Immersion of XWindow applications into a 3D workbench

Alexandre Topol
Centre d’Etudes et de Recherche en Informatique
Conservatoire National des Arts et Métiers (CNAM)
292, rue Saint-Martin, 75003 Paris, France
+33 1 40 27 22 47 topol@cnam.fr

ABSTRACT

The coexistence of 3D applications within 2D window managers is still difficult. We experiment a real-time immersion of XWindow applications into a 3D scene and allow minimal interaction with them. We use the Xlib structures and routines to catch the position size, and graphical content of the widgets.

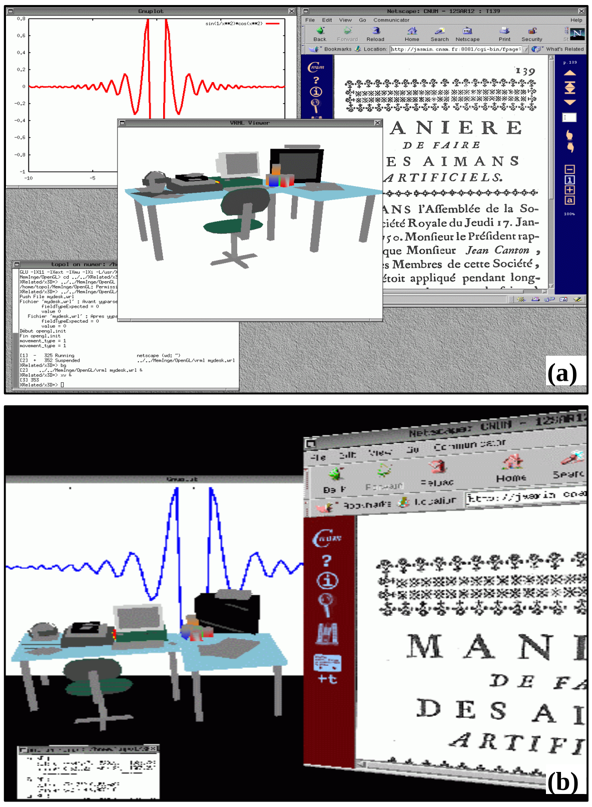
Keywords

3D GUI, 3D interaction, X protocol.

INTRODUCTION

There exists since a few years a general concern about the aging and limitations of the current GUI techniques. "Escaping flatland" seems among the promising alternatives. The work of Roberston et al. [6] has shown that 3D interfaces can improve organization and information filtering tasks by dynamically linking them to the navigation of the user. Carefully designed 3D metaphors could significantly ease the browsing through large (homogenous) information spaces. Although the ideal, general purpose, 3D workbench is still beyond reach, it is expected that it would necessarily coexists with the huge number of already developed applications using a 2D GUI. A similar problem occurred for the handling of keyboard-based applications (such as UNIX shells) when the XWindow system was developed.

Today's situation is unsatisfactory since flat windows and 3D scenes coexist inside a 2D window manager (possibly on different displays). The user must then switch back and forth to entirely different navigation methods. This is a frequent problem for the users of VRML browsers (fig. a). In [3], we generated on-the-fly VRML scenes for the walk through a digital library and we believe that our users will combine their reading work with consultations of other WWW-based digital libraries and writing with their favorite word processor. Other examples are collaborative VR environments, where the users would certainly base their interaction on 3D objects with the help of existing "flat" applications. A possible solution would be to revert the situation and immerse the flat window applications into the 3D scene, at user's will. We describe in this paper the



A 3D object in a 2D window (a) and 2D windows in a 3D workbench (b).

current state of development of such an application, based on the XWindow system (fig. b).

Related work

Space-scale interfaces like Pad++ [1] have investigated the use of 2D pans and zoom and demonstrated the importance of fluid interaction into an information space of unconstrained dimension. G. Leach et al. experienced a 3D window manager called MaW3 where 2D windows are placed inside a "tunnel" [5]. This project remained a prototype in which windows content stayed still. The tunnel metaphor constrains the user's navigation and therefore MaW3 can be considered as a 3D redrawing of the Pad++ interface. It is also uneasy to render additional 3D objects in this reduced volume. SpIn, a collaborative VR environment [4], offers a reasonable space to place 3D objects but the still focus only follows the rendering of a few (two) 2D windows. Our project should then be a mixture of each approach, offering unconstrained navigation (as with VRML browsers) within 2D windows and 3D objects.

A 3D "HACK" FOR THE XWINDOW SYSTEM

Our experiment is based on the XWindow system because its client/server organization allows the re-routing of the content of the structures stored by the X server. Client applications can also be executed on a remote workstation and this might be a useful feature if one want to use our 3D "hack" in a collaborative environment.

The first issue we investigated is the real-time capture of the application windows image (pixmap) and its rendering in a 3D scene using the Linux, OpenGL based, 3D engine we have developed in [3]. It is clear that the recovery of the pixmap itself is not a difficult issue since all screen grabbers (such as `xwd`) do so. The time requirement is however small in order to provide a fluid animation. Although it might appear to be a surprising method and a waste of computing power at first glance, one could argue that AT&T VNC [7], a widely used cross-platform screen emulator, is based on such method. Another problem is due to the OpenGL texture format, which requires a $2^n * 2^m$ pixel plane. Two solutions have been considered : get the window image and then scale it up to the required dimension, or re-scaling the window size before fetching the image. The first approach proved to be unrealistic due to the time cost of the operation, which is repeated for each frame and each window. The second method is used in the initialization part of our application. All the children of the root window are identified with a call to `XqueryTree` and then scaled to appropriate dimensions using calls to `XgetGeometry`.

Building the 3D scene is the second step of the initialization phase. Each window in the 3D scene is represented by a rectangle primitive with a texture retrieved by a call to `XgetImage`. No anti-aliasing is yet performed. It should be mentioned that the 3D windows could be organized very easily in a 3D scene such as the MaW3 tunnel [5] or the Web Forager [2].

At the time of this writing, we have only considered user interaction based on the right button mouse clicks which generate the four events `EnterNotify, ButtonPress, ButtonRelease, LeaveNotify` in that order. We first have to convert the 2D screen coordinates of the cursor into the 3D coordinate system using `gluUnproject` which performs the inverse projection. The closest rectangle is found using the OpenGL Select Buffer. The 3D click coordinates are compared with the upper-left corner of the 3D rectangle containing the window. This vector is then used for the computation of a 2D point by linear projection. The corresponding 2D widget is found using `XqueryTree`. The four Xevents are then sent to the Xserver with proper location. A minimal window managing is implemented, allowing windows translation inside the XY-plane and rotations around the X or Y axis. The camera position is keyboard controlled.

Our 3D engine refreshes periodically the textures mapped on each 3D rectangle. When a click occurs in a 3D window it changes the real window content which is in turn reflected in the 3D window. Window pixmap are also captured when the OpenGL rendering engine is idle. This is useful for animated windows, but the observed refresh rate was roughly 1 sec. for our hardware configuration and this should be improved.

FUTURE WORK

We have shown that XWindow applications could be immersed into 3D scenes with minimal software development. Although the techniques involved are very crude, they can be handled by the computational power of today's workstations and 3D graphic cards. All other interactions implemented in the XWindow system should be considered, at least focus and keyboard events. Our prototypal application is a simple X client application, but this should evolve into both a window manager (which has the responsibility for the placement, size and reparenting of the other windows) and a new Xserver. The window manager could force a window to meet the OpenGL requirements for texture sizes. An extended Xserver could handle all the graphic primitives in 3D coordinates and share its pixmap store with the OpenGL engine textures. We shall next be able to implement more sophisticated 3D interaction techniques and compare inside the same workbench the effectiveness of the proposals described by other authors.

REFERENCES

1. Bederson B. Pad++: A Zoomable Graphical Interface, in *Proc. of CHI'94* (Boston, MA, April 24-28, 1994).
2. Card, S., Robertson, G. and York, W. The WebBook and the Web Forager : an Information Workspace for the World-Wide-Web, in *Proc. of CHI'96* (Vancouver, April 13-18, 1996).
3. Cubaud, P.,Thiria, C. and Topol, A. Experimenting a 3D Interface for the Access to a Digital Library, in Proc. of *ACM DL'98* (Pittsburgh, June 23-26, 1998).
4. Dumas, C., Saugis, G., Chailloux, C. and Viaud, M-L. A 3D Interface for cooperative work, in *Proc. of CVE'98* (Manchester, UK, 17-19 June 1998).
5. Leach, G., Al-Qaimari, G., Grieve, M., Jinks, N. and McKay, C. Elements of the Graphical User Interface, in *Proc. of INTERACT'97* (Sydney, Australia, July 1997).
6. Robertson, G., Card, S. and Mackinlay, J. Information Visualization Using 3D Interactive Animation, *CACM*. 36(4), April 1993, pp. 57-71.
7. VNC : Virtual Network Computing http://www.uk.research.att.com/vnc/