

Yak

COLLABORATORS

	<i>TITLE :</i> Yak		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		January 18, 2023	

REVISION HISTORY

<i>NUMBER</i>	<i>DATE</i>	<i>DESCRIPTION</i>	<i>NAME</i>

Contents

1	Yak	1
1.1	Yak v1.5a Documentation	1
1.2	Copyright	2
1.3	Current users: Notez-bien	2
1.4	Introduction	3
1.5	Jokes	4
1.6	Limitations	4
1.7	Starting Yak	4
1.8	Changing Yak's settings	5
1.9	Toggles	6
1.10	AutoPoint	7
1.11	AutoPopToFront	7
1.12	RMB Activate	7
1.13	Key Activate	7
1.14	Click To Front	8
1.15	Click To Back	8
1.16	Screen To Back	8
1.17	Screen Activate	8
1.18	Wild star	9
1.19	Inclusion Patterns	9
1.20	AutoScreens	9
1.21	ClickScreens	9
1.22	PopWindows	9
1.23	ClickWindows	10
1.24	Miscellaneous Settings	10
1.25	ClickVolume	10
1.26	ScreenTimeOut	10
1.27	MouseTimeOut	10
1.28	MouseBlank Method	10
1.29	Defining Yak Hotkeys	11

1.30 Hotkey Actions	12
1.31 Dos Command	12
1.32 Close Window	13
1.33 Zip Window	13
1.34 Shrink Window	13
1.35 Expand Window	13
1.36 Cycle Windows	14
1.37 Open Palette	14
1.38 Screen to Front	14
1.39 Screen to Back	14
1.40 Activate Workbench	14
1.41 Center Screen	15
1.42 Blank Display	15
1.43 Insert Text	15
1.44 Insert Date	16
1.45 Show Yak Interface	16
1.46 AmigaDos Patterns	16
1.47 Date format strings	17
1.48 Problems	18
1.49 Program History	19
1.50 Compiling Yak	22
1.51 Credits	22
1.52 Contacting the Author	23

Chapter 1

Yak

1.1 Yak v1.5a Documentation

Yak Version 1.5a
by Martin W. Scott

TABLE OF CONTENTS

Copyright and Distribution

Current users: Notez-bien

Introduction

Limitations

Starting Yak

Changing Yak's settings

Toggles

Patterns

Miscellaneous

Hotkeys

Hotkey Actions

AmigaDos Patterns

Date format strings

Problems

Program History

Compiling Yak

Credits and Thanks

Contacting the Author
Supplementary Hotkeys.doc

1.2 Copyright

Yak (the binary, sources and documentation) is
Copyright © 1992, 1993 Martin W. Scott. All Rights reserved.

Yak is freely redistributable. The source is included, and you are permitted to modify it for personal use, but any modifications made must NOT be distributed. If you have made changes you think others would like, send them to me and I'll include them in future versions.

Since Yak is free, it comes with NO WARRANTIES. The author is not responsible for any loss or damage arising from the use of Yak; the user takes all such responsibility.

No charge may be made for Yak, other than a nominal copy fee. Yak may not be distributed with a commercial product without the author's prior consent. Yak must be distributed with all documentation intact and unaltered, and preferably with source too. Permission is expressly granted to Fred Fish to distribute on his fine collection of disks.

Although Yak is freeware, DONATIONS WOULD BE GLADLY ACCEPTED, either money or stuff you've written yourself. See
Contacting the Author

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 tooltip. This tooltip replaces the NOICON tooltip.

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.

Finally, the French documentation has been temporarily discontinued, to give my translator time to work on the new documentation. However, I thought it worth releasing v1.5 in the meantime.

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 spelling - 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:

- o AutoPoint (sunmouse) that only activates when mouse stops
can also specify which screens to include/exclude.
(Compatible with popup-menu type programs)
- o AutoPop windows (bring them to front) when they're
auto-activated;
- o KeyActivate windows (when key pressed);
- o Activate windows when menu button pressed;
- o Click windows to front or back; may also specify which
screens/windows to include/exclude.
- o Cycle screens with mouse;
- o Mouse and Screen blanking (hotkey blanking too);
- o 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 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 22K 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.

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". 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) and the AppIcon ones. 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

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 Activate
toggle below.

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

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

Screen To Back

Screen Activate

RMB Activate

Wild star

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.

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 Key Activate

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

1.14 Click To Front

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

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

See also:

ClickScreens

ClickWindows

1.15 Click To Back

Lets you push a window to the back of others by pressing and \leftrightarrow 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.

See also:

ClickScreens

1.16 Screen To Back

Using the same mouse sequence as Click To Back, move a screen to \leftrightarrow the back 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

1.17 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.

1.18 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.)

1.19 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
```

1.20 AutoScreens

```
AutoPoint
will work on screens whose title matches this pattern.
```

1.21 ClickScreens

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

1.22 PopWindows

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

1.23 ClickWindows

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

1.24 Miscellaneous Settings

Yak has a few other features controlled by these gadgets. They ↔
are:

ClickVolume
ScreenTimeOut
MouseTimeOut
MouseBlank Method

1.25 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.

1.26 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.

1.27 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".

1.28 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

1.29 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, consult the supplementary Hotkeys.doc documentation.

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

1.30 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

1.31 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 "`!command 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
Hotkeys.doc

1.32 Close Window

Argument: NONE

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

1.33 Zip Window

Argument: NONE

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

1.34 Shrink Window

Argument: NONE

Make the currently active window as small as possible.

1.35 Expand Window

Argument: NONE

Make the currently active window as large as possible.

1.36 Cycle Windows

Argument: NONE

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

1.37 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.

1.38 Screen to Front

Argument: NONE

Bring the rearmost screen to front.

See also:

`ScreenActivate`

1.39 Screen to Back

Argument: NONE

Push the front screen behind all others.

See also:

`ScreenActivate`

1.40 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).

1.41 Center Screen

Argument: NONE

Center horizontally the front screen.

1.42 Blank Display

Argument: NONE

Immediately blank the display.

See also:

ScreenTimeOut
.

1.43 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.

1.44 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".

1.45 Show Yak Interface

Argument: NONE

Show Yak's settings window (if necessary by bringing the Workbench screen to the front and opening the window). This is the same function that the CX_POPKEY hotkey performs.

1.46 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 Protect (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.

1.47 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.

1.48 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.

The Close window hotkey doesn't work for shell windows. This is because they handle intuition events differently from other windows.

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.

NOTE FOR AMOS USERS: I hate 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.

1.49 Program History

(* = new feature)

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.

v1.4a * Added RMB Activate toggle - window under mouse is activated when right mouse button pressed.

* New improved mouse-blanking: Method 'Sprites' is same as before, but method 'Copper' has been completely rewritten; it still only blanks sprite 0, but is now compatible with LacePointer.

- Now compiled with SAS/C 6.2.

v1.4 - Range-checking now performed on number-gadgets.

- NewLookMenus for WB 3.0.
- v1.3f * Settings window now font-sensitive (uses screen font).
- * ScreenToBack by mouse-clicks now activates first window on front screen when no window is under mouse (this helps programs like MagicMenu).
 - * Screen blanking now opens screen with same displaymode as front one.
- Further mouse-blanking bug removed.
 - Conflicts between window-cycle hotkey and key activation resolved, although they don't really go well together.
 - Window-cycle (via hotkey) now also cycles Workbench windows; when initially written, Yak emulated IHelp's (CBM OS2.0) function of cycling application windows only, but I get so many reports from people saying that this function doesn't work, I figured folks really want to cycle ALL windows.
 - More enforcer-hits removed (oh I wish I had an 030 etc., then I could fix these all by myself, instead of relying on others informing me).
- v1.3e - Version string added.
- Now a choice of mouse-blanking method (so quit the flames...).
- v1.3d - Fixed bug in KeyActivate routine (was losing first keystroke).
- v1.3c - Fixed small enforcer hit which may have caused Window cycle to fail.
- v1.3b * New ClickWindows pattern, allows you to specify which windows may be clicked-to-front (n.b. has no effect on click-to-back).
- * Mouse-blank timeout may now be specified.
 - [Click volume now specified via integer gadget].
- v1.3a - Now all Intuition calls are outside the input-handler to avoid possible deadlocks (thanks to Eddy Carroll for reporting this).
- * New hotkey: Screen to back. Previously, this was hardcoded to ICommand m, but can now be changed.
 - * New hotkey: Blank display. Instant screen-blank.
 - * Can now click screens to back in the titlebar (previously not permitted due to system conflicts).
 - * Palette runs asynchronously now, and you can have as many open as
-

you like. They MUST be closed before their parent screen closes, and before Yak is terminated.

- Now a disk-insertion halts screen-blanking.
- Centering of screens via hotkey adjusted by 1 pixel, to be consistent with Display, AutoCentre etc. (i.e. I was wrong...)
- If started from Workbench (the usual way) Yak now sets it's current directory to SYS:; this is then inherited by shells created with PopCommand.
- Some more mouse-blanking problems fixed. Also, screen-blanking mouse-flicker fixed - hardly anyone uses Yak's blanker, so it went unnoticed.
- Fixed small bug in channel allocation priority.
- Bug fixed in hotkey cancellation (by clearing string gadget). Previously used to cancel a hotkey only for current session, and still continue to show old key definition. How do I miss these things?

v1.3 * Executable is now overlaid. Now uses less memory than earlier versions, even with new features!

- * Can now specify which screens AutoActivate and ClickToFront/Back operate on, by means of a standard AmigaDos pattern.
 - * Date insertion uses locale.library-style format specification. (Unfortunately, date insertion feature now requires WB 2.1).
 - * AutoPopToFront feature added.
 - * Improved mouse-blanking: now Yak only blanks sprite 0, so programs that use other sprites are unaffected.
 - * PopCommand now retains paths (but not directory/stack in startup-sequence).
 - Auto-activation and key-activation now works more sensibly.
 - Somehow managed to remove volume check for clicking. Now a volume of zero means that no click will be attempted.
 - Date insertion now inserts key-up events to aid compatibility with other programs.
 - Window cycling (via hotkey) now works on all screens (previously just on WB, but I didn't mention it in the docs).
- NB - Contention code in v1.2c was rather severe in cases, and unnecessary if I changed the way Yak AutoActivated. Now Yak only AutoActivates (and AutoPops) when no qualifier or mouse-button is pressed. Contention-checking code has therefore been removed. Should be no problems with well-behaved programs.
-

NB - Settings file changed; now uses S:Yak.prefs instead of S:Yak.config; this is because format of that file changed. A 'Convert' program is supplied to create a 1.3 file from existing 1.2 file.

- [Changed prefs window layout; looks nicer now (I think).]

v1.2c - Inclusion of HotKey documentation (adapted from Stefan Becker's ToolManager documentation).

- Enforcer hit removed. (But there may be more).

- Improved contention-checking; now Yak's AutoActivate works with Dragit V2, for instance.

v1.2b - maximum length of hotkey definitions increased from 29 to 49 characters to allow (almost) all possible definitions.

v1.2a - small bug crept into v1.2; wasn't removing old hotkey when changing existing definition.

<history for v1.2 and below has been omitted>

1.50 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 */`.

1.51 Credits

Yak is written entirely in C, and compiled with SAS/C 6.2. 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 Mike Meyer et. al. for `Mg3`, 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.

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, and especially to Gaël Marziou who drew the nice Yak icons and translated the documentation into French -- thanks Gaël!

1.52 Contacting the Author

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

BEFORE END OF JULY 1993:

Martin W. Scott,
23 Drum Brae North,
Edinburgh EH4 8AT
United Kingdom

AFTER END OF JULY 1993:

Martin W. Scott, c/o
557 Great Western Road,
Ground flat right,
Aberdeen AB1 7PA
United Kingdom

OR BY EMAIL UNTIL END OF JULY 1993: mws@castle.ed.ac.uk
