xf86-input-joystick

Sascha Hlusiak

XDC2012, Nürnberg

abstract

xf86-input-joystick?

- **not** meant for playing games under X
- input module for the X.Org server to handle classical joysticks
- can emit pointer movement and key events
- **control the cursor with a joystick**

overview

why joysticks?

- cheap remote control
  - being lazy
  - watching movies from couch
  - control music
- interactive kiosk systems
  - replace fragile mouse/touchpad
- console and media center systems
  - Linux on XBox, PS2, ...
  - *Steam Big Picture* going there

classical joysticks

- a lot of different hardware configurations
- easily 6 or more axes, different types
  - directional pad
  - hat switch
  - analog sticks
- bunch of buttons, easily 10 and more
- force feedback

why a joystick input module?

- xorg-server knows pointers and keyboards
- **xf86-input-evdev** maps directly
- evdev only in Linux
- joysticks usually send absolute axis data
  $\Rightarrow$ direct mapping useless
  $\Rightarrow$ joystick specific event transformation needed

$$\underbrace{\begin{pmatrix}\text{absolute axes}\\ \text{buttons}\end{pmatrix}}_{\text{joystick data}} \Rightarrow \underbrace{px(a_i) = \int_t f(a_i)\frac{\text{px}}{\text{s}}}_{\text{transformation}} \Rightarrow \underbrace{\begin{pmatrix}\text{pointer}\\ \text{buttons}\\ \text{keys}\end{pmatrix}}_{\text{X events}}$$

joysticks on kernel level

kernel device

- usually Plug & Play (USB)
- event driven ⇒ no polling

Linux

- `/dev/input/js0` (joydev)
- `/dev/input/event0` (evdev)

(Free)BSD

- `/dev/input/js0` (linux-js)
- `/dev/uhid0` (usbhid)

⇒ lightweight abstraction layer

axis mappings

mapping mode

- relative (analog)
- accelerated (D-Pad)
- absolute (analog)

mapping type

- pointer movement
- scroll event
- key event sequence (e.g. cursor keys)
- none + raw valuator events (XI2)

button mappings

- pointer click
- pointer movement (accelerated)
- scroll events (accelerated)
- key sequences (e.g. Alt+Tab)
  - auto repeat applies
  - keyboard layout applies
- disable all events temporarily
  - still allows playing games

hotplugging and configuration

- can be hotplugged through udev and *xorg.conf.d*
  - allows specific configuration
  - easy pre-configuration through distributors
- no EVIOCGRAB
  - allows concurrent reads (e.g. for games)
  - watch out when hotplugging **js0** *and* **event0**
- supports input device properties
  - most configuration items can be changed at runtime
  - TODO: frontend

distribution default

- mostly installed unintentionally
  $\Rightarrow$ yields unexpected behaviour
  - default should be disabled/floating
  - make user aware to enable it during installation
  - popup from DE on hotplug event
  - way to activate/deactivate it on the fly
- no reasonable default configuration
  - example configuration
  - configuration wizard (*xorg.conf.d*)
  - frontend for runtime configuration (properties, dynamic hotplugging)

pointer vs. keyboard

- X.Org server strictly separates pointers and keyboards
- no hybrid input devices
  $\Rightarrow$ xf86-input-joystick has to create two input devices
  - properties only on pointer
  - keyboard layout and autorepeat on keyboard

keyboard configuration

- user wants KeySyms (space, Return, XF86AudioPlay, ...)
- driver emits scancodes
  - keyboard layout $\Rightarrow$ unwanted indirection layer
  - current layout unknown to driver
  - no custom keyboard layout from within driver
- different auto repeat rate possible
  $\Rightarrow$ difficult to configure

raw valuator events

- valuators 0 and 1 reserved for pointer movement
- optional raw valuators per axis
  - useful for XI2 applications (e.g. gimp)
  - less useful for games

platforms

- focus on Linux
  - end user desktop systems
  - interactive kiosk systems
  - console systems (XBox, PS2/3, ...)
  - Wii remote? Kinect?
    $\Rightarrow$ not classical joysticks
    $\Rightarrow$ different module
- support for Solaris, etc. possible
  $\Rightarrow$ not attractive enough

force feedback

- events from toolkit
- event infrastructure from client to server

possible features

- most properties implemented
  - still some work left
- **xinput** only “frontend” for properties
  - driver can deactivate/activate itself
  - GTK/Qt frontend with support for “profiles” would be nice

Thank you!