Two Pointer Input For 3D Interaction

Robert C. Zeleznik and Andrew S. Forsberg
Brown University site of the
NSF Science and Technology Center for
Computer Graphics and Scientific Visualization
Providence, RI 02912
{bcz,asf}@cs.brown.edu

Paul S. Strauss
Silicon Graphics Computer Systems
pss@sgi.com

ABSTRACT
We explore a range of techniques that use two hands to control two independent cursors to perform operations in 3D desktop applications. Based on research results in 2D applications, we believe that two-handed input provides the potential for creating more efficient and more fluid interfaces, especially for tasks that are context-sensitive or that have many degrees of freedom. These tasks appear frequently in 3D applications and are commonly broken down into a series of sequential operations, each controlling fewer degrees of freedom – even though this may dramatically change the character of the task. However, two-handed interaction, in theory, makes it possible to perform the same tasks using half the number of sequential steps since two previously sequential operations can be performed simultaneously. In addition, many forms of two-handed interaction may be simpler to use and to understand since they correspond to common interactions in the physical world. It is significant when tasks that need to be broken down into two sequential single-cursor steps, can be performed as a single fluid operation using two cursors.

CR Categories and Subject Descriptors: I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques; I.3.1 [Computer Graphics]: Hardware Architecture - Input Devices.

Additional Keywords: interaction, two-handed input, two cursors, direct-manipulation, 3D applications, 3D modeling, gestural input.

INTRODUCTION

We explore a range of techniques using two hands to control two independent cursors to perform operations in 3D desktop applications. The techniques presented here are a sampling of the possible interactions that are enabled by having both hands represented in the user interface. The desire to have two hands represented in the user interface stems from observations of both how people interact with traditional user interfaces and how people interact with objects in the physical world.

The traditional approach to user interface design for 2D computer applications is to channel all user input through a single 3-button (or fewer) mouse and a keyboard. Therefore, users typically interact with computers by either typing with both hands, or by using the mouse with their dominant hand (DH) and a small number of modifier keys with their non-dominant hand (NDH). Similarly, almost every 3D desktop interface utilizes the keyboard and the mouse (or a substantially equivalent input device like a joystick or tablet) for all interaction. There are a few applications that also incorporate a dialbox, which is often operated by the NDH. However, even in these applications, the interaction style is still predominantly sequential with the DH manipulating the mouse to make a selection, followed by the NDH modifying a continuous parameter of the selection with one or more dials.

In the physical world, tasks are often performed with both hands acting in concert, but not necessarily symmetrically, to achieve a single goal. For example, research has shown that when people are allowed to write on a piece of paper using only one hand, they are generally 20% less efficient than if they are allowed to use two hands [5]. The NDH allows the user to fluidly manipulate the orientation of the paper so that the DH can write most effectively. Similar examples exist for many other operations involving physical object interaction.

In addition, user interface research in 2D applications has also suggested that, in some cases, interfaces for compound continuous tasks that use one-handed input are less natural and less efficient than interfaces which split those compound tasks into possibly parallel sub tasks controlled by both hands [2]. In particular, Kabbash *et al* [8] experimented with a 2D interface for the compound task of drawing multiple line segments and assigning a different color to each segment. They noted both that two-handed interfaces, if designed properly, can increase performance over single-handed interfaces, and also that two-handed interfaces can fare worse than single-handed interfaces, especially when cognitive load is increased.

Our work, therefore, attempts to augment the typical one-handed 3D interaction model with a judicious use two-handed interfaces for particular tasks where we feel there is a strong possibility of improved performance. We informally evaluated our techniques with users in our lab, but we did not conduct any controlled user studies to further validate our belief that these techniques have performance advantages over single-handed interaction. In general, all of the two-handed interaction that we demonstrate gracefully degrades into conventional single-handed interactions merely by not using the NDH. The interfaces that we demonstrate have been implemented in an Inventor application [14], a VRML browser [15], and in a non-conventional gestural 3D application, Sketch [16]. We show that two-handed interaction can be used effectively to augment the expressive power of conventional 3D widgets, in addition to generating a novel, and arguably more efficient, style of fluid 3D interactions not possible with a single hand.

Several of the operations we present (eg. polyline editing, ungrouping, and object translation and rotation) are not specifically 3D operations, and may apply to 2D applications as well.

PREVIOUS WORK

Several research systems have attempted to incorporate two-handed interaction into 2D desktop applications. Krueger did pioneering research as part of his VIDEOPLACE system in which a camera tracked the position and motion of the users' hands in order to manipulate on-screen objects [9]. Bier and Buxton developed a two-handed tool glass and magic lens system for use in a 2D drawing program [1]. In their system, the user controlled two cursors. The NDH controlled the coarse placement of the tool, while the DH performed more precise selection and drawing operations. More recently, [10] presented a variety of two cursor interfaces for performing direct manipulation operations on objects in a 2D drawing program. Our work shares aspects from both the tool glass and Alias systems and adapts them to desktop 3D tasks.

In addition to these systems, a wide range of two-handed interaction has been incorporated into virtual reality systems. Notable work in two-handed interaction in a fishtank environment was was done by [7]. He embedded a six degree-of-freedom (DOF) tracker[1] in a prop[2] that was controlled by the NDH and then manipulated another 6 DOF tracker with the DH in order to define slicing planes through a 3D volume. More recently Shaw has developed a 3D modeling application in which the user controls two six DOF trackers [13]. The tracker controlled by the NDH is used to provide context (low frequency temporal-spatial scales of motion) including menu operations, rigid manipulation of the entire model and constraint mode specification. Asymmetrically, the bat controlled by the DH is used for more precise picking and fine manipulation (high frequency temporal-spatial scales of motion).

Two handed interaction is increasingly used in immersive virtual reality (VR) environments, with the current state-of-the-art exemplified by MultiGen's SmartScene product [11]. A user wears two data gloves in their system in order to perform a variety of 3D operations and gestures. There are four styles of interaction in SmartScene : 1) the user gestures with either or both hands to perform an operation or enter an interaction mode; 2) the user holds a menu in the NDH and makes a selection from it with the DH; 3) the user manipulates a 3D object using both hands simultaneously; and 4) the user gestures or performs direct manipulation with both hands to navigate around the 3D environment. Additional research in two-handed interaction for virtual reality is ongoing in many places and is increasingly based on fluid simultaneous manipulation of 3D objects and widgets with both hands [12] [7].

[1]a device that reports its position and orientation relative to fixed source
[2]a physical stand-in for a virtual object

In contrast to this body of prior work, we have concentrated on two-handed desktop interfaces for 3D applications because we feel that VR interfaces suffer a number of disadvantages that disappear in desktop environments. Desktop environments utilize input devices that are generally more accurate and more controllable than six DOF VR trackers. Desktop interfaces also allow a convenient, useful rest position that is not fatiguing. Lastly, desktop systems are currently far more pervasive and cheaper than VR systems.

The interfaces we present differ from existing two-handed 2D desktop interfaces because we address more complex, higher-DOF 3D operations. In addition, our interfaces differ from VR solutions because we use input devices with a combined 4 DOFs, whereas VR applications typically use input devices with a combined 12 or more DOFs. In some cases, we borrow the character of certain VR interface techniques, but still must adapt them to input devices with lower DOFs.

TWO-CURSOR INTERACTION TECHNIQUES FOR 3D

The following sections describe the 3D interaction techniques we have considered for use with two-handed, two-cursor input and the operations we have implemented.

TRANSFORMATIONS

In general, a single cursor can control only two DOFs; although, it is possible to map a two DOF input device to a three DOF operation by introducing a non 1-1 mapping from the user input to the 3D operation [16] [3]. Interfaces that rely on this mapping are usually hard for a user to predict. For example, except in simple cases, users generally only achieve desired results with virtual sphere rotation through trial-and-error.

We attempt to avoid this confusion by decomposing operations that have too many DOFs into suboperations, each having no more DOFs than our input devices. With our techniques it is possible to specify directly any configuration of the task DOFs within the range of the input. We expect that there may be 1-many techniques that are worth exploring, although we did not develop any usable examples. Furthermore, we found a number of 1-1 and many-1 mappings that were counter-intuitive and difficult to control. The following sections present only techniques that we believe are useful based on informal observation of our users.

Translating and Rotating objects We have developed an interaction for simultaneously translating and rotating an object in a plane embedded in 3D.[3] This technique makes it possible to make both small and large adjustments in position and orientation using a single fluid interface operation.

The three continuous DOFs in this interaction operation are mapped to the four continuous DOFs of the two cursors. The NDH cursor is used to select a point, P, on an object that is the base of an axis of rotation, A (see Figure 1). The DH cursor initially selects a second point, D. If the NDH cursor moves, the object will translate along the plane perpendicular to A in order to maintain pick correlation. In addition, the object always rotates so that D lies along the line between

[3]In Sketch, objects have a plane of interaction associated with them upon creation. In our Inventor implementation, the axis of rotation is determined by the manipulator geometry that the cursors are near when an object is grabbed. In general, choosing the axis of rotation is application-specific and may included a setup operation to explicitly define a plane in 3D.

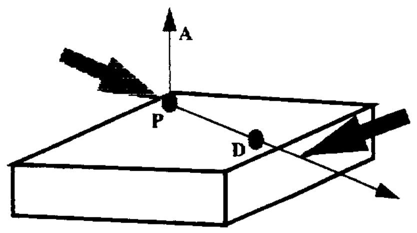

Figure 1: The DH cursor rotates the object around the vector A based at P. When the NDH cursor moves, the object translates and also rotates around the vector A so that the point D will lie along the line between the two cursors.

the two cursors. (Exact pick correlation with the DH cursor may not be maintained since the distance between the two cursors may change.) Thus, to rotate the object (around the NDH cursor), the user simply manipulates the DH cursor. To just translate the object, the user can either move both cursors by the same amount and in the same direction (which is hard to do accurately), or the user can easily resort to moving the object with just one cursor. When the user moves both cursors by different amounts, the object will both translate and rotate.

Rotation To specify the 3D orientation of an object, we modify the standard virtual sphere function as follows. First

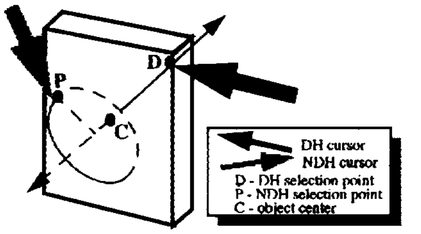

Figure 2: DH defines virtual sphere through point D. NDH rotates object around the axis CD.

we select a point on the surface of an object, D, with the DH cursor (see Figure 2). Next, we define the radius of the virtual sphere as the distance between the object's center, C, and D. Then standard virtual sphere rotation is performed as the DH cursor moves. The NDH cursor is used to select a second point, P, on the surface of the object. When the NDH cursor moves, the object rotates and P moves along the circle, defined by sweeping the line from P to D around the line from C to D, to maintain pick correlation. Note, the NDH cursor will always pick correlate with point P because: 1) we warp the NDH cursor when the DH moves, and 2) we constrain the NDH movement to be along the circle.

Scaling Two cursors also enable a straightforward scaling operation that is analogous to stretching a piece of rubber in the physical world. In this technique, the user places both cursors over an object. During subsequent interaction, the object scales and translates so that these two points will remain "pinned" under the cursors. Once again, the operation occurs in a pre-specified plane embedded in 3D, although this time the four DOFs of the cursor are mapped to four DOFs of the object (two scale DOFs and two translational DOFs) so that it is possible to maintain exact correspondence between the cursors and the two initially selected points on the object. In some cases, however, it is desirable to scale in only one dimension instead of two. This interaction is effected by pinning a point on the object to the NDH cursor, and then by using the DH cursor to stretch the the object along the single scale axis so that pick correlation is most closely maintained.

Vertex, Face, and Edge editing To manipulate individual features of a polyhedral object, such as edges, faces or vertices, we explored a number of related techniques that are similar to those used in Cosmo Worlds [4]. These techniques use the NDH to specify a motion constraint for the feature selected by the DH cursor. For example, the NDH can be positioned over a face of the polygonal object to define an interaction plane. Then the DH selects another edge, face or vertex, and moves it in a plane parallel to the plane selected by the NDH.

CAMERA CONTROLS – Specifying a viewpoint

Specifying a viewpoint in a 3D environment involves positioning the camera, orienting the camera, and setting the zoom factor (zoom is typically the focal length in a perspective projection and the film width and height in a parallel projection). All of our camera controls have been developed only for parallel projections, although we believe that with only minor modifications our controls should also be effective with a perspective camera.

Camera and Object Manipulation A natural operation for the NDH is to pan the camera about the scene. This is analogous to the way the NDH might position a piece of paper on a table. The DH is free to perform other operations such as translating an object or, in the Sketch system, drawing a gesture line that extends off the screen. This interaction demonstrates the use of two hands to simultaneously perform independent tasks. In addition to panning the camera, the NDH can be used in the Sketch system to zoom or rotate the camera while the DH manipulates objects.

Position, Zoom, and Orientation In general, to specify a camera's position, orientation and zoom, the user must control seven DOFs. In the Sketch system, however, we restrict the camera's rotation so that it cannot roll (i.e., the normal of the floor plane always projects to a vertical line on the film plane) – in effect reducing camera viewpoint specification to a six DOF problem. There is still no direct way to directly specify six DOFs with only four DOFs of input. We chose to decompose the camera specification problem into two separate interactions, each of which controls four DOFs. The techniques we describe assume that there is a pickable background everywhere in the scene.[4]

The first technique provides simultaneous camera control for zoom, yaw and two translational DOFs.[5] The user selects two points, one with each cursor, and then as the cursors move, the camera is translated, rotated around the normal to the ground plane, and zoomed so that the two selected points stay under the cursors (see Color Plate 2).

This process can be described by the following pseudocode:

- Pan the camera parallel to the film plane so that the point selected by the NDH maintains pick correlation.

[4]In situations where there is no pickable background, Sketch assumes a default surface which can result in non obvious manipulations.

[5]This technique is similar to the orbit and zoom technique found in the immersive VR SmartScene Application by MultiGen [11].

• Rotate the camera around the normal to the ground until the point selected by the DH cursor most closely maintains pick correlation.
• Scale the the camera's film plane by the ratio of the distance between the two selected points to the distance between the two cursors.
• Translate the camera again so the point selected by the NDH once again maintains pick correlation.

The second technique provides simultaneous camera control for tilt, yaw and two translational DOF's. Again, the user selects two points in the scene, then as the cursors move, the camera rotates and translates so that the selected points continue to pick correlate. As long as the distance between the two grabbed points is greater than or equal to the distance between the two cursors on the film plane, then both cursors will pick correlate exactly; otherwise the DH cursor pick correlation will drift off.

CAMERA CONTROLS – Navigation

Navigation in a 3D environment involves specifying how the camera should move from one position to another. Parameters which need to be specified include viewing direction and velocity. We augmented a standard VRML [15] navigation viewer with a second cursor that controls the camera's elevation off of the floor plane and the camera's tilt. Since the camera is continuously moving through a 3D scene, we do not use any pick correlation techniques.

Flying controls are appropriate for some scenes. In these situations, we modify the above technique by having the NDH control the camera's elevation off the floor and the camera's roll.

EDITING OPERATIONS

Line segments Two cursor input enables additional flexibility when creating and editing 2D line segments. [10] presents examples of using two cursors to create a variety of 2D objects including rectangles and lines. We use essentially the same techniques to draw line segments in the Sketch system. The starting and ending points of the line segment pick correlate with the NDH and DH cursors respectively.

In addition, we have adapted this rubber-banding technique for use within the Sketch application when axis-aligned lines need to be drawn. In these cases, we use the DH cursor initially to specify the starting point of a line segment, and then we rubber-band an axis-aligned line so that it most closely matches the current position of the DH cursor. Since frequently the user may decide after having started drawing the line that the starting point of the line segment is in the wrong place, we use the NDH to translate the entire line segment while the DH continues to rubber-band the line segment.

Polylines In addition to drawing line segments, we also use two-cursor input for editing polylines. This interaction allows the user to simultaneously manipulate two vertices of a 2D polyline. Although this interaction may seem simplistic, it turns out to be very useful when editing most shapes, and is particularly appropriate for editing long line segments that otherwise would require large movements of a single cursor in order to make potentially small changes.

Interactive shadows We have developed two new techniques for interacting with Interactive Shadows [6]. Interactive shadows are a mechanism for performing constrained planar translation, and single axis rotation of 3D objects. These shadows lie on a shadow plane that acts as if there were a directional light source shining in the opposite direction of the shadow plane normal. Our two techniques can be used independently or in conjunction with one another.

In the first technique, one cursor is used to pin down the shadow, while the other cursor drags the associated object in a straight line either closer to, or further from the shadow (see Color Plate 3). The meaning of this interaction is to translate the object along the normal of the plane where the shadow is defined – essentially adjusting the height of the object above the shadow plane.

Alternatively, the user may hold the object still with one cursor and translate the object's shadow in a straight line either closer or farther from the object with the other cursor. This interaction is particularly appropriate for the Sketch system, although it is meaningful elsewhere as well. The effect of pinning the object and moving the shadow is to translate the object along the viewing vector through the object's center. In essence, the object has the proper projection on the film plane, but its "depth" along the viewing vector is being adjusted.

Ungrouping In most 2D and 3D graphics systems, objects can be explicitly grouped together so that if one moves, so does the other. However, one cursor systems generally provide only menu operations for temporarily or permanently breaking that grouping relationship. With a two cursor system, it is straightforward to simply pin one object down with one cursor while manipulating the other object with the other. Similarly to the shadow manipulation techniques, both objects can be grabbed and moved independently, thus overriding the group constraint.

Duplication [1] show how tool glasses and magic lenses can be easily and effectively manipulated with a two cursor system. In the Sketch system, we have begun to incorporate the tool glass methodology for duplication operations. The DH cursor is used to draw a lasso around a group of objects to be duplicated. Then the NDH cursor is dragged to point within this lasso. The position of the NDH cursor establishes a context in which the lassoed objects act as an "ink well." If the DH cursor clicks on any object within the lasso, then that object, along with all the other objects within the lasso, is duplicated and dragged by the DH cursor. Finally, the NDH can drag the lassoed objects around in order to make a moving "ink well" of objects that can be duplicated with the DH like a tool glass.

DISCUSSION

Using two cursors in 3D environments allows the user to directly control more DOFs simultaneously than with a single cursor. However, not all mappings of application DOFs to cursor DOFs are understandable and controllable by the user. In particular, we experimented with a number of mappings for camera navigation that we thought would be effective, but in practice turned out to be difficult to control. The best choice of mappings seem to be the ones that have the strongest physical analogs. Thus, techniques in which one hand pins down a point in the scene, and the other hand manipulates relative to that point seem to work particularly well. In contrast, our early experience with users is that they find it more difficult to control entirely independent DOFs with two hands unless the interaction happens to correspond to physical intuition.

In situations where the interaction is fluid and understandable, two cursor interaction allows users to quickly and efficiently perform more complex 3D operations than with single cursor techniques. In addition, our experience with users is that they find the techniques to be more appealing than single cursor techniques.

Physical constraints of two handed interaction

There are two common approaches to two cursor input that present different physical problems for the user. The first approach is to use absolute input devices, like a puck and a pen on the surface of a tablet. The second is to use two relative input devices, like two mice.

When using two absolute devices, the user often runs into difficulty when their two hands interfere on the tablet surface. That is, it is not always possible to move one hand completely independently of the other since both hands occupy the same working area. In addition, some operations are asymmetric and may require that the user select a feature with the NDH that is on the side of the tablet nearer the DH and vice-versa. Nonetheless, there are advantages to using absolute input, especially that the user can develop a haptic memory of where to put their hands to perform certain operations. For example, to zoom in on a feature of a model, the user can move both hands to area that surrounds the feature and then move the hands to opposite corners of the tablet. The simple gesture of moving to the corners of the tablet after selecting points in the image is very easy to remember.

Alternatively, using relative input devices eases most of the difficulty with interference between the hands and reaching to inconvenient positions. However, according to informal questioning of our users, this approach also seems to require the that user be more aware of and have greater dexterity with their NDH.

We have not experimented with one absolute device and one relative device, but feel that might offer additional worthwhile compromises.

Context specification

In addition to controlling continuous DOFs, we have also found that two cursor input in the Sketch system enriches the set of operations that can be gesturally performed. This is important, since extending the single cursor version of Sketch with simple understandable gestures has been a problem. By using the NDH to define a context for the DH, we have found that we can extend Sketch in a very straightforward manner.

Work flow

Improving work flow is one of the principal reasons for exploring two cursor interaction. Our techniques demonstrate that a number of tasks can be performed in fewer steps using two cursors. However, we have not adequately determined how work flow may be interrupted when multiple different two cursor interaction techniques are incorporated into a single system. Our initial experience with integrating a number of techniques into Sketch and an Inventor viewer has been positive; however, we do not yet know if more conventional menu-oriented systems would adapt as well.

FUTURE WORK

We have begun to investigate how two-handed techniques can be incorporated both into research systems like Sketch, and into more widespread systems like Inventor. However, a more thorough investigation is justified.

Although we have found that users seem to find our techniques efficient and natural, more complete user studies might help to refine our interactions and determine the situations when they are most effective.

Further, we believe that continued exploration of two-handed input for 3D operations will result in additional effective interactions for standard operations, as well as novel interactions for new classes of 3D operations.

HARDWARE

This work was developed on a Hewlett Packard 735 workstation and an SGI Indigo 2 Extreme. One input device was a 12" x 12" Wacom Tablet that supported the simultaneous use of a 3-button stylus held in the DH, and a 4-button puck (although we only used 1 button) held in the NDH. A different configuration used a Contour serial mouse in addition to the workstation's mouse to specify two cursors.

ACKNOWLEDGMENTS

We thank the support of Andries van Dam and the Graphics Group as well as our sponsors: grants from NSF, NASA, Microsoft, Sun Microsystems and Taco; hardware from SGI, Hewlett-Packard, Sun.

REFERENCES

1. Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony DeRose. Toolglass and Magic Lenses: The see-through interface. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 73–80, August 1993.
2. W. Buxton and B. A. Myers. A study in two-handed input. *Human Factors in Computing Systems*, pages 321–326, 1986.
3. Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22(4):121–129, August 1988.
4. http://www.sgi.com/cosmo/.
5. Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *The Journal of Motor Behavior*, 19(4):486–517, 1987.
6. K. Herndon, R. Zeleznik, D. Robbins, D. Conner, S. Snibbe, and A. van Dam. Interactive shadows. In *Proceedings of UIST '92*, pages 1–6, November 1992.
7. Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassel. Passive real-world interface props for neurosurgical visualization. *Proceedings of CHI*, pages 452–458, 1994.
8. P. Kabbash, W. Buxton, and A. Sellen. Two-handed input in a compound task. *Proceedings of CHI*, pages 417–423, 1994.
9. Myron W. Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. Videoplace – an artificial reality. In *Proceedings of ACM CHI'85 Conference on Human Factors in Computing Systems*, pages 35–40, 1985.

10. G. Kurtenbach, G. W. Fitzmaurice, T, T. Baudel, and W. Buxton. The design and evaluation of a gui paradigm based on tablets, two-hands, and transparency. *Submitted for publication.*

11. D. P. Mapes and J. M. Moshell. A two-handed interface for object manipulation in virtual environments. *Presence*, 4(4):403–416, 1995.

12. Mark Mine. Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. Technical Report TR96-029, UNC Chapel Hill Computer Science Technical Report, 1996.

13. Chris Shaw and Mark Green. Thred: A two-handed design system. *Multimedia Systems Journal*, 3(6), November 1995.

14. P. Strauss and R. Carey. An object-oriented 3D graphics toolkit. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):341–349, July 1992.

15. http://vag.vrml.org/.

16. Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 163–170, August 1996.