Techniques for Aligning Objects in Drawing Programs

Roope Raisamo
Kari-Jouko Räihä
Department of Computer Science
University of Tampere
P.O. Box 607 (Kehruukoulunkatu 1)
FIN-33101 Tampere, Finland
{rr,kjr}@cs.uta.fi

ABSTRACT

Object alignment is one of the basic operations in drawing programs. Current solutions provide mainly three ways for carrying out this operation: either by issuing an alignment command, or by using direct positioning with the help of gravity active points, or by making use of constraints. The first technique has limited functionality, and the other two may be mysterious for a novice. We describe here a new direct manipulation tool for alignment. We show that while direct manipulation helps to make the tool intuitive, it has through iterative design evolved into a tool that also offers functionality not found in current commercial products.

KEYWORDS: drawing programs, alignment tools, direct manipulation, two-handed interaction, iterative design

A shorter version of this paper was published in the Proceedings of the ACM UIST'96 Symposium on User Interface Software and Technology.

1. INTRODUCTION

Direct manipulation has for more than a decade been one of the preferred styles of interaction. Drawing programs were among the first applications that made heavy use of direct manipulation. Recent developments in drawing programs have concentrated on intelligent techniques, such as the use of constraints. The basic direct manipulation behavior has remained more or less the same.

A closer look at drawing programs shows that the real world analogy, on which direct manipulation is based, has not been used nearly as much as possible. In this paper, we shall focus our attention on the alignment tool. We shall develop a new interaction technique that is more direct and understandable than the traditional ways of carrying out the same operations.

In addition to the interaction technique itself, the process used in its development is also of interest. Through careful iterative design, we were able to create a tool that, besides being easy to use and intuitive, also offers new functionality.

This work had two driving forces. One was the examination of the new interaction idea. Another was related to a larger project that is carrying out research on multimodal interaction in general, and two-handed interaction in particular. Two-handed interaction increases a user's degrees of freedom, when the user can manipulate two closely related properties of the alignment tool at the same time [11, 12]. However, two-handed interaction is a technique that needs very careful planning, as the results of previous work in the area point out [24]. Because of the many functions provided by our tool, the use of two hands for controlling it proved useful for a smooth handling of the various properties.

The rest of the paper is organized as follows. First, we will present previous work related to our subject. We then discuss our first solution to the alignment problem. This solution had several problems, both conceptual and related to the implementation. We describe the problems and their solutions, and present our current version of the tool. We conclude by briefly evaluating the tool.

2. PREVIOUS WORK

Alignment is a basic property of current drawing programs. One frequently needs to have objects on the same level either horizontally or vertically. The objects are usually aligned by their center points or by the bounding box of the selected objects.

It is hard to track down the history of alignment tools in drawing programs. They were introduced to a larger user population through commercial products such as Apple's MacDraw [26], but had existed earlier in various research prototypes (see [6] for some references) and in Xerox's workstations [3]. Although much of the development of Xerox Star was based on careful user testing [4], the early development of alignment tools has not been documented.

2.1. Command Based Alignment Tools

The most common way to align objects uses a two-step procedure:

(1) select the objects to be aligned, usually by pointing and clicking with the mouse; and

(2) issue the appropriate alignment command.

The second step of this procedure does not make use of direct manipulation, thereby differing from many other operations typically found in drawing programs. While our main interest is in studying whether other techniques could outperform the traditional command based approach, it is illustrative to study briefly different possibilities used in issuing the alignment command in step (2) above.

A popular technique is to place the alignment commands in menus, as in the Micrografx Draw for Windows program in Figure 1. A problem with menu commands is that they are usually textual, and thus provide no visual cues to the user. While horizontal and vertical alignment may be evident concepts to an expert user, the same does not hold for novice users, who typically need to think of the orientation of the alignment. In fact, our informal observations of novice users indicate that they often confuse vertical and horizontal alignment, particularly in aligning by the middle or center points.

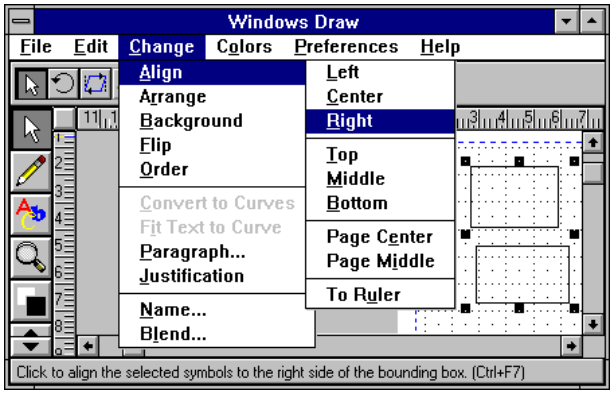

Figure 1: Traditional menu based alignment commands in Micrografx Draw for Windows.

Another possibility for issuing the alignment command is to use an alignment palette which provides alignment operations in graphical buttons. This method gives a user visual hints of what an operation would do if the user selects it. However, most palettes are so small that it may sometimes be quite difficult to see the meaning of the buttons. Figure 2 shows the alignment palette of ClarisDraw for Macintosh.

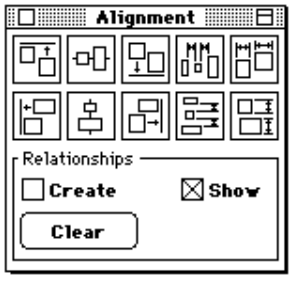

Figure 2: Alignment palette in ClarisDraw for Macintosh.

Visual cues are important, not only to make the choice of operations more intuitive, but to explain how the objects will move. The mere fact that the right sides of two objects are aligned does not indicate whether the leftmost object will move to the right, or the rightmost object to the left, or whether perhaps they will both move the same distance towards each other. This is something that has to be simply learned in menu based systems, but is directly visible in the palette.

The third method used in commercial products is to have only one alignment command in the menus, and to use a dialog box for providing additional parameters. This approach is used, e.g., in Adobe Illustrator for Macintosh (Figure 3).

The dialog box approach is obviously the slowest of the three for carrying out a basic alignment operation. On the other hand, it becomes possible to illustrate the effects of the operations, typically using some sample objects (Figure 3). Although these are not related to the actual selected objects, they may be helpful in teaching a novice the meaning of the operations. More importantly, it is possible to combine alignment in both dimensions into the same operation, a property that is not shared by either of the previous methods.

The dialog box of Figure 3 shares a conceptual problem with the hierarchical menus in Figure 1: the operations are always listed vertically, and the user has to map the command names to actual operations without visual cues that would support the mapping. Rotating one set of radio buttons so that the choices would be horizontal, matching the direction where the objects move, would already be an improvement [18].

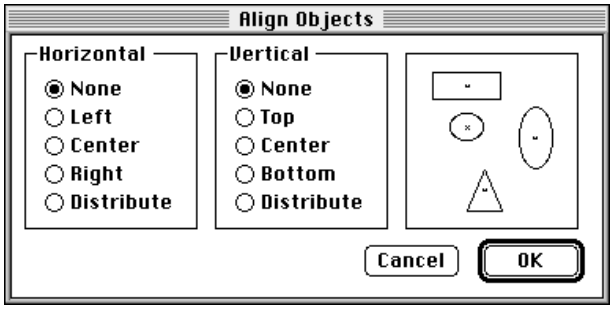

Figure 3: Alignment dialog in Adobe Illustrator 5.5 for Macintosh.

2.2. Advanced Techniques

In addition to being somewhat cumbersome to use, command based alignment tools offer only limited functionality. Advanced techniques for more complex alignment tasks have therefore been developed.

For a novice, the most natural way to align objects is simply to drag them to their desired positions. Such positioning is, of course, not very precise, but the technique can be enhanced by applying “gravity”. The drawing area can be equipped with gravity active points, and once an object is dragged close enough to such a point, it is snapped exactly to the point. To align a group of objects using this technique, each object still has to be dragged in position separately.

The most common form of gravity active points is a grid which can be turned on or off. Grid based alignment tools typically use a fixed grid, where the user is allowed to

control only the density of evenly spaced grid lines. Adobe Illustrator goes one better: it provides so-called guide objects which can be positioned and moved manually by the user. Figure 4 shows three guide objects: one vertical ruler, one horizontal ruler and one rectangular guide object. The guide objects are drawn with dotted lines.

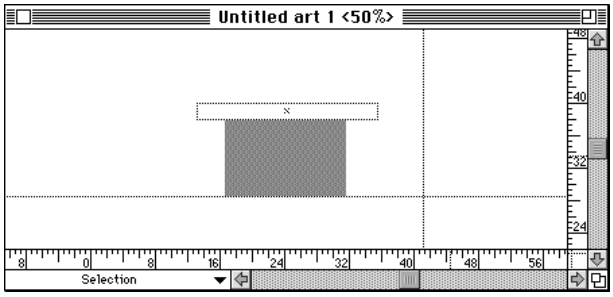

Figure 4: Three guide objects in Adobe Illustrator for Macintosh.

The power of gravity has been taken to a new level in a technique called snap-dragging [6, 7, 8], where the user can draw new alignment objects and define their gravity active points. This allows precision positioning along a rich set of curves, not just rectangular objects. The metaphor for alignment objects is that of draftsmen using ruler and compass techniques for precision drawing. A somewhat similar technique has been implemented (and patented!) in the commercial drawing program DrawingBoard [15].

Grids and other gravity based techniques are direct: positioning is caused by the user's direct actions. Another powerful, but indirect, technique makes use of constraints usually specified by the user. Constraint based tools relieve the user from fine grain positioning of objects. Instead, the system solves a set of constraints and automatically produces a transformed picture that satisfies the constraints. Such tools have a long history, going back all the way to Sketchpad [33].

Grids and constraints together provide so much power that to a novice, it may seem like magic. For grids alone, it may be hard for novices to understand their behavior or even to realize their existence [30, 31, 32]. Still, many authors [14, 34] have recently proposed an increased use of magic as the way to create the right illusions and to make the user feel more at home with a powerful interface. In the case of alignment tools, however, we feel that the operation is in most cases so simple that it should be realizable with (almost) no magic. The maxim "Simple things should be simple; complex things should be possible" was attributed to Alan Kay in [25]. While command based alignment tools do not make complex things possible, advanced techniques tend to achieve their power by making simple things harder, at least for a novice.

2.3. Direct Manipulation and Metaphors

The phrase "direct manipulation" was coined by Ben Shneiderman in 1982 [28, 29]. According to his definition, a direct manipulation application should:

- present the objects visually with an understandable metaphor;
- have rapid, complementary and reversible commands;
- display the results of an action immediately; and
- replace writing with pointing and selecting.

The concept of direct manipulation was carefully analyzed by Hutchins, Hollan, and Norman [22], who concentrate on two key issues:

- short distances, both from goals to the physical system ("gulf of execution") and from the physical system to goals ("gulf of evaluation"); and
- direct engagement.

The drawing programs discussed above do present the drawing objects visually, thereby providing the basis for direct manipulation. But it is also clear that in most of them the alignment operation is not visualized and does not give a feeling of direct engagement.

What is the reason for this? Is the alignment operation impossible to visualize? Certainly not. We shall present our solution later, but for now, think of this for a while: if you were asked to align a set of real world objects lying on your office desk, how would you proceed?

Metaphors are an essential part of human thinking and communication. That is why they are very important in direct manipulation interfaces, too. A proper selection of the metaphor can clearly help the learning and use of a computer system. Erickson [17] and Halskov Madsen [20] have listed issues that should be taken into account while designing a suitable metaphor. We have used these principles in the development of our alignment tool.

It is well known that metaphorical interfaces have their limitations: sooner or later the analogy breaks. Still, it is somewhat surprising that so little discussion on the current state of drawing programs can be found in the literature. The basic operations are based on direct manipulation; why is it so common to resort to commands when carrying out more advanced operations? Gravity based techniques use direct manipulation, but they extend the dragging operation to help in alignment, instead of providing a direct manipulation tool for the operation. For instance, Alan Cooper has directly manipulable tools on the top of the list in the chapter on new gizmos in his popular book [13].

To an expert, this may not seem as much of a problem. However, studies of CAD programs show that productivity is hampered partly because the possibilities offered by the programs are used poorly. Users tend to follow the habits learned in manual drawing. Bhavnani and John [5] suggest that the desirable practices should be taught to the users. Here we shall explore the other alternative, making the tools better fit the existing practices.

The system that is in spirit closest to ours is DrawingBoard [16] (not to be confused with the commercial system mentioned above that has the same name!), developed by Davor Dukic and Ian Benest in the University of York, UK. This system introduced a computer-generated counterpart for the ruler that is used to draw straight lines. However, this ruler is used in a bitmap based paint program while drawing on the screen: there is no alignment operation or tool in the program.

Of course, it may be that even if a natural visualization for the alignment operation can be found, it may not prove efficient in use. We intend to show that through careful design, an intuitive, functionally powerful and efficient tool can be created.

3. THE DESIGN CYCLE

We started our work by carefully considering the metaphor that would be appropriate for aligning objects. In this first stage we experimented with physical objects to guide us in the design.

Next, we developed the first prototype of a simple object-oriented drawing program called R2-Draw. It provides us with a convenient environment for exploring new interaction techniques. When the first implementation of our tool was ready, we ran a few informal user tests in our usability laboratory to see how the tool would behave in the hands of other users.

Several problems emerged, causing a redesign of many details (such as moving the center alignment property from a property of the tool into a property of the aligned objects), but the metaphor seemed to be intuitive. We tried to correct the previously found problems in the current implementation of the tool. We also added some new functionality that had not been implemented in the first prototype due to time restrictions.

In the following we first describe the initial design, and then elaborate on the role of two-handed control. The current version of the tool is then discussed in detail, followed by preliminary observations from informal usability tests.

4. THE INITIAL DESIGN

The first major design decision was to choose an appropriate metaphor for aligning objects. We decided to adopt a metaphor that would be as close to the real world as possible. This way the user can have an advantage of the previous experience in similar tasks with real objects.

We chose a ruler as the metaphor for our alignment tool, because many people are familiar with it and may have used it to align, or at least to push away, miscellaneous items found on their desks. And, even if they hadn't done it earlier, this action should be quite intuitive. Moreover, the ruler has a scale that can be used to measure the distances between objects during the alignment operation. The ruler is not used to draw straight lines, as in DrawingBoard [16], because that operation is obsolete in an object-oriented drawing program, which already has rubber band style drawing tools. We differentiate our tool from the usual use of the ruler by calling it a stick instead of a ruler. The name *alignment stick* refers to the property of the ruler which is its primary use in the alignment operation.

We also considered the situation in which some objects could be pulled with the stick instead of being pushed. This added functionality would have complicated the metaphor. Since our main targets were novice users and our main goal was an intuitive tool, we have currently decided against providing this mode.

Our real world based metaphor proved to be very useful since it allowed us to test the metaphor even before a single line of code was written. Our observations with a couple of bricks and one physical ruler showed us some of the potential problems related to the metaphor. To solve these, we used a pragmatic approach [10] in our metaphor design. According to this approach, if there clearly is a property of the metaphor which would be an obstacle for smooth functionality, it can be changed to act in a useful way.

Following this approach, we chose to leave out physical effects between objects. One example of these is the fact that no solid object can move through another solid object, which would make overlapping objects very difficult to handle. Also, if an object is pushed in the real world with a physical stick that is not parallel to the pushed side of the object, the object will eventually rotate so that it is lined with the stick – another feature that was judged undesirable.

There are two other exceptions from the real world metaphor: the length of the stick can vary and the objects can be locked. These exceptions were made to facilitate the manipulation of only a subset of the objects. If you use a ruler to push your pens on the desk, all of them will presumably move when they touch the stick. However, in a drawing program, some of the objects may already be where you want them to be. You don't want to spoil your previous work by pushing all of the objects. We therefore allow the user to change the length of the stick any time and to lock an object, so that it will not be affected by the stick. Each object has its own state (locked or unlocked) that can be changed from its pop-up menu.

Figure 5: The alignment stick in the active (a) and inactive (b) state.

The alignment stick is shown in its two operational states in Figure 5. The lower stick is in the inactive state, displayed when the user moves the pointer over an object with the mouse button not pressed. The upper stick is in the active state which means that when the stick is pushed, it aligns every unlocked object that it touches. The stick is implemented as a large mouse cursor, which changes to the normal pointer when moved outside of the drawing area.

The first implementation of R2-Draw is presented in Figure 6. The stick is activated and pushed upwards. Three of the objects have already been aligned by their bottom sides.

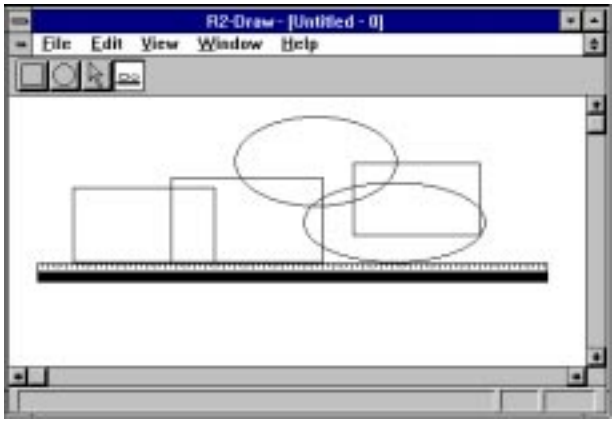

Figure 6: The first implementation of the alignment stick. The stick is moving up and has aligned three of the objects by their bottom sides.

In our first implementation the alignment stick had two operating modes: border alignment and center alignment. In the center alignment mode all the objects were

aligned by their center points until the stick was switched back to the border alignment mode.

Figure 7 shows the alignment stick in its typical use. First, the stick is in the inactive state (a). Next, the stick has been moved upward and activated. The stick has already touched the lower ellipse, which has moved a little upward (b). Finally, both of the ellipses are aligned by their bottom sides with a single action (c). In (d), we have grouped the ellipses and selected the center alignment property. We have also rotated the stick to do horizontal alignment for the group and the upper rectangle. The way of doing this will be discussed in the next section.

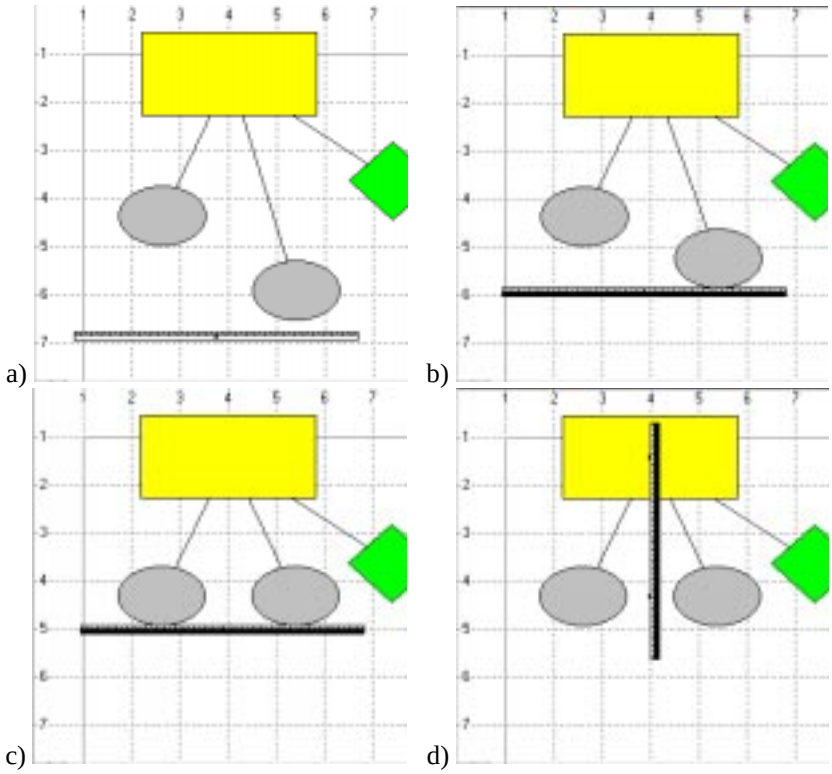

Figure 7: The alignment stick in use: a) The stick is inactive. b) The stick is activated and moving up. c) The stick has reached its intended position and the ellipses are vertically aligned by their bottom sides. d) The stick is rotated and the center alignment mode is selected. Two grouped ellipses and the rectangle are centered horizontally.

Our approach to the alignment operation solves the problem of visual cues we presented in our discussion of the command based alignment tools. The problem was that the user had no way of knowing how the objects will move when they are aligned with a command. With the alignment stick the objects are moved while they are pushed. So the user will see all the time how the objects are moving. We are using real time animation to show the current state of the drawing during these alignment operations. The user is also not likely to try to align the objects in the wrong direction (horizontal or vertical), because everything is expressed visually.

5. TWO-HANDED CONTROL

The alignment stick is a versatile tool. This can make it very difficult to use with just one pointing device. This was one of our main reasons for implementing it with a two-handed technique. A brief discussion of two-handed interaction follows.

Two-handed interaction is a relatively new technique in human-computer interaction. A well-known early experiment in the area was carried out by William Buxton and Brad Myers in 1986 [9]. This experiment already showed that using two hands in parallel can clearly speed up the work and make the learning of the work faster. One remarkable result of the experiment was that despite the sequential training of two operations, positioning and scaling, they were used in parallel by 6 of the 14 subjects without any suggestions by the researchers.

Stéphane Chatty [11, 12] has composed several guidelines for two-handed interaction. He divides two-handed systems into systems that use independent interaction, parallel interaction and combined interaction. In independent interaction the two hands are used simultaneously for two entire, distinct tasks, while in parallel interaction the tasks are equally important and somehow related. Combined interaction is the most sophisticated and preferable technique, where the two parallel tasks support each other and have a common goal. Kabbash *et al*. [24] have pointed out that combined interaction is the most understandable and probably the only form of two-handed interaction that is clearly better than traditional one-handed sequential interfaces. Such an implementation also exists for the snap-dragging technique (see Figure 14 in [8]).

We decided to use our primary pointing device (mouse) to control the position of the stick and the secondary pointing device (trackball) to accomplish actions that support the primary function of pushing objects to the desired position. The secondary action that we included in the first implementation was changing the width of the stick. This is done by rotating the trackball and can be accomplished while moving the stick with the mouse, resulting in parallel and combined interaction. The reason why these operations can be done in parallel is that they are used to accomplish the same goal and so the user need not divide his/her attention between many tasks, as would be the case in independent or parallel interactions. The choice of the trackball as the second input device was based on the results of Kabbash *et al.* [23], showing that a trackball was equally accurate with the non-dominant hand as a mouse and a tablet-with-stylus. Furthermore, the trackball is steady and this way overcomes some of the limitations (described in [2, 19]) of the non-dominant hand.

By using two input devices we achieved smooth control of the alignment tool despite the fact that it has so many functions. The tool is activated by pressing the left mouse button. It can be switched between the horizontal and vertical directions with the middle mouse button. The width of the stick is altered by using the trackball. Rolling the ball upwards or to the right increases the length and rolling downwards or to the left decreases it. The first implementation used the second input device only for this operation.

Following Chatty's [11] suggestion that everything should be possible without the second hand, when the right mouse button is pressed the mouse works as a substitute of the trackball. This behavior enables the use of the program while another hand or device is for some reason not available.

The user could switch between the border and center alignment modes by double-clicking the left mouse button or by issuing the command with a graphical button. Our user tests showed that it was difficult for users to remember that the center alignment mode existed, and even harder to remember how to switch to it.

6. THE CURRENT IMPLEMENTATION

We encountered a couple of problems while testing our first implementation with users. One major problem was that it was quite difficult to align and move objects exactly to the desired position. This was a consequence of our decision of not using any magic, but just intuitive direct manipulation. When the user tried to align the objects at the given position, the objects were frequently pushed too far. To fix this, the user needed to use another alignment operation and push the objects a little way back. But since the objects were of different size, the previous arrangement was lost during this corrective operation. That is why still another alignment operation was needed to accomplish the goal.

We made several improvements to the program to correct the problems of the first implementation. First, we added some magic: the common snap to grid operation. When applied with our alignment stick it functions so that when the stick is close to a line in the grid, the pushed objects are snapped to this line. This helps with those alignment operations that can use the grid, but it still does not solve the previous problem with arbitrary alignment operations.

The problem where the objects were pushed too far could be corrected in the first implementation also by switching to the pointer tool, by selecting the previously pushed objects and by dragging them back to the intended position. As you can imagine, this was quite a complex and time-consuming operation and therefore was not preferred by the users. We were convinced that we should not add any more functionality to the tool itself. So what we did was to add an easier way to switch between the alignment and pointer tools. We decided to assign the left button of the trackball to this operation. The button can be pressed any time during the alignment operation and as long as it is pressed, the user can manipulate the objects with the pointer tool. When the button is released, the program switches back to the alignment stick. To make things easier, the latest pushed objects are automatically selected during this temporary use of the pointer tool.

We also eliminated the double clicking transition to the center alignment mode. In fact, we removed the center alignment mode from the stick in order to simplify the tool. The first implementation of the tool had too much hidden functionality that we wanted to make more explicit. We accomplished this by taking some of the functionality out of the tool and moving it to the objects, an approach that is similar to the one in [21]. There, too, the same basic operation behaves differently with various objects, and this is reflected in the way that the objects are displayed.

In our case, we decided that center alignment should not be a property of the tool, but a property of the object that was aligned with the tool. This enabled us to use different alignment methods at the same time. We could, for example, align the center of a rectangle to the right side of another rectangle. This kind of alignment is not possible with traditional command based tools. After we were convinced that this was indeed a right decision, we decided to go even further: the user can currently set any

combination of nine alignment points, which are the center point and the eight points often used in object resizing handles. These points are set by using a separate tool to prevent overloading our stick metaphor (Figure 8). If no alignment points are set for an object, it is aligned by its border line.

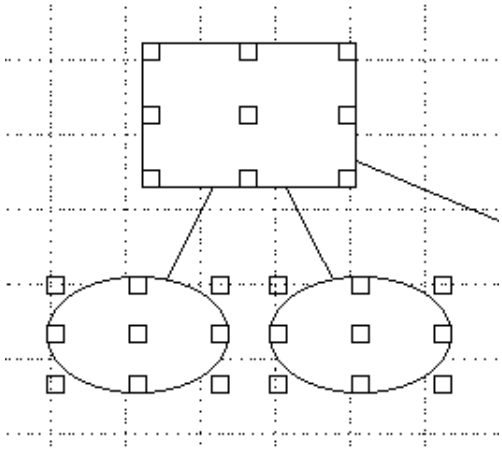

Figure 8: The alignment point setting tool in use. The user can click in the points that need to be active during the alignment operation. The use of this tool is optional.

We also gave the user more freedom in choosing the alignment direction. The stick can currently be rotated to any angle in addition to the previous horizontal and vertical directions. The rotation is accomplished by the trackball when the user has selected free angle alignment with the center mouse button. Simply rotating the trackball left or right rotates the stick around its center. After this addition, the alignment stick can be used just like the real one, only in two dimensions and remembering the few exceptions that we made to the ruler metaphor. Figure 9 summarizes the controls of the alignment stick.

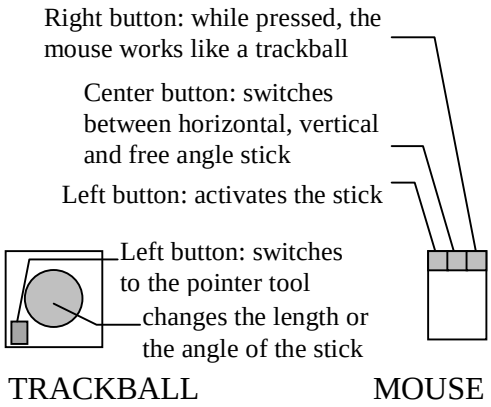

Figure 9: Controls of the alignment stick.

Figure 10 presents the current implementation where the user is drawing an ER diagram. The rectangles have been aligned by their top or right sides. The group of ellipses and the leftmost rectangle are centered horizontally, as in Figure 7d. From left to right and top to bottom, the tool palette has the following tools: pointer tool, ellipse tool, rectangle tool, line tool, rounded rectangle tool, diamond tool, the alignment stick (chosen) and a tool for selecting the alignment points. The buttons in the toolbar are

used to manipulate the files and the clipboard. In the upper right corner, 90° is the angle of the stick when rotating it to different orientations.

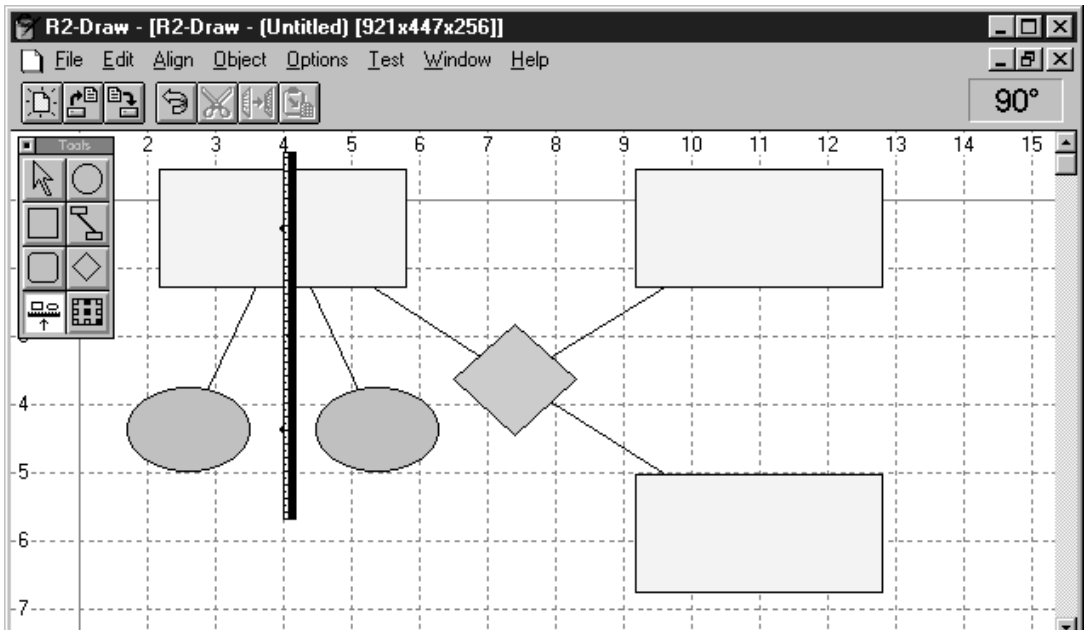

Figure 10: The current implementation of the R2-Draw program. The user is aligning objects horizontally by their center points.

7. DISCUSSION

In discussing the problems with direct manipulation, Hutchins, Hollan, and Norman [22] state: “Direct Manipulation interfaces have problems with accuracy, for the notion of mimetic action puts the responsibility on the user to control actions with precision, a responsibility that is often best handled through the intelligence of the system, and sometimes best communicated symbolically.” Our implementation of the alignment stick shows that indeed, a little magic is needed to handle precise positioning; but with this addition, direct manipulation provides the basis for a tool that is both intuitive and powerful.

Following our own goals of simplifying the alignment operations we came close to Chatty’s suggestion that the magic should be disabled when two hands are at work [11]. He applied that principle to a line segment drawing tool, where an invisible magic hand normally holds the other end of the segment. In Chatty’s work the magic was disabled so that the user could hold the other end with a secondary input device while dragging another with the primary input device. However, the magic that we added is not used in the same way as in the drawing of the line segments, but to help the user with alignment operations needing accuracy often difficult to achieve with human motor systems [2, 19]. This decreases the need for strict concentration during the operation.

It was interesting to note that novice users were from the beginning as fluent with the new alignment tool as they were with the older tools, whereas users that had previous experience with drawing programs required a short time period (15-20 minutes) to feel at ease with the alignment stick. This is yet another indication of the simple fact that

intuitive interfaces are based on learning and on previous experiences [22]. Seasoned users take some time to “unlearn” old habits and customs.

Because of the very nature of the alignment stick, it removes one step in the alignment process when compared with command based tools. In command based alignment, the user has to select the objects to be aligned and then issue the command. With our tool, the selection is not needed in the basic operations. The only thing that the user has to do is to push the objects. We believe that this simplification can have an effect on the performance of experienced users, too. This will be investigated during our follow-up studies.

On the other hand, it should be pointed out that advanced gravity based techniques, such as snap-dragging, allow alignment along a rich set of shapes, whereas the stick only provides linear alignment. Furthermore, the stick only aligns entire objects while other techniques can align at the level of individual control points. We have tried to achieve a balance between functionality and intuitiveness.

8. CONCLUSIONS

Careful design of the alignment tool gave us two valuable benefits: intuitive use and added functionality. This functionality developed even further during our iterative design process and informal user tests. Currently we are carrying out extensive usability testing to compare our alignment tool to the traditional ways of doing the same operations and to test our metaphor with different kinds of users. We also plan to experiment with large drawings, which may produce problems that do not come up with small examples and uncrowded screens.

Of course, object alignment is not one of the major problems or obstacles in current user interfaces. However, while new applications bring new problems into focus, it is also important to sit back and analyze carefully the established methods and techniques. At the operating system level this is already common practice: window manipulation, menu presentation and scroll bars have been studied empirically and improvements to common solutions have been suggested. Our work focuses in a similar way on one common primitive of drawing programs.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Computer Access Technology Corporation, Santa Clara, California. They gave into our use their ACCESS.bus [1] Windows application development kit, with which we did not need to focus on the hardware issues of two-handed interaction during our research.

REFERENCES

1. ACCESS.bus Technology Information Sheet. Computer Access Technology Corporation. [http://www.catc.com/ab_tech.htm]

2. J. Annett, M. Annett, P. T. W. Hudson, and A. Turner, The control of movement in the preferred and non-preferred hands. *Quarterly Journal of Experimental Psychology 31*, 1979, 641-652.

3. Ronald M. Baecker and William A. S. Buxton, Case study D: The Star, the Lisa, and the Macintosh. In *Readings in Human-Computer Interaction, A Multidisciplinary Approach*, R. M. Baecker and W. A. S. Buxton (Eds.), Morgan Kaufmann, 1987, 649-652.

4. William L. Bewley, Teresa L. Roberts, David Schroit, and William L. Verplank, Human factors testing in the design of Xerox's 8010 "Star" office workstation. *Human Factors in Computing Systems, CHI '83 Conference Proceedings*, 1983, 72-77.

5. Suresh K. Bhavnani and Bonnie E. John, Exploring the unrealized potential of computer-aided drafting. *Human Factors in Computer Systems, CHI '96 Conference Proceedings*, ACM Press, 1996, 332-339.

6. Eric Allan Bier and Maureen C. Stone, Snap-Dragging. *Proc. SIGGRAPH'86, ACM Computer Graphics, 20 (4)*, 1986, 233-240.

7. Eric A. Bier, Snap-Dragging: Interactive Geometric Design in Two and Three Dimensions. Technical Report EDL-89-2, Xerox PARC, September 1989.

8. Eric A. Bier, Maureen C. Stone, Ken Fishkin, William Buxton, and Thomas Baudel, A taxonomy of see-through tools. *Human Factors in Computer Systems, CHI '94 Conference Proceedings*, ACM Press, 1994, 358-364.

9. William Buxton and Brad A. Myers, A study in two-handed input. *Human Factors in Computer Systems, CHI '86 Conference Proceedings*, ACM Press, 1986, 321-326.

10. John M. Carroll, Robert L. Mack and Wendy A. Kellogg, Interface metaphors and user interface design. In *Handbook of Human-Computer Interaction*, M. Helander (Ed.), North-Holland, 1988, 67-85.

11. Stéphane Chatty, Issues and experience in designing two-handed interaction. *Human Factors in Computer Systems, CHI '94 Conference Companion*, ACM Press, 1994, 253-254.

12. Stéphane Chatty, Extending a graphical toolkit for two-handed interaction. *ACM UIST '94 Symposium on User Interface Software and Technology*, ACM Press, 1994, 195-204.

13. Alan Cooper, *About Face: The Essentials of User Interface Design*. IDG Books, 1995. [http://www.idgbooks.com:80/whats_hot/AboutFace]

14. Andreas Dieberger, On magic features in (spatial) metaphors. ACM SigLink Newsletter 4 (3), December 1995. [http://www.gatech.edu/lcc/idt/Faculty/andreas_dieberger/magic_features.html]

15. DrawingBoard. Ashlar Inc. [http://www.drawingboard.com/info.html]

16. Davor Dukic and Ian Benest, DrawingBoard: Adopting office metaphor for page composition. *Sun ´91 Conference Proceedings*, Sun UK User Group, Buntingford, UK, 1991, 227-239.

17. T. Erickson, Working with interface metaphors. In *The Art of Human-Computer Interface Design*, B. Laurel (Ed.), Addison-Wesley, 1990, 65-73.

18. James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

19. Yves Guiard, Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *The Journal of Motor Behavior 19 (4)*, 1987, 486-517.

20. Kim Halskov Madsen, A guide to metaphorical design. *Communications of the ACM, 37 (12)*, 1994, 57-62.

21. Stephanie Houde, Iterative design of an interface for easy 3-D direct manipulation. *Human Factors in Computer Systems, CHI '92 Conference Proceedings*, ACM Press, 1992, 135-142.

22. Edwin L. Hutchins, James D. Hollan, and Donald A. Norman, Direct manipulation interfaces. In *User Centered System Design*, D. A. Norman and S. W. Draper (Eds.), Lawrence Erlbaum, 1986, 87-124.

23. Paul Kabbash, I. Scott MacKenzie, and William Buxton, Human performance using computer input devices in the preferred and non-preferred hands. *Human Factors in Computer Systems, INTERCHI '93 Conference Proceedings*, ACM Press, 1993, 474-481.

24. Paul Kabbash, William Buxton, and Abigail Sellen, Two-handed input in a compound task. *Human Factors in Computer Systems, CHI '94 Conference Proceedings*, ACM Press, 1994, 417-423.

25. Daniel E. Lipkie, Steven R. Evans, John K. Newlin, and Robert L. Weissman, Star graphics: An object-oriented implementation. *Proc. SIGGRAPH'82, Computer Graphics 16 (3)*, July 1982, 115-124.

26. *MacDraw Manual*. Apple Computer, Inc., 1984.

27. Ken Pier, Eric Bier, and Maureen Stone, An introduction to Gargoyle: an interactive illustration tool. In *Document Manipulation and Typography, Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography (EP88)*, J. C. van Vliet (Ed.), Cambridge University Press, 1988, 223-238.

28. Ben Shneiderman, The future of interactive systems and the emergence of direct manipulation. *Behaviour and Information Technology 1*, 1982, 237-256.

29. Ben Shneiderman, Direct manipulation: A step beyond programming languages. *IEEE Computer 16 (8)*, August 1983, 57-69.

30. M. V. Springett, A. S. Grant, Interface semantics and users' device models: Identifying evaluation issues for direct manipulation design. *Proceedings of the HCI'93 Conference on People and Computers VIII*, 1993, 249-265.

31. M. V. Springett, A. S. Grant, A. G. Sutcliffe, Interface semantics and procedural knowledge: A study of novice understanding of MacDraw. *East-West International Conference on Human-Computer Interaction: Proceedings of the EWHCI'93*, Springer-Verlag, 1993, 241-256.

32. A. G. Sutcliffe, M. V. Springett, From user's problems to design errors: Linking evaluation to improving design practice. *Proceedings of the HCI'92 Conference on People and Computers VII*, 1992, 117-134.

33. Ivan E. Sutherland, Sketchpad: A man-machine graphical communication system. *AFIPS Conference Proceedings 23*, 1963, 323-328.

34. Bruce Tognazzini, Principles, techniques, and ethics of stage magic and their application to human interface design. *Human Factors in Computer Systems, INTERCHI '93 Conference Proceedings*, ACM Press, 1993, 355-362.

APPENDIX

Our implementation environment is shown in Figure 11. It was originally a 66 MHz Intel 486DX2 and currently a 90 MHz Intel Pentium. The current system has a fast Diamond Stealth Video VRAM graphics accelerator to speed up the bitmap copy operations that are constantly needed during the alignment animation. The two-handed interface is based on the ACCESS.bus serial interface [1], originally developed by Digital Equipment Corporation and Philips and currently provided as a completely open industry standard. The ACCESS.bus can be used to connect up to 125 parallel peripheral devices to a computer and there is a working multiple mouse driver available. Although we are not satisfied with every aspect of the driver software, the bus was a relatively easy way to study two-handed interaction in the Microsoft Windows environment. The implementations have been programmed in C++ using the Borland C++ for Windows compiler. The Windows interface is written using Borland's Object Windows Library framework.

Figure 11: The implementation environment. A three-button mouse is on the right and a trackball on the left.