3Desque: Interface Elements for a 3-D Graphical User Interface

Gavin Miller
Interval Research Corp.
DrGavin@aol.com

Sally Grisedale
Excite, Inc.
grisedale@earthlink.net

Kenneth T. Anderson
MediaOne Labs
kanderson@MediaOne.com

Abstract
This paper explores some design possibilities and constraints for 3D enhancements of graphical user interfaces. This is done with the aim of conveying more information in less screen space, while avoiding visual clutter. Design elements include the use of “slapped-back” windows obtained by projecting windows and icons into three dimensions; “trays”, an alternative to folders or piles; the use of transparent "beams" for indicating hierarchy; and the use of "periphs" for a form of fisheye projection on window borders. In addition we employ shading and shadows to enhance the interface, and automatic placement algorithms to prevent visually confusing occlusion. By considering these approaches together, rather than in isolation, the interactions between the different modifications are made explicit.

Keywords
Graphical user interface, 3-D, 2-D, affordance, projections, orthographic, perspective, piles, progressive disclosure, trays, periphs, beams, slapped-back windows, fisheye views.

1.0 Introduction

"Escaping this flatland is the essential task of envisioning information - for all the interesting worlds (physical, biological, imaginary, human) that we seek to understand are inevitably and happily multivariate in nature." Edward R. Tufte, Envisioning Information, 1990 [1].

Graphical user interfaces have had spectacular success over the last decade, becoming the de-facto standard for interaction with file structures and applications on personal computers. However, their intuitive ease-of-use and familiarity mask a number of underlying problems, [2]. These problems only become apparent as people try to use the interfaces to manipulate larger numbers of files or to orchestrate complex tasks. We list some of these problems below.

Visual Clutter:
When orchestrating complex tasks, such as copying between applications, users may end up with lots of overlapping windows which are only partially visible. This may lead to a cluttered visual field such as Figure 1.

The Visual Cliff:
When an object is off the edge of the visible portion of a window, there is very little indication that it is there. The window’s scroll-bar gives some indication of the size of the scrollable area, but not how many files occupy that space or their spatial arrangement. Different views, such as text lists, (Figure 2) may be more compact, but the visual cliff remains. The visual cliff also exists for the edge of the display. This may be particularly problematic for a scrollable desktop.

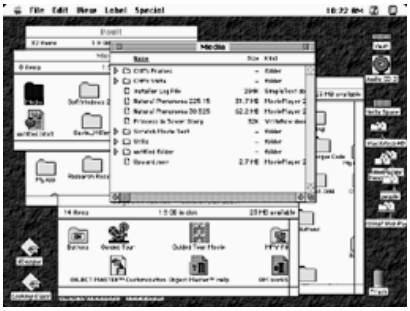

Fig. 1 Visual clutter in a window-based graphical user interface.

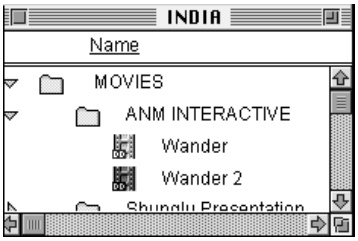

Fig. 2 Viewing a list of files.

Hierarchy as Memory
When opening a folder, the window grows out of the icon. However, once in position, the relationship is graphically lost and has to be replaced by an explicit query (such as command clicking on the title bar of the window) or remembered by the user.

Inefficient Layout
A window full of named icons does not use space efficiently. Each icon is surrounded by lots of empty

space. Icons cannot be packed more closely together because of the presence of file names (which tend to be wider than the icon). In addition, careless placement of icons or the use of file names which are too long can lead to icons that overlap and occlude each other.

Unexpected Icon Placement
Automatic placement of icons occurs as the result of a user specified event such as “Clean Up Window”. This can lead to unexpected large movements of the icons, often off the edge of the visible portion of the scrollable area.

Uninformative File Icons
File icons are typically small and uninformative, at best giving a small view of an image document, at worst merely indicating the type of document and its authoring application. This leads to an over-reliance on the names of files as an identification mechanism, which in turn encourages the use of long file names.

Uninformative Folder Icons
Folders give little indication as to their contents. A rare exception in the Macintosh user interface is the bulging of the trash can when it has contents, and shared folders [3]. Where modifications do occur, they usually indicate simple state changes such as empty or full, and fail to convey the actual contents of the folder. To browse the contents it is necessary to open the corresponding window. This uses a lot of space and contributes to visual clutter. We would rather have an interface mechanism which allows for progressive disclosure while being more compact than a window.

This paper tries to address some of these problems in the context of 3-D enhancements to a graphical user interface (dubbed 3Desque, pronounced "3 Dee Esk"). This allows for the introduction of new interface elements as well as an exploration of various projections to make representations more compact without losing context. By considering these approaches together, rather than in isolation, the interactions between the different modifications can be explored.

First, we discuss the genesis of the project and then introduce our development methodology and user study. Then we describe our approach to browsing individual documents at different levels of detail. Next, we introduce solutions to alleviate the visual cliff. Then we explore projecting windows and icons into three dimensions and discusses the trade-off between perspective and orthographic projections. Next, we introduce the idea of “trays” as an alternative to folders or piles [4], then we talk about the special need for automatic placement algorithms in 3-D projection and propose a solution. After that, we discuss the use of transparent beams for indicating hierarchy [5] [6]. We explore the use of shading and shadows to enhance the interface. Lastly, we end the paper with some conclusions.

2.0 Genesis of the Project
Every design project begins with an idea. In this case we became interested in exploring the use of 3D elements to extend the 2D graphical user interface of the Macintosh. The work began in 1994 as an exploratory project in Apple Computer, Inc.'s Advanced Technology Group and ran for nine months. The catalyst for this work was a discussion of the use of slapped-back windows to let a user keep the context of a window while using less space. As originally envisioned, the slapped-back window would have been planar and inactive, merely able to be flipped back up for subsequent interaction.

The use of animation, to flip the window back in space, was controlled by a button on the front edge. The movement was similar to the animated slabs used at the front of the PARC Information Visualizer [7] although at the time the project had not been a direct influence on our work.

A series of brainstorming sessions led to the graphic in Figure 3. 3D objects sit atop a planar slapped back window. The decision was made to keep text screen aligned for legibility at standard pixel resolutions.

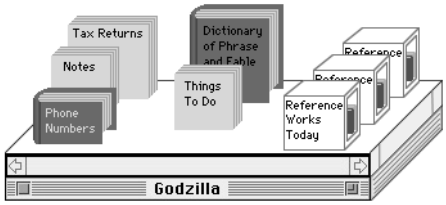

Fig. 3 Slapped-back window concept with 3D file icons.

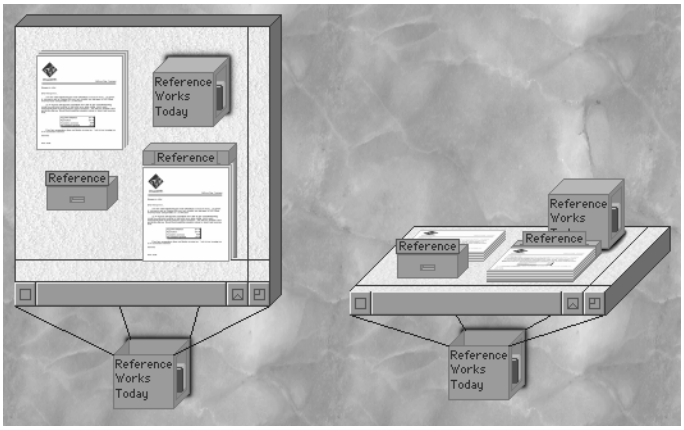

Fig. 4 Second design for 3D objects.

It was apparent that individual 3D elements would obscure each other and seemed to offer no particular advantages. Figure 4 shows a second design which attempted to replace the books with stacks of papers sitting on plinths, a

concept used in Piles [4]. Again object occlusion was a problem and the containers for objects seemed to offer no great advantages over current folders.

We did careful analysis of existing 2D interfaces and related work in the field of document management systems. At the time of developing the project we looked at work already produced at Apple Computer, Inc. This included: Piles, which allowed users to stack documents [4], HotSauce, a zoomable interface for browsing meta content tags [8], and Loretta Staples' work using perspective and lighting effects to enrich the visual vocabulary of the graphical user interface [9]. We also looked at related work in the field. Treemap used an automatically generated layout for overviews of data groups [10], "The Information Vizualizer" project at Xerox PARC showed a variety of 3D visualizations and interaction methods [5] [6] [7] [11] [12]. "Workscape" treated file representation like scenes from a movie "coming into focus along parallel dimensions" [13]. "Pad" used a zoomable alternative to the computer interface [14]. Fisheye views were used for viewing graphs by Sarkar and Brown [15] and more general views were employed by George Furnas [16].

From our analysis we came up with criteria outlined in the introduction. We used these as our guide in the subsequent development phase of the project.

2.1 Development Methodology and User Study

As they evolved, the graphical ideas for the user interface were first implemented in Macromedia Director[a] [17], to create the impression of the interface in action. This worked well for certain aspects of the design, such as developing the graphical look of the interface, simple animations and live dragging of individual elements. However, to explore more elaborate behaviors, such as the use of periphs (see Section 5.0), automatic layout and live shading, we had to implement the design in C++.

A usability study was designed to investigate participants' ideas and reactions concerning the use and functionality of 3Desque. The people who participated were male and female employees of Apple Computer, Inc. Their level of experience in using the Macintosh ranged from being casual users to expert developers. There were ten participants in the study. Each study lasted one hour and consisted of three parts.

2.2 Usability Study

In the first part of the usability study participants were shown a Macromedia Director[a] demonstration of 3Desque (based on the final design described later in this paper.) The demonstration was designed to familiarize participants with 3Desque. We encouraged them to comment on their first impressions e.g. what they understood, liked, or had problems with. While this led to anecdotal reactions rather than quantitative data, the user responses were helpful in gauging how familiar the users found the elements of the interface and whether they were confused or intrigued by its appearance.

2.3 Drag & Drop Behavior of 3Desque

Participants performed task scenarios designed to assess their understanding of 3Desque. The following actions were covered: selecting, dragging, dropping and finding files; selecting, dragging, dropping, opening, closing, re-sizing and browsing trays and windows in both 2D and 3D views; and scrolling slapped-back windows. We documented users' interactions with and comments about the interface. The scenarios assessed the participants' understanding of the dynamic nature of 3Desque.

2.4 Direct User Feedback on 3Desque Features

Opinions were elicited from participants on the individual features as well as the overall concept of 3Desque. The primary issues investigated were:

- User understanding of graphical proxies at a variety of scales and projections
- Drag, drop, open and close behavior of trays
- Functionality of slapped-back windows
- Efficacy of lighting effects

At the end of each session we asked users to evaluate each of the features of 3Desque as well as provide overall feedback using Nielson's heuristics [18]. All the interviews were video taped, and the results from the user study were then incorporated into the next version of the prototype.

3.0 Icons in Different Projections

In response to the problem of uninformative file icons, we placed a pictorial representation of the front page of the document onto the surface of the icon for that document. This is like an existing feature of the Macintosh User Interface, most often used by graphical applications like Adobe Photoshop[a], which use preview icons to represent images. We extended this approach to all document types including text files and sounds.

An icon could be viewed in three different projections (Figure 5), depending on the context. For example in a 2D window the icon appeared as a flat rectangle. In the 3D window, the icon was slapped back, (i.e. rotated about its bottom edge) in an orthographic projection. This made the icon a quadrilateral. Finally, when the icon was placed within a tray, (explained in the Section 7.0), the icon was rotated around its left-hand edge resulting in a "tilted icon". The icons were created using a two-pass resampling filter, equivalent to having a quadrilateral footprint for each final pixel. Vertical dimensions were compressed by a factor of three to one.

3.1 Responses to Different Icon Projections

In the study, we asked participants to select, drag, drop and find documents, some of which were their own, some were not, using the icons in a 2D view and a 3D slapped-back view. In the 2D view, participants didn't seem to have too much difficulty identifying their own files, however, they had difficulty making sense of files which were not of their own making. Strongly recognizable icons, like those for picture files could be identified, at least for the type of media. In future iterations, we may consider adding a graphic to the icon to represent the file format.

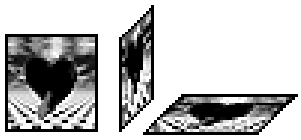

Fig. 5 An icon, a tilted icon and a slapped-back icon.

When the icons were slapped back, some of the participants had difficulty identifying the files, even their own. The complaints ranged from "not getting enough information", to problems "seeing it" because the shading was too dark, it was too small (because of the foreshortening effect of slapping back) or just not looking right.

Though there were several complaints about the 3-D slapped-back icons, we have evidence that people could recognize what objects were. Sue, for example, came across a bug where the wrong picture had been put onto the Icon. Other volunteers were able to select an icon after being given a description like, "move the spider icon into the media box". This is an indication that participants were able to understand more of the 3D icon than they might have realized. However, we should also consider removing or improving the slapped-back icon in a future version of the interface.

Fig. 6 A tilted icon with its proxy view.

4.0 Proxy Views of Documents

Any of the three icon projections can be used to reveal additional information during the browsing process. This involves moving the cursor over the icon while holding down a modifier key, in our case the option key. Doing so reveals a proxy view of the document (Figure 6).

A proxy view is much larger than the icon view and supports additional affordances for browsing the document summary information. This can be seen as an extension of the proxy views in Piles [4], to all icons in the interface.

4.1 User Study Results on Proxy Views

Users felt the proxy view provided quick and ample information on a document. However, holding down the option key was considered too burdensome by some users. Besides not fitting with the current Macintosh interface where modifier keys are reserved for shortcuts and non-default behaviors, users did not like to have to use two hands for a routine action. Frequently users are engaged in a number of office activities like answering the phone or drinking coffee while using their computers. Having to use two hands for a routine function cuts down on their ability to multitask. If possible, an alternative way to control this feature needs to be considered.

5.0 Avoiding the Visual Cliff

Currently, when an object scrolls off the edge of the visible portion of a window, it disappears. Previous approaches to having scrollable areas, that do not have this discontinuous property, include perspective walls [12] and fisheye views [15], [16]. Perspective walls maintain context in one direction by having additional angled faces that are to the left and right of the front scrolled area. The oblique orientation means that the sections of wall take up less screen space than if they were face on, and perspective foreshortening applies a non-linear re-scaling to the information on the wall.

Fisheye views have the advantage that they maintain context in a 2-D plane, enabling scrolling within the plane in any direction. They have the disadvantage that objects change size continuously as the view is scrolled. This is undesirable in a hierarchical interface, since some of the objects in the window may themselves contain affordances (such as the trays described later in this paper). Fisheye projections have a subtler problem, which is that their legibility changes based on the appearance of the individual icons. Icons which are low in contrast relative to the background, disappear as they become just a few pixels across.

The solution proposed for 3Desque was to have windows with an active scrollable area augmented with a border region. Within the scrollable region, the icons appear at a single scale. As they are scrolled onto the border region they are represented by a black rectangle which is scaled non-linearly based on the distance from the edge. The black rectangles or "periphs", are rendered onto the border

region using anti-aliasing so that they are legible even if they are less than a pixel across. In the corner regions of the border, the non-linear scaling is applied in both directions.

A 2-D window containing icons and periphs is shown in Figure 7, on the color sheet. It is clear that there are other objects off the edge of the scrollable area. The border for the window may seem like a waste of space, but provides an affordance to manipulate the window.

5.1 User Study Response to Periphs

In the user study, we asked participants to move documents from a tray to a window and between windows. We then asked them to re-size the window so that the file icon became a periph that they had to find by scrolling.

Half of the participants were uncertain about the scrolling window, they felt it moved unexpectedly. Scrolling the contents of a window was achieved by dragging on the space between icons directly. This is contrary to the current Macintosh OS convention where the same action would bring about an area selection rectangle. Since the borders no longer had scroll bars we were exploring other ways to scroll windows. This approach revealed the need for more explicit affordances on the windows, or a different cursor state.

As for the periphs, 7 of the 10 understood and appreciated the idea. One participant thought initially that the periphs were a bug, but by the end of half an hour he thought they were "really cool". We had deliberately left periphs out of the introduction, so that we could assess user's reactions to encountering them interactively. The impression of them being a bug arose because they are small line-like elements which usually only appear in current interfaces due to redraw errors in drawing algorithms. However, once they were seen as deliberate their functionality was understood and appreciated. Another participant wanted to see a proxy view from a periph, which we had not considered. In general, the participants expressed admiration for periphs, which were seen as a good solution to a difficult problem.

6.0 Icons and Windows in Projective Spaces

One way to use space efficiently is to let the user change the scale of graphical elements while maintaining the global context. In addition to fish-eye views and perspective walls, there is the idea of “slapped-back” windows. When slapping back a window from screen-aligned (i.e. 2-D) to a slapped-back representation (such as with 3-D foreshortening), the contents of the window is remapped using a geometric transformation. Different image transformations have the following trade-offs:

6.1 Scaling in One or Two Dimensions

With uniform scaleing, the window and its contents are resized proportionally (Figure 8). This has the disadvantage that text and icons must be constantly resized and affordances may become too small to use. However, such an interface can be useful for navigating quickly in and out of large amounts of data points for example HotSauce [8], or Pad[14]. The uniform scaling case is equivalent to an interface in perspective projection where the elements remain parallel to the screen, e.g. "Workscape" [13], and more recently, "Data Mountain" [19] and Pad Prints [20].

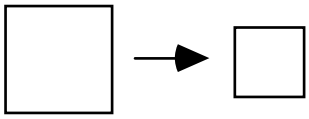

Fig. 8 A square scaled uniformly in two dimensions.

6.2 Perspective Projection

In a perspective projection, rectangles that are not parallel to the plane of the screen are transformed into quadrilaterals whose edges converge to vanishing points (Figure 9). This full perspective projection resizes and distorts icons and text continuously. However, it has the advantage that objects may be made smaller in a coherent way by moving them further away from the screen.

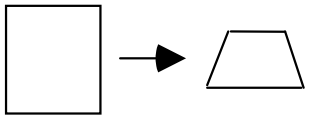

Fig. 9 A square slapped back in perspective projection.

6.3 Orthographic Projections

In an orthographic projection, a horizontal rectangle is scaled vertically by a constant factor and sheared horizontally by a constant angle (Figure 10). In our experiments the angle was chosen to be 45 degrees. This projection has the advantage that objects may be moved in depth without being resized. If a single orthographic projection is used, then 2-D artwork may be used to represent a 3-D object. This helps with efficient implementations employing sprite-based graphics.

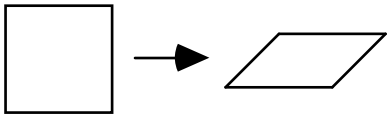

Fig. 10 A square slapped back in orthographic projection.

A more fundamental property of orthographic projections is depth ambiguity. The appearance of an object on screen will be independent of its depth. The only clue to depth is given by occlusion from other objects. This means that the depth representation can by a logical ordering of elements rather than an absolute position.

The advantage of this property of orthographic projections is that when objects have their ordering changed, their shape does not change as is shown in (Figure 11). By contrast, in a perspective projection, bringing a window nearer or further from the viewer would change the size and shape of its projection on the screen. It is possible to keep the outlines invariant in perspective projections as the ordering changes (Figure 12), but then the implied world-space scale of the two objects changes. It is this property that makes setting up perspective scenes so tedious in 3-D animation systems.

Needless to say, 2-D windows-based interfaces also exploit depth ambiguity since they are a special case of orthographic projections (with no scaling and zero shear). In 2-D, this leads to the intuition that we bring a window in front of another one. It is meaningless to say "I'll bring that window one inch nearer".

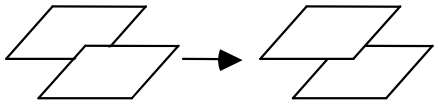

Fig. 11 Changing depth order in orthographic projection.

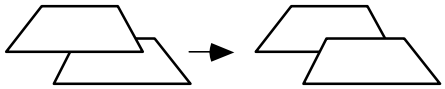

Fig. 12 Changing depth order in perspective projection.

Figure 13, on the color sheet, shows a window face-on and in slapped-back orthographic projection. The slapped-back version of the same window takes up much less screen area. The affordances for resizing, opening, closing, and switching projection, were put on the front edge of the window and remain constant in either projection.

Some design ideas in this paper, such as the periphs, are independent of whether the scene is 2-D or 3-D, others, like the current design of trays, depend on being in one of the 3-D projections.

6.4 User Response to Slapped-Back Windows

When working with the slapped-back window view, participants got used to navigating by the periphs on the edge of the window. However, they were working with a limited number of files (26) and folders (7) and worried about navigating a whole desktop. Seven of the participants did not like slapped-back windows, some of them felt: (1) they did not offer as much information as in the 2D window; (2) they looked distorted; and (3) they were not aesthetically pleasing when a number of them were open. The people who did like them did so because they would save screen space, and their ability to recognize the icons in this projection was not impaired.

7.0 Trays, Folders and Piles

As part of the exploration of 3-D metaphors for GUIs we wanted to come up with a new user-interface element which would be intermediate in functionality between a folder and a window. As stated in the introduction, folders have the problem that they indicate little about their contents and are not able to be browsed directly. Windows, on the other hand, use large amounts of screen space and display icons spaced far apart from each other. In designing trays we were inspired by the hierarchical list functionality of Macintosh list windows (Figure 2). Folder elements in a list have small affordances that allow them to be opened hierarchically. We wanted to create an equivalent representation for icons.

The Piles project [4] offered an interesting alternative mechanism, but it was not ideally suited to our purposes for several reasons. Since piles were generated by clustering icons together, piles provided no natural place for additional affordances. This was solved, by spontaneously generating a base (plinth) with additional buttons on it. A more serious problem was the geometry used to stack the icons. The structure of a pile naturally reveals just the edge of all the documents except the nearest one (Figure 14).

Fig. 14 Documents in a “pile".

This makes it very difficult to identify an individual icon within a pile except by browsing through each element in turn. The size of a pile is determined by the number of elements that have been heaped together. There is no natural way to control the number of elements displayed.

As an alternative, we devised an element called a “tray”. A tray is a horizontal box with a handle at one end. The handle may be used to grow or shrink the linear extent of the tray. Icons within a tray are shown rotated back around a vertical axis, thus creating a "tilted" icon. This enables the entire icon to be seen, rather than just the edge. Figure 15 illustrates this arrangement.

When many thumbnails are in a tray, the effect is rather like viewing a set of books on a shelf. This approach of stacking sideways on a tray, makes very good use of space. The number of icons displayed is controlled by the linear extent of the tray. A small border region shows periphs for

those icons not already visible. The name of the tray is indicated along the front edge. By placing one tray inside another it is possible to represent a hierarchy. A kinematic constraint system ensures that growing a tray also grows the parent tray on which it sits. This prevents one tray from overhanging another.

Fig. 15 Documents in a “tray".

Since folders are simpler and more compact than trays, it is desirable to support both representations. A 3-D interface should have an elegant and lightweight method to transform between the two. Note that when a tray is opened as a window, it leaves a rectangular footprint in its parent tray.

The major advantage of trays over folders is progressive disclosure while maintaining compactness. The major drawback of trays is that they can be abused. In particular, stacking a large number of trays on top of each other can entirely obscure an icon behind the stack. This is illustrated in Figure 16 on the color sheet. To solve this problem we might adapt the icon placement algorithm so that icons move to remain visible despite stacks of trays in front of them. This is an area for future work.

Finally, it is important to be able to see the names of icons to identify them. We have chosen not to show the names all the time (for reasons of compactness) but they must be available on request. This is invoked by placing the cursor over the tray while depressing the proxy-browse modifier key. There are two approaches to displaying names. One is to have the names appear in vertical banners above each icon. This is not very legible. A second approach is to have the names appear in captions, connected to the files by stringers. This arrangement is shown in Figure 17 on the color sheet.

While it looks unwieldy for permanent display, it is easy to browse temporarily. This approach does break down if too many names are viewed at once. The user finds it difficult to associate the names with the particular icons. In that case it is natural to employ the browsing mechanism for the individual documents by moving the cursor over the icons to see its full proxy view as in Figure 6.

7.1 User Study Response to Trays as Elements

In the study, we asked participants to drag and drop icons into a variety of trays and then to stack the trays to form a hierarchy. Though the concept of the trays was liked, certain features of this implementation were not: (1) the trays were deemed to be too drag intensive, (2) there was no quick way to view all the files not shown in a tray or to hide all the files in a tray, (3) the stacking order was thought to be counter intuitive (the parent tray is at the bottom of the pyramid rather than the top), and (4) it was disliked that the documents at the "front" of a tray (nearest the handle) were the first ones to disappear as the tray is resized. (This could be easily fixed by changing the layout algorithm).

Some participants had difficulty dragging files and trays. They were unable to tell when releasing the documents over the trays would mean that they would end up inside the tray rather than on the desktop. Users were constantly confused about what had happened to their files. As one participant said of the dropping process, "I feel like the space shuttle when it is trying to dock."

This problem was addressed in the next version of the system where a red line was drawn around the top surface of the tray when a currently grabbed document would be placed inside the tray if released.

8.0 Object Placement Algorithms

When implementing a 3-D GUI, it is necessary to take special care with icon placement algorithms (which in this case includes trays as well). Icon placement is an issue when a user drags an object from one place to another, for example rearranging icons within a window. This is especially important for overlapping 3-D objects. Two trays which overlap in 3-D space would violate ease of use requirements as well as breaking the assumptions of the sprite-based renderer.

For this version of 3Desque, a placement strategy was adopted as follows: Whenever an icon or tray was moved, it would have its vertical location snap to one of an evenly spaced set of rows. On being placed into the row, the icons to the left would be moved until an overlap no-longer occurred. This left-hand icon would then do the same to icons to its left, until an icon was encountered which did not need to move, or no more icons were found to the left. Icons to the right would be moved to the right in the same way. This has the effect of moving neighboring icons the least amount required to be consistent with not overlapping.

When a tray was being enlarged by dragging on the handle, the right-hand neighboring icons would be moved using the same algorithm. This was done on the fly, so the interface remains consistent in appearance during the dragging operation.

8.1 User Study Response to Object Placement

Participants were asked to move the icons and trays in between rows. Some people were uncertain quite where the icon or tray would land, when they released the item, but

once we improved the feedback, using the mechanism described in Section 7.1, people found this more predictable.

9.0 Window Hierarchy Display Using Beams

The use of a 3-D projection provided trays with a natural way to represent hierarchy. We wished to do the same for windows. To do this we adopted the technique of having transparent beams (Figures 18 and 19 on the color sheet), connecting a tray footprint to its corresponding window. As Loretta Staples pointed out in [9], "Transparency provides the opportunity for the simultaneous viewing of objects that might otherwise be obscured". Our display of hierarchy was similar in spirit to Cone Trees [5] and the Spiral Calendar [6].

Using these beams for slapped-back windows provided a natural graphical counter-part to the grow animations used in 2-D systems. (After clicking on a folder, the rectangles expand up before a window appears. This heralds its arrival and signals its parentage), unfortunately, in the user study, we found, that using beams to indicate parentage for several hierarchies at once, confused some users. A more restrained approach might be to only show the beams for the currently selected hierarchy.

10.0 Uses of Shading, Textures and Shadows

One disadvantage of orthographic projections is that they can look distorted and “flat”. A way to increase the sense of depth is to use depth cueing in which the objects became darker as they recede up the screen. Figures 18 and 19, on the color sheet, show the same scene with and without depth cueing.

Since the background and the scrollable regions of windows are textured, it is natural to use that texture to indicate something about the contents of the window or tray. To prevent visual clutter from obscuring periphs and control buttons, window and tray frames were kept untextured but they were colored using the average appearance of the texture. This average color became an additional method for identifying the folder even if too many icons or trays obscured the details of the texture. Figures 19 and 20 show the use of texturing to customize different windows and trays.

Finally to enhance the sense of visual realism, and so counteract the limitations of the projection, shadows were incorporated into the display engine. Since absolute depths are not available for windows, it was decided to only have shadows cast from objects onto those surfaces with which they were directly in contact, such as icons onto their parent tray and the tray onto the background in Figure 17. This use of shadows did tend to reinforce the sense of proximity for certain interface elements.

11.0 Conclusions

While the project began with an interest in 3D projections, the most successful elements, from the point of view of our users, were the use of pop-up proxies to show document contents, and periphs, to show additional context off the edge of windows. The trays were thought to be useful, but some users had problems manipulating them, and slapped-back windows were thought to be interesting but not as obviously useful as the other elements.

By building an entire system, rather than by studying the pieces in isolation, we were forced to address the behavior of elements as they interacted. This forced us to implement an automatic layout algorithm, which proved satisfactory at preventing overlaps, but still has room for improvement. The problem of stacked trays obscuring objects behind them was identified, but again needs further work. Our experience of testing the ideas, albeit informally, indicated that the use of 3-D elements to implement a graphical user interface provided real improvements in the user experience and was not just cosmetic.

12.0 Acknowledgments

We wish to thank Bruce Horn, Frank Casanova, Frank Crow and Apple Computer, Inc. for their support in the development and testing of 3Desque. We would also like to thank the reviewers for their ideas about how to explain our work more clearly.

13.0 References

[1] The idea of "flatland" is based on "Flatland: A Romance of Many Dimensions (London, 1884) and appears in "Envisioning Information" by Edward R. Tufte. Graphics Press (1990) 12.

[2] Malone,T.W. How do People Organize their desks? Implications for the design of Office information systems. ACM transaction on Office Information systems 1,1, (Jan. 83), 99-112.

[3] Apple Computer, Inc. Human Interface Guidelines: the Apple Desktop Interface. Addison-Wesley publishing Co., Inc. (1987).

[4] Mander,R., Solomon,G., and Wong,Y. A Pile based metaphor for supporting casual organization of information. ACM SIGCHI Conference. on Human Factors in Computing Systems proceedings. (1992), 627 - 634.

[5] Robertson,G., Mackinlay, J.D. and Card, S.K. Cone Trees: Animated 3D visualization of hierarchical Information. In Proc. ACM SIGCHI 91 Conference. on Human Factors in computing Systems (April 1991), 189-194.

[6] Card,S.K, with Pirolli,P., and Mackinlay,J.D. The Cost of Knowledge Characteristics Function. Display Evaluation for direct Walk Dynamic Information Visualizations. ACM SIGCHI 94 Conference. on Human Factors in computing Systems. (1994), 238-244.

[7] Robertson, G. Card,S.K and Mackinlay,J.D. Information Visualization Using 3D Interactive Animation. Communications of the ACM, (1993) 36 (4).

[8] http://www.xspace.net/hotsauce/index.html

[9] Staples, L. Representation in Virtual Space: Visual Convention in the Graphical User Interface. Proceedings from INTERCHI (1993), 348-354.

[10] Johnson, B. and Shneiderman, B. Space-filling approach to the visualization of hierarchical information structures. In Proceedings IEEE Visualization 191, pp. 284-291.

[11] Card, S.K., Robertson, G.G., & Mackinlay, J.D. The Information Visualizer, and Information Workspace. In Proceedings of CHI191, Human Factors in Computing Systems. ACM press (April-May 1991), 181-188.

[12] Mackinlay, J.D., Robertson, G.G. The perspective Wall: Detail and context smoothly integrated. in Proc. ACM SIGCHI 91 Conference. on Human Factors in computing Systems. (April 1991), 173-179.

[13] Ballay,J.M. Designing Workscape: An Inter disciplinary Experience. ACM SIGCHI 94 Conference. on Human Factors in computing Systems. (April 1991),189-194.

[14] Perlin,K. and Fox. Pad: An alternative approach to the computer interface. prc. SIGGRAPH 1994, ACM Press. (1993), 57-72.

[15] Sarkar,M., and Brown,M.H. Graphical Fisheye Views of Graphs. ACM SIGCHI 92 Conference. on Human Factors in computing Systems (1992), 83-92.

[16] Furnas,G.W. Generalized Fisheye Views. in Proc. ACM SIGCHI 86 on Human factors in Computing systems pages. (1986),16-23.

[17] Macromedia, Inc. Director Version 4, (1994).

[18] Nielson,J. Enhancing the Explanatory Power of Usability Heuristics. Proceedings ACM CHI 94 Conference, (Boston, MA, April 24-28, 1994) 152-158.

[19] Robertson, G., Czerwinski, M., Larson, K., Robbins. D.C., Thiel, D., and Maarten van Dantzich. Data mountain: Using Spatial Memory for Document Management. Proceedings of the 11th Annual UIST Conference. (Nov. 1998).

[20] Hightower, R.R., Laura T. Ring, Helfman, J.I., Bederson, B.B., Hollan, J.D. Pad Prints: Graphical Multiscale Web Histories. Proceedings of the 11th Annual UIST Conference. (Nov. 1998).