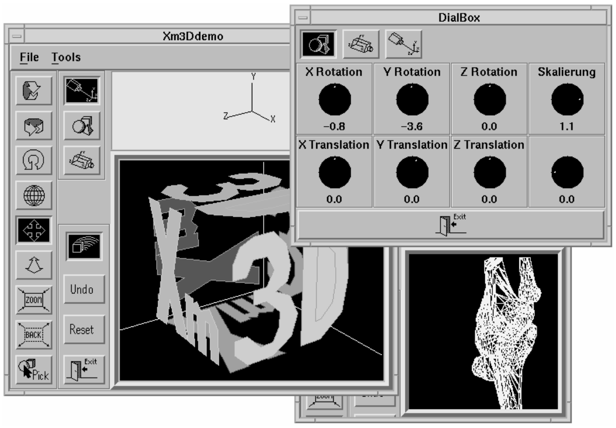
Gaphors: Combining Gestures and Metaphors in Order to Obtain a More Efficient 3D Interface

U. Bröckl-Fox and K. Hartenstein *



1 Abstract

Using metaphors mapping two dimensional mostly mouse driven interactions into the three dimensional object space becomes more and more state of the art for user interfaces manipulating three dimensional data.

This paper describes the usage of two dimensional, gesture driven metaphors augmenting the functionality of a widget set that manipulates three dimensional data. This offers a more intuitive (the three dimensional operation is entered by a more memorizable symbol, not an abstract menu) and faster way to manipulate three dimensional widgets.

A theoretical approach to model gesture driven three dimensional interactions and the expected results based on Fitts' law is presented. Then the practical implementation and embedding of the "gaphors" (i.e. Gesture driven metAPHOR) into the three dimensional widget set is described. The results based on end user experiments are stated.

2 Introduction

Interactive user interfaces manipulating three dimensional data always have been of high standards. With the increase in hardware performance accompanied by a drop in prices for workstations over the recent years, Graphical User Interfaces also became common for non graphical tasks. This put use and spreading of Graphical User Interfaces on a broader basis than it used to be at the times when Graphical Kernel Systems were developed. Furthermore this development led to the fact that the new Graphical User Interfaces have an enhanced functionality and are equipped with modern, mostly object-oriented application programmers interfaces. Examples for this wider functionality are event driven dialogs (the user, not the application rules the dialog flow), adaptability to different classes of users, a high number of different input media, dialog reversibility, considerable application independence of the dialog modules, easy to grasp metaphors and finally a standardization of user interfaces.

Therefore it seems reasonable to call for these functionalities for applications dealing with three dimensional data as well. But when designing a user interface that manipulates three dimensional data, several problems arise that are unknown for user interfaces for two dimensional data:

First, and above all, a standard workstation neither does have a three dimensional input medium that can be mapped directly into the three dimensional object space as it is possible with the two dimensional mouse for two dimensional object spaces, nor does it have a three dimensional output device. The standard graphical input device – the mouse – has to be projected into the three dimensional space. This led to the development of several mouse driven metaphors for three dimensional interfaces, see for example [CMS88], [Sut74], [Tho78], [WO90] especially to do three dimensional rotations using a mouse.

Second, if we accept the two dimensional input and output devices as given constraints when designing three dimensional interfaces, the projections

$$\textit{input}(\mathbb{R}^2) \rightarrow \textit{object-space}(\mathbb{R}^3) \rightarrow \textit{display}(\mathbb{R}^2)$$

cause a large variety of possible functions that can be performed using the mouse.

Third, among this variety of functions the user has to pick the appropriate function for the task to be accomplished. In Graphical User Interfaces this function selection is done by a mouse selection of a menu item or a button, therefore the mouse is used for two things: define *which* operation is to be performed (a rotation e.g.) and *give the data* (i.e. the angle of the rotation) for that operation.

This dual usage of the mouse – function selection and data specification – leads to an overhead of mouse movements: the user permanently moves back and forth between function selection (a menu bar e.g.) and the three dimensional display widget. Besides the mouse movement overhead the swapping causes a constant stress for the user: each time the three dimensional display is reentered with the mouse, a reorientation in the three dimensional window and a repositioning of the mouse has to be done.

Therefore it seems reasonable to avoid this overhead by introducing *gestures* that are entered by the mouse and that provide the function selection and the function's data at once. Once a gesture is entered, its function is determined from its shape using a pattern classification algorithm and then the function's data is extracted from the gesture's geometrical properties. An example of a gesture is shown in figure 1: the user enters an arrow in the three dimensional display, the operation, in this case a rotation, is performed with an angle that is determined from the length of the arrow entered.

*Both from: Institut für Betriebs- und Dialogsysteme; Universität Karlsruhe; Am Fasanengarten 5; Postfach 6980; 76128 Karlsruhe; Federal Republic of Germany; Fax: ++49 721 696989; Tel.: ++49 721 6084300;

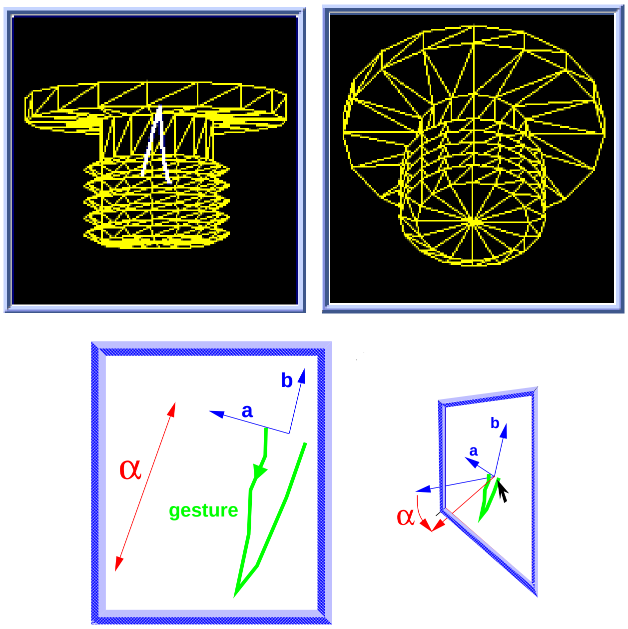


Figure 1: example gaphor

In the rest of the paper we will first discuss a theoretical approach to estimate the benefits of the gesture driven metaphors and then give an overview of the introduction of the gestures into the three dimensional widget set and finally state our theoretical and practical results.

3 Theoretical Approach

The first step in the development of a gesture driven interface for three dimensional widgets is to analyze the possible advantages of this interaction technique. The goal of this analysis is to show the expected decrease of time expensive redisplays especially when large quantities of geometry are manipulated. The approach is based on a model of a simplified 3D application which consists of one icon for function selection and a window for the graphical output.

3.1 Interaction Cycle

With the use of conventional interaction techniques the manipulation of three dimensional objects can be divided into five states (cf. figure 2). The *determination* of the function which leads to the desired manipulation the user has in mind, the *choice* of the determined function from a menu or iconbar, the *activation* of the corresponding icon, the *positioning* of the mouse cursor for data input and at last the *input* of the function data by moving the mouse in an appropriate way.

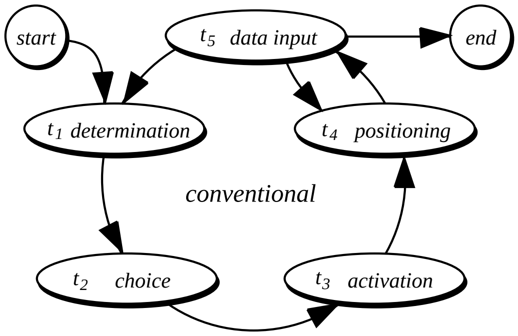


Figure 2: states of conventional interaction

The main features of a gesture are the representation of a function *and* its parameters *and* the direct drawing on the graphic output window. Therefore the usage of gestures for manipulating three dimensional objects shrinks three states of interaction (i.e choice, activation, and data input) into one state – the state of gesture input (cf. figure 3).

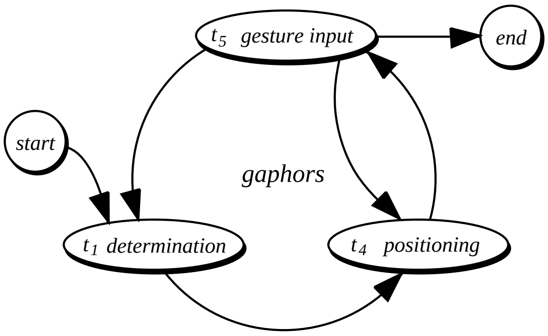


Figure 3: states of interaction with gaphors

3.2 Sample application

In order to compare the time spent on the interaction cycles of conventional interaction and gesture interaction the 2D extension of Fitts' law introduced in [BM92] is applied on a simple model of a 3D application as it is shown in figure 4. The model consists of a graphics output window of a variable extension and an icon that can stand for a menu item or an activation button.

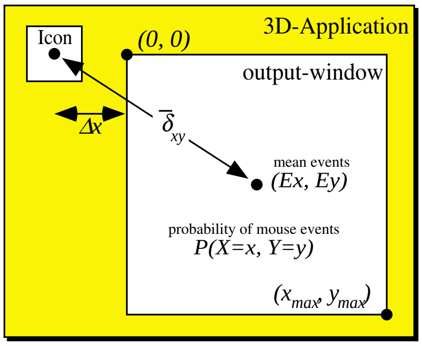


Figure 4: 3D sample application

The interaction times of the single states were determined as follows:

Determination: The time spent by the user to determine the required functions to transit from the current dialog state to the desired end state depends on the training of the user. A fully trained user, familiar to the application program due to daily usage, can manage this with less efforts than a user at times. Since this value is independent from the input method a value of one second is assumed. That approximately matches the mean of the values found for this task in a study of menu driven interfaces ([?]).

$$\boxed{t_1 \approx 1000\text{ms}}$$

Choice: Besides the educational level of the user the complexity of the application influences the execution time for this task. A large number of functions available – whose symbols or menus cannot be shown in parallel – has a negative influence on this time factor. From the user test data in [?] again a mean value of $\frac{1}{2}$ second is estimated.

$$\boxed{t_2 \approx 500\text{ms}}$$

Activation: To activate an application's function the mouse cursor is moved from the graphic display to the symbol representing the desired manipulating function and the function is activated by a single or double mouse click. To calculate the mean time spent for this first the mean distance

$$\overline{\delta}_{xy} = \sqrt{(E_x + \Delta_x)^2 + {E_y}^2}$$

from graphic display to the function's symbol (icon) has to be determined. From this the expectation value of the distribution of all mouse events is determined:

$$E_x = \sum_{j=0}^{y_{max}} \sum_{i=0}^{x_{max}} i \cdot P(X = x, Y = y)$$

$$E_y = \sum_{i=0}^{x_{max}} \sum_{j=0}^{y_{max}} j \cdot P(X = x, Y = y)$$

Here $P(X = x, Y = y)$ denotes the probability that a mouse event occurs at the coordinate (x, y) of the output window. [BM92] proposes the bivariate normal density as an appropriate model for this situation, whose mean lies in the mid of the output window:

$$\mathcal{V}(X) = N(\mu, \sigma^2), \quad \mathcal{V}(Y) = N(\lambda, \sigma^2)$$

This leads to the mean distance between the function's symbol and the output window:

$$\overline{\delta}_{xy} = \sqrt{(\mu + \Delta_x)^2 + \lambda^2}, \quad \mu = \frac{x_{max}}{2}, \quad \lambda = \frac{y_{max}}{2}$$

With a graphics output window with dimensions of arbitrary 700x700 pixels and a screen resolution of 92 dpi we get a mean distance of approximately 13cm. Applying Fitts' law ([BM92]) on this distance we get an activation time of about

$$t_3 = 230 + 166\frac{\text{ms}}{\text{cm}} \log_2(\frac{13\text{cm}}{1} + 1) \approx 862\text{ms}.$$

$$\boxed{t_3 \approx 862\text{ms}}$$

Positioning: Repositioning of the mouse cursor back to the graphic output window takes approximately the same time as for function activation since also a distance of $\overline{\delta}_{xy}$ has to be covered. Here assumed as as a simplification that the graphical object also has an extension of 1cm and lies in the middle of the window.

$$\boxed{t_4 \approx 862\text{ms}}$$

The mean drawing time of a gesture was calculated from 250 gesture examples as an amount of 600ms. With these assumptions the time comparison of the two interaction techniques is listed as follows:

state	conventional	gaphors
determination	1000 ms	1000 ms
choice	500 ms	-
activation	862 ms	-
positioning	862 ms	862 ms
data/gesture input	500 ms	600 ms
Σ	3724 ms	2462 ms

3.3 Theoretical Result

The condensation of three interaction states into one by using a gesture interface for the manipulation of three dimensional objects approximately leads to a time reduction of 32.7% opposed to conventional interaction techniques.

4 Experiments in Practice

To verify the results of the theoretical analysis of gesture interfaces for 3D manipulation, this interaction technique was implemented on the Xm3D platform [BKS92] which is a widget hierarchy based on Motif and PHIGS that was developed to manipulate three dimensional data.

4.1 Implemented gaphors

Figure 5 shows the gaphors implemented to manipulate three dimensional data such as translating, rotating and zooming. A few frequently used non-graphical operations like the *undo*-function were also integrated into the gesture driven user interface.

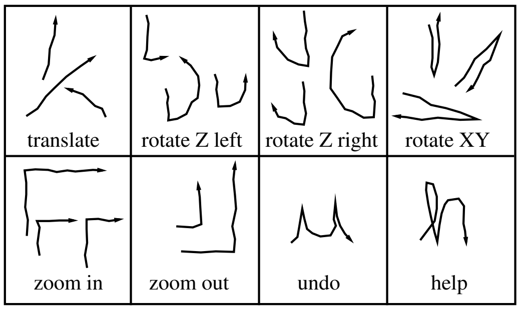


Figure 5: analyzed gaphors

The gesture of each gaphor is defined as an array of timestamped points

$$\mathcal{G} = [g_0, \ldots, g_P], \quad g_i = (x_i, y_i, t_i), \quad 0 \leq i \leq P$$

Throughout the following paragraphs the graphical, metaphorical and mathematical meaning of the most important gestures will be listed.

XY Translation:

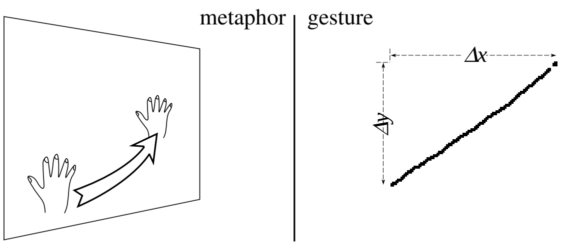


Figure 6: Translation

Metaphor: The translation metaphor corresponds to the translation of objects in real world. As widely known from two dimensional graphical user interfaces a three dimensional translation is accomplished by 1) snatching the object by pressing a mouse button, 2) translating the mouse cursor to the goal position and 3) finally let go the object by releasing the mouse button.

Gesture: The gesture to translate objects is a simple line whose start and endpoint mark the desired translation. If a translation into the direction of the view plane normal is desired, first a rotation of the whole scene has to be performed (most likely, but not limited to, by using a rotational gesture).

Parameters: Start point g_0 and end point g_p of the gesture.

Let $\mathcal{T}_X Y(\Delta_x, \Delta_y)$ be the function to translate on the XY-plane. Then entering the gaphor $\mathcal{G}_{XY-Trans}$ triggers the following translation (figure 6):

$$\mathcal{T}(g_P^x - g_0^x, g_P^y - g_0^y)$$

Z rotation:

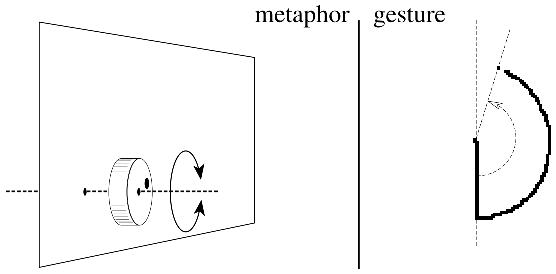


Figure 7: Z rotation

Metaphor: For rotations round the Z-axis the dial metaphor was implemented. The user thinks of a dial that is attached to a certain position on the output window. Turning this imaginary dial results into a rotation of the displayed geometries.

Gesture: The start point of the gesture defines the fulcrum of the dial metaphor. From this point a reference line and a circular arc that defines the angle of the rotation is drawn.

Parameters: Start point g_0, the fulcrum of the dial and the end point g_p of the gesture to compute the rotation's angle .

Let $\mathcal{R}_Z(x, y, \Delta\omega)$ be the function to rotate round the Z-axis [1] (= view plane normal) about an angle of $\Delta\omega$. Then entering the gaphor $\mathcal{G}_{Z-Rot}$ triggers the following rotation:

$$\mathcal{R}_Z\left(g_0^x, g_0^y, \arctan\left(\frac{g_P^y - g_0^y}{g_P^x - g_0^x}\right)\right)$$

XY rotation:

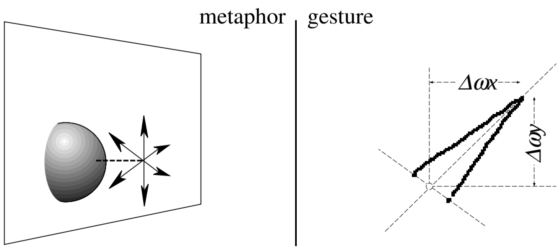


Figure 8: XY rotation

[1]To be more exact: the Z-axis that is translated x, y on the view plane.

Metaphor: The metaphor to rotate about the X- and Y-axis is the trackball metaphor. The user is imagining a sphere with a handle. The scene's whole geometry is rotated together with this imaginary sphere that is rotated by dragging the sphere's handle.

Gesture: The trackball metaphor is invoked by entering a V-shaped arrow whose orientation specifies the direction of the rotation. Its placement specifies the imaginary sphere's center and its length defines the angle of the rotation. The longer the arrow is, the bigger the angle will be.

Parameters: Start point g_0 und end point g_p and the extension $\Delta\omega_x$, $\Delta\omega_y$ of the gesture.

Let $\mathcal{R}_{XY}(x, y, \Delta\omega_x, \Delta\omega_y)$ be the function to rotate about the X- and Y-axis [2] and δ_x, δ_y denotes the width and length of the output window. Then entering the gaphor $\mathcal{G}_{XY-Rot}$ triggers the following rotation (cf. figure 4.1):

$$\mathcal{R}_{XY}(g_0^x + \frac{g_P^x - g_0^x}{2}, g_0^y + \frac{g_P^y - g_0^y}{2}, \Delta\omega_x, \Delta\omega_y)$$

$$\begin{aligned}
\Delta x_i &= g_{i+1}^x - g_i^x \\
\Delta y_i &= g_{i+1}^y - g_i^y \\
\theta_g(i) &= \arctan\left(\frac{\Delta x_i \Delta y_{i-1} - \Delta x_{i-1}\Delta y_i}{\Delta x_i \Delta y_{i-1} + \Delta y_i \Delta y_{i-1}}\right) \\
g_s &= g_j : \exists k : \theta_g(k) \\
&= \max\{\theta_g(i) : i \in 1 \ldots P-1\} \\
\Delta\omega_x &= \frac{\delta_x}{(g_s^x - \frac{g_P^x - g_0^x}{2}) \cdot 180} \\
\Delta\omega_y &= \frac{\delta_y}{(g_s^y - \frac{g_P^y - g_0^y}{2}) \cdot 180}
\end{aligned}$$

4.2 Classification

The simple gesture symbols of the chosen gaphors do not require a complicated classification system. Three realizations of the statistical pattern recognition, a simple neuronal network, an algorithmic and a fuzzy realization have been compared against each other with regard to their suitability for gesture recognition. The algorithmic realization by Dean Rubine [?] using a piecewise linear discriminator for classification showed best results in performance, recognition rate and ability. Figure 10 shows the recognition rate depending on the number of sample gestures used for a supervised learning, a rate of 92 % can be achieved.

[2]Again the X- and Y-axis that is translated x, y on the view plane is meant

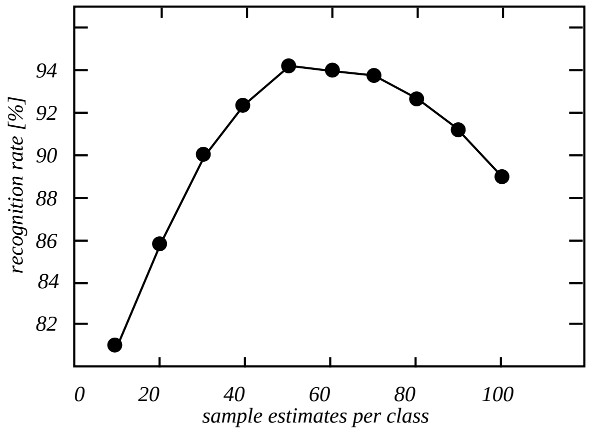


Figure 10: recognition rate

4.3 Practical Results

The mean interaction times of 79 comprehensive tests with the gesture interface point out the expected advantages and confirm the theoretical approaches. For large amounts of geometrical data, that is for long scene redrawing times ($t_{redraw} \in [0.5\text{sec}...15\text{sec}]$) gaphors offer an mean interaction time advantage of $\approx 30\%$ (cf. figure 11).

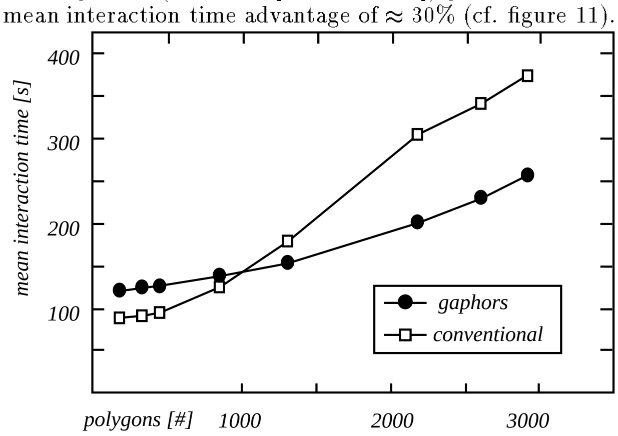


Figure 11: mean interaction times

Besides the mere interaction time advantage certainly the user acceptance of the gaphors is a better measure for the quality of the new interaction method. The Xm3D widgets offer both interaction techniques: mouse- and gesture-driven metaphors. We inserted a profiling mechanism to record the percentage of each method. The Xm3D widgets are still under development by several programmers and therefore developing, testing and presenting new features of the widgets causes a lot of three dimensional interaction. As depicted in Figure 12 the gaphors outbid mouse driven metaphors, especially if the gaphor's symbol is very simple.

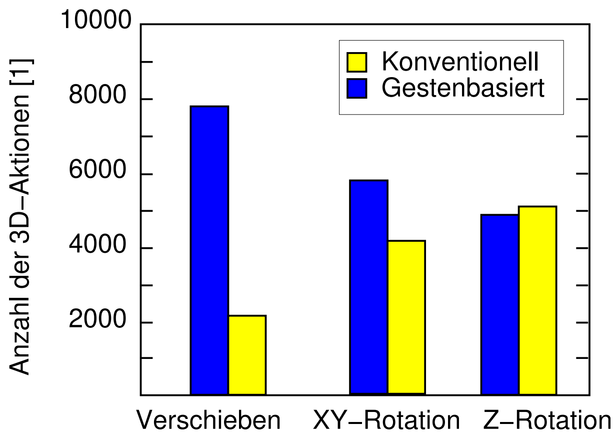


Figure 12: user acceptance

5 Conclusion

The theoretical analysis and the practical experiments concerning the introduction of mouse driven gestures and metaphors ("gaphors") into three dimensional user interfaces show that about $\frac{1}{3}$ of the interaction time to define the view of three dimensional scene can be saved. Therefore the gaphors became an integral part of our three dimensional widgets besides the traditional mouse driven metaphors.

Another field where the dual usage of the mouse (function selection and data specification) leads to an interaction overhead is the *multimedia* sector. We are currently integrating our gesture modules into an Graphical User Interface for an video sequence database [KKS+92].

Currently we are developing a vision-based, no more mouse-driven gesture interface, that promises to reduce the interaction time again, since the projection input($\mathbb{R}^2$) $\rightarrow$ object-space($\mathbb{R}^3$) is avoided.

References

[BKS92] U. Bröckl, A.J. Klingert, and A. Schmitt. Towards Standardized User Interfaces for Three-Dimensional Interaction: the Development of 3D Widgets Based on PEX and Motif. In *Proceedings of the Fourth Annual Conference of the European X User Group (EXUG)*, 1992.

[BM92] W. Buxton and S. MacKenzie. Extending Fitts' Law to Two-Dimensional Tasks. In *Proccedings of ACM CHI'92 Conference on Human Factors in Computing Systems*, pages 219–226. ACM-Press, May 1992.

[CMS88] M.-J. Chen, J. Mountford, and A. Sellen. A Study in Interactive 3-D Rotation Using 2-D Control Devices. *ACM Transactions on Computer Graphics*, pages 121–129, August 1988.

[KKS+92] W. Krueger, C. Knörzer, H. Stark, R. Kellerhals, and A. Schmitt. Bild- und Dateninformationssystem mit integriertem Computeranimationswerkzeug. Technical report, Forschungverbund Medientechnik Suedwest, 1992.

[Rie92] M. Riedel. Softwareergonomische Evaluation von Dialogbausteinen zur Verarbeitung dreidimensionaler Objektmengen, 1992.

[Rub91] Dean Rubine. Integrating Gesture Recognition and Direct Manipulation. In *Proceedings* *of the Summer '91 USENIX Technical Conference*, 1991.

[Sut74] I. Sutherland. Three-Dimensional Data Input by Tablet. *Proceedings of the IEEE*, 62(4):453–461, April 1974.

[Tho78] R. Thornton. Interactive Modeling in Three Dimensions through Two-Dimensional Windows. In *Third International Conference on Computers in Engineering and Building Design*, pages 204–211, Brighton, March 1978. IPC Science and Technology Press.

[WO90] C. Ware and S. Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. In *ACM Transactions On Computer Graphics*, pages 175–183, Oct. 1990.