NAME

olvwm − OPEN LOOK virtual window manager for X11 and OpenWindows

SYNOPSIS

olvwm [*options*]

DESCRIPTION

olvwm is a virtual window manager for the X Window System that implements parts of the OPEN LOOK graphical user interface. **olvwm** differs from **olwm** in that olvwm manages a virtual desktop which is larger than the actual screen. However, **olvwm** is equivalent in all other respects to, and supports all the features of **olwm.** This manual page discusses only the changes between **olvwm** and **olwm,** most of which deal specifically with the virtual desktop. For details on how **olwm** operates, including its host of options, see **olwm(1).**

OPTIONS

There is one new option to **olvwm.**

-allowSynthetic

Allow the receipt of synthetic events. Note that this argument only permits synthetic events to be enabled; whether or not synthetic events are delivered is controlled via the SyntheticEvents resource variable.

TERMINOLOGY

Throughout this manual, the following terms are used:

Screen:

This refers to a specific monitor; essentially, there is one screen for every monitor run by your workstation. Each monitor has a view into a particular desktop. In X terminology, this is the screen component of a particular display.

Desktop:

This refers to the entire area in which windows can reside on a given screen. In non-virtual window managers, a desktop and a screen are the same size. In a virtual window manager, though, a desktop is larger than the screen.

VDM or **Virtual Desktop Manager** is the window which represents visually the entire desktop: it provides a scaled-down version of the entire desktop.

Logical Screen:

If the desktop is running in default mode, it will display a grid, each square of which maps to the size of the monitor. Each square is termed a logical screen.

Current View:

is that part of the desktop which is currently displayed on the screen; this will usually (but need not always) correspond to a logical screen.

Virtual Window:

is a small rectangle displayed in the VDM; every window on the desktop has a corresponding Virtual Window in the VDM.

VIRTUAL DESKTOP MANAGER

When it is started, **olvwm** displays a virtual desktop manager window. The VDM is a reduced display of all windows active on the desktop at any time. The current view is outlined in the virtual desktop manager. The VDM always appears on the screen.

By default, the VDM (and hence the desktop) is divided into a grid; each square of the grid represents a screen size. The dividing lines between each logical screen are represented by dashed lines in the VDM. This division into logical screens is purely informational: if you like, windows can straddle these boundaries, the current view into the desktop can straddle these boundaries, and window can be moved at will between these boundaries. However, by default, most actions in the VDM will keep the current view along these boundary lines.

EVENTS IN THE VDM

The processing of events within the VDM depends on whether the mouse is pointing to the background area of the VDM or to a particular virtual window within the VDM.

EVENTS ON VIRTUAL WINDOWS

Events which occur in a particular virtual window behave just as if they were delivered to the corresponding application's frame. Thus, pressing the MENU button over a virtual window will bring up the frame menu and allow the real (and virtual) windows to be opened, closed, resized, etc. Pressing the SELECT button will select that real (and virtual) window; pressing the ADJUST button will add (or subtract) that window from the selection list. Pressing the FRONT or OPEN keys will raise/lower or open/close the real (and virtual) window.

SELECTing and dragging one or more virtual windows will move the real and virtual windows (just as in **olwm**). Note that if you drag the mouse outside of the VDM, the window can be moved onto to the screen. Conversely, when dragging a window on the screen, if the mouse moves into the VDM, the window will then be moved to the location on the displayed within the VDM. However, if part of the VDM is obscured, you cannot move a window into that part of the VDM.

Note that events are delivered ONLY to the real application's frame. Thus, typing characters or the COPY/CUT/PASTE keys has no effect.

It is possible to drop something onto a virtual window as if it were dropped onto the corresponding application; this allows you to, for example, drag a file from the mailtool on the screen to a file manager on another part of the desktop.

EVENTS ON THE VDM BACKGROUND

Events on the VDM background all work to change the current view, bringing windows which were formerly outside of the current view onto the screen (and moving windows which were on the screen outside of the current view). Keyboard events are described given the default Sun Type-4 keyboard mappings, but you can use xmodmap to set up your particular keyboard.

Keyboard Events:
The arrow keys (and other keys) can be used to shift the current view in the direction specified. See the section on ADDITIONAL KEY BINDINGS below. Keyboard events are also active if they occur on the root window in addition to the VDM background.

SELECT
mouse events: SELECTing on the VDM background allows you to drag the current view within the VDM. If the VirtualGrid resource is set to Visible or Invisible, movement of the visible region will be constrained to logical screen boundaries unless the CTRL button is pressed before and held while dragging the mouse. If the VirtualGrid resource is set to None, movement of the visible region will be unconstrained unless the CTRL button is pressed before and held while dragging the mouse.

DOUBLE CLICK
SELECT events: Double clicking the SELECT button on an area in the VDM background will move the current view to the logical screen containing the point where the mouse was double-clicked.

MENU mouse events: The MENU button brings up a (pinnable) menu which will allow movement based on full screen sizes in the direction indicated.

RESIZING THE VIRTUAL DESKTOP

You can use the resize corners on the VDM to resize the virtual desktop at will. If you make the virtual desktop smaller, windows which might be off the new virtual desktop will NOT be moved (though they will not be lost, either, and you can get them back by resizing the desktop again). Space added or subtracted is always done so from the right and bottom of the desktop (regardless of which resize corner you used).

STICKY WINDOWS

You'll notice that the virtual desktop manager never moves on your screen if you change views into the desktop. That's because the VDM is permanently "sticky."

Windows which are "sticky" never move position on the screen when you change your view into the desktop. To set a particular window as sticky, simply select "Stick" in its frame menu. You may similarly unstick the window via its menu.

[Note: Only base frames--those which can be iconified, as opposed to those which have a pushpin--are eligible to become sticky; other frames inherit the stickyness of their base frames. Thus, for most applications, either all windows are sticky or none of them are. The exception to this is applications which create two or more base frames; all base frames will be originally created as "unsticky" (but see the VirtualSticky resource below).]

CHANGES TO THE WINDOW MENU

The window menu of all windows has all the tools defined in **olwm.** In addition, the menu of base windows has the following command

Stick/Unstick

Affect the stickyness of the particular window. Windows which are sticky will always appear in the same place on the screen no matter which part of the virtual desktop you're looking at. Windows which are not sticky (by default, all windows except the VDM) will move when you change the current view on the virtual desktop.

CHANGES TO ROOT MENU

The following additional keywords can be used in the command field of a root menu menu item:

COLUMNS

This specifies the number of columns you'd like the menu to have. Each column within a menu will be filled in order of the items in the menu. The menu name must appear before this keyword, and the keyword itself must appear before the end of the menu.

STICK_UNSTICK_SELN

This specifies that the sticky state of the selected windows should be toggled.

WINMENU

brings up a pull-right menu containing the name of every window on the current monitor (if **olvwm** is managing multiple monitors, only windows on the current monitor will be shown). Windows which are iconified are preceded with a graphic symbol. Selecting one of these windows causes the view into the desktop to shift so that the selected window is visible; the selected window will be opened if iconic and raised to the top of the stacking order. This behavior can be altered; see *olvwmrc(5)* for more details. You may specify the number of columns in the pull-right menu by using a number after the WINMENU keyword. By default, the order of the menu is alphabetic; see the SortMenuType resource below for more options.

DIRMENU

brings up a pull-right menu containing the name of every file given in the named directory. This allows you to use a local programs directory for a tool repository; whenever tools are added to the directory, users will see the new choices. The directory name may contain environmental variables (e.g. $OPENWINHOME). You may also specify a regular expression as the file part of DIRMENU; e.g. $OPENWINHOME/bin/x* would display only those programs in the openwindows bin directory which begin with an x. Regular expressions do not apply to the directory name; $OPENWINHOME/b*/x* is an invalid DIRMENU entry.

X11 Bitmaps and Pixmaps

can be specified in the menu by specifying a filename enclosed in <> brackets. The filename specified must be in X11 bitmap form, XPM2 form (produced by IconEdit), XPM3 form, or Gif format. You can mix bitmap and string entries within a menu, though perhaps menus look better if you do not.

MOVE_DESKTOP

specifies that **olvwm** should move the view to either the logical screen specified after the MOVE_DESKTOP entry (e.g. MOVE_DESKTOP 3 moves to the third logical screen) or to the logical screen containing the point specified after the MOVE_DESKTOP entry (e.g.

MOVE_DESKTOP 2304 0 also moves to the third logical screen in a default configuration).

Here is a version of the example *olwm* root menu specification: the programs menu will have 2 columns; there will be a way to find particular windows off the "Find" submenu (and that menu will have 3 columns); whatever local tools are available can be found in /usr/local; and the cmdtool option will be displayed as an icon:

```
"My Custom Menu" TITLE

Programs        MENU
Programs              COLUMNS 2
    </usr/openwin/include/X11/bitmaps/terminal>  DEFAULT cmdtool
    "Text Editor"       textedit
    Mail           mailtool
    "File Manager"      filemgr
    Other          MENU
        "Other Tools"       TITLE
        "Shell Tool"        shelltool
        "Icon Editor"       iconedit
        Clock          clock
        "Perf Meter"        DEFAULT perfmeter
    Other          END
       "Local Tools"          DIRMENU      /usr/local
Programs        END PIN

"Repaint Screen"     REFRESH

"Properties ..."     PROPERTIES

"Find"          WINMENU     3

Exit        EXIT
```

MISCELLANEOUS NEW FEATURES

This section describes miscellaneous window manager features which are not directly related to the virtual nature of **olvwm.**

Upon receiving a USR1 signal, **olvwm** will execute a RESTART function just like the restart function available from the root menu.

The root workspace will not be repainted on exit if the PaintWorkspace resource is False.

The **olwm** resource DefaultIconImage will now accept either an X11 Bitmap file, a version 2 or 3 XPM image file or a GIF file. If you use an image file, remember that the DefaultIconMask must still be a X11 Bitmap; generally you will want to create a full mask in that case.

The **olwm** resource WorkspaceBitmap will accept X11 Bitmap files, version 2 or 3 XPM image files, or GIF files. The MaxMapColors resource can restrict the number of colors **olvwm** uses for each GIF file to help with colormap problems, but in general, specifying more than 1 GIF file will lead to colormap problems.

RESOURCES TO CONTROL THE VIRTUAL DESKTOP

See the man page for **olwm** for a complete description of resources. All resources of class OpenWindows and olwm described there are supported by **olvwm.** In addition, for **olvwm** only, resources of instance "olvwm" are also read. You can thus name the following resources as olvwm.<resource>, olwm.<resource>, or as OpenWindows.<resource> depending on your naming preference.

Resources marked with an * may be specified on a screen-by-screen basis; e.g. olvwm.screen0.resource and olvwm.screen1.resource, following the pattern in **olwm.**

VirtualDesktop *(string)******

Specifies the size of the virtual desktop. You can specify this in terms of screen sizes, so that a virtual desktop of 3 screens across and 2 screens high would appear as the string 3x2. You may also specify this in terms of absolute pixel sizes, e.g. 3800x1800, as long as the pixel size is greater than the size of your framebuffer. *Default value: 3x2*

PannerScale *(int)******

Sets the scale in which the virtual desktop manager will be drawn. *Default value: 15.*

VirtualGeometry *(string)******

Specifies the geometry string of the desktop manager in standard X11 format (wxh+x+y). The width and height, if specified, are ignored, but the x and y can be used to specify the initial location of the desktop manager. *Default value: +0+0*

VirtualIconGeometry *(string)******

Specifies the geometry string of the desktop manager icon in standard X11 format (wxh+x+y). The width and height are ignored, but the x and y can be used to specify the initial location of the desktop manager icon. *Default value: +0+0*

VirtualIconic *(Boolean)******

If true, the virtual desk manager will come up iconic. *Default value: False*

AllowMoveIntoDesktop *(Boolean)*

If true, allows windows to be moved from the screen into the VDM and vice versa. Otherwise, moving a window on the screen will only allow it to be placed in the visible screen; and moving a window in the desktop manager will not let it be dragged onto the screen. *Default value: True*

AllowArrowInRoot *(Boolean)*

If true, then key events in the root window will affect the VDM and thus move the view into the desktop (if the appropriate key is pressed). Otherwise, only key events when the VDM has the input event focus will have an effect. *Default value: True*

VirtualFont *(string)******

The font name used to display title of windows within the virtual desktop manager. *Default value: 5x8*

VirtualFontColor *(color specification)******

The color of the title within the virtual windows of the virtual desktop manager. *Default value: Black*

VirtualBackgroundColor *(color specification)******

The color of the background of the virtual desktop manager. *Default value: Based on the frame color of your desktop.*

VirtualForegroundColor *(color specification)******

The color of the virtual windows in the virtual desktop manager. *Default value: The frame color of your desktop.*

VirtualBackgroundMap *(Pixmap filename)******

The name of a file containing a pixmap to use as the background of the virtual desktop manager. This will be drawn with the VirtualBackgroundColor as its background color and the VirtualPixmapColor as its foreground color. The file can be either an X bitmap file, a Z Pixmap file, or a Gif file. Color reduction can be carried out on these images by specifying the MaxMapColors resource. If the file is NOT a bitmap file the virtual desktop window title is changed to be the filename part of the file and the Maximum Size window hint is set to the dimensions of the pixmap. *Default value: unused*

MaxMapColors *(number of colors)***
This integer value specifies the maximum number of colors that olvwm will try and allocate for the VirtualBackgroundMap. If the actual number of colors in the pixmap is greater, the colors are remapped to this number using a 'best match' algorithm (Heckbert Median Cut). *Default value: 200*

VirtualPixmapColor *(color specification)***
See above; this value only applies when the VirtualBackgroundMap has depth 1. *Default value: white*

VirtualSticky *(list of strings)*
The list of windows which should be sticky by default. Like the olwm MinimalDecor, this should be the list windows which should be sticky when they are created. The matching of these strings is done by the first word of the window's WM_NAME (string in its title bar), or by its WM_CLASS class/instance variables. *Default value: Null*

RelativePosition *(Boolean)*
If true, then user-specified window co-ordinates are assumed to be relative to the current view into the desktop; otherwise they are absolute with respect to the desktop. Setting to true will not allow you to bring up windows outside your current view (unless you specify very large or negative numbers), and setting to false will break some programs which save window states in an odd manner. You can defeat this variable on startup of applications by specifying which screen they should start on; see *olvwmrc(5). Default value: True*

VirtualGrid *(Visible, Invisible, or None)***
This controls two aspects of the VDM: If it's visible or invisible, the grid is on (but it's only drawn if it's visible). If it's None, the grid is off. If the grid is on, the VDM may only be resized in logical screen-sized increments. Otherwise, it may resized to arbitrary sizes. If the grid is on, SELECT/dragging the mouse within the VDM will constrain the move to logical screens (unless the CTRL button is held down); if the grid is off, moving the visible screen region with SELECT/drag will be unconstrained (unless the CTRL button is held down). *Default value: Visible*

VirtualGridColor *(color specification)*
If the VirtualGrid is Visible, it will be drawn in this color. *Default value: Same as VirtualFontColor*

VirtualRaiseVDM *(Boolean)*
If this is true, the VDM will always appear on top of the stacking order, never to be obscured by other windows. *Default value: False*

VirtualMoveGroup *(Boolean)*
If true, then whenever a window is moved to a different logical screen, then all followers of that window will also be moved a similar amount: this affects the window's icon and popups. In this way, icons and popups will always appear on the same logical screen as their corresponding base frame. Otherwise, only the selected windows will move when you move them, and related windows could possibly appear on different logical screens in the desktop. The trigger for changing between logical screens is the window's mid-point; all windows in the group will appear on the same logical screen as the mid-point of the moved window. *Default value: True*

UseImages *(UseNone, UseVDM, or UseAll)*
If set to UseNone, then window frame menus and the VDM motion menu will be text-based. If set to UseVDM, the window frame menus will be text-based, but the VDM motion menu will be a graphic menu. If set to UseAll, the window frame menus will be text- and image-based, and the VDM motion menu will be image-based. *Default value: UseVDM*

SortMenuType *(Enum)*
This option controls how windows appearing in the WINMENU command are sorted. If it is set to Alphabetic, then windows will be sorted alphabetically by name. If it is set to Youngest, then windows will be sorted from youngest to oldest. *Default value: Alphabetic*

VirtualDrawSticky *(True)*
If this option is set, sticky windows will have their virtual windows drawn in the VDM just like any other window. If it is false, then sticky windows will not have their virtual windows drawn in the VDM. *Default value: True*

StickyIcons *(Boolean)*
When set, StickyIcons will cause all non-sticky windows to become sticky when iconified. This makes all icons visible on every screen. All windows made sticky when iconified will become unsticky when reopened. *Default value: False*

StickyIconScreen *(Boolean)*
StickyIconScreen, when used in conjunction with StickyIcons, will cause all iconified windows that are made sticky to remember the screen on which they were closed. When one of these windows is reopened the user will be warped back to the screen from which the window was closed and the window is made unsticky. *Default value: False*

FreeIconSlots *(Boolean)*
By default, when an icon is created, it has that icon position forever (unless it is moved manually), even when the window is not iconified. This results in potentially large gaps in the icon area when lots of little programs are run (clocks, load averages, etc) which are rarely iconified. When this resource is true and a window is opened, if the icon position what not set manually the slots are freed. The effect is that iconified windows pack together in the corner specified by IconPlacement. If you move an icon manually, it maintains that position. *Default value: False*

UniqueIconSlots *(Boolean)*
This resource, when set to true, prevents icons from overlapping. If you move an icon into icon grid slot(s) occupied by other icons, those icons move. If those other icons were positioned manually, they are no longer viewed as manually placed (unless, of course, they are subsequently manually placed) *Default value: False*

IconGridHeight *(Integer)*
IconGridWidth *(Integer)* Previously the size of the icon grid was fixed. An icon takes up a number of grid slots, depending on it's size, and is aligned with the lower left corner of its lower left grid slot. Changing the size of the grid changes how icons arrange themselves. *Default value: 13 for each*

ADDITIONAL MISCELLANEOUS RESOURCES

These resources are not related specifically to the VDM, but are included as new features within **olvwm.**

InputFocusColor *(color specification)******
Specifying this attribute means that the frame of the window which currently has the input focus should be drawn in the color specified instead of in the normal frame color. The appearance of the title bar (with lines or indentation, depending on your input focus mode) will remain unchanged. *Default value: None (Use standard frame color)*

RaiseOnMove *(Boolean)*
If this is True, then whenever a window is moved, it is also raised to the top of the stacking order. *Default value: False*

AutoShowRootMenu *(Boolean)*
If true, the root menu will come up pinned at start up. *Default value: False*

AutoRootMenuX *(int)*
If AutoShowRootMenu is true, this is the X location where the menu will come up. *Default value: 0*

AutoRootMenuY *(int)*
If AutoShowRootMenu is true, this is the Y location where the menu will come up. *Default value: 0*

FullSizeZoomX *(Boolean)*
If this is set to True, then selecting Full Size from the window menu will zoom the window horizontally as well as vertically. *Default value: False*

NoDecor *(list of strings)*
Like Minimal Decor, this attribute takes a list of windows on which to operate; these windows should not be decorated at all. There is a slight difference between this attribute and setting the override_redirect flag on a window: the latter causes *olvwm* to ignore completely the window, while this attribute does not adorn the window or give it a menu, but still allows it to be manipulated via the WINMENU and Hot Key actions. Normal olwm keys are ignored by these windows, however.

ResizeMoveGeometry *(X geometry string or the word center)*
This resource only has an effect if ShowMoveGeometry and/or ShowResizeGeometry is set to True. This string specifies the location of the geometry box when it is displayed; it should be given as an X geometry string (wxh+x+y) although the width and height are ignored if specified. If you want the box to be centered regardless of your monitor resolution, use the string center instead of a geometry string. *Default value: +0+0*

PaintWorkspace *(Boolean)*
If true, **olvwm** will paint the root workspace with the color of the WorkspaceColor resource. If False, **olvwm** will not change root workspace. *Default value: True*

PointerWorkspace *(Boolean)*
If true, **olvwm** will change the cursor when it is over window frames and other olvwm-owned windows. If False, **olvwm** will not change the cursor. This is useful in conjunction with *xsetroot -cursor* if you want the specified cursor to be active over frames. *Default value: True*

SaveWorkspaceCmd *(String)*
If set, this is the command that will be run when the SAVE_WORKSPACE command is selected off the main menu. Otherwise, the default command is run. *Default value: owplaces -silent -multi -script -tw -output $HOME/.openwin-init*

SyntheticEvents *(boolean)*
This resource controls the delivery of synthetic events. This resource is only valid if the -allowSynthetic argument was given to olvwm on startup; if that argument was present and this resource is set to True, then olvwm will process synthetic events. Otherwise, synthetic events are ignored. *Default value: False*

ParentScreenPopup *(Boolean)*
If true the popups are placed on the same screen as their parent even if the parent is not currently visible. Otherwise popups always appear on the current screen. A suggested method is to place the following lines at the end of your .xinitrc file (before the final wait command):

```
xrdb -merge <<'EOF'
olvwm.ParentScreenPopup: False
EOF
```

In this way, your session will initialize with all popups with their parents but subsequenct popups will appear on the current screen. *Default value: True*

CURSOR RESOURCES

These set of resources change the default cursors the window manager gives certain classes it creates, such as resize buttons, close buttons, frames, icons, etc. The <cursor string> is described as follows:

<cursor font name> <cursor index> [<fore color> [<back color>]]

where options enclosed in []'s are optional. If the <back color> is left out, the default back color (which, in this version of OLVWM, is white) is used. If the <fore color> is left out, then the default cursor (foreground) color is used, which is black by default, unless set by the cursorColor resource (see below). Note that to supply the <back color>, you must supply the <fore color>. Also, if the color name you provide is invalid, OLVWM will return to the default colors, as described above. As per usual with resources, no errors are reported. The cursor font is assumed to be encoded so that the valid mask for the cursor follows the cursor index in the font; this is the usual behavior for font files.

There is an alternate form of <cursor string>:

<cursor name> [<fore color> [<back color>]]

where <cursor name> is either the name of the X cursor as defined in cursorfont.h (eg: XC_X_cursor, XC_cross, etc.), or the name of an OPEN LOOK cursor as defined in olcursor.h (e.g.: OLC_basic, OLC_busy).

The valid <cursor>s are:

CursorBasic
Specify the root window cursor. *Default: Basic Pointer (arrow pointing NW)*

CursorMove
Specify the cursor when a window is moved. *Default: Basic Pointer*

CursorBusy
Specify the cursor for when a window is "busy". *Default: Busy Pointer (pocket watch)*

CursorIcon
Specify cursor for the icon windows. *Default: Basic Pointer*

CursorResize
Specify cursor for the resize buttons and when window is being "rubber-band" resized. *Default: Bull's Eye*

CursorMenu
Specify cursor for menu windows. *Default: Basic Pointer*

CursorTarget
Specify cursor for window frames. *Default: Basic Pointer*

CursorCloseUp
Specify cursor for close button, when NOT pressed (the button is UP). *Default: Basic Pointer*

CursorCloseDown
Specify cursor for close button, when pressed (the button is DOWN). *Default: Basic Pointer*

CursorSpecialResize *<Bool> [<fg corner> <bg corner> <fg move> <bg move>]*
This is a special Cursor resource which turns on specialized resize cursors. If <Bool> is false, then the cursor for the resize button decorations and the "rubber-band" window are all specified by "Cursor.Resize" as discussed above. If <Bool> is true, then the following modifications are turned on: When the cursor is within the resize button decorations on a frame, the cursor becomes a corner of the appropriate orientation. When the button is selected and the window is being "rubber-band" resized, the cursor becomes the appropriate corner with an appropriately oriented arrow. The colors that you can specify are limited to prevent an obscene number of resources. You can

specify the colors (foreground and background) of the cursors when placed in the resize decoration without the mouse button down (described above as <fg corner> and <bg corner>), and you can specify the colors of the cursors when the mouse button is down the window is being resized (described above as <fg move> and <bg move>). Note that you must specify either all four colors or no colors.

ADDITIONAL KEY BINDINGS

Like **olwm, olvwm** uses key bindings for certain actions: all actions specified in **olwm** as well as an additional set of actions to control the view into the desktop. In the following list, the key mentioned in parentheses is the default binding for the given action; items marked with a plus sign '+' are items which exist in **olwm;** they are included here to describe what action they have when they occur on the root window or within the VDM. All other bindings are specific to **olvwm;** those items marked with an asterisk '*' involve keyboard grabs. Bindings which involve a keyboard grab (those with an asterisk) are always active, no matter where the input focus is; otherwise the action occurs only if the root window or the VDM has the input focus.

It's possible to change each of these bindings, see the **olwm** man page for more details.

GrabVirtualKeys " (Boolean)"
: This resource controls whether the keys below marked as involving grabs will be active or not. If this resource is set to False, then none of the bindings marked below with an asterisk will ever be active. *Default value: True*

Up *(up-arrow)* +
: Move the view up one screen.

JumpUp *(up-arrow+Ctrl)* +
: Move the view up ten screens.

HalfUp *(up-arrow+Shift)*
: Move the view up one-half screen.

VirtualUp *(up-arrow+Meta)* *
: Move the view up one screen.

VirtualJumpUp *(up-arrow+Ctrl+Meta)* *
: Move the view up ten screens.

VirtualHalfUp *(up-arrow+Shift+Meta)* *
: Move the view up one-half screen.

Down *(down-arrow)* +
: Move the view down one screen.

JumpDown *(down-arrow+Ctrl)* +
: Move the view down ten screens.

HalfDown *(down-arrow+Shift)*
: Move the view down one-half screen.

VirtualDown *(down-arrow+Meta)* *
: Move the view down one screen.

VirtualJumpDown *(down-arrow+Ctrl+Meta)* *
: Move the view down ten screens.

VirtualHalfDown *(down-arrow+Shift+Meta)* *
: Move the view down one-half screen.

Left *(left-arrow)* +
: Move the view left one screen.

JumpLeft *(left-arrow+Ctrl)* +
    Move the view left ten screens.

HalfLeft *(left-arrow+Shift)*
    Move the view left one-half screen.

VirtualLeft *(left-arrow+Meta)* *
    Move the view left one screen.

VirtualJumpLeft *(left-arrow+Ctrl+Meta)* *
    Move the view left ten screens.

VirtualHalfLeft *(left-arrow+Shift+Meta)* *
    Move the view left one-half screen.

Right *(right-arrow)* +
    Move the view right one screen.

JumpRight *(right-arrow+Ctrl)* +
    Move the view right ten screens.

HalfRight *(right-arrow+Shift)*
    Move the view right one-half screen.

VirtualRight *(right-arrow+Meta)* *
    Move the view right one screen.

VirtualJumpRight *(right-arrow+Ctrl+Meta)* *
    Move the view right ten screens.

VirtualHalfRight *(right-arrow+Shift+Meta)* *
    Move the view right one-half screen.

GoHome *(R11)*
    Move the view to the upper left corner.

VirtualHome *(R11+Meta)* *
    Move the view to the upper left corner.

UpLeft *(Home, R7)* +
    Move the view NorthWest one screen.

JumpUpLeft *(R7+Ctrl)* +
    Move the view NorthWest ten screens.

HalfUpLeft *(R7+Shift)*
    Move the view NorthWest one-half screen.

VirtualUpLeft *(R7+Meta)* *
    Move the view NorthWest one screen.

VirtualJumpUpLeft *(R7+Ctrl+Meta)* *
    Move the view NorthWest ten screens.

VirtualHalfUpLeft *(R7+Shift+Meta)* *
    Move the view NorthWest one-half screen.

DownLeft *(End, R13)* +
    Move the view SouthWest one screen.

JumpDownLeft *(R13+Ctrl)* +
Move the view SouthWest ten screens.

HalfDownLeft *(R13+Shift)*
Move the view SouthWest one-half screen.

VirtualDownLeft *(R13+Meta)* *
Move the view SouthWest one screen.

VirtualJumpDownLeft *(R13+Ctrl+Meta)* *
Move the view SouthWest ten screens.

VirtualHalfDownLeft *(R13+Shift+Meta)* *
Move the view SouthWest one-half screen.

UpRight *(R9)*
Move the view NorthEast one screen.

JumpUpRight *(R9+Ctrl)*
Move the view NorthEast ten screens.

HalfUpRight *(R9+Shift)*
Move the view NorthEast one-half screen.

VirtualUpRight *(R9+Meta)* *
Move the view NorthEast one screen.

VirtualJumpUpRight *(R9+Ctrl+Meta)* *
Move the view NorthEast ten screens.

VirtualHalfUpRight *(R9+Shift+Meta)* *
Move the view NorthEast one-half screen.

DownRight *(R15, PgDn)*
Move the view SouthEast one screen.

JumpDownRight *(R15+Ctrl)*
Move the view SouthEast ten screens.

HalfDownRight *(R15+Shift)*
Move the view SouthEast one-half screen.

VirtualDownRight *(R15+Meta)* *
Move the view SouthEast one screen.

VirtualJumpDownRight *(R15+Ctrl+Meta)* *
Move the view SouthEast ten screens.

VirtualHalfDownRight *(R15+Shift+Meta)* *
Move the view SouthEast one-half screen.

Screen1 *(F1)*
Move to the first logical screen in the desktop.

VirtualScreen1 *(F1+Meta)* *
Move to the first logical screen in the desktop.

Screen2 *(F2)*
Move to the second logical screen in the desktop.

VirtualScreen2 *(F2+Meta)* *
Move to the second logical screen in the desktop.

Screen3 *(F3)*
: Move to the third logical screen in the desktop.

VirtualScreen3 *(F3+Meta)* *
: Move to the third logical screen in the desktop.

Screen4 *(F4)*
: Move to the fourth logical screen in the desktop.

VirtualScreen4 *(F4+Meta)* *
: Move to the fourth logical screen in the desktop.

Screen5 *(F5)*
: Move to the fifth logical screen in the desktop.

VirtualScreen5 *(F5+Meta)* *
: Move to the fifth logical screen in the desktop.

Screen6 *(F6)*
: Move to the sixth logical screen in the desktop.

VirtualScreen6 *(F6+Meta)* *
: Move to the sixth logical screen in the desktop.

Screen7 *(F7)*
: Move to the seventh logical screen in the desktop.

VirtualScreen7 *(F7+Meta)* *
: Move to the seventh logical screen in the desktop.

Screen8 *(F8)*
: Move to the eighth logical screen in the desktop.

VirtualScreen8 *(F8+Meta)* *
: Move to the eighth logical screen in the desktop.

Screen9 *(F9)*
: Move to the ninth logical screen in the desktop.

VirtualScreen9 *(F9+Meta)* *
: Move to the ninth logical screen in the desktop.

Screen10 *(F10)*
: Move to the tenth logical screen in the desktop.

VirtualScreen10 *(F10+Meta)* *
: Move to the tenth logical screen in the desktop.

SaveWorkspace *(F10+Meta+Alt+Shift)* *
: Execute a Save Workspace command

ToggleDragWindows *(F6+Alt+Shift)* *
: Toggles the setting of the DragWindow resource

ToggleMoveGroups *(g+Alt)* *
: Toggles the setting of the VirtualMoveGroup resource

ToggleSizeFullZoomX *(F3+Alt+Shift)* *
: Toggles the setting of the FullSizeZoomX resource

ToggleSticky *(s+Alt)* *
: Toggles the sticky state of the window with input focus.

ADVANCED USAGE

olvwm will read a resource file ($HOME/.olvwmrc) for special customizations. These customizations allow the following:

HOT KEYS

You can specify that when a certain key (or, more likely, key in combination with modifiers) is pressed that certain actions are performed: you can warp to a particular application, open, raise, or close applications, execute applications, and quit applications.

SCREEN BINDINGS

You can specify that certain applications will always start on a particular logical screen.

MENU OPTIONS

You can alter the behaviour of WINMENU selections on a particular window.

Please see *olvwmrc(5)* for details of these options.

COPYRIGHTS

The X Window system is a trademark of the Massachusetts Institute of Technology.
OPEN LOOK is a trademark of AT&T.
OpenWindows is a trademark of Sun Microsystems, Inc.
Portions (c) Copyright 1989-1991 Sun Microsystems, Inc. Sun design patents pending in the U.S. and foreign countries. OPEN LOOK is a trademark of AT&T. Used by written permission of the owners.
Portions (c) Copyright Bigelow & Holmes 1986, 1985. Lucida is a registered trademark of Bigelow & Holmes. Permission to use the Lucida trademark is hereby granted only in association with the images and fonts described in this file.
Portions may be (c) 1990 Solbourne Computers.
Portions not covered under the above copyrights are (c) 1991 Scott Oaks.
Please see the LEGAL_NOTICES file for full disclosure of copyright information.

CREDITS

Most of the code in this application comes from release 3.0 of olwm, authored at Sun Microsystems and distributed in the contrib section of MIT's X11R5. [Note that nothing in xview3 or release 3.0 requires R5; they will happily run on R4 and exist in R5 simply because that's when they were released.]
The virtual desktop section was authored by Scott Oaks, who is responsible for its maintenance. Please direct any comments to scott.oaks@sun.com. This code is not supported by Sun Microsystems in any way.
Notwithstanding the above, the staff at Sun Microsystems, and especially Stuart Marks, deserve credit as original author of olwm for most of the work contained in this application.
Innumerable people have made fixes and enhancements to olvwm, and I'm very bad at listing them all (but drop me a note and I'll do better in a future release :-)). Thus, the following list is by no means comprehensive, but these people have all provided something to olvwm: Tor Arntsen, Paul Andrews, Rich Berlin, David Boyd, David Chase, John Durand, Alexander Dupuy, Paul Eggert, Eric Fifer, Duane Gibson, Jimmy Guggemos, Orest Jarosiewicz, Russel Kegley, Murray Laing, Tom McConnell, Blair MacIntyre, Chuck Musciano, Matthias Pfuetzner, David Plotkin, Harry Protoolis, Nir Samburski, Andreas Stolcke, Hinrich Tobaben, Larry Virden, Ken West, and Ron Winacott.