User Experience-UX

This book deals with the User Experience | UX Application Layer

Hints & Shortcuts

Every second counts

In today's ultra rapid/busy/chaotic world, distractions are part of the norms and it is bad news for productivity. It's no surprise that it takes us for ever to refocus and recenter on the task at hand. And way too many times a day. Even worst when you are on a deal-line and must create something.

Time to reset things and get back to basics

In order to maximize your productivity while minimizing physical necessary mouse movement needed to reach the window's controls and change in your eyes focus to reach those controls, you must learn and start using mouse & keyboard shortcuts. It has been proven again and again that fever distractions helps maintain focus and velocity. We believe that this one change in your work habits can yeld big improvement. To the point where it becomes muscle reflexes, and that's where you win.

Getting Started, again!

Using the Mouse

To get started or restarted, you need to know a few basic techniques for using the mouse and working with icons.

• To ***select an icon***, place the cursor over the icon, then click the left mouse button. Icons turn yellow or red (based on your preference) when they are selected.

• To ***select several icons***, press the ***Ctrl*** key while selecting the icons or press the left mouse button and drag the mouse. All icons touched by the box you create are selected.

• To ***open an icon***, place the cursor over the icon, then double-click the left mouse button. When you open an application icon, you run the application; when you open a folder icon, you open a window that shows the contents of that directory.

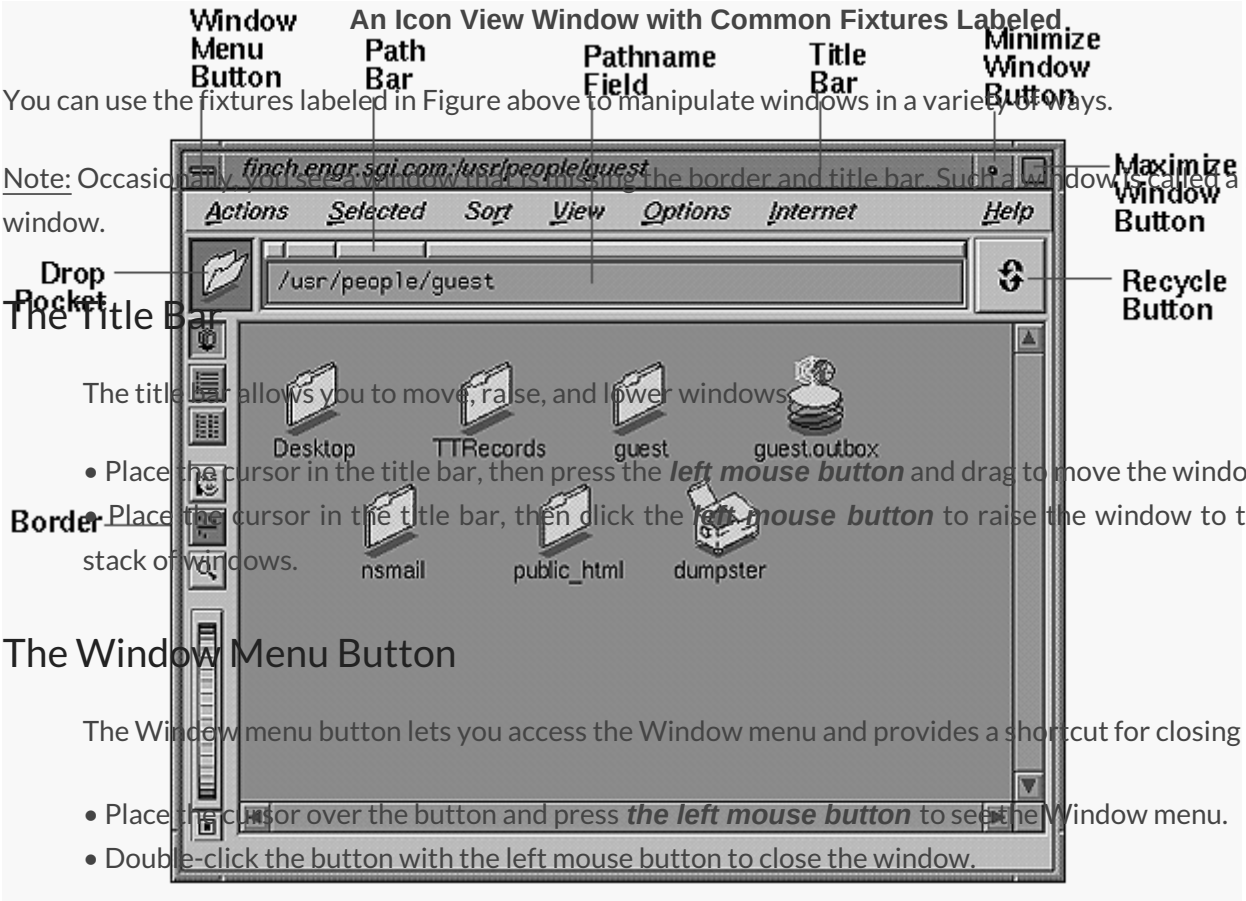
• To ***drag and drop icons***, place the cursor on an icon, then press the left mouse button and move the mouse.

Release the mouse button to drop the icon. Application and folder icons turn blue when they are ready to accept an icon. A drop pocket also turns blue when it is ready to accept an icon. This drag-and-drop technique lets you move icons from one directory to another, launch applications with a particular file, and place iconsin drop pockets.

• To display a ***pop-up menu that contains commands from the Selected menu***, place the cursor over a window or on the desktop background, then press the right mouse button.

Manipulating Windows

Most of the windows that appear on your screen have standard fixtures. Below is picture of an ***Icon View*** window. You open an Icon View whenever you open a directory—for example, when you double-click a folder icon.



You can use the fixtures labeled in Figure above to manipulate windows in a variety of ways.

Note: Occasionally, you see a window that is missing the border and title bar. Such a window is called a border-less window.

The Title Bar

The title bar allows you to move, raise, and lower windows.

• Place the cursor in the title bar, then press the ***left mouse button*** and drag to move the window.
• Place the cursor in the title bar, then click the ***left mouse button*** to raise the window to the top of the stack of windows.

The Window Menu Button

The Window menu button lets you access the Window menu and provides a shortcut for closing windows.

• Place the cursor over the button and press ***the left mouse button*** to see the Window menu.
• Double-click the button with the left mouse button to close the window.

The Minimize Button

Click the ***Minimize*** button with the ***left mouse button*** to turn the window into a small, square icon. This does not close the window or stop any of the programs that are running. It turns it into an icon that takes up less screen space.

The Maximize Button

Click the ***Maximize*** button with the ***left mouse button*** to make the window as large as it can be. Some

windows become as large as the screen; others change only slightly. If a window does not change or becomes smaller when you click the Maximize button, the window was already at its largest size.

To restore a maximized window to its original size, click the Maximize button again.

The Border

You can use the border in conjunction with the mouse buttons and keyboard keys to manipulate the window in a variety of ways:

- Place the cursor on the border, then press the ***right mouse button*** to open the Window menu.
- Place the cursor on the border. When you see a ***resize cursor***, drag the mouse to resize the window.
- Place the cursor on the border, then press the ***middle mouse button*** and drag to move the window.
- Place the cursor on the border, then press the ***left mouse button*** to raise the window.
- Place the cursor on the border, then press the ***CTRL key*** and left mouse button simultaneously to lower the window.

Window Interaction Shortcuts

Function	Keyboard + Mouse Shortcuts
Move a window	ALT + left mouse button ALT+F7 Middle mouse button on the Window's titlebar or border (like on IRIX)
Resize a window	ALT + right mouse button ALT+F8
Minimize a window	CTRL + SHIFT + left mouse button ALT+F9
Maximize a window	CTRL + SHIFT + right mouse button ALT+F10
Restore a window	CTRL + SHIFT + middle mouse button ALT+F5
Raise on top of all windows	ALT+F1
Lower down to the bottom	ALT+F3
Close window (application)	AFT+F4
Circle up overlapping windows	ALT + TAB

Circle down overlapping windows	SHIFT +ALT + TAB
Next window up	CTRL + TAB
Next window down	CTRL +ALT + TAB

Notes:
Must click inside the window area, excluding the the window's titlebar and frame. This doesn't apply for Circle and Next functions.

Desktop Interaction Shortcuts

Function	Keyboard + Mouse Shortcut
Start a new Winterm	SUPER + w
Start a new Adminterm	SUPER + a
Start a new Text Editor	SUPER + t
Open Home Directory	SUPER + h
Open Download Directory	SUPER + d
Open a Firefox Window	SUPER + f
Open a Google Chrome Window	SUPER + c
Start Slack Application	SUPER + s

Notes:
The above shortcuts works from anywhere.
The SUPER key is also known as the Windows or Apple key on the bottom section of you keyboard.

File Manager Interaction Shortcuts

Function	Keyboard + Mouse Shortcut
Select an Item	left mouse button

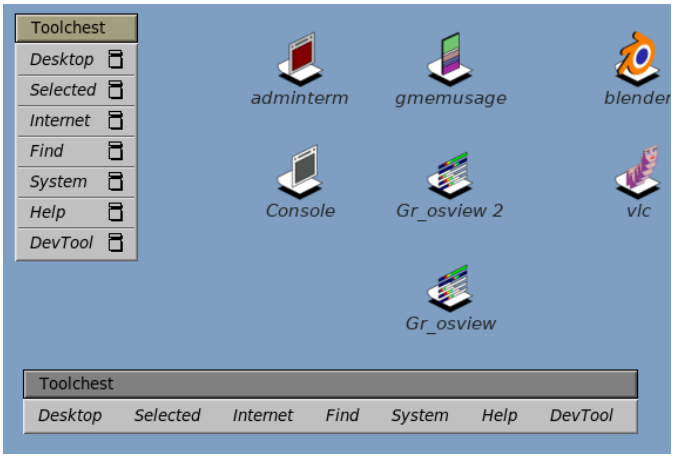
Select multiple Items (2 options)	- Hold CTRL key and make selection with ***left mouse button.*** - Use the rubber-ban by ***left mouse button*** and dragging the mouse to make your selection. Release button when done.
Adding to current selection	While holding the CTRL key make additional selection using one or the above methods.
Delete selected item(s)	CTRL + k. Warning, this function doesn't move the item(s) to the dumpster, it permanently delete them.
Drag selected item(s) to another location	You can drag selected item(s) with ***left mouse button down*** and dragging to either another File Manager window or by using the **UP** or **Home** icons in the top-left menu icon bar. Release the selection at their new destination. You will be asked if you want to move or copy them.
Send selected items to an application all at once	You can send your current selection of items to a predefined application by right mouse click on one of the selected item, then select the **Send To...** menu option from the new popup menu and select the receiving application. You can add or customize the list of options by clicking on the "customise" option on the **Send To...** popup menu
Start a Terminal from the current directory	To start a **Winterm** session from the current locatio by pressing the ` key once found on the upper left section of your keyboard. Right below the ESC
Start a File Manager window from the current directory in Winterm	To start a File Manager window from the current location inside **Winterm**, enter the command **fm** and press ENTER
Use interactive keyboard navigation	To start the interactive keyboard navigation from within a File Manager window by pressing / key once and navigate via the keyboard. Auto complete can assist you by pressing the **TAB** key.

Note:
The above shortcuts works only inside a File Manager window, except starting a File Manager from outside a File Manager :)

Toolchest

The Toolchest is an utility menu application that serves as the primary access point for desktop user interfaces. For example, users can access interfaces for everyday tasks such as customizing their desktop, accessing applications and Web tools, backing up and restoring their files, and finding information (help) on a variety of desktop topics. By default, the Toolchest is located in the upper left corner.

Toolchest in Classic Mode



Toolchest in Modern Mode

Extract from IRIX 6.5.x man page

Synopsis

```
$ toolchest [-horizontal | -vertical] [-decal | -nodecal]
```

Description

The toolchest program displays a list of buttons, each of which can invoke a useful command or present a submenu of additional buttons. The standard set of menu buttons in the IRIX Interactive Desktop toolchest is Desktop, Selected, Find, System, and Help. Each menu provides lists of useful commands for running top-level window management programs (in conjunction with 4Dwm), desk utilities, search utilities, system administration functions, or documentation-displaying programs.

Options

-horizontal	This option specifies that the toolchest should display the top level horizontally, rather than vertically.

-vertical	This option specifies that the toolchest should display the top level vertically. This is the default, but the option can be used to override saved resources.
-decal	This option specifies that the toolchest should display decals. For a vertical toolchest, a decal is displayed on each top level button that contains a menu. For a horizontal toolchest, one decal is displayed to the left of the toolchest. For an icon toolchest, no decals are ever displayed. This option is on by default.
-nodecal	This option specifies that the toolchest should not display decals.
-hidetitle	This option eliminates the title bar of the toolchest. This was the default in earlier releases but with the availability of desks, the titlebar is enabled by default and shows the name of the current desk.
-showtitle	This option shows the titlebar of the toolchest. This option is on by default.
-title	This option sets the title in the toolchest. If it is not set, the name "Toolchest" is used.

Users can change which buttons and/or menus are displayed by the toolchest program, as well as the command lines which will be launched by button selections.

The toolchest program reads a description of its buttons and menus from a menu description file when the program starts. The standard menu set is described by the system toolchest file **/opt/MaXX/etc/system.chestrc**, but users can customize their toolchest menu by providing either an auxiliary toolchest file named .chestrc or a user-customized toolchest file named .chestrc in their home directory. The system toolchest file includes the auxiliary file, and thus adds its contents to the default system menu, while the user-customized toolchest file overrides the system toolchest file. It is suggested that the auxiliary toolchest file be used for customization if possible, as future changes to the system toolchest will then be incorporated automatically. If a user-customized toolchest file is to be used, it is suggested that the system toolchest file be copied to **$HOME/.chestrc** as a starting point and then the user can modify the .chestrc file.

Remote toolchests launched via /usr/sbin/accessworkstation or from the Toolchest–Desktop–AccessFiles–Another User/By Remote Login) use the resource file /opt/MaXX/etc/remote.chestrc to show a limited number of menu choices.

Invoking toolchest with no arguments causes the user or system menu description file to be read. When one or more file names are specified to toolchest, the menu set is completely described by the list of files. Instead of files, directory names may also be specified, in which case all files in that directory that end with .chest are included, in alphabetical order. Specifying command line arguments is most useful in designing the menu layout; once this has been done, the user can concatenate the files named on the command line into a single file, name it .chestrc and put it in their home directory.

The menu tree is always rooted at a menu named "**Toolchest**"; thus the Menu **Toolchest { ... }** must be provided somewhere in the toolchest description. For example, if a menu named "**Tools**" with menu items "Gedit" and

"Gmemusage" needs to be added to the toolchest, the following should go in .chestrc :

```
Menu Tools
{
no-label f.separator
"Gedit" f.checkexec.sh.le "/usr/bin/gedit"
"Gmemusage" f.checkexec.sh.le "/opt/MaXX/bin/gmemusage"
}

Menu ToolChest

{
no-label f.separator
"Tools" f.menu Tools
}
```

Toolchest buttons and menus may be operated with either left or right mouse buttons (mouse buttons 1 or 3).

The default shell to run the commands is determined in the same way as is used by mwm, 4Dwm and 5Dwm. If the environment $MWMSHELL is defined, toolchest uses that. Otherwise, if the $SHELL is defined, it uses that. Otherwise, it uses /bin/sh. In the interests of performance, however, alternate "shell" versions of the execute commands are provided that force the use of /bin/sh regardless of the shell specified through the environment. Since /bin/sh is faster than some other shells, this can speed up launching commands. These alternate forms, used in the system toolchest, are described below, and are highly recommended unless you
have commands that are shell specific.

Files

```
/opt/MaXX/etc/system.chestrc
~/.chestrc
```

Resources

"Toolchest" is the resource class name for toolchest. "toolchest" is the resource instance name for toolchest and takes precedence over "Toolchest" when used to specify resources. However, for historical reasons, when invoked at system startup (from the Xsession.dt file), toolchest is invoked with the -name Toolchest option.

Customization

The actual location of the system-wide class resource file may depend on certain environment variables. The toolchest should not be resized. Do not set a size with geometry. The toolchest selects an optimal size based on the

font. To get a different sized toolchest, set the Toolchest*fontList: to some other font in your resources. For example, a smaller toolchest can be created with:

Change Font size in Classic Look and Feel

1. Edit the file $HOME/.maxxdesktop/Xdefaults.d/Xdefault.classic

```
toolchest*fontList:  -adobe-helvetica-bold-r-normal--10-*
```

2. Update the Desktop settings and restart toolchest

```
$ update-desktop
$ killall toolchest
$ toochest &
```

Change Font size in Modern Look and Feel

1. Edit the file **$HOME/.maxxdesktop/Xdefaults.d/Xdefault.modern**

```
*toolchest*renderTable:  xft
*toolchest*xft*fontType:  FONT_IS_XFT
*toolchest*xft*fontName:  Sans
*toolchest*xft*fontStyle:  Oblique
*toolchest*xft*fontSize:  10
```

2. Update the Desktop settings and restart toolchest

```
$ update-desktop
$ killall toolchest
$ toochest &
```

Menu Description Format

The format for the menu description file is a subset of that described for the Motif Window Manager, mwm, with a few extensions. toolchest recognizes the keyword menu and the operators f.menu, f.title, f.exec, f.separator, and f.label. In addition, the new operators f.checkexec, f.checkexpr, f.exec.sh, f.checkexec.sh, f.checkexec.sh.le, and f.checkexpr.sh have been added.

mwm-compatible Operators

The menu keyword is followed by a menu name field, then by a set of curly braces containing one or more menu description lines. Each such line has a label field, an operator field, and may have a target field.

The **f.title** operator defines a title label to be placed in the menu.

The **f.separator** operator causes a horizontal line to be drawn below the previous label.

The **f.menu** operator causes a subsidiary menu to be invoked when its label is selected. The menu may also contain the keyword dynamic which is used internally for several Desktop applications to dynamically update the toolchest's contents. There is currently no publicly available method for other applications to take advantage of this capability.

The **f.label** operator defines a label to be placed in the menu.

The **f.exec** operator defines a command to be executed behind a command button. If f.exec is used, no validation on the executability will be performed. If a more robust treatment at run-time is desired, refer to the extension operators described below. It is also possible to improve performance by using the shell versions of the exec operators, also defined below. Prior to executing the command, toolchest will load the environment defined in $HOME/.maxxdesktop/desktopenv. This file can be modified by selecting Desktop>Cusomize>Utilities from the Toolchest. Any commands that contain double or single quotes should be protected from the shell by preceding them with a backslash character. For example, if you wish to execute the command echo "Testing the toolchest" then you would add an entry to the chest's resource file as follows:

```
"Test" f.exec "echo \"Testing the toolchest\""
```

Unlike in mwm, the same menu may be named twice. Subsequent references to the menu add items to the menu. The primary purpose of declaring a menu twice is to add items to a system menu in a private file. For example, in the auxiliary toolchest file, a user could add a new pane to the top level by adding items to the ToolChest menu.

Extension Operators

In addition to the mwm-compatible operators, a set of extension operators are available.

The **f.checkexec** operator defines a command to be executed behind a command button. When selected, it behaves exactly like f.exec above. However, when the toolchest menus are being built, a validation check is run. If the command (the first argument in the command string) begins with a rooted path name (begins with "/"), then that path is checked for the existence and executability of the file. If the file does not exist, or if it is not executable, then this entry will be grayed out in the toolchest. Thus, the toolchest user will not be able to pick a menu item that will fail to execute. For example,

```
"Magnifier" f.checkexec "/usr/sbin/mag"
```

insures that /usr/sbin/mag is available to be run. If it has not been installed, for instance, the "Magnifier" item will be

grayed out.

The **f.checkexpr** operator defines a command to be executed behind a command button. When selected, it behaves exactly like f.exec above. However, when the toolchest menus are being built, a validation check is run. The f.checkexpr operator takes two double-quote-delimited expressions. The first expression is evaluated when the toolchest menus are being built; if this shell command expression fails (returns a nonzero status), then this entry will be grayed out in the toolchest. Since an arbitrary shell expression may be provided for this evaluation, a large degree of care may be exercised by the button programmer interested in protecting users from environmental dependencies which may lie within the actual command line itself. The second expression given to f.checkexpr defines the command to be executed when its button is selected, just as for f.checkexec and f.exec. For example, the command

"Flip Logo" f.checkexpr "test -x /usr/demos/bin/flip -a -r
/usr/demos/data/models/logo.bin" "/usr/demos/bin/flip
/usr/demos/data/models/logo.bin"

provides a test expression that insures the executability of the command and the readability of the critical data file. If either of these files has been deleted or lacks the required permissions, the "Flip Logo" item will be grayed out.

For improved performance, the various forms of f.exec all have shell versions of them, made by appending .sh to their names (i.e., f.exec.sh, f.checkexec.sh, and f.checkexpr.sh). Use of these versions forces the command to be run in /bin/sh instead of in $MWMSHELL or $SHELL. For certain shells, /bin/sh is faster. It is highly recommended that these shell forms be used (the system toolchest uses them). However, the older forms are provided for compatibility with older versions of toolchest.

Note that if you wish to see the sparkle launch effect when you launch from your menus, you need to add .le to the checkexec command. (i.e, f.checkexec.sh.le).

Supported Actions

Name	Description	Use MLaunch	Visual Effect	Validate Presence
f.menu	Menu			
f.separator	Separator			
f.exec	exec command	NO		
f.exec.sh	exec with sh command	NO		
f.exec.le	exec command + effect	NO	YES	
f.checkexec	use mlaunch + check	YES		
f.checkexec.sh	use mlaunch + check in SH	YES		YES

f.checkexec.sh.le	use mlaunch + check in SH	YES	YES	YES
f.dmb.le		YES	YES	YES
f.quit	Close Toolchest			
f.nop	No operation			

Nested Menus

The name on a menu line can be referenced by a f.menu operator from another menu description; this defines a cascading menu relationship.

Top level buttons displayed in the toolchest window are defined in the "Toolchest" menu. Multiple references to the same menu are not supported. Menu names should be unique. Consider the following (partial) definition:

```
menu ToolChest
{
"System" f.menu system
no-label f.separator
"Windows" f.menu windows
no-label f.separator
"Tools" f.menu tools
no-label f.separator
"Demos" f.menu demos
}

menu tools
{
"Tools" f.title
"Shell" f.checkexec.sh.le "xterm"
no-label f.separator
"Showmap" f.checkexec.sh.le "/usr/sbin/showmap"
"Makemap" f.checkexec.sh "/usr/sbin/makemap"
"Clocks" f.menu clocks
}

menu clocks
{
"" f.title
```

```
"Square Clock" f. checkexec. sh. le "/usr/sbin/clock"
"Analog Clock" f. exec. sh "xclock"
"Round Clock" f. checkexec. sh. le "oclock -bg black -fg \"dark red\""
"Digital Clock" f. exec. sh "xclock -digital"
}
```

...

Note that the "clocks" menu is cascaded from the "tools" menu, and that "tools" is cascaded from the "ToolChest" menu. So in this case the toolchest has the following top level buttons:

```
-----------
| System  |
|---------|
| Windows |
|---------|
| Tools   |
|---------|
| Demos   |
-----------
```

and selecting the "Tools" button will pop up a menu that looks something like this:

```
--------------
|   Tools    |
|============|
|   Shell    |
|------------|
| Showmap    |
| Makemap    |
| Clocks ->  |
--------------
```

Sample .chestrc file

The following sample .chestrc file adds a menu of my favorite things to the top level. This menu includes the program atlantis as well as a games sub-menu.

```
menu User
```

```
{
"My Favorite Things" □f.menu □mystuff
}

menu mystuff
{
"dolphins" □□□□f.exec "/usr/demos/bin/atlantis"
"Test Program" □□□f.exec "source ~/.variables;~/testprog"
"games" □□□□f.menu mygames
}

menu mygames
{
"flight simulator" □□f.exec /usr/demos/bin/flight
"arena" □□□□f.exec /usr/demos/bin/arena
}
```

NOTE
When the desktop is configured off with /etc/chkconfig desktop off, an alternate toolchest file called /usr/lib/X11/nodesktop.chestrc is used; it contains fewer entries and functions.

For more information about the entire IRIX Interactive Desktop environment, see the IID (1) man page.

Tellwm

tellwm like on IRIX is a shell front end to invoke the 5Dwm window manager functions. MaXX**desktop**'s tellwm is also compatible with IRIX's 4DWm

Synopsis

```
$ tellwm [-t] [-h] [-v] command
```

Description

The **tellwm** utility invokes window functions in a cooperating resident window manager program.

Cooperating window managers post the ***_SGI_TELL_WM*** property on the root window, containing a list of command strings they support externally. If the command argument given to **tellwm** matches this published protocol, **tellwm** forwards the command to the window manager for execution. **tellwm** exits with a non-zero status if the current window manager does not provide the cooperating property, or if it does not support the requested command. The following options are provided:

-t Test to determine whether the command can be sent, but don't send it. The exit status of **tellwm** may be checked to see if the
current window manager and its protocol support the command request given, but the requested command is not actually sent.

-v Print out version information.

Protocol

The 5Dwm window manager like 4Dwm are currently the only window managers which supports the ***_SGI_TELL_WM*** protocol property. 5Dwm like 4Dwm are publishing the following protocol of supported external commands:

auto_place, auto_save, beep,circle_down, circle_up, end_session, explicit_save, interactive_place, minimize_all, opaque, outline, pack_icons, quit, refresh, restart, restore_all, and ***save_configuration*****.**

For example, the shell command line

```
$ tellwm restart
```

causes 4Dwm to perform the same action as when the Restart item (f.restart) is invoked from the 4Dwm menu. Refer to the 4Dwm
documentation for definitions of these commands.

5Dwm adds the following protocols as extension only available to MaXX**desktop**:
debug and ***fast_restart*** .

Change 5Dwm mode into SGI IRIX 4Dwm mode

For all the die-hard IRIX users out there that can't stand the *click-and-raise* and/or the cumbersome *keyboard focus policy*, you are in luck. Follow this little step-by-step procedure to enable the SGI IRIX behaviour.

> The next version of MaXX Desktop will provide a simpler point-and-click behaviour toggle.

1- Edit the 5Dwm configuration file located in $HOME/.maxxdesktop/Xdefault.d/Xdefaults.5dwm, and change the following:

```
        5Dwm*keyboardFocusPolicy:           explicit
!       5Dwm*keyboardFocusPolicy:           pointer
```

to

```
!        5Dwm*keyboardFocusPolicy:           explicit
       5Dwm*keyboardFocusPolicy:           pointer
```

2- Reload your desktop settings - from a *winterm* session run the following command:

```
$ update-desktop
```

3- Restart 5Dwm in order to apply your changes - from a *winterm* session run the following command:

```
$ tellwm restart
```

> Just remember that you need to select the window's frame or titlebar, if you want to raise it. Unless you modify your 5Dwm ButtonBindings.

4- Tweak your 5Dwm button binding to allow window raise with CTRL+RIGHT-MOUSE-BUTTON. Edit your $HOME/.mwmrc file as described bellow

```
Buttons 5DwmButtonBindings
{
        <Btn1Down>              frame           f.raise
```

```
        Ctrl<Btn1Down>          window          f.raise
        <Btn1Click>             icon            f.restore_and_raise
        Shift<Btn1Down>         root            f.menu DefaultRootMenu
        <Btn2Down>              frame|icon      f.move
        <Btn3Down>              root            f.menu DefaultRootMenu
        <Btn3Down>              frame|icon      f.menu _5DwmWindowMenu_
        Alt<Btn1Down>           window          f.move
        Alt<Btn3Down>           window          f.resize
        Shift Ctrl<Btn1Down>    window          f.minimize
        Shift Ctrl<Btn2Down>    window          f.restore
        Shift Ctrl<Btn3Down>    window          f.maximize
        Ctrl<Btn1Down>          frame           f.lower
}
```

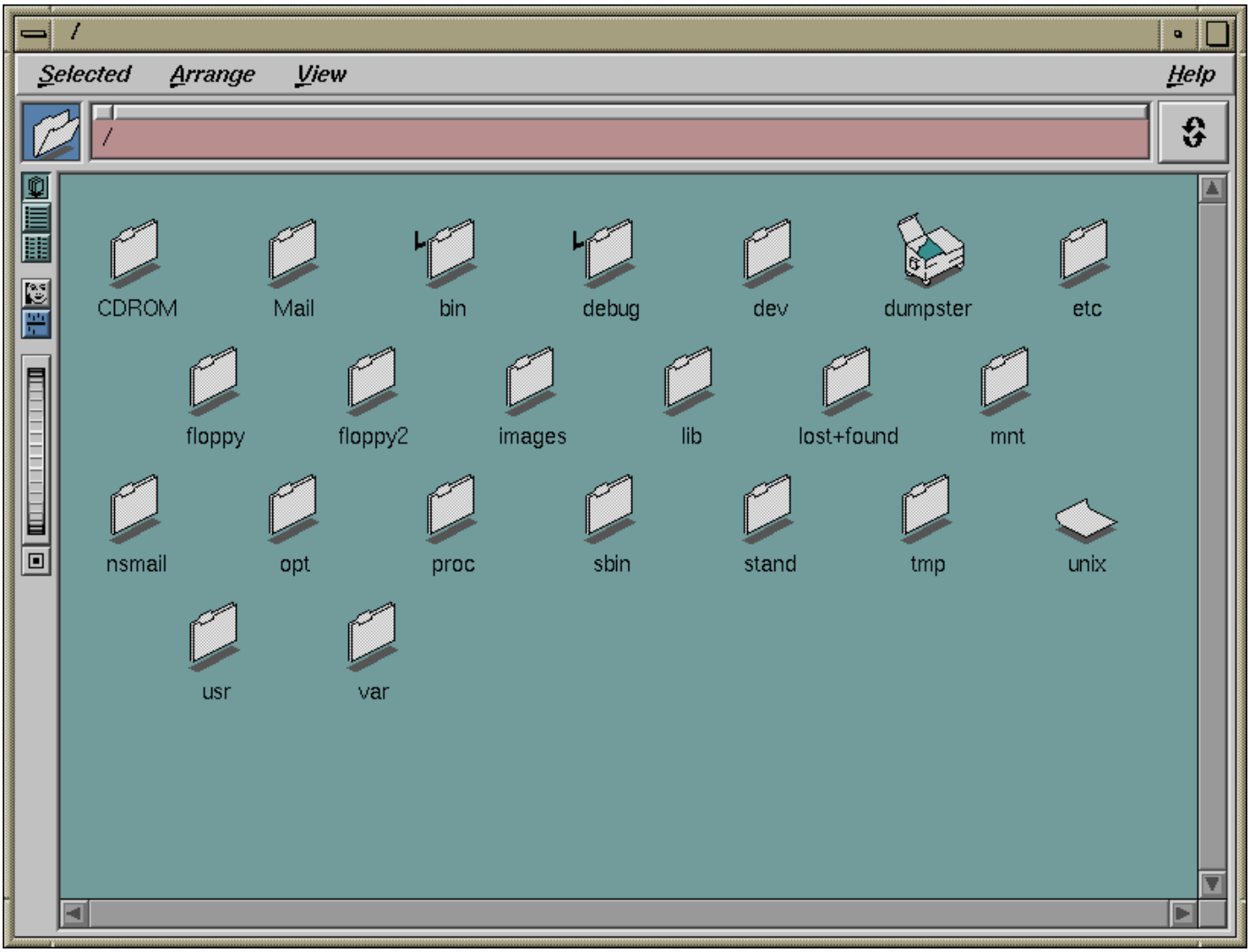
That's it! You are now back on an IRIX machine :) enjoy

Desktop Manager

work in progress. come visit again soon :)

File Manager

File Manager window. From this window, you can view file contents (example shown is a model of an X29 fighter jet, partly obscured by the System Manager window). The File Manager window is shown at the left, above the Desks Overview window.



Icon Catalog

Icon Catalog. Users can access icons from the different pages in the Icon Catalog. Some of the pages are: Applications, Demos, Desktop Tools, Media Tools, and Web Tools. Since the Icon Catalog is one of the first places users look when they need to find an application, you should add your product's icons to this catalog. By default, the Icon Catalog is below the Toolchest.

work in progress. come visit again soon :)