i3 - an improved dynamic tiling window manager

sECuRE beim NoName e.V.

powered by LaTeX, of course

12. März 2009

Geschichte

- „All window managers suck, this one just sucks less"?
- Desktop environment vs. window manager (GNOME, KDE, Xfce, ...)
- Stacking (e17, fluxbox, IceWM, fvwm, ...) vs Tiling (dwm, wmii, xmonad, ...)
- dwm, awesome, xmonad, ...: statisches Layout
- wmii, ion: dynamisches layout
- Problem an ion: tuomov (Lizenz, Kommunikation), Config, Look and feel
- Probleme an wmii: Xinerama-support, Xlib, undokumentierter code, nur Spalten, keine Reihen, Kleinigkeiten (titellose Fenster)

Screenshots!

Drücken Sie Mod1+2 um diese Demo zu starten.

Merkmale

- gut lesbarer, dokumentierter Code. Dokumentation.
- XCB anstelle von Xlib
- Xinerama done right^{TM}
- Spalten und Zeilen
- command-mode, wie in vim
- UTF-8 clean
- kein Antialiasing, schlank und schnell bleiben

Typische Kommunikation mit X

- Verbindung aufbauen
- Requests über die Leitung schicken (Fenster erzeugen)
- Eventloop starten, reagieren (Fenster zeichnen, Eingaben, . . .)

Was genau macht ein WM?

- Events umlenken
- Neue Fenster anzeigen/positionieren (MapRequest)
- Titelleisten malen (reparenting)
- Den Fokus verwalten
- Mit Hints umgehen (Fenstertitel, Fullscreen, Dock, . . .)
- Auf Benutzereingaben reagieren

Was an X toll ist

- Man hat an Window Manager gedacht (Mac OS X *hust*)
- Netzwerk-transparent (debugging, xtrace)
- Das Protokoll ist gut designed (Extensions möglich, simpel)

Protokoll, Beispielcode

```
1  int handle_map_request(void *prophs, xcb_connection_t *conn,
2                         xcb_map_request_event_t *event) {
3    xcb_get_window_attributes_cookie_t cookie;
4    xcb_get_window_attributes_reply_t *reply;
5
6    cookie = xcb_get_window_attributes_unchecked(conn, event->window);
7
8    if ((reply = xcb_get_window_attributes_reply(conn, cookie, NULL))
9        == NULL) {
10         LOG("Could not get window attributes\n");
11         return -1;
12   }
13
14   window_attributes_t wa = { TAG_VALUE };
15   wa.u.override_redirect = reply->override_redirect;
16
17   add_ignore_event(event->sequence);
18
19   manage_window(prophs, conn, event->window, wa);
20
21   return 1;
}
```

Was an X nicht so toll ist

- Einige race conditions vorhanden
- Man kann nicht fein genug angeben, welche Events man gerne hätte
- Xlib ist ziemlich eklig, aber es gibt ja xcb
- Bugs: Keyboard state wird nicht richtig übermittelt
- Ich empfehle auch: „Why X is not our ideal window system”
  `http://www.std.org/~msm/common/WhyX.pdf`

XCB

- `http://xcb.freedesktop.org/`
- „X-protocol C-language Binding”
- Klein, wartbar (aus einer Protokollbeschreibung auto-generiert)
- Sinnvoll benannte Funktionsnamen und Datentypen
- Threadsafe (nicht dass wir das bräuchten, aber. . .)
- Nutzt die Asynchronität von X aus
- Dokumentation? Ne, das ist was für Anfänger.
- xcb-util: XCB noch mal ein bisschen gekapselt, nützliche Funktionen abstrahiert

Xlib-Beispielcode

```
1  char *names[10] = {"_NET_SUPPORTED", "_NET_WM_STATE",
2  "_NET_WM_STATE_FULLSCREEN", "_NET_WM_NAME" /* ... */};
3  Atom atoms[10];
4
5  /* Get atoms */
6  for (int i = 0; i < 10; i++) {
7    atoms[i] = XInternAtom(display, names[i], 0);
8  }
```

XCB-Beispielcode

```
1  char *names[10] = {"_NET_SUPPORTED", "_NET_WM_STATE",
2    "_NET_WM_STATE_FULLSCREEN", "_NET_WM_NAME" /* ... */};
3  xcb_intern_atom_cookie_t cookies[10];
4
5  /* Place requests for atoms as soon as possible */
6  for (int c = 0; c < 10; c++)
7    xcb_intern_atom(connection, 0, strlen(names[c]), names[c]);
8
9  /* Do other stuff here */
10 load_configuration();
11
12 /* Get atoms */
13 for (int c = 0; c < 10; c++) {
14   xcb_intern_atom_reply_t *reply =
15     xcb_intern_atom_reply(connection, cookies[c], NULL);
16   if (!reply) {
17     fprintf(stderr, "Could not get atom\n");
18     exit(-1);
19   }
20   printf("atom has ID %d\n", reply->atom);
21   free(reply);
}
```

Xft

- „X FreeType”, library um antialiased fonts zu benutzen
- Benutzt man am besten mit Pango (rendert fonts) und Cairo
- Keine Möglichkeit, pixel fonts zu benutzen („x core fonts”)
- Was macht man (urxvt) also? Beide APIs benutzen, fallback fonts
- Was machen wir (i3)? misc-fixed-*-iso10646!
  ISO 10646 = Universal Character Set, selbe Zeichen wie Unicode
- Fontconfig/xft soll wohl x core fonts ablösen :-(

Ein paar Zahlen

- 6118 Zeilen Code, Dokumentation, Website, READMEs
- ~ 2800 Zeilen Sourcecode
- 73 KB groß (ohne debug symbols)

Benutzen

- git clone `git://code.stapelberg.de/i3`
- Debian: cd i3; dpkg-buildpackage; sudo dpkg -i ../i3-wm*deb
- non-Debian: cd i3; cat DEPENDS; make; sudo make install
- in ~/.xsession: exec /usr/bin/i3
- Siehe manpage i3(1)

Mehr infos

- git-webinterface: `http://code.stapelberg.de/git/i3`
- Website: `http://i3.zekjur.net`
- IRC: #i3 auf irc.twice-irc.de