Customizing the Motif Window Manager

This chapter describes how to set some of the basic **mwm** (Motif Window Manger) resources:

- Specifying colors for client window frames, icons, menus, and mattes
- Specifying window decorations
- Specifying the size and placement of windows and icons
- Specifying fonts
- Specifying input focus
- Specifying **mwm** messages using the message catalog

There are three types of **mwm** resources:

Component appearance resources

These resources specify appearance attributes of window manager components such as menus and icons.

Specific appearance and behavior resources

These resources are used to specify **mwm** appearance and behavior; for example, colormap and keyboard input focus policies. They are not set separately for different **mwm** components.

Client−specific resources

These **mwm** resources can be set for a particular client window or class of client windows. They specify client−specific icon and client window frame appearance and behavior.

For example, you could specify the following resources for **mwm** in your `.Xdefaults` file:

```
Mwm*background: LightBlue
Mwm*foreground: Black
Mwm*activeBackground: Blue
Mwm*icon*foreground: DarkSlateBlue

Mwm*keyboardFocusPolicy: pointer
Mwm*focusAutoRaise: true

Mwm*UseIconBox: true
Mwm*iconBoxGeometry: 3x2
Mwm*my_application*iconImage: ~/my.bitmap
```

Notice that the class name for **mwm** is *Mwm*.

Specifying Colors

Color resources for the window manager include a background color and a foreground color, top and bottom shadow colors, and pixmaps for shading. Names for colors vary from system to system and can also depend on locale. (See your system documentation for a list of valid color names on your system.)

Set the resources for color using the following general format:

```
Mwm*resource: color
```

For example, you might have the following specifications in your `.Xdefaults` file:

```
Mwm*background: LightBlue
Mwm*foreground: Black
```

Coloring Client Window Frames

The Motif Window Manager provides resources for specifying colors for the window borders of the currently active window and the inactive windows (see Table 5−1).

Table 1 Color Resources and What They Color

Inactive Window and Icon Resources	***Active Window and Icon Resources***	***Area Covered***
foreground	*activeForeground*	Foreground areas (text)
background	*activeBackground*	Background areas
topShadowColor	*activeTopShadowColor*	Top and left 3−D bevels
bottomShadowColor	*activeBottomShadowColor*	Bottom and right 3−D bevels

You should pick contrasting color schemes to make the active window readily distinguishable from its inactive counterparts. For example, if you would like the foreground and background of inactive window frames to be the reverse of the foreground and background of the active window frame, using the colors *Wheat* and *IndianRed*, you would specify these resources in your `.Xdefaults` file:

```
Mwm*background: IndianRed
Mwm*foreground: Wheat
Mwm*activeBackground: Wheat
Mwm*activeForeground: IndianRed
```

The default values for **mwm** color resources are based on the visual type of the screen, such as monochrome or 8−bit pseudocolor, and the values given to related resources. Table 5−2 shows the default values for a color display.

Table 2 Default Values for Appearance on a Color Display

Resource Specification	***Resource Value***
Mwm*activeBackground	*CadetBlue*
Mwm*activeBackgroundPixmap	**NULL**
Mwm*activeBottomShadowPixmap	**NULL**
Mwm*activeTopShadowPixmap	**NULL**
Mwm*background	*LightGrey*
Mwm*backgroundPixmap	**NULL**
Mwm*bottomShadowPixmap	**NULL**
Mwm*topShadowPixmap	**NULL**

Specifying the value of the *background* resource automatically generates default colors for the top and bottom shadow color resources. A **NULL** pixmap means that the color is solid, not patterned. The Motif Window Manager uses the following rules for generating default values for color resources:

- A top shadow color is generated by proportionally lightening the associated background color.
- A bottom shadow color is generated by proportionally darkening the associated background color.
- The foreground color is set to black or white, depending on the background color.

Table 5−3 indicates the default values for the appearance resources on a monochrome display.

Table 3 Default Values for Appearance on a Monochrome Display

Resource Specification	***Resource Value***
Mwm*activeBackground	*White*
Mwm*activeBackgroundPixmap	*50_foreground*
Mwm*activeBottomShadowPixmap	*foreground*
Mwm*activeTopShadowPixmap	*50_foreground*
Mwm*background	*White*
Mwm*backgroundPixmap	*25_foreground*
Mwm*bottomShadowColor	*Black*
Mwm*bottomShadowPixmap	*foreground*
Mwm*topShadowColor	*White*
Mwm*topShadowPixmap	*50_foreground*
Mwm*foreground	*Black*

Using Window Frame Pixmaps

Using a pixmap is a way of creating shades of colors. Pixmaps are drawn in the foreground color. The pixmap is composed of tiles that provide a surface pattern or a visual texture. The concept is analogous to using ceramic tiles to make a pattern or texture on a floor or countertop.

Generally, the fewer colors a display produces, the more important the pixmap resources become. A pixmap provides a way to mix foreground and background colors into a variety of color patterns. For example, with a monochrome display, you can use the pixmap resource to color window frame elements with shades of gray to achieve a 3−dimensional look.

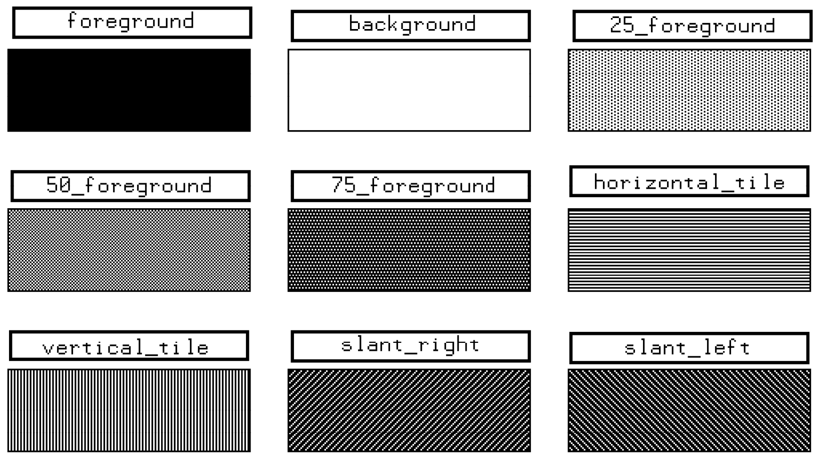
There are **mwm** pixmap resources for creating a pattern for the frame background and bevels of both inactive and active windows (see Table 5 −4).

Table 4 Using Pixmaps for Window Frames

Use this resource...	***To pattern these elements...***
backgroundPixmap	Background for inactive frames
bottomShadowPixmap	Right and bottom bevels of inactive frames
topShadowPixmap	Left and upper bevels of inactive frames
activeBackgroundPixmap	Background of the active frame
activeBottomShadowPixmap	Right and lower bevels of the active frame
activeTopShadowPixmap	Left and upper bevels of the active frame

Figure 5−1 illustrates the various bitmaps that are defined in the Motif environment.

Figure 1 Illustrations of Valid Bitmap Values



To obtain the bitmap styles illustrated in the previous figure, use the values listed in Table 5−5.

Table 5 Valid Pixmap Values

To pattern an element in this manner...	***Use th value..***
The foreground color	*foregr*
The background color	*backgr*
A mix of 25% foreground to 75% background	*25_for*
A mix of 50% foreground to 50% background	*50_for*
A mix of 75% foreground to 25% background	*75_for*
In horizontal lines alternating between the foreground and background color	*horizo*
In vertical lines alternating between the foreground and background color	*vertic*
In diagonal lines slanting to the right alternating between the foreground and background color	*slant_*
In diagonal lines slanting to the left alternating between the foreground and background color	*slant_*

The *cleanText* resource makes text easier to read on monochrome systems when a *backgroundPixmap* is specified. This resource controls the display of window manager text in the title area and in the move/resize feedback window. If the value is set to **True**, the text is drawn with a clear background. Only the background in the area immediately around the text is cleared. If the value is set to **False**, the text is drawn directly on top of the existing background.

Coloring Icons

Icon window frame elements are colored with the same resources as normalized window frame elements. The image part of an icon can be displayed with client−specific colors (see Table 5−6).

Table 6 Coloring Icon Images

To color this...	***Use this***
Icon image background	*iconImag*
Left and upper 3−D bevels	*iconImag*
Right and lower 3−D bevels	*iconImag*
Icon image foreground	*iconImag*

There are two pixmap resources available for use in shading icon images: *iconImageTopShadowPixmap* and *iconImageBottomShadowPixmap*. The default value for *iconImageTopShadowPixmap* is the icon top shadow pixmap, which is specified by **Mwm*icon*topShadowPixmap**. The default value for *iconImageBottomShadowPixmap* is the icon bottom shadow pixmap, which is specified by **Mwm*icon*bottomShadowPixmap**.

Resource specifications that color icons can have any of the following formats.

To color all clients regardless of class, the syntax is

```
Mwm*icon*resource : color
```

For example, the following specification in a resource file ensures that all icon backgrounds are the same color:

```
Mwm*icon*background: CadetBlue
```

To color specific classes of client icons, the syntax is

```
Mwm*icon*Clientclass*resource  : color
```

The colors specified with this resource specification take precedence over any other specification of this resource for this class of clients. For example, the following specification ensures that the icon background for **xterm** clients is the color *IndianRed* rather than *CadetBlue*:

```
Mwm*icon*Xterm*background: IndianRed
```

To color any client with an unknown class, the syntax is

```
Mwm*icon*default*resource : color
```

Coloring Menus

The color resources for the **mwm** Window Menu and Root Menu are the same as those for inactive windows and icons (see Table 5−1). Menus that appear in application programs are not affected by the values set for these resources. Default menu colors are determined by the type of display.

Resource specifications that color Menus can have any the following formats.

To color all clients regardless of class, the syntax is

```
Mwm*menu*resource : color
```

For example, the following specification in a resource file ensures that all Menu backgrounds are the same color:

```
Mwm*menu*background: CadetBlue
```

To color specific classes of clients, the syntax is

```
Mwm*menu*Clientclass*resource : color
```

The colors specified with this resource specification take precedence over any other specification for this resource for this class of clients.

To color any client with an unknown class, the syntax is

```
Mwm*menu*default*resource : color
```

You can also specify a color for a Menu with a specific name using the following syntax:

```
Mwm*menu*menuname*resource : value
```

For example, the following specification in a resource file sets the color of *my_menu*:

```
Mwm*menu*my_menu*background: SlateBlue
```

Coloring Mattes

A matte is a 3−dimensional border between the client's window area and the window frame. A matte can give an individual client, or class of clients, a distinct appearance. To configure a matte, you need to give the *matteWidth* resource a positive value. The *matteWidth* resource defines the width of the matte between the client and the window frame. The width is specified in pixels. The default value of 0 (zero) disables the matte.

For example, to specify a matte of 10 pixels around all **xload** windows, use the following specification in your `.Xdefaults` file:

```
Mwm*XLoad*matteWidth: 10
```

Matte resources use the same wording as window frame resources, but begin with the term *matte* (see Table 5−7).

Table 7 Matte Resources and What They Color

Matte Resource	***Area Colored***
matteBackground	Background areas
matteTopShadowColor	Top and left 3−D bevels
matteBottomShadowColor	Bottom and right 3−D bevels
matteForeground	Foreground areas

As with frame colors, the fewer colors a display can produce, the more value there is in using pixmap resources for mattes (see Table 5−8).

Table 8 Resources for Using Pixmaps with Mattes

Use this resource...	***To pattern these elements...***
matteBottomShadowPixmap	Right and lower bevels of mattes
matteTopShadowPixmap	Left and upper bevels of mattes

Resource specifications for coloring mattes can have any of the following formats.

To matte all clients regardless of class, the syntax is

```
Mwm*matteResource : value
```

For example, to create a 10−pixel wide yellow matte for every client window, use the following specification in a resource file:

```
Mwm*matteWidth: 10
Mwm*matteBackground: Yellow
```

To matte specific classes of clients, the syntax is

```
Mwm*Clientclass.matteResource : value
```

To matte any client of an unknown class, the syntax is

```
Mwm*default*matteResource : value
```

Specifying Window Decorations

Window decorations include a border, a maximize button, a minimize button, a Window Menu button, a title bar, and resize handles. Table 5−9 shows the values that may be used to set the window decorations for a window.

Table 1 Valid Window Frame Elements

Frame Element	***Description***
all	Includes all decoration elements (default)
border	Window border
maximize	Maximize button (includes title bar)
minimize	Minimize button (includes title bar)
none	No decorations
resizeh	Resize handles (includes border)
menu	Window Menu button (includes title bar)
title	Title bar (includes border)

For some applications, the full complement of window decorations may not be desirable. For example, a clock may not need resize handles. The Motif Window Manager has two resources for such situations: *clientDecoration* and *transientDecoration*.

The *clientDecoration* resource allows the user to choose how much decoration to put around each client. The default value is *all*, meaning that windows include all decorations unless the application removes one or more of them.

The *transientDecoration* resource allows you to choose the decorations around each transient window. A *transient window* is a relatively short −lived window, such as a DialogBox. The default value for this resource is *menu title resizeh*, meaning that transient windows have a title bar with a Window Menu button and resize borders. Even if a decoration is specified by the *transientDecoration* resource, **mwm** does not put it around a transient window unless that decoration is also specified by the *clientDecoration* resource.

The *clientDecoration* and *transientDecoration* resources can have more than one value specified at a time:

- If the first value in the list is preceded by nothing or by a + (plus sign), the window manager starts with no frame and assumes that the list contains those elements to be added.
- If the list begins with a − (minus sign), the window manager starts with a complete frame and assumes that the list contains elements to be removed from the frame.

The *clientDecoration* resource can have any of the following formats.

To add or remove elements from all classes of clients, the syntax is

```
    Mwm*clientDecoration: value
```

For example, remove the maximize button from all windows with the following specification in a resource file:

```
    Mwm*clientDecoration: −maximize
```

To add or remove elements from specific classes of clients, the syntax is

```
    Mwm*Clientclass .clientDecoration: value
```

For example, to remove the resize handles and the maximize button from all clocks displayed on the screen, use the following specification:

```
    Mwm*XClock.clientDecoration: −resizeh −maximize
```

To add or remove elements from any client with an unknown class, the syntax is

```
    Mwm*defaults*clientDecoration: value
```

The *transientDecoration* resource has the following syntax:

```
    Mwm*transientDecoration: value
```

For example, to remove the menu button from all transient windows, use the following specification in a resource file:

```
    Mwm*transientDecoration: title resizeh
```

The *iconDecoration* resource indicates the parts of an icon that are displayed (see Table 5−10).

Table 2 Valid Icon Elements

Icon Element	***Description***
label	The icon’s label, which may be truncated
image	The icon’s image
activelabel	The label of an active is not truncated

When you are using an icon box, the default value for *iconDecoration* is *label image.* Without an icon box, the default value of the resource is *activelabel label image.* For example, use the following specification to eliminate the label part of icons:

```
Mwm*iconDecoration: image
```

For window decoration, the Motif Window Manager also has the *frameStyle* resource, which lets you control the look of the window and its border. Assigning a value of *WmRECESSED* to this resource makes the window appear to be recessed from its border. Assigning a value of *WmSLAB* shows a flat window and border.

Sizing Windows

You can control the size of a window with specifications that match the following format:

```
Mwm*sizeResource   :  value
```

For most applications, sizes are specified in pixels, although some applications use units that make sense for the application. For example, terminal windows are sized in characters and lines rather than in pixels.

The *frameBorderWidth* resource specifies the width of a client window frame border with shadow elements, but without resize handles. The default value, usually about 5 pixels, is based on the size and resolution of the screen.

The *limitResize* resource controls the ability to enlarge a window beyond the client's maximized size. The default value of **True** limits a window's size to no greater than the maximum size specified by the *maximumClientSize* resource, or the default maximum size assigned by **mwm**. The value of **False** allows a window to be resized to any size.

The *maximumClientSize* resource controls the maximum size of a maximized client. The value of this resource is specified either as width by height, interpreted in terms of the units that the client uses, or with the values *vertical* or *horizontal,* which causes the *Maximize* operation to resize the window only in the specified direction. If this resource is not specified, the size of the screen is the default value.

The *maximumMaximumSize* resource controls the maximum size of a client window as set by the client. The dimensions are given in pixels. The default value of this resource is twice the screen width and height.

The *resizeBorderWidth* resource specifies the width of a client window frame border with resize handles and shadow elements. The default value, usually about 10 pixels, is based on the size and resolution of the screen.

The *resizeCursors* resource indicates whether the resize cursors are displayed when the pointer is in the window resize border. The default value **True** causes the appropriate resize cursor to appear when the pointer enters a resize handle in the window frame. The value of **False** prevents resize cursors from being displayed.

Controlling Window Placement

You can control the initial placement of client windows with specifications that use the following format:

```
Mwm*resource : value
```

Some of the resources for window placement are described in the following text.

The *clientAutoPlace* resource determines the position of a window when the window has not been given a specific position. The default value of **True** positions a window with the top left corner of the frame offset horizontally and vertically. The value of **False** causes the currently configured position of the window to be used. In either case, **mwm** attempts to place the window so the entire window appears within the boundaries of the screen.

The *interactivePlacement* resource controls the initial placement of new windows on the screen. The value of **True** changes the shape of the pointer to an upper−left−corner bracket before a new window is displayed, so that you can choose a position for the window. When the default value of **False** is used, the window is placed according to its initial configuration attributes or the values of other **mwm** resources such as *clientAutoPlace.*

The *moveThreshold* resource controls the sensitivity of dragging operations. The value of the *moveThreshold* resource is the number of pixels that the pointer must be moved with a button pressed before a drag operation is initiated. This resource is used to prevent window or icon movement when you unintentionally move the pointer during a click or double−click action. The default value is 4 pixels.

The *positionIsFrame* resource determines how client window position information is interpreted. When the default value **True** is used, the position information is relative to the position of the window frame. When the value is **False**, the position information is relative to the position of the client window itself.

The *positionOnScreen* resource controls clipping of new windows by screen edges. The default value **True** causes a window to be placed, if possible, so that it is not clipped. If clipping cannot be avoided, a window is placed so that at least the upper left corner of the window is on the screen. The value of **False** causes a window to be placed at the requested position even if it is totally off the screen. The *iconPinned* and *clientPinned* resources prevent windows from being moved when one of the window manager's virtual desktop capabilities, such as interactive panning, is in use. When the default value **False** is used, the icon or client window is panned whenever the root window is panned. If the value is **True**, the icon or client window will not be moved during panning operations.

The *showFeedback* resource controls when feedback information is displayed. It controls both window position and size feedback during move or resize operations and initial client placement. It also controls window manager DialogBoxes (see Table 5−11).

Table 1 Feedback Options

Name	***Description***
all	Shows all feedback (default value)
behavior	Confirms behavior switch
kill	Confirms on receipt of **KILL** signal
move	Shows position during a move
none	Shows no feedback
placement	Shows position and size during initial placement
quit	Confirms quitting **mwm**
resize	Shows size during a resize operation
restart	Confirms restarting **mwm**

The value for this resource is a list of names of the feedback options to be enabled or disabled; the names must be separated by a space. If an option is preceded by a − (minus sign), that option is excluded from the list. The sign of the first item in the list determines the initial set of options. If the sign of the first option is a − (minus sign), **mwm** assumes all options are present and starts subtracting from that set. If the sign of the first option is a + (plus sign) or not specified, **mwm** starts with no options and builds a list from the resource specification.

For example, you could use the following specification in your `.Xdefaults` file:

```
Mwm*showFeedback: placement resize behavior restart
```

This specification provides feedback for initial client placement and resize and enables DialogBoxes to confirm the restart and set behavior functions. It disables feedback for move operations.

To specify the maximum client size for specific classes of clients, the syntax is

```
Mwm*Clientclass .maximumClientSize: width×height
```

For example, to specify that **xload** clients should be maximized to no more than one sixty−fourth of the size of a 1024×768 display, use the following specification:

```
     Mwm*XLoad.maximumClientSize: 128×96
```

To specify the maximum client size for any client with an unknown class, the syntax is

```
     Mwm*defaults*maximumClientSize: width×height
```

Sizing Icons

Each icon image has maximum and minimum default sizes as well as maximum and minimum allowable sizes. The following two resources control icon image size:

- The *iconImageMaximum* resource limits the maximum size of an icon image. The largest value allowed is 128×128 pixels. The default value is 50×50 pixels.
- The *iconImageMinimum* resource limits the minimum size of an icon image. The smallest value allowed is 16×16 pixels and is also the default value.

When calculating limits for image size, remember that the width of an icon is the image width plus the icon frame plus the space between icons. The height of an icon is the image height plus the icon frame plus the space between icons. The amount of icon decoration and the size of font used in the icon label also affects the height of the icon.

The window manager sizes an icon depending on the size of the image in relation to the specified maximum and minimum sizes (see Table 5−12).

Table 1 Icon Size Affects Treatment of an Icon

If an icon image is...	***The window manager.***
Smaller than the minimum size	Acts as if no image were specified
Within maximum and minimum limits	Centers the image within the maximum area
Larger than the maximum size	Clips the right side and bottom of the image to fit the maximum size

Placing Icons

By default, the window manager places icons in the lower left corner of the workspace. Successive icons are placed in a row proceeding toward the right. Icons are prevented from overlapping by resource specification. An icon is placed in the position it last occupied if no icon is already there. If that position is taken, the icon is placed at the next free location.

The *iconAutoPlace* resource indicates whether the window manager arranges icons in a particular area of the screen or places each icon at the window location when it is iconified. The value **True** indicates that icons are arranged in a particular area of the screen, determined by the *iconPlacement* resource. The value **False** indicates that an icon is placed at the location of the window when it is iconified. The default is **True**.

The *iconPlacement* resource is available only when *iconAutoPlace* has the value **True**. The *iconPlacement* resource specifies the arrangement scheme the window manager uses when placing icons on the workspace. The default value is *left bottom* (see Table 5−13).

Table 1 Icon Placement Values

If you want this icon placement...	*Choose this option...*
From left to right across the top of the screen, new rows below	*left top*
From right to left across the top of the screen, new rows below	*right top*
From left to right across the bottom of the screen, new rows above	*left bottom*
From right to left across the bottom of the screen, new rows above	*right bottom*
From bottom to top along the left of the screen, new columns to the right	*bottom left*
From bottom to top along the right of the screen, new columns to the left	*bottom right*
From top to bottom along the left of the screen, new columns to the right	*top left*
From top to bottom along the right of the screen, new columns to the left	*top right*

The *iconPlacementMargin* resource specifies the distance between the edge of the screen and the icons. The default value is equal to the default space between icons.

The resources that place icons use the following format:

```
Mwm*resource : value
```

For example, if you want automatic placement of icons starting at the top of the screen and proceeding down the right side, use the following specifications in a resource file:

```
Mwm*iconPlacement: top right
Mwm*iconAutoPlace: True
```

The Motif Window Manager allows you to use an icon box to display icons in a separate **mwm** window. The icon box is a scrollable window that displays icons in rows and columns. The *useIconBox* resource enables the window manager's icon box facility. The value of **True** places icons in an icon box. The default value of **False** places icons on the root window.

The icon box is displayed in a standard window management client frame. Client−specific resources for the icon box can be specified by using *iconbox* as the client name.

Resources for icon boxes have the following format:

```
Mwm*iconbox*resource : value
```

Resources that can be used with the icon box are *clientDecoration*, *windowMenu*, and all window resources dealing with color, shadow, and matte.

The *iconBoxGeometry* resource sets the initial size and placement of the icon box. If the *iconBoxGeometry* resource is set, the largest dimension of the size determines whether the icons are placed in a row or a column. The default policy is to place icons in rows going from left to right, top to bottom. The value of this resource is a standard window geometry string with the following syntax:

```
<width>x<height> {+-}<xoffset> {+-}<yoffset>
```

For example, use the following specification in your `.Xdefaults` file for an icon box 3 icons wide and 2 icons high:

```
Mwm*iconBoxGeometry: 3x2
```

The actual size of the icon box window depends on the *iconImageMaximum* and *iconDecoration* resources. By default, **mwm** places the icon box in the lower left corner with space for six icons in a single row.

Every client window that can be iconified has an icon in the icon box, even when the window is in the normal state. The icon for a client is put into the icon box when the client becomes managed by the window manager, and is removed from the icon box when the client withdraws from being managed.

Icons for windows in the normal state are visually distinct from icons for windows that are iconified. Icons for windows that are iconified look like standalone icons. Icons for windows that are in the normal state appear flat and are optionally grayed out. The value of **True** for the *fadeNormalIcon* resource grays out icons for normalized windows. The default value is **False**.

The text and image attributes of icons in icon boxes are determined in the same way as for standalone icons; that is, by using the *iconDecoration* resource.

Specifying Fonts

The *renderTable* resource specifies the style of text characters in the title area, menus, and icon labels. The default font is called *fixed*, a fixed −width font that does not make adjustments for a particular character's size and shape. The fonts available on your system are usually listed in directories under `/usr/lib/X11/fonts`. You can set the *renderTable* resource to any basename of a font listed in the font directories.

You can specify a font for all **mwm** components by using the following syntax:

```
Mwm*renderTable: font_name
```

You can specify a font for a particular **mwm** component (menu, icon, or client) by using the following syntax:

```
Mwm*menu*renderTable: font_name
```

Remember that the size of the font affects the size of the client window frames, icons, and menus. The fonts used by applications managed by **mwm** are not affected when you set the *renderTable* resource.

Setting Input Focus Policy

Use the *keyboardFocusPolicy* resource to specify how a window becomes active and receives keyboard input. The default value for this resource is *explicit*, which moves the keyboard input into a new window only when the window is explicitly selected. Keyboard input goes to the selected window regardless of the location of the pointer until you select another window for keyboard input. The default focus−selection action is pressing Button 1 in a window. Explicit focus policy is also known as "click−to−type."

The other input focus policy is called *pointer*. Under this policy, keyboard input is always sent to the window that currently contains the mouse pointer. To change the default input focus policy, use the following specification in your `.Xdefaults` file:

```
     Mwm*keyboardFocusPolicy: pointer
```

When the focus policy is set to *pointer*, the window with the input focus will not automatically be raised to the top of the stack.

The *focusAutoRaise* resource determines whether the window with input focus is raised. This resource has a default value of **True** when the keyboard focus is *explicit*, and a default value of **False** when the keyboard focus is *pointer*. If you are using *pointer* policy and you want the window with input focus to automatically be raised to the top of the stack, use the following specification:

```
     Mwm*focusAutoRaise: True
```

To avoid flickering in the display while the pointer moves across a number of stacked windows, there is a brief delay before the window with the input focus is raised to the top of the stack. This delay is controlled by the *autoRaiseDelay* resource. The default is 500 milliseconds.

Localizing mwm

Dialog widgets display three default buttons. In English−speaking locales, these default buttons generally display the words "Help," "Cancel," and "OK." However, you may customize these buttons to display words appropriate to your locale. To customize these buttons, just specify the appropriate word or words for the **XmNmessageString**, **XmNcancelLabelString**, and **XmNokLabelString** resources.

For example, suppose that you are in a French−speaking locale. In this case, you can customize the dialogs by adding the following lines to your `.Xdefaults` file:

```
     Mwm*confirmQuit*messageString:                 Quitter Mwm
     Mwm*confirmQuit*cancelLabelString:             Annuler
     Mwm*confirmQuit*okLabelString:                 Confirmer

    ! Restart Dialog
    ! −−−−−−−−−−−−−−

     Mwm*confirmRestart*messageString:              Redemarrer Mwm
     Mwm*confirmRestart*cancelLabelString:          Annuler
     Mwm*confirmRestart*okLabelString:              Confirmer

    ! Toggle Default Dialog
    ! −−−−−−−−−−−−−−−−−−−−−

     Mwm*confirmDefaultBehavior*messageString:      Utiliser le Comportement Par Defaut
     Mwm*confirmDefaultBehavior*cancelLabelString:  Annuler
     Mwm*confirmDefaultBehavior*okLabelString:      Confirmer

    ! Toggle Custom Dialog
    ! −−−−−−−−−−−−−−−−−−−−

     Mwm*confirmCustomBehavior*messageString:       Utiliser le Comportement Personnalise
     Mwm*confirmCustomBehavior*cancelLabelString:   Annuler
     Mwm*confirmCustomBehavior*okLabelString:       Confirmer
```