Yak

COLLABORATORS

	TITLE : Yak		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Yak

1.1   Yak v2.00 Documentation

```
                              Yak Version 2.00
                  by Gaël Marziou and Philippe Bastiani
                      released on 13 March 1995.
```

```
    Introduction...


            Features

            Required System

            Installation

            Limitations

            Current users: Notez-bien
              The commodity program...


            Overview

            Starting

            Tooltypes
              The preferences editor...


            Overview

            Starting the editor

            Tooltypes

            Main Window

            Mouse Cycling Window
```

```
                    Blanking Window

                    Miscellaneous Window

                    Hotkeys Window
                       Other topics...


                    Copyright and Distribution

                    Problems / Non Problems

                    Program History

                    Credits and Thanks

                    Development team

                    Contacting the Authors

                       INDEX
```

1.2   Yak v2.00 Documentation: Features

```
                   Features
```

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:

```
        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 CLI Command or an ARexx script;
                Insert text (with embedded hotkeys);
                Insert date (in custom format if locale present);
                Close/Zip/Shrink/Enlarge windows;
                Move/Cycle screens and windows;
                Activate Workbench;
                Blank display;
                Pop up a palette on front screen (needs reqtools);
                Menu shortcut

        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.
```

```
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.

Fully localized when running OS 2.1 or more!

And  it's  quite  small!
```

1.3   Yak v2.00 Documentation: Jokes

```
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)
```

1.4   Yak v2.00 Documentation: Required System

```
                Required System

Yak requires version 2.04 of the Amiga operating system or higher.
Some features are only available in 3.0+ system version.
```

It supports commodities exchange standard and
                localization
                 with
locale.library

1.5   Yak v2.00 Documentation: Localization

Localization

Yak is fully localized. It currently supports english as built-in language
and danish, dutch, finnish, 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.
Credits for the translator should go into the 'about' requester which
string identifier is About_Yak_STRING.

1.6   Yak v2.00 Documentation: Installation

Installation

Double-click on one of the icons located in the Installation drawer,
according to the language you want to use.
Simply follow the steps. Help is available almost everywhere.

1.7   Yak v2.00 Documentation: Limitation

Limitations

Yak only has a plain screen-blanker. Since the introduction of OS 2.0,
there has been a profusion of fancy modular screen blankers. I think most
people have their own favourite fancy blanker, so I didn't include one in
Yak, don't ask for one.

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).

1.8   Yak v2.00 Documentation: Notez-bien

                  Current users: Notez-bien

Yak's preference file format has changed again. A
                  Convert
                   program is
supplied to create a 2.0 preference file from the 1.xx preference files.
The ClickWindows pattern will be lost!.

Yak has being split into two commodities:
     -
                  Yak main commodity
                        -
                  Yak's preferences editor
                  Of   course,   there have been   many other changes, and you ←
                        should browse
through this document to accustomise yourself with them.

1.9   Yak v2.00 Documentation: Convert program

Convert program

If you are a user of previous version of Yak, you can convert your own
settings to the new preferences format. Just double click on the convert
icon or run it from a shell.

In case you use tooltype LANGUAGE for previous version of Yak, you can also

set it for Convert to benefit of automatic naming of hotkeys in your
favorite language.

1.10   Yak v2.00 Documentation: Yak commodity

            Yak commodity

The commodity program will simply install and process all the actions
specified within the preferences editor (see
            The Preferences Editor
            , for
more information about setting the preferences).

On demand Yak may also launch the Preferences editor either via HotKey, or
via AppIcon.

You can control it via the Commodities Exchange.

1.11   Yak v2.00 Documentation: Starting Yak

Starting Yak

Yak is designed to be run from the sys:WBStartup drawer, but may be 'run'
from a Shell (not recommended, as it will use significantly more memory).

It 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

Yak doesn't use Shell parameters.

In any case, Yak reads the icon parameters.

1.12   Yak v2.00 Documentation: Yak tooltypes

Using tooltypes

The only tooltypes that Yak takes are the standard Workbench ones (such as
DONOTWAIT), the standard Commodities ones (such as CX_POPUP), and the
specific tooltypes ones (such as LANGUAGE). 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
PREFSPRG      STRING       Full name of preferences editor  "SYS:Prefs/Yak"
```

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

The CX_PRIORITY may be useful in enabling Yak and other commodities to work
better together.

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 preferences editor
is launched.   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 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.

1.13   Yak v2.00 Documentation: The Yak's preferences editor

The Yak's Preferences Editor

With the preferences editor you can manage the global configuration of Yak.
This configuration gets automatically loaded by the Yak handler.

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).

The windows are AppWindows: if you drop a preferences file on the editor
the preferences will be loaded.

You can control the editor via the Commodities Exchange.

1.14   Yak v2.00 Documentation: Starting the preferences editor

Starting the preferences editor

o This program may be started simply double-clicking on it's icon.

Per default the program is located within "SYS:Prefs" drawer.

o On demand the Yak's handler may launch the editor:

    Double click its icon;

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

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

    Start Yak again (from Workbench tool icon);

    Via the Commodities Exchange (on the Extras disk).

1.15   Yak v2.00 Documentation: Yak editor tooltypes

Using tooltypes

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

ToolType	Category	Description	Default
CX_PRIORITY	INTEGER	Priority of this commodity	0
FONT	STRING	Name of the font to use for GUI	Not specified
LANGUAGE	STRING	Name of language to use	Not specified
CREATEICONS	BOOLEAN	Create icons	YES

The FONT tooltype is used to force the preferences editor to use
your favorite font instead of the screen font. For example, if you want
the GUI to use Helvetica 15 set FONT=helvetica 15 or and FONT="helvetica 15".

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.

When the CREATEICONS tooltype is set to YES, the editor will create an icon
for every preferences file that is created with the Save As menu item.

1.16   Yak v2.00 Documentation: Main window

        Main Window

o To the top of the window, the 3 gadgets groups allow you to specify basic
  settings:

                 Windows Activation

                 Screens Activation

                 AutoPopToFront
             o Several buttons open up others windows with advanced settings.
  There are:

                 Mouse Cycling

                 Blanking

                 Edit Hotkeys

                 Miscellaneous
             o This window has several attached
             menus
                 .

o The lower 3 buttons are used to control the configuration:

  With the Save button you can save the configuration permanently into the
  file ENVARC:Yak.prefs and into the file ENV:Yak.prefs before quitting the
  editor.
  For a temporary change, use the Use button which will save the
  configuration into the file ENV:Yak.prefs   (this file will not survive a
  machine reset) and will quit the editor.
  Click on the Cancel button or on the CLOSE gadget to quit the editor
  without saving.

1.17 Yak v2.00 Documentation: Menus

The menus

o Project menu:

  With the Open, Save and Save As items you can load and save the
  configuration.
  The About item opens an information requester.
  Selecting the Hide item will hide the editor without saving.
  Selecting the Quit item will leave the editor without saving.

o Edit menu:

  The Reset To Default item sets the default configuration.
  The Last Saved item loads the last saved configuration from the file
  ENVARC:Yak.prefs.
  With the Restore item you can load the configuration that is active.

```
o Settings menu:

   With  the Create Icons item,  you  can choose whether  the Save As item
   should create an icon or not.
```

1.18   Yak v2.00 Documentation: Windows Activation

```
                Windows Activation

The following toggles are available:


             Key Activate
                  Activates window on a key pressed


             RMB Activate
                  Activates window with the right mouse button


             MMB Activate
                  Activates window with the middle mouse button


             AutoPoint
                  Activates the window under the mouse
```

1.19   Yak v2.00 Documentation: Key Activate

1.20   Yak v2.00 Documentation: Key Activate

1.21   Yak v2.00 Documentation: Key Activate

```
                Key Activate

Activates window under mouse when key is pressed.

You should only need one of AutoPoint or KeyActivate.

Note that the KeyActivate function only take place when  NO mouse button is
pressed.  Not only does this avoid conflict  with   other   programs,   it
provides a way of preventing activation when it's not desired.

See also:
                AutoPoint
```

1.22   Yak v2.00 Documentation: RMB Activate

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.

1.23   Yak v2.00 Documentation: MMB Activate

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.

1.24   Yak v2.00 Documentation: AutoPoint

                AutoPoint

When actived, the window under the mouse will be activated. 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.

AutoPoint settings:

    -
                AutoScreens
                    -
                AutoPoint Delay
                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

1.25   Yak v2.00 Documentation: AutoScreens

                AutoScreens

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

See also
                    Default Title
                    and
                    AmigaDos Patterns
                    .

1.26   Yak v2.00 Documentation: AutoPoint Delay

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 ?.

1.27   Yak v2.00 Documentation: Screens Activation

Screens Activation

When actived, 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 a window of this screen,
this window can be :

- the window that was active last time you have visited this screen by
cycling screens via Yak's hotkeys.

- the window under the mouse-pointer if this screen has never been visited
before.

- the first window of this screens if both previous conditions failed.

1.28   Yak v2.00 Documentation: AutoPopToFront

                    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.

AutoPopToFront setting:

    –
                    PopWindows
                  Note that the AutoPopToFront and AutoPoint 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:
                    AutoPoint

1.29   Yak v2.00 Documentation: PopWindows

                    PopWindows

AutoPopToFront will work on windows whose title matches this pattern.

See also
                    AmigaDos Patterns
                    .

1.30   Yak v2.00 Documentation: Default Title

                    Default Title of a screen

The default title of a screen doesn't vary. Windows of applications can
change the screen title but not the default screen title.
For example, the title of the workbench screen is often changed by
applications when their own windows are active but the default title of the
workbench screen remains unchanged.
So, when setting a screen pattern you should use the default title of the
screen which can be found when no window is active on this screen.
Most of the time, finding the default title will be very easy but in some
rare cases you will gain to use an utility such as ARTM.

See also
                    AmigaDos Patterns
                    .

1.31   Yak v2.00 Documentation: AmigaDos Patterns

                    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 default tilte, 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#?)

Yet another example : I don't want AutoActivation on BrowserII's screen.

The defaul title (when no window is active) is BrowserII followed by the
names of the authors and the copyright years. As these years will evolve,
I decide to exclude BrowserII screen by using :

          ~(BrowserII#?)

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

1.32   Yak v2.00 Documentation: Mouse Cycling Window

                      Mouse Cycling Window

Clicking on the "Mouse Cycling" gadget opens up a new window which allows
you to configure some cycling actions on reception of mouse events.

These actions are:

              Window To Front

              Window To Back

              Cycle Screens
              When you've finished editing, click the Ok gadget on to return to ←
                  the main
settings window.

1.33   Yak v2.00 Documentation: ClickToFront

                      Window to front

This allows you to bring a window to the front of others via mouse buttons.

ClickToFront settings:

   –
                  Active
                   –
                  ScreenToFront
                   –
                  Required Clicks
                   –
                  Exclude Workbench Window
                   –
                  Include Screens
                   –
                  Definition
              Default: will bring any window (except for the Workbench window) ←
                  to front

```
        by double-clicking over it with left mouse button.
```

1.34   Yak v2.00 Documentation

```
Active: use this gadget to enable or to disable the correspounding action.
```

1.35   Yak v2.00 Documentation: ScreenToFront

```
Screen To Front: when there's no window or a backdrop window under mouse,
                 clicking  to  front will bring to front the screen under
                 mouse.
```

1.36   Yak v2.00 Documentation: Required Clicks

```
Required Clicks: number of mouse clicks required to trig the action.
```

1.37   Yak v2.00 Documentation: Include Screens

```
        IncludeScreens: an pattern on screens default title which allows  ←
           you to
        specify on which screens this action is active.

See

         AmigaDos Patterns
        .
```

1.38   Yak v2.00 Documentation: Exclude Workbench Window

```
Exclude Workbench Window: when on means that ClickToFront action is
                         disabled for the workbench window.
```

1.39   Yak v2.00 Documentation: ClickToBack

```
                Window to back

This allows you to push a window to the back of others via mouse buttons.

ClickToBack settings:

  -

                Active
```

```
                       _
               ScreenToBack
                   _
               Required Clicks
                   _
               Include Screens
                   _
               Definition
             Default: will push any window to the back of others by pressing  ←
                 and holding
        the left mousebutton, then clicking the right mousebutton.
```

1.40   Yak v2.00 Documentation: Screen to back

```
Screen To Back: when there's no window or a backdrop window under mouse,
                clicking  to back will bring  the screen  under mouse to
                back.
```

1.41   Yak v2.00 Documentation: CycleScreens

```
               Screens cycling

This allows you to cycle through screens via mouse buttons.

CycleScreens settings:

   _

                 Active
                   _
                 Required Clicks
                   _
                 Include Screens
                   _
                 Definition
               Default: will cycle through all screens by double-clicking with  ←
                   middle
          mouse button.
```

1.42   Yak v2.00 Documentation: Blanking Window

```
               Blanking Window

This window contains a few gadgets to control blanking:


               MouseBlank Method

               Blank mouse on key pressed
```

                        TimeOut
                    When you've finished editing, click the Ok gadget on to return to ←
                        the main
settings window.

1.43   Yak v2.00 Documentation: MouseBlank Method

                    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

1.44   Yak v2.00 Documentation: Blank mouse on key pressed

Blank mouse on key pressed

This is a toggle if set, mouse pointer will be blanked as soon as you hit
the keyboard. If it is not set, mouse will blanked only on time out.

1.45   Yak v2.00 Documentation: TimeOut Settings

                    TimeOut Settings

With this gadgets, you can set up timeout for screen and mouse blanking:

                    ScreenTimeOut

                    MouseTimeOut

1.46   Yak v2.00 Documentation: ScreenTimeOut

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.

1.47   Yak v2.00 Documentation: MouseTimeOut

```
               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".

1.48   Yak v2.00 Documentation: Miscellaneous Window

```
               Miscellaneous Window
```

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

```
               Click Volume

               Wild star

               Black border

               No Click

               UNIX Dirs

               MMB Shift

               UNIX Root

               Full Workbench
              When you've finished editing, click the Ok gadget on to return to  ←
                 the main
settings window.
```

1.49   Yak v2.00 Documentation: Click Volume

```
Click Volume
```

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.

Note that, when you don't release a key, the auto-repeat will simulate the
same key pressed several times, but Yak will emit one click in order to
avoid slowing down scrolling using arrow keys in a text editor for example.

1.50   Yak v2.00 Documentation: Wild star

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.)

1.51   Yak v2.00 Documentation: Black Border

Black Border

When on, Yak will add a black border to all your screens. This feature is
only available in 3.0+ system version as it uses a new flag of the graphics
library.

1.52   Yak v2.00 Documentation: No Click

No Click

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

1.53   Yak v2.00 Documentation: UNIX Dirs

UNIX Dirs

When on, allow you to use:

    '..' as parent directory
    '.' as current directory

in all directories operations.

This is done by patching the following standard libraries functions:

     AssignLate
     AssignPath
     CreateDir
     DeleteFile
     LoadSeg
     Lock
     MakeLink
     MatchFirst
     NewLoadSeg
     Open
     ParsePattern
     Rename
     SetComment
     SetFileDate
     SetProtection

1.54   Yak v2.00 Documentation: UNIX Root

```
                UNIX Root

When on, allow you to use:

   '/'  as root directory of current device (equivalent to amigados ':')

in all directories operations.

This feature is only available when
                UNIX Dirs
                is on.

WARNING: as this option changes the meaning of a symbol understood by
         amigados, it may confuse some applications.
```

1.55   Yak v2.00 Documentation: MMB Shift

```
MMB Shift

When on, Yak will translate the middle mouse button (on a 3 buttons mouse)
into the left shift key.
More accurately, this will happen only when pressing left mouse button
while holding middle mouse button. This make easier to select multiple
icons on the workbench or multiple files in a file-requester without having
to touch keyboard.
Other usages of middle button in other key combinations won't get changed.
```

1.56   Yak v2.00 Documentation: Full Workbench

```
                Full Workbench

When on, the workbench title bar will be hidden and the workbench backdrop
window will be expanded so that it covers the whole workbench screen.
This is useful if you use a backdrop picture and want it to occupy the
whole space on your workbench screen.
The screen cycling gadget (in top right screen corner) will be hidden too
but that's not a problem because you can cycle trough screens by using Yak's

                Mouse Cycling
                feature or
                Screen to Back
                hotkey action.

Note: This has no effect if you don't use a backdrop workbench window.
    Some applications like GNU emacs makes the workbench title bar back
      to visible when they start.
```

1.57   Yak v2.00 Documentation: Hotkeys Window

```
              Hotkeys Window
```

Clicking the "Edit Hotkey" gadget on 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.

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 Hotkeys window:
- the left-hand lists the available
              actions
              ,
- the right-hand 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 press the Add button
below the Hotkey list.

To delete a hotkey from Yak's list, use the Delete button.

To edit an existing hotkey, first click its name the right-hand list. Then,
to change:
- its name, enter the new name into the gadget below the list;
- its state, use the cycle gadget;
- its
              hotkey description string
              , press the Definition button;
- its
              options
              , press the Option button.

When you've finished editing hotkeys, click the Ok gadget on to return
to the main settings window.

See:
              Hotkey Actions

              Hotkey Description String

1.58   Yak v2.00 Documentation: Hotkey Description String

```
              Hotkey Description String
```

Here, you can define or modify hotkeys used by Yak.

The current hotkey description is displayed within the string gadget.
There are two transparent ways to define a hotkey:

```
                   Automatic Definition

                 Manual Definition
                 Once you've finished editing a hotkey, click the Ok gadget on to  ←
                       register
the new description string.
Click the Cancel gadget on to abort the hotkey definition.
```

1.59   Yak v2.00 Documentation: Automatic Definition

```
                   Automatic Definition
```

```
Simply choose the hotkey class (by clicking on the left-hand cycle gadget);
then hit the keys and mouse buttons, you want to use: Yak convert them into
an input description string.

Toggle mode is used.  So, if you press a key twice it will be removed from
the defintion.

With the stroke gadget (the right-hand cycle gadget) you can set the method
by which the hotkey will be activated.

See also:
                   More on Hotkeys
```

1.60   Yak v2.00 Documentation: Manual Definition

```
                 Manual Definition
```

```
Click on the string gadget and type in the hotkey description string: Yak
parses string and updates gadgets.


WARNING: The commodities.library is not bug free:

        -here is a wrong description string:
               'NumericPad a'

        -here are some unusable description strings:
               'NumericPad [' and 'NumericPad ]'  (with a french keyboard)

See also:
                   More on Hotkeys
```

1.61   Yak v2.00 Documentation: Hotkey Options Window

```
        Hotkey Options Window
```

Here, you can define or modify options used by each hotkey.

Each action type has its own options editing window.

Once you've finished editing the options, click on the Ok gadget to register them.
Click the Cancel gadget on to abort editing.

See:
```
        Hotkey Actions
        list.
```

1.62   Yak v2.00 Documentation: Hotkey Actions

```
        Hotkey Actions
```

The many actions available are:

```
        Execute Command

        Close Window

        Zip Window

        Shrink Window

        Expand Window

        Move Window

        Cycle Windows

        Open Palette

        Screen to Front

        Screen to Back

        Move Screen

        Activate Workbench

        Blank Display

        Insert Text

        Insert Date

        Show Yak Interface
```

```
                         Set default public screen

                         Menu shortcut
```

1.63   Yak v2.00 Documentation: Execute Command

```
                  Execute Command

o Action:
  Executes a CLI command or an ARexx script.
  The program is executed asynchronously.


o Options:

  –

                    Action on screens
                    (default: 'no screen change')

  – Specifications of program:
     –

                    Type
                    (default: 'CLI')
     –

                    Stack size
                    (default: '4000')
     –

                    Priority
                    (default: '0')
     –

                    AREXX Port
                    (default: 'AREXX')
     –

                    Name
```

1.64   Yak v2.00 Documentation: Action on screen

```
Action on screen

You can set the screen which should be moved to front before the program
is started. This one can be:

   –the current screen,
   –the Workbench,
   –the default public screen.
```

1.65   Yak v2.00 Documentation: Program type

```
Program type
```

Two different types of programs are supported:

```
    -CLI commands,
    -ARexx scripts.
```

1.66   Yak v2.00 Documentation: Stack size

```
Stack size
```

This sets the stack size of the new process which runs the program.

Note: only available for CLI commands.

1.67   Yak v2.00 Documentation: Priority

```
Priority
```

This sets the priority of the new process which runs the program.

Note: only available for CLI commands.

1.68   Yak v2.00 Documentation: AREXX Port

```
AREXX Port
```

This sets the AREXX port where to send the AREXX command. If you are using
a simple AREXX script the AREXX Port should be the default one: AREXX.

Note: only available for AREXX commands.

1.69   Yak v2.00 Documentation: Program name

```
              Program to execute
```

The file name of the program to start.
There is no need to prepend a 'run' or a 'rx' command.

Some CLI commands:

  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 command to "Dir df0:".

Note: if the CLI command generates any output (or requires input), a
console window will open. You can of course specify redirection (as in the
shell).

See also:
                Problems

1.70   Yak v2.00 Documentation: Close Window

                Close Window

o Action:
  Close the selected window (this is equivalent to clicking on the window's
  close gadget).

o Options:

  −

                Window selection
                (default: 'active window').

1.71   Yak v2.00 Documentation: Window selection

Window Selection

You may now specify on which window this hotkey is active.
This one can be:

  −the Active Window,
  −the Window Under Mouse,
  −a window whose title match with the specified.

1.72   Yak v2.00 Documentation: Zip Window

```
                Zip Window
```

o Action:
  Zip the selected window (this is equivalent to clicking on the window's
  'Toggle size' gadget).

o Options:

  _

                Window selection
                (default: 'active window').

1.73   Yak v2.00 Documentation: Shrink Window

```
                Shrink Window
```

o Action:
  Shrink the selected window.

o Options:

  _

                Window selection
                (default: 'active window').

  _

                Resizing parametres
                (default: 'horizontaly' and 'verticaly'
                                'keep the screen title bar visible').

1.74   Yak v2.00 Documentation: Expand Window

```
                Expand Window
```

o Action:
  Expand the selected window.

o Options:

  _

                Window selection
                (default: 'active window').

  _

                Resizing parametres

```
                        (default: 'horizontaly' and 'verticaly'
                                        'keep the screen title bar visible').
```

1.75   Yak v2.00 Documentation: Resizing parametres

```
Resizing parametres

You may size the window: -horizontally,
                         -vertically,
                         -in two directions;

and keep the screen title bar visible.
```

1.76   Yak v2.00 Documentation: Move Window

```
                  Move Window

o Action:
  Move the selected window.


o Options:

    -

                Window selection
                (default: 'active window').

    -

                Moving parametres
                (default: 'left upper corner').
```

1.77   Yak v2.00 Documentation: Moving parametres

```
Moving parametres (Screen/Window)

You may move the screen/window: -horizontally,
                                -vertically,
                                -in two directions;

and keep: -the screen title bar visible (only available for the
                                        'Move Window' hotkeys),
          -the active window visible (only available for the
                                     'Move Screen' hotkeys).
```

1.78 Yak v2.00 Documentation: Cycle Windows

```
                Cycle Windows

o Action:
  Cycle windows. Useful for getting at deeply 'buried' windows.


o Options:

  _

                Screen selection
                (default: 'frontmost screen').


  _

                Exclude workbench drawers
                (default: 'no')
  _

                Back cycle windows
                (default: 'no')
  _

                Activate only
                (default: 'no')
  _

                By task name
                (default: 'no')
```

1.79 Yak v2.00 Documentation: Screen selection

```
                Screen Selection

You can now specify on which screen this action is active.
This one can be:

  -the Rearmost Screen  (only available for the
                        'Screen To Front' hotkeys),
  -the FrontMost Screen (not available for the
                        'Screen To Front' hotkeys),
  -the Active Screen,
  -the Screen Under Mouse,
  -a screen whose title match with the pattern specified.
   (see
                Default Title
                and
                AmigaDos Patterns
                ).
```

1.80 Yak v2.00 Documentation: Exclude workbench drawers

Exclude Workbench drawers

When selected, the hotkey is disabled on the Workbench drawers.

1.81   Yak v2.00 Documentation: Back cycle windows

Back cycle windows

Cycle windows in the other direction.

1.82   Yak v2.00 Documentation: Activate only

Activate only

When off, it causes windows to be depth re-arranged, this is the default
behavior well know by Yak 1.x users..
When on, windows stay as they are, they just get activated.

1.83   Yak v2.00 Documentation: By task name

                By task name

When on, you can now specify a
                pattern
                  on the name of the task which
handles some windows.

To find the name of a task attached to a window, you can use ARTM and in
the window description, you will find the name of this task on the line
"UserPort: ... mp-SigTask".

Some examples:

To exclude ToolManager windows (docks), you can use the following pattern:
      "~(ToolManagerHandler)"

To cycle only BrowserII windows, you can use the following pattern:
      "#?BrowserII#?"

1.84   Yak v2.00 Documentation: Open Palette

                Open Palette

o Action:
  Open a palette.

o Options:

  –

                action on screens
                (default: 'no screen change')

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.

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.

1.85   Yak v2.00 Documentation: Screen to front

                Screen to front

o Action:
  Bring the selected screen to front.

o Options:

  –

                Screen selection
                (default: 'rearmost screen').

See also:
                Screens Activation

1.86   Yak v2.00 Documentation: Screen to back

                Screen to back

o Action:
  Push the selected screen behind all others.

o Options:

  –

                Screen selection
                (default: 'frontmost screen').

See also:
                    Screens Activation

1.87   Yak v2.00 Documentation: Move Screen

                 Move Screen

o Action:
  Move the selected screen.

o Options:

   –

                Screen selection
                (default: 'frontmost screen').

   –

                Moving parametres
                (default: 'centre screen');

1.88   Yak v2.00 Documentation: Activate Workbench

Activate Workbench

o Action:
  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).

o Options:
  NONE

1.89   Yak v2.00 Documentation: Blank Display

Blank Display

o Action:
  Immediately blank the display.

o Options:
  NONE

1.90   Yak v2.00 Documentation: Insert Text

                    Insert Text

o Action:
  Insert text into the read stream.

o Options:

  –

                    Text to be inserted

1.91   Yak v2.00 Documentation: Text Format

Text format

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.

1.92   Yak v2.00 Documentation: Insert Date

                       Insert Date

o Action:
  Insert the date into the read-stream (and so into the currently active
  window).

o Options:

    –

                 Date format string
                 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.
If you are unlucky enough to be running AmigaDos 2.0, the date is in
standard DD-MMM-YY format.

1.93   Yak v2.00 Documentation: Show Yak Interface

                     Show Yak Interface

o Action:
  Launch Yak's preferences editor.
  This is the same function that the CX_POPKEY hotkey performs.

o Options:

    –

                   action on screens
                   (default: 'no screen change')

1.94   Yak v2.00 Documentation: Set default public screen

Set default public screen

o Action:
  Set the current screen as the default one.   This will work only if the
  current screen is a public one.

o Options:
  NONE

1.95   Yak v2.00 Documentation: Menu shortcut

                  Menu shortcut

o Action:
  Sends an input event to menu of the current active window, simulating a user
  choice in the menu (if any). This can be useful to add some keyboards
  shortcuts to applications that don't have ones or not enough.

o Options:

  –

                  Menu numbers
                .

1.96   Yak v2.00 Documentation: Menu numbers

Menu numbers

The arguments specify the menu number, the item number and optionnally the
subitem number. All these numbers start at 0 following the Intuition rule
(i.e first menu is menu 0 and not 1).

Note that a bar label counts as an item, if before the item you wan to
access there are several bar labels you must take them into account while
specifying the item number.

However, don't be afraid if you make mistakes in specifying the argument
as Yak do all needed checkings before sending a menu event in order to
avoid disturbing your application menu handler.

Here are some examples:

You find that the workbench has not enough menu shortcuts :

To access the "Last Message" item of the "Workbench" menu :
        – Menu number: 0
        – Item number: 4
        – Subitem   : no

To access the "Only Icons" subitem of the "Show" sub menu of the "Window"
menu : – Menu number   : 1
        – Item number   : 7
        – Subitem   : yes
        – SubItem number: 0

1.97   Yak v2.00 Documentation: Date format string

Date format string

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.

1.98   Yak v2.00 Documentation: More on Hotkeys

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.

1.99   Yak v2.00 Documentation: Copyright

                     Copyright and Distribution

1.100   Yak v2.00 Documentation: Problems / Non Problems

```
                Problems / Non Problems
```

There are a few problems that we are currently aware of, but there are also
problems that are not really problems.

```
            Current Dir for Execute Command

            OneKeyII and Yak

            Mouse blanking

            Mouse Cycling and Caps Lock
```

1.101   Yak v2.00 Documentation: Current Dir for Execute Command

A shell created by a Execute Command hotkey doesn't have the 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.

1.102   Yak v2.00 Documentation: OneKeyII and Yak

OneKeyII users: disable OnekeyII when you edit a hotkey!

1.103   Yak v2.00 Documentation: MouseBlanking

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
an highres mouse pointer : use Sprites blanking or Commodore MouseBlank
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.

1.104   Yak v2.00 Documentation: Mouse Cycling and Caps Lock

You have setup mouse cycling function to bring window to front to double
left click, it works OK but when you put CapsLock on, it doesn't work any
more. That's not a problem, that's a feature, from the commodity point of
view, CapsLock is a qualifier among others.

Anyway, Yak's flexibility allows you to solve your problem : just edit the
string gadget whhere the definition is displayed and add the following
(without the quotes) '-CapsLock' which means that CapsLock will be ignored
as a qualifier for this definition.

1.105   Yak v2.00 Documentation: Program History

Program History

 (* = new feature)

 v2.00   Major update

 <history for v1.60 and below has been omitted>

1.106   Yak v2.00 Documentation: Credits and Thanks

Credits and Thanks

Yak is written entirely in C, and compiled with SAS/C 6.51 but it can also
be compiled with DICE.

The Prefs editor GUI was created first using 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 WbPath, a very useful little link-module by Ralph Babel.

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.

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 Nicola Salmoria who explained me the principle of the "black
border" feature and for his great package NewIcons.

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

1.107 Yak v2.00 Documentation: Yak development team

Yak development team

Yak development is a team work.

     Programming

        Gael Marziou
        Philippe Bastiani

     Translations, suggestions and beta testing

        Alex Galassi   : italian doc, installer script and catalog.
        Ingolf Koch   : german doc, installer script and catalog.
        Christian Høj   : danish doc, installer script and catalog.
        Patrick van Beem : dutch installer script and catalog.
        Johan Billing   : swedish catalog.
        Peter Eriksson   : swedish installer script.
        Arttu Kärpinlehto: finnish installer script and catalog.

1.108 Yak v2.00 Documentation: Contacting the authors

            Contacting the authors

We 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_Marziou@grenoble.hp.com

and:

            Philippe Bastiani
            7 Boulevard Ricard
            13003 MARSEILLE
            FRANCE

Before reporting a problem, try to figure out if it's really a problem,
read carefully the doc and the
                Problems / Non Problems
                section.

1.109   Yak v2.00 Documentation

                Index

        AutoPoint

        AutoPopToFront

        Blackborder

        Blanking

        Click Volume

        ClickToBack

        ClickToFront

        CycleScreens

        Hotkey Actions

        Hotkey Description String

        Key Activate

        MMB Activate

        MMB Shift

        No Click

        RMB Activate

        Screens Activation

        UNIX dirs

        UNIX root

        Wild star

        Windows Activation