ICAT 2001
December 5-7, Tokyo, Japan
THE VIRTUAL REALITY SOCIETY OF JAPAN

pwm: Using 2-Dimensional Applications in an Immersive Multiscreen Display

Yoshisuke Tateyama[1], Tetsuro Ogi[1,2] & Michitaka Hirose[2]

[1]Gifu MVL Research Center, Telecommunications Advancement Organization of Japan, Techno-Plaza 211, 4–179–1, Sue, Kakamigahara, Gifu, 509–0108, Japan
[2]University of Tokyo
tateyama@acm.org

Abstract

This paper describes the pwm (plate window manager; Figure 1) system which enables a user to manipulate existing 2D GUI applications in the virtual 3D space projected by an immersive multiscreen display, in terms of both its significance, and its method of implementation. Also we show the ability of currently avaliable hardware to implement the pwm, and evaluated its performance. We have compared five different users with respect to their 2D GUI pointing performance at the pwm with a normal 2D GUI manipulation environment, which consists of a 2D display and a mouse. With our current pwm implementation, it takes 50 % more time for a user to complete a pointing task than in a 2D environment.

Key words: 2D GUI, VR, IPT, the X Window System, 6 DoF dragging

Fig. 1: A user operating 2D applications with the pwm in the COSMOS.

1. Introduction

Using recently developed 3D display technologies, a user can now experience an artificial, but very realistic 3D world. These displays are currently relatively scarce, but in the future, we will use 3D displays as standard display devices in the same way that we use 2D displays today. Compared with 3D displays, 2D displays are less expressive, but 2D window systems have many useful applications, and these are now so widespread that we have come to consider them as indispensable "tools". It is a very simple concept to use our 2D window system applications in immersive virtual 3D space without applying any modifications to their source codes, but it opens up a wealth of possibilities for both 3D application programmers and users.

For instance, in a scientific visualization system, programmers can concentrate on the task in hand while implementing existing 2D tools, such as graphing packages, parameter editing and so on. On the other hand, users can make use of facilities that may not have been considered by the programmers, such as sketching pictures of some of their ideas, writing text, obtaining information by browsing the web, table calculations, statistical calculations, and so on.

The human race has always liked to record a large amount of information onto 2D planes, such as rock walls, stones, paper, and so on. Today, we are nearly all familiar with tools like text editors, web browsers, drawing tools, multimedia authoring tools and so on. These are all essentially based in only two dimensions. Even when advanced 3D interaction techniques and usable true-3D applications are readily available in virtual 3D space, there will still be a need to handle some 2D objects within them.

There are some systems that can render 2D GUI plates as if they are in virtual 3D space[1][2]. Many of these use a 2D display, and they are usually based on a mouse and a keyboard. However, 2D GUI plates systems work extremely well in an immersive multiscreen display (IMD) such as CAVE[3], CABIN[4], and COSMOS[5].

Without a stereo display, any plates that do not face to the plane of the display will simply appear deformed, and it is very difficult for a user to visualize any information that is on them. With a stereo display, a user can see a slanted plate, because we can see a picture on a slanted picture frame easily in the real world so in a large field of view 3D stereo display, a user can see more plates within a limited resolution.

In addition, the user's head position must be tracked. When there is a necessity to look at different areas of a plate, the user will tend to gaze around the plate unconsciously and instinctively. Without head tracking, the user becomes conscious of the difference, and has to make

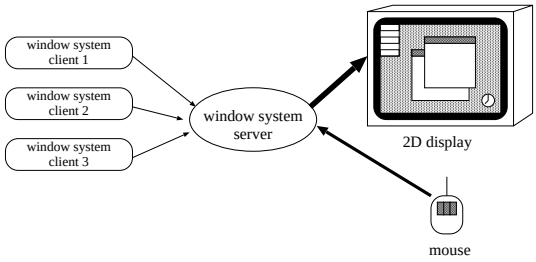

Fig. 2: A 2D window system employing a client-server architecture.

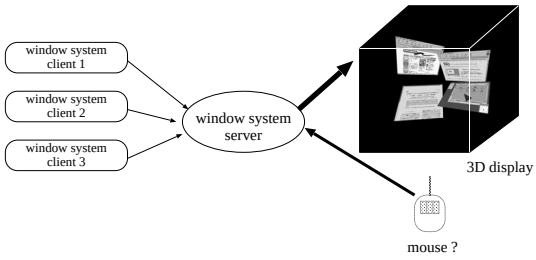

Fig. 3: A 3D window display server can provide utilities for existing applications.

an effort to manipulate that plate into a better orientation.

Therefore, each plate must be manipulated in a highly interactive manner. In 2D display-based systems, the user must manipulate plates in 3D space using a mouse. It is very difficult to manage 6 DoF with 2D input devices, and therefore 2D GUI plates in 3D space must be manipulated by 6 DoF input devices.

Our 2D GUI applications can be operated using a 2D GUI pointer. With a 2D display, there is a desk in front of the user, so it is easy to maniupulate a 2D GUI pointer with a mouse[1][6][2]. In an IMD, the user is usually standing up, so a mouse is not so easily available (Figure 1). The IMD user usually manipulates virtual objects with an input device which has a 6 DoF position sensor. In the `pwm`, a user can manipulate plates and can move a 2D GUI pointer device using a 6 DoF dragging technique. An appropriate input device has a 6 DoF sensor and more than one button. Using a 6 DoF dragging technique, while the "dragging" button is pressed, a target object can change it's orientation relative to its previous orientation. Compared with control techniques based on 6 DoF absolute positioning, this dragging technique requires less working space and less effort.

With this implementation of the `pwm`, a user can manipulate client applications of the X Window System without applying any modifications to their source codes.

Fig. 4: An Immersive six-screen display: COSMOS.

2. A 3D Window Display Server

Existing window systems, like the X Window System [7], employ a client/server architecture. One or more display screens are managed by a display server, and a server also manages the input devices, such as the keyboard and the mouse. All window system applications work as clients, which request to display their contents, and receive actions from the user (Figure 2). A 3D display server is basically a 2D window system display server that displays a window image on the surface of an object floating in a 3D space, instead of on a 2D display screen (Figure 3). Most applications will work without any modifications. `Pwm` is an example of a simple 3D window display server, with one or more rectangular plates floating in its 3D space. Each plate has the image of a 2D window. A user can manually place plates where she or he wants them within this virtual 3D space, and can simultaneously manipulate the window with a moving mouse pointer, or by clicking the mouse buttons.

3. An Input Device for a `pwm`

`Pwm` is intended for use in an immersive multiscreen display like COSMOS [5] (Figure 4), and an example of a `pwm` user is shown in Figure 1. In an immersive multiscreen display environment there is usually no provision for either desks or chairs. More importantly, in an immersive multiscreen display that has both a ceiling and floor screen, such as COSMOS or CABIN, desks should not be used at all, because they could easily break a ceiling screen. The user is therefore usually standing up, so using a keyboard or a mouse becomes very difficult.

In our system we use a proprietary NINTENDO 64 games controller (Figure 5) as the input device for the `pwm` . This has 14 different buttons, so applications programmers can easily add functions to their applications, bind functions to particular buttons, test them and so on. It also has a joystick, but the `pwm` does not make use of it. To produce the correct stereo images, the position of the viewer's eyes must be tracked. Usually, an immersive multiscreen display system incorporates one or more positioning sensors with 6 degrees of freedom (DoF), 3

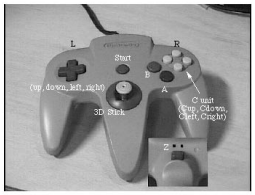

Fig. 5: A NINTENDO 64 controller.

Fig. 6: A 6 DoF drag technique.

DoF for position sensing and 3 DoF to monitor orientation. We attached one of these sensors to a NINTENDO 64 controller to work as a 6 DoF pointing device.

4. Interaction Techniques

In the `pwm`, a user can position plates manually, but not automatically. When a user manipulates 2D windows plates in 3D space, they can carry out at least three kinds of interactions:

(1) placement of each plate (including orientation)
(2) user-positioning in virtual space
(3) 2D GUI manipulations

In the `pwm`, the user can simultaneously move the position of a plate and its orientation, which we call a 6 DoF position. To move a plate to a new 6 DoF position, the user employs a 6 DoF drag technique, as shown in Figure 6. By using the 6 DoF dragging function on the controller, the user highlights the relative difference between the plate's new 6 DoF position and its previous one. The controller's Z button is designated as the "drag" button, that is, when a user presses that button, a plate starts moving. While he/she is pressing the drag button, a plate is moving continuously. When the user releases the button, the plate position is fixed to a new position in virtual space. To process this 6 DoF drag interaction, `pwm` handles the orientations of each object represented by using the relevant rotation quaternion [8][9]. Let the time when the drag button is pressed be t_0, the controller position at time t be $\mathbf{p_c(t)}$, the controller orientation rotation quaternion be $q_c(t)$, the target window plate position be $\mathbf{p_w(t)}$ and the target window plate orientation rotation quaternion be $q_w(t)$. While the user is dragging, the target window position:

$$(\mathbf{p_w(t)}, q_w(t))$$

is calculated by the following equations;

$$\begin{aligned} \mathbf{p_w(t)} &= \mathbf{p_w(t_0)} + \alpha(\mathbf{p_c(t)} - \mathbf{p_c(t_0)}) \\ q_w(t) &= q_c(t) q_c^{-1}(t_0) q_w(t_0) \end{aligned}$$

where α is the acceleration constant. Currently we set $\alpha = 5.0$.

When the user want to view a plate's content more precisely, he/she can pull the plate closer, or move his/her head closer to the plate. This hands-free interaction technique is available in 3D displays where it is possible to track the position of the viewer's head, such as immersive multiscreen display environments, head mounted displays and so on. These types of interaction are so natural for humans that a novice user would instinctively carry them out immediately when he/she wants to look at the plate, without the need for any explanation of the system.

A 2D GUI mouse pointer has 2 DoF, but the `pwm` uses a 6 DoF pointing device. Although the 3 DoF relating to rotation information can be ignored without any problems, the `pwm` must convert the remaining 3 DoF positioning information to a 2 DoF pointer position. The controller's 3 DoF position on a global coordinate system is mapped to the window plate's local coordinate system. Let the mouse pointer position on the plate coordinate system be $\mathbf{p_p(t)}$, the translation matrix for position $\mathbf{p}$ be $\mathbf{T(p)}$ and the rotation matrix for the quaternion q be $\mathbf{R}(q)$. While the user is dragging, $\mathbf{p_p(t)}$ is calculated by the following equation;

$$\begin{aligned} \mathbf{p_p(t)} = \quad & \mathbf{p_p(t_0)} \\ & + \beta \cdot \mathbf{T^{-1}(p_w(t))}\ \mathbf{R^{-1}}(q_w(t))\ (\mathbf{p_c(t)} - \mathbf{p_c(t_0)}) \end{aligned}$$

where β is the acceleration constant. Currently we set $\beta = 1500.0$ (pixels/meter).

In addition to these three kinds of interactions, the `pwm` has two other kinds of possible interaction. One is the process of selecting a plate, and the other is selecting the appropriate operation mode from "moving plate" mode and "moving a 2D GUI pointer" mode.

5. Current Implementation

The `pwm` uses the X Window System as it's 2D GUI window system. There are two ways in which this can be implemented:

- a real 3D display server that recognizes the X Window System protocols.
- a proxy display system that retrieves windows' images that are stored on an X server.

The `pwm` is implemented in the latter way (Figure 7). A proxy display system has no direct way of recognizing changes of a window's contents, but this has some merit.

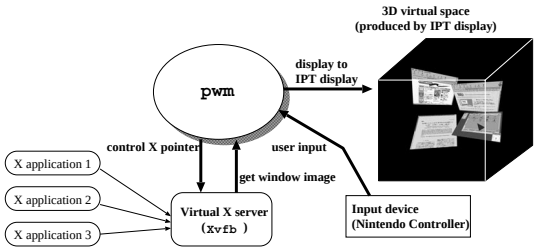

Fig. 7: Current `pwm` implementation: a proxy display system.

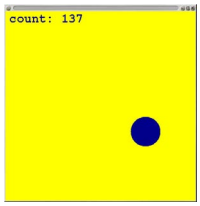

Fig. 8: A window image of a picking experiment application.

Firstly, implementation is easy. Secondly, many systems can display the same images that a display server provides at the same time. Thirdly, if the X protocol is revised, real 3D display servers will need to be revised, but proxy displays will not.

The `pwm` has two remarkable functions. One is its ability to retrieve the latest window images from a real display server. The `pwm` uses `Xvfb` as a real X display server. `Xvfb` is an X display server that is available in a sample implementation distribution package from The Open Group X Project Team (X Version 11, Release 6.4). `Xvfb` does not have any screens that users can see, but it has a virtual screen that it uses to display its contents. The `pwm` periodically retrieves windows images from `Xvfb` because it cannot sense changes of window contents.

The other function is to fake user inputs against the X display server. To achieve this function, the `pwm` uses an XTEST extension. XTEST is one of the extensions of an X display server that enables X clients to move a mouse pointer, to press mouse buttons, to release mouse buttons, to press keyboard keys and to release keyboard keys.

The `pwm` is based on the glCABIN library, which was developed at the University of Tokyo. It is based on the OpenGL API. COSMOS, where the `pwm` is implemented, contains the Polhemus ULTRATRAK as its 6 DoF position tracking system. COSMOS has a total volume of 3 cubic meters, in which it has six screens, and uses an SGI Onyx2 InfiniteReality as the main computer system. This has 16 CPUs (MIPS R10000 250 MHz), a 4 Gbyte main memory, 6 graphics pipes and an IRIX 6.5 OS. This computer system can handle a large 2D image whose size is 1024 × 1024 pixels in RGBA mode as a single texture image.

The `pwm` is working at over 35 Hz rendering refresh rate at COSMOS when it displays 10 plates, each of which displays a window that consists of 800 × 600 pixels. Each plate is displayed at a size of 0.8 × 0.6 meters. In this `pwm` implementation, the rendering refresh rate has little relation to the number of plates, because there is only ever one special plate that has a live window image. The content of the special plate is refreshed by retrieving the image from its corresponding window twice every second. In addition, when a user presses a virtual mouse button, and after waiting 300 milliseconds, the `pwm` retrieves and refreshes the window image. In this implementation of the `pwm`, this special window is the "default root window."

In this configuration a user can read any character by looking at an individual plate more closely. The `pwm` is described by only about 3500 lines (without the glCABIN library) in C++ programming language. This `pwm` implementation supports the pressing and releasing of three kinds of mouse buttons, and a user can operate any X application just as he/she could do with a mouse. The `pwm` is also functional at other immersive multiscreen displays, namely CABIN [4] at the University of Tokyo and at CAVE [3] in Yokohama, which was built temporarily for iGrid 2000 at INET 2000. During this demonstration, the `pwm` was working constantly for more than 2 hours.

6. A Picking Experiment

With the `pwm`, a user can manipulate 2D applications in a 3D display, but the user may feel some difficulty in manipulation when compared with a usual 2D environment consisting of a display and a mouse. To compare a user's picking performance at the `pwm` in COSMOS with that in a 2D environment, we use a modified Fitts' law model.

According to [13], when the index of task difficulty of the task is

$$ID = \log_2(A/W + 1)$$

where A is the distance from the starting point to the end point, and W is the width of the target, the movement time (MT) to select the target is

$$MT = a + b\ ID$$

where a and b are constants. We use an experimental application (Figure. 8) which proceeds as follows:

(1) Choose a point randomly.
(2) Draw a circle there.
(3) When clicked by the user, go to (1).

The same application was used for the `pwm` (Figure. 9) and in a 2D environment (Figure. 10). The five subjects were all familiar with using a mouse. The values for the radii of the circles, R, were 10, 25, or 40 pixels ($2R = W$). The distance, A, were between 20 and 627 pixels, so the values of ID were between 1.0 and 5.01. Each subject

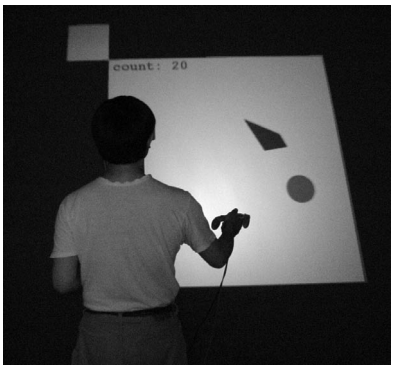

Fig. 9: A picking experiment at COSMOS.

Fig. 10: A picking experiment at a normal 2D environment.

clicked more than 299 times in each environment. The duration of the movement time, MT, is measured from the time when a user starts to move a pointer until the time when he/she presses a button.

A scatter plot graph of the MT – ID relationship for each subject in each environment is fitted by a linear regression line (Table 1). For each of the subjects, MT in the 2D environment is less than in the `pwm` .

We gathered the data for all of the subjects into each scatter plot graph at each environment, and fitted them to each linear regression line:

$$\begin{aligned} \mathrm{MT}_{\texttt{pwm}} &= 6.9 \times 10^2 + 2.1 \times 10^2 \times \mathrm{ID} \\ \mathrm{MT}_{2D} &= 4.3 \times 10^2 + 1.5 \times 10^2 \times \mathrm{ID} \end{aligned}$$

From these equations, the values of MT were calculated when $ID = 1$ or $ID = 5$ (Table 2). These results show that a user in teh `pwm` completes a pointing task within 155 % time of the time that it takes a user in a 2D environment.

7. Related Work

Fisher *et al.* [14] described the idea of a display environment in which the data manipulation and system monitoring tasks are organized in virtual display space around the operator. Seven years after this, Dykstra [1] and Feiner *et al.* [6] implemented the X Window System

Table 1: Measured motion time comparing the `pwm` with a 2D environment which consists of a 2D display and a mouse.

Subjects	2D ($\times 10^{-1}$ sec)	`pwm` ($\times 10^{-1}$ sec)
A	$4.7 + 1.0 \times \mathrm{ID}$	$4.9 + 2.3 \times \mathrm{ID}$
B	$4.7 + 1.2 \times \mathrm{ID}$	$8.1 + 1.9 \times \mathrm{ID}$
C	$4.5 + 1.8 \times \mathrm{ID}$	$9.9 + 1.2 \times \mathrm{ID}$
D	$4.7 + 1.6 \times \mathrm{ID}$	$6.5 + 2.2 \times \mathrm{ID}$
E	$3.3 + 2.0 \times \mathrm{ID}$	$9.4 + 1.9 \times \mathrm{ID}$

Table 2: Comparing the `pwm` with a 2D environment at ID = 1 or ID = 5.

	ID = 1	ID = 5
$\mathrm{MT}_{\texttt{pwm}}$ (ms)	910.4	1750.0
MT_{2D} (ms)	589.4	1189.0
$\mathrm{MT}_{\texttt{pwm}} - \mathrm{MT}_{2D}$ (ms)	321.4	560.0
$\frac{\mathrm{MT}_{\texttt{pwm}}}{\mathrm{MT}_{2D}} \times 100$ (%)	154.6	147.2

display servers, which displayed windows on a plate floating in a virtual 3D space. Dykstra's implementation is based on a 2D display, and does not work with a stereoscopic display. The significance of this technology is definitely different whenever a stereoscopic view is available. We initially implemented a 2D display version of the `pwm` rather like Dykstra's system, but when we used it with a mouse, we found that manipulation was too difficult, and we decided that it would not become a viable replacement of normal 2D display servers. Both a stereoscopic display and a 6 DoF pointing device are essential requirements. The implementation provided by Feiner *et al.* is based on a "desktop" augmented reality system. They use a traditional keyboard and mouse as the input devices and they do not use 6 DoF "pointing" devices to move the plates or to move the mouse pointer.

The Task Gallery [2] also basically assumes a 2D environment such as [1], but proposes some sophisticated interaction techniques.

8. Discussions

In the `pwm`, a user must move plates around manually. Fisher *et al.* described an idea where plates are placed automatically around a user, and the user can manipulate them by speech and by gesturing interaction. Feiner *et al.* implemented a system where a plate is mapped onto a portion of a sphere, which is positioned about the user's head. In usual 2D GUI environments, windows are initially placed automatically, but we sometimes prefer to control the position of the windows manually. Therefore, both approaches must be integrated.

There is some research, such as that by Heath[10] who made a 3D widgets toolkit that gives a 3D look to 2D GUI

widgets. Such an approach is good for users, because they will appear as seamless in 3D space. However since the widgets' working principals are basically 2D, their depth information has less meaning. From the user's viewpoint, and according to the availability of the application's functions, these 3D widgets appear to offer little difference from 2D widgets.

In a a genuine 2D GUI manipulation environment, the combination of a 3D display and a 6 DoF pointing device would not always be better than a combination of 2D displays, a keyboard, and a mouse. From the standpoint of 3D application development researchers, the importance of this technology lags behind the development of 3D virtual space technologies. However, as we described earlier, `pwm` techniques will be used with "real" 3D applications in a single 3D Display at the same time. The 2D GUI manipulation in 3D-space technology is necessary if we are to make 3D displays our main display technology.

9. Future Work

We must overcome the serious limitation that a user can only manipulate one plate for 2D GUI operations. Ideally, all the `pwm` plates must be "clickable", and when a window is created, a plate must rise automatically at the appropriate position and orientation. In the prototype system described in this paper, all but one of the plates are a snapshot image of the special plate made when the user give the order to create it. A user can create a snapshot of the special plate whenever he wants, just by pressing a button. When this "multi-clickable-plates" version of the `pwm` is implemented, we will be able to experiment with the effectiveness of current interaction techniques, and to investigate more sophisticated interaction techniques. We must also investigate other potential interaction techniques, for example, re-sizing the window, selecting a plate using a virtual beam pointer and changing the user's position in virtual space as if he/she is on a flying magic carpet. Keyboard input must be supported. If a user operates the `pwm`, it is likely that he/she will also soon want to input one or more data strings.

10. Conclusions

We propose a method of using a 2D GUI window system of the type that is currently widely used, in the virtual 3D-space environment produced by an immersive multi-screen display. In the `pwm`, a user can use existing X applications without any modifications, can "click" windows on a special plate, create plates as the snapshot image of a special plate and move any plate with a "6 DoF drag" technique. We showed the ability of currently avaliable hardware to implement the `pwm`, and evaluated its performance. We have compared five different users with respect to their 2D GUI pointing performance at the `pwm` with a normal 2D GUI manipulation environment, which consists of a 2D display and a mouse. With our current `pwm` implementation, it takes 50 % more time for a user to complete a pointing task than in a 2D environment.

Acknowledgements

The authors would like to thank the Gifu prefecture for offering us the use of COSMOS, and Toshio Yamada for his efforts to maintain CABIN and COSMOS.

References

1. Dykstra, P.: " X11 in Virtual Environments: Combining Computer Interaction Methodologies," *proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, (1993).
2. Robertson, G., Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., Gorokhovsky, V.: "The Task Gallery: A 3D Window Manager," *proceedings of CHI 2000*, (2000).
3. Cruz-Neira, C., Sandin, D. J. & DeFanti, T. A.: "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," *in COMPUTER GRAPHICS Proceedings, Annual Conference*, pp.135 – 142 (1993).
4. Hirose, M., Ogi, T., Ishiwata, S. & Yamada, T.: "Development and Evaluation of Immersive Multi-screen Display CABIN," *Systems and Computers in Japan, Scripta Technica* **30**(1), pp. 13 – 22 (1999).
5. Yamada, T., Hirose, M. & Iida, Y.: "Development of Complete Immersive Display: COSMOS," *in Proceedings of VSMM 98*, pp.522 – 527 (1998).
6. Feiner, S., MacIntyre, B., Haupt, M. & Solomon, E.,: "Windows on the World: 2D Windows for 3D Augmented Reality," *in UIST '93* (1993).
7. Scheifler, R. W. & Gettys, J.: "The X Window System," *ACM Transactions on Graphics* **5**(2), 79 – 109 (1986).
8. Shoemake, K.: "Animating Rotation with Quaternion Curves," *in SIGGRAPH '85*, pp. 245 – 254 (1985).
9. Hearn, D. & Baker, M. P.: "*Computer Graphics — C version*," Prentice Hall (1997).
10. Heath, D.: "Virtual User Interface (VUI) – A windowing System for VR," *in Second International Immersive Projection Technology Workshop* (1998).
11. Rohlf, J. & Helman, J.,: "IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics," *in SIGGRAPH '94*, pp.381 – 394 (1994).
12. Strauss, P. S.: "IRIS Inventor, A 3D Graphics Toolkit," *in OOPSLA '93*, pp.192 – 200 (1993).
13. MacKenzie, I. S.: "Movement Time Prediction In Human-Computer Interfaces," *Graphics Interface '92*, pp. 140 – 150, (1992).
14. Fisher, S. S., McGreevy, M., Humphries, J. & Robinett, W.: "Virtual Environment Display System," *in Interactive 3D Graphics* (1986).