February 1988

Jack: A Toolkit for Manipulating Articulated Figures

Cary B. Phillips
University of Pennsylvania

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-28.

Jack: A Toolkit for Manipulating Articulated Figures

Abstract

The problem of positioning and manipulating three dimensional articulated figures is often handled by ad hoc techniques which are cumbersome to use. In this paper, we describe a system which provides a consistent and flexible user interface to a complex representation for articulated figures in a 3D environment. Jack is a toolkit of routines for displaying and manipulating complex geometric figures, and it provides a method of interactively manipulating arbitrary homogeneous transformations with a mouse. These transformations may specify the position and orientation of figures within a scene or the joint transformations within the figures themselves. Jack combines this method of 3D input with a flexible and informative screen management facility to provide a user-friendly interface for manipulating three dimensional objects.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-28.

JACK:
A TOOLKIT FOR MANIPULATING ARTICULATED FIGURES

Cary B. Phillips
Norman I. Badler

MS-CIS-88-28
GRAPHICS LAB 20

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

April 1988

Acknowledgements: This research is partially supported by Lockheed Engineering and Management Services, the Pennsylvania Benjamin Franklin NASA Grant NAG-2-426 Partnership, NSF CER Grant MCS-82-19196, NSF Grants IST-86-12984 and DMC-85-16114, and ARO Grants DAA29-84-9-0027, DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory.

Jack:
A Toolkit for Manipulating Articulated Figures

Cary B. Phillips
Norman I. Badler

Computer Graphics Research Laboratory
Department of Computer and Information Sciences
University of Pennsylvania
Philadelphia, Pennsylvania 19104-0389

February 29, 1988

Abstract

The problem of positioning and manipulating three dimensional articulated figures is often handled by ad hoc techniques which are cumbersome to use. In this paper, we describe a system which provides a consistent and flexible user interface to a complex representation for articulated figures in a 3D environment. **Jack** is a toolkit of routines for displaying and manipulating complex geometric figures, and it provides a method of interactively manipulating arbitrary homogeneous transformations with a mouse. These transformations may specify the position and orientation of figures within a scene or the joint transformations within the figures themselves. **Jack** combines this method of 3D input with a flexible and informative screen management facility to provide a user-friendly interface for manipulating three dimensional objects.

1 Introduction

Many animation and simulation systems perform sophisticated operations on geometric figures in a particular configuration, but they leave the actual positioning of the figures to ad hoc techniques such as keyboard input of numerical joint angles or valuator input from a mouse. This requires the user to remember complex information about the coordinate frames with which objects are constructed and the axes around which the joints revolve. All too often, such

systems provide minimal capabilities for describing basic 3D position and orientation, forcing the user to adapt to a primitive set of commands.

Much work has shown the value of kinesthetically appropriate feedback[7,5,16]. Systems which do provide direct manipulation with multi-dimensional input devices often do not provide adequate feedback on how the motion of the input device produces world space transformations. Typically, several dozen degrees of freedom must be manipulated. For example, the human body model used by TEMPUS[2] has 18 joints and 48 degrees of freedom. Without adequate means of handling this complexity, systems can easily become cumbersome to use and the problem of figure positioning becomes a tremendous burden for the user.

Jack is a system being developed at the University of Pennsylvania which provides a flexible and general user interface for manipulating complex articulated structures, particularly human figures in a 3D working environment. **Jack** is not a complete system in itself, but a toolkit of routines and operators for displaying and manipulating geometric figures. **Jack** provides input and control for applications involving lighting and image rendering, anthropometric modeling, dynamics analysis, and keyframe and constraint-based animation and simulation.

Jack incorporates a simple but powerful mechanism for manipulating homogeneous transformations. These transformations describe the position and orientation of the figures and the displacement of joints, as well as other significant points in space or on geometric objects. **Jack** combines a flexible object representation with a visually informative screen management facility to yield a user-friendly working environment.

In this paper we describe the goals and philosophies behind the design of the **Jack** interface, as well some experiences with it. In Section 2, we discuss the problem of figure positioning and describe some desirable features of positioning systems. In Section 3, we discuss previous work in the area of figure positioning and three dimensional input techniques. In Section 4, we discuss the **Jack** interface and object representation. In Section 5, we describe the techniques which **Jack** uses to manipulate articulated figures. Finally, we summarize the approaches which **Jack** has taken to the various aspects of the problem of figure positioning.

2 Figure Positioning

Much of the recent research effort in animation has been aimed at developing new techniques for describing and generating motion, with the hope that this will provide the user with better control over the figures in a scene. These new techniques show great promise, but there has been little effort towards improving the fundamental interaction between the user and the objects. Work which has been done in this area has dealt primarily with arranging rigid bodies in space and has not properly considered the problem of manipulating complex

articulated structures.

The task of manipulating articulated figures is very common, particularly in animation. For keyframe animation systems, the importance of static positioning is obvious. However, tools for static positioning have usefulness outside of the domain of scene composition and keyframe animation. For example, systems which generate motion sequences from dynamic simulation[1,10,19] must still provide a mechanism for describing initial configurations and providing force and velocity information. Any such system will greatly benefit from a well-designed user interface.

Bier taxonomizes the problem of scene composition into interactivity, anchor richness, end condition richness, and smoothness[5]. Interactivity describes the interface with the user: keyboard, dials, mouse, etc. Anchors refer to the axes and planes with respect to which transformations may be specified. The end conditions involve the *values* of the transformations, i.e. angles and distances. Smoothness pertains to the technique of displaying the interaction.

Figure positioning has traditionally been difficult because systems failed in each of these categories. The failings of most interfaces can be classified as follows:

- Improper visual feedback of the current state of the figures
- Improper visual feedback of the results of transformations
- Inability to properly anticipate the effect of input
- Inability to easily describe transformations with respect to arbitrary reference frames

The ability to get immediate feedback on how the parameters of a figure affect its appearance has an enormous impact on the usability of a system. In animation, most figure positioning and geometric operations are performed on a trial and error basis, and the user is not always interested in precise measurements and exact positions. The ability to easily move a figure around in real time gives the user a better idea of what position the figure should assume.

The importance of direct manipulation for figure positioning is clearly obvious. The use of keyboard input discourages experimentation because of the time it takes to enter the values and see the results, as well as the difficulty in interpreting the results and determining if they caused the desired effect. A system which requires the user to enter numerical values for explicit joint angles or translational axes and distances can never hope to become an intuitive and flexible user tool. Text-driven input techniques are useful in scripting and in automated computations, but for an interactive system, the capability of moving figures directly is of crucial importance.

The most critical requirement of any interactive system is that it provide the user with the capability of easily and efficiently accomplishing a set of desired

goals. Shneiderman has said that "when an interactive system is well designed, it almost disappears, enabling the user to concentrate on his work or pleasure[17]". In the case of three dimensional design and animation systems, this translates to the ability to virtually reach into the environment and move objects and figures around at will.

Based on this desire, we observe the following characteristics of a useful object manipulation system:

- Changing the view should be so effortless it is almost transparent to the user. This principal comes from the fact that when a person is presented with an object to observe, he will pick it up, turn it around and look at it from several directions. The user of an interactive design system should be able to do likewise.

- The display should give intuitive real-time feedback on the movement. The screen should be visually informative, and the user should be able to drag objects to see the effect of different positions. Recent advances in graphics workstations have made this goal easier to achieve. This makes the process of trial and error much more effective.

- The user should be able to easily predict what motion of the input device will yield the desired object motion. Many systems have failed to provide usable manipulation tools because world space transformations are encoded in the space of the input device, making the results of the input difficult to anticipate.

- Many types of movement (global/local translation/rotation) should be available at the user's fingertips. The user should be able to experiment freely and quickly with position and orientation, and the response time should not be hampered by having to repeatedly pick axes or reference items from a menu or keyboard.

- Anchors and end conditions should be rich. The user should be able to specify exact distances and angles as well as approximate positions.

3 Previous Work

Recent work has attempted to relieve the burden of figure positioning by automating the task completely. Many exciting and promising techniques have been developed, based on inverse kinematics[11,9], dynamics[19,1,9], and constraint based optimization[3,4,20]. While these are very valuable techniques for generating realistic motion and precise positions, it can be difficult to manipulate a figure exclusively in this way. Such systems should be developed in conjunction with techniques for direct manipulation.

In the case of constraint satisfaction this is particularly true, since the constraints themselves are usually specified relative to positions and orientations of points on the objects or in world space. Without a simple mechanism for positioning the constraints, the user is faced with the same old problem. This problem can easily turn a potentially powerful animation system into a frustration for the user. This underscores the importance of a simple and intuitive way of moving objects and points around in a three dimensional environment.

3.0.1 3D Input Devices

Since the task of positioning articulated figures is inherently three dimensional, some attempts have been made at using three dimensional input devices. In particular, [3] and [16] describe experiments using a 3SPACE Digitizer[1], a magnetic device which senses the position and orientation of a hand-held wand. The experiments described in [3] conclude that the multiple degrees of freedom of the input device were difficult to control simultaneously, and that it became easier to control the device with some of the degrees of freedom disabled. This, along with the lack of proper visual feedback, has limited the usefulness of this system.

3.0.2 3D Input From 2D Devices

Several techniques have been developed for describing three dimensional transformations with a two dimensional input device, such as a mouse. Nielson describes techniques for mapping the motion a two dimensional mouse cursor to 3 dimensional translation and rotation, based on the orientation of the projection of a world space coordinate triad onto the screen [12]. Bier describes transformations formed by reference coordinate frames called *skitters* and *jacks*[5]. The user first positions the skitters and jacks in space and then uses their position and orientation to specify certain transformations. The skitters and jacks may be positioned directly on faces or along edges, or freely in space. Their position and orientation are manipulated by a set of dials.

4 The Jack Interface

Jack exhibits may desirable features of a three dimensional manipulation system. It can be viewed as a database manager which creates, displays, and manipulates articulated figures, and provides input in terms of geometric configurations to routines which perform such things as image rendering, dynamic simulation, and animation. **Jack** maintains multiple windows, and each window is a view into a geometric environment. Windows may view completely

[1]3SPACE is a trademark of Polhemus Navigational Sciences Division, McDonnell Douglas Electronics Company

different geometric environments, or they may provide different views of the same environment. **Jack** has a single "movement" operator, which is used to move anything which may be described by a homogeneous transformation. This uniformity helps the user cope with the complexity of the various operations.

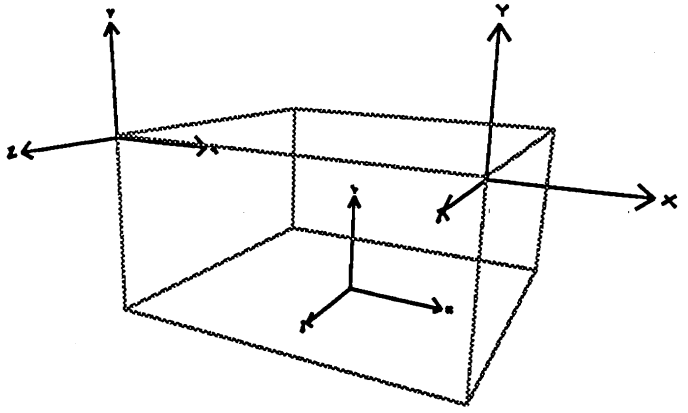
4.1 Articulated Figures

Jack is a primarily a user interface which controls the interaction with articulated figures represented in a system called **peabody**. **Peabody** represents figures composed of rigid segments connected by joints, also under the influence of constraints. Joints connect segments through "attachment points" called *sites*. A site is a local coordinate frame specified with respect to the base coordinate frame of the segment to which it belongs. Joints connect sites belonging to different segments within the same figure. Constraints are pseudo-joints between arbitrary sites in the environment. The segment is the basic geometric primitive. The state variables of each segment represent its mass and moment of inertia, as well as its geometry, stored as a boundary representation.

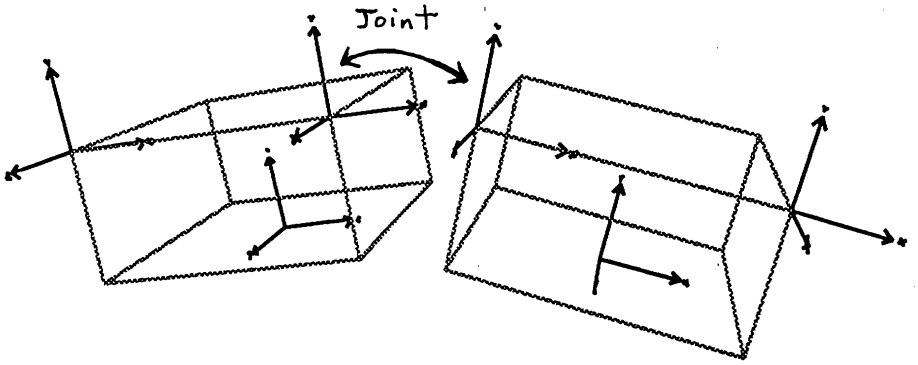
The user treats figures as arbitrary collections of segments connected by joints, without necessarily imposing a hierarchy upon them. **Peabody** maintains information about the global position and orientation of each segment in the environment. This global information is computed internally from the site and joint transformations. The user is encouraged to think of the geometric objects as an arbitrary graph structure of segments connected by joints. The segments are the nodes of the environment graph, and the joints are the edges. Figures are maximal subgraphs spanned only by internal joints.

When **peabody** computes the position and orientation of each segment, it first computes a spanning tree of the environment. Then it traverses the tree to compute the state information. This tree need only be recomputed when a new joint or segment is created or deleted, i.e. when the topology of the environment graph is altered.

Since this tree is computed internally, the user does not have to think of a figure as a strict hierarchy with a specific root. This is a simple but very important concept. Most figures with which we are concerned, particularly human bodies, are in fact hierarchical. However, it is advantageous to think of them as general graph structures which may be connected to other parts of the environment in arbitrary ways. These connections become important when assigning constraints, adjusting joints and moving figures, which we describe below. In **Jack**, figures may be easily rerooted interactively and can be rooted to any site on the figure. This makes it very easy to do such things as rotating a figure around a hand or pivoting around a foot. The figures may also be connected *to* other figures in the environment, not just the world frame. This arbitrary connect scheme makes it easy to specify transformations with respect to arbitrary frames.



(a) A segment with multiple sites



(b) A joint between two sites

Figure 1: The Peabody Data Structure

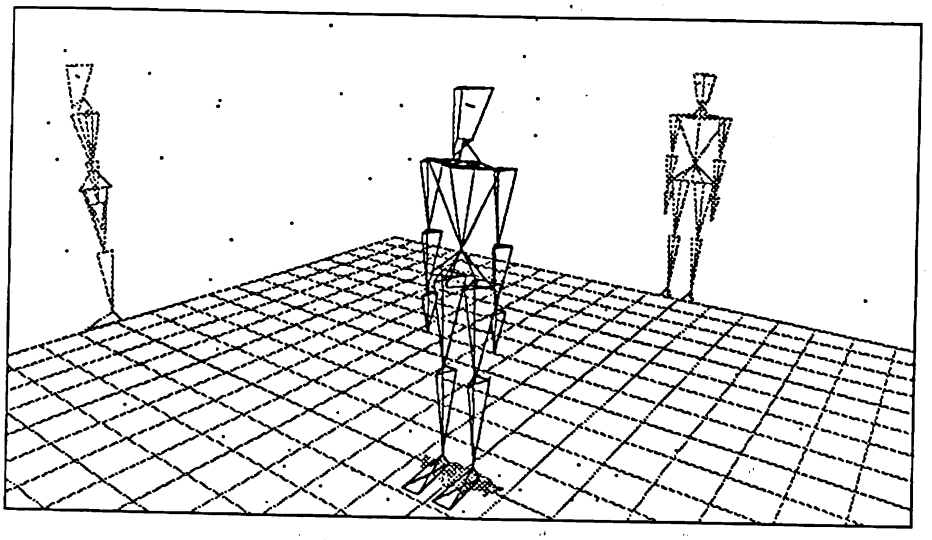


Figure 2: A **peabody** figure

4.2 The User Interface

Jack displays the screen in a visually informative way by drawing a ground plane grid, which gives a perception of the orientation of the world coordinate system. It draws the orthogonal projections of the figures in the scene on each of the coordinate axis planes. The projections are drawn in a darker color than the figures themselves, so they do not heavily distract from the scene. This gives an easily interpreted visualization of the arrangement of the figures. These projections are like shadows from infinitely distant light sources. Since all three projections are closely placed on the screen, the user can quickly reference the orientation and relative placement of neighboring objects in the scene.

All aspects of the display are optional and may be disabled. The user may choose to display the vertices, edges, faces, or sites associated with each segment on an individual basis. The sites are displayed as labeled coordinate axes. The orthogonal projections of the individual segments may be disabled as well. This allows the display to be easily tailored to suit a particular application, since the all forms of display may not be appropriate for all tasks at hand.

Jack is a menu driven system, but most of the commands in the menus have several options which can be invoked through keystrokes as the command is being executed. **Jack** controls almost all of the interaction through the mouse, which has three buttons, and sits on an optical pad beside the display monitor. In general, pressing and releasing the mouse buttons are different events, and many functions require the user to hold down a mouse button while moving the

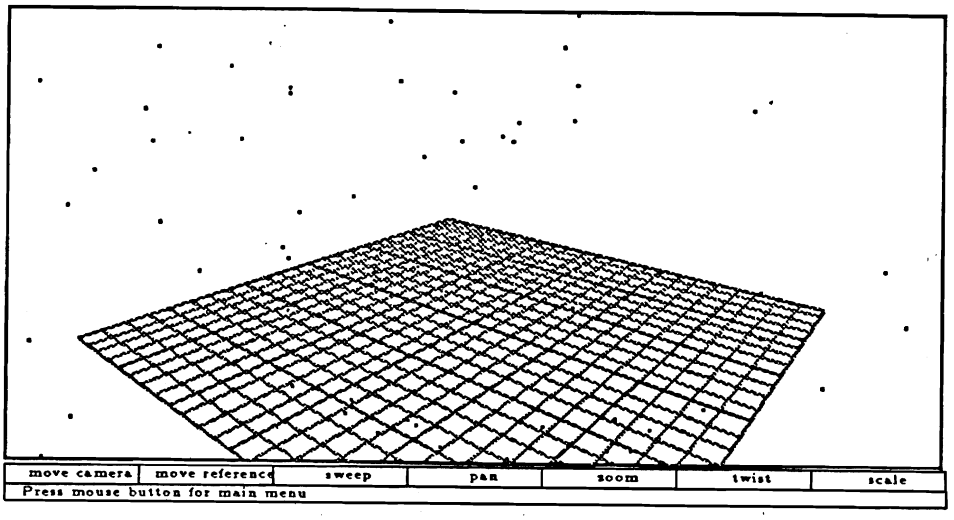


Figure 3: The **Jack** screen

mouse around on its pad.

Jack provides a simple mechanism for picking figures by pointing at them with the mouse. The user may pick segments, joints, sites, or entire figures, as well as individual vertices, edges, and faces of the segments. The picking mechanism waits for a mouse button to be pressed, then generates a "pick list" of items which currently lie under the mouse cursor. The user may select an item from the pick list by cycling through the pick list by clicking a second mouse button while still holding down the original button. When the original button is released, the item is selected.

All commands are available through the menus, but they can also be entered from the keyboard, either directly or in scripts. The scripts may be either parametrized or unparametrized. Unparametrized scripts operate on preselected figures. Parametrized scripts select the necessary figures as they are need by the operations.

The execution philosophy of **Jack** is similar to the EMACS text editor[18]. Command sequences may be bound to convenient keystrokes on a user-defined basis, and the user can develop sequences of commands interactively and bind them to keys. This provides the user with a powerful way of tailoring the working environment to suit a particular task.

5 Manipulating Objects

Many operations require positioning a figure or reference point. This movement is performed by moving the mouse, and the resulting position is displayed on the screen as the movement takes place. The movement operator has several components, which allow the user to either translate or rotate the transform with respect to various axes. These axes are determined from the state of the three mouse buttons and the shift and control keys, and may change as the command executes by releasing and pressing the various keys. These keys are used to specify whether the type of transformation, i.e. translation or rotation, and the reference axes and the reference frame. The individual mouse buttons select the appropriate axes. The control key specifies whether the transformation is rotation or translation. The shift key allows the user to change the reference frame. When the desired position is achieved, the movement is terminated by hitting the escape key.

The movement operator operates on two transformations: the *reference* transform, which is a global, and a *relative* transform, which is specified with respect to the reference. The reference transformation remains fixed; the relative transformation is continuously updated as the movement takes place. Some type of "action" is usually performed as this is done, such as updating the global position and orientation of figures and joints in the scene. This basic operator is used in may situations throughout **Jack** to move various things, such as figures and joints, as well as individual vertices, edges, and faces of geometric primitives.

5.1 The Mouse Line

Jack describes all motion with respect to the ray in the world coordinates which is cast through the location on the screen where the mouse cursor lies. This line in space is referred to as the *mouse line,* and it can be easily computed by an inversion of the viewing transformation. **Jack** restricts movement to lie in certain user-selected planes described below, and it determines world coordinate points by intersecting the mouse line with these planes.

5.2 Translation

Under normal operation, translations along the x, y, and z axes of an object's local coordinate frame are encoded in the left, middle, and right mouse buttons, so that pressing down any mouse button enables translation along that axis. During the movement operation, when the user presses a button, translation is enabled along that axis. When the user presses two mouse buttons, translation is enabled along those two axes, i.e. in the plane spanned by those axes. Since the mouse is a two-dimensional device, it is not possible to translate along three axes simultaneously, so pressing three buttons at once has no effect.

When the buttons go down, a vector is drawn in world coordinates describing the translational axes. As the user moves the mouse pad around, the object moves in world coordinates so that its location lies under the mouse cursor. As the figure moves, its global and local coordinates are displayed on the screen so that the exact position is available as well.

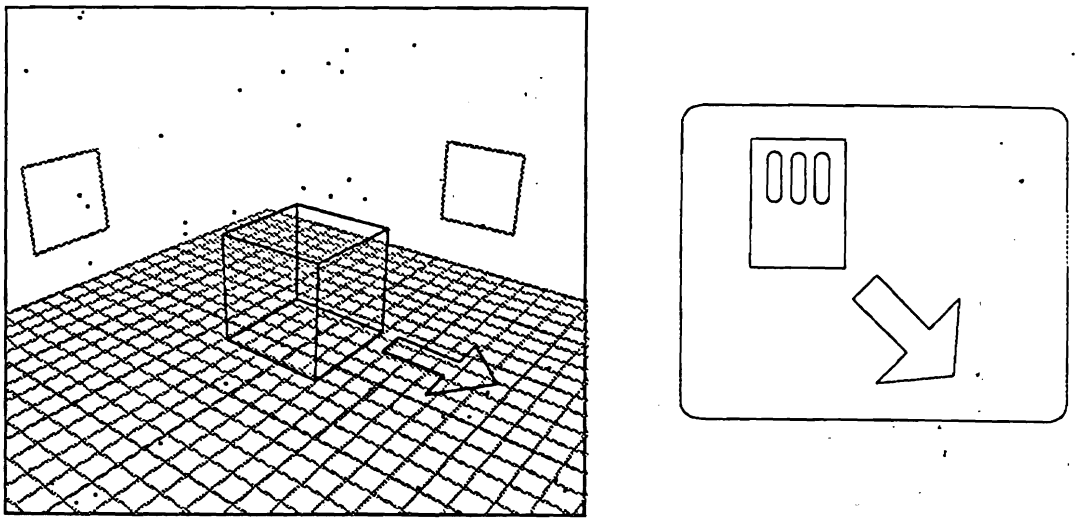


Figure 4: Translation

Translation along a single axis may be achieved by pressing only one button. The mouse may move anywhere on the screen, but the translation is restricted to a particular axis is space, which maps to a line on the screen. Therefore it is not possible to move the location of the figure along the axis such that it appears under the mouse cursor. In this case, **Jack** determines the position of the figure from the point along the translational axis which is nearest to the mouse line. **Jack** repeatedly repositions the mouse cursor so it lies along the line of translation.

5.3 Rotation

Rotation is accomplished in a similarly intuitive way, by requiring the user to move the mouse around in circles on its pad. The three mouse buttons are encoded as rotation around the x, y, and z axes. When the user presses down on a button, a "wheel" is displayed at the origin of the figure describing the rotational axis. This wheel lies in the plane in which the rotation is to take place, with the origin of the wheel at the rotational axis. Then a vector is

drawn from the current mouse line intersection with this plane, and as the user moves the mouse around in this plane, the figure stays fixed with respect to this reference vector. Therefore, if the user moves the cursor around in circles on the screen centered at the origin of the object, the object rotates around the three dimensional axis.

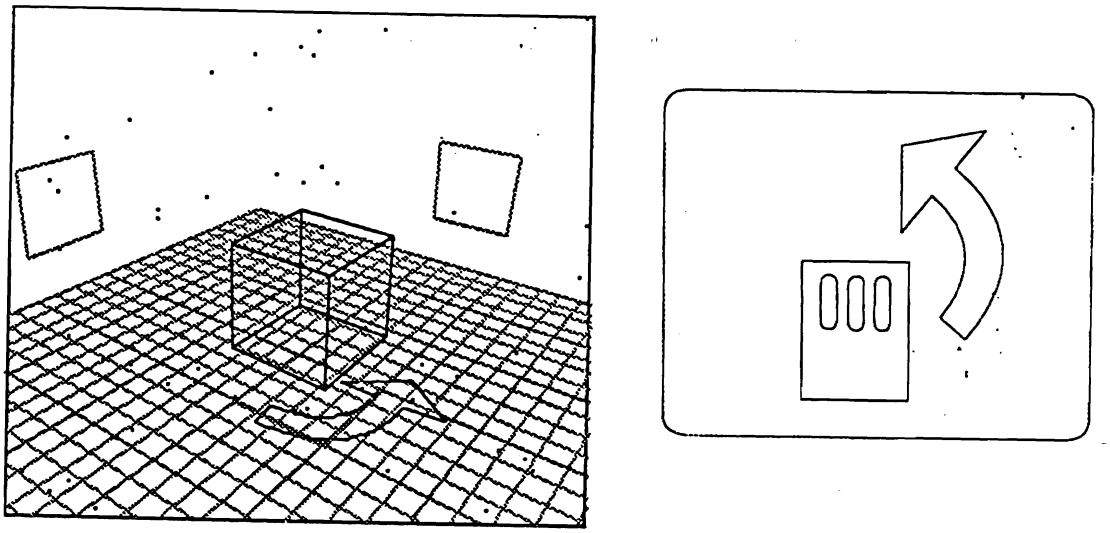


Figure 5: Rotation

5.4 Joints

These moving operations are helpful for positioning figures in world space, but usually an articulated figure is composed of a number of joints, having specific degrees of freedom. **Peabody** provides a mechanism for associating arbitrary rotational and translational degrees of freedom with a joint. In this case, the transformation at the joint is restricted to these axes. A simple and common example is the euler angles, which are defined as a rotation about the z axis, followed by a rotation around the y axis, followed by rotation about z.

In the case of transforming a joint which has specific degrees of freedom, the same type of movement is employed, but it is simply encoded in the mouse buttons in a different way. A properly defined joint will never have more than six degrees of freedom. The first three axes are encoded in the left, middle, and right buttons, respectively. The second three are activated by holding down the shift key. It is highly unusual to have a single joint with more than three degrees of freedom. Such a joint could more effectively be represented by a simple homogeneous transformation.

This mechanism has left unspecified how the joint displacement affects the position of the figure. When a joint is adjusted, one segment remains fixed with respect to the world frame and the other moves. In a strict hierarchy, this is a simple matter since there is a well defined "proximal" and "distal" segment. But **peabody** represents figures as a general graph, so it is ambiguous which segment remains fixed and which moves. In this case, **peabody** takes the proximal and distal segments from the underlying spanning tree of the environment, so that the segment which remains fixed is the one which is on the side of the figure which is rooted to the world. Since it is very easy to re-root a figure, it is a simple matter to arrange a transformation keeping any arbitrary segment fixed in space.

5.5 Reference Frame

The transformations described above are local to the coordinate frame being transformed, so that if an object is rotated and then translated, the translation will be with respect to the rotated axes. The transformations may be specified with respect to the parent transformation by holding down the shift key. In the case of a joint, the parent is the segment on the "other" side of the joint. In the case of a figure, the transformation it taken with respect to the site to which the figure is connected. If this site is attached to the world coordinate frame, the transformation is relative to the world. If a figure is rooted to another figure, the motion is relative to that figure.

Since the control and shift keys are close together on the keyboard, they are easy to press in conjunction. Thus, the user may specify global or local rotation or translation by holding down some combination of the control and shift key and the three mouse buttons. The motion continues in the fashion until the user presses the control key, ending the transformation. The transformation may also be aborted by typing `^C`.

key	*effect*
left mouse	x translation/rotation
middle mouse	y translation/rotation
right mouse	z translation/rotation
`CNTL`	rotation
`SHIFT`	transform w.r.t. parent

Figure 6: Key bindings during movement

5.6 Observations

A seeming drawback of this technique is that the user must remember the current orientation of the x, y, and z axes. However, the translational and rotational icons, along with the coordinate axis projections, give the user a simple means of determining this information quickly. Typically, the user has a particular direction in world coordinates in mind when he initiates a transformation. Since the icons are displayed as the keys are pressed and before the motion begins, the user can easily cycle through the available axes to select the appropriate set.

For instance, the user may want to "turn the figure around," which may technically involve rotating 180° about the y axis. The user can initiate a "move figure" operation, press the control key to specify rotation, and press the left, middle, and right mouse buttons in turn and begin moving the mouse when the appropriate axis is displayed. This technique frees the user from having to remember that the y axis is the appropriate axis. This avoids overloading the screen with information by making it available at the user's fingertips as he needs it.

Another seeming drawback of this technique is the inability to translate or rotate an object along an axis parallel to the line of sight, since in this configuration, small differences in the screen coordinates of the mouse may correspond to large distances in world coordinates. However, this is a transformation which the user should be discouraged from performing anyway. The first prerequisite for manipulating a figure, by computer or by hand, should be to position the figure in a convenient view.

5.7 End Conditions

These techniques provide a flexible way of manipulating transformations interactively, but it can be difficult to specify precise angles and distances in this way. This can be achieved in several ways. The most straightforward is to simply enter the coordinates from the keyboard, which can be done by typing `^K` during the motion, after which the user will be prompted for the desired angles or distances. This need only be done when a specific known distance or angle is needed.

While a motion is taking place, the user may "snap" the transformation to a particular face, edge, or node with a single keystroke. The snapping is controlled by three keys: `^F` for snapping to a face, `^E` for snapping to an edge, `^N` for snapping to a node. When one of these keys is pressed while moving a transform, the user is prompted to pick the appropriate item, and the transform is moved to tangency with the selected item. This tangency is accomplished by translating the transform to the desired node, edge, or face along the shortest possible distance, without rotation.

It is also possible to snap a transform to the direction of an edge or face. To

do this, the user first orients the transform so that it is approximately aligned with the desired edge or face and then types `^X^E` to snap to an edge, or `^X^F` to snap to a face. The snapping mechanism then takes the local coordinate axis of the transform which is closest to the desired orientation and adjusts the transform so that that axis is aligned the transformation with the edge or face.

6 Moving the View

Jack has a simple set of routines for manipulating the view, based on the operations of *sweeping, panning,* and *zooming.* The sweep operation sweeps the camera around horizontally and vertically on a virtual circular track, keeping it focused at the same reference point. The pan operation does the opposite, changing the orientation of the camera but keeping it at a fixed position. The zoom operation translates the camera along its line of sight. zoom can be performed simultaneously with either the horizontal or vertical sweep.

All three of these operations can be performed during the same viewing action, similar to the movement operator. By default, the left and middle mouse buttons controls the horizontal and vertical swing of the camera during a sweep operation. The right button controls zoom. Any two may be used in combination. A vertical motion of the mouse with the middle button down causes the camera to move up and down; a horizontal motion of the mouse with the left button down causes the camera to move side to side. When the shift key is pressed, the viewing operation changes to a pan, in which the camera says in the same location and rotates around on its axis. The left and middle mouse buttons control the horizontal and vertical motion just as with the sweep operation.

The view associated with each window provides a powerful visual tool. Internally, the view is represented by a site coordinate frame: the global position and orientation of the site determine the viewing transformation of the window. By default, a standard camera figure is created for each window with the view attached to it, but the view may be attached to any site in the environment. This provides a convenient means of positioning certain types of figures, such as light sources. The user may attach the view to a light, and then adjust the view, seeing in the window the objects on which the light shines.

This also provides a simple mechanism of performing view assessment. Our human figure representation has sites in each eye. The user may create two windows side by side, and attach the views in the window to the sites in the eyes. Then the user sees in the windows exactly what the figure would see given his current position in the environment. This is particularly useful during animation and simulation, when a motion sequence may be developed from a third point of view and then viewed from the perspective of the human figure.

7 Conclusion

Jack provides a flexible and easy-to-use interface for displaying and manipulating complex articulated figures. The windows, orthographic projections, and movement icons provide good visual feedback on the current state of the environment and the effect of the input. Since all three dimensional position and orientation information is provided by the same operator, there is a consisted interface for all aspects of the system. Since the movement operator generates homogeneous transformations based on the screen location of the mouse cursor, it is easy to anticipate the effect which movements of the input device will have on the objects. Since **peabody** is a very general mechanism for representing articulated structures, it is very easy to define attachment and reference points to specify transformations with respect to arbitrary frames.

References

[1] Armstrong, W. W., and Mark Green, "The Dynamics of Articulated Rigid Bodies for Purposes of Animation," *The Visual Computer 1,* No. 4, 1985.

[2] Badler, Norman I., Jonathan D. Korein, James U. Korein, Gerald Radack, Lynne S. Brotman, "Positioning and Animating Human Figures in a Task-Oriented Environment," *The Visual Computer 1,* No. 3, 1985.

[3] Badler, Norman I., Kamran Manoochehri, David Baraff, "Multi-Dimensional Input Techniques and Articulated Figure Positioning by Multiple Constraints," In *Proceedings of 1986 Workshop on 3D Interactive Computer Graphics,* (Chapel Hill, NC, October 23-26, 1986), ACM, New York, 1987.

[4] Badler, Norman I., Kamran Manoochehri, and Graham Walters, "Articulated Figure Positioning By Multiple Constraints", *Computer Graphics and Applications,* Vol. 7, No. 6, June, 1987.

[5] Bier, Eric "Skitters and Jacks: Interactive Positioning Tools," In *Proceedings of 1986 Workshop on 3D Interactive Computer Graphics,* (Chapel Hill, NC, October 23-26, 1986), ACM, New York, 1987.

[6] Bier, Eric, "Snap-Dragging," *Computer Graphics 20,* No. 3, 1986.

[7] Britton, E.G. J.S. Lipscomb, M.E. Pique, "Making Nested Rotations Convenient for the User." *Computer Graphics 12,* No. 3, 1978.

[8] Forrest, A.R., "User Interfaces for Three-Dimensional Geometric Modeling." In *Proceedings of 1986 Workshop on 3D Interactive Computer Graphics,* (Chapel Hill, NC, October 23-26, 1986), ACM, New York, 1987.

[9] Girard, Michael and A.A. Maciejewski, "Computational Modeling for the Computer Animation of Legged Figures," *Computer Graphics 19,* No. 3, 1985.

[10] Isaacs, Paul M. and and Michael F. Cohen, "Controlling dynamic simulation with kinematic constraints", *Computer Graphics 21*, No. 4, 1987.

[11] Korein, James U., "A Geometric Investigation of Reach," MIT Press, Cambridge, MA, 1985.

[12] Nielson, G.M. and Dan Olsen, Jr., "Direct Manipulation Techniques for Objects Using 2D Locator Devices," In *Proceedings of 1986 Workshop on 3D Interactive Computer Graphics,* (Chapel Hill, NC, October 23-26, 1986), ACM, New York, 1987.

[13] Phillips, Cary, "Programming With Jack," unpublished programmer's manual.

[14] Phillips, Cary, "Using Jack," unpublished user's manual.

[15] Pique, M.E. "Semantics of Interactive Rotations." In *Proceedings of 1986 Workshop on 3D Interactive Computer Graphics,* (Chapel Hill, NC, October 23-26, 1986), ACM, New York, 1987.

[16] Schmandt, Christopher, "Spatial Input/Display Correspondence In a Stereoscopic Computer Graphics Workstation," *Computer Graphics 17*, No. 3, 1983.

[17] Shneiderman, Ben, *Designing the User Interface,* Addison Wesley, 1987.

[18] Stallman, Richard, *Gnu Emacs Users Manual.*

[19] Wilhelms, Jane "Using Dynamics for the Animation of Articulated Bodies Such as Humans and Robots," *Proceedings of Graphics Interface '85,* Montreal, 1985.

[20] Witkin, Andrew, Kurt Fleischer, and Alan Barr, "Energy Constraints on Parametrized Models," *Computer Graphics 21*, No. 3, 1987.