SOFTWARE SYSTEMS FOR MODELING ARTICULATED FIGURES

Cary B. Phillips

Computer Graphics Research Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6389

Abstract

Research in computer animation and simulation of human task performance requires sophisticated geometric modeling and user interface tools. The software tools for a research environment should present the programmer with a powerful but flexible substrate of facilities for displaying and manipulating geometric objects, yet insure that future tools have a consistent and friendly user interface.

Jack is a system which provides a flexible and extensible programmer and user interface for displaying and manipulating complex geometric figures, particularly human figures in a 3D working environment. It is a basic software framework for high-performance Silicon Graphics IRIS workstations for modeling and manipulating geometric objects in a general but powerful way. It provides a consistent and user-friendly interface across various applications in computer animation and simulation of human task performance. Currently, Jack provides input and control for applications including lighting specification and image rendering, anthropometric modeling, figure positioning, inverse kinematics, dynamic simulation, and keyframe animation.

1 Introduction

The great promise of computer graphics is visualization, the ability to answer difficult problems and convey complex information through computer generated images. The problem for researchers in computer graphics is how to generate images which convey such useful information. Recent advances in computer hardware have revolutionized the capabilities of graphics simulation systems. Today's hardware is capable of displaying large numbers of graphics primitives in real time. The task now is to take full advantage of these new graphics capabilities in software modeling systems.

This charge applies especially to software for modeling geometric objects. The importance of visualization in geometric modeling is quite obvious, but the application of computer graphics goes beyond simply generating static synthetic images. A modeling systems should give it user the ability to *manipulate* the models, and the models should behave in a way which conveys information back to the user.

User interface toolkits such as X-windows provide good tools for designing interfaces to many types of software programs, but they do not provide adequate tools for constructing interfaces for manipulating geometric objects. Developers of software for computer animation and simulation need similar sorts of tools for assisting in the higher-level task of modeling, displaying, and manipulating complex geometric figures.

Jack is a system being developed at the University of Pennsylvania to support research in human task performance in the Computer Graphics Laboratory. Its goal is to provide a consistent and easy-to-use programmer utility for modeling, displaying, and manipulating complex articulated structures, and at the same time provide a consistent and convenient user interface across various applications. Jack runs on the 4D Series Silicon Graphics IRIS Workstations, and its intended user community consists primarily of engineers with a basic understanding of robotics and geometric modeling concepts.

Jack is very general in its ability to model articulated figures, but its primary purpose is to support

human factors analysis. The object modeling facilities in Jack are designed to handle the special difficulties of modeling and manipulating human figures in a 3D working environment. There are many sources of support for this project, each with its own emphasis and application:

- NASA Johnson Space Center and Lockheed Engineering and Management Services, Graphics Analysis Facility of the Man/Systems Division: primarily Space Shuttle and Space Station applications, with major interest in reach, fit, and view analyses, with active interest in strength models, zero-gravity dynamics simulation, and language-based task processing.
- NASA Ames Research Center: the A^3I project to simulate all aspects of a helicopter mission is the application, with Jack providing the pilot model and forming the basis for workload computations. The pilot's mission and tasks are provided by an external AI-based simulator.
- Army Research Office, the Human Engineering Laboratory at Aberdeen Proving Grounds: application to multi-operator vehicles, with a primary interest in evaluation of reach, comfort, strength, workload, and cooperative behavior.
- Pacific Northwest Laboratories, Battelle Memorial Institute: application to control a mobile robot mannequin used to test suit designs for permeability to chemical and biological agents, with a primary interest in animation control, safe path determination, collision avoidance, and motion feasibility.
- State of Pennsylvania Benjamin Franklin Partnership: technology development in Artificial Intelligence methods to aid human factors evaluation.
- National Science Foundation: representations and systems to assist in the interactive and automatic generation of natural, animated human motion.

This project greatly benefits from its home in a Computer and Information Science Department because computational tools and techniques are essential for such a broad spectrum of human performance problems.

This paper gives an overview of Jack. It explains many of the features built into Jack, including its object modeling facilities as well as its interaction mechanism for manipulating figures through direct manipulation and inverse kinematics.

2 The Software Engineering Aspect

In order for computer graphics to fulfill its great promise, software developers must be careful to craft their systems to be effective *tools*. The graphics images must be merely the means to the end and not a burden to support.

This is especially important in a research environment, where researchers need to experiment with new types of algorithms and techniques, yet produce software tools which will be usable by non-programmers. In such an environment, it is particularly important to develop flexible and extensible software tools, since it is not always possible to fully anticipate future demands of the software. In this type of environment, the modeling software cannot be a "black box" because it is frequently necessary to extend it or customize it in various ways.

Jack is designed to be easily extensible. It is a command driven system: the user typically executes commands by selecting items from pop-up menus, but commands may also be entered explicitly from the keyboard or read from command files. Jack is very modular in that each command or group of commands roughly corresponds to a specific utility. During development, certain unnecessary utilities may be omitted, drastically reducing the overhead of the development process.

Jack follows a well-defined mechanism for defining commands and controlling input from the user. This mechanism makes it very easy to customize Jack for specific applications. Many times such applications evolve into sophisticated standard utilities. The simplicity with which commands may be written and added to the menus make it easy to manage the software as it grows and encorporates new utilities.

3 The Peabody Object Representation

The heart and soul of geometric modeling software is the representation for geometric objects. This involves much more that just the shape of the objects. The models must represent information about how the objects *behave*, how they simulate the behavior

of the real-world objects they represent. The purpose of the graphics facilities in the software is simply to convey information about the models. If the underlying model is not rich in information, the graphics facilities will have little use.

Jack is primarily a user interface which controls the interaction with articulated figures represented by a system called peabody. The name *peabody* refers to both the internal data structures representing the geometric objects and to the external language for describing and storing them. Peabody objects are described in text files, and Jack can be viewed as a graphical editor for constructing and manipulating these objects. By analogy, this editor is to the geometric objects what a word processor is to English prose.

Peabody represents *figures* composed of rigid *segments* connected by *joints*, which may also be under the influence of certain *constraints*. The segment is the basic geometric primitive. The state variables of each segment represent its mass and moment of inertia, as well as its surface geometry, which is a boundary representation. Segments also represent material properties such as reflectance parameters and light emission values.

Joints connect segments through attachment frames called *sites*. A site is a local coordinate frame specified with respect to the base coordinate frame of the segment to which it belongs. Segments may have any number of sites. Joints connect sites belonging to different segments within the same figure. Constraints are pseudo-joints which express relationships between arbitrary sites in the environment.

3.1 Articulated Figures

One of the most important features of the peabody representation is its model for the articulation of the figures. Figures may be composed of any number of segments connected arbitrarily by joints. Since each segment may have any number of sites, and the sites may be located anywhere on the segment, the "skeleton" of the figure is easy to define even if the segment does not have a distinct proximal and distal end.

The joints may have arbitrary degrees of freedom, which peabody represents as zero to six rotational or translational components. A zero degree of freedom joint is like a bolt and is not manipulatable. Each degree of freedom may have upper and lower limits on it range of motion, and the limits are enforced during interaction. The degrees of freedom also represent stiffness and dampening information for dynamic simulation.

The user treats figures as arbitrary collections of segments connected by joints, without imposing a predefined hierarchy upon them. Jack encourages the user to think of the geometric objects as an arbitrary graph of segments connected by joints. It computes the global position and orientation of each segment internally by first computing a spanning tree of the environment. 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 predefined root. This simplifies the operation of "rerooting" a figure, either to attach a figure to another figure, or to change the point of attachment of a figure to the world coordinate frame. This scheme makes it easy to specify transformations with respect to arbitrary reference frames.

The arbitrary figure root is important for manipulating figures which must maintain contact which certain points in space. This is especially true of manipulating figures in a zero gravity environment, where figures may be attached through arbitrary foot or hand restraints. When a figure is attached to a hand restraint in zero gravity, a bend in the elbow results in the movement of the entire body, rather than a movement of the hand and arm, which remains fixed at the point of attachment.

3.2 Human Figure Models

Although Jack is primarily a tool for human factors analysis, it makes no formal distinction between human figures and other geometric objects. All objects are described by peabody, and the human figure models used are described by data files designed to model the human figure in the specific ways. This model is external to the software itself, which makes it possible to model figures with differing degrees of complexity. Different body models may be used in different circumstances. For example, engineers in the Graphics Analysis Facility at JSC have developed a model for the Extra-Vehicular Activity suit which has restricted ranges of motion.

In addition, the actual geometry of the individual segments is defined separately from the topology of the figure, which is defined in terms of the locations of the joint centers. This makes it possible to use different body geometries with the same underlying topologies. Jack currently has three basic body geometries. Each model consists of 31 segments with 29 joints. The

first model consists of 109 polygons and represents a very crude, almost stick-figure, approximation to the human body. An intermediate model consists of 408 polygons and resembles a robot-like figure. Finally, a highly complex model has been derived from laser scan data of human subjects. This model consists of 4571 polygons.

3.3 Anthropometry

The syntax of peabody language loosely resembles data structure definitions in a traditional programming language. The peabody language employs a powerful mechanism for parsing arithmetic expressions. These expressions may be used as part of the definition of the figures, so that figures may be parameterized.

The ability to parameterize figures allows Jack to easily model human figures of arbitrary anthropometric proportions. An auxiliary facility called SASS is a spreadsheet program which allows the user to create peabody human figure models of arbitrary anthropometric sizes, based either on percentiles from specific populations or from actual numerical values. It generates parameters for girth, joint limits, and centers of mass. Currently, SASS uses NASA trainee population data from the NASA Man-Systems Integration Manual, Chapter 3 (NASA-STD-3000).

4 The Jack Window System

Jack uses the Silicon Graphics IRIS window manager, **4Sight**, which run under the Unix operating system. This window manager allows the user of the workstation to create multiple windows and run different graphics programs simultaneously. Jack creates windows which provide views of geometric objects. These windows may be moved and reshaped just like the other window manager windows. This allows Jack to be used as a "tool." The user may easily shift back and forth between using Jack and using the underlying operating system.

Jack derives most of its input from a three-button mouse, with a little input required from the keyboard. It is a menu-driven system, and commands are generally executed by selecting items from the pop-up menus. Although Jack maintains unique names for all geometric objects, it is usually possible to refer to objects by pointing at them with the mouse. Most of the keyboard input is in the form of single keystrokes to invoke certain options. Very little typing is required, although it is possible to control Jack completely without the mouse if necessary. Jack avoid being too cryptic in its keystroke bindings by displaying information about the bindings whenever the user is in a position to need them.

The execution philosophy of Jack is to select a high level operation first, and then select the operands. The user executes commands from a menu, such as **move figure**, then he pick the appropriate object by pointing at it with the mouse. Finally, he specifies the *value* of the operation, i.e. a transformation, which is usually manipulated interactively. Most operations such as moving are terminated by hitting a special key, such as the escape key.

4.1 Jack Windows

The parameters of each Jack window are easily to tailor for specific applications and situations. By default, Jack displays the screen in a visually informative way by drawing a ground reference plane grid, giving a perception of the orientation of the world coordinate system. Jack draws orthogonal projections of the figures in the scene on each of the coordinate axis planes. These projections roughly resemble shadows from three infinite orthogonal light sources and serve as quick reference cues for the position and orientation of the figures. The projections are drawn in a darker color than the figures themselves, so they do not distract from the rest of the scene. 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. These projections may be easily disabled, and may also be enabled on a segment by segment basis.

Most aspects of the display are optional. The user may choose to display the vertices, edges, or faces of each segment. The edges are drawn in wireframe. The face may be shaded and z-buffered, illuminated by multiple light sources using the lighting model hardware of the IRIS workstation. With the IRIS hardware, there is no significant performance penalty involved in the shading, and the user is free select either shaded or wireframe display. The sites associated with each segment may also be displayed. Sites are drawn as a labeled *xyz* coordinate axis frame. This allows the display to be tailored to suit a particular application, since all forms of display may not be appropriate for all tasks.

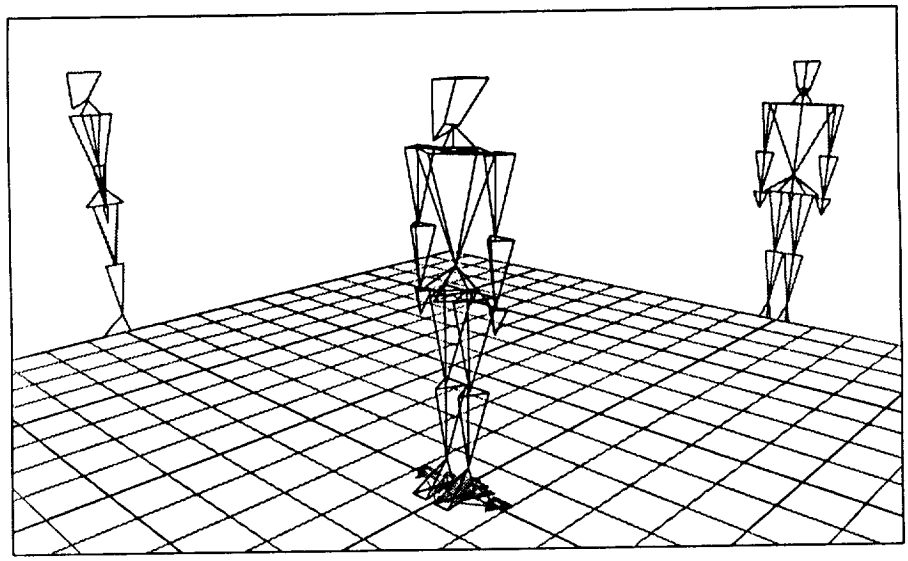

Figure 1: The Jack screen, with a human figure model

4.2 Viewing Facilities

It is especially important for modeling systems to provide good facilities for change the view. In the real world, when someone is presented with an object to observe, the natural reaction is to look at it from different directions, either by picking it up and moving it around, or if it is too large to handle, by walking around it and looking at it from all sides. Software which attempts to convey geometric information through images should provide a similar ability.

Jack has a flexible way of manipulating the view. The view in each Jack window is described by the global position and orientation of a specific site on a peabody figure. By default, a "camera" figure accompanies each window to represent the view. Moving the view in a window corresponds to moving the camera figure in a special way. This is especially beneficial in the Jack windowing environment, since the user may create different windows, each with a different camera. The camera figures may be displayed just like any other figure, so it is possible to see and manipulate in one window the camera of another window.

Jack allows the user to change the view by what it calls *sweeping* and *panning*. The sweeping operation moves the camera in circular arcs centered at a constant reference point, called the *view reference point*. The view may swing horizontally or vertically, or it may zoom in and out, all controlled by the mouse. This is beneficial for viewing a particular point in space from different directions. Panning is the opposite of sweeping: the location of the camera remains fixed while it pivots up or down. The view reference point changes as the camera turns. This is useful for looking side to side, but it is also an easy way to move the view reference point around in space.

The view in each Jack window may alternatively be "attached" to any site on any figure. This makes it possible to attach the view to the eyes of a human figure model and *see* in the window what the figure *sees*. This is particularly beneficial during the animation of the motion of a figure. The figure may be manipulated from a secondary point of view, and the animation may be played back both from the point of view of the figure or from a fixed point.

Another application of the viewing mechanism is positioning light sources. A special light source positioning facility temporarily attaches the view to the light and then allows the user to adjust the view. The user sees in the window where the light shines.

5 Object Manipulation Facilities

An important characteristic of truly manipulatable computer models of geometric figures is quick turnaround time between the user's decision to position the figure in a certain way and the time he actually accomplishes his goal. Most figure positioning tasks in an interactive 3D environment either require great precision, such as moving objects to tangency

or contact, or are very general, in which case rough positions are sufficient and precision is not an issue. Generally, the more detailed the positioning task, the more input may be required from the user. It is important for modeling software to be able to handle both cases well because the user should not be forced to enter complex input for a simple positioning task.

Jack provides several facilities for moving and manipulating objects. The user may use great precision when necessary, but may also quickly and intuitively move objects around in the workspace without the overhead of a complex positioning algorithm. The direct manipulation scheme is useful for position objects with gross movements. The inverse kinematics facility allows the user to position objects automatically to achieve multiple simultaneous goal positions.

5.1 Direct Manipulation

Jack uses a "view based" 3D direct manipulation scheme. All of the rotational and translational input is obtained from the mouse, but the movement of the mouse by the user is coupled to the current view of the object being manipulated, so that there is a direct and intuitive relationship between the direction the user moves the mouse and the direction the object moves.

For 3D translation, the user selects an axis of translation by holding down a button on three button mouse, but after the axis has been selected, the direction of movement of the mouse which cause movement of the object along that axis is determined by the line which that axis in space makes on the screen. The user may also translate objects in a plane by holding down two mouse buttons simultaneously, in which the movement of the object is constrained to lie in that plane and the location of the object is determined by the point in the plane which lies underneath the mouse cursor in the current view. The effect is a intuitive way of translating objects, since the object "follows" the mouse on the screen.

Rotations in 3D are accomplished with a rotation "wheel", which is a graphical icon describing the axis of rotation. The user selects the axis of rotation by holding down a single mouse button, and the wheel appears to demonstrate the selected axis. The rotation is accomplished by moving the mouse around the perimeter of the rotation wheel. The effect is also fairly intuitive, since the user moves the mouse in circles around the object to cause the object to rotate.

The rotation and translation mechanisms are used both for moving figures in the world coordinate frame and for manipulating the displacements of joints. Joints may have either rotational or prismatic components in their degrees of freedom, and the user may manipulate the joint using the direct manipulation facilities. If the joint has limits, the limits are obeyed during the interaction, and the joint is not allowed to violate the limit.

5.2 Inverse Kinematics

The direct manipulation facilities in Jack make it easy to position entire figures and manipulate individual joints by hand. But many positioning tasks involve manipulating many joints simultaneously until a certain condition is satisfied, such as tangency or point-to-point contact. Jack has a sophisticated inverse kinematics facility which uses a gradient descent algorithm to solve for a set of joint angles, within the defined joint limits, which satisfy a number of geometric "goals". The user-selectable parameters of the "reach" are the goal site, the end effector, and the set of joints to be manipulated during the reach. The objective function may encorporate a weighted combination of position and orientation. During the solution of a multiple goal reach, each goal may have a separate weighting factor, which specifies the relative importance of each goal if the goals are not all simultaneously reachable.

There are several variants of the reach algorithm. First, the *active* reach attempts to model the behavior of a real human subject performing a reach. It attempts to solve the reach with af user-specified chain of joints, but if the goal is not reachable, joints are added to the joint chain, working towards the body root, until the chain includes all joints between the end effector and the root.

Another variant of the reach algorithm is a *pointing* reach, which is useful for orienting the head and eyes of a human figure for looking at a particular point in space. The user input is similar to the ordinary reach, but the algorithm manipulated the joints so that the line of sight of the end effector is directed towards the goal.

5.3 Keyframe Animation

Jack has a sophisticated keyframe animation subsystem which allows the user to define *groups* and *actions*. Groups are sets of "things which change over time", typically joints and constraints. Actions are primitive sequences of changes to the values of the elements of a group. *Keyframes* are sets of values for

the elements of a group. A *scene* is a collection of possibly overlapping actions.

The animation facility may be used directly for keyframing known movements or interpolating between specific positions. It may alternatively be used as a means of collecting, storing, and playing back motions generated from external means, such as from external dynamic simulation software. Typically, the dynamic simulation produces output at specific time slices, which may be greater or less then the desired frame rate for playing back the motion sequence. The features of the animation system allow this to be easily controlled.

References

[1] 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.

[2] Grosso, Marc, Richard R. Quach, and Norman I. Badler, "Anthropometry for Computer Graphics Human Figures." Technical Report, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1989.

[3] Phillips, Cary B, and Norman I. Badler, "Jack: A Toolkit for Manipulating Articulated Figures" Proceedings of ACM/SIGGRAPH Syposium on User Interface Software, Banff, Alberta, Canada, 1988.

[4] Phillips, Cary, "Programming With Jack," Technical Report, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1989.

[5] Phillips, Cary, "Using Jack," Technical Report, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1989.

[6] Zhao, Jianmin and Norman I. Badler, "Real-time Inverse Kinematics with Joint Limits and Spatial Constraints" Technical Report MS-CIS-89-09, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia, PA, 1989.