XCWM:
The X11 Composite Window Manager Library

Jess VanDerwalker
Background

• Portland State University Capstone project
  – Original team:
    Ben Carr, Ben Huddle, Aaron Skomra, Braden Wooley, Jess VanDerwalker
  – Sponsors:
    Jamey Sharp - XCB development
    Josh Triplett
    Jeremy Huddleston Sequoia - XQuartz
XCWM Goals

• API that allows display of X11 clients in non-X environments
• Increase code reuse and improve maintainability
• Lead to deprecation of specific DDX's in server code base
• XCB as interface to X protocol
• DAMAGE and Composite extensions
Welcome to freedesktop.org

freedesktop.org is open source / open discussion software projects working on interoperability and shared technology for X Window System desktops. The most famous X desktops are GNOME and KDE, but developers working on any Linux/UNIX GUI technology are welcome to participate.

freedesktop.org is building a base platform for desktop software on Linux and UNIX. The elements of this platform have become the backend for higher-level application-visible APIs such as Qt, GTK+, XUL, VCL, WINE, GNOME, and KDE. The base platform is both software and specifications.

Software

freedesktop.org hosts any "on-topic" software projects. If you have a project that fits into our mission and needs hosting, please make a request using our forums, mailing lists, or issue tracker.
Where we are...

- Basic window manager functionality
- Support for a subset of ICCCM and EWMH protocols
- Functional mouse and keyboard input
- Support for override-redirect windows
Basic functionality

**XCWM:**
- Create and maintain data structures
- Catch and process X events, pass to WM code
- Catch and process WM events, pass to X
- Provide access to data

**Window Manager:**
- Render X client pixmaps with native decorations
- Respond to XCWM events, update windows accordingly
- Pass local window events to XCWM
Basic window management

dummy driver

XtoQ

By default, xman starts by creating a small window that contains "buttons" (places on which to click a pointer button). The buttons, Help and Quit, are self-explanatory. The third creates a new manual page browser window; you may use this open a new manual page any time xman is running.

A new manual page starts up displaying this help information. A manual page contains three sections. In the upper left the menu buttons. When the mouse is clicked on either of the menu is popped up. The contents of these menus is described directly to the right of the menu buttons is an information display. This display usually contains the name of the directory or being displayed. It is also used to display warning messages current version of xman. The last and largest section is the information display. This section of the application contains a list of manual pages to choose from or the text of a manual page.

To use xman pull down the Sections menu to select a manual. When the section is displayed, click the left pointer button of the manual page that you would like to see. Xman will direct page listing with the manual page you selected.

That should be enough to get you started. Once you understand the basics of how to use xman, take a look at the rest of this manual: the advanced features that are available to make using xman efficient.
Event handling

From X:
- while loop with xcb_wait_for_event() running in a separate thread
- Events passed to WM through WM provided callback function

From WM:
- Passed to X through XCWM function calls
- WM provides asynchronous dispatching if necessary
Client window mapped

XCB_MAP_NOTIFY received from server

Determine if window is being managed

Query attributes of window

Send window created event through callback

size
desition
override-redirect
WM_NAME
WM_SIZE_HINTS
_NET_WM_WINDOW_TYPE
...

XCWM

Query window attributes through accessors

Create NSWindow based on attributes

Render new window to screen

XtoQ
Damage event and window update path

**XCWM**

1. Damage event received from server
2. Determine damaged bounds and update window data
3. Damage event sent through callback function
   - Get damaged bounds of window.
   - Damaged area of window marked for update

**XtoQ**

1. Get copy of pixmap from server
   - Get pixmap for area to redraw
   - Redraw called on damaged area
   - Render pixmap to screen
2. Redrawn area subtracted from window damage object

**XCWM event thread**

**AppKit window update thread**
Keyboard and mouse input

Input event caught by AppKit → Event type determined → XCWM handler called using asynchronous dispatch

XtoQ → XCWM

XTEST call to inject input event

Server
Determine input destination client from focused window / mouse position
Future work

- More ICCCM / EWMH protocol support
- Rethink event loop
- XtoSomethingElse
- Replace XTEST
Want to contribute?

**XCWM git repository:**

http://cgit.freedesktop.org/xorg/lib/libxcwm/

**Contact:**

Jess VanDerwalker:  jvanderw@freedesktop.org

washu@sonic.net