SIMULATING HUMANS: COMPUTER GRAPHICS, ANIMATION, AND CONTROL

Norman I. Badler
Cary B. Phillips[1]
Bonnie L. Webber
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

Oxford University Press

March 25, 1999

[1]Current address: Pacific Data Images, 1111 Karlstad Dr., Sunnyvale, CA 94089.

To Ginny, Denise, and Mark

Contents

Preface

The decade of the 80's saw the dramatic expansion of high performance computer graphics into domains previously able only to flirt with the technology. Among the most dramatic has been the incorporation of real-time interactive manipulation and display for human figures. Though actively pursued by several research groups, the problem of providing a virtual or synthetic human for an engineer or designer already accustomed to Computer-Aided Design techniques was most comprehensively attacked by the Computer Graphics Research Laboratory at the University of Pennsylvania. The breadth of that effort as well as the details of its methodology and software environment are presented in this volume.

This book is intended for human factors engineers requiring current knowledge of how a computer graphics surrogate human can augment their analyses of designed environments. It will also help inform design engineers of the state-of-the-art in human figure modeling, and hence of the human-centered design central to the emergent notion of Concurrent Engineering. Finally, it documents for the computer graphics community a major research effort in the interactive control and motion specification of articulated human figures.

Many people have contributed to the work described in this book, but the textual material derives more or less directly from the efforts of our current and former students and staff: Tarek Alameldin, Francisco Azuola, Breck Baldwin, Welton Becket, Wallace Ching, Paul Diefenbach, Barbara Di Eungenio, Jeffrey Esakov, Christopher Geib, John Granieri, Marc Grosso, Pei-Hwa Ho, Mike Hollick, Moon Jung, Jugal Kalita, Hyeongseok Ko, Eunyoung Koh, Jason Koppel, Michael Kwon, Philip Lee, Libby Levison, Gary Monheit, Michael Moore, Ernest Otani, Susanna Wei, Graham Walters, Michael White, Jianmin Zhao, and Xinmin Zhao. Additional animation help has come from Leanne Hwang, David Haynes, and Brian Stokes. John Granieri and Mike Hollick helped considerably with the photographs and figures.

This work would not have been possible without the generous and often long term support of many organizations and individuals. In particular we would like to acknowledge our many colleagues and friends: Barbara Woolford, Geri Brown, Jim Maida, Abhilash Pandya and the late Linda Orr in the Crew Station Design Section and Mike Greenisen at NASA Johnson Space Center; Ben Cummings, Brenda Thein, Bernie Corona, and Rick Kozycki of the U.S. Army Human Engineering Laboratory at Aberdeen Proving Grounds; James Hartzell, James Larimer, Barry Smith, Mike Prevost, and Chris Neukom of the A^3I Project in the Aeroflight Dynamics Directorate of NASA Ames Research Center; Steve Paquette of the U. S. Army Natick Laboratory; Jagdish Chandra and David Hislop of the U. S. Army Research Office; the Army Artificial Intelligence Center of Excellence at the University of Pennsylvania and its Director, Aravind Joshi; Art Iverson and Jack Jones of the U.S. Army TACOM; Jill Easterly, Ed Boyle, John Ianni, and Wendy Campbell of the U. S. Air Force Human Resources Directorate at Wright-Patterson Air Force Base; Medhat Korna and Ron Dierker of Systems Exploration, Inc.; Pete Glor

and Joseph Spann of Hughes Missile Systems (formerly General Dynamics, Convair Division); Ruth Maulucci of MOCO Inc.; John McConville, Bruce Bradtmiller, and Bob Beecher of Anthropology Research Project, Inc.; Edmund Khouri of Lockheed Engineering and Management Services; Barb Fecht of Battelle Pacific Northwest Laboratories; Jerry Duncan of Deere and Company; Ed Bellandi of FMC Corp.; Steve Gulasy of Martin-Marietta Denver Aerospace; Joachim Grollman of Siemens Research; Kathleen Robinette of the Armstrong Medical Research Lab at Wright-Patterson Air Force Base; Harry Frisch of NASA Goddard Space Flight Center; Jerry Allen and the folks at Silicon Graphics, Inc.; Jack Scully of Ascension Technology Corp.; the National Science Foundation CISE Grant CDA88-22719 and ILI Grant USE-9152503; and the State of Pennsylvania Benjamin Franklin Partnership. Martin Zaidel contributed valuable LaTeX help. Finally, the encouragement and patience of Don Jackson at Oxford University Press has been most appreciated.

Norman I. Badler
University of Pennsylvania

Cary B. Phillips
PDI, Sunnyvale

Bonnie L. Webber
University of Pennsylvania

Chapter 1

Introduction and Historical Background

People are all around us. They inhabit our home, workplace, entertainment, and environment. Their presence and actions are noted or ignored, enjoyed or disdained, analyzed or prescribed. The very ubiquitousness of other people in our lives poses a tantalizing challenge to the computational modeler: people are at once the most common object of interest and yet the most structurally complex. Their everyday movements are amazingly fluid yet demanding to reproduce, with actions driven not just mechanically by muscles and bones but also cognitively by beliefs and intentions. Our motor systems manage to learn how to make us move without leaving us the burden or pleasure of knowing how we did it. Likewise we learn how to describe the actions and behaviors of others without consciously struggling with the processes of perception, recognition, and language.

A famous Computer Scientist, Alan Turing, once proposed a test to determine if a computational agent is intelligent [Tur63]. In the Turing Test, a subject communicates with two agents, one human and one computer, through a keyboard which effectively restricts interaction to language. The subject attempts to determine which agent is which by posing questions to both of them and guessing their identities based on the "intelligence" of their answers. No physical manifestation or image of either agent is allowed as the process seeks to establish abstract "intellectual behavior," thinking, and reasoning. Although the Turing Test has stood as the basis for computational intelligence since 1963, it clearly omits any potential to evaluate physical actions, behavior, or appearance.

Later, Edward Feigenbaum proposed a generalized definition that included action: "Intelligent action is an act or decision that is goal-oriented, arrived at by an understandable chain of symbolic analysis and reasoning steps, and is one in which knowledge of the world informs and guides the reasoning." [Bod77]. We can imagine an analogous "Turing Test" that would have the

subject watching the behaviors of two agents, one human and one synthetic, while trying to determine at a better than chance level which is which. Human movement enjoys a universality and complexity that would definitely challenge an animated figure in this test: if a computer-synthesized figure looks, moves, and acts like a real person, are we going to believe that it is real? On the surface the question almost seems silly, since we would rather not allow ourselves to be fooled. In fact, however, the question is moot though the premises are slightly different: cartoon characters are hardly "real," yet we watch them and properly interpret their actions and motions in the evolving context of a story. Moreover, they are not "realistic" in the physical sense – no one expects to see a manifest Mickey Mouse walking down the street. Nor do cartoons even move like people – they squash and stretch and perform all sorts of actions that we would never want to do. But somehow our perceptions often make these characters *believable*: they appear to act in a goal-directed way because their human animators have imbued them with physical "intelligence" and behaviors that apparently cause them to chase enemies, bounce off walls, and talk to one another. Of course, these ends are achieved by the skillful weaving of a story into the crafted images of a character. Perhaps surprisingly, the mechanisms by which motion, behavior, and emotion are encoded into cartoons is *not* by building synthetic models of little creatures with muscles and nerves. The requisite animator skills do not come easily; even in the cartoon world refinements to the art and technique took much work, time, and study [TJ81]. Creating such movements automatically in response to real-time interactive queries posed by the subject in our hypothetical experiment does not make the problem any easier. Even Turing, however, admitted that the intelligence sought in his original test did not require the computational *process* of thinking to be identical to that of the human: the external manifestation in a plausible and reasonable answer was all that mattered.

So why are we willing to assimilate the truly artificial reality of cartoons – characters created and moved entirely unlike "real" people – yet be skeptical of more human-like forms? This question holds the key to our physical Turing Test: as the appearance of a character becomes more human, our perceptual apparatus demands motion qualities and behaviors which sympathize with our expectations. As a cartoon character takes on a human form, the only currently viable method for accurate motion is the recording of a real actor and the tracing or transfer ("rotoscoping") of that motion into the animation. Needless to say, this is not particularly satisfying to the modeler: the motion and actor must exist prior to the synthesized result. Even if we recorded thousands of individual motions and retrieved them through some kind of indexed video, we would still lack the freshness, variability, and adaptability of humans to live, work, and play in an infinite variety of settings.

If synthetic human motion is to be produced without the benefit of prior "real" execution and still have a shot at passing the physical Turing Test, then models must carefully balance structure, shape, and motion in a compatible package. If the models are highly simplified or stylized, cartoons or caricatures will be the dominant perception; if they look like humans, then they will be

expected to behave like them. How to accomplish this without a real actor showing the way is the challenge addressed here.

Present technology can approach human appearance and motion through computer graphics modeling and three-dimensional animation, but there is considerable distance to go before purely synthesized figures trick our senses. A number of promising research routes can be explored and many are taking us a considerable way toward that ultimate goal. By properly delimiting the scope and application of human models, we can move forward, not to replace humans, but to substitute adequate computational surrogates in various situations otherwise unsafe, impossible, or too expensive for the real thing.

The goals we set in this study are realistic but no less ambitious than the physical Turing Test: we seek to build computational models of human-like figures which, though they may not trick our senses into believing they are alive, nonetheless manifest animacy and convincing behavior. Towards this end, we

- Create an interactive computer graphics human model.
- Endow it with reasonable biomechanical properties.
- Provide it with "human-like" behaviors.
- Use this simulated figure as an agent to effect changes in its world.
- Describe and guide its tasks through natural language instructions.

There are presently no perfect solutions to any of these problems, but significant advances have enabled the consideration of the suite of goals under uniform and consistent assumptions. Ultimately, we should be able to give our surrogate human directions that, in conjunction with suitable symbolic reasoning processes, make it appear to behave in a natural, appropriate, and intelligent fashion. Compromises will be essential, due to limits in computation, throughput of display hardware, and demands of real-time interaction, but our algorithms aim to balance the physical device constraints with carefully crafted models, general solutions, and thoughtful organization.

This study will tend to focus on one particularly well-motivated application for human models: human factors analysis. While not as exciting as motion picture characters, as personable as cartoons, or as skilled as Olympic athletes, there are justifiable uses to virtual human figures in this domain. Visualizing the appearance, capabilities and performance of humans is an important and demanding application (Plate 1). The lessons learned may be transferred to less critical and more entertaining uses of human-like models. From modeling realistic or at least reasonable body size and shape, through the control of the highly redundant body skeleton, to the simulation of plausible motions, human figures offer numerous computational problems and constraints. Building software for human factors applications serves a widespread, non-animator user population. In fact, it appears that such software has broader application since the features needed for analytic applications – such as multiple

simultaneous constraints – provide extremely useful features for the conventional animator. Our software design has tried to take into account a wide variety of physical problem-oriented tasks, rather than just offer a computer graphics and animation tool for the already skilled or computer-sophisticated animator.

The remainder of this chapter motivates the human factors environment and then traces some of the relevant history behind the simulation of human figures in this and other domains. It concludes with a discussion of the specific features a human modeling and animation system should have and why we have concentrated on some and not others. In particular, we are not considering cognitive problems such as perception or sensory interpretation, target tracking, object identification, or control feedback that might be important parts of some human factors analyses. Instead we concentrate on modeling a virtual human with reasonable biomechanical structure and form, as described in Chapter 2. In Chapter 4 we address the psychomotor behaviors manifested by such a figure and show how these behaviors may be interactively accessed and controlled. Chapter 5 presents several methods of motion control that bridge the gap between biomechanical capabilities and higher level tasks. Finally, in Chapter 6 we investigate the cognition requirements and strategies needed to have one of these computational agents follow natural language task instructions.

1.1 Why Make Human Figure Models?

Our research has focused on software to make the manipulation of a simulated human figure easy for a particular user population: human factors design engineers or ergonomics analysts. These people typically study, analyze, assess, and visualize human motor performance, fit, reach, view, and other physical tasks in a workplace environment. Traditionally, human factors engineers analyze the design of a prototype workplace by building a mock-up, using real subjects to perform sample tasks, and reporting observations about design satisfaction. This is limiting for several reasons. Jerry Duncan, a human factors engineer at Deere & Company, says that once a design has progressed to the stage at which there is sufficient information for a model builder to construct the mock-up, there is usually so much inertia to the design that radical changes are difficult to incorporate due to cost and time considerations. After a design goes into production, deficiencies are alleviated through specialized training, limits on physical characteristics of personnel, or various operator aids such as mirrors, markers, warning labels, etc. The goal of computer-simulated human factors analysis is not to replace the mock-up process altogether, but to incorporate the analysis into early design stages so that designers can eliminate a high proportion of fit and function problems before building the mock-ups. Considering human factors and other engineering and functional analyses together during rather than after the major design process is a hallmark of Concurrent Engineering [Hau89].

It is difficult to precisely characterize the types of problems a human factors engineer might address. Diverse situations demand empirical data on human capabilities and performance in generic as well as highly specific tasks. Here are some examples.

- Population studies can determine body sizes representative of some group, say NASA astronaut trainees, and this information can be used to determine if space vehicle work cells are adequately designed to fit the individuals expected to work there. Will all astronauts be able to fit through doors or hatches? How will changes in the workplace design affect the fit? Will there be unexpected obstructions to zero gravity locomotion? Where should foot- and hand-holds be located?

- An individual operating a vehicle such as a tractor will need to see the surrounding space to execute the task, avoid any obstructions, and insure safety of nearby people. What can the operator see from a particular vantage point? Can he control the vehicle while looking out the rear window? Can he see the blade in order to follow an excavation line?

- Specific lifting studies might be performed to determine back strain limits for a typical worker population. Is there room to perform a lift properly? What joints are receiving the most strain? Is there a better posture to minimize torques? How does placement of the weight and target affect performance? Is the worker going to suffer fatigue after a few iterations?

- Even more specialized experiments may be undertaken to evaluate the comfort and feel of a particular tool's hand grip. Is there sufficient room for a large hand? Is the grip too large for a small hand? Are all the controls reachable during the grip?

The answers to these and other questions will either verify that the design is adequate or point to possible changes and improvements early in the design process. But once again, the diversity of human body sizes coupled with the multiplier of human action and interaction with a myriad things in the environment leads to an explosion in possible situations, data, and tests.

Any desire to build a "complete" model of human behavior, even for the human factors domain, is surely a futile effort. The field is too broad, the literature immense, and the theory largely empirical. There appear to be two directions out of this dilemma. The first would be the construction of a computational database of all the known, or at least useful, data. Various efforts have been undertaken to assemble such material, for example, the NASA sourcebooks [NAS78, NAS87] and the *Engineering Data Compendium* [BKT86, BL88]. The other way is to build a sophisticated computational human model and use it as a subject in simulated virtual environment tests. The model will utilize an ever-expanding human factors data set to dictate its performance. Upon some reflection, it appears that database direction may

start out as the smoother road, but it quickly divides into numerous sinuous paths pot-holed with data gaps, empirical data collection limitations, and population-specific dependencies. The alternative direction (using a computational model underlying any data) may be harder to construct at first, and may have many detours for awhile, but gradually it leads to more destinations with better roads.

This metaphor carries a philosophy for animating human movement that derives from a computer science rather than an empirical point of view. We cannot do without the efforts of the human factors community, but we cannot use their work *per se* as the starting point for human figure modeling. Computer scientists seek computationally general yet efficient solutions to problems. Human factors engineers often analyze a succession of specific tasks or situations. The role we play is transforming the specific needs of the engineer or analyst into a generalized setting where some large percentage of situations may be successfully analyzed. There is sufficient research required to solve general yet difficult problems to justify building suitable software in a computer science environment. The expectation is that in the long run a more specific case-by-case implementation approach will be economically impractical or technologically infeasible.

As we continue to interact with human factors specialists, we have come to appreciate the broad range of problems they must address: fit, reach, visibility, comfort, access, strength, endurance, and fatigue, to mention only some of the *non-cognitive* ones. Our approach is not a denial of their perception and *analysis*, rather it is an alternative view of the problem as *modeling*. Broadly speaking, modeling is the embodiment within computer databases or programs of worldly phenomena. Models can be of many types:

- Mathematical formulations: physical equations of motion, limb strength in tables of empirical data, evaluation formulas measuring workload or fatigue.

- Geometric and topological models: structures representing workplace objects, human body segments, paths to follow, joints and joint limits, spaces that can be reached, attachments, and constraints.

- Conceptual models: names for things, attributes such as color, flexibility, and material, relationships between objects, functional properties.

Of course, modeling (especially of the first sort) is a significant and fundamental part of many studies in the human factors domain, but it has been difficult to balance the needs of the engineer against the complexity of the modeling software. Often, the model is elaborated in only a few dimensions to study some problem while no global integration of models is attempted. Clearly the broadest interpretation of modeling draws not only from many areas of computer science such as artificial intelligence, computer graphics, simulation, and robotics, but also from the inherently relevant fields of biomechanics, anthropometry, physiology, and ergonomics.

The challenge to embed a reasonable set of capabilities in an integrated system has provided dramatic incentives to study issues and solutions in three-dimensional interaction methodologies, multiple goal positioning, visual field assessment, reach space generation, and strength guided motion, to name a few. The empirical data behind these processes is either determined from reliable published reports or supplied by the system user. By leaving the actual data open to the user, the results are as valid as the user wishes to believe. While this is not a very pleasant situation, the inherent variability in human capability data makes some error unavoidable. Better, we think, to let the user know or control the source data than to hide it. This attitude toward validation is not the only plausible one, but it does permit flexibility and generality for the computer and allows final judgment to be vested in the user.

Lest there be concern that we have pared the problem down so far that little of interest remains, we change tactics for awhile and present an historical view of efforts to model humans and their movements. By doing so, we should demonstrate that the human factors domain mirrors problems which arise in other contexts such as dance, sports, or gestural communication. The criteria for success in these fields may be more stringent, so understanding the role and scope of human movement in them can only serve to strengthen our understanding of more mundane actions.

1.2 Historical Roots

Interactive computer graphics systems to support human figure modeling, manipulation, and animation have existed since the early seventies. We trace relevant developments with a sense more of history and evolution rather than of exhaustive survey. There are numerous side branches that lead to interesting topics, but we will sketch only a few of those here.

Three-dimensional human figure models apparently arose independently from at least six different applications.

1. Crash simulation. Automobile and aircraft safety issues led to the development of sophisticated codes for linked mass deceleration studies. These programs generally ran in batch mode for long hours on mainframe computers. The results were tabulated, and in some cases converted to a form that could animate a simple 3D mannequin model [Fet82, Wil82, BOT79]. The application was characterized by non-interactive positioning and force-based motion computations with analysis of impact forces to affected body regions and subsequent injury assessment.

2. Motion analysis. Athletes, patients with psychomotor disabilities, actors, or animals were photographed in motion by one or more fixed, calibrated cameras. The two-dimensional information was correlated between views and reconstructed as timed 3D data points [RA90]. This

data could be filtered and differentiated to compute velocities, accelerations, torques and forces. Visualization of the original data validated the data collection process, but required human figure models. Often just wire-frames, they served in a support role for athletic performance improvement, biomechanical analysis, cartoon motion [TJ81], and training or physical therapy [Win90]. Related efforts substituted direct or active sensing devices for photographic processing [CCP80]. Presently, active motion sensing is used not only for performance analysis but gestural input for virtual environments (for example, [FMHR87, BBH+90] and many others).

3. Workplace assessment. The earliest system with widespread use was SAMMIE [KSC81]. This problem domain is characterized by interactive body positioning requirements and analyses based on visual inspection of 3D computer graphics models. Fast interaction with wire-frame displays provided dynamic feedback to the workplace evaluator. Other modeling tools were developed, such as CAR II [HBD80], Combiman [BEK+81], and Crew Chief [MKK+88, EI91] to provide validated anthropometric or capability data for real populations.

4. Dance or movement notation. The specification of self-generated, purposive, aesthetically-pleasing, human movement has been the subject of numerous notational systems [Hut84]. Dance notations were considered as a viable, compact, computationally tractable mode of expression for human movement due to their refinement as symbolic motion descriptions [BS79]. An animation was to be the debugging tool to validate the correctness of a given notated score. The direct creation of movement through a notational or numeric interface was also considered [CCP80, CCP82, HE78].

5. Entertainment. People (or at least animate creatures) are the favorite subject of cartoons and movies. Two-dimensional animation techniques were the most widely used [Cat72, BW76, Lev77, Cat78]. In an effort to avoid rotoscoping live actors, early 3D modeling and animation techniques were developed at the University of Utah, Ohio State University, and the New York Institute of Technology [Wes73, Hac77, Stu84, Gom84, HS85a].

6. Motion understanding. There are deep connections between human motion and natural language. One of these attempted to produce a sort of narration of observed (synthetic) movement by characterizing changes in spatial location or orientation descriptions over time [Bad75]. More recently, the inverse direction has been more challenging, namely, producing motion from natural language descriptions or instructions [BWKE91, TST87].

Our earliest efforts were directed at language descriptions of object motion. Specifically, we created representations of 3D object motion and directional

adverbials in such a way that image sequences of moving objects could be analyzed to produce English sentence motion descriptions [Bad75, Bad76]. We then extended the model to articulated figures, concentrating on graphically valid human figure models to aid the image understanding process [BOT79]. This effort led to the work of Joseph O'Rourke, who attempted model-driven analysis of human motion [OB80] using novel 3D constraint propagation and goal-directed image understanding.

To improve the motion understanding component we focused on motion representations specially designed for human movement [WSB78, BS79]. An in-depth study of several human movement notation systems (such as Labanotation [Hut70] and Eshkol-Wachmann [Hut84]) fostered our appreciation for the breadth and complexity of human activities. Our early attempts to re-formulate Labanotation in computational models reflected a need to cover at least the space of human (skeletal) motion. We investigated input systems for Labanotation [BS76, Hir77], although later they were discarded as a generally accessible means of conveying human movement information from animator to computer figure: there was simply too much overhead in learning the nuances and symbology of the notational system. Moreover, concurrent developments in three-dimensional interactive computer graphics offered more natural position and motion specification alternatives. The final blow to using Labanotation was its lack of dynamic information other than timing and crude "accent" and phrasing marks.

The movement representations that we developed from Labanotation, however, retained one critically important feature: goal-directedness for efficient motion specification. Given goals, processes had to be developed to satisfy them. A simulation paradigm was adopted and some of the special problems of human movement simulation were investigated [BSOW78, BOK80, KB82]. Others studied locomotion [CCP82, Zel82, GM85, Gir87, Bru88, BC89], while we concentrated on inverse kinematics for reach goals [KB82, Kor85]. We were especially anxious to manage multiple reach and motion goals that mutually affected many parts of the body. Solving this problem in particular led to later re-examination of constraint-based positioning and more general and robust algorithms to achieve multiple simultaneous goals [BMW87, ZB89]. Chapter 4 will discuss our current approach in detail.

Our study of movement notations also led to an appreciation of certain fundamental limitations most of them possessed: they were good at describing the changes or end results of a movement (*what* should be done), but were coarse or even non-specific when it came to indicating *how* a movement ought to be performed. The notator's justification was that the performer (for example, a dancer) was an *expert system* who knew from experience and training just how to do the notated motion. The transformation from notation into smooth, natural, expressive movements was part of the art. The exception to the nearly universal failure of notational systems to capture nuances of behavior was Effort-Shape notation [Del70, BwDL80]. We began a study of possible computational analogues to the purely descriptive semantics of that system. By 1986 a model of human movement emerged which integrated the

kinematic and inverse kinematic approach with a dynamic, force-based model [Bad89]. Major contributions to dynamics-based animation were made by others, notably [AG85, AGL87, WB85, Wil86, Wil87, IC87, HH87, Hah88]. Recently we combined some of the characteristics of the dynamics approach – the use of physical torques at the body joints – with goal-directed behavior to achieve *strength guided motion* ([LWZB90] and Chapter 5).

While we were actively engaged in the study of motion representations, concurrent developments in interactive systems for the graphical manipulation of a computerized figure were being actively implemented at the University of Pennsylvania. Implementation and development has been a strong experimental component of our research, from the early positioning language based on Labanotation concepts [WSB78], to the next generation frame buffer-based system called *TEMPUS* [Kor85, BKK+85], to the direct manipulation of the figure with a 6-axis digitizer [BMB86], and finally to our present Silicon Graphics workstation-based system *Jack*™[1] [PB88, PZB90] (Chapters 2 and 4).

As our experience with interactive graphical manipulation of a figure matured, we returned to the connections between language and motion we had begun in the mid-1970's. The manipulation of the figure for task analysis begged for more efficient means of specifying the task. So we began to investigate natural language control for task animation [Gan85, BG86, Kar87, Kar88, Kal90, KB90, KB91]. New representations for motion verbs and techniques for defining and especially *executing* their semantics were investigated. A simple domain of panel-type objects and their motions were studied by Jeff Gangel. Robin Karlin extended the semantics to certain temporal adverbials (such as repetitions and culminations) in a domain of kitchen-objects. Our present effort is exemplified here by the work of Jugal Kalita and Libby Levison. Kalita studied verbs of physical manipulation and used constraints in a fundamental fashion to determine generalized verb semantics. Levison makes explicit connections between a verb's semantic representation and the sorts of primitive behaviors and constraints known to be directly simulatable by the *Jack* animation system (Chapter 6).

Given that natural language or some other artificial language was to be used to describe tasks or processes, a suitable simulation methodology had to be adopted. In his HIRES system, Paul Fishwick investigated task and process simulation for human animation [Fis86, Fis88]. Output was produced by selecting from among pre-defined key postures. For example, an animation of the famous "Dining Philosophers" problem using five human figure models was produced by simulation of the petri net solution in the HIRES simulator. By 1989 we replaced HIRES by a new simulation system, YAPS, which incorporated temporal planning with imprecise specifications [KKB88], task interruption, and task time estimation based on human performance models [EBJ89, EB90, BWKE91] (Chapter 6).

This brings us to the present *Jack* system structure designed to accommo-

[1] *Jack* is a registered trademark of the University of Pennsylvania.

date as many of the historical applications as possible within an integrated and consistent software foundation. To begin to describe that, we need to review what human modeling capabilities are needed and what problem implementation choices we might make.

1.3 What is Currently Possible?

Since we wish to animate synthetic human figures primarily in the human factors engineering domain, we should decide what features are essential, desirable, optional, or unnecessary. Only by prioritizing the effort can such a large-scale undertaking be managed. Given priorities, implementation techniques and trade-offs may be investigated. Though we may often draw on existing knowledge and algorithms, there are many fundamental features which we may have to invent or evolve due to the specific structure of the human figure, characteristics of human behavior, timing demands of real-time interaction, or limitations of the display hardware. Accordingly, a variety of human figure modeling issues will be examined here to introduce and justify the choices we have made in our broad yet integrated effort.

The embodiment of our choices for human modeling is a software system called *Jack*. Designed to run on Silicon Graphics 4D workstations, *Jack* is used for the definition, manipulation, animation, and human factors performance analysis of virtual human figures. Built on a powerful representation for articulated figures, *Jack* offers the interactive user a simple, intuitive, and yet extremely capable interface into any three dimensional world. *Jack* incorporates sophisticated yet highly usable algorithms for anthropometric human figure generation, a flexible torso, multiple limb positioning under constraints, view assessment, reach space generation, and strength guided performance simulation of human figures. Of particular importance is a simulation level which allows access to *Jack* by high level task control, various knowledge bases, task definitions and natural language instructions. Thus human activities can be visualized from high level task understanding and planning as well as by interactive specification.

One can think of *Jack* as an experimental environment in which a number of useful general variables may be readily created, adjusted, or controlled: the workplace, the task, the human agent(s) and some responses of the workplace to internally or externally controlled actions. The results of specific instances of these input parameters are reported through computer graphics displays, textual information, and animations. Thus the field of view of a 50^{th} percentile male while leaning over backwards in a tractor seat as far a possible may be directly visualized through the graphic display. If the figure is supposed to watch the corner of the bulldozer blade as it moves through its allowed motion, the human figure's gaze will follow in direct animation of the view.

In the following subsections, several desiderata are presented for human models. Under each, we summarize the major features – with justifications and benefits – of the *Jack* software. The detailed discussions of these features

and their implementation constitute the bulk of the remaining chapters.

1.3.1 A Human Model must be Structured Like the Human Skeletal System

To build a biomechanically reasonable figure, the skeletal structure should resemble but need not copy that of humans. We can buffer the complexity of actual bone shapes, joint types and joint contact surfaces with requirements for interactive use and external motion approximations. For example, rotational joints are usually assumed to have a virtual center about which the adjacent body segments move. While such simplifications would not be appropriate for, say, knee prosthesis design, there appears to be little harm in variations on the order of a centimeter or so. Of course, there are situations where small departures from reality could affect the verisimilitude of the figure; accordingly we have concentrated on rather accurate models for the torso and shoulder complex. Many other software systems (or manual methods) presume a fixed shoulder joint but it is obvious that this is not true as the arm is elevated.

1. *Jack* has a fully linked body model including a 17 segment flexible torso with vertebral joint limits. In general, individual joints may have one, two, or three degrees of freedom (DOFs). Related groups of joints, such as the spine or the shoulder complex, may be manipulated as a unit. The result is reasonable biomechanical realism with only modest computational overhead.

2. The *Jack* shoulder mass joint center is posture-dependent. Accurate shoulder motion is modeled through an explicit dependency between arm position and clavicle rotation. The figure therefore presents appropriate shoulder and clavicle motions during positioning. The shoulder joint has spherical (globographic [EP87]) limits for improved motion range accuracy.

3. All joint rotations are subject to limits. During manipulation, rotations propagate when joint limits would be exceeded.

4. A fully articulated hand model is attached.

5. The foot is articulated enough to provide toe and heel flexibility. If more DOFs were required, they could be easily added.

1.3.2 A Human Model should Move or Respond Like a Human

Ideally, the motions presented by a simulated figure will be biomechanically valid. They should not only appear "human-like," but they should be validated against empirical data for real subjects under similar conditions. This

goal is desirable but difficult to reach in a generalized motion model precisely because such a model must allow interpolation and extrapolation to situations other than those originally measured. Models are needed to provide reasonable interpretations of data that by necessity must be sampled rather coarsely, in specific situations, and with a collection of specific subjects. The closest we can get to the ideal is to provide generic mechanisms that incorporate whenever possible empirical data that a user believes to be valid up to the degree of error permitted for the task. Rather than hide such data, *Jack* takes the view of an open database where reasonable default human anthropometric, strength or performance data is provided, but user customizing is the rule rather than the exception.

1. *Jack* permits multiple figures to simultaneously inhabit an environment. Multi-person environments and operator interactions may be studied for interference, view, and coordinated tasks.

2. A number of active behaviors are defined for the figure and may be selected or disabled by the user. Among the most interesting is constraining the center of mass of the entire figure during manipulation. This allows automatic balance and weight-shifting while other tasks such as reaching, viewing, bending over, etc. are being performed. Spare cycles on the workstation are used inbetween human operator inputs to constantly monitor the active constraints and move the body joints towards their satisfaction.

3. People usually move in ways that conserve resources (except when they are deliberately trying to achieve optimum performance). If strength information as a function of body posture and joint position is available, that data may be used to predict certain end-effector motion paths under specified "comfort" conditions. Thus the exact motions involved in, say, lifting a weight are subservient to the strength model, comfort and fatigue parameters, and heuristics for selecting among various movement strategies. By avoiding "canned" or arbitrary (for example, straight line) motion paths, great flexibility in executing tasks is provided.

4. The *Jack* hand model has an automatic grip. This feature saves the user from the independent manipulation of large numbers of joints and DOFs. The user specifies a grip type and an optional site on the object to be grasped. When possible, the hand itself chooses a suitable approach direction. Though frictional forces are not modeled, the positioning task is greatly aided by the hand's skill. Once gripped, the object stays attached to the hand and moves along with it until explicitly freed by the user.

1.3.3 A Human Model should be Sized According to Permissible Human Dimensions

Effectively, there is no such thing as a "average" human. Statistically one must always prescribe a target population when talking about percentiles of size, weight, or stature. A person may be 50^{th} percentile in stature, 75^{th} percentile in weight, but 40^{th} percentile in lower leg length. Human dimensional variability is enormous but not arbitrary. Within a given population, for example, 5^{th} percentile legs might never be found on anybody with 95^{th} percentile arms, even though the population allows such sizes individually. Moreover, dimensions are not just limited to lengths, stature, and weight, but include joint limits, moments of inertia for each body segment, muscle strengths, fatigue rates, and so on. For proper behaviors, one must be able to instantiate a properly sized figure with appropriately scaled attributes, preferably from a known population suitable for the required task analysis.

1. The *Jack* anthropometric database is not proprietary. All data is readily available and accessible. Consequently, it is easily customized to new populations or sets of individuals. Some databases are available, such as NASA astronaut trainees, Army soldiers, and Society of Automotive Engineers "standard people."

2. A database may consist of either population statistics or individuals. If populations, then percentile data points are expected in order to define body dimensions. If individuals, then explicit information for each person in the collection is separately stored. For example, the NASA astronaut trainees constitute an explicit list of individuals, while the Army soldier data is statistically derived.

3. Enough information about a human figure must be stored to permit body sizing, display, and motion. We use overall segment dimensions (length, width, thickness), joint limits, mass, moment of inertia, and strength. Geometric properties are used during graphical manipulation; physical properties aid active behaviors (balance), dynamic simulation, and strength guided motion.

4. With so many DOFs in a body and many useful physical attributes, there must be a convenient way of accessing, selecting, and modifying any data. *Jack* uses a spreadsheet-like interface to access anthropometry data. This paradigm permits simple access and data interdependencies: for example, changing leg length should change stature; changing population percentile should change mass distribution.

5. Seeing the results of changing body dimensions is important to understanding how different bodies fit, reach, and see in the same workplace. *Jack* allows interactive body sizing while the body itself is under active constraints. Changing a figure seated in a cockpit from 95^{th} percentile to

5^{th} percentile, for example, creates interesting changes in foot position, arm postures and view.

1.3.4 A Human Model should have a Human-Like Appearance

As we argued earlier, a human model's appearance has a lot to do with our perception of acceptable behavior. The more accurate the model, the better the motions ought to be. Providing a selection of body models is a convenient way to handle a spectrum of interactive analysis and animation requirements. For quick assessments, a human model with simplified appearance might be fine. If the skin surface is not totally realistic, the designer can move the view around to check for sufficient clearances, for example. When the completed analysis is shown to the boss, however, an accurate skin model might be used so that the robotic nature of the simpler model does not obscure the message. Looking better is often associated (in computer graphics) with being better.

1. The "standard" or default body model in *Jack* strikes a balance between detail and interactive manipulation speed. It appears solid, has a 17 segment flexible torso, has a reasonable shoulder/clavicle mass, has a full hand, and has a generic face. A hat is added to avoid modeling hairstyles. The figure is modeled by surface polygons to take advantage of the available workstation display capabilities.

2. There are times when more accurate skin surface models are needed. For that, computerized models of real people are used. These are derived from a database of biostereometrically-scanned bodies.

3. For extra realism, clothing is added to body segments by expanding and coloring the existing segment geometry. Besides the inherent desirability of having a virtual figure in a work environment appear to be dressed, clothing will also affect task performance if adjustments are made to joint limits or if collision tests are performed.

4. Facial features may be provided by graphical texture maps. By showing a specific face a particular individual may be installed in the scene, the figure may be uniquely identified throughout an animation, or generic gender may be conveyed. Moreover, if the face is animated, an entire communication channel is enabled.

1.3.5 A Human Model must Exist, Work, Act and React Within a 3D Virtual Environment

We do not live in Flatland [Abb53] and neither should virtual figures. Three-dimensional environment modeling is the hallmark of contemporary computer-aided design (CAD) systems. The workplace will frequently be constructed electronically for design, analysis, and manufacturing reasons; we merely add

human factors analysis to the list. Design importation facilitates up-front analyses before commitment to production hardware. Since geometric models are readily constructed, we must be sure that our body models and interactive software are compatible with the vast majority of CAD modeling schemes so the two can work together in the same virtual space.

1. Since *Jack* manipulates surface polygon geometry, simple features are provided to interactively construct and edit the geometry of the virtual workplace. *Jack* is not intended as a substitute for a good CAD system, but CAD features are provided for convenience. For example, if a designer finds a problem with the imported workplace, the offending region can be immediately edited in *Jack*. When the changes prove acceptable, the designer can use the data values from *Jack* to modify the "real" model in the external CAD system. While not optimal, this avoids any tendency to migrate the working model into *Jack* and bypass the more generalized CAD features provided by most CAD systems.

2. Standardized geometric data transfer to and from *Jack* would be highly desirable, but the state of standards in geometric modeling still leaves some important gaps. For example, few CAD systems adequately model articulated objects. Currently, *Jack* imports models from several common CAD vendors through a less satisfactory scheme of explicit translators from external geometry files. This situation will change to a standard as soon as possible.

1.3.6 Use the Computer to Analyze Synthetic Behaviors

What would a real person do in a real environment? How can we get a virtual human to behave appropriately and report similar experiences? People are constantly managing multiple simultaneous constraints or tasks, for example, staying balanced while reaching to lift a box, or walking while carrying a cup of coffee. The essential parallelism of human motion demands an approach to behavior animation that does not just cope with parallelism but exploits it. We will be interested mostly in psychomotor and viewing behaviors; clearly auditory and cognitive tasks are worthy of inclusion but are not dealt with here.

1. *Jack* allows the user to specify, and the system maintains, multiple simultaneous position and orientation goals. For instance, a typical posture might involve reaching with two hands while looking at a target and staying balanced. There are constraints on the positions of the hands and the orientation of the eyes dictated by the task, and balance constraints required by gravity. Additionally, we might want the torso to remain upright or, if seated, for the pelvis to tilt to a more relaxed posture. There are too many possible human body configurations to manage every combination by specialized rules; our approach is to

use a global solution technique, *inverse kinematics*, to satisfy the given constraints subject to the inherent joint limits of the body.

2. While in a posture, the strength requirements of the figure may be displayed on screen. This interactive strength data display shows all torque loads along any selected chain of body segments. If a load is attached to an end-effector, for example, it permits the easy visualization of the distribution of that additional weight on the body.

3. What the figure is looking at is often a critical question in human factors analysis. In *Jack*, the direction of eye gaze is controlled through constraints to some environmental location or object site. If that site moves, the gaze will follow. Since eye movement will affect head orientation, the effect of gaze direction can propagate (because of joint limits) to the neck and torso and hence influence overall body posture.

4. In *Jack*, the user can see what the figure sees from an internal or external perspective. Internally, a separate graphics window may be opened which shows the view from the selected eye. The image appears naturally shaded and it moves as the figure's gaze is adjusted by direct manipulation or constraints. Concurrent changes to the environment and visible parts of the figure's own "self" are displayed in the view window. If field of view is critical, a *retinal projection* may be used where a polar projection displays workplace features based on their angle from the fovea. Although the image thereby appears distorted, the actual field of view may be superimposed to assess the range of foveal or peripheral perception. For the external perspective, a selected field of view is displayed as a translucent pair of view cones, one for each eye. The cones move with the eyes. Objects in view are shadowed by the translucent cones. Any overlapped region is clearly in the area of binocular vision.

5. It is not yet feasible to do real-time collision detection between all the moving objects in a complex environment. By various simplifications, however, sufficient capabilities may be presented. On fast displays with real-time viewing rotation (such as the Silicon Graphics workstations), the ability to rapidly change the view means that the user can quickly move about to check clearances. Another method used in *Jack* is to optionally project three orthogonal views of a figure and other selected objects onto back, side, and bottom planes. These three additional views give simultaneous contact and interference information and are used during interactive manipulations. Often, users will work with shaded images and detect collisions by noting when one object visually passes into another. The alternative to visual collision detection is direct object–object interference computation. Usually limited to a selected body segment and a convex object, this method is slower but guarantees to detect a collision even if it would be difficult to see.

6. Sometimes it is valuable to visualize the entire reachable space of an end-effector of the figure. Empirical data is sometimes available in the literature [NAS78, NAS87], but it is collected on specific subjects and does not readily extend to figures with differing anthropometry or joint limits. By constructing the reach space for a given figure in a given posture as a geometric object, the reach space may be viewed and objects may be readily classified as in or out of the reach space.

1.3.7 An Interactive Software Tool must be Designed for Usability

A system to build, move, and analyze virtual humans should be usable by mere mortals with a modest training period. Isolating the potential user community by requiring unusual artistic skills would be counter-productive to our wider purposes of aiding design engineers. Existing interaction paradigms (such as pop-up menus or command line completions) should be followed when they are the most efficacious for a particular task, but new techniques will be needed to manage and control three-dimensional articulated structures with standard graphical input tools. The interface should be simple yet powerful, comprehensive but easy to learn.

1. The *Jack* user interface is designed for fast response to multiple constraint situations. Real-time end-effector interactive dragging through arbitrary length joint chains means that the user can watch the figure respond to reach or movement tasks. The paradigm of manipulating one joint angle at a time is possible, but almost useless. The goal-directed behaviors provide an enormous benefit to the user in allowing the specification of *what* is to be done while constraint satisfaction handles the *how* through positioning interdependencies of the entire body structure. Dragging also permits quick experimentation with and manual optimization of postures.

2. By taking advantage of the Silicon Graphics display hardware, *Jack* shows the user shaded or wireframe displays during interaction for natural images and easy real-time visualization.

3. For the highest quality images, *Jack* provides its own multi-featured ray-tracing and radiosity programs.

4. Besides direct user input, *Jack* may be controlled through scripted commands (in the *Jack command language*) built in the course of interactive manipulation. This saves time and trouble in setting up complex situations, establishing a body posture, or trying a series of actions.

5. *Jack* also allows external control through operating system "sockets" to other programs, simulations, or real sensors, providing hooks into external data sources for virtual environments or networked remote systems

sharing a common virtual dataspace. *Jack* itself can share an environment with other *Jack*'s on the network. This could be used for novel cooperative workgroup applications.

6. The standard *Jack* user interface consists of just a three button mouse and keyboard. It is simple to learn and no special hardware devices are required unless a virtual environment setup is desired. The interaction paradigms in *Jack* include menu-driven or typed commands, on-line help, and command completion. It is easy to use after only a day or so of training.

7. The direct manipulation interface into three dimensions implemented in *Jack* is both highly efficient and natural to use. Depending only on the mouse and keyboard, it offers a friendly, kinesthetic correspondence between manually comfortable hand motions, on-screen displays, and three-dimensional consequences. Three dimensional cursors, rotation wheels, and orthogonal projections of the principal view provide excellent visual feedback to the user.

8. *Jack* provides multiple windows with independent camera views for complex analyses, multiple points of view, and internal and external eye views.

1.4 Manipulation, Animation, and Simulation

There are important distinctions between *manipulation*, *animation*, and *simulation*. Geometric manipulation is the process of interactive scene composition, or the interactive specification of positions and postures for geometric figures, usually on a trial and error basis. Manipulation usually involves movement of the figures, but the movement serves to assist in the control process and is generally not worth saving as a memorable motion sequence. Rather, the purpose of manipulation is to get the figures into a desired static posture, although the posture need not remain static afterwards. Manipulation is inherently real-time: objects move as a direct response to the actions of the user. In short, interactive manipulation is not necessarily choreography.

Animation, on the other hand, is choreography. In computer animation, the goal is to describe motion, and the animator usually imagines the desired motion before beginning the animation process. Of course, experimentation may lead to revisions, like an illustrator who erases lines in a drawing, but the computer does not serve so much to answer questions as to obey orders. Animators measure the success of a computer animation system in terms of how well it serves as a medium for expressing ideas.

Simulation is automated animation, and the concern is again with motion. The system generates the motion based on some kind of input from the user ahead of time. The input usually consists of objectives and rules for making decisions, and it is generally less specific than with animation. The user knows

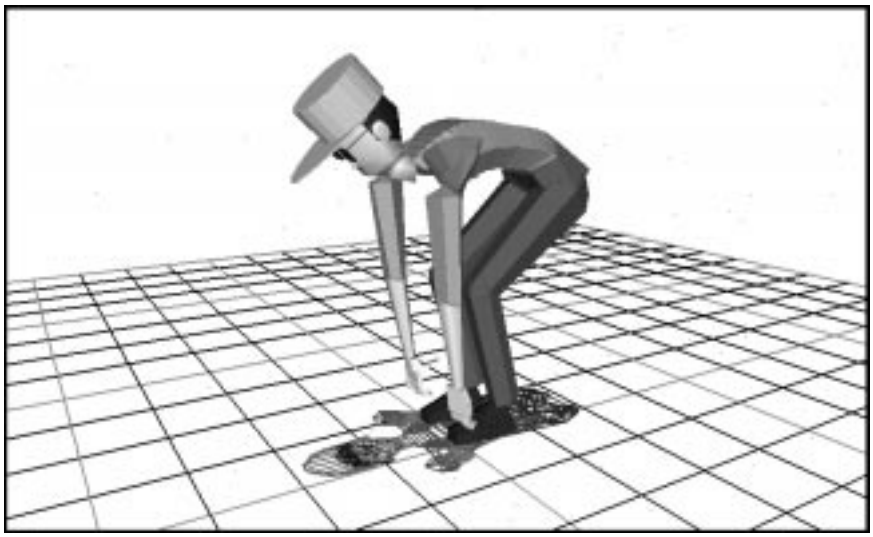

Figure 1.1: Is it the Motion or the Posture?

less about what motion should result. The job of the simulator is to predict what would happen under certain circumstances and inform the user of the results. Sometimes simulation can generate animation, as in the case of the animation of physics and natural phenomena. Simulation of human figures generally implies some modeling of human capabilities to deliberately off-load some of the low-level positioning overhead from the animator.

Animation and simulation have been studied extensively, but manipulation of articulated figures has not received the attention it deserves. Volumes of research discuss animation techniques and simulation algorithms, but most research directed at interactive manipulation deals either with the low-level input mechanisms of describing 3D translations and rotations, or with the numerical issues of real-time dynamics. For example, consider the task of bending a figure over to touch its toes (Fig. 1.1. Is the bending motion important, or is it just the final posture that is critical? In animation, it's the motion: the motion must look realistic. In simulation, the motion must *be* realistic. In manipulation, the finer points of the posture are critical. Is the figure balanced? Are the knees bent? Where is the head pointed? How are the feet oriented? The motion through which the manipulation system positions the figure is not important in itself. It serves only to assist the user in arriving at the posture.

1.5 What Did We Leave Out?

It is only fair that in this exposition we are clear about what our existing software does not do. There are choices to be made in any implementation, but

the vastness of the human performance problem demands scope boundaries as well. There are fascinating problems remaining that we have not touched. Some of these problems are being examined now, but the early results are too premature to report. The activity in this field is amazing, and there will surely be advances in modeling, animation, and performance simulation reported each year.

A glance at the *Engineering Data Compendium* [BL88], for example, will quickly show how much information has been collected on human factors and simultaneously how little is available interactively on a computer. But even without much thought, we can place some bounds on this study.

- We have ignored auditory information processing, environmental factors (such as temperature and humidity), vibration sensitivity, and so on. While critical for harsh environments, we seek useful approximations to first order (geometric) problems to see if a figure can do the task in the absence of external signals, distress, or threats. The probable degradation of task performance may be established from intelligent (manual) search through existing publications. While it would be attractive to include such data, we have not begun to acquire or use it yet.

- We have bypassed super-accurate skin models because the enfleshment of a figure is so dependent on individual physiology, muscle tone, muscle/fat ratio, and gender. For the kinds of analyses done with whole body models, errors of a centimeter or so in skin surfaces are subordinate to anthropometric errors in locating true joint centers and joint geometry.

- We have avoided injury assessment, as that is another whole field developed from anatomy, crash studies, or national exertion standards. Of course, *Jack* could be used in conjunction with such systems, but we have not tried to connect them yet.

- Because we have taken a generalized view of human factors and restricted analyses to reach, fit, view, and strength, we have necessarily avoided any workplace-specific performance data. For example, action timings will be most accurate when measured with real subjects in a real environment. We have concentrated on measurements in environments prior to physical construction to save cost and possible personnel dangers.

- We have only touched on perceptual, reactive and cognitive issues. Many workers are engaged in control-theoretic activities: sensing the environment and reacting to maintain some desired state. We see this capability eventually driving the figure through the simulation interface, but at the present we have not modeled such situations. Instead we are investigating natural language instructions which generate an animation of the simulated agent's "artificial intelligence" understanding of the situation.

- Finally, we are leaving issues of learning for later study. There is much to be said for an agent who not only follows instructions but who also learns how to do similar (but not identical) actions in comparable situations in the future. Learning might first occur at the psychomotor level, for example, to figure out the best way to lift a heavy weight. Later, learning can extend to the task level: to repeat a variant of a previously successful plan when the overall goals are encountered again.

These issues are exciting, but we need to start with the basics and describe what exists today.

Chapter 2

Body Modeling

In order to manipulate and animate a human figure with computer graphics, a suitable figure must be *modeled*. This entails constructing a satisfactory surface skin for the overall human body shape, defining a skeletal structure which admits proper joint motions, adding clothing to improve the verisimilitude of analyses (as well as providing an appropriate measure of modesty), sizing body dimensions according to some target individual or population, and providing visualization tools to show physically-relevant body attributes such as torque loads and strength.

2.1 Geometric Body Modeling

In computer graphics, the designer gets a wide choice of representations for the surfaces or volumes of objects. We will briefly review current geometric modeling schemes with an emphasis on their relevance to human figures.

We classify geometric models into two broad categories: boundary schemes and volumetric schemes. In a boundary representation the surface of the object is approximated by or partitioned into (non-overlapping) 0-, 1-, or 2-dimensional primitives. We will examine in turn those representations relevant to human modeling: points and lines, polygons, and curved surface patches. In a volumetric representation the 3D volume of the object is decomposed into (possibly overlapping) primitive volumes. Under volumetric schemes we discuss voxels, constructive solid geometry, ellipsoids, cylinders, spheres, and potential functions.

2.1.1 Surface and Boundary Models

The simplest surface model is just a collection of 3D points or lines. Surfaces represented by points require a fairly dense distribution of points for accurate modeling. Clouds of points with depth shading were used until the early 1980's for human models on vector graphics displays. They took advantage of

the display's speed and hierarchical transformations to produce the perceptual depth effect triggered by moving points [Joh76] (for example, [GM86]).

A related technique to retain display speed while offering more shape information is to use parallel rings or strips of points. This technique is used in *LifeForms*™[1] [Lif91, Cal91]. Artistically positioned "sketch lines" were used in one of the earliest human figure models [Fet82] and subsequently in a Mick Jagger music video, "Hard Woman" from Digital Productions.

Polygons

Polygonal (polyhedral) models are one of the most commonly encountered representations in computer graphics. The models are defined as networks of polygons forming 3D polyhedra. Each polygon (primitive) consists of some connected vertex, edge, and face structure. The polygons are sized, shaped, and positioned so that they completely tile the required surface at some resolution. Polygon models are relatively simple to define, manipulate, and display. They are the most common models processed by workstation hardware and commercial graphics software. In general, polygons are best at modeling objects meant to have flat surfaces, though with a large enough number of polygons quite intricate and complex objects can be represented. "Large enough" may mean hundreds of thousands of polygons!

All viable *interactive* human figure models are done with polygons, primarily because the polygon is the primitive easiest to manage for a modern workstation such as the Silicon Graphics machines. With real-time smooth shading, polygon models of moderate complexity (several hundred polygons) can look acceptably human-like; accurate skin models require thousands. A realistic face alone may require two or three thousand polygons if it is to be animated and if the polygons are the only source of detail.

Polygon models are too numerous to cite extensively. Ones with an interesting level of detail have been used in Mannequin software from Biomechanics Corporation of America [Pot91], various movies such as "Tony de Peltrie" from the University of Montreal [Emm85], and detailed synthetic actor models of Marilyn Monroe and Humphrey Bogart from Daniel Thalmann and Nadia Magnenat-Thalmann [MTT90, MTT91b]. They even model details of skin deformation by applying physical forces to polygon meshs [GMTT89, MTT91b].

The polygon models used in *Jack* are polygonal with two different levels of detail. The normal models have a few hundred polygons. More accurate models obtained from actual scans of real bodies have several thousand polygons. (See Section 2.1.3.)

Curved Surfaces

Since polygons are good at representing flat surfaces, considerable effort has been expended determining mathematical formulations for true curved surfaces. Most curved surface object models are formed by one or more para-

[1] LifeForms is a registered trademark of Kinetic Effects, Inc.

metric functions of two variables (bivariate functions). Each curved surface is called a patch; patches may be joined along their boundary edges into more complex surfaces. Usually patches are defined by low order polynomials (typically cubics) giving the patch easily computed mathematical properties such as well-defined surface normals and tangents, and computable continuity conditions between edge-adjacent patches. The shape of a patch is derived from control points or tangent vectors; there are both approximating and interpolating types. The former take the approximate shape of the control vertices; the latter must pass through them. There are numerous formulations of curved surfaces, including: Bezier, Hermite, bi-cubic, B-spline, Beta-spline, and rational polynomial [Far88, BBB87].

Various human figure models have been constructed from curved patches, but display algorithm constraints make these figures awkward for real-time manipulation. They are excellent for animation. provided that sufficient care is taken to model joint connections. This is a good example of where increased realism in the body segments demands additional effort in smoothing and bending joint areas properly. Curved surface models were used in a gymnastic piece [NHK86] and the Academy Award-winning "Tin Toy" [GP88].

2.1.2 Volume and CSG Models

The volume and CSG models divide the world into three-dimensional chunks. The models may be composed of non-intersecting elements within a spatial partition, such as voxels or oct-trees, or created from (possibly overlapping) combinations of inherently 3D primitive volumes.

Voxel Models

The first volumetric model we examine is the voxel model. Here space is completely filled by a tessellation of cubes or parallelopipeds called voxels (volume elements). Usually there is a density or other numerical value associated with each voxel. Storing a high resolution tessellation is expensive in space but simple in data structure (just a large 3D array of values). Usually some storage optimization schemes are required for detailed work (1K x 1K x 1K spaces). Special techniques are needed to compute surface normals and shading to suppress the boxiness of the raw voxel primitive. Voxel data is commonly obtained in the medical domain; it is highly regarded for diagnostic purposes as the 3D model does not speculate on additional data (say by surface fitting) nor suppress any of the original data however convoluted.

Voxel models are the basis for much of the scientific visualization work in biomedical imaging [FLP89]. The possible detail for human models is only limited by the resolution of the sensor. Accurate bone joint shapes may be visualized, as well as the details of internal and external physiological features. These methods have not yet found direct application in the human factors domain, since biomechanical rather than anatomical issues are usually addressed. Real-time display of voxel images is also difficult, requiring either

low resolution image sets or special hardware [GRB+85].

Constructive Solid Geometry

One of the most efficient and powerful modeling techniques is constructive solid geometry (CSG). Unlike the voxel models, there is no requirement to regularly tessellate the entire space. Moreover, the primitive objects are not limited to (uniform) cubes; rather there are any number of simple primitives such as cube, sphere, cylinder, cone, half-space, etc. Each primitive is transformed or deformed and positioned in space. Combinations of primitives or of previously combined objects are created by the Boolean operations. An object therefore exists as a tree structure which is "evaluated" during rendering or measurement.

CSG has been used to great advantage in modeling machined parts, but has not been seriously used for human body modeling. Besides the mechanical look created, real-time display is not possible unless the CSG primitives are polygonized into surfaces. When the set of primitives is restricted in one way or other, however, some useful or interesting human models have been built.

Single Primitive Systems

The generality of the constructive solid geometry method – with its multiplicity of primitive objects and expensive and slow ray-tracing display method – is frequently reduced to gain efficiency in model construction, avoid Boolean combinations other than union, and increase display speed. The idea is to restrict primitives to one type then design manipulation and display algorithms to take advantage of the uniformity of the representation. Voxels might be considered such a special case, where the primitives are all coordinate axis aligned and integrally positioned cubes. Other schemes are possible, for example, using ellipsoids, cylinders, superquadrics, or spheres.

Ellipsoids have been used to model cartoon-like figures [HE78, HE82]. They are good for elongated, symmetric, rounded objects. Unfortunately, the shaded display algorithm is nearly the same as the general ray-tracing process.

Cylinders have also been used to model elongated, symmetric objects. Elliptic cylinders were used in an early human modeling system [Wil82]. These primitives suffer from joint connection problems and rather poor representations of actual body segment cross-sections.

Superquadrics are a mathematical generalization of spheres which include an interesting class of shapes within a single framework: spheres, ellipsoids, and objects which arbitrarily closely look like prisms, cylinders, and stars. Simple parameters control the shape so that deformations through members of the class are simple and natural. Superquadrics are primarily used to model man-made objects, but when overlapped can give the appearance of faces and figures [Pen86].

Spheres as a single primitive form an intriguing class. Spheres have a simplicity of geometry that rivals that of simple points: just add a radius.

There are two methods of rendering spheres. Normally they are drawn as regular 3D objects. A human modeled this way tends to look like a large bumpy molecule. Alternatively, spheres may be treated like "scales" on the modeled object; in this case a sphere is rendered as a flat shaded disk. With sufficient density of overlapping spheres, the result is a smoothly shaded solid which models curved volumes rather well. A naturalistic human figure was done this way in our earlier TEMPUS system [BB78, BKK+85, SEL84]. We stopped using this method as we could not adequately control the sphere/disk overlaps during animation and newer workstation display technology favored polygons.

Potential Functions

An interesting generalization of spheres which solves some major modeling problems is to consider the volume as a potential function with a center and a field function that decreases monotonically (by an exponential or polynomial function) from the center outward. There is no "radius" or size of the potential function; rather, the size or surface is determined by setting a threshold value for the field. What makes this more interesting is that potential functions act like energy sources: adjacent potential functions have overlapping fields and the resultant value at a point in space is in fact the sum of the fields active at that point. Thus adjacent fields blend smoothly, unlike the "creases" that are obtained with fixed radius spheres [Bli82]. Recently, directional dependence and selective field summation across models have been added to create "soft" models that blend with themselves but not with other modeled objects in the environment [WMW86, NHK+85]. Potential functions were originally used to model molecules, since atoms exhibit exactly this form of field behavior, but the models have an amazing naturalistic "look" and have been used to great effect in modeling organic forms including human and animal figures [NHK+85, BS91]. The principal disadvantages to potential functions lie in properly generating the numerous overlapping functions and very slow display times. They remain an interesting possibility for highly realistic models in the future.

2.1.3 The Principal Body Models Used

[2]The default polyhedral human figure in *Jack* is composed of 69 segments, 68 joints, 136 DOFs, and 1183 polygons (including cap and glasses). The appearance is a compromise between realism and display speed. No one is likely to mistake the figure for a real person; on the other hand, the movements and speed of control are good enough to convey a suitably responsive attitude. The stylized face, hat, and glasses lend a bit of character and actually assist in the perception of the forward-facing direction.

For more accurate human bodies, we have adapted a database of actual body scans of 89 subjects (31 males and 58 females) supplied by Kathleen

[2]Pei-Hwa Ho.

Robinette of Wright-Patterson Air Force Base and used with her permission. The original data came in contours, that is, slices of the body in the transverse plane [GQO+89]. Each body segment was supplied as a separate set of contours. A polygon tiling program was used to transform the contours of each body segment into a surface representation.

In order to represent the human body as an articulated figure we needed to first compute the proper joint centers to connect the segments together. Joint center locations were computed through the coordinates of anthropometric landmarks provided with the contour data.

Real humans are not symmetrical around the sagittal plane: our left half is not identical to our right half. This was the case with the contour data. For consistency, rather than accuracy, we used the right half of the body data to construct a left half and then put them together. We also sliced the upper torso to take advantage of the seventeen segment spine model (Section 2.3). The resulting human body model has thirty-nine segments and about 18,700 polygons compared to the 290 slices in the original data.

2.2 Representing Articulated Figures

Underneath the skin of a human body model is a representation of the skeleton. This skeletal representation serves to define the moving parts of the figure. Although it is possible to model each of the bones in the human body and encode in the model how they move relative to each other, for most types of geometric analyses it is sufficient to model the body segments in terms of their lengths and dimensions, and the joints in terms of simple rotations. There are some more complex joint groups such as the shoulder and spine where inherent dependencies across several joints require more careful and sophisticated modeling.

The increasing interest in recent years in object-oriented systems is largely due to the realization that the design of a system must begin with a deep understanding of the objects it manipulates. This seems particularly true in a geometric modeling system, where the word "object" takes on many of its less abstract connotations. It has long been an adage in the user interface software community that a system with a poorly designed basic structure cannot be repaired by improving the interface, and likewise that a well designed system lends itself easily to an elegant interface.

This section describes **Peabody**, which represents articulated *figures* composed of *segments* connected by *joints*. The **Peabody** data structure has a companion language and an interactive interface in *Jack* for specifying and creating articulated figures. The data structure itself maintains geometric information about segment dimensions and joint angles, but it also provides a highly efficient mechanism for computing, storing, and accessing various kinds of geometric information. One of the principal tasks requested of **Peabody** is to map segment dimensions and joint angles into global coordinates for end effectors.

Peabody was designed with several criteria in mind:

- It should be general purpose. It should be able to represent many types of figures of tree-structured topology. It should not be hard coded to represent a specific type of figure, such as a human figure or a particular robot manipulator.
- It should have a well developed notion of articulation. Rather than concentrating on representations for primitive geometric shapes, **Peabody** addresses how such shapes can be connected together and how they behave relative to each other.
- It should represent tree-structured objects through a hierarchy. The inverse kinematics positioning algorithm can calculate and maintain the information necessary to simulate closed loops.
- It should be easy to use. The external user view of the figures should be logical, clear, and easy to understand. Understanding the figures should not require any knowledge of the internal implementation, and it should not require any advanced knowledge of robotics or mechanics.

2.2.1 Background

Kinematic Notations in Robotics

The most common kinematic representation in robotics is the notation of Denevit and Hartenberg [Pau81, DH55]. This representation derives a set of parameters for describing a linkage based on measurements between the axes of a robot manipulator. The notation defines four parameters that measure the offset between subsequent coordinate frames embedded in the links, or segments: 1) the angle of rotation for a rotational joint or distance of translation for a prismatic joint; 2) the length of the link, or the distance between the axes at each end of a link along the common normal; 3) the lateral offset of the link, or the distance along the length of the axis between subsequent common normals; and 4) the twist of the link, or the angle between neighboring axes. The notation prescribes a formal procedure for assigning the coordinate systems to the links in a unique way.

The objective behind these kinematic notations in robotics is to develop a standard representation that all researchers can use in the analysis and description of manipulators. There are several types of manipulators that are extremely common in the robotics research community. The adoption of a standard representation would greatly simplify the process of analyzing and implementing robotics algorithms since so many algorithms are described in the literature using these manipulators.

Animation Systems

Computer graphics and animation literature seldom addresses syntactic, or even semantic, issues in representations for mechanisms, except as background

for some other discussion of an animation technique or system.

Most interactive animation systems such as GRAMPS [OO81], TWIXT [Gom84], and BBOP [Stu84, Ste83], as well as commercial animation packages such as Alias [Ali90] and Wavefront [Wav89] only provide a mechanism of attaching one object to another. In this way, the user can construct hierarchies. When the user manipulates one object, its child objects follow, but there is no real notion of articulation. The attachments simply state that the origin of the child object is relative to the origin of the parent.

Many animation systems are non-interactive and are based on scripts that provide a hierarchy only through a programming language interface. Examples of such systems are ANIMA-II [Hac77], ASAS [Rey82] and MIRA-3D [MTT85]. In this kind of system, the hierarchy is hard-coded into the script, possibly through an interaction loop. A hierarchy designed in this way is very limited, except in the hands of a talented programmer/animator who can write into the animation a notion of behavior.

Physically-Based Modeling Systems

Physically based modeling systems such as that of Witkin, Fleisher, and Barr [WFB87] and Barzel and Barr [BB88] view the world as objects and constraints. Constraints connect objects together through desired geometric relationships or keep them in place. Otherwise, they float in space under the appropriate laws of physics. There is no notion of articulation other than constraints. This forces the burden of maintaining object positions entirely to the algorithms that do the positioning. For simple objects, this is conceptually pleasing, although for complex objects it is computationally difficult. If a system represents joints like the elbow as a constraint, the constraint must have a very high weighting factor in order to ensure that it never separates, requiring very small time steps in the simulation. This may also complicate the user's view of objects such as robots or human figures, which are inherently articulated. We believe it is important to differentiate the relationship between body segments at the elbow and the relationship between a hand and a steering wheel.

2.2.2 The Terminology of Peabody

Peabody uses the term *environment* to refer to the entire world of geometric objects. The environment consists of individual *figures*, each of which is a collection of *segments*. The segments are the basic building blocks of the environment. Each segment has a geometry. It represents a single physical object or part, which has shape and mass but no movable components. The geometry of each segment is represented by a *psurf*, which is generally a polyhedron or a polygonal mesh but can be of a more general nature.

The term *figure* applies not only to articulated, jointed figures such as a human body: any single "object" is a figure. It need not have moving parts. A figure may have only a single segment, such as a coffee cup, or it may be

composed of several segments connected by *joints,* such as a robot. Sometimes the term "object" denotes any part of the **Peabody** environment.

Joints connect segments through attachment frames called *sites*. A site is a local coordinate frame relative to the coordinate frame of its segment. Each segment can have several sites. Joints connect sites on different segments within the same figure. Sites need not lie on the surface of a segment. A site is a coordinate frame that has an orientation as well as a position. Each site has a *location* that is the homogeneous transform that describes its placement relative to the base coordinate frame of its segment.

Segments do not have specific dimensions, such as the length, offset, and twist of Denevit and Hartenberg notation, because the origin can lie anywhere on the segment. The location of the axes of the joints that connect the segment are phrased in terms of this origin, rather than the other way around. The measurement of quantities such as length is complicated, because segments may have several joints connected to them, and none of these joints is designated in the definition as the "parent."

Joints may have several DOFs, which are rotational and translational axes. Each axis and its corresponding angle form a single rotation or translation, and the product of the transform at each DOF defines the transform across the joint, defining the placement of the sites, and thus the segments, that the joint connects.

The directionality of a joint is important because it defines the order in which the DOF transforms are concatenated. Because these transforms are not commutative, it is essential that the order is well-defined. This is an especially important feature of **Peabody**, since it is sometimes convenient to define the direction of the joint in a way different from the way the joint occurs in the figure hierarchy. An example of this is the human knee. Although it may be useful to structure the hierarchy of a human body with the root at the foot, it is also appealing to have the joints at both knees defined in the same manner.

2.2.3 The Peabody Hierarchy

Peabody avoids imposing a predefined hierarchy on the figures by encouraging the user to think of figures as collections of segments and joints, none with special importance. However, there must exist an underlying hierarchy because **Peabody** is not equipped to handle closed-loop mechanisms. (Closed loop structures are managed through the constraint satisfaction mechanism.) The structure of the **Peabody** tree is defined by designating one site on the figure as the *root*. The root site roughly corresponds to the origin of the figure, and it provides a handle by which to specify the location of the figure. Viewing the figure as a tree, the root of the figure is the root of the tree. The root site of a figure may change from time to time, depending upon the desired behavior of the figure.

This means there are two representations for the hierarchy, one internal and one external. There are many advantages to having a dual representation

of the hierarchy. First, it allows the hierarchy to be inverted on the fly. Most models have a natural order to their hierarchy, emanating from a logical origin, but this hierarchy and origin may or may not correspond to how a model is placed in the environment and used.

The choice of the figure root is particularly important to the inverse kinematics algorithm, since the algorithm operates on chains of joints within the figure. At least one point on the figure must remain fixed in space. Because the internal representation of the hierarchy is separate, the user maintains a consistent view of the transform across a joint, regardless of how the figure is rooted.

The example below illustrates the **Peabody** hierarchy. Each segment has its base coordinate frame in the middle and an arc leading to each of its sites. The transform along this arc is the site's location. Each site may have several joints branching out from it, connecting it downwards in the tree to sites on other segments.

```
figure table {
    segment leg {
        psurf = "leg.pss";
        attribute = plum;
        site base->location = trans(0.00cm,0.00cm,0.00cm);
        site top->location = trans(5.00cm,75.00cm,5.00cm);
    }
    segment leg0 {
        psurf = "leg.pss";
        attribute = springgreen;
        site base->location = trans(0.00cm,0.00cm,0.00cm);
        site top->location = trans(5.00cm,75.00cm,5.00cm);
    }
    segment leg1 {
        psurf = "leg.pss";
        attribute = darkslategray;
        site base->location = trans(0.00cm,0.00cm,0.00cm);
        site top->location = trans(5.00cm,75.00cm,5.00cm);
    }
    segment leg2 {
        psurf = "leg.pss";
        attribute = darkfirebrick;
        site base->location = trans(0.00cm,0.00cm,0.00cm);
        site top->location = trans(5.00cm,75.00cm,5.00cm);
    }
    segment top {
        psurf = "cube.pss" * scale(1.00,0.10,2.00);
        site base->location = trans(0.00cm,0.00cm,0.00cm);
        site leg0->location = trans(95.00cm,0.00cm,5.00cm);
        site leg2->location = trans(5.00cm,0.00cm,5.00cm);
```

```
        site leg3->location = trans(95.00cm,0.00cm,195.00cm);
        site leg4->location = trans(5.00cm,0.00cm,195.00cm);
    }
    joint leg1 {
        connect top.leg3 to leg1.top;
        type = R(z);
    }
    joint leg2 {
        connect top.leg2 to leg0.top;
        type = R(z);
    }
    joint leg3 {
        connect top.leg4 to leg2.top;
        type = R(z);
    }
    joint leg4 {
        connect top.leg0 to leg.top;
        type = R(z);
    }
    root = top.base;
    location = trans(0.00cm,75.00cm,0.00cm);
}
```

2.2.4 Computing Global Coordinate Transforms

The root site for the figure is the one at the top of the tree, and its global location is taken as given, that is, not dependent on any other element of the environment. The root, the site locations, and the joint displacements uniquely determine the global location of every site and segment in the tree in terms of a product of transforms from the root downward.

The computation of the coordinate transforms for each segment and site in the downward traversal of the tree requires inverting the site locations that connect the segment to other segments lower in the tree. It may also require inverting joint displacements if the joint is oriented upwards in the tree. Computationally, this is not expensive because the inverse of a homogeneous transform is easy to compute, through a transpose and a dot product.

2.2.5 Dependent Joints

[3]The human figure can be abstracted as an object which is to be instantiated (into a certain pose) by any specification of all the joint angles. While any pose can be represented by a set of joint angles, it is not always possible to supply a full and reasonable set of angles. Often, for example, there is a

[3]Jianmin Zhao.

natural grouping of joints such as the torso or shoulder mass that typically work together. Arbitrary (admissible) joint angles for the joints in the group may not represent a legitimate posture: they are functionally dependent on each other.

Conceptually, these dependencies compromise the notion of the joint and joint angle and blur the boundary of the object definition. It seems that not all joints are created equal. Rather than have a system which tries to cope with every joint the same way, we take a more practical approach. We use a *joint group* concept in **Peabody** to accommodate joint dependency so that the relationship is coded into the object definition rather than the application program.

A joint group is a set of joints which are controlled as one entity. Internal joint angles are not visible outside of the group: they are driven by the group driver. The driver is nothing but a mapping from a number of parameters (counterparts of joint angles of the independent joint) to joint angles of its constituent joints. Those independent parameters will be called *group angles*. Similar to the joint angles of the independent joint, the group angles of the joint group are subject to linear constraints of the form

$$\sum_{i=1}^{n} a_i \theta_i \leq b_i \tag{2.1}$$

where θ's are group angles and n is the number of θ's, or number of DOFs of the joint group. There may be many such constraints for each group.

There can be many applications of the joint group. Forearm pronation and supination change the segment geometry, so one way to manage that within the geometry constraints of psurfs is to divide the forearm into a number of nearly cylindrical sub-segments. As the wrist moves, its rotation is transmitted to the forearm segments such that distal sub-segments rotate more than proximal ones. The segment adjacent to the elbow does not pronate or supinate at all. Fingers could be managed in a similar fashion by distributing the desired orientation of the fingertip over the three joints in the finger chain. The most interesting examples, though, involve the torso and the shoulder. We address these cases in the next sections.

2.3 A Flexible Torso Model

[4]Human figure models have been studied in computer graphics almost since the introduction of the medium. Through the last dozen years or so, the structure, flexibility, and fidelity of human models has increased dramatically: from the wire-frame stick figure, through simple polyhedral models, to curved surfaces, and even finite element models. Computer graphics modelers have tried to maximize detail and realism while maintaining a reasonable overall display cost. The same issue pertains to control: improving motion realism

[4]Gary Monheit.

requires a great number of DOFs in the body linkage, and such redundancy strains effective and intuitively useful control methods. We can either simplify control by simplifying the model, thereby risking unrealistic movements; or complicate control with a complex model and hope the resulting motions appear more natural. The recent history of computer animation of human figures is focused on the quest to move the technology from the former situation towards the latter while simultaneously forcing the control complexity into algorithms rather than skilled manual manipulation.

This point of view motivates our efforts in human figure modeling and animation, as well as those of several other groups. Though notable algorithms for greater animation power have addressed kinematics, dynamics, inverse kinematics, available torque, global optimization, locomotion, deformation, and gestural and directional control, the human models themselves tended to be rather simplified versions of real human flexibility. In the early 1980's we warned that increased realism in the models would demand ever more accurate and complicated motion control; now that the control regimes are improving, we must return to the human models and ask if we must re-evaluate their structure to take advantage of algorithmic improvements. When we considered this question, we determined that a more accurate model of the human spine and torso would be essential to further realism in human motion.

Although many models have appeared to have a flexible torso, they have been computer constructions of the surface shape manipulated by skilled animators [Emm85]. We needed a torso that was suitable for animation, but also satisfied our requirements for anthropometric scalability. Thus a single model of fixed proportions is unacceptable as human body types manifest considerable differences. (A similar type of flexible figure is found in snakes [Mil88, Mil91], but the anthropometry issues do not arise. Moreover, this snake animation is dynamics-based; humans do not need to locomote by wiggling their torsos and so a kinematics model was deemed adequate.) Zeltzer and Stredney's "George" skeleton model has a detailed vertebral column, but it is not articulated nor is it bent during kinematic animation [Zel82]. Limited neck vertebral motion in the saggital plane was simulated by Willmert [Wil82]. Various body models attempt a spine with a 3D curve but shape and control it in a *ad hoc* fashion.

If the spine were realistically modeled, then the torso, a vessel connected and totally dependent on the spine, could then be viewed and manipulated interactively. So we undertook the development of a far more satisfactory and highly flexible vertebral model of the spine and its associated torso shape.

The conceptual model of the spinal column is derived from medical data and heuristics related to human kinesiology. The spine is a collection of vertebrae connected by ligaments, small muscles, vertebral joints (called processes), and intervertebral discs [BBA88]. Nature has designed the spine for support of the body's weight, stability of the torso, flexibility of motion, and protection of the spinal cord [AM71, Hol82].

The spine moves as a column of vertebrae connected by dependent joints, meaning that it is impossible to isolate movement of one vertebral joint from

the surrounding vertebrae [Lou83]. Muscle groups of the head, neck, abdomen and back initiate the movement of the spine, and the interconnecting ligaments allow the movement of neighboring vertebrae [BBA88, Wel71].

2.3.1 Motion of the Spine

Anatomy of the Vertebrae and Disc

The spinal column consists of 33 vertebrae organized into 5 regions [BBA88]: cervical, thoracic, lumbar, sacral, and coccyx.

The vertebrae are labeled by medical convention in vertical descending order: C1–C7, T1–T12, L1–L5, and S1–S5. Which regions should be considered part of the torso? The cervical spine lies within the neck. The sacrum and coccyx contain vertebrae that are fixed through fusion [AM71]. Since the mobile part of the torso includes the 12 thoracic and 5 lumbar vertebrae, all together 17 vertebrae and 18 joints of movement are included in the torso model.

Each vertebra is uniquely sized and shaped, but all vertebrae contain a columnar body and an arch. The body is relatively large and cylindrical, supporting most of the weight of the entire spine. The vertebral bodies increase gradually in size from the cervical to the lumbar region [AM71].

The arch supports seven processes: four articular, two transverse, and one spinous [AM71]. The processes are bony protrusions on the vertebra that aid and limit the vertebral motion. The transverse and spinous processes serve as levers for both muscles and ligaments [BBA88]. The articular processes provide a joint facet for the joint between successive vertebral arches. These processes, due to their geometry, cause the vertebrae to rotate with 3 DOFs. Ligaments and small muscles span successive vertebral processes. They give the spinal column its stability. Because of this strong interconnectivity, spinal movement is modeled as interdependent movements of neighboring joints.

Vertebrae are each separated by intervertebral discs. The disc has 3 parts [Lou83]:

- **nucleus pulposus** - the sphere in the center, consisting of 85% water
- **annulus fibrosus** - the fibers running as concentric cylinders around the nucleus
- **cartilaginous plates** - a thin wall separating the disc from the vertebral body.

The disc changes shape as the neighboring vertebrae bend. But, since the nucleus is 85% water, there is very little compression. The disc can bulge out spherically, as force is applied to the columnar body above or below. Therefore, overall the disc does not function as a spring, but as a deformable cylindrical separation between vertebrae, supporting the theory that the vertebrae do not slide, but rotate around an axis [Lou83].

Range of Movement of Each Vertebra

Vertebral movement is limited by the relative size of the disks, the attached ligaments, and the shape and slant of the processes and facet joints. Statistics for joint limits between each successive vertebra have been recorded and compiled [Lou83]. Also, the spine has a natural shape at rest position. The initial joint position of each vertebra is input to the model.

The range of movement of each region of the spine is different. For instance, the optimum movement of the lumbar region is flexion or extension. The thoracic area easily moves laterally, while flexion/extension in the sagittal plane is limited. The cervical area is very flexible for both axial twisting and lateral bending. The joint limits for each region affect how much that joint is able to participate in any given movement. The posture of the torso is a result of the specialization of the spinal regions [Wil75].

Effect of the Surrounding Ligaments and Muscles

The vertebrae are interconnected by a complex web of ligaments and muscles. If the force initiated by a muscle group is applied at one joint, the joint moves and the neighboring joints also move to a lesser degree. Some joints farther away might not be affected by the initiator joint's movement.

It is possible to deactivate joints that are not initiating the movement. This action is achieved by simultaneous contractions of extensor and flexor muscles around the spinal column [Wil75]. Depending on the force of these resisting muscles, the joints on or near the joint closest to the resistor will move less than they would if the resisting force had not been applied. The final position of the spine is a function of the initiator force, the resisting muscle, and the amount of resistance.

2.3.2 Input Parameters

The spine is modeled as a black box with an initial state, input parameters, and an output state [MB91]. To initiate movement of the spine, several input parameters are introduced. These parameters are:

joint range FROM and TO: Within the total number of joints in the spine, any non-empty contiguous subset of vertebral joints may be specified by two joint indices. These joints indicate which part of the spine is active in movement. For example, the user specifies movement in the range between T5 and T10. All other joints are frozen in the movement.

initiator joint: The joint where movement begins, usually the joint with greatest motion.

resistor joint: The joint that resists the movement. This may be equated to a muscle that contracts and tries to keep part of the spine immobile.

resistance: The amount of resistance provided by the resistor joint.

spine target position: This is a 3D vector describing the target position after rotation around the x, y, and z axis. The target position is the sum of all joint position vectors in the spine after movement succeeds.

zero interpolation: A value of "yes" indicates that movement is interpolated through the joint rest position. A value of "no" indicates that only the joint limits are used to interpolate movement.

2.3.3 Spine Target Position

The joint between each vertebra has three degrees of rotation. The spine will move toward the target position by rotating around the three possible axes [Lou83]:

ROTATION OF THE SPINE		
flexion/extension	Forward/backward bending	Rotation around x axis
axial rotation	Twisting	Rotation around y axis
lateral bending	Side bending	Rotation around z axis

The position of the flexion rotational axis for each vertebral joint has been measured from cadavers, and is not equidistant to the two adjacent vertebrae, but is closer to the bottom vertebra [Lou83]. The origin of the axis of movement determines how the vertebrae move. When the torso is modeled on the spine, the axis also directly determines how the torso changes shape.

Elongation and compression are absent from the model. The hydrophilic intervertebral disc, when submitted to prolonged compression induces a slight decrease in height due to fluid leakage. Conversely, after a long period of rest or zero-gravity, the spine elongates by maximum filling of the nucleus pulposus (at the center of the disc) [Lou83]. Dehydration during a day's activity can result in a loss of height of 2 cm in an adult person. In any short duration of movement the disc is essentially incompressible, and therefore elongation is imperceptible [Hol81].

Shearing or sliding (translational movements) of the vertebrae would lead to variation in the intervertebral separation. This would not be allowed by the mechanics of the intervertebral disc [Lou83]. Therefore, the assumption is made that for normal activities the three degrees of rotational movement are the only ones possible for each vertebral joint.

2.3.4 Spine Database

Any human figure can have a wide variety of torso shapes. Also, each person has a different degree of flexibility and range of movement. In order to model the position and shape changes of an individual's spine, a database has been designed for creating a unique set of features for the spine and torso. Medical data is the source of the database elements of an average person [Lou83]. The database consists of the size of each vertebra in the x, y, z dimension,

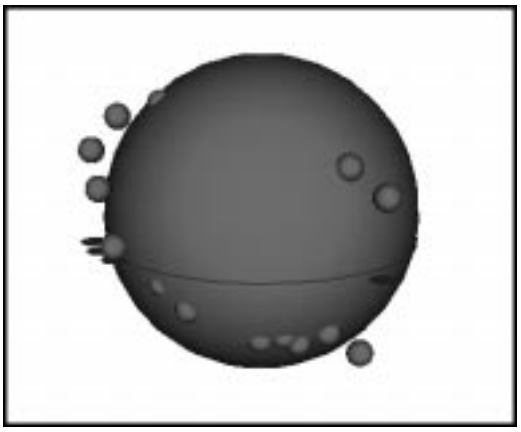

Figure 2.1: Spherical Trajectory of the Shoulder.

the intervertebral disc size, the joint limits (3 rotations with 2 limits per rotation), and the joint rest (initial) position. In Section 4.2.3 we will see how spine movement is realized.

2.4 Shoulder Complex

[5]It is well known that the movement of the humerus (the upper arm) is not a matter of simple articulation as most computer graphics models would have it. The movement is caused by articulations of several joints – glenohumeral joint, claviscapular joint and sternoclavicular joint. Collectively, they are called the *shoulder complex* [EP87, ET89].

In *Jack*, the shoulder complex is simplified with two joints – one connecting the sternum to the clavicle and the other connecting the clavicle to the humerus [GQO+89]. We call the former joint the *clavicle joint* and the latter the *shoulder joint*. This simplification implies that the rotational center of the humerus lies on a spatial sphere when the upper arm moves. It turns out that it is very close to empirical data collected with a 6-D sensor attached to the external midpoint between the dorsal and ventral side of the right upper arm [Mau91]. The z-axis of the sensor was along the longitudinal axis of the upper arm with the positive z axis direction pointing proximally, and the negative y-axis pointing into the upper arm. The sensor was placed 10 inches from the shoulder (the extremal point of the humerus).

From the experimental data the trajectory of the shoulder can be easily computed. We fitted the trajectory by the sphere which yields minimum average residual error. The result is quite satisfactory: the radius of the sphere is 6.05cm, and the average error (the distance from the trajectory point to the sphere) is 0.16cm. (Figure 2.1). Notice that the radius of the

[5]Jianmin Zhao.

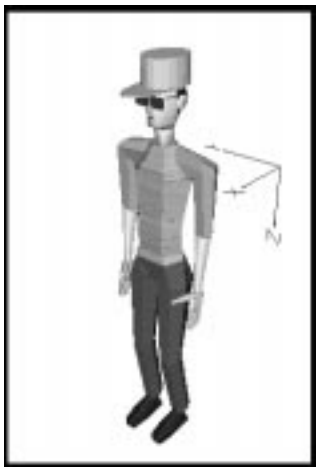

Figure 2.2: A Neutral Human Figure.

sphere is not the same as the dimension of the clavicle. This is due to the fact that the shoulder complex has indeed three joints instead of two. The net result can be modeled by two joints, however, if we put the center of the clavicle joint inbetween the two clavicle extremes.

In our applications we feel that two joints are adequate to model the shoulder complex. So, we modeled the shoulder complex by grouping these two joints into a joint group. Now we need to define the group angles and the way they drive the internal joint angles.

2.4.1 Primitive Arm Motions

It will be convenient to have the group angles of the shoulder complex describe the arm's motion in a natural way. To focus on this motion, we assume that the elbow, the wrist and all joints down to fingers are fixed. What are the arm's primitive motions? Mathematically, any three independent arm's motions will suffice. But careful selection will pay off in positioning ease.

The arm's motion can be decomposed into two parts: spherical and twisting motions. The motion which moves the vector from the proximal end to the distal end of the upper arm is called spherical motion, and the motion which leaves this vector unchanged is called twisting.

In a neutral body stance, consider a coordinate system where the z axis points vertically downward, the x axis points towards the front, and the y axis points to the right. If we place the starting end of the vector from the proximal end to the distal end of the upper arm at the origin of the coordinate system, the terminating end will stay on a sphere with the center at the origin when the arm moves. The spherical motion can be further decomposed into two motions: one which moves the arm vector along the longitude of the sphere (elevation), and another which moves the vector along

the latitude (abduction). To describe the current status of the arm vector, we need a convention for zero elevation or abduction. Let us define the amount of elevation as the unsigned angle ϕ between z axis and the arm vector and, for the left arm, the amount of abduction as the signed angle θ between the $-y$ axis and the projection of the arm vector on the xy plane. The positive abduction is defined when the absolute angle from $-y$ to the projection is less than 180°. Joint limits can be specified in terms of these spherical or "globographic" limiting angles [EP87].

2.4.2 Allocation of Elevation and Abduction

Naturally, the joint group shoulder complex will have elevation, abduction and twist as its group angles. They will be realized by internal joints — the clavicle and the shoulder joints. The amounts of elevation and abduction of the arm are allocated to the shoulder joint and the clavicle joint, while the twist is allocated to the shoulder joint alone.

According to clinical data, Otani gave a formula for distributing elevation and abduction to the shoulder and clavicle [Ota89]:

$$\phi_c = \cos(\theta)\beta_1 + (1-\cos(\theta))\beta_2 - 90 \tag{2.2}$$

$$\theta_c = 0.2\,\theta \tag{2.3}$$

$$\phi_s = \phi - \phi_c \tag{2.4}$$

$$\theta_s = \theta - \theta_c \tag{2.5}$$

where ϕ and θ are total elevation and abduction of the shoulder complex, subscripts "c" and "s" stand for the portions carried by the clavicle and shoulder joints, respectively, and

$$\beta_1 = \begin{cases} 0.2514\phi + 91.076 & \text{for } 0 \le \phi \le 131.4 \\ -0.035\phi + 128.7 & \text{for } \phi > 131.4 \end{cases} \tag{2.6}$$

$$\beta_2 = \begin{cases} 0.21066\phi + 92.348 & \text{for } 0 \le \phi \le 130.0 \\ 120.0 & \text{for } \phi > 130.0 \end{cases} . \tag{2.7}$$

2.4.3 Implementation of Shoulder Complex

In **Peabody**,

$$R(x, y, z) \tag{2.8}$$

is used to denote a generic rotation about the axis (x, y, z). Since we need rotations about some coordinate axis, let

$$R_x(\omega) \tag{2.9}$$

represent the rotation about x axis by ω degrees. Analogous notation will be used for the y and z axes.

Peabody uses row vector convention. So a string of rotations, when read from the right, can be interpreted as applying the first rotation (the rightmost

one), then applying the next one about the rotated reference frame, and so on. When read from the left, it should be interpreted as applying successive rotations about the fixed starting frame.

With respect to the standing reference frame the spherical motion can be described as

$$R(1,0,0) * R(0,0,1), \tag{2.10}$$

that is, to achieve elevation and abduction of amounts ϕ and θ, the arm can rotate about the x axis by ϕ, followed by rotating about the unrotated y axis by θ. This is just spherical coordinates which designate a point on a sphere. The first problem we shall encounter is the singularity inherent to the spherical coordinate system.

Dealing with Singularities of Spherical Coordinates

As is known, the spherical coordinates have two singularities: when the elevation is 0 or 180°, it represents the pole (south pole for 0 or north pole for 180) no matter what the longitude (abduction amount) is. To see how this would affect the description of the spherical motion of the arm, let us do a small experiment. Starting with your left arm hanging down at your side, elevate your arm by 90°, then abduct by 90°, and finally elevate by −90°. Now see where your hand is. You will find that your hand comes back but with an axial twist. Where does this twist come from? It means that the pair of so defined elevation and abduction motions are not independent from twist. As long as elevation is zero, the arm vector would not change. This is nothing but a twist, as our decomposition of spherical motion and twisting motion intended. The final coordinates are (0, 90), since the last elevation cancels the first one and leaves the "abduction" there. To compensate for this unwanted twist, we untwist the arm by the amount that the "abduction" would induce before the shoulder leaves the zero elevation. Therefore, we need a joint of three cascaded rotations as follows,

$$R(0,0,1) * R(1,0,0) * R(0,0,1) \tag{2.11}$$

Let τ, ϕ and θ be desired twist, elevation and abduction, respectively. Then the joint angles (or displacement, in **Peabody** language) should be

$$(\tau - \theta, \phi, \theta). \tag{2.12}$$

The values in (2.12) are the amount of rotation about each of the corresponding axes in (2.11). The minus θ term in the first joint angle is to compensate for the unwanted twist induced by the abduction θ (so realized). As a matter of fact, the amount of twist is a relative quantity. It is meaningful only if we have zero twist assumed for any direction of the arm. So (2.12) should be

$$(\tau_0 + \tau - \theta, \phi, \theta), \tag{2.13}$$

where τ_0 is a function of $(\phi,\ \theta)$, which defines the zero twist for each $(\phi,\ \theta)$. Function τ_0 should be such that

$$\tau_0(0,\theta) = \text{constant}, \tag{2.14}$$

since $(0,\theta)$ denotes the same direction of the arm regardless of the value of θ. Now elevation by $90°$, followed by abduction of $90°$, and then elevation of $-90°$ would wind up with

$$(\tau_0 - 90, 0, 90).$$

Since $R_x(0) =$ identity matrix, the final matrix $R_z(\tau_0 - 90) * R_x(0) * R_z(90)$ would be equal to $R_z(\tau_0)$, which is exactly the starting configuration without twisting.

Now let us take care of the other singular point, that is, when $\phi = 180°$. At this point, $R_x(180) \neq$ identity. But we have

$$R_x(180) * R_z(\theta) = R_z(-\theta) * R_x(180).$$

To deal with this singularity, we require that

$$\tau_0(180,\theta) = 2\theta + \text{constant} \tag{2.15}$$

to compensate for unwanted twist induced by "abduction" θ. When $\phi = 180°$,

$$R_z(\tau_0(180,\theta) + \tau - \theta) * R_x(180) * R_z(\theta) \tag{2.16}$$

$$\begin{aligned} &= R_z(\tau_0(180,\theta) + \tau - \theta) * R_z(-\theta) * R_x(180) \\ &= R_z(\tau_0(180,\theta) + \tau - 2\theta) * R_x(180). \end{aligned}$$

(2.15) guarantees that that the final configuration is independent of the abduction by θ.

Combining (2.14) with (2.15), a possible choice of τ_0 could be

$$\tau_0(\phi,\theta) = \frac{\phi}{90}\theta. \tag{2.17}$$

This is the choice in our current implementation. To achieve more natural zero twist definition, we need to fine tune τ_0. One possibility is shown in Section 4.1.2 based on [Hut70] and [BOK80]. But (2.14) and (2.15) are required to deal with the singularity due to the spherical coordinate system.

Dealing with Hierarchical Nature of Connection

So far, we have focused on the shoulder joint. However, as we argued, the shoulder complex will consist of two joints – shoulder and clavicle joints. The total amount of elevation and abduction will be distributed to shoulder and clavicle joint according to the formula in Section 2.4.2.

In the *Jack* human figure model, the clavicle joint has the form

$$R(1,0,0) * R(0,1,0), \tag{2.18}$$

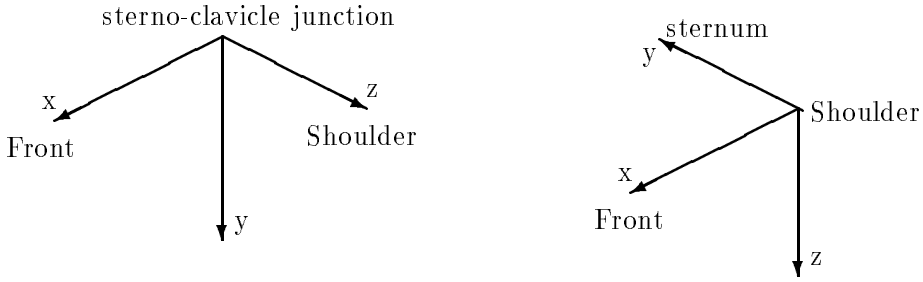

Figure 2.3: Site Orientations at Clavicle and Shoulder Joints.

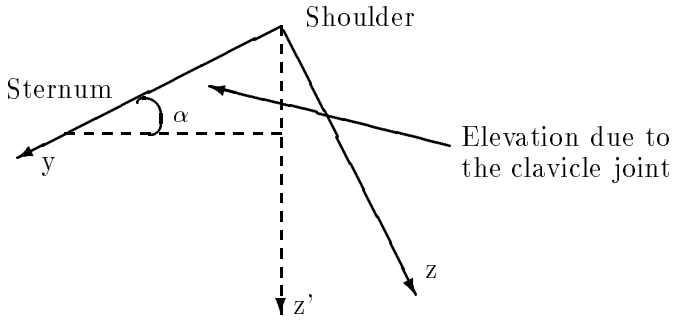

Figure 2.4: Tilted Shoulder Frame.

since the site orientation at the clavicle joint is a little bit different from that at the shoulder joint (Figure 2.3). We do not need to worry about the singularity problem here as we did with the shoulder joint because the arm, which the shoulder complex drives, is not aligned with the center of this joint. The amount of elevation and abduction of the clavicle is given in $(2.2 \sim 2.5)$. The clavicle is closer to the (normal) root of the human figure than the shoulder. This means that the global (relative to the neutral torso) orientation of the site at the shoulder joint will be altered by the movement of the clavicle: the z axis of the site at the shoulder would no longer point vertically downward. This causes the trajectory of the hand drawn by "abduction" to no longer be on the horizontal plane. Notice that the x axis will always be horizontal, and hence the meaning of elevation is not challenged. To protect "abduction" against this alteration, we need to do a transformation of the site coordinates.

After some movement of the clavicle, the shoulder reference frame is tilted as shown in Figure 2.4. (The Figure displays only the plane $x = 0$. The x axis

does not show up in the Figure, but should be understood by the right-hand principle.) The tilted abduction is due to the tilted z axis. The "correct" abduction should be the rotation about the dotted z' axis. This axis is not constant due to variable clavicle elevation, but the **Peabody** language requires a constant axis. To circumvent this restriction, we can perform a transformation of coordinates. The rotation axis described in (x, y', z') is simply $R(0, 0, 1)$. The matrix representation of a rotation in a different coordinate system is just a similarity transformation of the coordinate transformation matrix. Let vectors in (x, y', z') be primed. The transformation of coordinates is

$$v = R_x(\alpha)v' \tag{2.19}$$

where α is the amount of elevation allocated to the clavicle joint. The rotation about z' axis by ω represented in (x, y', z') is $R_{z'}(\omega)$ but, when represented in the old coordinate frame (x, y, z), it would be

$$R_x(\alpha)R_z(\omega)R_x(-\alpha). \tag{2.20}$$

Substituting this for the second $R(0, 0, 1)$ in the shoulder joint cascaded rotations (2.11), we get

$$R(0, 0, 1) * R(1, 0, 0) * R(1, 0, 0) * R(0, 0, 1) * R(1, 0, 0).$$

We do not need to substitute for the first $R(0, 0, 1)$ in (2.11), because it is there for twist and to compensate for unwanted twist caused by the abduction allocated to the shoulder joint. Two contiguous rotations about the x axis can be combined into one. The final form of the shoulder joint becomes

$$R(0, 0, 1) * R(1, 0, 0) * R(0, 0, 1) * R(1, 0, 0). \tag{2.21}$$

Let τ be the amount of twist and the notations of elevation and abduction be as in Section 2.4.2. The joint angles for the clavicle take values

$$(\theta_c, \phi_c) \tag{2.22}$$

with respect to (2.18), and the joint angles for the shoulder take values

$$(\tau + (\frac{\phi}{90} - 1)\theta_s, \phi, \theta_s, -\phi_c). \tag{2.23}$$

with regard to (2.21). It is not difficult to verify that the hand orientation does not depend on the abduction allocated to the shoulder joint, θ_s, when the total elevation, ϕ, is 0 or 180°.

2.5 Clothing Models

[6]In most workplace environments we have encountered, clothed figures are the norm and would be expected by the designer. Adding clothing to a human

[6]Eunyoung Koh.

figure improves its graphical appearance and realism. Clothes modeling can be done in many ways ranging from very simple to more realistic but complicated. The simplest clothing technique is to change the attributes of certain segments of the body figure; for example, by modifying the colors of the pelvis and upper leg segments we get the effect of a body wearing short pants. This is not quite as silly as it sounds, because the body segment geometry can be created with a clothed rather than bare-skinned shape. The best but more complicated approach is to drape and attach clothing over a body to simulate the intricate properties of garments.

Besides improving realism, there is a practical human factors aspect to clothing. Clothing constrains movement by restricting the joint angle limits. Preliminary attempts to analyze this problem use collision detection over a geometric clothes model.

2.5.1 Geometric Modeling of Clothes

Rigid clothing models are created by designing special segment psurfs. Thus a shirt or jacket would have, say, five parts: one for the torso, and two for each limb segment. Clothing exists independently of a given figure model as a library of objects which can be selectively placed on a model at user determined sites. This database is modifiable through typical geometric editing commands. A clothing item is positioned by matching key points on the clothing to key points on the body segments. A global deformation algorithm [SP86] can be used to fit the clothing piece correctly on the segment.

One apparent problem with geometrically modeled clothing occurs when the human figure moves joints. Since the clothing model is not deformable, there are gaps between segments. (This is in fact true even without clothing if the figure is modeled with polyhedral meshes. As the geometry is carried on the segment, it inherits the geometric transformation without any compensation for the interaction of material, flesh or clothes, at the joint.) Extra work is necessary to alleviate the joint gap problem. A gap filling algorithm has been developed to make up these gaps when animating. It connects the boundaries of two adjacent segments by generating spline surfaces using the tangent information of the two segments at the boundaries.

As an initial attempt to develop geometric models of clothes, a sweatshirt and pants were designed (Plate 2). Each segment of a clothing item is a psurf whose geometry and position are closely related to a corresponding body segment. The following is a step by step procedure for geometric clothes design.

1. Determine the size of clothes:

   In conventional clothing design, circumferences are measured at certain positions of the body in order to determine the clothing size. In our approach, the maximum segment breadth in two orthogonal directions are measured instead of circumferences.

   The following list shows the positions at which the maximum breadths are measured and lists their corresponding slices from the accurate biostereometric bodies (Section 2.1.3).

   - For trousers

waist	the 6th slice of lower_torso
hip	the 9th slice of hip_flap
upper leg	the first slice of upper_leg
lower leg	the last (24th) slice of lower_leg

   - For a sweatshirt

neck	the first slice of upper_torso
chest	the 6th slice of upper_torso
breast	the 12th slice of upper_torso
waist	the first slice of lower_torso
upper arm	the 8th slice of upper_arm
lower arm	the last (16th) slice of lower_arm

   Also, a measure of the length from the neck to the bottom of the shirt is provided to determine the length of the shirt. These sizes are specified in advance for each article of clothing.

2. Create psurfs:

   The geometry of each segment is determined by a set of body slices. A new set of slices for clothing is constructed by sculpturing the body slices depending on the shape of the segment and the specified sizes in the previous step. The fundamental idea in the construction is to pick a few thick slices and duplicate them appropriately along the segment after scaling. Scaling is done by linear interpolation so that the scaled slices may match with the specified maximum breadth sizes at the positions designated.

   The completed surface definition of a clothes segment can be obtained by tiling the slices. Tiling is performed by generating rectangles from the data points which define two adjacent slices.

3. Attach clothes segments to human body:

   Each clothes segment can be attached to the corresponding body segment by a joint which is located at the upper part of that segment.

The clothing shape can be easily modified by changing the slice definition of the clothes. For example, folded sleeves, short sleeves, and short pants can be simulated by simple modification or deletion of slices.

2.5.2 Draping Model

The most realistic clothing can be created by simulating the support and draping of pattern pieces of arbitrary shape. Wrinkling, folding, and the effects of gravity are displayed through a relaxation method or a finite element method. Pattern pieces may also be stitched at seams and draped simultaneously. Pattern pieces of different lengths may be sewn together, resulting in an oversewing effect.

The draping of the pattern pieces is done on a figure in a static posture. Interference testing is done in the draping algorithm to make sure that the pattern pieces slide over the surface of the figure without penetrating the surface (Plate 3).

There are several methods to simulate the draping of a square piece of cloth, isolated from other cloth, which are based on a relaxation method. Feynman [Fey86] uses a formula which minimizes the energy of a cloth and tries to simulate the shape of thin flexible membranes under the influence of force fields and rigid bodies. The local minimum of the cloth is found by moving each of the grid points in turn toward a position which decreases the energy of the cloth. The energy expression of a cloth is described as:

$$E_{total}(S) = k_s s(S) - k_b b(S) - k_g g(S)$$

where $s(S), b(S), g(S)$ represent the effects of strain, bending, and gravity. The parameters k_s, k_b, k_g control the relative strengths of these three effects: a large k_s means the cloth is difficult to stretch; a large k_b means the cloth is stiff and resists bending; and a large k_g means the cloth is heavy.

Relaxing a single point is the process of moving it so that the energy of the cloth of which it is a part is decreased. The method used to relax a single point first finds the direction in which the point would most like to move: the direction of the negative gradient of the energy as a function of position. Then it moves the single point in that direction so that its energy is minimized.

Feynman suggests using a multigrid method to speed up the relaxation sweeping process. However, it must be used carefully to avoid distortion. He also introduces fixed points in order to forbid the cloth to move into a solid.

Weil [Wei86] considered the problem of hanging a piece of cloth by fixing some constraint locations of the cloth. The cloth is represented as a rectangular grid (u, v) of 3D coordinates (x, y, z). His method is a two phase algorithm. The first part approximates the surface within the convex hull in (u, v) space of the constraint points; that is, all the interior points are placed on catenaries. The second phase uses an iterative relaxation process to minimize maximum displacement of all the points in the grid up to a given tolerance.

Terzopoulos, Platt, Barr and Fleisher [TPBF87] use elasticity theory to describe the behavior of a deformable object. The model responds in a natural

way to applied forces, constraints, and impenetrable obstacles. The equations of motion governing the dynamics of the deformable bodies under the influence of applied forces is given by

$$\frac{\partial}{\partial t}(\mu\frac{\partial r}{\partial t}) + \gamma\frac{\partial r}{\partial t} + \frac{\delta\mathcal{E}(\nabla)}{\delta\nabla} = f(r,t),$$

where $r(a,t)$ is the position of the particle a at time t, $\mu(a)$ is the mass density of the body at a, $\gamma(a)$ is the damping density, and $f(r,t)$ represents the net externally applied forces. $\mathcal{E}(\nabla)$ is a functional which measures the net instantaneous potential energy of the elastic deformation of the body.

To create animation with this model, the motion equation is solved numerically, integrating through time. This model is active in the sense that it responds to forces and interacts with objects.

2.6 The Anthropometry Database

[7]While animation research may be content with demonstrating action on a convincing human form, there is often only a single carefully structured figure involved. Its body dimensions may be estimated or obtained by measurement of a specific individual. In contrast, engineering human factors applications have long been concerned with construction of valid ranges of human forms based on empirically measured populations such as aircraft pilots, flight attendants, or astronaut trainees. These engineering applications recognized the need for a variety of accurately scaled body dimensions to facilitate reach and fit analysis [Fet82, HBD80, KSC81, Doo82]. Unfortunately, most of these systems are either proprietary, hard-wired to some particular population, non-interactive, or otherwise difficult to use with contemporary graphical systems. *Jack*, however, permits an open and accessible database for human dimensional data. Interactive access is provided through a novel spreadsheet-like interface.

2.6.1 Anthropometry Issues

Anthropometry, the science of human body measurement, has been an area of interest throughout history [LRM88]:

> In his authoritative book "A History of the Study of Human Growth," Professor Tanner writes that the ancient Greeks, as well as sculptors and painters of the Renaissance, measured the human body to estimate body proportions and, thus, reproduce life–like images of varying sizes. Interest in absolute size developed later in the 17^{th} and 18^{th} centuries out of military concerns. The European armies preferred taller soldiers, and recruiting officers became anthropometrists. Interest in scientific study of growth and in the

[7]Marc Grosso, Susanna Wei.

relative importance of nature versus nurture in explaining human variability has been pronounced since the 19^{th} century.

The vast majority of work in "modern" anthropometry has been done by anthropologists who were studying the effects of some environmental factor on some population. While there are studies dating back to the mid– to late–1800's, more recent studies covering groups of adults (i.e. populations) from around the world are summarized in the *Anthropometry Source Book* [NAS78]. Its two volumes have become one of the foundation sources for contemporary anthropometry.

Anthropometric studies differ greatly in the number and kind of measurements selected. They all report a statistical analysis of the values of each measurement, giving at least a median with standard deviation and the maximum and minimum values. The studies typically report the above values along with intermediate values at selected percentiles of the population, typically 1^{st}, 5^{th}, 25^{th}, 50^{th}, 75^{th}, 95^{th} and 99^{th}, since body size data does not vary linearly with percentile.

Some of the data found in these studies was used in the *NASA Man–Systems Integration Manual* [NAS87], as the basis for the estimated measurements for male and female astronauts in the year 2000, using the body dimensions of American males and Japanese females. It is felt that these populations provide the maximum range in body sizes in the developed world today since the American male is among the largest of males and the Japanese female is the smallest of females. There is a growth rate factor which is used to adjust the values for projection to the year 2000.

The measurements selected for inclusion in the *NASA Man–Systems Integration Manual* were chosen to meet the various needs of NASA and were not intended to be a complete set of measurements for all purposes or for all possible users. These measurements were publicly available, however, and detailed enough to satisfy many ergonomic analysis requirements. They served as the basis for the human figure model we developed but are are not complete enough to totally describe it [GQO+89, GQB89]. Some needed measurements and data are missing; though most of the missing values can be found in the *Anthropometry Source Book*, there were a number of measurements required for our model which were not easy to track down. Where this occurred, intelligent estimates have been made based upon data values from closely related measurements (possibly from a different population) or by calculating the values from other measurements. In no case were the undefined values set arbitrarily.

2.6.2 Implementation of Anthropometric Scaling

Each body segment or structure having associated geometry contains length, width, and depth (or thickness) attributes[8]. Therefore, we require a minimum

[8] We presently ignore segment shape changes though we realize their importance for realistic animation.

of seventy–two (72) measurements to describe the physical dimensions of our human figure.

Psurfs describe the shape of each segment. Anthropometric scaling modifies the segment dimensions as well as any associated psurfs. It is very simple to change to alternative polygon models, e.g. to the detailed contour bodies, to vary detail in the segment while preserving the correct anthropometric scale. Each psurf for the various segments is stored in a normalized format where the z (length) dimension ranges from 0 to +1, and the x (depth) and y (width) dimensions range from -1 to +1. In order to display these psurfs, using either real measurements for a person or percentile measurements for some specified population, the psurfs must be scaled.

Body definition files containing the desired values can be created (or modified) by manually entering the body part names and their values in the proper format, but this is clumsy and rarely used. A superior approach uses the Spreadsheet Anthropometry Scaling System (**SASS**) which will be discussed in detail in Section 2.7.

2.6.3 Joints and Joint Limits

There are three different types of human body joints [TA75]: Fibrous, Cartilaginous, and Synovial. Of these three we are only concerned with the synovial joints (joints with joint cavities). The synovial joints are categorized based upon the shape of the articulating surface of the joint. There are seven sub–types of synovial joints found in the human body [MB77, TA75]. These subtypes are:

- Monaxial (or uni–axial) joints (1 DOF)

  a. Hinge joints. A convex surface of one bone fits in a concave surface of another bone. This joint allows movement in only one plane, usually extension and flexion, similar to that of a door hinge. Examples are the elbow joint, knee joint, ankle joint, and interphalangeal joints (joints in the toes and fingers).

  b. Pivot joint. A rounded, pointed, or conical surface of one bone articulates with a shallow depression in another bone. The primary motion of this joint sub–type is rotation. Examples are shown by the supination and pronation of the palms, atlas–axis joint (Alanto–Axial joints located at the very top of the spine), and radioulnar joint (between radius and ulna in forearm).

- Bi–axial joints (2 DOFs)

  a. Condyloid joints. These are the joints like those at the heads of the metacarpals (hand bones), i.e. the knuckles, which is the best example of this type of joint.

b. Ellipsoidal joints. The oval–shaped condyle (end) of one bone fits into the elliptical cavity of another bone. This type of joint permits side–to–side and back–and–forth movements (in the principal axes of the ellipse). Examples are shown by the flexion and extension and abduction and adduction of the wrist (radiocarpal) joint.

- Tri–axial (or multi–axial) joints (3 DOFs)

  a. Saddle joint. Both bones in this joint are saddle–shaped, that is convex in one direction and concave in the other. This type of joint is essentially a modified ellipsoidal joint and has more freedom of movement. Saddle joints allow side–to–side and back–and–forth movements as well as rotation. An example is the joint between the trapezium and metacarpal bones of the thumb (carpometacarpal joint of the thumb).

  b. Ball and socket joints. A ball–like surface of one bone fits into a cup–like depression of another bone. These joints permit flexion–extension, abduction–adduction, and rotation. Examples are the hip and shoulder joints.

  c. Gliding (or plane) joints. Bones involved have flat or nearly flat articulating surfaces. Movement can occur in almost any plane, with side–to–side and back–and–forth movements the most common. The movements are always slight. Examples of this type of joint can be found between the carpal (wrist) bones (intercarpal joints), between the tarsal bones (foot/ankle) (intertarsal joints), between the sacrum (lower end of the spine) and ilium (a hip bone) (the sacro–iliac joint), between the sternum (breast bone) and clavicle (collar bone), between the scapula (shoulder blade) and clavicle, between the individual vertebral arches, at the heads and at the tubercles of the ribs, and at the front ends of the costal (rib) cartilages.

Each joint in the human body has a range of motion over which it will allow movement to occur. A joint's range of motion is determined by a number of factors including joint type, muscle size at the joint, muscle tension (tonus) for the muscles at the joint (i.e. fitness of the person), ligament stretchability or give, amount of fatigue, and training adaptations for the joint. The term flexibility is frequently used to describe the influence that each of the components listed above has on joint movement.

Joint range of motion, described in terms of angles, is measured in degrees for each DOF, that is, each plane in which movement is allowed at a joint. When a joint has more than one DOF, the range of motion at the joint for each DOF may be variable because one DOF may influence the others. Also, for joints which are influenced by muscles crossing two joints (as in some muscles of the thigh, for example) there may be a two joint dependency on the joint limit.

Jack incorporates upper and lower joint limits for every single DOF joint. For two DOF joints, independent limits for each DOF are used. But the shoulder is treated as a three DOF system with spherical joint limits and a function that relates the default upper arm orientation to the upper arm position (Section 2.4). *Jack* respects these joint limits during both interactive positioning and inverse kinematic reaching.

2.6.4 Mass

As dynamic simulations achieve ever more realistic animation, mass information becomes essential. Fortunately, along with stature, mass is among the most common body measures taken. There have been a number of studies which have determined that each of the various body segments contributes a certain percentage of the total body mass; this percentage determines the mass of each individual segment. The mass percentages used are average percentile values for a fit male population as would be found in the NASA male crewmember trainees. The distribution may very well differ for the average general population or a population which is skewed toward either the small/light weight (like horse racing jockeys) or large/heavy weight (like American Football lineman). The segment mass percentages are also likely to be different for female subjects.

2.6.5 Moment of Inertia

The concept of moment of inertia is important when attempting to describe the dynamic behavior of a human figure. These values are needed when determining the motion of a figure under the influence of forces (both external and internal), moments, and instantaneous transfers of momentum (i.e. collisions). When considering human figure modeling the common forces and moments effecting the human figure include:

1. gravity: a force acting at the center of mass of each segment with a magnitude proportional to the segment's mass.

2. internal forces generated by muscles: forces actually acting at some insertion point along the length of the segment but modeled as a driving moment applied at the joint.

3. reaction forces generated by the figure's surroundings: for example, the normal forces and friction forces applied to the figure's hand by the surface it is leaning on.

4. external forces: for example, weights lifted by the figure, levers the figure attempts to pull, etc.

5. collisions: usually approximated as an instantaneous change in velocity of the point on the figure being struck.

2.6.6 Strength

Human strength capabilities have been found crucial for more realistic and natural human motions (Section 5.3). Human strength (maximum torques) is defined as muscle group strengths and is stored on a joint DOF basis. Modeling strength this way allows different people to possess different capacities in different muscle groups. Thus, the strength differences between two people – such as a dancer and a pianist – can be readily modeled and illustrated. Each DOF of a joint has two joint movements which are associated with two different muscle groups. For example, an elbow joint is modeled to have one DOF because it can only rotate around one axis. Its rotational movements are flexion and extension, corresponding to the flexor and extensor muscle groups. Therefore, strength data of flexion and extension are stored for an elbow joint. Each muscle group strength is modeled as a function of body position, anthropometry, gender, handedness, fatigue, and other strength parameters [AHN62, Lau76, NAS78, AGR+81, AGR+82, Imr83, CA84, HJER86, MS86, NAS87]. In terms of body position, we choose a more generalized model that takes the effects of adjacent joint angles into consideration [Sch72]. For example, the muscle group strengths of a shoulder joint are modeled to be functions not only of the shoulder angles but also of the elbow angle.

2.7 The Anthropometry Spreadsheet

[9]Given the large number of data items needed for anthropometric body sizing, a spreadsheet–like format was a natural choice for the user interface. We called it **SASS**: the Spreadsheet Anthropometry Scaling System.

SASS was originally developed with one idea in mind, i.e., generating the dimensions of each segment of a human figure based upon population supplied as input. The human model used by the current version of SASS consists of thirty-one segments (body structures), of which twenty-four have a geometrical representation. For each of those twenty-four segments, there are three dimensions which are required, namely, length, width, and thickness. This means that at least these seventy-two measurements should be available.

The psurf geometry of each segment must be scaled by real measurements for a person, or percentile measurements for some specifiable population. **SASS** generates figure files with the appropriate description of the segment dimensions and joint limits, so that *Jack* can display the resulting figure.

SASS uses population statistic data to create generic human figures. Alternately, **SASS** has a built-in database that stores anthropometric data for (real) individuals and provides an interactive query system for database access.

SASS allows flexible interactive access to all variables needed to size a human figure described structurally by a **Peabody** body file. The **SASS** screens, as shown in Fig. 2.5 and more diagrammatically in Fig. 2.6, are

[9]Richard Quach, Francisco Azuola, Welton Becket, Susanna Wei.

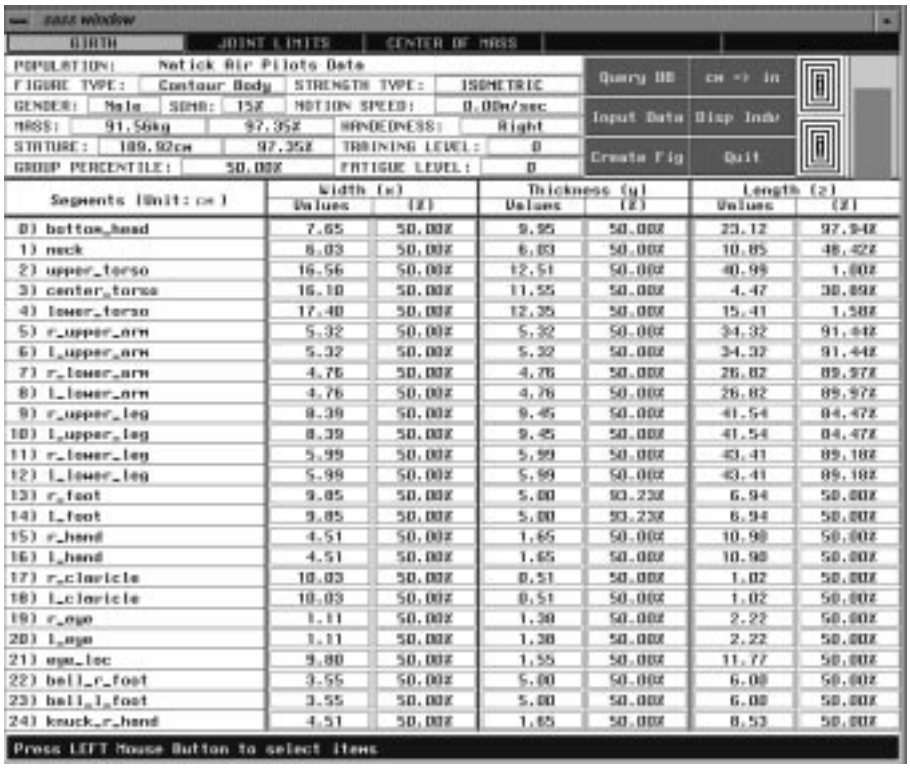

Figure 2.5: Sample Display From SASS.

divided into different sections including anthropometric group selection, global data, command menu, local data.

Data that may be accessed is organized into anthropometric "groups". The current version can handle four groups: segment girth, joint limits, segment center of mass, and strength.

The global data section of the spreadsheet is intended to allow a "whole body" view of the current figure parameters. Currently, the six items considered for any human figure are: **population, sex, figure type, mass, stature**, and **overall percentile**. It is important to realize that since **SASS** is a relational spreadsheet, modifying any data in this section will affect the values of the individual segments. For example, changing the figure's percentile will cause the data to be scaled in other appropriate segments contributing to stature.

The data section is used for the display of individual segment data and their corresponding percentiles. The leftmost column is reserved for the segment names, while the other six columns are used for the data and percentile display. The segment name column cannot be modified. The data is read in from the selected population input file.

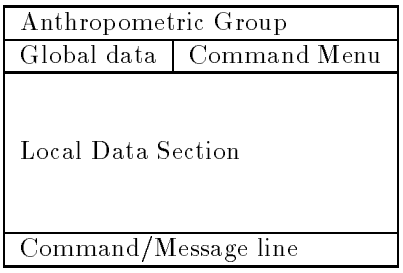

Figure 2.6: SASS Screen Layout.

Data and its corresponding percentile is modified by simply moving the locator device to the desired cell and pressing on a button. Changing any segment percentile will change its corresponding dimension. **SASS** keeps a current measurement unit type for each group (**in, cm, deg, rad**). Unit conversion is performed when necessary.

2.7.1 Interactive Access Anthropometric Database

An anthropometric database stores attribute data for a specific set of individuals rather than a population. Each person has associated groups of anthropometric data: girth (segment dimensions), joint limit, center of mass, and strength. Each group of anthropometric data is stored in a separate relation.

In the application of a task simulation, it is very important to find an individual with the requisite anthropometric characteristics such as body dimensions, joint limits, and strength. **SASS** provides a query system using pop-up menus to allow the user to select the people with the desired characteristics from the anthropometric database. Therefore, the user does not need to know the database structure nor its query language.

The user can query on different anthropometric characteristics separately or in combination using operations *and, or, greater than, equal to, less than,* etc. For example, the user can inquire about people with the desired strength capabilities alone, or query about individuals with the required strength capabilities, body dimensions, joint limits, and center of masses together. The individuals that satisfy the query and their global data are stored in a list called the *query list*. After examining the global information in the query list, the user can choose all or some of these individuals and store them in the selected list. The detailed anthropometric data of each individual in the selected list can be displayed on the anthropometric spreadsheet. If desired, the user can also create the **Peabody** structure files for those selected individuals by using the *SASS* command Create Figure.

2.7.2 SASS and the Body Hierarchy

Our first version of **SASS** handled segments by gathering them in a simple list. This was good enough to represent any number of segments, but presented some inconveniences in defining relations among the segments. In a substantial modification, the structure was changed to a hierarchical tree. At the bottom of the tree the leaves correspond to the segments. The internal nodes correspond to conceptual body parts composed of sets of actual body segments, for example, the "arm" consists of the upper arm, lower arm, and hand segments. Thus concepts such as the "length of the arm" can be well-defined. A figure can be defined as a collection of body parts, joined together by joints. Each body part, in turn, can be defined as a collection of body segments connected by joints.

2.7.3 The Rule System for Segment Scaling

The introduction of the body part hierarchy permits **SASS** to determine and use attributes of body parts as well as individual segments. For example, **SASS** defines a rule for computing the height of an individual as the sum of the segments' lengths in a path that goes from head to feet. For those segments in the path, the rule allows varying their lengths if the stature changes and, vice versa, to change the stature if the length of any of the segments in the path changes. There is an alternate rule that keeps the stature fixed and adjusts the segments' lengths accordingly, if the length of one of them varies. Another rule includes changing the mass according to the stature and, conversely, changing the stature according to a specific mass value.

The underlying criterion for doing the stature changes is a linear one. The segments in the stature path are head, neck, upper torso, center torso, lower torso, upper leg, lower leg, and feet. The length of each of these segments, except for the feet, is computed as the length in the z coordinate. For the feet, the length is the y coordinate (since for the feet, the z coordinate is the longitudinal dimension). The thickness and width of the segments are not affected by these changes, for there is no rule to decide the effects of stature changes in these parameters. The updating process must be done carefully, for it might happen that modifying the length of a given segment violates the range of possible stature values admitted by the current population or conversely, if the stature is changed, it might not be satisfiable by variations in the segment lengths.

The other case considers fixed stature. The idea is to adjust the segments' lengths along the stature path if the length of one of them varies, such that the global length (stature) remains constant. While this might appear easy to do at first, it is not a trivial matter. Recall that segment dimensions are based on population percentile data suitably interpolated and are therefore restricted to legitimate ranges. Furthermore, the stature itself is restricted by a "real" set of values (for each of the percentiles). When the user specifies a particular change in the length (or other dimension) of a given segment, the underlying

rule attempts to satisfy the constraint of fixed stature, that is, it tries to keep the stature value constant. For example, assume the length of the head has decreased. To keep the stature fixed, the lengths of the other segments in the stature path must vary in the opposite way. Currently, the modification is done in a linear fashion since there are no rules to define this otherwise. But it might be the case that in the updating process one of the segment's dimensions (length) cannot be satisfied, that is, the resulting dimension is out of the range established by the $5 - 95^{th}$ percentile values. In this situation, the rule sets the length to its closest limit (5^{th} or 95^{th} percentile value) and tries to satisfy the requirement of fixed stature by modifying the remaining segments in the path. Notice that there is a possibility that the stature cannot be kept constant. There is one more step involved in the updating process for fixed stature: if the stature is varied by the user the segments change correspondingly if possible.

It is important to understand the back and forth process that goes on between body parts and segments. If the overall body is supposed to be 50^{th} percentile, the body parts need not all be 50^{th} percentile. In fact, we do not have a rule yet to specify the percentile of the body parts (segment-wise) for a given global body percentile. That information must come from multi-variate analysis of population data. So we must be able to change dimensions of the body parts or segments to comply with all the possible *valid* compositions of a 50^{th} percentile body. If the stature is modified, then a new global percentile is computed. For that new global percentile, we have a specific rule telling us what the possible compositions are. Unfortunately, the compositions are not unique since they depend on the population data used. To illustrate this, suppose we have the following (partial) composition set percentiles: feet 30%, legs 45%, torso 60%, head 40%,... for a 50^{th} percentile body. Then suppose we want to change the stature in such a way that the resulting body percentile is 60^{th} percentile, and the analogous (partial) composition set is (feet 40%, legs 56%, torso 50%, head 40%,...). Then we scale the objects in the stature path (which are those listed in the composition sets) to comply with this second composition set. But we must be sure that there is no conflict in doing so; for instance, the feet might be able only to grow from 30% to a 40% under the given population. In general, different populations will have different compositions.

In this example, the compositions were stated at the body part level. There must be an equivalent composition at the segment level: the segment version of the composition for the 50^{th} percentile figure is, for instance, upper leg (45%, lower leg 60%) assuming legs decompose in two pieces. But what if the compositions, even though being based on a particular population data, are not available for all the possible percentiles? We would have to use a fixed composition or else interpolate compositions (if it is sound to do that) and make sure a given segment's length is not violated (according to its percentile range) when trying to go from a composition for the 50^{th} percentile figure to that of the 60^{th}.

Anthropometrists who argue against the cavalier usage of percentiles in

characterizing body size distributions would seem to be on pretty safe ground. Since the only obvious alternative involving enumerating all the individuals in a database we are stuck with the population statistics available. As multivariate population dimension distribution data becomes available, **SASS** will be ready to utilize it for proper figure creation.

2.7.4 Figure Creation

SASS can produce a file containing the scaling of a figure which *Jack* then interprets to create the figure file. Scaling files offer a particular benefit over direct creation of the figure in **SASS**. Consider the situation in which the *Jack* user wants to determine the percentile ranges of a figure that can satisfy a given task, that is, the problem of finding the specific figure percentile (or range) that can fit in a particular workplace. One can attempt to read each of the possible figure files out of *Jack* libraries and try to keep the figure in the desired position. The other, more sensible, option is not to load different figure files, but instead, to load different scaling files. Then the same figure can be scaled using all these different files to find the one that best suits the given environment. This is faster and more appealing to the user.

2.7.5 Figure Scaling

The scales in the figure scaling file are obtained directly from the dimensions of the body segments or parts. Thus, this scaling file represents the dimensions specified in the (population's) girth file, for a given figure percentile. If there were only one scale factor for each dimension of a segment, there would be some scaling mismatches towards the ends of the segments. This is in fact a major problem, especially for the low resolution polyhedral body model. For instance, the scaled upper leg appears to be too thick and too wide, in comparison to the lower leg. The same problem occurs with the upper arm and the lower arm. The scaling of the pelvis of the simple polyhedral body seems to be too wide and thick, while the torso appears to be too narrow and short. There are various solutions to these problems. The simplest one is to adapt the data to the model by modifying the scaling factors to obtain a good (looking) figure scaling. There are no strict rules for this, though. The rule we use is to consider body lines as second order continuous curves. There are no abrupt changes from one body part to the next one (assuming no deformations). Thus we approximate (in a rather arbitrary way) the scaling factors to achieve this continuity. The largest discrepancies are the ones mentioned above. Other minor ones are the scaled neck being too wide and hands being too narrow. In general, the scaling factors are not changed by more than 10% in the simple polyhedral *Jack* figure.

The scaling factors generated by **SASS** are mapped into the contour body with almost no modifications necessary (for the contour figure case, adjustments are done to the torso, legs, arms, and hands). These adjustments do not go over 5% of the actual values. Again, one must keep in mind that even

though the contour model is a more accurate representation of the human body, it is not a perfect one. Moreover, we must remember that the **SASS** scaling factors file is created based on a given population and the figure resulting from that scaling might not completely match a real human being (for suppose that the population's average torso length is greater than the torso length of a given individual and the population's average leg length is smaller than that of the same individual, then we end up with a not so real scaling for the contour model). Thus, even though we have assumed some adjustments are required, it is still necessary to prove if this is the right way to proceed. So far, the criterion that prevails is to display a good-looking (well proportioned) human figure.

2.8 Strength and Torque Display

[10]Human strength information has applications in a variety of fields. In physical education, it can be used to classify participants for specific activities or to search for a body position or motion which will improve an athlete's performance; in medicine, it can be used as an indicator for a muscular injury or disease; and in ergonomics and human factors, it can be used to guide the design of workspace or equipment to reduce work related injuries, and to help in personnel selections to increase work efficiency [MKK+88, McD89, Eva88, CA84]. Human strength information can also be used in a human task simulation environment to define the load or exertion capabilities of each agent and, hence, decide whether a given task can be completed in a task simulation [EBJ89].

To convey the multi-dimensional characteristics of strength data, we use special data display methods that use human figures together with two or three dimensional graphics. Various forms of *strength box* and *strength bar* displays allow visualization of strength parameters, give a dynamic changing view of the effects of parameters on strength, and show safe and forbidden regions in terms of different strength capabilities of individuals or populations [WB92].

We define strength (maximum torque) to be the maximum achievable joint torque. It is modeled as muscle group strength and is stored on a joint DOF basis in the human figure model. Each DOF of a joint has two joint movements which are associated with two different muscle groups. For example, an elbow is modeled to have one DOF. It can only rotate around one axis; its rotational movements are flexion and extension which correspond to the flexor and extensor muscle groups. Therefore, strength information of flexion and extension is stored for an elbow joint. Each muscle group strength is modeled as a function of body position, anthropometry, gender, handedness, fatigue, and other strength parameters [AHN62, AGR+81, NAS87, HJER86, Imr83, Lau76, NAS78].

[10]Susanna Wei

2.8.1 Goals of Strength Data Display

Strength depends on a set of parameters. To properly display these parameters, one should use a multi-parameter system that can convey the meaning of these parameters as well as their values. Unfortunately, existing multi-dimensional display methods [And60, Ber83, GFS71, Har75, And72, Che73, KH81] are unsuitable in presenting the human body-specific semantics of strength parameters. They are also visually ineffective in displaying strength data because they fail to tie the parameters to the body properties they describe. We prefer to define multi-dimensional graphing methods which use the body itself as a context for the parameter display.

Strength data displays show the current force and torque capabilities of a figure in a given posture or motion. A good strength data display should show the direction and magnitude of forces and torques. Since strength depends on many parameters, for example, body posture, body size, gender, fatigue, training, and handedness, it is useful to observe interactively how the change of each parameter affects strength. Finally, a strength data display should show the comparative capabilities of different individuals or populations.

2.8.2 Design of Strength Data Displays

An effective strength data display conveys the meaning of the parameters and their values [Eva85, CBR, EC86, EPE88, MKK+88]. To design displays that can portray the multi-dimensional nature of strength effectively, we use the human figure as a context [Wei90].

The *Jack* strength data display system evaluates the strength fields in the current figure's body definition file. The strength data for the people who are related (in the database sense) to the given figure displayed on the screen can be obtained by querying the strength database. For example, suppose Fred is a NASA crewman. The required joint strength data for Fred can be calculated from the equations stored in the strength fields of Fred's body definition file. However, any strength data for other people in the NASA crewmen population must be queried from the strength database. If end effector forces are not stored in the strength database, they can be calculated.

Strength Box Display

Six orthogonally intersecting rods are used to show the end effector forces in different directions at a given body posture. They can also be used to show muscle group strengths of a three DOFs joint at a given body configuration. The direction of the given force (or strength) is shown by the position and the orientation of the rod. It is also indicated by the text at the side of rods. The length of each strength rod shows the magnitude of the force/strength value at a given direction of movement. As the body configuration changes, the rod lengths are changed to reflect changing strength.

The strength rods in Figure 2.7 show the hand forces at **up, down, left, right, push,** and **pull** directions of the displayed human figure at the cur-

rently displayed body configuration. The internal box represents average female strength while the outer box represents average male strength given the same posture. The length of the interior segment of the rod represents the average strength value of females and the length of the exterior segment of the rod indicates the difference between the female and male average strengths. Thus, the average male strength is represented by the length of the whole rod. The box itself is used to show the boundary of the maximum forces of the range of motions in six different directions. The length of the red line attached to the end of each rod shows the difference between the maximum force and the force of the current body position shown on the screen at a given direction. The resulting display of the strength rods and the box is called the *strength box* display. The user can find the body configuration associated with the maximum force for the entire range of motions at a given direction by interactively adjusting the body posture until the end of the given rod touches the corresponding edge of the box.

This strength box display can be modified in many ways to show other strength or force data. For example, it can be changed to display data for a one or two DOF joint or to show the effects of different parameters:

- Two collinear rods can be used to display the strength data for a one DOF joint, and four coplanar rods can be used to display the strength data of a two DOF joint.

- Each rod can be divided into one or more segments by using two different colors to show the comparative muscle group strength of a strength parameter that has one or more values, respectively. This is shown for a male/female comparison in Figure 2.7. Strengths corresponding to dominant and nondominant hands can also be shown in this fashion. A three-segment rod can be used to show strengths of three different percentiles of a population.

- A two-segment rod can be modified to show the maximum strength and required torque of a joint. For a given load applied at an end effector, we can calculate the required torque via static force analysis at each joint in the active joint chain.

- A trace can be incorporated in the strength display to show the movement path of an end effector. Two different trace colors can be used to show the safe region where the maximum strength (of all joints) is greater than the required torque (green), and the forbidden region where the maximum strength (of at least one joint) is less than the required torque (red). (Figure 2.8). The number "20 lbs." written on the cube at the end effector is a reminder of the current load.

The strength box display is mainly designed to show the strengths for a "single" joint or the forces of "one" end effector. Although it can also be used to display strengths for more than one joint by showing them in two or

Figure 2.7: Strength Box Display of Hand Forces for Males and Females of a Population.

more strength box displays, it may not be easy to compare the values across different boxes. To effectively display strengths for more than one joint or end effector, we use the strength bar display.

Strength Bar Display

The *strength bar* display is used to show forces of end effectors or strengths of joints in a given body chain. Figure 2.8 illustrates a strength bar display that shows the maximum muscle group strengths and required torques of joints in the highlighted body chain when the hand is holding a load of 20 lbs. If the maximum muscle group strength is greater than the corresponding required torque, the interior segment of a bar shows the required torque in red. Otherwise, the interior segment of a bar shows the maximum muscle group strength in green. The exterior segment of a bar is always used to show the difference between the maximum strength and the required torque. If the maximum strength is greater than the required torque, the exterior segment is shown in green, otherwise it is shown in red. The bar (joint) with the required torque exceeding the maximum strength can be indicated by

Figure 2.8: Strength Bar Display for Maximum Strengths and Required Torques.

highlighting. A one-segment bar in green indicates that the required torque is zero; the length of the green bar shows the value of the maximum strength. Similarly, a one-segment bar in purple indicates that the required torque is equal to the maximum strength.

The strength bar display can also be modified to show strength or force data in different applications. We list some simple extensions in the following.

- Similar to a rod in the strength box display, each bar in the display can also be divided into a number of segments by using various colors to show strengths corresponding to different values of a parameter.

- Multiple viewports can be used to display the strengths associated with different values of a parameter. For example, we can use two viewports to show the strength bar displays for the dominant and non-dominant hand of a given individual. Comparing strength values from different strength bar displays is as easy as comparing strength values within a strength bar display because the 2D screen location of the display does not affect the visual length of each bar.

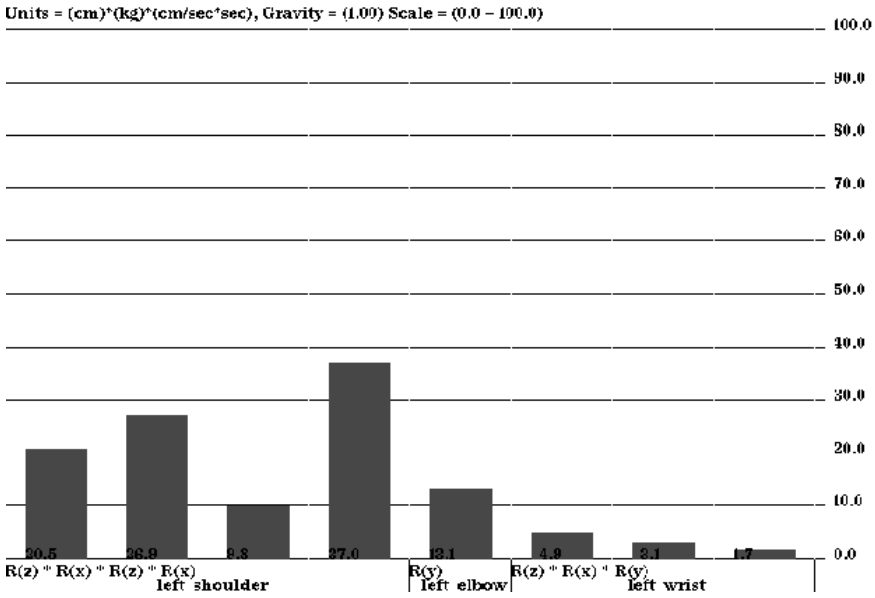

Figure 2.9: Torques Computed by Static Analysis.

The strength bar display can be used to show strengths of any number of joints in a body chain. It gives a very clear view of the simultaneous effects of the body configuration on several muscle group strengths. It also shows the safe and forbidden regions of each joint in a body chain. This display method does not depend on a particular strength model: it only shows whatever data is currently available for the figure.

Whole body strength and torque display

[11]Using static analysis, torques throughout the body may be computed given any posture and any loading condition. The strength bar display can show the individual joint torques (Figure 2.9). If suitable strength data for each muscle group is available, then whole body loading may be assessed.

Of course, with so many joints the interpretation of multiple graphs becomes difficult. To alleviate this problem we map the strength or torque data directly onto the contour body (Figure 2.10)[12]. The mapping interpolates a given value at a joint onto the adjacent contour body polygon vertices. A parameter controls how far from the joint the mapped color will propagate. A typical color scale is based on mapping zero torque load to white and maximum strength to blue. Reacted torques exceeding the joint's maximum strength are colored red. Since only one attribute can be selected for

[11]Hyeongseok Ko, Susanna Wei, Michael Kwon

[12]This will also work for the simpler polyhedral body but is much less interesting.

Figure 2.10: Contour Body with Color Coded Segments Representing Torque Load.

a joint, the maximum load at multiple DOFs is used to select the color. As the mapping and color interpolation across each polygon take advantage of the Silicon Graphics workstation display speed, the visualization may even be interactively observed as the posture and loads change.

Chapter 3

Spatial Interaction

This chapter describes the basic architecture of the *Jack* interactive system. The primary tools available to the *Jack* user involve direct manipulation of the displayed objects and figures on the screen. With articulated figures, movement of one part will naturally affect the position of other parts. Constraints are used to specify these relationships, and an inverse kinematics algorithm is used to achieve constraint satisfaction. As a consequence of user actions, certain global postural manipulations of the entire human figure are performed by the system. This chapter presents the direct spatial manipulations offered in *Jack* and shows how constraints are defined and maintained. One particular application of the body constraints is included: the generation of the reachable workspace of a chain of joints.

3.1 Direct Manipulation

3D direct manipulation is a technique for controlling positions and orientations of geometric objects in a 3D environment in a non-numerical, visual way. It uses the visual structure as a handle on a geometric object. Direct manipulation techniques derive their input from pointing devices and provide a good correspondence between the movement of the physical device and the resulting movement of the object that the device controls. This is *kinesthetic correspondence.* Much research demonstrates the value of kinesthetically appropriate feedback [Bie87, BLP78, Sch83]. An example of this correspondence in a mouse-based translation operation is that if the user moves the mouse to the left, the object moves in such a way that its image on the screen moves to the left as well. The lack of kinesthetic feedback can make a manipulation system very difficult to use, akin to drawing while looking at your hand through a set of inverting mirrors. Providing this correspondence in two dimensions is fairly straightforward, but in three dimensions it is considerably more complicated.

The advantage of the direct manipulation paradigm is that it is intuitive:

it should always be clear to the user how to move the input device to cause the object to move in a desired direction. It focuses the user's attention on the object, and gives the user the impression of manipulating the object itself.

3.1.1 Translation

Several techniques have been developed for describing three dimensional transformations with a two dimensional input device such as a mouse or tablet. Nielson and Olson [NO87] describe a technique for mapping the motion of a two dimensional mouse cursor to three dimensional translations based on the orientation of the projection of a world space coordinate triad onto the screen. This technique uses a one-button mouse, and it compares the 2D displacement of the mouse cursor to the screen projection of the six world coordinate axes and causes a differential movement in world space along the axis whose projection is closest to the mouse movement. For example, if the view is positioned such that the world coordinate x axis points left, then moving the mouse to the left will cause a $+x$ translation. This provides good kinesthetic correspondence, but it has problems if two of the axes project onto the screen close to one another, since it will not be able to distinguish between the two. In other words, it is highly dependent on the view.

3.1.2 Rotation

Rotations are considerably more complex, but several techniques have been developed with varying degrees of success. The most naive technique is to simply use horizontal and vertical mouse movements to control the world space euler angles that define the orientation of an object. This technique provides little kinesthetic feedback because there is no natural correspondence between the movements of the mouse and the rotation of the object. A better approach, described by Chen et al [CMS88], is to make the rotation angles either parallel or perpendicular to the viewing direction. This makes the object rotate relative to the graphics window, providing much greater kinesthetic feedback.

The problem with screen-space transformations is that it is impossible to make movements around either the global or local axes. In an integrated geometric environment, it is more common to move objects relative to either the global or local coordinate frame, rather than along axes aligned with the screen. For example, the simple task of raising an object vertically requires translating along the global y axis. Unless the view in the graphics window is perfectly horizontal, the vertical direction in screen coordinates is not exactly vertical. As another example, the task of moving a hand forward may require moving along an axis aligned with the body, not the screen.

Evans, Tanner, and Wein [ETW81] describe a rotation technique that suggests a turntable on which objects sit. Movements of the mouse in circles around the origin of the turntable cause the turntable, and thus the object,

to rotate. There must also be a way of positioning the turntable underneath the object.

Chen, Mountford, and Sellen [CMS88] also describe a technique originally developed by Evans, Tanner, and Wein [ETW81] known commonly as the *virtual sphere.* This technique simulates the effect of a trackball centered around the object's origin. You "grab" the trackball with the mouse and rotate it much as you would rotate a physical trackball with a single finger. The virtual sphere is an efficient technique for certain operations, as Chen et al verify experimentally. However, since the rotation is not confined to a specific axis, it can be difficult to rotate around a particular axis. It is nearly impossible to make precise rotations around global coordinate axes.

Chen et al describe an experimental comparison between several techniques for 3D rotation. The subjects were asked to rotate a geometric object, in the shape of a house, to align it with a similar object in a random orientation. They were measured for both speed and accuracy. The techniques evaluated included several angle-based rotations with and without kinesthetic correspondence, and the virtual sphere. The studies generally showed that the virtual sphere was the best, out-performing the others in both precision and speed.

The virtual sphere is good at "tumbling" objects, when the path of their rotation is not important. This may be the case for objects floating in space. However, in an integrated modeling environment, the technique has some limitations because it does not allow constrained movement. Because of its free-form nature, it is very difficult to rotate an object around a single axis at a time, global or local, which is often required. For example, to turn around a human being standing on the floor requires rotating only around the vertical axis. With the virtual sphere, it is nearly impossible to rotate precisely around only one axis at a time.

An improvement over the virtual sphere is proposed by Shoemake [Sho92]. His "ArcBall" approach uses the visual correspondence between arcs on a hemisphere and 3D quaternions to define simultaneously both a rotation angle and axis. Any 2D screen point input has a well-defined rotational value. The ArcBall appears best for tumbling objects, but a constrained axis formulation is essentially similar to the *Jack* local rotation operation described below.

3.1.3 Integrated Systems

Bier's *snap-dragging* technique [Bie86, Bie87, Bie90] simulates gravity between objects and takes advantage of surface geometry to constrain and control object movements. The user first positions *jacks* in space using a 3D cursor called the *skitter.* The jacks are coordinate frames that serve as anchors for other operations. The skitter slides along faces and edges, controlled by the mouse or through dials. The technique determines the position and orientation of the visible surface beneath the mouse in order to control the position and orientation of the skitter. Jacks can be placed with the skitter and then used to specify rotation axes or end-points for angles.

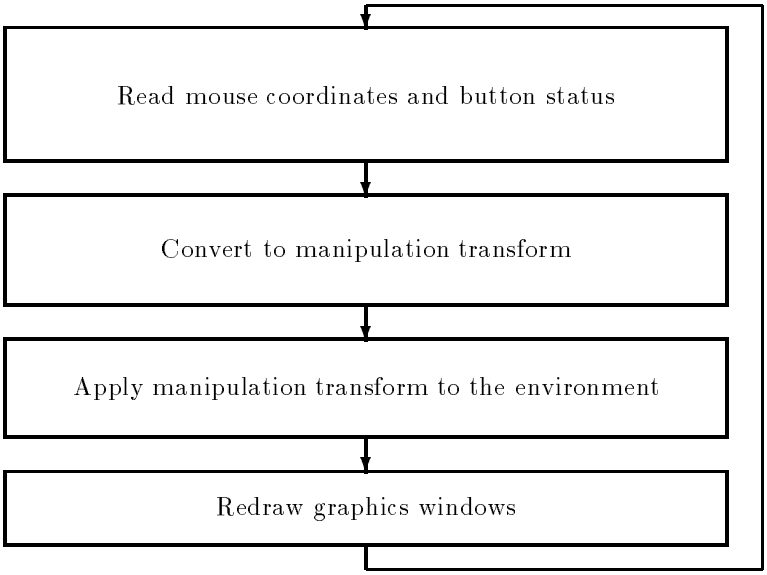

Figure 3.1: The manipulation loop.

This technique exploits the geometric structure of the objects, but it provides little help for manipulating positions and orientations in the absence of geometry. This means that the technique does not work especially well for manipulating points in open space as is often required.

3.1.4 The Jack Direct Manipulation Operator

The 3D direct manipulation mechanism in *Jack* interactively describes a global homogeneous transform. Internally, this is called the *manipulation transform.* There are many different commands in *Jack* that require the user to enter three-dimensional translational and rotational quantities. Each command may interpret the transform in its own way, possibly mapping it to a local coordinate system.

The user manipulates this transform through the 3-button mouse, together with the `SHIFT` and `CONTROL` keys on the keyboard. The keys alter the interpretation of the mouse buttons. Each mouse button corresponds to an axis in space, using a mapping scheme described below. The direct manipulation mechanism can alter the manipulation transform based on the selected axis by rotating around it, translating along it, or translating in a plane perpendicular to the axis. This characterizes the three primitive types of direct manipulation: linear translation, planar translation, and rotation [PB88]. The manipulation procedure is a loop, shown in Figure 3.1, that continues until the user terminates it.

The *Jack* user interface is modal: each manipulation command places *Jack* in a mode where the mouse buttons and keyboard keys are interpreted

SHIFT	CONTROL	mouse buttons	action
		left	linear transl along global x axis
		middle	linear transl along global y axis
		right	linear transl along global y axis
		left and middle	planar transl in global xy plane
		left and right	planar transl in global xz plane
		middle and right	planar transl in global yz plane
	CONTROL	left	rotation around global y axis
	CONTROL	middle	rotation around global y axis
	CONTROL	right	rotation around global z axis
SHIFT		left	linear transl along local x axis
SHIFT		middle	linear transl along local y axis
SHIFT		right	linear transl along local y axis
SHIFT		left and middle	planar transl in local xy plane
SHIFT		left and right	planar transl in local xz plane
SHIFT		middle and right	planar transl in local yz plane
SHIFT	CONTROL	left	rotation around local y axis
SHIFT	CONTROL	middle	rotation around local y axis
SHIFT	CONTROL	right	rotation around local z axis

Table 3.1: Axis mappings for manipulation.

as instructions to move the transform in question. How the movement is interpreted depends upon the command. This mode is terminated by hitting the ESCAPE key. While in the manipulation mode, the mouse buttons and keys behave as described below.

The user interface for the manipulation operation encodes by default the left, middle, and right mouse buttons to control translations along the x, y, and z axes, respectively, of the global coordinate frame. When the user presses down any mouse button, it enables translation along that axis. When the user presses two mouse buttons, translation is enabled in the plane spanned by those two axes. With this technique, it is not possible to translate along three axes simultaneously, so pressing three buttons at once has no effect.

Rotation is signified by holding down the CONTROL key. In this case, the mouse buttons are interpreted as rotations around the x, y, and z axes of the global coordinate. Only one rotation button may be selected at once.

The user can change the axes of translation and rotation to the local coordinate frame of the manipulation transform by holding down the SHIFT key. The CONTROL key still signifies rotation, but the rotational axes are local to the manipulation transform instead of the global axes. Table 3.1 summarizes how the state of the keys and mouse buttons translates into the transform axis.

Jack relies on a set of graphical icons to inform the user about the axes of

translation and rotation. The manipulation transform is drawn as a labeled six-axis coordinate frame. The translation icon is a transparent arrow. The rotation icon is a spoked wheel. Each icon is overlaid against the objects themselves, but since they are transparent, they do not intrude too severely.

The Mouse Line

The translation and rotation of the manipulation transform is determined interactively by the ray in the world coordinates that is cast through the location on the screen where the mouse cursor lies. This line in space is referred to internally as the *mouse line*, and it can be easily computed by an inversion of the viewing transform. The mouse line serves as a *probe* into the environment with which to move the manipulation transform. This notion of a probe is useful in describing the implementation, although it is not one that is visible to the user. The user has the feel of moving the object itself.

Linear and angular displacements are computed by intersecting the mouse line with lines and planes defined by the origin of the manipulation transform and the translation or rotation axis using a scheme described below.

Linear Translation

As described above, linear translation takes place when the user presses one mouse button. The mouse may move anywhere on the screen, but the translation is restricted to the particular axis, determined by which mouse button was pressed. This axis projects to a line on the screen. The translation icon illustrates this line, and it also instructs the user to move the mouse in that direction on the screen. Ideally, the user moves the mouse exactly along this line, but in practice the mouse will not follow the line precisely. The position of the manipulation transform is the point along the translational axis that is nearest to the mouse line. Figure 3.2 shows the translation icon.

Planar Translation

Planar translation is actually somewhat simpler than linear translation because its two DOFs more closely match those of the mouse. The plane of translation passes through the origin of the manipulation transform, and is spanned by the two axes defined by the selected mouse buttons. The technique is to intersect the mouse line with the plane of translation and supply the point of intersection as the origin of the manipulation transform. This means that the object automatically goes to the location in the plane that lies underneath the mouse cursor. Figure 3.3 shows the planar translation icon.

The user can translate in the global or local xy, xz, or yz planes, but in practice the linear and planar translation techniques provide a comfortable pattern of use involving only planar translation in the xz plane. This is the horizontal plane, and it is the most intuitive to visualize. The user can comfortably translate objects in the horizontal plane using planar translation, and then raise and lower them using linear translation along the y axis.

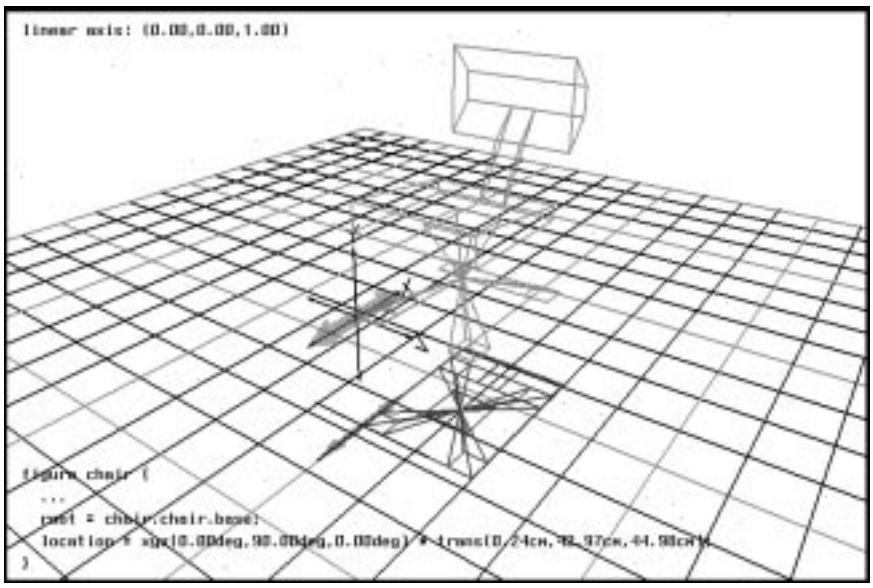

Figure 3.2: Linear translation.

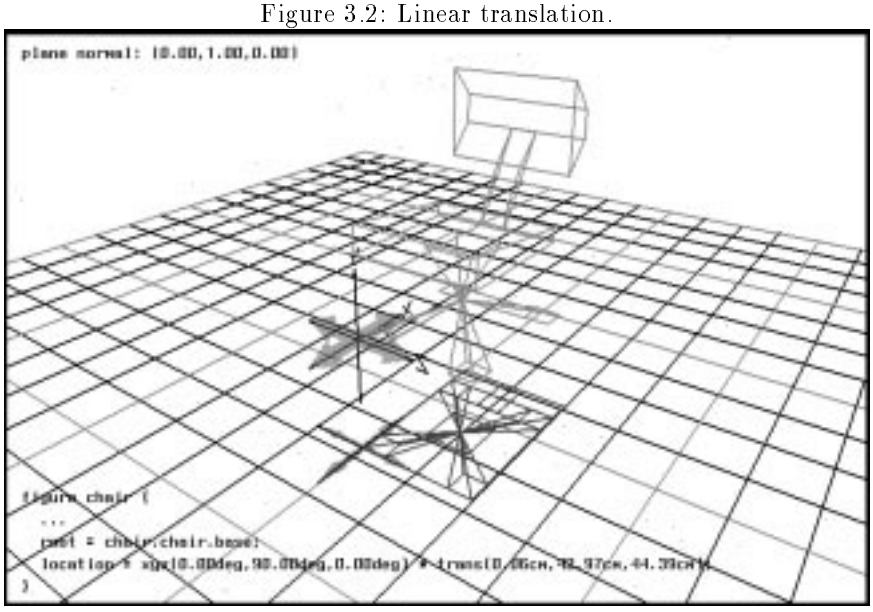

Figure 3.3: Planar translation.

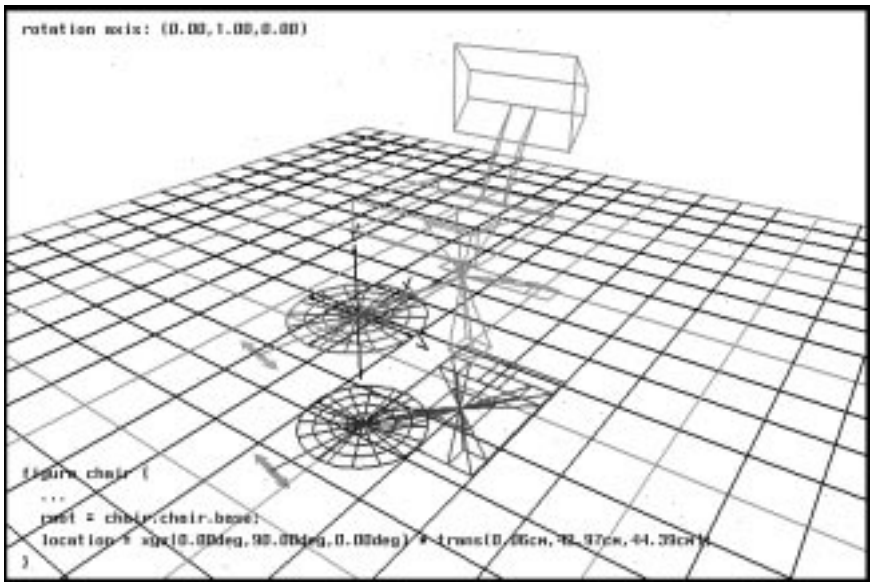

Figure 3.4: Rotation.

Rotation

The user interface for rotation requires the user to move the mouse around in circles on its pad. This is similar in some ways to the turntable technique of [ETW81] described earlier, except that the turntable used a rotation angle in screen coordinates, not world coordinates, making the angular displacement independent of the viewing direction. The *Jack* operator provides a more direct feel over the rotation because it gives the sense of actually holding on to the wheel.

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 manipulation transform 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 intersection point between the plane and the mouse line. This vector forms an extended *spoke* of the wheel, and as the user moves the mouse around in this plane, *Jack* computes a new spoke and then measures the angular distance between it and the original spoke. This determines the angle of rotation. The sensation that the user gets is one of twisting a crank by moving the mouse around in circles on the screen. Figure 3.4 shows the rotation wheel icon.

If the rotation involves a **Peabody** joint with joint limits, the inadmissible sector of the rotation wheel is colored red. Mouse movements will cause the wheel to rotate beyond its limits, but the rotated segment will not move past the limit. Thus the semantics of the user's interaction are consistent with the

free rotation case, yet object behavior is natural.

3.2 Manipulation with Constraints

The term "constraint" has many meanings and applications. Some researchers use it to mean a very low level concept. Isaacs and Cohen [IC87] use the term to mean essentially any kinematic control over a joint or group of joints during dynamic simulation. The constraint removes DOFs from the system in a straightforward way. Witkin and Kass [WK88] and Girard [Gir91] use the term to mean a global optimization criterion, such as minimum expended energy. Sometimes, the term means an desired relationship, which in computer graphics usually implies a geometric one. This is the case of Nelson's Juno system [Nel85]. Many researchers use the term to mean specifically a desired *spatial* relationship, that is, goals for reference points. This is usually the meaning in physically based modeling, as in the dynamic constraints of Barzel and Barr [BB88].

Constraints that mean geometric goals may be interpreted with an additional temporal component. Most constraint-based systems like those just mentioned feature the ability to vary the effect of constraints over time. Given this, it is rather nebulous whether the temporal component is a part of the constraint definition. We feel that it is better to view a constraint instantaneously as a static entity, although its parameters can be changed over time by some means external to the constraint itself.

3.2.1 Postural Control using Constraints

Most formulations of positioning algorithms in robotics and computer animation are principally concerned with motion — smooth motion. In robotics, this is necessary because jerky motion could damage a manipulator. In animation, jerky motion looks bad. For postural control the motion is not as important because the static posture is the principal objective. This means that for interactive postural control, it may be possible to entertain options which are not available to robotics and animation.

In particular, the requirements for postural control are, first, that the technique accommodate massively redundant figures, and second, that it perform fast enough for interactive manipulation, even with complex, highly constrained figures. The third concern is that it generate smooth movement. The interactive postural control process consists of placing the figure in a sequence of postures closely spaced in time and space, giving the illusion of motion for purposes of assisting in the postural specification process.

We use inverse kinematics for posture determination [ZB89]. It dispenses with physical interpretations and solves a minimization problem for pure functions using a nonlinear programming algorithm. The approach uses a variable-metric optimization technique. The function it minimizes is a defined through a linear combination of kinematic constraints as defined below. The objective