Using Perspective in 3D File Management: Rotating Windows and Billboarded Icons

John R. Maltby
School of Commerce and Management, Southern Cross University
john.maltby@scu.edu.au

Abstract

The evaluation and comparison of 2D and 3D workspace environments is a hot topic. Much has been said concerning the promise of 3D workspaces but much less has been realised. Part of the problem appears to be the difficulty of developing 3D GUIs with high usabilities and effective navigation mechanisms. Many early environments simulated a 3D office but all suffered from a range of problems, from poor navigation to issues of spatial cognition. Indeed, most researchers now consider an office to be an inappropriate metaphor for an effective 3D computer environment. This paper describes the development of a 3D workspace based directly on the WIMP metaphor as opposed to a desktop or office metaphor. The environment uses rotating transparent 2D windows in a 3D world and presents icons as billboards.

1. Introduction

The WIMP/desktop metaphor employed by operating systems such as Microsoft Windows (MS) and others is considered by many to be both outdated and inefficient (eg. [1][2]). Such operating systems are sometimes known as 2½D GUIs [3], as they allow the overlapping of windows and other objects (effectively in a third dimension of infinitesimal size). Here, we shall refer to them as simply 2D interfaces.

Over the last few years, GUI front ends to convert Microsoft Windows and other operating systems into 3D environments have become available. Two well known ones are Win3D [4] and 3D Top [5]. More sophisticated 3D workspace managers have also been built, such as the Task Gallery [6], 3Dwm [7], Project Looking Glass [8][9], SphereXP [10] and Metisse [11]. The latter is a complete X Window management system, rather than a custom 3D desktop. Unfortunately, the Task Gallery, developed by Microsoft, has not been made generally available and its future is unclear. However, SphereXP is in a public beta and 3Dwm, Project Looking Glass and Metisse are available as open source.

Although many users find 3D environments fun to use, there is little evidence that the use of 3D images make an interface more usable, or that they can improve efficiency or productivity. A key premise of many of the current designs is that 3D virtual environments can more effectively engage spatial cognition and perception than can 2D environments. However, to date, research studies have provided conflicting conclusions: certain studies have indicated that spatial memory can help locate items in a 3D environment (e.g. [6][12][13][14]); others have indicated the reverse (e.g. [15][16]). The result is that all of currently available 3D GUIs have been less than successful, with many suffering from major navigation problems [17]. Part of the problem in obtaining evidence that the use of 3D images can improve efficiency or productivity is the large number of differences that arise between a typical 3D environment and a normal 2D desktop. This makes it difficult to identify what is better about 3D and what is not.

2. The Problem

In a computer environment, users need to:

- find and execute multiple applications;
- import and export data between applications;
- move and copy data from one location to another.

The 2D GUI in the form of the desktop metaphor offers the majority of computer users an evolution in efficiency over and above the CLI (command line interface). Nevertheless, the 2D GUI has been criticised as seriously flawed [1][18]. In particular, problems arise from a continued need by users to reposition windows in order to perform simple tasks such as file copying, cutting and pasting objects, etc. Trying to position two or more windows simultaneously on the screen in order to provide the correct juxtaposition for the job in hand can be both fiddly and time consuming. This manipulation and repositioning of windows to enable tasks to be performed is well documented and is known as *window thrashing* [19]. The problem is caused by the constraints imposed by limited screen real estate. Many solutions have been suggested, including:

- large displays;
- multimon (dual or multiple head displays);
- virtual reality;
- virtual 2D desktops;

- use of a virtual third dimension.

All of these provide some way of increasing screen real estate, either directly (as with larger or dual monitors) or indirectly (as with virtual 2D and 3D environments). It is the use of a virtual third dimension that interests us here. In particular, we are keen to develop an application with a sensible design that can be used to identify usability problems in a 3D workspace manager that can be easily compared and contrasted with its everyday 2D equivalent.

3. 3D Office Simulations and Desktops

A common misconception is that simply adding a third virtual dimension to an existing metaphor will provide a functional and efficient 3D environment. A typical example of this approach is to extend the 2D desktop metaphor to an office metaphor in which the user is allowed to move around in a virtual 3D office (or even a 3D office complex, with multiple rooms or volumes). An environment of this nature is shown in Figure 1.

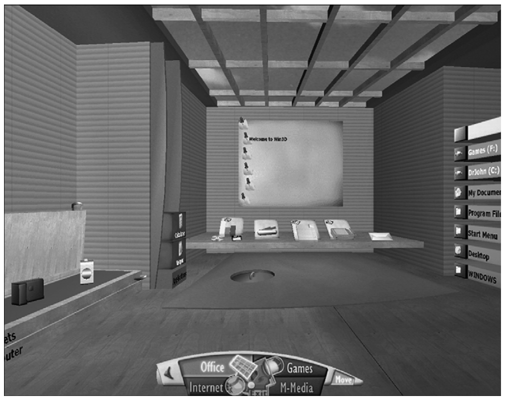


Figure 1. Win3D

Such office-type environments are in general very inefficient because they have navigation mechanisms more appropriate to games than to workspace managers. Indeed, 3D games and workspace managers are very different animals. In a 3D game of the first-person variety (eg Doom/Quake), the user has to move from room to room or area to area (or more correctly, volume to volume), seeking some goal, blasting monsters and dodging missiles along the way. Often, the player gets lost, cannot remember where that last cache of ammunition or health pack was, and has to navigate a time-consuming and hazardous connection of rooms and tunnels to get from point A to point B. Although this challenge can be exciting and rewarding in its own right, the navigation methods employed are not designed to let the player find something quickly. The aim of the simulation is to make the user feel like he or she is in a real environment, walking, running and firing missiles like a real person. But the environment itself (like a real-world multi-volume environment) is not conducive to efficiently locating objects and transferring objects from one point in space to another (as would be required, for example, when cutting and pasting objects such as files). In a game, it's *supposed* to be difficult to get from point A to point B. The bottom line is that many people cannot find things in a real office, let alone a virtual one. And transferring funds using the internet is much more efficient than driving around in a car visiting banks!

The needs of a 3D workspace manager are in total contrast to those of a first-person 3D game. In a workspace environment, the user needs to locate objects quickly, bring objects into close proximity in the workspace (so they can be "worked upon") and allow the efficient transfer of objects from different points in space in near or adjacent volumes. This can only be achieved by providing a mechanism for the user to identify the relevant volumes easily and move them into close proximity in 3D space in the minimum time. Even if we were to consider keeping the office metaphor, we would surely insist that any walls (and floors?) provided no obstacles to movement: why walk round walls when, in a virtual world, you can walk through them? Indeed, given the infinite number of possibilities presented by a programmable virtual environment, we would be guilty of limited vision if we insisted that our 3D workspace manager emulates a real physical space, with all the limitations that this implies. As stated in [20] "An intriguing possibility is that enhanced 3D interfaces might offer simpler navigation, more compelling functionality, safer movements, and less occlusion, than 3D reality, especially for information exploration and visualization tasks" (p.12).

The workspace managers shown in Figure 2 and 3 are an improvement on the office/rooms/game scenario. In 3D Top (Figure 2), the individual rooms or volumes are linked through hyperspace and the user can navigate to any other volume by clicking on the link. Icons within a specific volume float in 3D space can be selected and activated.

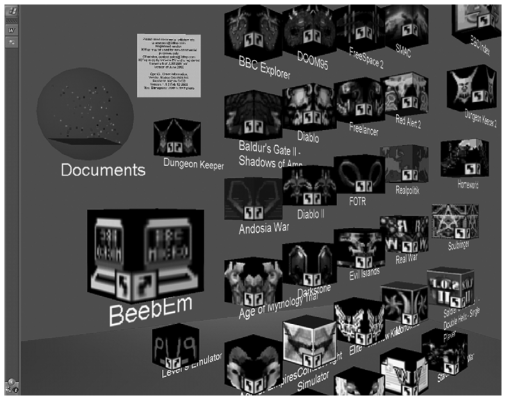


Figure 2. 3D Top

In SphereXP (Figure 3) icons are stored on the inside of a sphere. However, in neither of these managers can different volumes of icons be viewed simultaneously: in general, therefore, it is not possible to copy or move an icon easily from one volume to another.

A user is still left with the feeling that this is something of a gimmick: attractive to look at, perhaps, and fun to play with, but not conducive to efficient task completion in the workplace.

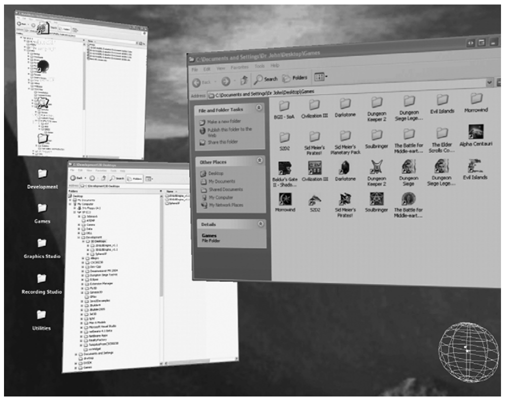


Figure 3. Sphere XP

Figure 4 shows Project Looking Glass, also known as lg3d. This design utilises a different approach. In lg3d, 3D objects, windows and applications are placed within a 2D desktop framework with a 3D look-and-feel. As well as the ability to handle custom 3D applications, on a Unix / Linux platform lg3d will run existing X11 applications in windows that can be rotated and manipulated in 3D space. This includes the ability to copy icons from one window to another whilst windows are rotated in 3D space. It is even possible to enter text into an application whilst the window containing the application is oriented in a plane other than that of the desktop. However, lg3d only allows the manipulation of 3D objects against an essentially 2D backdrop; it does not currently allow movement of the user viewpoint in 3D. Looking Glass is still in an early development stage, but it seems to offer considerable promise.

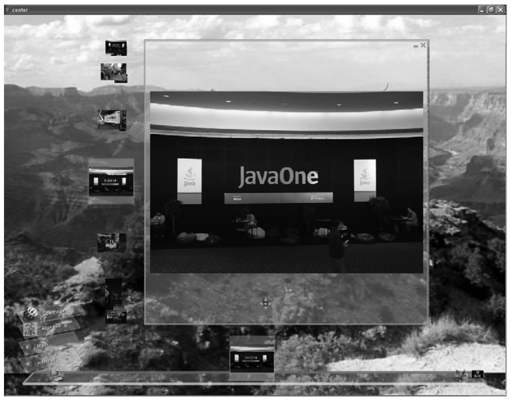


Figure 4. Project Looking Glass

4. What Does 3D Have To Offer?

In terms of common computer tasks associated with everyday work practices in a 2D environment, the answer to this question is currently uncertain. In [21], advantages of 3D User Interfaces are highlighted in the statement "… spatial representation and navigation of information can have significant advantages over other forms, given innate human skills for navigating and handling objects in the real world. People act naturally in spatial worlds, finding their way around by using sophisticated, but largely unconscious, sensori-motor skills. Spatial skills are thus deeply encoded in the human mind and are predominantly perceptual rather than cognitive in nature. But space also has meaning for people, and they find spatial locations intrinsically memorable" (p.2). This sounds good, but unfortunately proof that these factors make 3D workspace managers more usable and efficient than 2D ones remains elusive. Whilst such advantages may well be realised by the use of virtual reality environments, they are not always afforded in the same way when representing a 3D image on a conventional 2D computer monitor. One has to remember that any 3D image that we think we see on such a computer screen is actually virtual. In particular, a static 3D virtual image is just a 2D image and as such it offers no inherent advantage. For example, an object that is moved away from the viewer in virtual 3D space appears smaller but, if the motion is not observed, the same end result can be achieved by scaling an object in 2D space. In the same way, an object that has been rotated with respect to the viewer takes on the same appearance as an object that has been distorted in 2D space. For static images seen from a static viewpoint, therefore, it becomes difficult to argue that 3D is necessarily any better than 2D. But things change when an object or the viewpoint that we see the object from move, because we become aware of perspective. From this basis, we can argue that perspective is the most important single factor that separates a virtual 3D representation in 2D from simple 2D. It is perspective that provides the illusion of 3D in a 2D display and it is perspective that allows a user to see previously hidden objects by changing the viewpoint in order to "look behind" obstructing objects.

If 3D is to offer any advantage, we need to utilise perspective in a virtual 3D workspace so that it improves the transparency and effectiveness of the interface. This means that we must provide "natural" 3D motion control in a 3D virtual environment where perspective improves task efficiency over a 2D WIMP environment. We also need to show that improvements in usability afforded by 3D are greater than those that can be achieved by modifying 2D interfaces. It follows that 3D can only be an improvement over 2D if perspective can be used to improve the efficiency of task management in a way that is not possible using 2D techniques such as object scaling, distortion and transparency.

This is tough call - and one that may not even be possible. But we can start by looking at how we might extend existing 2D interfaces to take advantage of perspective without detracting from whatever 2D mechanisms may exist. Note that exploiting perspective does not mean making everything 3D; rather, we need to create an environment that uses perspective to increase the ease of location and the efficiency of manipulation of objects within it without burdening the user with the intricacies of navigation.

5. The Design

As a means of evaluating the effect of an additional degree of freedom on the traditional WIMP interface, we have designed and developed a test environment to allow the manipulation of 2D windows in 3D space and the cutting and pasting of icons between these windows. The more important issues considered were:

- Can we make a 3D workspace environment that is essentially similar to a 2D environment, but with any advantage that might be afforded by an extra spatial dimension?
- How can navigation in 3D environments (often a problem) be made efficient and transparent?
- How many degrees of freedom should be provided?
- How can perspective be used to maximum advantage without causing confusion?
- Two common objects in a 2D GUI which are arguably the most important are windows and icons. How should these be represented in 3D?

Consideration of these issues led to the following design requirements:

1. The environment must allow perspective to change the user's viewpoint.
2. Windows need only be 2D representations that can be rotated in 3D space.
3. Icons in windows must be *readily identifiable*. There is little point in making icons 3D or rotating them with respect to the user. Therefore, billboarding[1] should be used to display such icons so that they always present the user with the maximum frontage possible (thus enabling easy identification and selection). This also helps to cement in the perspective interpretation of what is being viewed.
4. Windows must have transparent client areas in order to support billboarding.
5. It must be possible to navigate the desktop in an efficient and intuitive manner.
6. It must be possible to manipulate individual windows (translate and rotate) and icons (select and drag) in an efficient and intuitive manner.
7. The need to scroll window file contents is reduced, at least in a test situation. As windows can be manipulated in three dimensions, it is possible to maintain a client area large enough to hold all icons for windows with less than about 30 icons.

[1] Billboarding was used in early 3D game development to display 2D bitmaps as apparent 3D objects (thus saving processing time in comparison to displaying real 3D objects). The technique rotates an image with respect to its environment to ensure that it always faces the camera. Provided the image represents an object which is understood to be symmetrical about its vertical axis (such as a tree), the illusion can be created that the image represents a solid object.

6. Navigation Issues

In order to provide maximum flexibility, it was decided to allow the user to manipulate in 3D both objects within a set viewport and the viewport (ie camera) itself, independent of any objects. At the same time, the number of degrees of freedom (DOF) available was limited to those necessary, in order to reduce complexity and potential confusion for the user. Thus the only entities it was thought desirable to rotate were the windows themselves and the camera. Whilst each of these was allowed to have a full 3 translational DOF, it was considered necessary only to provide one rotational DOF about the vertical axis (y-axis). No other rotation was deemed desirable; in particular, the full rotational freedom of yaw, pitch and roll (as in a flight simulator) was deliberately avoided. This resulted in 4 modes of movement being considered, labelled as follows:

- Camera Translation Mode (CTM)
- Window Translation Mode (WTM)
- Camera Orientation Mode (COM or GM)
- Window Orientation Mode (WOM or TCM)

Each translational mode has 3 DOF associated with it (translate along x, y or z axes) and each orientation mode has 2 DOF associated with it (translation along the z axis and rotation about the y axis). TCM stands for Toy Car Mode, as using this mode is just like driving a radio controlled car that can steer but cannot reverse. In this case, the window being moved behaves like the car and the user can "drive" it round the 3D workspace. GM stands for Game Mode, as using this mode is like playing a game of Doom or Quake: the user can manipulate their viewpoint by walking forward or turning to the left or right. The mode (window or camera) is selected by selecting a window for the window mode or the camera (by clicking on the background) for the camera mode.

To implement the above, some consideration was given to using a specialised 3D input device, such as a spaceball. However, many games have shown that it is possible to map traditional mouse input to 3D motion with a high degree of transparency (despite claims that there is "a poor match between the goal of such a navigation activity, the control device, and the skills of the average user" [22] (p.1)). Because of this and the high degree of typical user familiarity with a mouse, a decision to use this device was taken here.

A traditional mouse can be considered to have 2 translational degrees of freedom (DOF). These are achieved by dragging left-right and away-toward (normally translated into up-down in a 2D environment). If the mouse has a scroll wheel, this can be used to provide a third translational DOF (often used to zoom in 3D games). The presence of a left and right buttons offers the opportunity to convert mouse drags into additional rotational degrees of freedom. After much consideration and experimentation, the mouse was mapped to 4 DOF, three translational and one rotational. The mapping is achieved as follows:

- With the left button down
  - mouse left-right drag converts to left-right translation (x-axis)
  - mouse away-toward drag converts to away-toward translation (z-axis)
- With the right button down
  - mouse left-right converts to rotation about the vertical (y-axis)
  - mouse away-toward converts to movement *along the local z-axis of the object being moved*
- The mouse scroll wheel converts to up-down translation (y-axis), irrespective of any button presses.

It must be said that this is much easier to demonstrate than to describe and that proficiency in use of this 3D navigation mechanism is not difficult to achieve.

7. Development of the Prototype

Development initially commenced in MS Windows. Rather than code a 3D engine from scratch, an existing 3D game engine [23] was modified and extended. This engine was written in C++ using OpenGL for the graphics and DirectX for the keyboard and mouse input routines. One of the problem issues in development was the need to select objects in virtual 3D space using the mouse. Selecting 3D objects from a 2D representation using a normal 2D mouse pointer is problematic as there is ambiguity of depth. The method used here identifies all hits on windows and icons along the z direction with the mouse pointer and stores them in order of depth in a buffer. It then assumes that the user wants to identify the hit on the nearest object. Once identified, the object can be moved by dragging. If the object is a window, then it can be moved in 3D. If it is an icon, movement is restricted to the 2D plane of the screen as this is sufficient to allow a drag and drop.

Figure 5 shows the effect of opening and positioning multiple windows. A grid is provided to help establish the correct perspective. This can be toggled on and off. In the prototype, five folders are provided which can be clicked on and opened into five windows. As can be seen, the windows are transparent, with a title bar (which can be activated) and a close box. The transparency is to facilitate the icon billboarding. The size of a window is dictated by the number of icons that the window needs to display. This size is automatically adjusted if icons are removed or added. In the MS Windows prototype, there were no scroll bars, as windows of any size can be displayed by moving the window and camera with respect to each other (effectively zooming). A virtual 3D desktop can only work by providing the correct perspective cues of relative size and occlusion. The icons are therefore all the same size when viewed from the same distance: this provides a depth cue when windows are moved with respect to the camera.

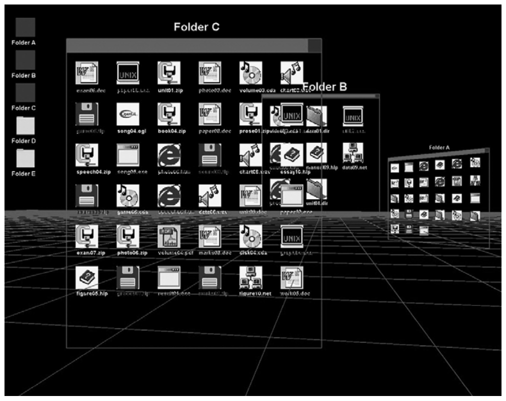


Figure 5. Three out of 5 folders open, each at a different position and orientation

Figure 6 shows multiple windows open, each moved a different distance and orientation from and to the camera. The effects of billboarding can be clearly seen. The advantage of this technique is that the icon images are never distorted (as they would be if the icons maintained their orientation with respect to the plane of their parent window). The disadvantage is that as the rotation of the window becomes more acute, the icons at the front start to obscure those at the back. However, this may well be better than distorting both the icons and the associated text labels (and, of course, a rotation angle can always be chosen that reduces window space but still allows all icons in the window to be individually selected). Although obscuration of icons may be an issue if the oriented window is required as a *source* of an icon drag, it is not a problem if the window is to be used as a *target*, as any angle of rotation allows the depositing a dragged icon. This is true even if the target window is edge on to the camera so that only the front row of billboarded icons is visible. Nevertheless, further investigation is needed here.

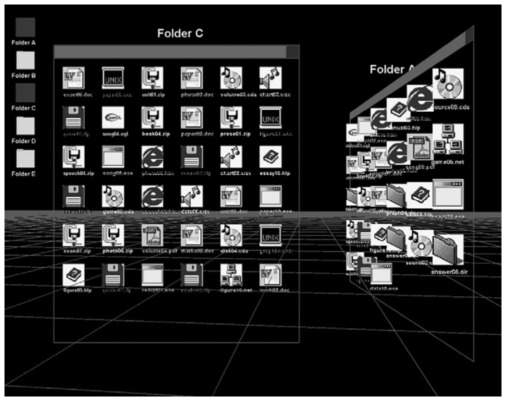


Figure 6. Viewpoint and windows manipulated to allow easy cut and paste of icons (note the billboarding)

From Figure 6, it can be seen that perspective provides a mechanism for reducing the screen real estate occupied by a window. In theory, therefore, 3D perspective should reduce window thrashing, either because windows are reduced in size by translation away from the user or rotated at an angle, or a combination of both. We need to know whether or not this is more efficient than resizing and scrolling a normal 2D window; this needs to be the subject of further research.

For empirical quantitative research to be undertaken, it is necessary to log user data automatically. The system currently monitors user actions and dumps screen images to file, either to order by key press or automatically on icon deposit and close down. The data recorded includes the task time, number of window and icon manipulations (opens, closes, scrolls and drags) and the number of 3D operations (CTM, WTM, COM and WOM moves).

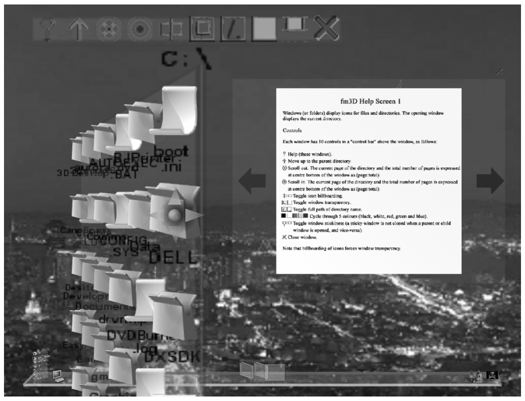
8. Evaluation of the MS Windows Prototype

An early evaluation of the prototype was undertaken by [24] as part of a student project. Using 20 university students as participants in a two-group experiment (10 per group), the user window manipulations necessary to copy and move files in the 3D environment were compared with the equivalent operations required to achieve the same outcomes in a "look-alike" 2D environment. It was concluded that there was no significant difference in usability between the two environments. From observation, users seemed to find the 3D navigation reasonably easy to learn and undertake. Given that most users who tested the environment where familiar with 2D navigation but not with the 3D equivalent, this might be considered a promising result. Again, however, further investigation is required.

9. Porting to Project Looking Glass

The MS windows prototype has been ported to the Project Looking Glass environment as an open source Java project under the GNU public licence. The original 3D game engine code had to be abandoned and replaced with the lg3d development environment, making it necessary to re-code the project effectively from scratch. The need for a project name arose, so it was christened fm3d (fm3d can be downloaded from the Sun Project Looking Glass site [25]). As the Project Looking Glass environment has a fixed camera space, it was necessary to dispense with both the CTM and COM navigation modes. It was also necessary, to fit in with the existing conventions used by lg3d, to reconfigure the WTM and WOM navigation modes so that mouse away-toward drag converts to up-down translation and the mouse scroll wheel converts to away-toward translation. As indicated earlier, this is in line with current convention in applications when translating the 2D motion of a mouse to 3D. However, in this author's view, the translation used in the MS Windows prototype described earlier provides a more direct (and therefore more efficient and pleasing) mapping[2].

Some modifications and additions were made to the Java version. Each window was provided with a billboarded control bar containing the controls necessary for window manipulation and appearance. These include moving between directories, scrolling icons in and out of a window and toggling on and off billboarding, transparency and window stickiness. Icons for these controls are seen at the top of the screen shot shown in Figure 7.



fm3D Help Screen 1

Windows (or folders) display icons for files and directories. The opening window displays the current directory.

Controls

Each window has 10 controls in a "control bar" above the window, as follows:

? Help (these windows).
↑ Move up to the parent directory.
⊗ Scroll out. The current page of the directory and the total number of pages is expressed at centre bottom of the window as (page:total).
⊙ Scroll in. The current page of the directory and the total number of pages is expressed at centre bottom of the window as (page:total).
Toggle icon billboarding.
Toggle window transparency.
Toggle full path of directory name.
Cycle through 5 colours (black, white, red, green and blue).
Toggle window stickiness (a sticky window is not closed when a parent or child window is opened, and vice-versa).
✕ Close window.

Note that billboarding of icons forces window transparency.

Figure 7. Left-hand side: fm3d window with control bar, transparent and orientated; right-hand side: help screen

The original MS Windows prototype did not access the operating system or read or write to the system hard drive, the icon distribution being provided by a data file. However, the fm3d version works like a true file manager and allows real file copies, moves and deletes. A left button drag from the source to the target window automatically copies, whilst a similar right button drag automatically moves. A right button drag from a source window to the desktop provides the option (via a small menu) of a delete.

To make this workable with large numbers of files, it was necessary to limit the number of icons visible in a window and to introduce some mechanism of scrolling through a window's contents. Traditional scrollbars are clearly 2D devices, required to overcome the limitations of limited 2D real estate. Something better is needed in 3D environments. The solution adopted is to distribute the icons in a window over as many "panes" as required, with each pane holding a maximum of 30 icons. Only one pane is visible at a time; this is in the client area of the window, with the number of the pane and the total number of available panes displayed in the window as a ratio (e.g. 3:11). When a scroll is requested, the icons in the next pane in the direction identified (in or out) initially appear in a parallel plane (as indicated in Figure 8) and are then moved automatically over a time increment to replace the displayed pane within the client

[2] Even if such mapping were proven to be a better, it is unlikely to become standard (cf. the QWERTY keyboard is still with us!).

area (easier to demonstrate, of course, than to describe). Scrolling in increases the page number and scrolling out decreases it. The result is that it is easy to locate a page within the total number of pages, with the scrolling providing an animated visual clue that reinforces the 3D representation.

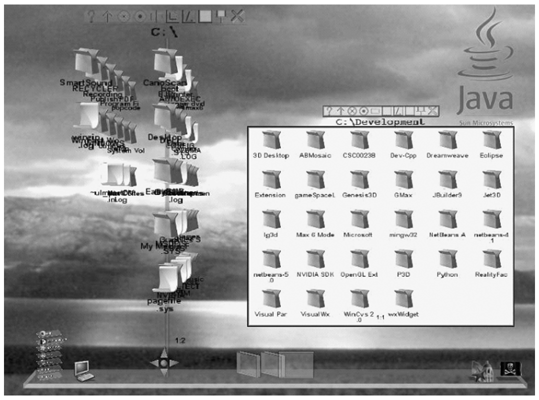


Figure 8. Left-hand side: start of a window "scroll in" – the next page of icons appears as a second layer before moving in real time to replace the icons in the client area of the window; right-hand side: opaque window in the plane of the desktop

10. Conclusions

Many students have tried the MS Windows version. From observation, these users seemed to find the 3D navigation reasonably easy to learn and undertake and none of the navigation modes described here appear to be difficult to become proficient in, at least not when users are given some practice time. The author and developer of the software (as might be expected) finds them almost second nature. The development of fm3d and its incorporation into Project Looking Glass indicate that there is plenty of potential for experiment with the 3D manipulation of objects in workspace managers and that the future of 3D may well be served by desktops of this nature (rather than of the "office" variety). Moving around in the real 3D world may be enjoyable, but it is not a particularly efficient modus operandi.

Two factors in the current design that require proper investigation are window transparency and billboarding. Although billboarding of icons seems attractive to users, it is not obvious that it is efficient or even a good way to do things. Whilst it allows icons to maximise their frontage, it also obscures icons. In a window oriented at 90° to the user, only the “front” column of icons is visible. Although this is not a problem when depositing icons after a copy or a move, it can restrict access to icons if the window is used as a source for copying or moving.

The mechanisms adopted here to reduce window thrashing still require the user to position and re-position windows, and the efficiency of this must reflect to some degree (no matter how usable the environment) upon the skill of the user. A way to remove this limitation may be for workspace managers to provide an automatic positioning mode, where windows are oriented, scrolled and positioned by the system in order to optimise access (thus saving the user the need to do any thrashing). At least one system that uses a constraint-based layout has been developed in 2D [26]. The equivalent in a 3D environment would need to include the use of perspective to provide a visualisation that is both efficient and easy to work with.

It is hoped that this paper might provide a starting point for research into how *perspective* in a virtual 3D environment can aid efficiency of task completion. Only with success in this direction can the continued use of 3D be justified as a valid mechanism for workspace managers.

11. References

[1] Landay, J.A., J.I. Hong, S. Klemmer, J. Lin & M. Newman. Informal PUIs: No Recognition Required. In *Proceedings of AAAI, Spring Symposium (Sketch Understanding Workshop)*. Stanford, CA. 2002.

[2] van Dam, A. *Beyond WIMP*, IEEE Computer Graphics and Applications, 20 (1), 50-51. 2000.

[3] Leach G., Al-Quaimari, G., Grieve, M., Jinks, N. and McKay, C. Elements of a Three-Dimensional Graphical User Interface. In *Interact '97: 6th IFIP International Conference on Human-Computer Interaction*, Sydney, Australia, 69-76. July 1997.

[4] Available: http://www.clockwise3d.com/Home_shockwave.html

[5] Available: http://www.3dtop.com/

[6] Robertson, G., van Dantzich, M., Czerwinski, M., Hinkley, K., Theil, D., Robbins, D., Risden K. and Gorokhovsky, V. The Task Gallery: A 3D Window Manager. In *Proceedings of CHI ’2000, Human Factors in Computing Systems*, The Hague, ACM press, 494-501. April 1-6, 2000.

[7] Rosinger J., Leach, G. and Al-Qaimari. 3DWM: A 3D Window Manager Implementation. In *Proceedings of the Australian Conference on Computer-Human Interaction, OZCHI*, Charles Sturt University, Wagga, Wagga, NSW, Australia. 1999.

[8] Heiss, J. *Going 3D with Project Looking Glass, The Source for Developers*. Available: http://java.sun.com/developer/technicalArticles/J2SE/Desktop/lookingglass/. October, 2004.

[9] Available: https://lg3d-core.dev.java.net/

[10] Available: http://www.hamar.sk/sphere/

[11] Chapius O. and Roussel N. Metisse is not a 3D Desktop! In *Proceedings of UIST'05, the 18th ACM Symposium on User Interface Software and Technology*, ACM Press, 13-22, October 2005.

[12] Tavanti, M. and Lind, M. 2D vs 3D, Implications on Spatial Memory. In *Proceedings of the IEEE InfoVis 2001 Symposium on Information Visualization*, San Diego, 139-145. October 22-13, 2001.

[13] Ark, W, Dryer, D., Sleker, T. and Zhai, S. Representation Matters: The Effect of 3D Objects and a Spatial Metaphor in a Graphical User Interface. In H. Johnson, N. Lawrence, C. Roast (Eds), *People and Computers XIII, Proc of HCI’98*, Springer, 209-219. 1998.

[14] Robertson, G. , Czerwinski, M., Larson, K., Robbins, D., Thiel, D. and van Dantzich, M. Data Mountain: Using Spatial Memory for Document Management. In *Proceedings of UIST '98, 11th Annual Symposium on User Interface Software and Technology*, 153-162. 1998.

[15] Cockburn, A. and McKenzie, B. Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments, In *Proceedings of CHI 2002*, Minneapolis, Minnesota, USA. April 20-25, 2002.

[16] Sebrechts, M., Vasilakis, J., Miller, M., Cugini, J. and Laskowski, S.J. Visualization of Search Results: A Comparative Evaluation of Text, 2D and 3D Interfaces, In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkley, CA, 3-10. August 1999.

[17] Tan, D.S., Robertson, G.G. and Czerwinski, M. Exploring 3D Navigation: Combining Speed-Coupled Flying with Orbiting. In *Proceedings of CHI 2001, Human Factors in Computing Systems*, Seattle, WA, 418-424. April 2001.

[18] van Dam, A. *User Interfaces: Disappearing, Dissolving and Evolving*, Communications of the ACM, 44 (3), 50-52. 2001

[19] Henerson, D. *Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface*. ACM Transactions on Graphics (TOG), 5(3) 211-243. 1986.

[20] Shneiderman, B. *Why Not Make Interfaces Better than 3D Reality?* IEEE Computer Graphics and Applications, 23(6), 12-15. 2003.

[21] Waterworth, J.A., Personal Spaces: 3D Spatial Worlds for Information Exploration, Organisation and Communication. In R. Earnshaw and J. Vince, eds., *The Internet in 3D: Information, Images, and Interaction*. New York: Academic Press, 1997.

[22] Hanson, A. and Wernet, E. Constrained 3D Navigation with 2D controllers. In *Visualization '97*, IEEE Computer Society Press. 1997.

[23] Hawkins, K. and Astle, D. *OpenGL Game Programming*, Prima Tech, Roseville, 1-777. 2001

[24] Adjeiwaa, M. "Study of the Window Thrashing Problem in 2D and 3D Workspace Environments". Honours Thesis, School of Multimedia and Information Technology, Southern Cross University, Lismore, Australia. 2003.

[25] Available: https://lg3d-incubator.dev.java.net/

[26] Brados, G., Nichols, J. and Borning A. *SCWM – an intelligent constraint-enabled window manager*. In Proceedings of the AAAI Spring Symposium on Smart Graphics. IEEE Computer Society Press, March 2000.