Input handling in wscons and X

Matthieu Herrb

OpenBSD/X.Org

EuroBSDCon, October 18, 2008

`http://www.laas.fr/~matthieu/talks/eurobsdcon2008.pdf`

Plan

Agenda

Introduction

Input device : the other important thing for desktop machines.

- Long history
- Components :
  - Kernel,
  - X server,
  - X libraries,
  - applications.
- New kind of devices : touch-pads, touchscreens, remote controls, multi-touch devices etc.

What is an input device ?

$\boxed{\longrightarrow \text{ a device that allows a user to input data to the computer...}}$

Devices that produce data from the environment of the computer (cameras, temperature or fan speed sensors, GPS, accelerometers) are not considered in this talk.

Sample input devices

- serial console
- keyboard
- mouse
- joystick
- tablet
- touchscreen
- remote control

$\longrightarrow$ abstraction : input events : keys, button, valuators,....

Agenda

The tty interface

Text consoles (teletypes, glass consoles ex. VT100)

2 layers in a tty driver :

- RS232 itself (speed, width, parity, flow control...),
- the line oriented character based protocol.

Implementation : POSIX termios(7).

The graphical workstation

Graphical display + keyboard + mouse

Generally seen as different devices by the kernel.

Examples : SunOS vuid devices, HP HIL, etc.

The PC world

Three generations :

- PC/XT and AT keyboard controller. Mouse was separate (RS232 or proprietary bus interfaces).
- PS/2 style. Common (but specific) driver for one keyboard and one mouse. Not hot-pluggable.
- USB specific keyboard and mouse device classes. Hot-pluggable.

First generations of Unix for PC had only basic input devices handling capabilities. Most of the work was done in XFree86.

Today’s systems

- Generalized USB keyboards and mices.
- More protocol handling done in the kernel (Linux evdev driver, wscons, etc.)
- Hot-pluggable

New devices

Touch-screens are becoming more and more popular.

New technologies expand touch-screens possibilities :

Touch screens specific issues

- Screen and input device bound together.
- Calibration needed.
- Pointer : visible or not ?
- no motion-only events : only clicks and drag
- stylus or finger ?
- multitouch...

Agenda

wscons

Originally written by Chris Demetriou and Matthias Drochner in NetBSD. Also used by OpenBSD.
Provide a high-level console driver.
Features :

- Text consoles with tty interface and terminal emulation (VT220)
- Uses either text or bit-mapped from the display device
- events interface for input devices : keyboards, mice, touch-pads, tablets, touch-screens
- attaches AT, PS/2, USB, proprietary devices

API : `<sys/dev/wscons/wsconsio.h>`

wscons and multiple devices

Multiple devices and hot-plug are handled :

- one device node per device,
  - `/dev/wskbd0, /dev/wskbd1, etc.`
  - `/dev/wsmouse0, /dev/wsmouse1, etc.`
- through *muxes* devices which multiplexes all events to one single device node :
  - `/dev/wskbd`,
  - `/dev/wsmouse`

Problem : muxes completely hide the multiple devices from user-land.

WScons configuration

Two utilities :

wsconscfg	configures wsdisplays and console emulation
wsconsctl	configures input device parameters for console mode.

- Keyboard mapping
- Keyboard auto-repeat
- Mouse parameters
- Screen saver

WScons : current and future work

NetBSD :

- synaptics touch pad support recently added.
- touch screen support.
- UTF-8 support.

OpenBSD :

- touch screen support
- legacy keyboard types

All : *write/update documentation...*

Agenda

Legacy X input : core devices

- One keyboard,
- One mouse with at most 5 buttons.

Processed between :

DDX :	Device Dependant X
DIX :	Device Independent X

Keyboard handling

- DIX expects key up/down events from DDX, using *keycodes* to represent the keys.
- keycodes are then translated to *keysyms* and presented to clients.
- XFree86 pushed to standardize keycode among all DDX, using the AT keyboard codes found in `atKeynames.h`.
- Autorepeat can be handled by the X server, or just use the hardware feature if present.
- libX11 (client-side) translates keysyms sequences to strings (ASCII, Latin-1, UTF-8 or whatever) and other events.

WSkbd and the DDX keyboard driver

Two ways to handle wscons keyboards in X

- through the wsdisplay device (`/dev/ttyC0` or `/dev/ttyE0`) in raw mode.
  advantages : standards keycodes are seen by X, all keyboards supported by the kernel are useable under X, no extra configuration
  drawbacks : X sees only one keyboard, one layout.
- through the wskbd device (`/dev/wskbd0`) directly.
  advantages : separate layouts
  drawbacks : need to configure the device explicitly.
  X needs to be taught about new wscons keyboard types.

Pointer devices

Mouse, touchpad (relative coordinate events) :

- PS/2, USB devices handled by *wsmouse(4)*.
- Serial devices handled explicitetly by the X mouse driver.
- the X mouse driver knows several protocols.

Tablets, touchscreens (absolute coordinates events) :

- can either be handled by wsmouse (w/ support for absolute coordinates)
- or by a specific X driver

The Linux evdev driver

Recent Linux kernel and X use a generic event model, similar to wscons. Some differences :

- Linux events device identify themselve as mouse, keyboard, etc.
- Only one X driver (xf86-input-evdev) manages all kind of hardware
- xf86-input-evdev uses HAL to manage hotplug and configuration of individual devices. Configuration moves to `.fdi` files.

Merging wscons support in xf86-input-evdev has been suggested, but probably not worth the pain.

The Xinput extension

Goal : drop the restrictions on input devices :
$\longrightarrow$ many keyboards, pointer devices

- Xinput is not optional anymore
  The mouse and keyboard drivers are Xinput drivers.
- Xinput is being cleaned up / rewritten
- now provides support for input hot-plug, device properties, MPX.

Input device properties

Arbitrary type values attached to an input device, can by modified at run-time.

- set middle mouse button emulation
- configure new pointer acceleration strategies
- calibration data for touchscreen drivers
- etc.

$\longrightarrow$ coming in xserver 1.6

The XKB extension

Goal : add configurability to keyboard mappings.

Future work :

- clean-up code, remove dead code.
- merge xkbcomp functionality into the X server.
- remove all pre-xkb input handling
  (XKB becomes mandatory).

Multi-head / multi-seat

How to bind input devices to heads....

- code exists in X to handle this under Linux
- no way to configure multiple independent wsdisplay devices in wscons

Special case : touchscreens – the input device is physically attached to a particular screen...

MPX

Multi **P**ointer **X**

- Virtual pointers - cursors
  - attached to zero or more physical devices
  - provide the events to the applications
- Virtual keyboards - focus
  - attached to zero or more physical devices
  - provides the events to the applications

$\longrightarrow$ coming into xserver 1.6

Toolkits

Xaw/Motif/Gnome/KDE/...

- interpret the X events and convert them to action
- handle things like double-click, mapping wheel events to scroll commands, etc.
- multi-touch or gesture handling belongs to toolkits or applications.

Agenda

User experience

Problems from the end-user point of view that need improvements :

- keyboard layouts
- mouse emulation
- mouse actions configuration
- touch-pad gestures
- touchscreen calibration
- latency between input and output

Conclusion

- Input plays an important role in user interfaces too
- Some new things : new device software evolutions

Bibliography

- Daniel Stone *Input Overview and Plans* `http://www.fooishbar.org/talks/xds2007-input.pdf`
- Peter Hutterers' Blog : `http://who-t.blogspot.com/`
- Tiago Vignatti *Improving X input latency* `http://www.inf.ufpr.br/vignatti/talks/XDS08-ImprovingInputLatency.pdf`

Questions ?