Emulating a laser pointer over X windows

an fvwm implementation

Lucio Chiappetti IASF Milano 14 Sep 09 14:59

Introduction Operation Download Updates

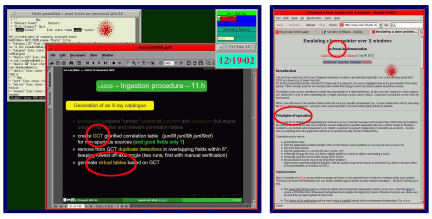
Introduction

I do not know about you, but to me it happens sometimes to make a presentation (typically I use Latex-Beamer generated PDF) or a demo (e.g. of some web site).
If I do the presentation locally, with the PC connected to a projector, it is easy to highlight items in the presentation with a laser pointer. This is usually (more or less barely) more visible than trying to move the mouse pointer around the screen.

Nowadays it also occurs sometimes to make the presentation in a videoconference. In this case one cannot use a laser pointer, so I looked for a way to have something like a bright and large screen cursor which I could move around the screen with my local mouse.

What I describe here is the solution which works for me in my specific environment (i.e. a Linux workstation with X, and using the fvwm window manager). I also give some (untested) hints I received while trying to find my solution.

Here are some snapshots of "pointers" of different shape (all red) superimposed over various windows.



Principles of operation

Since I am not so skilled in X programming to write new cursors (and also because such cursors may work in the root window, but will not necessarily be able to override the cursors defined by a specific application, like the Acrobat reader or Firefox), I decided to use instead semi-transparent icons which I associate to a generic application (I currently use an xterm ... but one can use anything since the application will never be opened, but only used in iconized form) and can then move around.

I therefore need a way to

- generate the icons
- start the application iconized (ideally at the current mouse cursor position) (i.e. turning the laser pointer on)
- move the icon around
- stop the application (i.e. turning the laser pointer off)
- eventually change the icon (e.g. from a simple pointer to a circle or ellipse surrounding a word)
- eventually park the icon in some corner of the screen
- the icon shall of course stay on top of all other windows
  and its mouse operation shall not interfere with the normal usage of the mouse or keyboard (e.g. click to advance slides in the presentation, or to follow an URL)

Implementation

Since I normally use fvwm as my window manager of choice, it was natural for me to look for a solution under such context.
This does not mean that somebody else can adapt a similar logical solution to other window managers ... but that's entirely up to you.

- The generation of the icons is a relatively simple and environment-independent choice. In my case I used the gimp to create a PNG file with a transparent background and a simple drawing of my choice. Whatever tool you use, make sure to record the xy sizes in pixels of

your icons.

- The choice of the application and the way to start it iconified should also be environment independent. E.g. `xterm -iconic`
- The way to associate an (initial) icon to (an instance of) the application is instead fvwm-dependent. If I start the application with a particular name resource `xterm -name Laser -iconic`, I can then use a style named Laser to make the association
- To make sure that the icon appears in the place I want, and not parked in some position decided by the window manager, to make sure the icon has no disturbing title string, and that it is displayed above any other window, are instead all window manager dependent issues. In fvwm they are (mostly) dealt with the definition of the style, but if you use other window managers you have to find your ways.
- Also the way to associate the actual start of (an instance of) the application with an user operation (an item in the root menu, a combination of mouse clicks, a combination of functional keys) is window manager dependent.
  A root menu item is not particularly useful for typical presentations, since one is probably displaying the presentation in full screen and the root menu is not available.
  Also simple mouse button clicks are likely to interfere with other operations (one should probably use some unusual modifiers, like Meta-MB1 or Alt-MB2 or Ctrl-MB3)
  Usage of function keys is probably the simpler way.
- Similarly the way to interact with a running (instance of the) application to move the icon or make it disappear or change it, can all be associated to actions like pressing a function key, again a window manager dependent way.

Alternate implementations

I received two alternate (and window-manager independent) suggestions to solve this problem, which I haven't tested since I was already half-way my solution which does not require to install anything but just exploits fvwm.

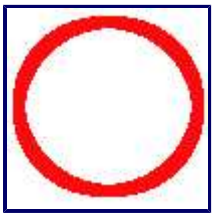
- Jonathan Kotta suggested the use of gromit (looks promising though perhaps heavier), a program which draws on top of an X screen
- Viktor Griph described a way (instructive) to preload custom cursors onto an application, I am taking the liberty to point to where his posting is archived

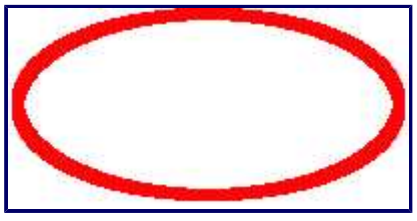
Command interface

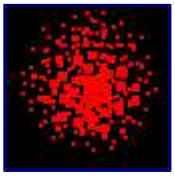
- I defined the following ways to switch the laser pointer on
  - From a root menu item (mainly for testing)
  - Using shift-MB1 with the mouse
  - The above cases start with the current Laser style
  - shift-F4 starts a circular cursor centered at the mouse position
  - shift-F5 starts an elliptical cursor centered at the mouse position
  - shift-F6 starts a "laser beam" cursor centered at the mouse position
  - the above cases should start on top of any window
- The following keys works when the mouse pointer is in the active area of the icons (i.e. the non-transparent part)
  - F1 initiates a movement of the icon (terminate by clicking anything)
  - F2 parks the icon at the bottom right corner of the screen
  - F3 switches the laser off (i.e. terminates the application)
  - F4 changes the shape of the pointed icon to circular
  - F5 changes the shape of the pointed icon ton elliptical
  - F6 changes the shape of the pointed icon to "laser beam"

Download and installation

- The only items which may possible require downloading are the **icons** (unless you want to create your own). These are the three sample icons I use in their natural size. Just use your browser to retrieve them.


100×100 pix circle


200×100 pix ellipse


80×80 pix "laser beam"

The icon files shall be stored in some directory, like a place where you keep other personal icons, or the directory where the `.fvwm2rc` configuration file is.

- The remainder of the "installation" just consist in copying lines into the `.fvwm2rc` configuration file.
- The next step is to **make fvwm aware of such directory** which is achieved just making sure it is appended to the ImagePath line in the `.fvwm2rc` configuration file.

  ImagePath .....:/<yourdirectorypath>
- The next step is to choose the application (I use xterm), and to **associate one of the icons** to it. One has also to define the **command used to start** the application.
  I start the application iconified, and with a special name resource, which in fvwm can be achieved with a run string like Exec xterm -name Laser -iconic
- In order to make sure the **icon stays always on top** of all other windows on all desktops, has no associated title but just the icon, is not parked in an eventual icon box but is created at the mouse position in my case I use a particular style LaserBase defined in `.fvwm2rc`.
  The example works starting from my pre-existing `.fvwm2rc`. Tuning might be necessary in your case.

  The potential styles LaserA LaserB LaserC are de facto not used but just define the possible association of the three icons. The only usage is in the next line. I give all three styles to allow you to choose a default.

  The style Laser is initialized to one of those and associates the (default) icon to the application exploiting the name resource.

  This is the relevant bit of `.fvwm2rc`:

  ```
  Style LaserBase StickyIcon, StickyAcrossDesks, Layer 9, !IconTitle, IconBox none,
  PositionPlacement UnderMouse
  Style LaserA Icon beam.png,  UseStyle LaserBase
  Style LaserB Icon circle.png,  UseStyle LaserBase
  Style LaserC Icon ellipse.png, UseStyle LaserBase
  Style Laser UseStyle LaserA
  ```
- To **start the application from the root menu** add this to the definition of it in `.fvwm2rc`:
  + "LaserPointer " Exec xterm -name Laser -iconic
- The rest of this stuff can all be grouped together at the end of your `.fvwm2rc`:
- This sequence of lines defines a **"laser switch on" function** which can, if called with an argument equal to A, B or C, start a pointer with the shape defined by the style LaserA LaserB LaserC or otherwise by the current Laser style.

  ```
  DestroyFunc PointLaser
  AddToFunc PointLaser
    + I Exec xterm -name Laser$0 -iconic
  ```
- As example, a line like this associates a default startup to shifted mouse button 1
  Mouse 1 A S PointLaser
  while a line like this associates a startup with the laser beam icon to key P when pressed in the root window
  Key P R A PointLaser A
  None of these lines is mandatory.
- Defining a **function key for laser switch on at current mouse position** is definitely trickier.
  One has first to define an auxiliary function which moves the newly created icon to the current mouse position, offset by half of the icon sizes (so that it is centered), and associates the particular icon. The function requires three pseudo-arguments passed via environment variables.

  ```
  DestroyFunc rePointLaser
  AddToFunc rePointLaser
    + I ThisWindow ("Laser*") Move m-$[xof]p m-$[yof]p
    + I ThisWindow ("Laser*") WindowStyle Icon $[ik].png
    + I UnsetEnv xof
    + I UnsetEnv yof
    + I UnsetEnv ik
  ```

  One has then to associate an event handler to activate the function when a new icon is created

  ```
  DestroyModuleConfig Event-PL: *
  ```

```
*Event-PL: Cmd Function
*Event-PL: add_window rePointLaser
```

Finally one can bind to a functional key the function with specific arguments. In our case we use shift-F4/5/6 and start all instances with the Laser style.

```
PointerKey F4 A S PipeRead 'echo "SetEnv xof 50"   && echo "SetEnv yof 50" && echo
"SetEnv ik circle"   && echo "PointLaser "'
PointerKey F5 A S PipeRead 'echo "SetEnv xof 100" && echo "SetEnv yof 50" && echo
"SetEnv ik ellipse" && echo "PointLaser "'
PointerKey F6 A S PipeRead 'echo "SetEnv xof 40"   && echo "SetEnv yof 40" && echo
"SetEnv ik beam"   && echo "PointLaser "'
```

- The definition of **function keys to change icon** with unshifted F4/F5/F6 is much simpler and operates on any instance due to the wildchar style

```
PointerKey (Laser*) F4 I N WindowStyle Icon circle.png
PointerKey (Laser*) F5 I N WindowStyle Icon ellipse.png
PointerKey (Laser*) F6 I N WindowStyle Icon beam.png
```

- The definition of **function keys to move, park and switch off** with unshifted F1/2/3 is even simpler

```
PointerKey (Laser*) F1 I N Move
PointerKey (Laser*) F2 I N Move -0 -0
PointerKey (Laser*) F3 I N Close
```

Update history

S/w V1.0 and web page established on 10 Sep 2009
Web page last updated on 14 Sep 09 14:59

Bugs

More a(n unavoidable) feature: once the "move" operation is initiated on the icon, the usage of the mouse and keyboard is inhibited (because they are grabbed by the window manager) until one clicks to terminate the move.

The way I handle styles is possibly clumsy.

Contacts

I have no time to actively mantain this software other than for my own use. However I will be glad to receive communication of problems (or improvements) to it at my e-mail address `lucio` in domain `lambrate.inaf.it`.

Acknowledgment

In addition to the untested alternate solutions quoted above, I gratefully acknowledge a lot of useful suggestions by Thomas Adam which were essential to make the present one work.