NAME

FvwmIconMan − an fvwm icon manager

SYNOPSIS

FvwmIconMan is spawned by fvwm, so no command line invocation will work.

DESCRIPTION

FvwmIconMan is an icon manager modeled after the TWM icon manager. The user may have multiple icon managers, each of which armed with a list of window types which it manages. For example, the user may have one manager which lists only emacs windows, and another which lists everything else. You may also specify what resolution each icon manager uses, for example, one icon manager may manage windows on all desks, and another may manage only those on the current desk, page or screen. FvwmIconMan can display the miniature icons provided by fvwm for its managed windows. The managers may have a maximum number of columns (and so grows vertically), a maximum number of rows (and then grows horizontally), or stay at a fixed size, and adjust the size of the window buttons to fit (think win95's Taskbar). And when support is compiled in for the X Shape extension, then the manager windows may be shaped.

You can specify actions to be run when mouse, or key events are received. For example, you could bind the first mouse button to iconify the selected window, and make bindings for the arrow keys to navigate the manager window without the mouse.

FvwmIconMan can be set to display which window currently has the keyboard focus, and by binding the select event (see below) to the fvwm Focus function, you can emulate the TWM icon manager's behavior.

INITIALIZATION

During initialization, FvwmIconMan searches though the fvwm configuration file for the options which are described below. It is highly recommended that you make FvwmIconMan be a sticky window. And if you want to make use of the followfocus option, and/or binding an action to Focus, then you should make FvwmIconMan clicktofocus. Also, when using the Shape option, it's recommended that the FvwmIconMan window not be decorated at all by fvwm.

INVOCATION

FvwmIconMan can be invoked by inserting the line 'Module FvwmIconMan' in the .fvwm2rc file. If FvwmIconMan is to be spawned during fvwm's initialization, then this line should be placed in the StartFunction declarations, or it can be bound to a menu, mouse button, or keystroke to invoke it later.

If you wish to run FvwmIconMan in a transient mode, such as with the built in window list, then pass "-Transient" as an argument. The invocation "Module FvwmIconMan -Transient" will do nicely. In this mode, FvwmIconMan will pop up one manager window directly under the cursor. When the mouse button is released, it will execute the appropriate action, and then exit. Things are somewhat complicated by the fact that you can specify that FvwmIconMan creates multiple manager windows, behavior which is unsuitable when running transiently. So, when running transiently, FvwmIconMan will only create one manager window. Use the manager id 'transient' to specify options for this manager window.

FvwmIconMan may accept an alias name as an argument. For example, "Module FvwmIconMan FvwmIconMan-Variant2".

CONFIGURATION OPTIONS REFERENCE CHART

FvwmIconMan has acquired quite a few options. I assume others share my dislike of paging though a long man page, so here is a terse reference chart describing the available options. They are described in more detail in the next section.

```
Name            Description                 Default
```

NumManagers number of managers 1
Action binds command to event Mouse 0 N sendcommand Iconify
Background default background gray
ButtonGeometry size of button in pixels
Colorset default colorset
DontShow list of windows to ignore
DrawIcons use mini icons false
FocusAndSelectButton flat grey black
FocusAndSelectColorset
FocusButton style for focused buttons up grey black
FocusColorset
FollowFocus show which win has focus false
Font 8x13
Foreground default text color white
Format describes button label "%c: %i"
IconName manager icon name FvwmIconMan
IconAndSelectButton up black grey
IconAndSelectColorset
IconButton style for icon buttons up black grey
IconColorset
ManagerGeometry size of manager in buttons 0x1
MaxButtonWidth max width of a button
MaxButtonWidthByColumns
NoIconAction animate iconification NOP
PlainButton style for normal buttons up black grey
PlainColorset
ReliefThickness size of button relief 2
Resolution global/desk/page/screen page
Reverse normal, icon or none none
SelectButton style for selected buttons flat black grey
SelectColorset
Shape use shape extension false
Show list of windows to show
ShowOnlyIcons only icons visible false
ShowNoIcons icons are not displayed false
ShowTransient transient windows visible false
ShowOnlyFocused only focused visible false
Sort keep managers sorted name
SortWeight weight for sorting
Tips Tool Tips mode none
TipsDelays Tool Tips mapping delays 1000 300
TipsFont Font for Tool Tips default fvwm font
TipsColorset Tool Tips Colorset 0
TipsFormat describes Tips label the Format value
TipsBorderWidth Tool Tips border size 1
TipsPlacement Tips placement vs button updown
TipsJustification Tips Just vs button leftup
TipsOffsets Tips placement Offsets 3 2
Title manager title FvwmIconMan
TitleButton style for title button raisededge black grey
TitleColorset
UseWinList honor WinListSkip? true

CONFIGURATION OPTIONS

With the exception of the nummanagers option, all of the options may be defined on a per-manager basis. So, for example, the user may have his emacs manager with a red foreground, and his xterm manager with a blue one. A configuration line may therefore have one of two forms:

*FvwmIconMan: OptionName OptionValue
To specify that the *OptionName* takes the value *OptionValue* for all managers.

*FvwmIconMan: ManagerId OptionName OptionValue
To specify that the option *OptionName* takes the value *OptionValue* for manager *ManagerId*. *ManagerId* may either be a positive integer, or the string "transient". An integer id refers to managers which FvwmIconMan creates when running normally, and an id of "transient" refers to the single manager which FvwmIconMan creates when running transiently.

The old syntax, that uses an asterisk instead of white spaces before *ManagerId* and *OptionName*, is supported too, but it is obsolete now.

The following options may be specified:

*FvwmIconMan: NumManagers *num*
num is a positive integer specifying the total number of icon managers. Since FvwmIconMan would like to know how many managers there are before handling any manager specific options, this should come first. The default is 1.

*FvwmIconMan: [id] Action *type binding*
Binds an FvwmIconMan command to an event. *Type* may be one of the values: Key, Mouse, or Select. Actions are described in the following section ACTIONS.

*FvwmIconMan: [id] Background *background*
Specifies the default background color.

*FvwmIconMan: [id] ButtonGeometry *geometry*
Specifies the initial geometry of an individual button in pixels. If the specified height is 0, then the button height is determined from the font size. X and Y coordinates are ignored.

*FvwmIconMan: [id] Colorset *colorset*
The default colorset used. Overrides background and foreground.

*FvwmIconMan: [id] DrawIcons *value*
If your version of fvwm is capable of using mini icons, then this option determines if FvwmIconMan displays the mini icons. Otherwise, it generates an error message. "true" means that mini icons are shown for iconified windows, "false" that mini icons are never shown, and "always" that mini icons are shown for all windows.

*FvwmIconMan: [id] FocusAndSelectButton *style* [*forecolor backcolor*]
Same as the plainbutton option, but specifies the look of buttons which are both selected, and have the keyboard focus.

*FvwmIconMan: [id] FocusAndSelectColorset *colorset*
Works like focusandselectbutton but uses colorsets instead. The style setting can still only be applied with focusandselectbutton.

*FvwmIconMan: [id] FocusButton *style* [*forecolor backcolor*]
Same as the plainbutton option, but specifies the look of buttons whose windows have the keyboard focus.

*FvwmIconMan: [id] FocusColorset *colorset*
Works like focusbutton but uses colorsets instead. The style setting can still only be applied with focusbutton.

*FvwmIconMan: [id] FollowFocus *boolean*
If *true*, then the button appearance reflects which window currently has focus. Default is false.

*FvwmIconMan: [id] Font *font*
Specifies the font to be used for labeling the buttons. The default is 8x13.

*FvwmIconMan: [id] Foreground *foreground*
Specifies the default foreground color.

*FvwmIconMan: [id] Format *formatstring*
A printf like format string which describes the string to be printed in the manager window for each managed window. Possible flags are: %t, %i, %c, and %r for the window's title, icon title, class, or resource name, respectively. The default is "%c: %i". **Warning**: m4 reserves the word *format*, so if you use m4, take appropriate action.

*FvwmIconMan: [id] IconName *iconstring*
Specifies the window icon name for that manager window. *Iconstring* may either be a single word, or a string enclosed in quotes. The default is "FvwmIconMan".

*FvwmIconMan: [id] IconAndSelectButton *style* [*forecolor backcolor*]
Same as the plainbutton option, but specifies the look of buttons whose windows are iconified and the button is selected.

*FvwmIconMan: [id] IconButton *style* [*forecolor backcolor*]
Same as the plainbutton option, but specifies the look of buttons whose windows are iconified.

*FvwmIconMan: [id] IconAndSelectColorset *colorset*
Works like IconAndSelectButton but uses colorsets instead. The style setting can still only be applied with iconbutton.

*FvwmIconMan: [id] IconColorset *colorset*
Works like iconbutton but uses colorsets instead. The style setting can still only be applied with iconbutton.

*FvwmIconMan: [id] ManagerGeometry *geometry*
Specifies the initial geometry of the manager, in units of buttons. If *height* is 0, then the manager will use *width* columns, and will grow vertically once it has more than *width* windows. Likewise, if *width* is 0, it will use *height* rows, and grow horizontally. If both are nonzero, then the manager window will be exactly that size, and stay that way. As columns are created, the buttons will narrow to accommodate. If the geometry is specified with a negative y coordinate, then the window manager will grow upwards. Otherwise, it will grow downwards.

*FvwmIconMan: [id] MaxButtonWidth *width*

Defines a maximum for the width of a button (in pixels). By default there is no maximum. A value of 0 resets the default. The maximum is only used with a non growing manager (the ManagerGeometry option specifies non zero width and height).

*FvwmIconMan: [id] MaxButtonWidthByColumns *col*

This is another way to set the button width. col is the number of columns of icons. The button width is determined by dividing the total width of FvwmIconMan by the number of columns. For example if the width of FvwmIconMan manager is 1024, MaxButtonWidthByColumns is 4 then MaxButtonWidth is 256. This is useful when you do not know, at config time, the width of the manager, for example, for a swallowed FvwmIconMan.

*FvwmIconMan: [id] NoIconAction *action*

Tells FvwmIconMan to do *action* when a NoIcon style window is iconified or de-iconified. Relevant coordinates are appended to *action* so that the icon can be traced to an FvwmIconMan button. An example action is "*FvwwmIconMan: NoIconAction SendToModule FvwmAnimate animate". A blank or null action turns this feature off.

*FvwmIconMan: [id] PlainButton *style* [*forecolor backcolor*]

Specifies how normal buttons look. *style* may be one of *flat*, *up*, *down*, *raisededge*, or *sunkedge*, and describes how the button is drawn. The color options are both optional, and if not set, then the default colors are used. If on a monochrome screen, then the *style* option is ignored, but must still be set.

*FvwmIconMan: [id] PlainColorset *colorset*

Works like plainbutton but uses colorsets instead. The style setting can still only be applied with plainbutton.

*FvwmIconMan: [id] ReliefThickness *num*

num is an integer specifying the number of pixels thick that the relief at the edge of non-flat buttons should be. Setting this to 0 will produce flat buttons, as if the values for *FocusAndSelectButton*, *FocusButton*, *IconAndSelectButton*, *IconButton*, *PlainButton*, *SelectButton*, and *TitleButton* were all set to *flat*. If*num* is ne gative, the button will be inverted as if you had used *Reverse* for all classes.

*FvwmIconMan: [id] Resolution *resolution*

Specifies when the manager will display an entry for a certain window. *resolution* may take one of the following values: global, desk, page, screen, !desk, !page, or !screen. If global, then all windows of the appropriate type (see the show and dontshow options below) will be shown. If desk, then only those windows on the current desk are shown. If page, then only those windows on the current page are shown. If screen, then only those windows on the current Xinerama screen are shown. !desk reverses the sense of desk, displaying only those windows not on the current desk. Likewise, !page shows only those windows not on the current page and !screen shows only those windows not on the current Xinerama screen. The default is page. If Xinerama is not active or only a single screen is used, page and screen are equivalent.

This configuration line is respected when FvwmIconMan is running as well, the resolution is changed dynamically.

*FvwmIconMan: [id] Reverse *class*

Causes certain classes of buttons to have their relief lines reversed so that up and down styles are reversed. This has no affect on flat buttons. The class can be icon, normal or none. The default is

none.

*FvwmIconMan: [id] SelectButton *style* [*forecolor backcolor*]
Same as the plainbutton option, but specifies the look of buttons when the mouse is over them.

*FvwmIconMan: [id] SelectColorset *colorset*
Works like selectbutton but uses colorsets instead. The style setting can still only be applied with selectbutton.

*FvwmIconMan: [id] Shape *boolean*
If *True*, then use make the window shaped. Probably only useful if you have multiple columns or rows. If FvwmIconMan wasn't compiled to support the Shape extension, this generates an error message. When using shaped windows, it's recommended that a fvwm style is made for FvwmIconMan that has no borders. Otherwise, fvwm will get confused.

*FvwmIconMan: [id] Sort *value*
If *name*, then the manager list is sorted by name. If *namewithcase*, then it is sorted by name sensitive to case. If *id*, then the manager list is sorted by the window id, which never changes after the window is created. If *weighted*, then the manager list is sorted by weight (see the description of *sortweight* below). Or it can be set to *none*, which results in no sorting. Default is *name*.

*FvwmIconMan: [id] SortWeight *weight pattern-list*
Assigns the specified *weight* to windows that match *pattern-list*. The list is made up of patterns of the form *type=pattern*, where type is one of *class*, *resource*, *title*, or *icon*, and pattern is an expression of the same format used in the fvwm style command (minimalistic shell pattern matching). Multiple sort weights can be given. Each window is matched against the list of sort weights, in order, and is given the weight from the first match. Lower-weighted windows are placed first in the manager list. For example:

```
*FvwmIconMan: Sort       weighted
*FvwmIconMan: SortWeight 1 class=XTerm title=special*
*FvwmIconMan: SortWeight 10 class=XTerm
*FvwmIconMan: SortWeight 5
```

In this example, xterm windows whose titles start with "special" (weight 1) are listed first, followed by everything but other xterms (weight 5), and the other xterms (weight 10) are listed last. If no default weight (empty pattern list) is given, the default weight is 0. Only relevant if the sort type is set to *weighted*.

*FvwmIconMan: [id] Title *title-string*
Specifies the window title string for that manager window. *Titlestring* may either be a single word, or a string enclosed in quotes. The default is "FvwmIconMan". This will be drawn in the title bar of the manager window, if any, and in the title button, which is the button drawn when the manager is empty.

*FvwmIconMan: [id] TitleButton *style* [*forecolor backcolor*]
Same as the plainbutton option, but specifies the look of the title button (the button drawn when the manager is empty). The manager's title is drawn in the title button.

*FvwmIconMan: [id] UseWinList *boolean*
If *true*, then honor the WinListSkip style flag. Otherwise, all windows are subject to possible management according to the show and dontshow lists.

The two following options control which windows get handled by which managers. A manager can get two lists, one of windows to show, and one of windows to ignore. If only the *show* list is given, then that manager will show only the windows in the list. If only the *DontShow* list is given, then the manager will show all windows except those in the list. If both lists are given, then a window will be shown if it is not in the *DontShow* list, and in the *Show* list. And finally, if neither list is given, then the manager will handle all windows. Each list is made up of patterns of the form *type=pattern*, where type is one of *class*, *resource*, *title*, or *icon*, and pattern is an expression of the same format used in the fvwm style command (minimalistic shell pattern matching). Quotes around the pattern will be taken as part of the expression. If a window could be handled by more than one manager, then the manager with the lowest id gets it.

*FvwmIconMan: [id] Show *pattern list*
If a window matches one of the patterns in the list, then it may be handled by this manager.

*FvwmIconMan: [id] DontShow *pattern list*
If a window matches one of the patterns in the list, then it may not be handled by this manager.

*FvwmIconMan: [id] ShowTransient *boolean*
Show transient windows in the list (default false).

*FvwmIconMan: [id] ShowOnlyIcons *boolean*
Only iconified windows are shown if *boolean* is true.

*FvwmIconMan: [id] ShowNoIcons *boolean*
Only windows that are not iconified are shown if *boolean* is true.

*FvwmIconMan: [id] ShowOnlyFocused *boolean*
Only window with the focus is shown if *boolean* is true.

The following two options control tips.

*FvwmIconMan: [id] Tips *value*
where *value* can be always, needed or false. Default is false, no tips are displayed. With always, tips are enabled. With needed, a tip is displayed only if either the button string is truncated or the tip string is not equal to the button string. This configuration line is respected when FvwmIconMan is running as well.

*FvwmIconMan: [id] TipsDelays *delay* [*mappeddelay*]
where *delay* and *mappeddelay* are time out values in milliseconds. If no *mappeddelay* is given *delay* is assumed. Default is 1000 300. When the cursor is on a button, FvwmIconMan wait *delay* milliseconds before displaying the tip. In the case where a tip is already mapped and the cursor goes to another button, FvwmIconMan waits *mappeddelay* milliseconds before displaying the new tip.

*FvwmIconMan: [id] TipsFont *fontname*
Specifies the font to be used for tips. Default is the default fvwm font.

*FvwmIconMan: [id] TipsColorset *colorset*
Specifies the colors for tips window. Default is colorset 0.

*FvwmIconMan: [id] TipsFormat *formatstring*
Similar to the Format option but for the tips window. The default is the format string from the Format option.

*FvwmIconMan: [id] TipsBorderWidth *pixels*
Specifies the border width (in pixels) of the tips window. Default is 1.

*FvwmIconMan: [id] TipsPlacement *value*
where *value* can be up, down, right, left, updown or leftright. This value specifies the position of the tips window relative to its button. Default is updown where buttons on the top half of the screen get tips below the button, otherwise the tips are above the button.

*FvwmIconMan: [id] TipsJustification *value*
where *value* can be leftup, rightdown or center. Specifies the justification (direction) of the tips window relative to its button after the tips window has been placed. Default is leftup which means that if a tip is placed above or below its button, then the left border of the tip and of the button are aligned. If the tip is placed on the left or on the right of its button, leftup aligns the top borders. rightdown and center work like leftup but in different directions. The alignment is adjusted by the TipsOffset option. See next option.

*FvwmIconMan: [id] TipsOffsets *placementoffset justoffset*
where *placementoffset* and *justoffset* are offsets in pixels for the TipsPlacement and TipsJustification configuration option. Default is 3 2.

ACTIONS

Actions are commands which may be bound to an event of the type: a key press, a mouse click, or the mouse entering a window manager button - denoted by the action types *Key*, *Mouse*, and *Select*.

Normally, actions bound to a mouse click are executed when the button is pressed. In transient mode, the action is executed when the button is released, since it is assumed that FvwmIconMan was bound to some mouse event. A tip/warning: FvwmIconMan still keeps track of the mouse button and any modifier keys in this case, so if you bind FvwmIconMan to say, meta-button3, then it would be wise to ensure that the action you want to execute will be executed when the meta-button3 event occurs (which would be the button release, assuming you kept your finger on the meta key).

The syntax for actions are:

Key actions: Key *Keysym Modifiers FunctionList*
Keysym and *Modifiers* are exactly the same as for the fvwm *Key* command.

Mouse actions: Mouse *Button Modifiers FunctionList*
Button and *Modifiers* are exactly the same as for the fvwm *Mouse* command.

Select actions: Select *FunctionList*

A *FunctionList* is a sequence of commands separated by commas. They are executed in left to right order, in one shared context - which currently only contains a pointer to the "current" button. If a button is selected (typically by the mouse pointer sitting on it) when the action is executed, then the current button is initialized to that button. Otherwise, it points to nothing.

Most of the available commands then modify this "current" button, either by moving it around, making it

become the selected button, or sending commands to fvwm acting on the window represented by that button. Note that while this current button is initialized to be the selected button, the selected button does not implicitly follow it around. This way, the user can send commands to various windows, without changing which button is selected.

Commands take five types of arguments: *Integer*, *Manager*, *Window*, *Button*, and *String*. A *String* is a string specified exactly as for fvwm - either in quotes or as a single word not in quotes. Again, you may bind a sequence of commands to an event, by listing them separated by commas.

Window and *Button* types look exactly the same in the .fvwm2rc file, but are interpreted as either specifying a managed window, or a FvwmIconMan button representing a window. They can either be an integer (which is interpreted module N where N is the number of buttons - so 0 is the first and -1 is the last), or one of the strings: *Select*, *Focus*, *Up*, *Down*, *Right*, *Left*, *Next*, *Prev*. *Select* and *Focus* refer to the currently selected or focused button or window. *Up*, *Down*, *Right*, and *Left* refer to the button or window above, below, to the right of, or to the left of the current button in the manager window, allowing navigation around the manager window. *Next* and *Prev* designates the window, button, or manager after or before the current button, allowing navigation of the one dimensional list of windows which is drawn in the manager window. If the manager is sorted, *Next* and *Prev* move through the windows in the sorted order.

The *Manager* type can either be an integer, *Next*, or *Prev*. The meaning is analogous to that of the*Button* type, but in terms of the integral index of the managers, restricted to managers which are nonempty.

The following functions are currently defined:

bif *Button Integer/String*
: A relative branch instruction. If *Button* is *Select* or *Focus*, then take the branch if there is a selected button or a focused button. If *Button* is an integer, then branch if nonzero. If it is one of *Up*, *Down*, *Right*, *Left*, *Next*, *Prev*, then the branch is taken when the current button can move in that direction. If the branch is taken, then *Integer* commands are skipped. No backwards branches are allowed.

bifn *Button Integer/String*
: The complement of bif. The branch is taken if *Button* evaluates to false, by the criteria listed for bif.

gotobutton *Button*
: Sets current button to *Button*. If *Button* is an integer, then the current button is set to *Button* modulo the number of buttons, in the whichever manager contains the selected button, if any.

gotomanager *Manager*
: Sets button to button 0 of *Manager*. This will only go to a visible, nonempty manager. So an integral argument is taken modulo the number of such managers.

jmp *Integer/String*
: Executes a relative jump of *Integer* instructions. Backwards jumps are not allowed. The jump is computed relative to the instruction following the jmp.

label *String*
: Provides a label that previous instructions can jump to. It will not be visible to subsequent jump instructions, and the same label can be used multiple times in the same instruction list (though it would be perverse to do so.)

print *String*
: Prints *String* to the console. Useful for debugging actions.

printdebug
: Prints defined actions to the console. Should only be used by developers. To enable this command, set CONFIG and FUNCTIONS variables to '1' in the modules/FvwmIconMan/debug.h and recompile this module.

quit Quits FvwmIconMan.

refresh Causes all manager windows to redraw themselves.

ret Stop executing the entire action.

searchback *String*
: Sets button to button before the current one whose printed string in the manager window matches specified *String*, which may contain wildcards.

searchforward *String*
: Sets button to button after the current one whose printed string in the manager window matches specified *String*, which may contain wildcards.

select Selects the current button, if any. If a select action has been specified, it will then be run. Therefore, it is considered unwise to set the select button in the select action.

sendcommand *Command*
: Sends the fvwm command *Command* to the window represented by the current button, if any.

warp Warps cursor to current button, if any.

Examples:

```
gotobutton select, gotobutton Down, select
```
Selects the button below the currently selected button. Since the current button is already initialized to the selected button, this may be shortened to "gotobutton Down, select".

```
gotobutton Up, select
```
Selects the button above the currently selected button.

```
gotobutton 0, select
```
Selects the first button of the current manager. If there is no current manager, which is the case when no button is selected, then this does nothing.

```
gotobutton -1, select
```
Selects the last button of the current manager.

```
gotobutton focus, select
```
Selects the button corresponding to the focused window.

```
gotobutton focus, Iconify
```
Sends the fvwm command Iconify to the focused window. Note that this does not change the selected

button.

```
bif Next 3, gotobutton 0, select, ret, gotobutton Next, select
```
If a button is selected, and it's the last button, go to button 0. If it's not the last button, go to the next button. Otherwise, do nothing. Basically, this action cycles through all buttons in the current manager.

```
bif select 7, bif focus 3, gotomanager 0, select, ret, gotobutton focus, \
  select, ret, gotobutton down, select
```
This is good for sending to FvwmIconMan with a SendToModule command. If there is a selected button, it moves down. Otherwise, if there is a focused button, it is selected. Otherwise, button 0 of manager 0 gets selected.

```
bif select Select, bif focus Focus, gotomanager 0, select, ret, label Focus, \
  gotobutton focus, select, ret, label Select, gotobutton down, select
```
Same as previous, but using the label instruction.

In addition to being bound to keys and mice, actions can be sent from fvwm to FvwmIconMan via the SendToModule command. Don't quote the command when using SendToModule. Also, due to a bug in the current version of fvwm, don't quote FvwmIconMan either.

SAMPLE CONFIGURATIONS

This first example is of a the simplest invocation of FvwmIconMan, which only has one manager, and handles all windows:

```
############################################################
# Load any modules which should be started during
# fvwm initialization
ModulePath /usr/lib/X11/fvwm:/usr/bin/X11
Module FvwmIconMan

# Make FvwmIconMan title-bar-less, sticky, and give it an icon
Style "Fvwm*"      Icon toolbox.xpm,NoTitle,NoHandles,Sticky
Style "FvwmIconMan" HandleWidth 5, Handles, BorderWidth 5


############################################################
############################################################
#Definitions used by the modules

*FvwmIconMan: NumManagers       1
*FvwmIconMan: Resolution        global
*FvwmIconMan: Background        slategrey
*FvwmIconMan: Foreground        white
*FvwmIconMan: Font              7x13
*FvwmIconMan: ButtonGeometry    100x0
*FvwmIconMan: ManagerGeometry   1x0-0+0
```

This example is the Reader's Digest version of my personal configuration. It has two managers, one for emacs and one for everything else, minus things with no icon title. Only windows on the current page are displayed. The use of the *drawicons* and *shape* options requires that fvwm and FvwmIconMan are compiled with the correct options. Note how the geometry and show options are specified per manager, and the others are common to all:

```
Style "FvwmIconMan"  NoTitle, Sticky, WindowListSkip, BorderWidth 0
Style "FvwmIconMan"  HandleWidth 0


Key F8 A N SendToModule FvwmIconMan bif select Select, bif focus Focus, \
  gotomanager 0, select, sendcommand WarpToWindow, ret, label Focus, \
  gotobutton focus, select, sendcommand WarpToWindow, ret, label Select, \
  gotobutton prev, select, sendcommand WarpToWindow
Key F9 A N SendToModule FvwmIconMan bif select Select, bif focus Focus, \
  gotomanager 0, select, sendcommand WarpToWindow, ret, label Focus, \
  gotobutton focus, select, sendcommand WarpToWindow, ret, label Select, \
  gotobutton next, select, sendcommand WarpToWindow

*FvwmIconMan: NumManagers 2
*FvwmIconMan: Resolution  page
*FvwmIconMan: Background  steelblue
*FvwmIconMan: Foreground  white
*FvwmIconMan: Font        7x13
*FvwmIconMan: UseWinList  true
*FvwmIconMan: DrawIcons   true
*FvwmIconMan: Shape       true
*FvwmIconMan: FollowFocus true
*FvwmIconMan: Sort        name
*FvwmIconMan: PlainButton          up white steelblue
*FvwmIconMan: SelectButton         down white steelblue
*FvwmIconMan: FocusButton          up white brown
*FvwmIconMan: FocusAndSelectButton down white brown
*FvwmIconMan: TitleButton          raisededge white steelblue
*FvwmIconMan: NoIconAction         "SendToModule FvwmAnimate animate"

*FvwmIconMan: 1 Title           "Emacs windows"
*FvwmIconMan: 1 IconName        "FvwmIconMan: Emacs"
*FvwmIconMan: 1 Format          "%i"
*FvwmIconMan: 1 Show            resource=emacs resource=gemacs
*FvwmIconMan: 1 ManagerGeometry 1x0-400+0
*FvwmIconMan: 1 ButtonGeometry  200x0

*FvwmIconMan: 2 Title           "All windows"
*FvwmIconMan: 2 IconName        "FvwmIconMan: all"
*FvwmIconMan: 2 Format          "%c: %i"
*FvwmIconMan: 2 DontShow        icon=Untitled
*FvwmIconMan: 2 ManagerGeometry 2x4-0+0
*FvwmIconMan: 2 ButtonGeometry  200x0

*FvwmIconMan: transient Geometry 194x100
*FvwmIconMan: transient DontShow icon=Untitled
*FvwmIconMan: transient Action   Mouse 0 A sendcommand select select Iconify

*FvwmIconMan: Action    Mouse   1 N sendcommand Iconify
*FvwmIconMan: Action    Mouse   2 N sendcommand WarpToWindow
*FvwmIconMan: Action    Mouse   3 N sendcommand "Module FvwmIdent FvwmIdent"
*FvwmIconMan: Action    Key     Left  N gotobutton Left, select
*FvwmIconMan: Action    Key     Right N gotobutton Right, select
*FvwmIconMan: Action    Key     Up    N gotobutton Up, select
```

```
*FvwmIconMan: Action    Key     Down  N gotobutton Down, select
*FvwmIconMan: Action    Key     q     N quit
```

UNFINISHED BUSINESS

There is one bug that I know of. A honest to goodness solution to this would be appreciated. When an icon manager is set to grow upwards or leftwards, on some machines it may wander occasionally.

It doesn’t handle windows without resource names as gracefully as it should.

AUTHOR

Brady Montz (bradym@cs.arizona.edu).

THANKS

Thanks to:

David Berson <berson@cs.pitt.edu>,
Gren Klanderman <greg@alphatech.com>,
David Goldberg <dsg@mitre.org>,
Pete Forman <gsez020@compo.bedford.waii.com>,
Neil Moore <amethyst@maxwell.ml.org>,
Josh M. Osborne <stripes@va.pubnix.com,
Adam Rice <wysiwyg@glympton.airtime.co.uk>,
Chris Siebenmann <cks@hawkwind.utcs.toronto.edu>,
Bjorn Victor <victor@delial.docs.uu.se>.

for contributing either code or truly keen ideas.