.

Danke auch an Martin Huttenloher, der die Piktogramme für Schubladen und AmigaGuide-Dateien entworfen hat. Ich habe sie aus seinem hervorragenden MagicWB-Paket entnommen.

Danke an Osma Ahvenlampi, der das schöne Piktogramm für Yak im MagicWB-Stil gemalt hat.

Danke an Nicola Salmoria, der mir erklärt hat, wie "Schwarzer Rand" programmiert wird.

Und, last but not least: Danke an Martin Scott, der Yak erschaffen hat.

1.68 Adresse des Autors

Bitte schicken Sie Kommentare, Vorschläge, Fehlerberichte, Lobeshymnen :), Geld usw. an:

Gaël Marziou
Cidex 103
38920 CROLLES
FRANKREICH

ODER EMAIL (habe ich lieber): Gael_Marziou@grenoble.hp.com

1.69 Mehr über Tastenbefehle

Dieses Kapitel habe ich mit freundlicher Genehmigung von Stefan Becker aus der Anleitung zu seinem Programm Toolmanager V2.1 übernommen.

Wie man einen Tastenbefehl definiert

Diese Kapitel beschreibt wie man einen Tastenbefehl als einen Input Description String definiert, der dann von Commodities ausgewertet werden kann. Jedes Mal, wenn ein Tastenbefehl ausgeführt wird, erzeugt Commodities eine Ereignis, das dann von ToolManager dazu benutzt wird Programmobjekte zu aktivieren oder Dock-Objekte umzuschalten. Ein Description String hat die folgende Syntax:

```
[<Klasse>] {[-][<Qualifier>]} [-][upstroke] [<Tastenkode>]
```

Alle Befehlsworte können groß oder klein geschrieben werden.

Klasse beschreibt die InputEvent-Klasse. Dieser Parameter ist optional und falls er weggelassen wird, dann wird die Vorgabe rawkey benutzt. Siehe

InputEvent-Klassen

.

Qualifier sind "Signale", die gesetzt oder nicht gesetzt sein müssen zu dem Zeitpunkt, an dem der Tastenbefehl ausgeführt wird, sonst wird kein Ereignis erzeugt. Für jeden Qualifier, der gesetzt sein soll, müssen Sie das Befehlswort angeben. Alle anderen Qualifier müssen dann nicht gesetzt sein. Falls Sie einen Qualifier ignorieren wollen, dann setzen sie ein - vor sein Befehlswort. Siehe

Qualifier

.

Normalerweise wird ein Ereignis erzeugt, wenn eine Taste gedrückt wird. Falls das Ereignis generiert werden soll wenn die Taste losgelassen wird, dann müssen Sie das Befehlswort upstroke angeben. Wenn sowohl beim Drücken als auch beim Loslassen der Taste ein Ereignis erzeugt werden soll, dann müssen sie das Befehlswort -upstroke angeben.

Der Tastenkode ist abhängig von der InputEvent-Klasse. Siehe

Tastenkodes

.

InputEvent-Klassen

Qualifier

Tastenkodes

Beispiele für Tastenbefehle

Achtung: Wählen Sie ihre Tastenbefehle sorgfältig, denn Commodities hat eine hohe Priorität in der InputEvent-Handlerkette,

d.h. vorgegebene Definitionen werden übergangen.

1.70 InputEvent-Klassen

```
                InputEvent-Klassen
==================

   Commodities unterstützt die meisten der InputEvent-Klassen, die von
dem input.device erzeugt werden. Diese Sektion beschreibt die Klassen,
die nützlich für ToolManager sind.

rawkey
     Dies ist die vorgegebene Klasse. Sie beschreibt alle Ereignisse,
     die durch die Tastatur erzeugt werden können. Zum Beispiel
     erzeugt rawkey a oder a jedesmal ein Ereignis, wenn die Taste "a"
     gedrückt wird. Sie müssen einen Tastenkode für diese Klasse
     angeben. Siehe
                rawkey
                .

rawmouse
     Diese Klasse beschreibt alle Ereignisse, die durch die Maus
     erzeugt werden können. Sie müssen einen Tastenkode für diese
     Klasse angeben. Siehe
                rawmouse
                .

diskinserted
     Ereignisse dieser Klasse werden generiert, wenn eine Diskette in
     ein Laufwerk gelegt wird. Diese Klasse besitzt keine Tastenkodes.

diskremoved
     Ereignisse dieser Klasse werden generiert, wenn eine Diskette aus
     einem Laufwerk genommen wird. Diese Klasse besitzt keine
     Tastenkodes.
```

1.71 Qualifier

```
Qualifier
=========

   Einige Befehlsworte wurden erst bei Commodities V38 eingeführt.
Diese sind mit einem * markiert.

lshift, left_shift *
     Linke Shift-Taste.

rshift, right_shift *
```

```
     Rechte Shift-Taste.

shift
     Irgendeine Shift-Taste.

capslock, caps_lock *
     Caps-Lock-Taste.

caps
     Irgendeine Shift-Taste oder die Caps-Lock-Taste.

control, ctrl *
     Control-Taste.

lalt, left_alt *
     Linke Alt-Taste.

ralt, right_alt *
     Rechte Alt-Taste.

alt
     Irgendeine Alt-Taste.

lcommand, lamiga *, left_amiga *, left_command *
     Linke Amiga-/Kommando-Taste.

rcommand, ramiga *, right_amiga *, right_command *
     Rechte Amiga-/Kommando-Taste.

numericpad, numpad *, num_pad *, numeric_pad *
     Dieses Befehlswort muß angegeben werden, wenn eine Taste von der
     Zehnertastatur benutzt wird.

leftbutton, lbutton *, left_button *
     Linke Maustaste. Siehe unten.

midbutton, mbutton *, middlebutton *, middle_button *
     Mittlere Maustaste. Siehe unten.

rbutton, rightbutton *, right_button *
     Rechte Maustaste. Siehe unten.

repeat
     Dieser Qualifier ist gesetzt, wenn die Tastenwiederholung aktiv
     ist. Dies ist nur sinnvoll für die InputEvent-Klasse rawkey.

   Achtung: Commodities V37 hat einen Fehler, der die Benutzung von
leftbutton, midbutton und rbutton als Qualifier verhindert. Dieser
Fehler wurde in V38 behoben.
```

1.72 Tastenkodes

```
                Tastenkodes
===========
```

Jede InputEvent-Klasse besitzt ihre eigenen Tastenkodes:

rawkey

rawmouse

1.73 rawkey Tastenkodes

Tastenkodes für die InputEvent-Klasse 'rawkey'

Einige Befehlsworte wurden erst bei Commodities V38 eingeführt. Diese sind mit einem * markiert.

a-z, 0-9, ...
ASCII-Zeichen.

f1, f2, ..., f10, f11 *, f12 *
Funktionstasten.

up, cursor_up *, down, cursor_down *
left, cursor_left *, right, cursor_right *
Cursor-Tasten.

esc, escape *, backspace, del, help
tab, comma, return, space, spacebar *
Spezial-Tasten.

enter, insert *, delete *
page_up *, page_down *, home *, end *
Tasten der Zehnertastatur. Diese Tastenkodes müssen mit dem Qualifier numericpad benutzt werden!

1.74 rawmouse Tastenkodes

Tastenkodes für die InputEvent-Klasse 'rawmouse'

Diese Befehlsworte wurden erst bei Commodities V38 eingeführt. Sie sind nicht verfügbar in V37.

mouse_leftpress
Drücke die linke Maustaste.

mouse_middlepress
Drücke die mittlere Maustaste.

```
mouse_rightpress
     Drücke die rechte Maustaste.

   Achtung: Um einen dieser Tastenkodes zu benutzen, müssen sie auch
das entsprechende Qualifier-Befehlswort angeben, z.B.

     rawmouse leftbutton mouse_leftpress
```

1.75 Beispiele für Tastenbefehle

```
Beispiele für Tastenbefehle
===========================

ralt t
     Rechte Alt-Taste festhalten und "t" drücken.

ralt lalt t
     Rechte und linke Alt-Taste festhalten und "t" drücken.

alt t
     Irgendeine Alt-Taste festhalten und "t" drücken.

rcommand f2
     Rechte Amiga-Taste festhalten und die zweite Funktionstaste
     drücken.

numericpad enter
     Enter-Taste auf der Zehnertastatur drücken.

rawmouse midbutton leftbutton mouse_leftpress
     Mittlere Maustaste festhalten und die linke Maustaste drücken.

diskinserted
     Eine Diskette in ein Laufwerk einlegen.
```

1.76 Lokalisation

```
                Wie Sie sehen, ist Yak jetzt lokalisiert.  Die vorliegende ↩
                   deutsche Version
(Installer, Katalog und Anleitung) wurde übersetzt von

  Ingolf Koch
  Wellenkampstraße 38
  32791 Lage, Lippe
  Tel. (+49) (0)5232 2700
  E-mail:   ingolf@mathematik.uni-bielefeld.de
  IRC nick: Balin

Bitte schicken Sie Anregungen, Kommentare, Verbesserungsvorschläge ..., die
die deutsche Übersetzung betreffen, an diese Adresse  (nach Möglichkeit per
```

E-mail).

Zur Zeit gibt es noch sechs weitere Sprachen für Yak: Englisch (die eingebaute Sprache), Französisch, Italienisch, Schwedisch, Niederländisch und Dänisch.

Wenn Sie Yak in eine andere Sprache übersetzen wollen, lesen Sie am besten das Kapitel "Localization" in der englischen Anleitung durch. Außerdem sollten Sie mit

Gaël Marziou

Kontakt aufnehmen.