12-1986

Modeling and Animating Human Figures in a CAD Environment

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

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-86-88.

Modeling and Animating Human Figures in a CAD Environment

Abstract

With the widespread acceptance of three-dimensional modeling techniques, high-speed hardware, and relatively low-cost computation, modeling and animating one or more human figures for the purposes of design assessment, human factors, task simulation, and human movement understanding has become feasible outside the animation production house environment. This tutorial will address the state-of-the-art in human figure geometric modeling, figure positioning, figure animation, and task simulation.

Disciplines

Computer Engineering | Computer Sciences

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-86-88.

MODELING AND ANIMATING HUMAN FIGURES IN A CAD ENVIRONMENT

Dr. Norman Badler

MS-CIS-86-88
GRAPHICS LAB 14

Department Of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

December 1986

Acknowledgements: The research reported here from the University of Pennsylvania is partially supported NASA Contract NAS9-17239, NSF CER Grant MCS 8219196, Lockheed Engineering and Management Services, and ARO Grant DAA6-29-84-K-0061, DAA29-84-9-0027.

MODELING AND ANIMATING HUMAN FIGURES IN A CAD ENVIRONMENT

Dr. Norman I. Badler
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

ABSTRACT
With the widespread acceptance of three-dimensional modeling techniques, high-speed hardware, and relatively low-cost computation, modeling and animating one or more human figures for the purposes of design assessment, human factors, task simulation, and human movement understanding has become feasible outside the animation production house environment. This tutorial will address the state-of-the-art in human figure geometric modeling, figure positioning, figure animation, and task simulation.

INCORPORATING HUMAN FIGURE MODELS INTO 3-D CAD

As more complex environmental 3-D objects are designed with computer-aided systems, incorporating a human figure into the design or the evaluation of a design is becoming crucial. Though not new, the graphical basis for creating, modeling, and controling one or more human figures in a 3-D world is expanding and the application base is growing. Human figure models have long been used in cockpit and automobile occupant studies [14, 12, 28]; now they are finding application in space station design, maintainence assessment, and product safety studies.

There are several methodologies for integrating human figure models into a CAD system model:

- The self-contained system: the human figure modeling system contains an internal CAD system with which to build the object models.

- The CAD integral: the CAD system contains a figure model which can be manipulated in the same fashion as other designed objects. Often the figure model is selected from a fixed set of size, shape, and position choices.

- The CAD adjunct: the CAD system produces output files which may be used by the human figure modeler to establish the environment; in return, the figure modeler can return the body as an object to the CAD system.

For example, historical human factors systems tended toward the first option, such as Combiman [8] and Sammie [19]. With newer CAD modeling systems, the tendency swung to the second option whereby a specific model was built into the system, such as BUFORD [12] and PLAID [10]. Most recently, the third option has proven viable with the appearance of TEMPUS [3] as an adjunct to an existing CAD system (in fact, PLAID). Most research-oriented efforts find this route the path of least resistance as it separates the more difficult human factors problems from the reasonably well understood geometric design system.

With the expansion of the 3-D CAD field to include a large number of competent and competitive products, the future of human figure modeling must be to fit into this environment rather than supplant it. That is, the human figure must become just one other object to the design system, albeit one with

very special capabilities, requirements, and size variability. Thus the first two options have been the historical route to integrating people into the designed environment, while our own efforts have focused on the third approach as the more adaptable and viable way of managing existing software technology. This tutorial therefore intends to address the issues surrounding the incorporation of human figure models into an existing (or contemplated) 3-D, solid modeling, CAD environment.

Given that the third option is the one which most cleanly separates the issues of human figure modeling and CAD, there are a number of capabilities which must be addressed by a body modeling system:

- Creating and selecting individual or statistical human figure models, body sizes, and clothing.

- Providing interfaces to the CAD object information: object files, representation formats, and solid models.

- Providing graphics output: there must be suitable graphics for both bodies and objects, with the possibility of real-time viewing.

- Describing and animating the tasks to be performed.

- Offering appropriate analysis tasks: reach assessment, view assessment, collision detection, and population selection.

We will discuss all these items in the subsequent presentation. For a consistent on-going context, however, the discussion will be based on the TEMPUS system developed at the University of Pennsylvania. TEMPUS has been a relatively recent human factors system, was carefully designed and implemented from a Computer Science perspective, is undergoing active and continual development, and extends into some novel areas we feel are essential to human factors analysis.

ANTHROPOMETRY

Human bodies come in a wide variety of sizes. *Anthropometry* is concerned with measurement of a body to establish segment lengths (which cannot themselves be directly measured) and statistics on the distribution of body sizes in some population. A set of *standard anthropometric measurements* may be taken from a subject and segment lengths computed by regression equations to size all body segments. A standard system for this computation has been developed by Analytics, Inc.: the CAR system [18]. Unfortunately their data seems to have been derived from male subjects, as the female data is just scaled. Though male/female differences may have been unimportant to the early pilot simulators, a broader application base has now made this essential. Segment length data is essential for the proper computation of body position during reach and view assessment and the establishment of postures such as sitting.

A distinction must be made between specific individuals in a database and *generic* individuals computed by regression formulas or statistical data. In fact, one should have the ability to customize bodies to examine typically worse case situations. For example, TEMPUS lets the user construct segment lengths by a combination of percentiles and actual lengths. Thus a user can create a generic individual, for example, based on a 50^{th} percentile body but with 90^{th} percentile arms, 10^{th} percentile legs, and a given specific torso length.

Besides segment lengths, the body somatotype is important in determining the girth of various body segments. Though not as important for reach assessment, girth information is essential for proper clearance assessment and impacts joint limits. Torso and pelvic width may be the limiting factors in sitting comfort. Arm girth may make reaching into tight spaces difficult. During normal breathing, the torso may expand about 4%.

In the case of space shuttle and space station design it is important to know that the body height increases up to two inches due to the lack of gravity and the consequent expansion of the spine and joints. Thus the segments are not evenly scaled; rather, the length difference must be made up in the torso.

The body is represented as a tree structure of joints and segments. Each joint is characterized by a rotation angle or angles, and each segment by its length between the *proximal* and *distal* joints. (Proximal means closer to the body center and distal, further.) One distinguished joint is made the *body root*: sometimes the center of the pelvis (where the spine connects to the pelvis) and sometimes the center of mass. In the latter case, the location of the body root is not fixed in relation to another body joint; rather, there is a *prismatic* (sliding, variable length) "invisible" segment between the body center of mass and some actual body joint such as the center pelvis.

The location of the body root is important from a computational standpoint since it is from there that the nested transformations will be computed. Changing the body root to some other body joint is a useful and desirable capability. It permits some body joint to be considered to be *fixed* relative to the world coordinate system. While the center of mass is the proper such point for the human figure in free-fall, the body root can be established elsewhere for other effects. For example, during a walk, the body pivots around the ankle; during that phase of motion setting the body root at the ankle will allow simple expression of the rotation of the whole body relative to it [7, 35]. Of course, there may be countermotions in other joints, such as the hips.

The actual motion of the joints is not purely rotational. The geometry of some joints is in fact quite complex (for example, consider shoulders and knees) and may be only loosely approximated by pure rotations. Most human factors simulations appear to ignore these differences, though they are important for other biomechanical, medical, and prosthetic applications. We shall assume pure rotations, however, since the impact on human factors assessments appears to be reasonably small. Where it matters most is in the shoulder and this can be managed by using a *clavicle* joint to model the shoulder lift as the arm is raised [21].

For the most part, the segments are considered rigid. The one major exception is the spine which may be approximated by multiple segments or, better, by a spatial curve [21]. The difficulty with the former case is the coordination of the individual segments given desired global motions; for example, it is hard to prevent "jittering" of adjacent segments of a vertebral model [33]. The advantage to the latter is that the curve may be controlled by natural tangent vectors at each end. The length of the curve is tricky to control, but for limited torso motions it can be assumed relatively correct [21]. Without the curved spine the body shape is clearly unacceptable for extreme sideways or forward positions.

Body movements are restricted by joint limits and the presence of the rest of the body. While joint limits for simple hinges (one degree-of-freedom, such as the elbow) are reasonably easy to assess, limits for complex spherical joints are more difficult. Korein [21] used spherical polygons to describe the possible configurations of the distal segment to a joint. In general, accurate joint motion and joint limit models are difficult to obtain since joint geometry is never exactly spherical and limits are somewhat dependent on the individual. Body models often have a *comfortable* and an *extreme* value for the joint limits. Again, hard data is difficult to obtain.

The problem of avoiding collisions with the body itself is more work. Joint limits describe the relationship between adjacent segments, but for non-adjacent segments other comparisons must be used. Geometric comparison of the segment shape is required. With the use of bounding boxes, the computation cost may be kept to a minimum [6]. On the other hand, one can sometimes ask the user to visually assess the viability of a pose. Using the front and back clipping planes to delimit a "slice" of the scene through the figure and the object surfaces in question, the view can be established approximately parallel to the object surface to visually check collision or contact relationships. This scheme obviously benefits from fast hardware.

One of the foremost tasks of a human figure modeling system is the determination of the reachable space of the figure in some pose. This space has been determined empirically for particular individuals and populations and synthetically from joint limits [21]. Given the reach space, the objects in and out of the space may be simply determined by distance measurement or point-in-polyhedron tests.

All the anthropometry data is affected by external factors such as various suits or clothing. *Shirtsleeved* individuals naturally have more freedom of movement than those in space suits. Besides the effective girth and joint limit differences, there will be changes in the location of obstacles attached to the body (life-support systems, power supplies, communications gear, etc.). A human figure

modeling system should have the power to substitute different body segment structures in order to model the additional suit burden. In TEMPUS, body spheres can be differentially colored to simulate the appearance of tight, skin level clothing. Using a "Bubbble-editor" running on a Silicon Graphics Iris, any body segment shape may be customized. For other types of models, a relevant CAD system could be used provided that the resulting shapes are subject to correct parametric scaling to anthropometric data values.

If the body is to be subject to external forces (as in done in the crash simulation systems [28]), then additional information on the mass and center of mass must be stored with each segment. A model of joint resistance and muscle tension is frequently used to establish the proper joint reaction to applied torques. Many of the models limit the analysis to the 2-D situation, though robust 3-D models exist. Highly refined for occupant injury and safety assessment, they are nonetheless specialized application programs designed to simulate body reactions rather than actions.

A current challenge for anthropometry is the determination of reasonable strength data for individual joints. Unfortunately the state of data collection is not very good, nor are the experiments controlled over a well-specified set of test points. Individual variability is also a difficult problem and statistical data may be off considerably for a given individual. Strength data is inherently multi-dimensional, further complicating data collection; for example, wrist strength depends on arm position, wrist orientation, direction of force or torque applied, and fatigue. The data is not even symmetric: the torque generated clockwise at a certain wrist orientation and position may be quite different from that generated counterclockwise. Tables of strength data often specialize to particular body positions and tasks (such as moving a control stick) and are more concerned with statistics over populations rather than the range of torque values over a representative set of positions suitable for interpolation. One of the more comprehensive strength modeling systems is still relatively limited, confining its analysis mostly to the lifting task [13].

We are presently building a very general anthropometric database which greatly extends the current TEMPUS database and includes enough data to represent all the above information as best we can. Based on an implementation of a relational database (RDB from DEC), it will provide all the power of the associated database management system and query language with the structure and values needed to build specific or generic individuals for use with the rest of the modeling and animation system.

MULTIPLE FIGURE TYPES

Probably no one object has so challenged the abilities of current computational models as much as the human figure. The inherent variablity in shape and size, the supple covering of skin, and the extreme flexibility of the joints contribute to the modeling difficulty. Consequently there have been as many versions of human figure models as there are representational techniques. Among the methods used for body display are stick figures (joints and linear segments), surface lines, polygons (including models with more or less detail), curved surfaces, spheres, ellipsoids, potential functions, and superquadrics. Each scheme has advantages and disadvantages, of course. (See Table 1.)

One solution to this plethora of possibilities is to use multiple representations within the modeling system depending on the function desired. For example, polygons might be used for quick rendering, while curved surfaces might be used for efficient data storage. TEMPUS uses two different polygon models for quick display, and spheres for better segment shape rendering.

Another use for multiple representations is to select one according to the actual size of the displayed object on the screen [11]. The idea is to use the least detailed model consistent with good appearance. Used in flight simulators for environmental objects [34], it might be useful in situations with several people where some might appear smaller than others and require simpler representations.

With any representation the problem of building a model is considerable. Besides requiring that it be parametrized so that the segment length and girth data can be used to instantiate particular bodies, the general shape must be carefully designed for 3-D viewing, acceptable joint shape under rotation, and visual appeal. Various biostereometric sensing techniques may be used to generate segment data. The best models appear to be hand-crafted with interactive systems.

representation	advantages	disadvantages
stick	fast display	no depth information no twist information no girth information
surface lines	shows shape fast display	no joint smoothing see-through appearance
polygons	quick solid display can smooth shade	need many for detail robotic look joints difficult
curved surfaces	smooth surfaces better joints	more work to display
spheres	rounded segment shape easy to customize shape	can be bumpy tend to flat shading
ellipsoids	few needed	stylized shapes display more difficult
potential functions	smooth shape joins organic forms	expensive solid display non-adjacent segment blending
superquadrics	broad shape set	expensive solid display

Table 1: Body display types; advantages and disadvantages of each.

An important problem is that joint rotation effects surface shape. The skin deforms over the solid bone joints; this structure is not usually represented in the model and so must be simulated. One solution is to use spheres (either as spheres or polygon tesselations approximating spheres) centered at each joint. Joint motion therefore always leaves some fraction of the sphere visible. Another solution is to model the joint area as a curved surface patch with boundary curves somewhat away from the joint within the adjacent segments. Then joint motion changes the segment positions in which the boundary curves lie, thereby changing the shape of the patch. By carefully selecting the control points and the curve type, quite natural joint shape can be maintained [2, 24].

Other important characteristics of the human figure model are facial features and named surface points. The face allows unambiguous interpretation of the torso orientation. Various named body points can be used to refer conveniently to grip or touch points for limbs, eye position, seat center, etc. While facial expression is not normally needed for human factors work, it is possible that at least eye position would be useful to watch during a complex task. Facial expressions are an important communication channel and their animation is not too difficult [27]. With sound-synchronized speech motions [25], the figure could be made to describe the tasks it is being asked to perform!

GRAPHICS DISPLAY AND INTERACTION CONSIDERATIONS

We have already noted that the CAD system can be effectively separated from the human figure model. In this case, it is crucial that the available CAD system generate suitable 3-D solid models for transfer to the body positioning system. For the designed environments most often encountered which require human figures, the *boundary representation* approach appears quite satisfactory. (The alternative, *constructive solid geometry*, seems less appropriate due to the necessarily slow display time for the complex environment via ray-tracing.) The use of 2-D drafting-type systems is not satisfactory due to the inability to show a solid figure in a solid shaded 3-D world.

The inverse problem, that of providing a properly scaled and modeled human figure for the CAD system, obliges us to consider a model representation that can in fact be utilized by the CAD modeler. In practice, this would probably constrain the result to be passed as polygons, possibly obtained as tesselationss of more complex models such as curved patches, spheres, or superquadrics.

The TEMPUS system is a good example of a well-structured system for integrating human figure models with a CAD package. Figure 1 shows a block diagram of the major components of TEMPUS. In particular, note that the interfaces between TEMPUS and the external software environment are designed to allow TEMPUS to communicate with other CAD systems provided that database translators are available. TEMPUS provides its own hierarchic polyhedral surface modeling system (*SurfsUP*) rather than relying on the external host system. (While an IGES interface [22] could be used, we have found that it generates interface files that are too large for the complexity of environmental and body objects we use. In addition, the state of the solid modeling and boundary representation components of IGES are not yet generally accepted and standardized.)

During the following discussion, many of the features of TEMPUS will be described, so we will not delve into the diagram deeply now. TEMPUS itself is almost entirely written in VAX Pascal and runs under VAX VMS. Some of the lowest level graphics routines are FORTRAN; the highest quality rendering code is in portable C. The user interface is based on mouse, tablet, or keyboard interaction; the display is normally raster graphics images. The solid surface rendering component is callable through the underlying extended Core graphics system. The anthropometry section is based on a Pascal implementation of a database for bodies and their segment lengths. We are in the process of converting the anthropometric database to DEC RDB and severing it from TEMPUS. A body will be constructed or retrieved from the anthropometric database and re-formatted into a simple file. This file will then be used by TEMPUS for body display, reach analysis, and so on, avoiding a dependency on incorporating the RDB calls directly into TEMPUS (which is already big enough). All user inputs are processed by a User Interface Management System with a powerful *macro* facility for saving a re-playable record of the interactive session, saving body poses, or outputting an arbitrary sequence of commands for later playback, command extension, and simple convenience.

In designing or evaluating a human figure modeling system for CAD applications, it is important to remember who will be the system users and what their goals are going to be. In particular, there is a natural trade-off between display speed and display appearance. While some systems use line drawings for speed, they suffer from visual interpretation ambiguities. The cost of solid renderings can be prohibitive unless special (expensive) hardware or patient users are available. In TEMPUS, for example, the user may freely alternate between body display types so that rough positioning can proceed with rather coarse polyhedral models, then be switched to detailed sphere representation models for final adjustment. The cost of either approach is not severe: only a second or so for the model with about 150 polygons and about 10 seconds for the 2000 sphere *Bubblepeople*. Figure 2 shows four of the body styles available in TEMPUS. Figure 3 shows a closer view of the "Bubblewoman" model.

For shaded display TEMPUS uses three different methods. The simplest and least accurate is just a back-to-front polygon scan conversion mode supported directly in the underlying Core graphics system. Figure 2 was created this way. The second is a z-buffer mode which takes advantage of appropriate z-buffer display hardware (Figure 3). The third mode is a highly capable scan-line rendering system (CARICATURE) with the ability to handle multiple concentrated light sources, anti-aliasing, transparency, texture mapping, and various shading models (Figure 4).

The other graphics trade-off involves the interface to the user. Is she a programmer or not? Should the system be command- or menu-driven? How much knowledge of the computer system is required? In TEMPUS we opted for a menu-driven interface with a powerful macro facility so that more experienced users could customize their own command sequences. This approach seems to work quite well in practice. The principal users of TEMPUS are already non-programmer computer users, and TEMPUS appears much like a turnkey system to them. There would have been additional power for some users were TEMPUS given a "programmer's" interface; in retrospect we could have done this better though all TEMPUS code was sufficiently well structured. The main problem is refering to the numerous possible structures present in the system: objects (and their subparts), bodies (and their subparts), cameras, lights, reference coordinate systems, etc. The interactive interface frees the user

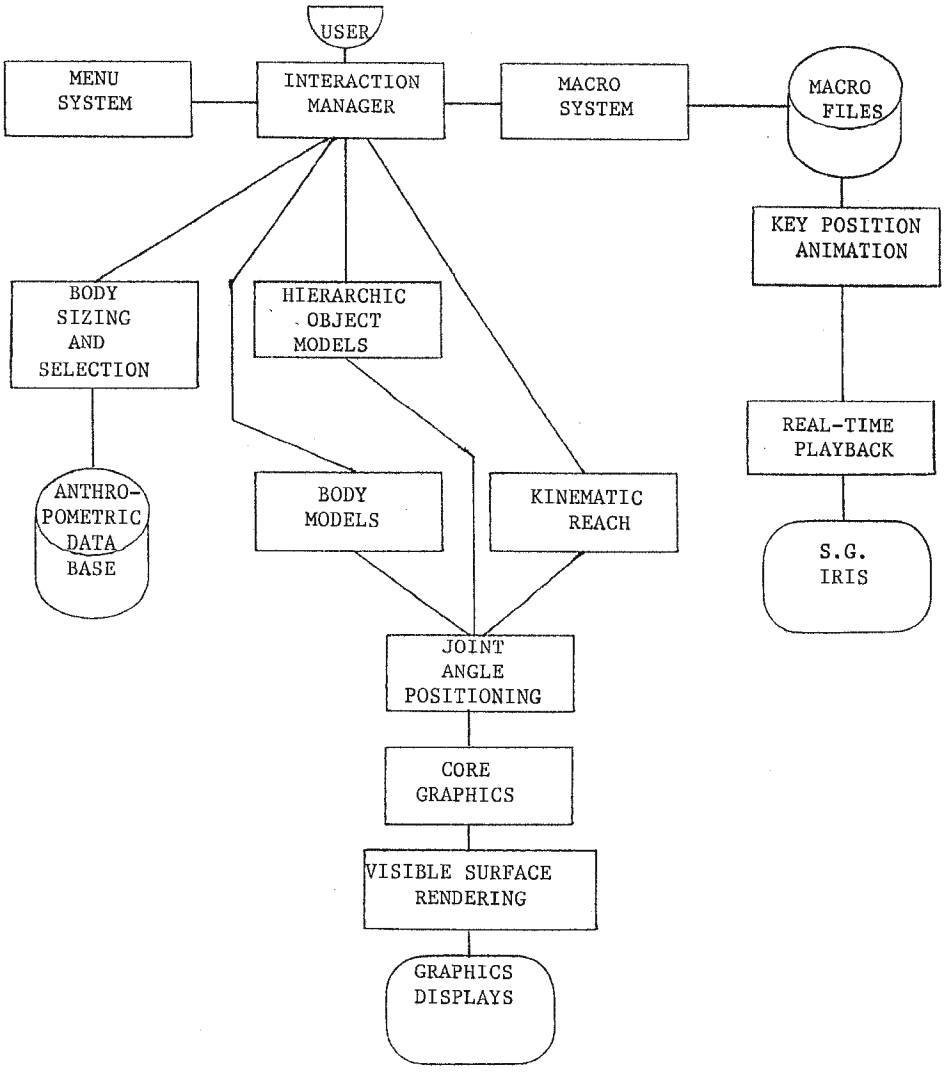


Figure 1: The TEMPUS system organization.

from the details of the computational representation and permits reference directly by name or world position from the image.

The internal structure of the interface is important for maintainability and extensability. TEMPUS uses Core graphics (our own implementation) for device independence. Output can be directed to a frame buffer (Grinnell), a z-buffer system (Lexidata Solidview), a Tektronix 4014, or a hardcopy laserwriter. For input, all user interaction is mediated by a User Interface Management System that handles viewport erasure, menu display, the macro system, a checkpoint facility, and input device mapping. The interface also permits TEMPUS to be run in a batch mode with macro-only input and no graphics. Images can be routed to a file for later viewing. This is especially valuable for complex image rendering.

The graphics output of a solid appearing model has additional benefits besides better appearance. In particular, biomechanical variables can be mapped into body part color; for example, maximum joint torques could be mapped into a color scale and displayed directly at each joint. The advantage is simply that the body structure and position is already shown directly and do not have to be separately tabulated; the additional dimensions are mapped onto local characteristics of the body display.

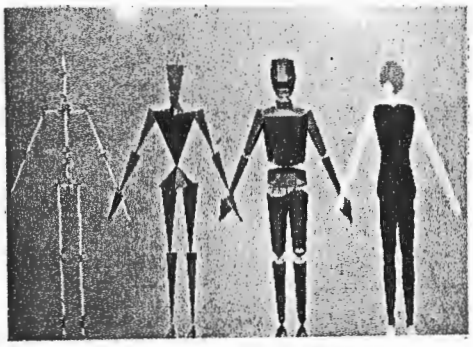


Figure 2: Bodies in TEMPUS: stick, "Mr. T", "Polybody," and "Bubbbleperson."

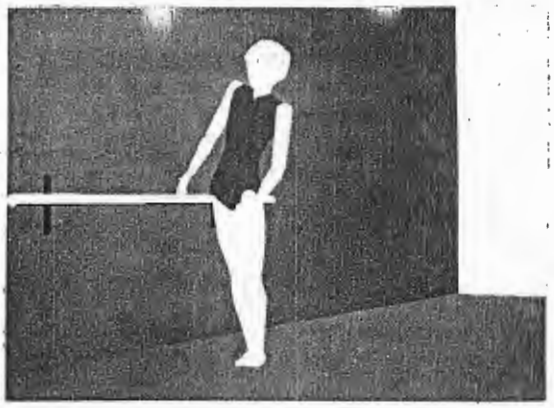


Figure 3: "Bubblewoman" figure in TEMPUS.

Figure 4: Four polygon bodies rendered with CARICATURE from TEMPUS.

Collision detection is an important analysis task. In an interactive system we can trade computation time for user time if a suitably fast display is available. This means that rather than computing the geometric intersection between body segments and between the body and the environment, we allow the user to alter the viewing position and the clipping planes in such a way that any collisions are seen directly. Of course, computation must be used if the system is being run in a batch mode.

This last point brings up the issue of real-time display. The ability to see complex objects in real time is surely valuable, though it comes at a price. The principal decision, given a fixed amount of funds available for a human factors workstation, may be whether to invest in one real-time system or several less capable ones. Rather than tie the decision to very specific display hardware, TEMPUS takes the position that real-time playback can be used if available, otherwise a number of less capable workstations or displays can be supported. Given the current availability of high performance workstations such as the Silicon Graphics Iris 3030, the decision regarding TEMPUS is perhaps dated. It is now possible to incorporate near-real-time raster display for modest complexity scenes and bypass the host-intensive image generation process on a frame buffer peripheral. As we will see later, the issue of real-time positioning and motion control is now much easier to solve than it was a few years ago. Moreover, such systems can be built in the context of well-established raster graphics rather than in the waning light of vector systems to achieve the required speed of 3-D display.

POSITION AND MOTION CONTROL

One of the purposes of a human figure modeling system is to permit a user to position a figure in a 3-D environment and perform various assessments of capability. The two principal methods of positioning are via *kinematics* and *dynamics*. *Forward kinematics* is the positioning of a point by traversing the joint angle rotation and segment length matrices from some root node outward to the leaves of the body tree. This is standard graphics knowledge. Much harder is the problem of *inverse kinematics*, that is, determining the joint angles so that the end effector (right hand grip, for example) is to be at some particular point in space. There are several ways of computing inverse kinematics, most borrowed from robotics. Analogously, there are *forward dynamics* and *inverse dynamics* which use forces and torques to describe motion. In forward dynamics, joint torques produce acceleration, velocity, and position directly from Newton's Laws. Inverse dynamics computes joint torques given the path, velocity, and acceleration of the end effector.

In general, joint angle changes are simple to input by the user and are very simple to implement, but are difficult to control in order to achieve a specific end effector goal. Several different joint angle systems can be used, including roll-pitch-yaw, Euler angles, halfplane-deviation-twist, and quaternions. TEMPUS uses halfplane-deviation-twist though we are likely to move toward quaternions for reasons that will be mentioned later.

Since one of the principal tasks of a human figure modeling system is reach assessment, some form of inverse kinematics must be available to make positioning practicable. Inverse kinematics can range from simple triangle computations [21] to more complex pseudo-inverse systems [16]. In any case, joint limits must be respected. The joint limit issue is further complicated by the presence of clothing and environmental constraints such as walls, seats, and restraining harnesses. One of the best measures of generality of a human factors modeling system could be, in fact, the flexibility of the inverse kinematics available for general body positioning. The easier it is to position the ends of the limbs, the easier it will be to assess reach. View assessment may require the head to be used as the end effector. Below we will examine a very general approach to this problem of inverse kinematics which we are building into TEMPUS.

Specifying a reach goal is not quite as simple as it might seem at first. Goal points may be on environmental objects, on the body itself, or just in space (such as for the direction of view). While points may be selected directly from the screen display, the general specification of a location and orientation requires that the system have a good, accessible representation for object coordinate systems. In TEMPUS, points for reach or view may be specified in terms of a *reference coordinate system* located in the world, on a convenient object, or on a body in the workplace. Using commands to *align*, *rotate about*, or *move relative to*, a user can quickly indicate the desired relationship between a point and a reach end effector, the view direction, or another coordinate system. The latter feature is used to orient and position one object with respect to another, giving TEMPUS the capability of

workplace redesign through transformational changes without modifying the underlying geometric structure of the objects.

Once given a goal (for a limb, for example) we may wish to view goal achievement as as series of steps from the present position. The inbetween configurations may be manually examined for collisions. If suitable hardware is available, the motion may be displayed in real-time. The technique of animating from one position to another based on interpolations of some sort is called *key-positioning* or *key-pose* animation. Interpolation methods are based on suitable formulations of spline curves to insure that the given *key* values are achieved and that the starting and ending motions are smooth and joinable [20, 30]. The interpolations are performed over convenient parameters: if goal positions are used, then a straight-line path from the present end effector position to the required one is used and inverse kinematics determines the inbetween joint angles along the limb segment chain for each step of the movement. Interpolating joint angles directly is less work computationally, but results in a less satisfactory motion since the path of the end effector will not necessarily be the shortest one to the goal. The advantage of the scheme described by Steketee and Badler is that the motion along the path of movement may be controlled independently of the geometric form of the path and vice versa. With simpler splining schemes changing the key values will alter both the path and the kinetics along the path simultaneously.

Animation of locomotion or zero-gravity movements requires additional effort. Body motions are reactions to external and internal forces so that dynamic analysis is needed [16, 17]. Although forward dynamics is easy to compute, the problem of specifying the proper forces, torques, and joint characteristics is not. Considerable effort has been expended in developing crash simulations, for example, in which the body is acted upon by external (deceleration) forces, flinging the limbs about in space. The control, in this case, is rather simple since only the deceleration and any restraints (known in advance) are considered. Moreover, the crash simulations require the body to be represented without closed loop chains, that is, as a strict tree structure with no cycles. This means that dynamics with the two hands locked together or with the hands and feet both restrained cannot be computed. In comparison, the task of determining the forces needed to make a figure walk, fall, or sit up is much more difficult [31, 32, 1] and may involve closed loops. In practice, it appears that a mix of kinematics and dynamics is needed to control effectively an animated figure; for example, Girard and Maciejewski use dynamics to compute the path of the center of mass of the figure, then use kinematics to force the legs to achieve the hip positions so determined. Wilhelms has also found that kinematic positioning is a good starting point for estimates of forces: by computing the accelerations and velocities from kinematically interpolated goals, reasonable joint torques may be computed from inverse dynamics. These torques may then be modified to achieve particular variations in the original motions by forward dynamics.

The approach to dynamics taken in TEMPUS is to use a general mechanical engineering tool to solve for the body dynamics as a spatial linkage mechanism [26]. The interface to this system is currently under development.

Recently, we have been investigating a very general formulation of the reach problem. Considering the application of the human figure to human factors problems, we determined that there was one general situation where standard kinematic approaches were inadequate: positioning arbitrary body joints in space simultaneously. This means, for example, that there could be simultaneous *constraints* on any and all body joints. Some might be involved in reaches, some in restraints, and some in positions based on environmental objects. A figure sitting in a chair and restrained by a lap belt, for example, is constrained by his posture, the belt, and the reach task. By formulating these constraints in a simple fashion and solving them effectively simultaneously, we have achieved a speed and positioning flexibility unmatched in any other system. The algorithm, POSIT, is an adjunct to TEMPUS positioning and permits arbitrary multiple constraint situations to be used to establish body poses: reach goals, restraints, and environment objects [4]. The algorithm works by considering goals as springs with various user-specified spring constants attached to selected joints. When the body joints are "released," they move to reduce the spring forces acting on the constrained and intermediate joints by a recursive algorithm. This algorithm is repeated until there is essentially no change in position. It is very simple and natural to specify and watch the achievement of two-hand reaches, simultaneous hand and foot reaches, and reaches under lap, shoulder, foot, torso, etc. restraints. Positioning a figure in a chair is a matter of specifying a few goals for the appropriate body joints. Since the goals and constraints are springs, there is no penalty for unachievable goals: one can simply

measure the failure distance. By changing the spring constants, some goals can be favored over others; for example, the restraints could be maintained at the expense of achieving the reaches. In addition, the algorithm runs in near-real-time on a Silicon Graphics Iris 3030 workstation. POSIT works by trading off complicated inverse kinematics for a simple iteration of a recursion based on a very simple inverse kinematic formulation (essentially a triangle system). In practice, the POSIT system reach goals and other constraints are specified interactively with a simple body model and a Polhemus 6-axis digitizer.

Extensions in progress for POSIT include joint angles represented by *quaternions* [29], joint angle limits, orientation as well as positional goals, and constraint regions. The latter are 0, 1, 2, or 3-D regions within which a point is permitted to reside. The weighting function for the strength of the constraint is described as a constant, linear, or potential function over the region.

With these extensions is a change to the interface to POSIT to remove its dependence on interactive user input. We are designing and implementing a *constraint language* so that the POSIT positioning algorithm can run as a procedure independent of the actual interface. Among other things, the constraint language will permit the specification of goals in arbitrary spatial regions, to arbitrary body parts, and with more general weighting functions. Goals such as "keep the feet on the floor," "keep your hand on the railing," or "keep your eye on the ball" will be naturally expressed.

ARTIFICIAL INTELLIGENCE CONTROL METHODS

As wonderful as interactive computer graphics is, we cannot ignore the fact that ultimately it is some *task* being performed that is under study. The task might be as simple as "does this person (or some considerable fraction of the population) fit in this space?" But in the future it appears as if the tasks are more likely to be phrased as "can this person (or population) *do* this task in this space?" The task orientation means that much of the purely positioning aspects of the tasks might not need to be described graphically at all.

Consider the problem of simulating a task to be carried out by several crewmembers of the space station. They have written procedures for what they are to do, but we must assess whether they can perform the task cooperatively, whether they will get in each other's way, whether there are enough hands available to insure safe and effective execution of the task, and whether all necessary actions are in fact carried out. This scenario is more than just the repetition of some action over a given population, and it is not something that can be done one person at a time. Controlling the actions of multiple figures in task-oriented environments is not simply a graphical interaction problem, but rather demands a more general approach.

We are using Artificial Intelligence methods to build such flexible control structures with the *HIRES* system [15]. HIRES is a production rule simulation system which allows the description and execution of a process represented as one or more sets of production rules completely describing the process at some level of detail. The production rules are *if-then* kinds of statements: if the pattern of the left-hand-side (*if*) matches known information stored in the knowledge base, then the code on the right-hand-side (*then*) is executed. This code can add, delete, or change assertions in the knowledge base, permitting other rules to *fire*. A set of production rules constitutes a process description since it describes the conditions for action and the consequences of those actions.

If there is more than one level of detail in the HIRES process representation, the user can move between the levels as desired to run the simulation at greater, lesser, or *alternative* levels of detail. For example, the space station crewmember task may be described as a series of state changes at a high level: don suit, open airlock, enter airlock, close airlock, depressurize, open exterior airlock door, egress into space, etc. At a lower level, it might be described as a series of reaches and views. HIRES is responsible for synchronizing the activities of each crewmember, each of whose tasks will be described by suitably sequenced or conditional actions.

Since certain tasks may be more crucial from a human factors viewpoint, these tasks might be simulated at a lower, more detailed level; other less important tasks may be simulated at a high level. For example, in an elevator simulation built in HIRES (Figure 5), the top two levels are *quantitative simulations* based on simple running or disabled elevator states (level 1) and a kind of user's view of "which floor is the elevator now on?" (level 2). The third level is a queuing system model of the

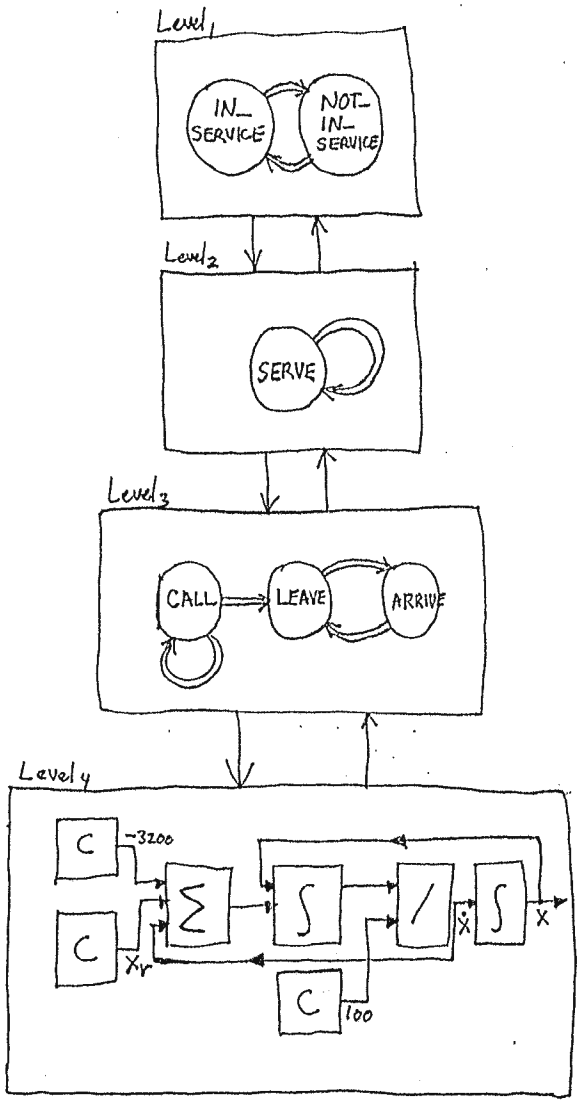


Figure 5: The levels of the elevator simulation in HIRES.

control algorithm used by the elevator to service customer requests. The fourth level is a continuous system simulation of the physical control of the elevator in the shaft. The HIRES user can move freely between levels; transition production rules provide knowledge generalization when moving up to higher levels and a probabilistic or discrete specialization when moving down.

As the elevator example illustrates, the tasks or processes need not be described in production rules directly: HIRES provides preprocessors for several convenient description formats such as Petri nets, augmented transition networks, continuous system simulation, discrete system simulation, and scripts (time-tagged assertions). Though internally stored as production rules, the external manifestation is in a form most convenient to the particular problem and the user's analysis of it.

Within HIRES, it is possible to build general production rules to model an agent's abilities, preferences, and goals. These *agent models* allow the expression of many features of a human model that would not easily fit into a more restricted CAD and graphical approach. In particular, we can express handedness ("use the pilot's right hand for reaches unless it is occupied by another task or cannot reach the goal without violating the current restraints"), responsibility ("the commander must

move first" or "mission-specialist-1 is not allowed to touch panel A-113"), ability ("mission-specialist-2 is sleeping"), concurrency ("hold down button PB-2 while moving dial V-7"), and knowledge ("passenger P-1 does not know where any flight controls are located"). Once the task scenario and interaction rules are determined, HIRES can simulate the situation and produce a series of constraint assertions for execution by the TEMPUS positioning system POSIT.

Figure 6 shows a Petri net formulation of the interaction of five people in the so-called *Dining Philosophers* problem. There are five people sitting around a table eating with chopsticks, but there are only five chopsticks available, one between each pair of people. The chopsticks cannot be passed around. A person can eat only when he is able to grab both of his adjacent chopsticks; otherwise he must wait. One consequence of these conditions is that at most two people can eat at any instant of time. Figure 7 shows hardcopy output of one frame of the situation simulated by HIRES from the Petri net description of the process.

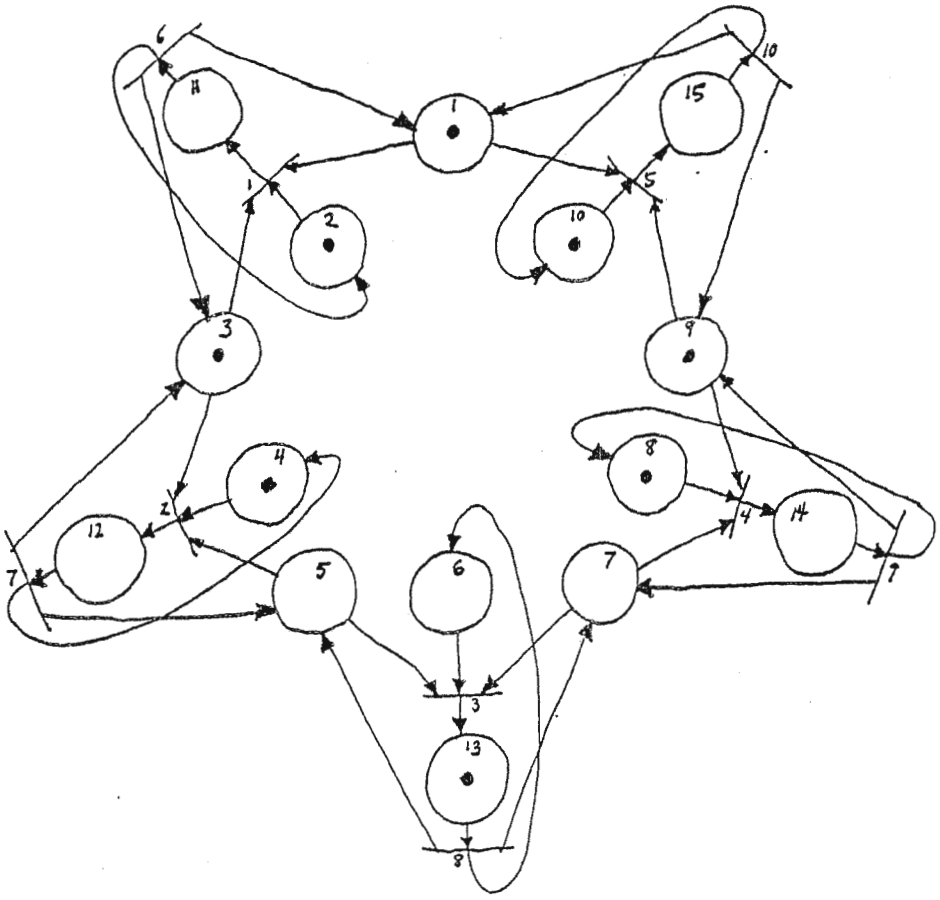


Figure 6: Petri net process description for the Dining Philosophers problem.

There are a number of knowledge base issues which must be addressed by HIRES to realize this and other testing scenarios. In general, however, the basic mechanism is satisfactory. The usual questions of knowledge base representations, production rule execution efficiency, and knowledge acquisition are the most important. We presently have a simple assertional knowledge base in HIRES and are investigating its extension to a frame-based system build on top of DEC RDB. For the production rule engine, we expect to extend its *forward chaining* simulation activity to a *backwards chaining* planning

capability. This would dramatically increase HIRES' ability to solve problems rather than just ex (simulate) commands. Finally, the insertion of the task and agent models should be done interacti from the anthropometric database and possibly a command or language interface. We will return the latter issue shortly.

A different problem that could be addressed by robotics techniques is that of *collision avoidance*. Knowing the layout of objects in the environment, a robot can be instructed to avoid collisions on

Figure 7: A scene from the hit movie, "The Dining Philosophers."

way to pick up objects or negotiate a path. Unfortunately, the most effective algorithms are based the assumptions of a relatively small number of degrees of freedom (usually 6 or less for the typic robot), a fixed environment of objects, and often a fixed base robot [23, 9]. The typical human fac problem violates most of these assumptions: the body has numerous degrees of freedom even whe restrained, the environment is complex and movable, and the whole body may be free to move (fo example, in space). For the near future we expect that collision avoidance will be done by a combination of algorithmic techniques, such as setting up *negative* goals for nearby objects and bc joints within the POSIT framework, or general geometric planning with HIRES, and manual viewi techniques as outlined earlier.

NOVEL INTERFACES

Let us return to the user of a human factors system and see how input to the system might be impr in the light of the preceding discussion. First of all, it is certain that there is a significant need to interact with the human figure as directly as possible. Since the graphical interaction takes the for joint angles, reach goals, and body positions, natural 3-D locations and orientations must be specif We are using a Polhemus 6-axis digitizer to input such information to POSIT [4]. In addition, the Polhemus is used to position and orient workplace objects and the actual viewing *camera*. Such in cuts down dramatically the number of individual command selections required to achieve a partic positioning task.

Another area in which the user interface is being improved is at the task level. We have already discussed HIRES and the agent models. The task itself may be specified in a natural or artificial language. This is exactly the situation at NASA: procedures are carefully written as a *checklist* wh simplifies the syntax of an action to a standardized format. Whenever the format is insufficient, natural language is used. Charts, graphs, tables and other diagrams are also used as needed. Any system which expects to simulate the actions of agents in this environment should be able to interp the same types of information without requiring any explicit conversion by the user except, perhap typing it in. (There is no standard computerized database for these checklists, though such an effo under consideration at NASA.)

To provide such a language-oriented interface to TEMPUS, we built the system diagrammed in Fi 8 [5]. We have taken a natural language parser and adapted it to process natural language *and* checklist syntax for a reasonable set of action verbs (Table 2) describing operations on panel type objects. The meaning of each field of the verb *case frame* is shown in Table 3.

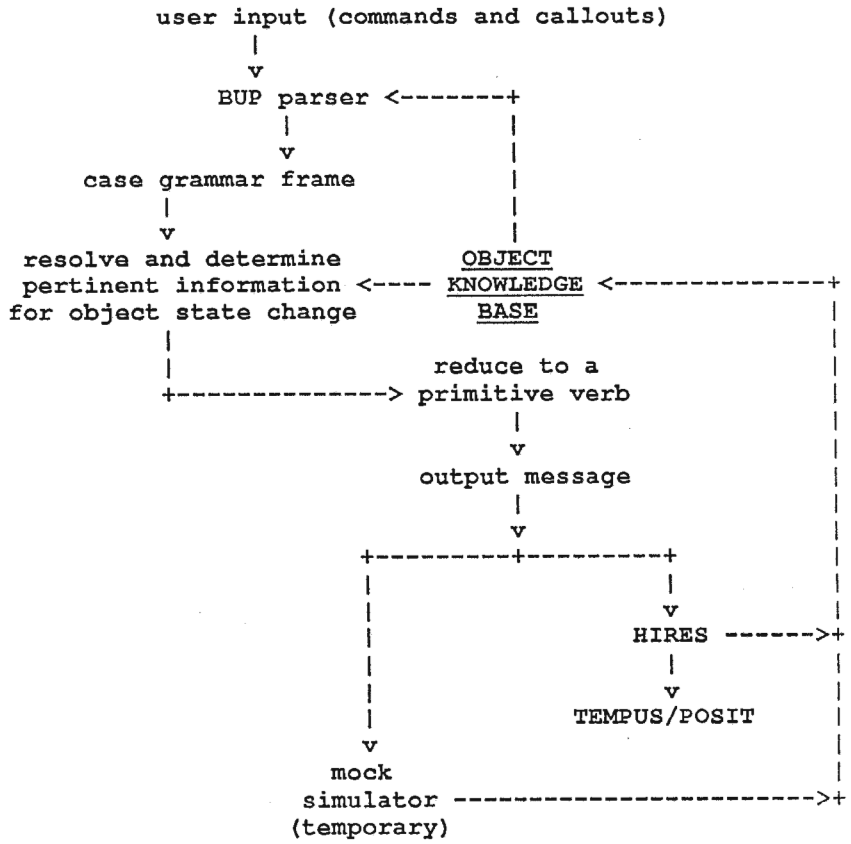


Figure 8: Organization of the natural language and checklist interface.

The panel objects include switches, pushbuttons, toggles, valves, meters, and indicators. Figure 9 shows a diagram of a sample panel with a wide variety of objects.

```
PUT:                  [(Agent) Object Locative]
TURN:    [(Agent) Object (Locative) (State)]
ROTATE:  [(Agent) Object (Locative) (State)]
OPEN:    [(Agent) Object (Locative) (State)]
CLOSE:   [(Agent) Object (Locative) (State)]
MOVE:    [(Agent) Object (Locative) (State)]
PUT ON:                      [(Agent) Object]
PUT ASIDE:                   [(Agent) Object]
PUT DOWN:          [(Agent) Object (Locative)]
PUT OUT:                     [(Agent) Object]
TURN ON:                     [(Agent) Object]
TURN OFF:                    [(Agent) Object]
TURN UP:               [(Agent) Object (State)]
TURN DOWN:             [(Agent) Object (State)]
TURN OUT:                    [(Agent) Object]
TURN OVER:             [(Agent) Object (State)]
OPEN UP:                     [(Agent) Object]
CLOSE DOWN:                  [(Agent) Object]
```

Table 2: Verb Case Frames. The parenthesized items are optional.

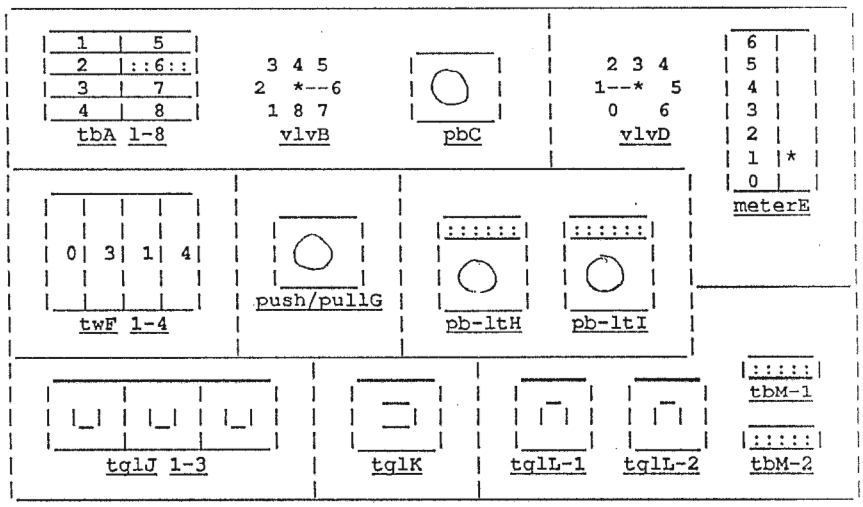
```
ACTION:      the type of action
AGENT:       the instigator of an action
OBJECT:      an obligatory case found or implied with each verb
INSTRUMENT:  the object used to perform the action
LOCATIVE:    the location (source - destination) of the object
DIRECTIVE:   the direction (source - destination) of the object
STATIVE:     the state (initial - final) of the object
TIME:        the time (initial - final) used for the action
TEMPORAL:    temporal conditions imposed on the action
```

Table 3: Case Frame Roles

There is a symbolic representation for each panel object so that, for example, the connection between an indicator light, a valve, and a push-button switch can all be made explicit to the parser as it tries to understand the command. The possible states of each device and the allowable transformations from one state to another are also stored. For example, a toggle structure is:

```
(CONTROL
   :name "TOGGLE K"
   :texture-map ()
   :message ()
   :type-of SWITCH
   :sub-type TGL
   :locative PANEL1
   :direction (LEFT RIGHT)
   :states (OFF ON)
   :movement (DISCRETE MM LINEAR ((OFF ON) 20 5))
   :current ON)
```

The result is that much of a task consisting of panel object reaches can be specified symbolically rather than graphically. The task description is converted into a HIRES script; HIRES can then simulate the task with a given agent or try several possible agents.



- tbA, tbM-1 and tbM-2 are talkback indicators that indicate an *on* or *off* state.
- vlvB and vlvD are rotary valves.
- pbC is a push button.
- meterE is a bar meter indicator.
- twF are thumbwheel switches.
- push/pullG is a switch that moves in the Z axis.
- pb-ltH and pb-ltI are push-buttons with an indicator light.
- tglJ-1, tglJ-2, tglJ-3, tglK, tglL-1, and tglL-2 are toggle switches.

Figure 9: Sample panel layout with various control and indicator devices.

Since panel-type objects are small and geometrically rather complex (toggle switches, valves, pushbuttons) we do not believe that all this detail must be represented in actual solid models. Rather, we represent the panel (which is usually flat anyway) as a polygon and apply a *texture map* to its image which may be either an actual photograph of the panel or a synthetic 2-D drawing of the salien objects. By associating the position of the device in the texture map with the symbolic description of the device in the databases for the natural language parser and HIRES, the location of the device may be used for reaches, views, etc. without the overhead of all the geometric complexity. Moreover, if tl panel layout is being designed concurrently with the human factors analysis, changes to it may be reflected immediately in the display of the texture on the polygon panel. There is no need to invoke a 3-D modeller; a 2-D graphic editor or just a quick photo digitization will suffice. Even more attractive is the fact that the texture map need not even be drawn on the graphics display unless a detailed imag is required: the *task* which creates the reach action is in control and if it is achieved the hand (for example) will go to the proper point in space where the control or device *should* be; if that reach cannot be achieved, then a suitable message will be issued.

ACKNOWLEDGMENTS

The research reported here from the University of Pennsylvania is partially supported by NASA Contract NAS9-17239, NSF CER Grant MCS-82-19196, Lockheed Engineering and Management Services, and ARO Grant DAAG29-84-K-0061 including participation by the U.S. Army Human Engineering Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the United States Government and its granting agencies.

REFERENCES

1. Armstrong, W. W. and Mark Green. "The dynamics of articulated rigid bodies for purposes of animation". *The Visual Computer 1*, 4 (1985), 231-240.

2. Badler, Norman I. and Mary Ann Morris. Modelling flexible articulated objects. Online Conf. Computer Graphics '82, October, 1982, pp. 305-314.

3. Badler, Norman I., Jonathan D. Korein, James U. Korein, Gerald Radack, and Lynne S. Brotman. "Positioning and animating human figures in a task-oriented environment". *The Visual Computer: The International Journal of Computer Graphics 1*, 3 (1985).

4. Badler, Norman I., Kamran H. Manoochehri, and David Baraff. Multi-dimensional input techniques and articulated figure positioning by multiple constraints. Proc. Workshop on Interactive 3D Graphics, New York, NY, October, 1986.

5. Badler, Norman I. and Jeffrey S. Gangel. Natural language input for human task description. Proc. Instrument Society of America ROBEXS '86: The Second International Workshop on Robotics and Expert Systems, June, 1986, pp. 137-148.

6. Badler, Norman I., Joseph O'Rourke, and Hasida Toltzis. "A spherical representation of a human body for visualizing movement". *IEEE Proceedings 67*, 10 (Oct. 1979), 1397-1403.

7. Badler, Norman I., Joseph O'Rourke, and Bruce Kaufman. "Special problems in human movement simulation". *Computer Graphics 14*, 3 (1980), 189-197.

8. Bapu, P., S. Evans, P. Kitka, M. Korna, and J. McDaniel. User's guide for COMBIMAN programs. No. AFAMRL-TR-80-91, Univ. of Dayton Research Institute, Jan, 1981. U.S.A.F. Report.

9. Brooks, Rodney. Solving the find-path problem by good representation of free space. Proc. AAAI National Conf. on Artificial Intelligence, Pittsburgh, PA, 1982, pp. 381-386.

10. Brown, Jeri W. Using computer graphics to enhance astronaut and systems safety. Proc. 15th Symposium on Space Safety and Rescue, International Academy of Astronautics, 33rd International Astronautical Federation Congress, Paris, France, 1982, pp. 1-8.

11. Clark, James. "Hierarchical geometric models for visible surface algorithms". *Comm. of the ACM 19*, 10 (Oct. 1976), 547-554.

12. Dooley, Marianne. "Anthropometric modeling programs -- A survey". *IEEE Computer Graphics and Applications 2*, 9 (Nov. 1982), 17-25.

13. Evans, Susan M. R.. *Ergonomics in manual workspace design: Current practices and an alternative computer-assisted approach.* Ph.D. Th., Center for Ergonomics, University of Michigan, Ann Arbor, MI, 1985.

14. Fetter, William. "A progression of human figures simulated by computer graphics". *IEEE Computer Graphics and Applications 2*, 9 (Nov. 1982), 9-13.

15. Fishwick, Paul A. *Hierarchical Reasoning: Simulating Complex Processes over Multiple Levels of Abstraction.* Ph.D. Th., Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1986.

16. Girard, Michael and A. A. Maciejewski. "Computational modeling for the computer animation of legged figures". *Computer Graphics 19*, 3 (1985), 263-270.

17. Girard, Michael. Interactive design of 3-D computer-animated legged animal motion. Proc. Workshop on Interactive 3D Graphics, New York, NY, October, 1986.

18. Harris, R., J. Bennet, and L. Dow. CAR-II - A revised model for crew assessment of reach. report 1400.06B, Analytics, Willow Grove, PA, June, 1980.

19. Kingsley, E., N. Schofield, and K. Case. "SAMMIE-a computer aid for man-machine modeli *Computer Graphics 15*, 3 (Aug. 1981), 163-169.

20. Kochanek, Doris H. U. and Richard H. Bartels. "Interpolating splines with local tension, continuity, and bias control". *Computer Graphics 18*, 3 (1984), 33-41.

21. Korein, James U.. *A Geometric Investigation of Reach.* MIT Press, Cambridge, MA, 1985.

22. M. H. Liewald and P. R. Kennicott. "Intersystem data transfer via IGES". *IEEE Computer Graphics and Applications* 2, 3 (May 1982), 55-63.

23. Lozano-Pérez, Tomás and Michael Wesley. "An algorithm for planning collision-free paths among polyhedral obstacles". *Comm. of the ACM* 22, 10 (Oct. 1979), 560-570.

24. Nisselson, Jane. Computer Graphics in the Fashion Industry. Proc. Graphics Interface '86, Vancouver, 1986, pp. 1-6.

25. Parke, Frederic. "Parameterized models for facial animation". *IEEE Computer Graphics and Applications* 2, 9 (Nov. 1982), 61-68.

26. Paul, Burton and Ronald Schaffa. DYSPAM User's Manual. Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania.

27. Platt, Stephen. *A Structural Model of the Human Face.* Ph.D. Th., Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, 1985.

28. Prasad, Priyaranjan. An overview of major occupant simulation models. Proc. Society of Automotive Engineers, 1984.

29. Shoemake, Ken. "Animating rotations with quaternion curves". *Computer Graphics 19*, 3 (19[illegible]). 245-254.

30. Steketee, Scott and Norman I. Badler. "Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control". *Computer Graphics 19*, 3 (1985), 255-262.

31. Wilhelms, Jane and Brian A. Barsky. Using dynamics for the animation of articulated bodies such as humans and robots. Proc. Graphics Interface '85, Montreal, 1985, pp. 97-104.

32. Wilhelms, Jane. Virya - A motion editor for kinematic and dynamic animation. Proc. Graphics Interface '86, Vancouver, 1986, pp. 141-146.

33. Willmert, K. D. "Visualizing human body motion simulations". *IEEE Computer Graphics and Applications* 2, 9 (Nov. 1982), 35-38.

34. Yan, Johnson K. "Advances in computer-generated imagery for flight simulation". *IEEE Computer Graphics and Applications 5*, 8 (Aug. 1985), 37-51.

35. Zeltzer, David. "Motor control techniques for figure animation". *IEEE Computer Graphics and Applications* 2, 9 (Nov. 1982), 53-59.