Yak

COLLABORATORS			
	TITLE : Yak		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 2, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Yak

1.1 Yak v1.57 Documentation

```
                                        Yak Version 1.57
                              by Gaël Marziou
                              released on 21 December 1993.
```

TABLE OF CONTENTS

1.2 Copyright

1.3 Current users: Notez-bien

Yak's preference file format has changed again, with ↩ preferences being split into two files, "S:Yak.prefs" and "S:Yak.hotkeys". A Convert program is supplied to create the 1.5 preference files from a 1.3/1.4 preference file. There are two flaws in the conversion: you must reenter the PopCommand and DateFormat strings using the new Hotkey Definition window (see section Hotkeys for details on how to do this); the other flaw is that a hotkey will be created to "Show settings window", which is probably superfluous, given the discussion on CX_POPKEY below.

There are some important changes between v1.4 and 1.5 of Yak. Most importantly, the stack requirement has been increased to 4500 bytes, so do

remember to change the stack size in the icon you use to start Yak.

Secondly, the AppIcon is by default OFF, and should be explicitly enabled using the APPICON=TRUE tooltype. This tooltype replaces the NOICON tooltype.

Thirdly, the CX_POPKEY tooltype is used to determine the hotkey which opens Yak's settings window, and this key cannot be altered or overridden within the Hotkey definition window (unlike in previous versions); however, you can create other hotkeys which also open Yak's window.

Fourthly, Yak has been localized and supports now english as built-in language and french and german with catalogs provided. If you want to localize Yak into your own language, please contact me (see section
Contacting the author
and
Localization
.

Of course, there have been many other changes, and you should browse through this document to accustomise yourself with them.

1.4 Introduction

Yak stands for "Yet Another Kommodity" (never was any good at ↩
speling - see

Jokes
), and is a mouse/window manipulation program along the same lines ↩
as
DMouse, MightyMouse etc.

Why write another one? None of the others (and I've looked at almost all of them) were quite right for me. Yak has the following features:

- AutoPoint (sunmouse) that only activates when mouse stops
  can also specify which screens to include/exclude.
  (Compatible with popup-menu type programs)

- AutoPop windows (bring them to front) when they're
  auto-activated;

- KeyActivate windows (when key pressed);

- Activate windows when menu button pressed;

- Click windows to front or back; may also specify which
  screens/windows to include/exclude.

- Cycle screens with mouse;

- Mouse and Screen blanking (hotkey blanking too);

- Extensible hotkey system (like FKey's) with actions to:

```
    Execute a Dos Command;
    Insert text (with embedded hotkeys);
    Insert date (in custom format if locale present);
    Close/Zip/Shrink/Enlarge windows;
    Cycle screens and windows;
    Activate Workbench;
    Center frontmost screen;
    Blank display;
    Pop up a palette on front screen (needs reqtools);

  o KeyClick with adjustable volume;

  o No Click option (for drives).

  o Wildstar option (like StarBurst) lets you use '*' as wildcard.

  o Optional on-screen AppIcon to bring up preferences window.

  o Nice gadtools settings window;
```

Look familiar? It's a combination of the AutoPoint, ClickToFront, Blanker and IHelp/FKey standard commodities (on the Extras disk) with a hint of KCommodity and DMouse thrown in.

And it's quite small: it takes up about 29K on disk, and about the same when running. It achieves this memory efficiency by using overlays; the settings window code is only loaded when needed. That means that Yak uses more memory when you opens Yak window configuration (about 45K).

1.5 Jokes

These might be old, but they're good:

Q. What's the difference between a goldfish and a goat?
A. A goldfish mucks around in fountains!

And if you got that one, you'll definitely get this one...

Q. What's the difference between a magic wand and a truncheon (baton).
A. A magic wand is used for cunning stunts.

(I *always* say that last answer wrong)

1.6 Limitations

Yak only has a plain screen-blanker. Since the introduction of OS 2.0, there has been a profusion of fancy screen blankers, such as Spliner, ASwarm (various incarnations), FracBlank and more. I think most people have their own favourite fancy blanker, so I didn't include one in Yak.

There is no mouse-acceleration. The system default one (settable via the Input preferences) is more than adequate as far as I am concerned. If I

get enough requests, I'll add faster acceleration (so far, hardly any received).

1.7 Starting Yak

Yak is designed to be run from the sys:WBStartup drawer, but may ↩ be 'run'
from the Shell (not recommended, as it will use significantly more memory). All Yak settings are stored in two files, a prefs-file, "S:Yak.prefs", and a hotkey file, "S:Yak.hotkeys". The best thing to do is to use the installation script then when you first use Yak, configure it the way you like, then save that configuration for later invocations.

NB: Yak needs a stack size of 4500 bytes. You should set this in Yak's icon, or if starting it from the Shell, use the following commands:

```
stack 4500
run >nil: yak
```

The only tooltypes that Yak takes are the standard Workbench ones (such as DONOTWAIT), the standard Commodities ones (such as CX_POPUP), the AppIcon ones and the LANGUAGE one. Those tooltypes are:

ToolType	Category	Description	Default
CX_POPKEY	HOTKEY	Key to show settings window	RCommand Help
CX_PRIORITY	INTEGER	Priority of this commodity	0
CX_POPUP	BOOLEAN	Show settings window on startup	NO
APPICON	BOOLEAN	If TRUE, an AppIcon is made	FALSE
ICONNAME	STRING	Name of AppIcon	"Yak!"
ICONXPOS	INTEGER	x-coordinate of AppIcon	floating
ICONYPOS	INTEGER	y-coordinate of AppIcon	floating
LANGUAGE	STRING	Name of language to use	Not specified

You should also have the tooltype "DONOTWAIT" set if you want to start Yak from your SYS:WBStartup drawer.

The AppIcon facility is optional, and by default is off. Specify APPICON=TRUE if you want it. If you do, Yak puts an AppIcon onto the Workbench screen, and when it is double-clicked, the Settings Window is opened. The icon imagery is obtained from the icon Yak was started from, which allows you to customise the AppIcon to your colours/resolution simply by changing Yak's icon.

The CX_PRIORITY may be useful in enabling Yak and other commodities to work better together. See, for example, notes on the
RMB/MMB Activate
toggle
below.

The LANGUAGE tooltype has been added for people using Workbench in one language and Yak in another one. For example, french friends of mine are used to english workbench but want to use Yak in french so they just have to set

the LANGUAGE tooltype as this :

```
    LANGUAGE=français
```

Of course, those of you who want to use same language for both Yak and workbench don't have to worry about this tooltype.

1.8 Changing Yak's settings

All of Yak's settings may be modified by means of it's settings window.
Once Yak is started, you may bring up this window by the following means:

Pressing RCommand Help (i.e. the Right Amiga key and the Help key).
This key-sequence is configurable (via the CX_POPKEY tooltype);

Double-clicking Yak's AppIcon (if this feature is set);

Start Yak again (from Workbench tool icon);

Via the Commodities Exchange (on the Extras disk).

The window then opened contains many gadgets, grouped into classes. These classes are:

Toggles

Patterns

TimeOut

Miscellaneous

Hotkeys

Remember that you must press the RETURN, ENTER or TAB key once you have edited a string gadget, so that the change is registered. Simply clicking outside it will lose the changes. (The TAB key activates the next string gadget for text entry).

Also (and this applies particularly to the hotkey window), Intuition by default does not let you cut and paste between string gadgets, but if you want this, you should get "NewEdit" by Uwe Roehm, which adds clipboard copying and pasting (it's on a Fish disk and ftp sites).

Additionally, the two gadgets "Hide" and "Quit" perform the standard commodity operations respectively of hiding the settings window and terminating Yak. Clicking the window's close-gadget is equivalent to hiding the interface rather than ending Yak.

The settings window also has a menu with "Hide" and "Quit" items (which operate as the gadgets of the same name), plus items "Load" and "Save". All changes to Yak's settings will be lost unless you select the "Save"

item. If you wish to restore you're last-saved settings, you may use the "Load" item.

1.9 Toggles

The following toggles are available:

AutoPoint

AutoPopToFront

Key Activate

Click To Front

Click To Back

Cycle Screens

Screen Activate

MMB Activate

RMB Activate

Wild star

No Click

1.10 AutoPoint

Activate window under mouse. This behaves almost exactly like Commodore's AutoPoint commodity, in that it only activates a window when the mouse stops. AutoPoint is compatible with popup-menu type programs such as the excellent MagicMenu.

Note that the AutoPoint and AutoPopToFront functions only take place when NO qualifier (mouse or keyboard) is pressed. Not only does this avoid conflict with other programs, it provides a way of preventing activation/popping when it's not desired.

See also:

AutoPopToFront

AutoScreens

1.11 AutoPopToFront

Only operative when AutoPoint is set, this tells Yak to bring windows to the front as well as activating them. The exception is when the window under the mouse has a requester showing.

See also:

AutoPoint

AutoScreens

PopWindows

1.12 RMB Activate

When selected, the window under the mouse will be activated when the right mouse button is pressed, regardless of the status of AutoPoint. This is useful in getting the menu you want without either waiting for AutoPoint to activate the window or clicking into the window to make it active.

For instance, when using the screen depth gadgets, the new front screen is not activated, but with this toggle set, clicking the RMB will get the correct menus.

Note 1: If there is no window under the mouse, the first window on the screen will be activated.

Note 2: To work properly with popupmenu-type programs, Yak's CX_PRIORITY may need to be higher than the popupmenu program's CX_PRIORITY.

1.13 MMB Activate

When selected, the window under the mouse will be activated when the middle mouse button is pressed, regardless of the status of AutoPoint. This is useful in getting the menu you want without either waiting for AutoPoint to activate the window or clicking into the window to make it active.

Note 1: If there is no window under the mouse, the first window on the screen will be activated.

1.14 Key Activate

Activates window under mouse when key is pressed. You should only need one of AutoPoint or KeyActivate.

1.15 Click To Front

Lets you bring a window to the front of others by double-clicking ↩
over it.

As with AutoActivate/AutoPopToFront features, clicking to front and back is
disabled whilst a keyboard qualifier is pressed.

When
Cycle Screens
is activated, this feature works for screens, too.

See also:
ClickScreens

ClickWindows

1.16 Click To Back

Lets you push a window to the back of others by pressing and ↩
holding the
left mousebutton, then clicking the right mousebutton.

As with AutoActivate/AutoPopToFront features, clicking to front and back is
disabled whilst a keyboard qualifier is pressed.

When
Cycle Screens
is activated, this feature works for screens, too.

See also:
ClickScreens

1.17 Cycle screens

Using the same mouse sequence as Click To Back (or Click To ↩
Front), move a
screen to the back (or to the front) of others if:

either the window under the mouse is a backdrop
window (like the main Workbench window)
or there is only one window on the screen.

See also:
Click To Back

Click To Front

1.18 Screen Activate

When checked, Yak will activate screens that it shuffles by hotkeys (i.e. Screen Cycle hotkey and LCommand m hotkey). This is a toggle because conflicts arise with some programs. This is similar (but not identical) to Steve Tibbet's WindX. By activating screens I mean that it activates the window under the mouse-pointer of the new front screen.

1.19 Wild star

When on, enables the use of * as an AmigaDos pattern-matching character (like MSDOS and UNIX *). (This is what the StarBurst program does.)

1.20 No Click

When on, stops your floppy drives from "clicking" when they're empty.

1.21 Inclusion Patterns

The pattern gadgets allow you to specify which screens and windows are affected by various features of Yak. All patterns are inclusion patterns, i.e. the title of the window or screen must MATCH the pattern for the relevant feature to work. All patterns are standard AmigaDos Patterns.

The following patterns are available:

AutoScreens

ClickScreens

PopWindows

ClickWindows

1.22 AutoScreens

AutoPoint
will work on screens whose title matches this pattern.

1.23 ClickScreens

```
                Click To Front
                 and
                Click To Back
                 will work on screens whose title
matches this pattern.
```

1.24 PopWindows

```
                AutoPopToFront
                 will work on windows whose title matches this pattern.
```

1.25 ClickWindows

```
                Click To Front
                 will work on windows whose title matches this pattern.
```

1.26 TimeOut Settings

```
                With this gadgets, you can set up timeout for screen and mouse  ↩
                   blanking.


                 ScreenTimeOut

                 MouseTimeOut
```

1.27 Miscellaneous Settings

```
                This window contains a few other features controlled by these  ↩
                   gadgets.
They are:


                 Click Volume

                 AutoPoint Delay
                #
                 MMB replaces

                 MouseBlank Method
```

1.28 ClickVolume

```
Controls  the volume of the KeyClick (the sound made when you press a key).
A  volume  of  zero  means 'no click' (yes, that's obvious, but when set to
zero, the audio device won't be opened at all).  Maximum volume is 64.
```

1.29 AutoPoint Delay

Controls how long Yak should be waiting after mouse has stopped before activating window under mouse. This value must be into the 0 to 5 interval which correspound to 10 ms step.
A delay of zero means no delay, obvious isn't it ?.

1.30 MMB replaces

Controls what qualifier MMB (Middle Mouse Button) should replace.

1.31 ScreenTimeOut

If no user input (mouse or keyboard) occurs over this period (of seconds), the screen will blank. Set it to 0 to disable screen blanking altogether. The blanking is performed by opening a 2-color screen with the same displaymode as the frontmost screen.

1.32 MouseTimeOut

If mouse isn't moved in the period specified, the mouse pointer ↩ will blank.
This is only operational if the
MouseBlank Method
is not set to "None".

1.33 MouseBlank Method

This gadget determines the method by which the mouse pointer is ↩ blanked.
"None" disables mouse-blanking altogether, "Sprites" means blank mouse by disabling (all) sprites, and "Copper" means blank mouse by modifying copper list. This latter option only disables sprite 0 (the mouse-pointer), so terminal programs using a sprite for the cursor work okay, but the method is a bit less robust (the mouse occasionally comes back on).

See also:
Problems

1.34 Defining Yak Hotkeys

Clicking on the "Hotkeys..." gadget opens up a new window which ↩
lets you
create, edit and delete hotkeys. Making a key a hotkey means that when the key is pressed, Yak performs some action (of which there are many to choose from). Hotkeys are defined using a hotkey description string, which is a very flexible method of defining input events. For details on creating hotkey descriptions, see
More on Hotkeys
.

You can have as many hotkeys as you like, and each action may pertain to more than one hotkey.

There are two lists in the Hotkey Definition window; the left-hand one lists the available actions, and the other lists the hotkeys currently defined for the selected action.

To add a new hotkey, first select the action you wish it to perform (by clicking its name in the left-hand list). Then click on the Add gadget below the Hotkey list. The string gadget below this list will become active, and you should type in the hotkey description string.

For certain actions, other gadgets will become active. The Options gadget (the cycle gadget below the Actions list) determines what happens to screens when the hotkey is pressed. Currently, the options are to do nothing, to bring the Workbench to the front, and to bring the default public screen to the front. Typically this will be used in Dos Command hotkeys to automatically show the screen a window opens on.

The Argument gadget is a string gadget which becomes active for certain Action types, and provides a means of setting a string to be attached to the Hotkey.

You can edit an existing hotkey by selecting it and editing the necessary gadgets. And of course the Delete gadget deletes a hotkey from Yak's list.

When you're finished editing hotkeys, click on the "Return..." gadget to return to the main settings window, or the close gadget to 'hide' Yak. Remember to save your changes using the "Save" menu item of the main settings window.

See
Hotkey Actions

1.35 Hotkey Actions

The many actions available are:

Dos Command

Close Window

Zip Window

Shrink Window

Expand Window

Cycle Windows

Open Palette

Screen to Front

Screen to Back

Activate Workbench

Center Screen

Blank Display

Insert Text

Insert Date

Show Yak Interface

Back Cycle Windows

1.36 Dos Command

Argument: command to execute

Executes the Dos command as specified in the Argument string. Note that you may use the Options gadget to bring the Workbench or default public screen to the front (useful if the command causes a window to open).

The command is executed asynchronously, so there is no need to prepend a run command. Also, if the command generates any output (or requires input), a console window will open. You can of course specify redirection (as in the shell).

Examples:

A hotkey to open a shell:

This is traditionally attached to the hotkey "lcommand esc", and mine is set up to run the command

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

Note the use of the ALT flag in the console specification, which is poorly documented (read "not mentioned"). I actually use two hotkeys, one to start a normal shell, and one to start a CShell.

A hotkey to free unused memory:

SAS/C uses shared libraries that can often fill precious chip memory. I have a hotkey set up with the command "avail >nil: flush" which frees this memory.

A hotkey to list contents of each disk inserted:

Set the hotkey to "diskinserted", and the Argstring to "Dir df0:".

See also:

Problems

More on Hotkeys

1.37 Close Window

Argument: NONE

Close the currently active window (this is equivalent to clicking on the window's close gadget).

1.38 Zip Window

Argument: NONE

Zip the currently active window (this is equivalent to clicking on the window's "Toggle size" gadget).

1.39 Shrink Window

Argument: NONE

Make the currently active window as small as possible.

1.40 Expand Window

Argument: NONE

Make the currently active window as large as possible.

1.41 Cycle Windows

Argument: NONE

Bring the rearmost window to the front. Useful for getting at deeply 'buried' windows.

1.42 Back Cycle Windows

Argument: NONE

Works the inverse way as "Cycle Windows": puts the frontmost window to background.

1.43 Open Palette

Argument: NONE

Open a palette on the frontmost screen. The palette is run asynchronously, and you can open as many as you want (subject to memory). However, Yak cannot be terminated while palettes remain open. The Options gadget is enabled for this command (so you can open a palette specifically on the Workbench screen, if you so wish.

NOTE: You must have reqtools.library installed on your system for this action to work.

WARNING: Always close the palette window before causing the screen it's on to close, otherwise you'll at least be left with an open screen, and at worst crash the system.

1.44 Screen to Front

Argument: NONE

Bring the rearmost screen to front.

See also:
ScreenActivate

1.45 Screen to Back

Argument: NONE

Push the front screen behind all others.

See also:
ScreenActivate

1.46 Activate Workbench

Argument: NONE

Activate a Workbench window (and if necessary, bring the Workbench screen to the front). This enables you to access the Workbench menus without having to find a Workbench window to activate (if, for instance, you had a shell window the size of the screen).

1.47 Center Screen

Argument: NONE

Center horizontally the front screen.

1.48 Blank Display

Argument: NONE

Immediately blank the display.

See also:
ScreenTimeOut
.

1.49 Insert Text

Argument: text to be inserted

Inserts the text specified in the Argument string into the read stream. This string is preprocessed as follows:

```
  \n    converted to carriage-return
  \r    converted to carriage-return
  \t    converted to tab
  \   converted to backslash \
  <hotkey desc> converted to specified hotkey
  \<    converted to <
```

Because of this preprocessing, insertion strings can perform many useful tasks. For example, I have a hotkey set up to insert my name and the date, using the argument string (without the quotes):

```
  "Martin W Scott, <lcommand d>"
```

Here, the hotkey "lcommand d" is another Yak hotkey I have set up to insert the date. By using more complicated strings, you can create simple macros for other programs.

CAVEAT: Embedded hotkey strings, though useful, should be used with care. In particular, you must avoid recursive definitions, e.g.

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

Pressing f1 or f2 results in an endless loop. If you are silly enough to do this, start the Commodities exchange and make Yak inactive. Then select the Exchange's Show Interface gadget and delete/redefine the offending hotkey(s).

Another thing to be aware of is that strings that call other hotkeys (e.g. the date insertion example above) may not work as you might think. Suppose the Argument string was "<lcommand d>\n". This would actually create a carriage return and THEN the date, because by the time Yak gets the "lcommand d" hotkey, the carriage return has gone through the input handler and been sent to the active window.

1.50 Insert Date

Argument: date format string

Insert the date into the read-stream (and so into the currently active window). If you are running AmigaDos 2.1 or above, you may customise the format of the date inserted. This format is specified in the Argument string. See
Date format strings
. If you are unlucky enough to be running AmigaDos 2.0, the date is in standard DD-MMM-YY format.

Example: the format "%e %B %Y" generates dates of the form "1 May 1993".

1.51 Show Yak Interface

Argument: NONE

Show Yak's settings window. This window will open on your current screen if it is a public screen otherwise it will bringing to the front the default public screen before opening the window on it.This is the same function that the CX_POPKEY hotkey performs.

1.52 AmigaDos Patterns

AmigaDos patterns are used to include/exclude a list ↩
of named screens/windows for a particular feature. These pattern specifications aid in compatibility with other programs you may use.

Pattern matching is case-sensitive. "Amiga" is not the same as "AMIGA". The standard AmigaDos patterns available are:

```
  ? Matches a single character.
  # Matches the following expression 0 or more times.
  (ab|cd) Matches any one of the items separated by '|'.
  ~ Negates the following expression.  It matches all strings
    that do not match the expression (aka ~(foo) matches all
    strings that are not exactly "foo"
  [abc] Character class: matches any of the characters in the class.
  a-z Character range (only within character classes).
  % Matches 0 characters always (useful in "(foo|bar|%)").
  * Synonym for "#?", not available by default.  Available if

                Wild star
                 option is set.
```

If you're not used to patterns, you may find all of that quite daunting. Consult your system manual for further details. There are two basic things you'll want: either a finite list of names that the feature should be enabled on, or a finite list for which it should be disabled. To ENABLE a feature on all objects (be they screens or windows, as appropriate) use the "#?" pattern (matches everything). To enable a feature on N objects named "name1" to "nameN", use

```
  (name1|name2| ... |nameN)
```

and to DISABLE the feature for these names, prepend a tilde ~, viz.

```
  ~(name1|name2| ... |nameN)
```

An example: I don't want AutoPopToFront popping the Workbench window or any Protext (WP from Arnor) window, so exclude them with the pattern

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

Note that the second 'name' is actually a pattern, which matches any title with the text "Arnor" in it.

Another example: I don't want AutoActivation on Directory Opus's screen. It doesn't show its title in the program so I have to use ARTM or Xoper to find the screen's name, and find that it's "DOPUS.1". Figuring that the "1" would bump to "2" if I ran two copies, I decide to exclude all DOpus screens using

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

Note 1: Specifying the Workbench screen in a pattern is a bit tricky, as it's title keeps changing, depending on what window is active. For most purposes, a pattern such as "#?Workbench#?" will match, but some applications set the Workbench title to a descriptive string of their application.

Note 2: Screens or windows with titles that are unset (i.e. are NULL) always pass the patterns.

1.53 Date format strings

For date-insertion hotkeys, you must specify a locale-style date format string (and need to be running AmigaDos 2.1 or higher). The available formatting options under locale.library are as follows:

```
  %a - abbreviated weekday name
  %A - weekday name
  %b - abbreviated month name
  %B - month name
  %c - same as "%a %b %d %H:%M:%S %Y"
  %C - same as "%a %b %e %T %Z %Y"
  %d - day number with leading 0s
  %D - same as "%m/%d/%y"
  %e - day number with leading spaces
  %h - abbreviated month name
  %H - hour using 24-hour style with leading 0s
  %I - hour using 12-hour style with leading 0s
  %j - julian date
  %m - month number with leading 0s
  %M - the number of minutes with leading 0s
  %n - insert a linefeed
  %p - AM or PM strings
  %q - hour using 24-hour style
  %Q - hour using 12-hour style
  %r - same as "%I:%M:%S %p"
  %R - same as "%H:%M"
  %S - number of seconds with leadings 0s
  %t - insert a tab character
  %T - same as "%H:%M:%S"
  %U - week number, taking Sunday as first day of week
  %w - weekday number
  %W - week number, taking Monday as first day of week
  %x - same as "%m/%d/%y"
  %X - same as "%H:%M:%S"
  %y - year using two digits with leading 0s
  %Y - year using four digits with leading 0s
```

That list is pretty exhaustive, and should handle most needs; you can insert your own text freely in the format string. Some examples:

```
  "The time is %X"  gives (e.g.)  "The time is 20:44:16"
  "Have a nice %A!" gives (e.g.)  "Have a nice Monday!"
```

If you need more details, consult the AutoDocs on locale.library if you have them.

1.54 Problems

There are a few problems that I am currently aware of. Firstly, a shell created by a Dos Command hotkey doesn't have the stack or current directory as set at boot-time (in the Startup-Sequence). It DOES retain your path, though. Your Shell-Startup file should set the CD and the stack you need. By default, processes started in this way have the system boot disk (SYS:)

as their current directory.

If you're not happy with Yak's mouse-blanking, you could try the Commodore MouseBlank commodity (WB3.0), which should blank the mouse on all displays correctly.On AGA machines, Copper blanking causes problems if you are using a highres mouse pointer. Use Sprites blanking instead.

NOTE FOR AMOS USERS: I don't like AMOS (that's enough of my opinion), partly because it is so system unfriendly. It completely steals the input stream, so that mouse blankers (in programs like Yak) kick-in, thinking there's been no input, and the mouse isn't restored, because there's no mousemoves to unblank it. Because Yak uses a rather bad blanking method, problems can occur (mouse vanishes and won't come back). Two solutions:

1) Use 'Copper' blanking.
2) Set MouseBlankTime to zero. You'll still have key blank, but no timed blank.

Then the AMOS problem of the pointer disappearing should be solved.

1.55 Program History

(* = new feature)

v1.57 - Main Window setting will now remember where the 2 other windows were before closing.

- Now compiled with SAS C 6.50, code size reduced by 2 Kb !

v1.56 - Windows setting will now open always just under screen title bar whatever is the height of the screen font.

* Added a "MMB Activate" toggle.

- Close Window hotkey now works also for shell windows.

* Added a swedish and a dutch catalog.

* Windows setting will now open on current screen if it is a public one otherwise they will open on default public screen.

- Bug fixed in finding which window is under mouse when several screens are visible at the same time on display. Thanks to Pierre Carette who reported the bug and gave me a fix for it.

* New behavior ClickToFront, it now behaves as ClickToBack regarding screens if "Cycle Screens" toggle is selected.

* Added a window "Miscellaneous" to make easier integration of new features.

* AutoPoint Delay is now configurable from 0 to 5 by step of 10 ms.

```
 * Added two new icons for MagicWB 8 colors environment.

 * Added german and italian amigaguide documentations.


v1.55 - Fixed a stupid bug that prevents Yak 1.54 from running under Workbench
 2.04

 - Windows settings layout enhanced when using small height font.

 - Installer script now will not overwrite Yak's icon previously installed
 to avoid loosing tooltypes changes made by users.

 - Changes in archive distributed on aminet, source has been put in another
 archive and the other icons have been removed from distribution.

 * Added a new hotkey "Back Cycle Windows"

 * Added an italian catalog and installation script.


v1.54 * Added a tooltype LANGUAGE to select a different language than  the
   one saved in locale.prefs.

 * Added a NoClick feature for floppy drives.

 * Added a german catalog and its installation in the installer script.

 - Greatly reduced size of Yak (about 10 kb less!) It has been obtained
 by using CATCOMP_BLOCK instead of CATCOMP_ARRAY.


v1.53 - Fixed a stupid bug that displayed an uneeded requester when using
     other language than french.

 * Added localization for error messages (french catalog updated)

 - Fixed a bug in the installer script about installation of french
 catalog.


v1.52   * Added localization and french catalog.

 * Added installer script.

 - New version numbering to use RCS archiving.

   - Fixed a little bug in NewMenus rendering in 3.0.

 - Cosmetic changes to documentation and interface.

 - Now compiled with SAS C 6.3


v1.5a  - Stack requirement increased (may have been outgrowing previous
   stack size).
```

```
  - NOICON tooltype removed, APPICON tooltype added in its place.

  - Further tweaking of new Hotkey system.


 v1.5 * Extensible hotkey system added.

  * Revamped GUI.

  * Documentation now in AmigaGuide format (oh really??).

  * Date insertion now also works on 2.0 machines.

  - New preference file format, and now also uses a hotkey file.
    Prefs files for earlier versions cannot be loaded.


 <history for v1.4 and below has been omitted>
```

1.56 Compiling Yak

Yak has been written to compile under SAS/C, and as from v1.4a, using SAS/C version 6.2. The code passes cleanly with ANSI checking, so users of other ANSI-compliant compilers should have little problem recompiling - the only places changes may be needed are in SAS-specific keywords (like __saveds).

Yak uses overlays, but this can be compiled out by removing the definition USE_OVERLAYS in SCOPTIONS.

GTB 2.0b was used to generate the user-interface. The generated code (popup.c, popup.h) needs only minimal modification, specifically to use NewLookMenus under WB3.0 and above; changes are indicated by comments beginning with four asterisks, e.g. /**** ADDED */.

1.57 Credits

Yak is written entirely in C, and compiled with SAS/C 6.3. Thanks to Reza Elghazi for help in the transition to 6.2.

The Settings window gadgetry was created using the excellent (apart from the v2.0 glitch:) GadToolsBox v2.0b, from Jaba Development.

Yak makes use of reqtools.library, which is Copyright Nico François. Thanks must also go to Steve Koren for SKsh, Matt Dillon for DMouse (which answered many of my how-to questions), and Kai Iske for KCommodity, which is where the KeyClick sound was 'borrowed'.
Yak also uses WB2CLI, a very useful little link-module by Mike Sinz.

Thanks also to Heinz Wrobel for his wonderful port of RCS to the Amiga, to Pierre Carette and Sylvain Rougier for BrowserII, Martin Korndörfer for his MagicMenu and to my friends of the french Amiga mailing list who helped me for the french localization of Yak.

Thanks to Ingolf Koch who has updated the german translation of the catalog previously done by Olaf Gschweng, and who translated the installation script and the documentation ! Danke schön ;-)

Thanks to Alex Galassi who did the italian catalog, installation script and. the documentation ! Grazie ;-)

Thanks to Johan Billing who did the swedish catalog and to Peter Eriksson who did the swedish installer script.

Thanks to Patrick van Beem who did the dutch and installer catalog.

The HotKey definition documentation is taken from the ToolsManager distribution, by kind permission of Stefan Becker.

Thanks to Stefan Sticht for his public domain MouseBlanker commodity - this is where I pinched the 'Copper' mouse-blanking method.

And a big thank-you to all those people who have written to me about Yak with suggestions and bug reports.

Thanks also to Martin Huttenloher for his wonderful icons package : MagicWB.

Thanks to Osma Ahvenlampi who draw a beautiful icon for Yak in MagicWB style.

And last but not least, thanks to Martin Scott who created Yak.

1.58 Contacting the Author

I can be reached with comments, suggestions, bug reports, praise, money etc. at the following addresses:

    Gaël Marziou
    Cidex 103
    38920 CROLLES
    FRANCE

OR BY EMAIL (preferred): gael@gnlab030.grenoble.hp.com

1.59 More on HotKeys

This information is adapted from the ToolManager V2 documentation, and is reproduced with kind permission of that program's author, Stefan Becker.

How to define a Hot Key
=======================

   This chapter describes how to define a Hot Key as an Input Description String, which is then parsed by Commodities. Each time a

Hot Key is activated Commodities generates an event which is used by a Commodity in the chain. A description string has the following syntax:

```
[<class>] {[-][<qualifier>]} [-][upstroke] [<key code>]
```

All keywords are case insensitive.

'class' describes the InputEvent class. This parameter is optional and if it is missing the default 'rawkey' is used. See InputEvent classes.

Qualifiers are "signals" that must be set or cleared by the time the Hot Key is activated; otherwise no event will be generated. For each qualifier that must be set you supply its keyword. All other qualifiers are expected to be cleared by default. If you want to ignore a qualifier, just set a '-' before its keyword. See Qualifiers.

Normally a Hot Key event is generated when a key is pressed. If the event should be generated when the key is released, supply the keyword 'upstroke'. When both press and release of the key should generate an event, use '-upstroke'.

The key code is depending on the InputEvent class. See Key codes.

Note: Choose your hot keys *carefully*, because Commodities has a high priority in the InputEvent handler chain (i.e. will override existing definitions).

InputEvent classes

Commodities supports most of the InputEvent classes that are generated by the input.device. This section describes those classes that are most useful for Hot Keys.

'rawkey'
: This is the default class and covers all keyboard events. For example 'rawkey a' or 'a' creates an event every time when the key "a" is pressed. You must specify a key code for this class. See rawkey key codes.

'rawmouse'
: This class describes all mouse button events. You must specify a key code for this class. See rawmouse key codes.

'diskinserted'
: Events of this class are generated when a disk is inserted in a drive. This class has no key codes.

'diskremoved'
: Events of this class are generated when a disk is removed from a drive. This class has no key codes.

```
Qualifiers
==========

   Some keyword synonyms were added to Commodities V38. These are
marked with an '*'.

'lshift', 'left_shift' *
     Left shift key.

'rshift', 'right_shift' *
     Right shift key.

'shift'
     Either shift key.

'capslock', 'caps_lock' *
     Caps lock key.

'caps'
     Either shift key or caps lock key.

'control', 'ctrl' *
     Control key.

'lalt', 'left_alt' *
     Left alt key.

'ralt', 'right_alt' *
     Right alt key.

'alt'
     Either alt key.

'lcommand', 'lamiga' *, 'left_amiga' *, 'left_command' *
     Left Amiga/Command key.

'rcommand', 'ramiga' *, 'right_amiga' *, 'right_command' *
     Right Amiga/Command key.

'numericpad', 'numpad' *, 'num_pad' *, 'numeric_pad' *
     This keyword *must* be used for any key on the numeric pad.

'leftbutton', 'lbutton' *, 'left_button' *
     Left mouse button. See note below.

'midbutton', 'mbutton' *, 'middlebutton' *, 'middle_button' *
     Middle mouse button. See note below.

'rbutton', 'rightbutton' *, 'right_button' *
     Right mouse button. See note below.

'repeat'
     This qualifier is set when the keyboard repeat is active. Only
     useful for InputEvent class 'rawkey'.

   Note: Commodities V37 has a bug which prevents the use of
'leftbutton', 'midbutton' and 'rbutton' as qualifiers. This bug is
```

fixed in V38.

Key codes

Each InputEvent class has its own key codes:

Key codes for InputEvent class 'rawkey'

Some keywords and synonyms were added to Commodities V38. These are marked with an '*'.

'a'-'z', '0'-'9', ...
: ASCII characters.

'f1', 'f2', ..., 'f10', 'f11' *, 'f12' *
: Function keys.

'up', 'cursor_up' *, 'down', 'cursor_down' *
'left', 'cursor_left' *, 'right', 'cursor_right' *
: Cursor keys.

'esc', 'escape' *, 'backspace', 'del', 'help'
'tab', 'comma', 'return', 'space', 'spacebar' *
: Special keys.

'enter', 'insert' *, 'delete' *
'page_up' *, 'page_down' *, 'home' *, 'end' *
: Numeric Pad keys. Each of these key codes *must* be used with the 'numericpad' qualifier keyword!

Key codes for InputEvent class 'rawmouse'

These keywords were added to Commodities V38. They are not available in V37.

'mouse_leftpress'
: Press left mouse button.

'mouse_middlepress'
: Press middle mouse button.

'mouse_rightpress'
: Press right mouse button.

Note: To use one of these key codes, you must also set the corresponding qualifier keyword, e.g.

```
rawmouse leftbutton mouse_leftpress
```

Examples for Hot Keys

```
======================

'ralt t'
     Hold right Alt key and press "t"

'ralt lalt t'
     Hold left *and* right Alt key and press "t"

'alt t'
     Hold either Alt key and press "t"

'rcommand f2'
     Hold right Amiga key and press the second function key

'numericpad enter'
     Press the Enter key on the numeric pad

'rawmouse midbutton leftbutton mouse_leftpress'
     Hold middle mouse button and press the the left mouse button

'diskinserted'
     Insert a disk in any drive.
```

1.60 Localization

```
Yak is now localized.  It currently supports english as built-in language and
dutch, french, german, italian and swedish with supplied catalogs. If you want
to localize Yak to your own language, then you're welcome. There's no need to
be a developper to make a translation, the only requisite is to know very well
your language :-)

Be aware that there are several things that can be translated :

Yak itself by writing a catalog.
The installer script.
The amigaguide documentation.

You can translate part of this or the whole as you want.

In the Catalogs directory, you will find a file named yak.ct which contains the
strings  used  by  Yak  interface and which has been built from yak.cd that
you an also find in the Catalogs directory..
It is ready to be translated, let's take an example :

MISCELLANEOUS_STRING

; Miscellaneous;


So,  if  you  want to translate "Miscellaneous" into french, you should put its
translation on the empty line as this :

MISCELLANEOUS_STRING
Divers
```

```
; Miscellaneous;
```

So, when you have translated all the strings of yak.ct you can either use catcomp or one of his PD replacements such as CatEdit, KitCat or FlexCat to generate your catalog or you can send me yak.ct so that I will generate the catalog myself.
There are 5 lines of credits in Yak menu reserved to the translator so don’t hesitate to put your name here and your address.