Overview of the X Window System

Paul B. Cohen

Massachusetts Microelectronics Center
M^{2}C

Contents

1 Introduction

The X Window System, developed at Project Athena at MIT, is designed to manage windows on a graphics workstation. **VEM** is a multi-window application which runs on top of the X Window System. In addition to **VEM**, a number of tools in the **OCT** tool suite can take advantage of the X Window System. Thus, an understanding of the use of the X Window System is essential for the effective use of the **OCT** tool suite.

This document is intended as an introduction to the use of the X Window System. The essential features of the user interface will be covered in a tutorial fashion. It is not intended to be a comprehensive guide to the X Window System. After completing this tutorial, you should be familiar enough with the X Window System to effectively use the **OCT** tools, and to begin do discover more of the capabilities of the X Window System on your own.

1.1 About the Tutorial

Since this document is prepared as a tutorial, you should read this document while logged in to a graphics workstation running the X Window System. This way, you can try out the commands and experiment with the X Window System. Command you should type are noted as follows:

```
type this command
```

- Paragraphs like this describe an operation which is somewhat more complicated than simply typing a command. Such operations generally require the mouse. You should read the paragraph before proceeding.

⋆ Experience has shown that there are certain operations which novice users often find confusing. These operations are noted with the symbol ⋆. Read these paragraphs carefully before proceeding.

- Some operations require that you *click* one of the mouse buttons. When you are asked to *click* a mouse button, you should press the button down and immediately *release* it without moving the mouse. When you are asked to *press* one of the buttons, you should *press* the indicated button down and *hold* it down while performing the indicated operation, (generally involving mouse movement), then *release* the button when the operation is complete.

- There are also times when you will need to press and hold one of the keys on the keyboard while performing an operation using the mouse. These keys will be noted by a box, as in $\boxed{\text{Shift}}$. The notation $\boxed{\text{\^{}d}}$ indicates you should press and hold the control (Ctrl) key while pressing the indicated key.

Throughout this tutorial, reference is made to the $\boxed{\text{meta}}$ key. This key may be labeled $\boxed{\text{Compose Character}}$ or $\boxed{\text{alt}}$ on your keyboard. Many of the operations we will do in this tutorial will require you to press and hold the $\boxed{\text{meta}}$ key while performing an operation using the mouse.

1.2 Window Managers

Under X, a user can choose from a number of programs known as *window managers*. These programs allow you to move, reshape, reorder, and iconify windows. Different window managers provide different interfaces to manipulating windows.

The X Window System considers the characters you type or the mouse buttons you press to be *events*. Window managers interpret certain events and determine which window the event was intended for. Some window managers *focus* events on a particular window automatically when you position the mouse cursor over the window of interest. Other window managers require you to manually focus the input on a particular window, and all events

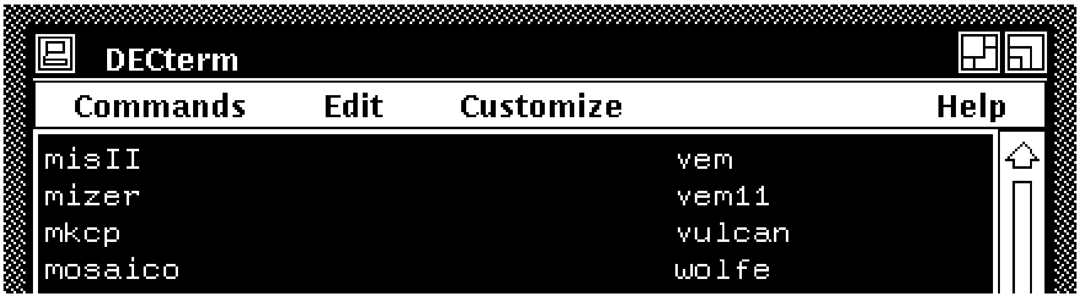

Figure 1: Sample DECterm with dxwm title bar

will be directed to that window regardless of the position of the mouse cursor. The popular window manager `uwm` is an example of a window manager which automatically focuses input on the window directly under the cursor[1]. `Dxwm` is an example of a window manager which requires you to manually focus input on a particular window. Choosing a window manager is a matter of personal taste. Generally the window manager is started when you log in.

This tutorial will describe the basic operation of the X Window System using `uwm` and `dxwm`. If you are using one of these window managers, you only need to complete the section pertaining to that window manager.

1.2.1 Which Window Manager?

If you aren't sure which, if any, window manager is running, there are a few simple tests you can perform.

- The window manager `dxwm` places a title bar over all of the windows on the screen. If the top of the terminal window looks like figure 1 then `dxwm` has been started. (The Commands, Edit, Customize, and Help *buttons* are part of the DECterm terminal emulator `dxterm`, and are not provided by the window manager.) `Dxwm` also creates a window named "Icon Box" at the top of the screen. If such a window exists, then it is safe to assume `dxwm` is running.

[1] Uwm can be "told" to focus input on a particular window, but this is not the default operation.

When you first logged in, you may have heard a loud "beep" from the terminal. `Uwm` does this to indicate it has succesfully started. If you're not sure, try the following simple test.

- Move the cursor over the terminal window. Press and hold the $\boxed{\text{meta}}$ key, and *press* the right mouse button. While *holding* the right mouse button and the $\boxed{\text{meta}}$ key down, move the mouse. If the outline of the window moves, then `uwm` or an equivalent window manager has been started. When you *release* the left mouse button, the window will move to a new location indicated by the outline.

- If no window manager has been started, you can start `uwm` by typing:

  ```
  uwm &
  ```

  `Dxwm` is distributed by Digital Equipment Corporation. If you are using a workstation from Digital, you can use `dxwm` *instead* of `uwm` if you prefer. If so, type:

  ```
  dxwm &
  ```

It is important to note that you should be running only *one* window manager. If two ore more window managers are running, the results will be unpredictable, and probably undesireable.

If a window manager has not been started, you should choose one and edit your `.login` file to start the window manager each time you log in to a workstation.

The remainder of this tutorial assumes that either `uwm` or `dxwm` has been started on the workstation.

2 Using uwm

This section assumes that the window manager `uwm` is running on the workstation.

- Try moving the mouse cursor in and out of the terminal window you used to log into the workstation. Notice how the cursor shape changes when you do this.

- With the cursor inside the terminal window, try executing a simple command such as:

  ```
  ls
  ```

  When the cursor is in the window, the characters are read by the terminal emulator, and the command is executed.

- Move the cursor outside the terminal window, and try executing the

  ```
  ls
  ```

  command again. When the cursor is outside the window, the characters are discarded.

- Create a new window by typing:

  ```
  xterm &
  ```

  After a few seconds, a window outline will appear.

- You can move the window outline by moving the mouse. Move the outline to a different part of the screen from the login window and *click* the left button. A new terminal emulator will appear.

- Position the cursor in the new terminal window and type:

  ```
  ls
  ```

  once again. Notice that the command and the response appear in the new terminal window.

- Practice typing characters in each terminal emulator by moving the mouse cursor between them.

A simple but useful window manager function is to move windows around the screen.

- Move the cursor into any terminal window. With the $\boxed{\text{meta}}$ key depressed, *press* the right mouse button. A 3×3 grid will appear over the terminal window.

WINDOW OPS
(De)Iconify
Move
Resize
Lower
Raise

Figure 2: uwm Window Ops menu

- As you move the mouse, the 3×3 grid will move around the screen. When you *release* the right mouse button, the window itself will move to the new position.

When the left or middle mouse buttons are pressed along with the $\boxed{\text{meta}}$ key, small menus will appear. The menus for the left and middle buttons are a bit different.

- With the $\boxed{\text{meta}}$ key pressed, *press* the left mouse button. A menu of window operations labeled **window ops** will appear, as shown in figure 2.
- While continuing to hold the $\boxed{\text{meta}}$ key and the left mouse button down, move the cursor over the `Move` command and *release* the left mouse button. (Do not release the $\boxed{\text{meta}}$ key.
- Continue to press the $\boxed{\text{meta}}$ key down, and move the cursor over a terminal window. *Press* the left mouse button, and note the 3×3 grid appears, as before. You can now move the grid around the screen, and when you *release* the left mouse button, the terminal window will move.

Moving windows may be done either from the **window ops** menu, or by using the $\boxed{\text{meta}}$ key and the right mouse button. Moving windows is done frequently enough that is is well worth its own mouse button.

It is often useful to change the size of a window. This is done with the `Resize` command from the **window ops** menu.

- Once again, press and hold the $\boxed{\text{meta}}$ key and *press* the left mouse button to bring up the **window ops** menu. This time, move the cursor

Top & Left	Top only	Top & Right
Left	All	Right
Left & Bottom	Bottom only	Right & Bottom

Figure 3: Resize command operations

over the **Resize** command, and *release* the mouse button (but not the meta key).

- Move the cursor to a point somewhere near the bottom center of one of the terminal windows.
- *Press* the left mouse button again, and the 3×3 grid will once again appear. In addition, a small box with the current size, in characters, (80×24) will appear near the top of the window.
- Move the cursor up and down to resize the grid, and *release* the left mouse button to change the window size itself.

Each of the 9 portions of the 3×3 grid allows different edges of the window to change, while fixing the remaining edges. Figure 3 indicates which edges of the window can change based on the area of the window where the left mouse button is pressed.

- Try using the **resize** command from different areas of the terminal window and observe the effect.

When using multiple windows, there are times when the window of interest is "underneath" another window, and is partially obscured. The **Raise** and **Lower** commands will *reorder* the windows.

- Move one of the terminal windows so that two windows partially overlap. (If you don't have at least 2 terminal emulator windows, create one with the command `xterm &`

- From anywhere on the screen, press and hold the meta key, and then *press* the left mouse button. The **window ops** menu will appear.
- Move the cursor over the `Raise` command, and *release* the left mouse button.
- Move the cursor over the "bottom" window. With the meta key depressed, *click* the left mouse button. This will raise the bottom window to the top of the pile.
- Try using the `Raise` and `Lower` commands to control which windows are fully visible, and which are hidden.

There are times when it is useful to move a window "out of the way" for a while, especially when the screen is becoming cluttered with windows. This is done by *iconifying* a window.

- Use the meta key and the left mouse button to highlight the `(De)Iconify` command on the **window ops** menu. *Release* the left mouse button, and move the cursor over one of the terminal windows.
- Click the left mouse button over the terminal window. The window will shrink to a small box, known as an *icon*
- Use either method of window movement discussed earlier to move the *icon* to a convenient place (usually near the edge of the screen).
- Use the meta key and the left mouse button to highlight the `(De)Iconify` command on the **window ops** menu. *Release* the left mouse button, and move the cursor over the iconified window.
- With the meta key depressed, *click* the left mouse button, and the window will return to its original size and position.

The `(De)Iconify` command operates as a "toggle switch", in that the command will reverse the state of the window under the cursor when it is executed.

There is a second window operations menu, called **Extended Window Operations**, which is invoked by holding the meta key and pressing the

EXTENDED WINDOW OPS
Create Window
Iconify at New Position
Focus Keyboard on Window
Freeze All Windows
Unfreeze All Windows
Circulate Windows Up
Circulate Windows Down

Figure 4: uwm Extended Window Ops menu

middle mouse button. The commands on this menu are shown in figure 4. Since these commands, while useful, are used less frequently than the **Window Ops** commands, they will be described briefly. You are encouraged to experiment with the window operations from both the **Window Ops** and **Extended Window Ops** menus.

The `Create Window` command allows you to create another terminal window, like the `xterm &` command. This is handy when you are running a long program in one window, and don't want to interrupt it to create a new window. `Iconify at New Position` operates in a manner similar to the combination of `(De)Iconify` and `Move` on the **window ops** menu. `Focus Keyboard on Window` causes all keyboard and mouse events to be focused on a particular window, regardless of where the cursor is on the screen. `Freeze All Windows` will stop all of the windows from changing, `Unfreeze All Windows` allows them to change once again. `Circulate Windows Up` reorders the windows by moving the top window to the bottom, and moving all other windows up one position. `Circulate Windows Down` brings the bottom window to the top, moving all others down 1 position.

It is often useful to remove some of the windows you have created.

⋆ Move the cursor to one of the X Window System you created, but be sure it is not in the window you originally used when you logged in.

- The window can be closed by typing:

  ```
  exit
  ```

  or by pressing ^d. (If you press ^d or type `exit` while the cursor is in the login window, you will be logged out and all of the X Window

System will be closed.)

The window management operations described in this section can be used to manipulate any of the windows on the screen. This includes **VEM** windows.

3 Using dxwm

This section assumes the window manager `dxwm` has been started on your workstation.

- Try moving the mouse cursor in and out of the terminal window you used to log into the workstation. Notice how the cursor shape changes when you do this.
- With the cursor inside the terminal window, try executing a simple command such as:

  ```
  ls
  ```

  When the cursor is in the window, the characters are read by the terminal emulator, and the command is executed.
- Move the cursor outside the terminal window, and try executing the

  ```
  ls
  ```

  command again. Even though the cursor is not in the X Window System, input is *focused* on the X Window System by `dxwm`. The title bar should change color (or from normal to reverse video) to indicate which window keyboard and mouse events are focused on.
- Create a new window[2] by typing:

  ```
  xterm &
  ```

[2] On systems from Digital Equipment Corporation, two terminal emulators are generally provided. For this tutorial, we will use `xterm`. The other terminal emulator is called `dxterm`. If you prefer `dxterm`, feel free to use it in place of `xterm`. This section of the tutorial applies to either terminal emulator. `Dxterm` has additional capabilities not covered here.

After a few seconds, a new terminal window will appear on the screen. Note that the title bar indicates input remains focused on the original terminal window.

In order to enter commands in the new terminal window, we will need to focus the input on that window.

- Move the cursor over the new window, and *click* the left mouse button. Notice the title bars of the two terminal window change to indicate all input is now focued on the new terminal window. (When you are using **VEM**, *clicking* the left mouse button in a **VEM** graphics window may create an unwanted point in the window. *Clicking* over the title bar to focus input on the window will avoid this).

- Try entering a simple command such as:

  ```
  ls
  ```

  and note where the command and the response appear.

- Try different combinations of window focus and cursor position to understand the idea of manually focusing input on a window.

One very useful operation controlled by a window manager is moving windows around the screen.

- Move the cursor to the center area of the title bar in the newly created terminal window.

- *Press* the left mouse button. While *holding* the left mouse button down, move the mouse slightly. Note the outline of the window moves as you move the mouse.

- Move the outline of the window to some new location on the screen and *release* the left mouse button. The terminal window will move to the new location.

- Note that after the window has been moved, input is focused on that window.

Often it is useful to change the size of a window.

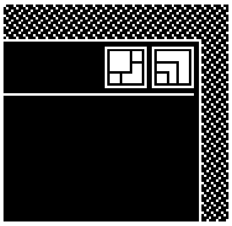

Figure 5: Right hand edge of the title bar

- At the right hand edge of the title bar there are 2 small boxes. Figure 5 shows a sample of the upper right hand corner of a terminal window with the boxes of interest.
- Move the cursor over the rightmost box, and *press* the left mouse button. Note that the cursor changes shape.
- Move the new cursor up and to the right a bit. Note how the outline of the window changes size as the cursor is moved. You can increase or decrease the size of the window by moving the cursor.
- *Release* the left mouse button when the window outline is at some new size. Note thate the terminal window changes size when the left mouse button is *released*

If you have mutliple windows which overlap on the screen, the window you have focused your input on is at the top of the "pile" of windows. Sometimes it is desireable to have another window at the top, even if input is not focued on it. The other box in near the right hand edge of the title bar is used to *reorder* the windows.

- Move the cursor over the other box at the right hand edge of the title bar (figure 5) , and *click* the left mouse button. Notice that the window is now covered, at least partially, by any other windows which may be on the screen, but input is still focused on the window you just "lowered".
- Try various combinations of lowering windows with different focus settings.
- To raise a window, simply focus input on that window in the normal manner.

Figure 6: Left hand edge of the title bar

There are times when we might want to move a window "out of the way" for a while. We can do this by *iconifying* the window.

- At the left hand edge of the title bar is small box with the name of the host system (or "DECterm" if you are using dxterm) along side. Figure 6 shows a sample of the upper left hand corner of a terminal window on a system named "hera".

- Move the cursor over this box in one of the terminal window, and *click* the left mouse button. The window is now iconified, and the icon is in the "Icon Box" window at the top of the screen. Note that the icnonfy box is still at the left hand edge of the icon.

- Move the cursor to the Icon Box window, and position it over the small box at the left hand edge of the iconified window. (Iconified windows are noted by the highlighted box. The non-highlighed boxes "reserve" the position for windows which are not currently iconified.)

- *Click* the left mouse button over the iconify box, and the iconified window will be de-iconified. That is, it will return to its previous size and position on the screen.

- Try iconifying and de-iconifying the various windows on the screen.

It is often useful to remove some of the windows you have created.

⋆ Move the cursor to one of the terminal window you created and focus input on that window. Be sure it is not in the window you originally used when you logged in.

- The window can be closed by typing:

`exit`

or by pressing $\boxed{\hat{}\,\mathrm{d}}$. (If you press $\boxed{\hat{}\,\mathrm{d}}$ while the input is focused on the login window, you will be logged out and all of the terminal window will be closed.)

The window management operations described in this section can be used to manipulate any of the windows on the screen. This includes **VEM** windows.