Do-it-Yourself Window Maker Themes

Desktop Designer

Flexible window managers with a small footprint like Window Maker don’t come two a penny. This article shows you how to create your own desktop designs and themes for use with the Window Maker window manager to make you feel at home.

BY ANDREAS KNEIB

Before we look into creating customized icons, menu bars, and backgrounds, let’s take a look at some basic steps.

Although the window manager is included with most distributions, you might prefer to download and install a brand new version from *http://www.windowmaker.org*. An important point to watch out for here is the fact that you will need to explicitly add support for KDE or GNOME to the program. The first step is to unpack the tarball.

The following syntax calls the *configure* script to add KDE/GNOME support:

```
./configure --enable-kde ⮐
--enable-gnome
```

Additional options, such as support for Open Look (*--enable-openlook*) are to be found in the *INSTALL* file in the Window Maker directory created when you unpacked the tarball; alternatively, you can type *./configure --help*. If you do not need to specify any parameters just follow the standard *configure / make / make install* procedure. You can be logged on as a normal user for the first two commands:

```
./configure
make
```

Then log on as *root* and complete the installation by typing:

```
make install
ldconfig
```

SuSE Linux users can run *SuSEconfig* instead of *ldconfig*. The former takes slightly longer, as the SuSE script runs both *ldconfig* and a few other programs.

After adding the following line:

```
exec wmaker
```

to *~/.xinitrc* or *~/.xsession*, you should be able to launch the new desktop.

First Impressions

Let’s take an introductory tour of Window Maker. The center mouse button displays a list of active windows. If you do not have a three-button mouse, just press the left and right buttons simultaneously.

The launching pad for programs is hidden under the right mouse button. Right click on the desktop to open the application menu. The *WPrefs* tool allows you to configure the menu. If you right click on a window frame, you can move applications to another desktop, scroll, or edit the properties of the current window. More options become available when you click on an icon – and there should be several of them on your screen right now.

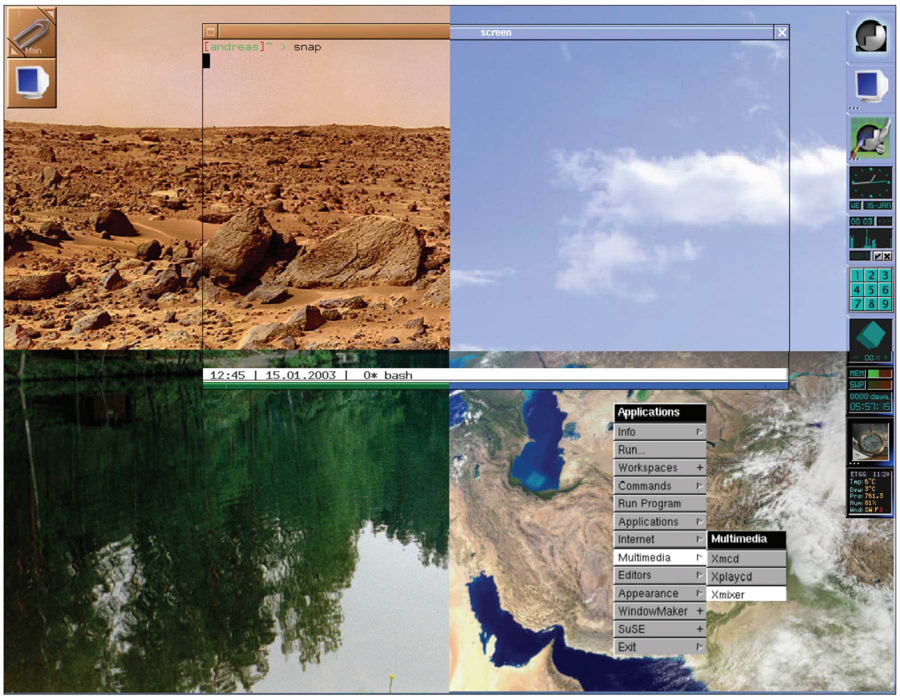


Figure 1: One window manager, many faces

Docked

The icon at the top right of your screen is the *WMDock*. It is the anchor point of a virtual mooring chain to which you can add further application icons, as shown in Figure 2. Tools that are docked, but not active, are displayed as icons with three dots. Of course, you can right click on this gap to modify the properties. As you might expect, *Settings…* displays the settings in detail, *Launch* executes and *Kill* kills a program. And to hide an application – guess what – choose *hide*.

Another icon containing a paper clip sits in the top left corner of the screen (Figure 3). This is known as *WMClip* and follows a similar principle to the application dock. You will note two arrows in the corners of the screen that allow you to switch desktops. The number of the desktop you are currently working on is displayed above the paper clip. *WMClip* provides a wide range of options and we would like to look at just one of two of the most important ones.

For example, you can use the *Clip Options* to specify whether applications are allowed to hide the clip (*Keep on Top*), or docked icons automatically (*Collapse*), or if the clip should (*Auto-attract Icons*). You can use the *Selected* option to choose icons that you want to remove from the clip (*Remove Icon*), or move to another desktop (*Move Icon To*).

To tidy up all the program icons lying around on your desktop by adding them to the clip, select *Attract Icons*. Most of the clip's remaining functions are very similar to the application dock.

A Question of Preferences

As you can see in Figure 2, Window Maker provides various small tools, also called *DockApps* [1]. They attached to the dock or clip just like icons, and their usefulness ranges from irreplaceable (*wmweather*) to just toys (*wmmatrix*). Having said that, after using them for a while you probably won't want to do without any of these cubes; their fun potential and addictive qualities are not to be underestimated. If your distributor hasn't been too generous with the selection of DockApps provided with Window Maker, try the DockApp Warehouse [2], which offers over 200 applications. You can use the mouse to attach these programs to the dock or clip, to move or organize them.

Before we get down to the creative part of the story, let's just have a look at the control center for this window manager, *WPrefs*. Its icon displays the GNUstep logo, a brush and a screwdriver (Figure 4).

User-friendliness is one of the *Window Maker Preferences Utility*'s strong points. You can check *Balloon-Help* to enable bubble help for each section. Sections are organized in categories such as window management, with its many preferences, through to ergonomics, search paths and mouse operations, all the way to keyboard shortcuts and application menu customization.

This is particularly useful if you're not too keen on using a text editor to change configurations. As the main topic in this article is creating your own themes, it makes sense to specifically point out the *Appearance Preferences* option that conveniently allows you to compile menu bars and icons by pointing and clicking, although your creations may have some rough edges at first.

So let's look at some customization steps in the *Applications Menu Definition* section of *WPrefs*. As Figure 5 shows, menu bars can be edited by simply dragging icons out of them, or inserting new icons. Double-click on the field to add a new text, or single click to define the action that this element will perform a future.

Styles

You may have noted the *Themes* and *Styles* sections while experimenting with Window Maker's application menu. Both of these options completely changed the appearance of the desktop; so it makes sense to have a closer look at them.

Let's start with the *Styles* – Window Manager is supplied with quite a few defaults. Clicking on *Appearance / Styles / Autumn* gives an autumnal look to icons and scroll boxes. So what happened?

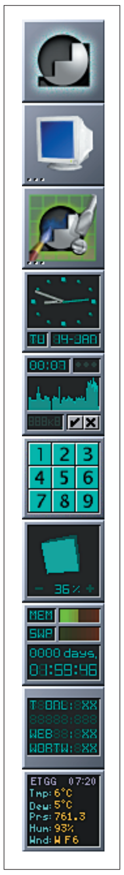


Figure 2: Docked Applications

Figure 3: WMClip, the paper clip

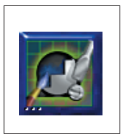
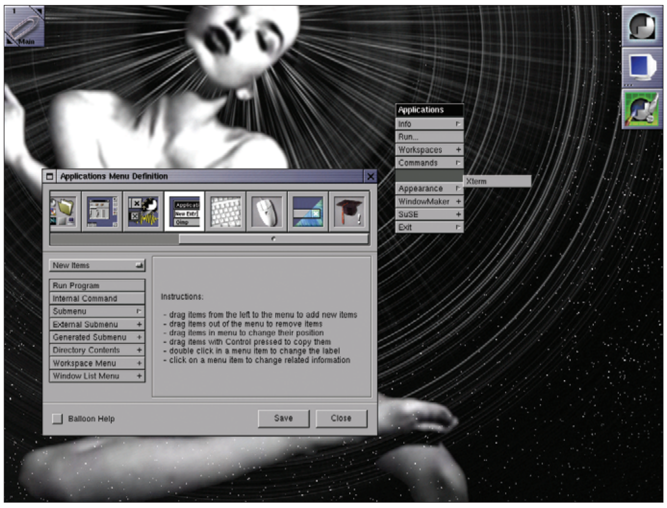


Figure 4: The Wprefs command center



Instructions:

- drag items from the left to the menu to add new items
- drag items out of the menu to remove items
- drag items in menu to change their position
- drag items with Control pressed to copy them
- double click in a menu item to change the label
- click on a menu item to change related information

Figure 5: Compiling a menu bar

Looking closely at the *Appearance / Styles* in *WPrefs* reveals two things: You were working with tool called *setstyle*, and this tool applies itself to the */usr/local/share/WindowMaker/Styles* and *~/GNUstep/Library/WindowMaker/Styles* paths. The tool may be restricted to loading style files, but you can still see the manpages launched by typing *man setstyle*.

Now let's look for the *Autumn.style* file in the system global directory */usr/local/share/WindowMaker/Styles*. If you're using packages provided by your own distribution, these paths may vary. Thanks to intuitive variable names, the structure is quite clear. Listing 1 shows the file.

The whole configuration is surrounded by two curly brackets with the first line setting the *TitleJustify* variable, which is used to justify the titles in window title bars. Thus, you can *center* titles, instead of left or right justifying them with the *left* or *right* switch.

The next six lines are for font management. *ClipTitleFont* specifies the front for the paper clip icon, *WindowTitle Font* specifies the font for window titles, *MenuTitleFont* refers to the header of the application menu, and so on. The tool *xfontsel* is ideal for your own creations.

HighlightColor defines the color of highlighted elements, and *HighlightText-Color* the appropriate text. *ClipTitleColor* defines the color of the font specified by the *ClipTitleFont* variable. Any collapsed icons in the clip are assigned color specified by *CClipTitleColor*.

FTitleColor, *PTitleColor*, and *UTitleColor* specify the text color for the title bar of the window with the current focus (*FTitle*), a main window without the current focus (*PTitle*), or of a generic window without a focus (*UTitle*).

Things start to get a bit more complicated, when you get around to configuring the *FTitleBack*, *PTitleBack*, and *UTitleBack* lines, which specify the appearance of the title bars. The values here are made up of 3 elements surrounded by brackets: *(hgradient, darkred, black)*. *hgradient* means a horizontal color gradient from *darkred* to *black*. In addition to *hgradient* there are other variants, such as *dgradient* (diagonal gradient) and *vgradient* (vertical gradient).

As far as the colors are concerned, it is interesting to note that you are not restricted to specific color names such as *black*. If you prefer a more granular resolution, you can use hexadecimal sequences to define a palette:

```
FTitleBack = (vgradient, ⮐
"#94CA73", "#BDF78C")
```

just like an RGB definition:

```
FTitleBack = (hgradient, ⮐
"rgb:50/5a/5e", "rgb:20/2a/2e")
```

And if that still doesn't make you happy, you might be interested in adding an image of your own:

```
FTitleBack = (spixmap, ⮐
title.jpg, yellow);
```

Let's get back to Listing 1: *ResizebarBack* refers to the lower frame, which is used to define the size of the window. *Menu-*

Listing 1: Autumn.style

```
{
  TitleJustify = center;
  ClipTitleFont = "-*-helvetica-medium-r-normal-*-10-*-*-*-*-*-*-*";
  WindowTitleFont = "-*-helvetica-bold-r-normal-*-12-*-*-*-*-*-*-*";
  MenuTitleFont = "-*-helvetica-bold-r-normal-*-12-*-*-*-*-*-*-*";
  MenuTextFont = "-*-helvetica-medium-r-normal-*-12-*-*-*-*-*-*-*";
  IconTitleFont = "-*-helvetica-medium-r-normal-*-8-*-*-*-*-*-*-*";
  DisplayFont = "-*-helvetica-medium-r-normal-*-12-*-*-*-*-*-*-*";
  HighlightColor = white;
  HighlightTextColor = black;
  ClipTitleColor = white;
  CClipTitleColor = gray20;
  FTitleColor = white;
  PTitleColor = gray;
  UTitleColor = white;
  FTitleBack = (hgradient, darkred, black);
  PTitleBack = (hgradient, indianred, black);
  UTitleBack = (hgradient, peru, black);
  ResizebarBack = (hgradient, peru, black);
  MenuTitleColor = white;
  MenuTextColor = white;
  MenuDisabledColor = gray80;
  MenuTitleBack = (hgradient, firebrick, black);
  MenuTextBack = (hgradient, peru, black);
  IconBack = (dgradient, "#efb573", "#734221");
  IconTitleColor = white;
  IconTitleBack = "#8b0000";
  MenuStyle = normal;
}
```

TitleColor refers to the color of the fonts in menu title bars, *MenuTextColor* specifies the color of the text within a menu. The value specified in *MenuDisabled Color* is used to disable a menu area, that is to make it unclickable. *MenuTitleBack* specifies the background for *MenuTitle Color*. *MenuTextBack* (text background) and *MenuTextColor* (text color), as previously described, follow a similar pattern.

The *IconBack* line defines an icon's color, but also allows you to specify an image:

```
IconBack = (tpixmap, ➥
Icon.jpg, black)
```

IconTitleColor and *IconTitleBack* are used to configure the color of the title background and title text half a programme icon. And finally, *MenuStyle* is responsible for defining the format of the menu elements, where *normal* refers to a button format, and *flat* provides a flat surface.

Themes

So, how can we make a theme from a style? People obviously do because Freshmeat [3] has literally thousands of them to offer. Let's follow some extremely simple steps first.

Select a theme with the background image in *Appearance / Themes*. Then select a style in *Appearance / Styles*. Finally, save the results by selecting *Appearance / Save Theme* in the menu and typing an appropriate name. Looking again at this point in *WPrefs* you might stumble across a command called *getstyle -t* and the path to the theme directory in the *~/GNUstep* directory.

```
getstyle -t mytheme
```

in the command line would create exactly this file in your home directory; of course you will now find it in *~/GNUstep/Library/WindowMaker/ Themes*. The *setstyle mytheme* syntax loads your theme and displays it in Window Maker.

For the best results, you should start with the background image, maybe one that you have found on the Internet [4] or created with Gimp [5]. Note that 800 x 600 images do not look good if you are using a screen resolution of 1024 x 768, or even 1152 x 870. Now move the image to the *~/GNUstep/Library/ WindowMaker/Backgrounds* directory, to list it under *Appearance / Background / Images*.

The following command can save you a lot of effort:

```
wmsetbg -u -t ~/GNUstep/Library➥
/WindowMaker/Backgrounds/MyImage
```

Now select a style that suits your background, and follow the steps mentioned previously. You might then like to load the file with your favorite editor and gradually edit the template until you are completely satisfied with Window Maker's new outfit. And let's not forget that the window manager is capable of using images for icons, menus and scroll boxes (Listing 2).

This example introduces two new concepts: *spixmap* and *tpixmap*. The value for *spixmap* in the first line of Listing 2 specifies that the image file *FTitle.jpg* will be used to fill the window bar, *FTitleBack*. The main color in this image is *yellow*, and the image is scalable. *tpixmap* the same thing with the image and the color, however, it tiles the image (Figure 6). *~/GNUstep/Library/WindowMaker/Pixmaps* (typically in */usr/local/share/ WindowMaker/*, although your distribution may change this) is one of the directories in which Window Maker will search for images.

However, the convention of creating a themes folder does make things more tidy. This directory is used for storing any relevant files, has the *.themed* suffix, and can be found below *Themes*. Of course, there is a simple command to perform this task without all that copying:

```
getstyle -p ~/GNUstep/Library/➥
WindowMaker/Themes/YourTheme
```

Finally, a few words on Freshmeat projects [3]: using *tar* to unpack theme tarballs may not be everyone's idea of fun. The *wmthemeinstall* [6] tool simplifies the management. Have fun creating your own themes. ■

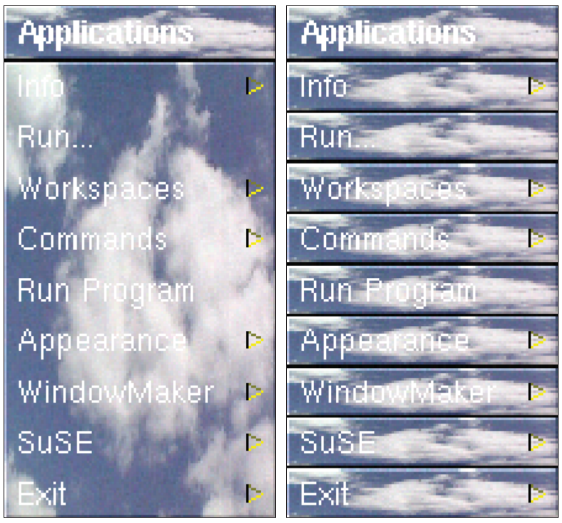


Figure 6: A scaled and tiled menu image

Listing 2: MyStyle example

```
{
    FTitleBack = (spixmap, FTitle.jpg, yellow);
    PTitleBack = (spixmap, PTitle.jpg, "#72797d");
    UTitleBack = (spixmap, UTitle.jpg, "#12191d");
    MenuTitleBack = (spixmap, MenuTitle.jpg, "#72797d");
    MenuTextBack = (spixmap, MenuText.jpg, "#72797d");
    ResizebarBack = (spixmap, Resize.jpg, yellow);
    IconBack = (tpixmap, IconBack.jpg, black);
    WorkspaceBack = (spixmap, Background.jpg, black);
}
```

INFO

[1] Jo Moskalewski: "Dockapps", Linux Magazine, Issue 3, p120

[2] DockApp Warehouse: *http://www.bensinclair.com/dockapp/*

[3] Themes: *http://themes.freshmeat.net*

[4] Downloadable images: *http://www.google.com/advanced_image_search*

[5] The Gimp darwing tool: *http://www.gimp.org*

[6] WMThemeInstall tool: *http://www.linux.org/apps/AppId_3367.html*