3D Input

Ioannis Tsiombikas

nuclear@member.fsf.org

May 9, 2008

Part 1: Intuitive 3D Interaction

Interacting with 3D Environments

Common input devices inadequate for 3D interaction.

- Keyboard (arrow keys?)
- Mouse
- Joystick / Pad
- Trackball
- Touchpad

How do we cope?

Design our programs to work well with limited degrees of freedom. Example: most 3D computer games.

Come up with complicated user interaction schemes. Example: most 3D modelling software.

The real solution: 6-dof input devices

6 Degrees of Freedom

6-dof devices allow full 3D control: translation along all 3 axes, and rotation around all 3 axes.

3Dconnexion drivers

The space navigator is a USB HID device. All input events are forwarded to user space through the linux event interface (evdev).

3Dconnexion provides:

- *3dxsrv*, a user space driver (daemon) apps talk to, in order to get 6dof events.
- *magellan*, an SDK (library) apps may use to communicate with the daemon.

Problem is. . .

. . . unfortunately, they both SUCK.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

What's wrong?

What's wrong with the driver?

- It's proprietary, binary only, 32bit x86 only.
- It uses a convoluted X11-based protocol to talk to clients.
- It depends on motif (?!)
- It doesn't work most of the time . . .

What's wrong with the SDK?

- It's proprietary, source given with restrictive licensing terms.
- The API is an ugly, inconsistent mess.

The *spacenav* project

The spacenav project, is an effort to provide free software alternatives to both 3dconnexion's driver and SDK library.

`http://spacenav.sourceforge.net`

Spacenav project: *spacenavd*

Spacenavd is the free replacement for the 3dxsrv driver.

- Free software, released under the GNU GPL v3.
- New simpler, and more sane, application communication protocol, based on unix domain sockets.
- Supports the 3d connexion X11 protocol, making it a transparent drop-in replacement for the proprietary driver.
- Separate GUI configuration tool available.

Spacenav project: *spacenavd*

Spacenavd is the free replacement for the 3dxsrv driver.

- Free software, released under the GNU GPL v3.
- New simpler, and more sane, application communication protocol, based on unix domain sockets.
- Supports the 3d connexion X11 protocol, making it a transparent drop-in replacement for the proprietary driver.
- Separate GUI configuration tool available.

Spacenav project: *spacenavd*

Spacenavd is the free replacement for the 3dxsrv driver.

- Free software, released under the GNU GPL v3.
- New simpler, and more sane, application communication protocol, based on unix domain sockets.
- Supports the 3d connexion X11 protocol, making it a transparent drop-in replacement for the proprietary driver.
- Separate GUI configuration tool available.

Spacenav project: *spacenavd*

Spacenavd is the free replacement for the 3dxsrv driver.

- Free software, released under the GNU GPL v3.
- New simpler, and more sane, application communication protocol, based on unix domain sockets.
- Supports the 3d connexion X11 protocol, making it a transparent drop-in replacement for the proprietary driver.
- Separate GUI configuration tool available.

Spacenav project: *libspnav*

Libspnav is the free replacement SDK library.

- Free software, released under the 3-clause BSD license.
- Works with both spacenavd *and* the proprietary 3dxsrv daemon.
- New simple, orthogonal API.
- Source-compatible magellan API wrapper. Existing apps may be recompiled with libspnav to remove magellan licensing terms.

Spacenav project: *libspnav*

Libspnav is the free replacement SDK library.

- Free software, released under the 3-clause BSD license.
- Works with both spacenavd *and* the proprietary 3dxsrv daemon.
- New simple, orthogonal API.
- Source-compatible magellan API wrapper. Existing apps may be recompiled with libspnav to remove magellan licensing terms.

Spacenav project: *libspnav*

Libspnav is the free replacement SDK library.

- Free software, released under the 3-clause BSD license.
- Works with both spacenavd *and* the proprietary 3dxsrv daemon.
- New simple, orthogonal API.
- Source-compatible magellan API wrapper. Existing apps may be recompiled with libspnav to remove magellan licensing terms.

Spacenav project: *libspnav*

Libspnav is the free replacement SDK library.

- Free software, released under the 3-clause BSD license.
- Works with both spacenavd *and* the proprietary 3dxsrv daemon.
- New simple, orthogonal API.
- Source-compatible magellan API wrapper. Existing apps may be recompiled with libspnav to remove magellan licensing terms.

Project status

Both spacenavd and libspnav work fine as replacements of their proprietary counterparts on linux.

Future plans:

- Support more UNIX systems (FreeBSD is the next target).
- Support alternative event sources (XInput, old serial spacemice).
- Provide greater configurability.
- Push the driver and SDK to GNU/Linux distributions.

Project status

Both spacenavd and libspnav work fine as replacements of their proprietary counterparts on linux.

Future plans:

- Support more UNIX systems (FreeBSD is the next target).
- Support alternative event sources (XInput, old serial spacemice).
- Provide greater configurability.
- Push the driver and SDK to GNU/Linux distributions.

Part 2: Immersion

VR head-tracking

The process of actively tracking the user's head position in relation to the screen, and translating the user's head motions into corresponding movement of the viewpoint in the virtual environment.

My VR head-tracking experiment

Inspired by: CAVE, wii-remote-guy.

Main objective: simplicity. No need for expensive electromagnetic sensors (CAVE), or exotic input devices (wii). Commodity hardware should be used to achieve the same effect (web camera).

Three steps:

- Capture image from web camera.
- Process image to find the 2D image coordinates of the user's head.
- Calculate 3D position relative to the screen, and set it as the new 3D environment viewpoint.

My VR head-tracking experiment

Inspired by: CAVE, wii-remote-guy.

Main objective: simplicity. No need for expensive electromagnetic sensors (CAVE), or exotic input devices (wii). Commodity hardware should be used to achieve the same effect (web camera).

Three steps:

- Capture image from web camera.
- Process image to find the 2D image coordinates of the user's head.
- Calculate 3D position relative to the screen, and set it as the new 3D environment viewpoint.

My VR head-tracking experiment

Inspired by: CAVE, wii-remote-guy.

Main objective: simplicity. No need for expensive electromagnetic sensors (CAVE), or exotic input devices (wii). Commodity hardware should be used to achieve the same effect (web camera).

Three steps:

- Capture image from web camera.
- Process image to find the 2D image coordinates of the user's head.
- Calculate 3D position relative to the screen, and set it as the new 3D environment viewpoint.

My VR head-tracking experiment

Inspired by: CAVE, wii-remote-guy.

Main objective: simplicity. No need for expensive electromagnetic sensors (CAVE), or exotic input devices (wii). Commodity hardware should be used to achieve the same effect (web camera).

Three steps:

- Capture image from web camera.
- Process image to find the 2D image coordinates of the user's head.
- Calculate 3D position relative to the screen, and set it as the new 3D environment viewpoint.

Step 1: Video frame capture

Created a reusable, fairly generic, webcam library: “libwcam”.

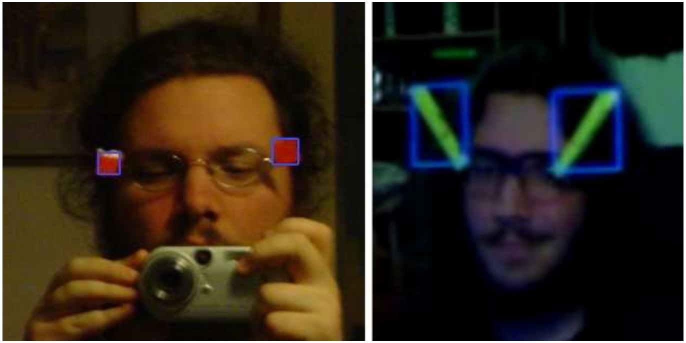
Captures frames from a webcam (or any other video input device) using *video4linux2*.

Important considerations for video frame capture:

- Fast framerate is needed (30 fps ideally).
- Auto exposure, and auto white balance will screw up the image processing algorithm, so they must be disabled.

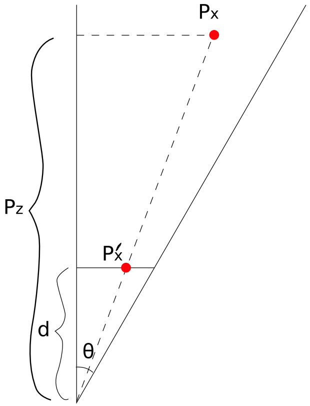
Step 2: Image Processing

Devised a very simple algorithm to detect two clusters of similarly-colored pixels in the image within a given tolerance.



Step 3: Calculate 3D head position

- Use image distance between the two points to calculate depth.
- Invert the webcam projection to get 3D position.



Implementation details

Program A

- Runs continuously grabbing frame off the camera at a fixed framerate.
- Processes the image and extracts 2 points.
- Allows user interaction to set target color, threshold etc.
- Listens to a UNIX socket for other processes interested in these points, and feeds them with the data as soon as they are available.

Program B

- Runs the OpenGL virtual environment.
- Connects to program A through the UNIX socket and as soon as it gets the data through the socket it calculates 3D position and updates display.

Implementation details

Program A

- Runs continuously grabbing frame off the camera at a fixed framerate.
- Processes the image and extracts 2 points.
- Allows user interaction to set target color, threshold etc.
- Listens to a UNIX socket for other processes interested in these points, and feeds them with the data as soon as they are available.

Program B

- Runs the OpenGL virtual environment.
- Connects to program A through the UNIX socket and as soon as it gets the data through the socket it calculates 3D position and updates display.

The End

Thank you for listening.
Questions ?