A High-Level Approach to the Animation of Human Secondary Movement

by

Claudia L. Morawetz

B.Sc., University of Toronto, 1984

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
April 1989

Approval

Name:
Claudia L. Morawetz

Degree:
Master of Science

Title of Thesis:
A High-Level Approach to the Animation of Human Secondary Movement

Examining Committee:

Dr. Ze-Nian Li, Chairman

Dr. Thomas W. Calvert
Senior Supervisor

Dr. Nick Cercone
Supervisor

Dr. John C. Dill
Supervisor

Dr. Romas Aleliunas
Centre for Systems Science
Simon Fraser University
External Examiner

April 14, 1989
Date Approved

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

A High-Level Approach to the Animation of Human Secondary Movement

Author: _____
(signature)

CLAUDIA MORAWETZ
(name)

April 21/89
(date)

Abstract

Animating human figures is one of the most challenging tasks in the field of computer graphics. The complexity of articulated bodies makes the integration of movement between the various joints a difficult problem. Ideally, a human animation system should provide the animator with high-level control, while still producing life-like movements. Animated actors are more believable if they display subtle gestures, or secondary movement, consistent with the characters they portray. Animators should not be encumbered with having to specify these detailed movements while they assign high level tasks to actors, such as walking or grasping an object.

In this thesis, an ideal framework for an animation system is described in which animators specify goals for actors, and these goals are executed with varying styles depending on the actor's character. A component of this framework has been implemented to demonstrate how secondary movement can be generated from a high-level specification. In this system, gesture specification functions capture information on how to execute gestures in a natural way. A graph is used as an underlying representation of movements. Using this representation, gestures can be interrupted and continued by other gestures at any time. The actors in the system are each given the goal of walking. This goal will be attained by the actors in a wide variety of styles depending on the personality and moods assigned to them. Animators can thus obtain appropriate secondary movement while focusing on primary goals for their actors.

Acknowledgements

All the wonderful people I have met have made working on my thesis much more enjoyable. The graphics lab and all its enthusiastic members provided a very inspiring work environment. Many of my colleagues in computing science deserve mega-thanks for helping me pull everything together at the end.

A few people who were extremely helpful while I was working on my thesis deserve special mention. I would like to first thank my supervisor, Dr. Tom Calvert, who guided me into a fascinating area of computer graphics and who has always been very encouraging in all my endeavours. I am indebted to Chris Welman whose patience and crystal clear explanations helped me quickly learn many graphics concepts. Howard Hamilton has always been very supportive and has motivated me to continue when I thought I'd never make it! Many thanks also to Shari Beck for being a special friend and for being as enthusiastic as I about my work!

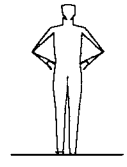
I would also like to acknowledge the School of Computing Science and Simon Fraser University which provided me with financial support in the forms of teaching assistantships and a Graduate Fellowship. I also appreciated receiving research assistantships from my supervisor and Dr. Binay Bhattacharya.

Table of Contents

List of Figures

Chapter 1
Introduction

As computer graphics becomes an increasingly popular tool for creating animations[1], ways must be found to increase the animator's productivity by making these tools more flexible and easier to use. Commercial animation systems, available from companies such as Cubicomp [Cubicomp], Wavefront [Wavefront], or Alias [Alias], guide the animator from the conception of an animation, through object modelling, motion specification and rendering to final production. However, motion specification in all of these systems is based largely on keyframing, which is often tedious. In particular, keyframing motion of an articulated body often results in awkward, unrealistic movement. Recently several methods have emerged which allow human motion to be specified at a high level, and which produce realistic movement by giving animators control over the resulting movement without encumbering them with low-level details. This thesis will examine a method for obtaining secondary movement through a high-level specification.



1.1. Background

The task of making a computer animation is simplest if the objects to be animated are rigid. As the number of articulations in the object increases, the complexity of animating the object also increases. Furthermore, if the links between joints are flexible, such as those in a worm, then producing natural movement is even more complicated [Miller 88].

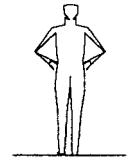
Articulated bodies can be modelled as a hierarchy of links connected at rotational joints, each with three degrees of freedom [Zeltzer 82a, Calvert 88]. A model of the human body can have over 200 degrees of freedom [McGhee 76] making control of all of these parameters over time an

[1]Words or phrases used in this thesis which may not be considered standard terminology in the field of computing science can be found in the Glossary in appendix A

enormous task. Thus presenting animators with the task of specifying keyframes does not seem to be an appropriate way for them to specify human motion. A program which takes the animator's high-level specifications and automatically generates the keyframes would be more appropriate.

A popular approach to animating articulated bodies has been to consider the dynamics of the body after applying forces and torques. This produces very realistic motion since the bodies obey the laws of physics when they move. Many successful animations have been produced where at least part of the movement is specified dynamically [Girard 85, Wilhelms 87, Bruderlin 88]. The drawback of this method is that the animator will normally have to describe motion by specifying the numerical values of the forces applied to different limbs. This is not at all intuitive, and so the appropriate numbers can only be attained by much trial and error.

Other approaches to animation have concentrated on developing a motion control language, or in general some sort of high-level interface for the user. Lower level control parameters for the movement algorithms are computed based on the user interaction at the higher level. From this idea stemmed goal-directed animation systems in which the user specifies a goal and the computer determines how to achieve that goal [Csuri 81, Drewery 86, Korein 82a]. These systems often determine how to accomplish a goal by consulting a knowledge base. A knowledge base contains specific structured information about a problem encoded in such a way as to be readable, understandable and easily modifiable. This knowledge is usually entered into the knowledge base and maintained by a knowledge engineer, whose role is to incorporate expertise about a domain into a system which uses this expertise for a particular purpose. In an animation system, knowledge engineers need not be familiar with computer graphics: they only need to understand the language in which the knowledge base encodes its expertise. The expertise, when applied to solving goal-directed movement generation, will ensure the production of realistic movement.

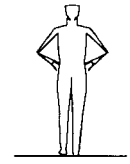


Knowledge bases have been successfully used in goal-directed animation systems [Zeltzer 83]. The animator communicates with the system through a high-level goal-oriented language. Because knowledge about how to accomplish these goals is embedded in the knowledge base, the level of knowledge required by the user is reduced.

Goal-directed systems have been successful in animating the achievement of primary goals specified by the user. However very little work has been done on applying these types of systems to character animation, in which actors' motion can be identified as certain character types. This thesis presents a method for individualizing actors' movements by providing the animator with a high-level animation system.

1.2. Proposed Work

Primary movement can be defined as the minimum movement required of a person to accomplish a predetermined task. In contrast to this, *secondary movement* is the collection of gestures a person carries out, largely sub-consciously. These gestures are consistent with the personality of that person and the moods he or she feels at a given time. Examples of secondary movement are lowering or scratching one's head, or brushing hair out of the eyes. Existing goal-directed systems are generally concerned with executing primary movement - simulating movement that will satisfy an actor's objectives. Secondary movement tends to be ignored, although if included it would make an actor's movements more realistic.

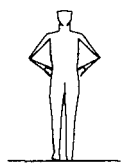


This thesis, addresses the simulation of the subtle types of movement that people display while achieving their primary objectives. The approach used is to embed algorithms that produce secondary movement in the system, relieving the animator from having to specify it. The justification for this method is that an animator would like to concentrate on the overall movement of an actor, but would still desire the small gestures that make the actor's motion more believable.

In the field of psychology, secondary movement is best equated with *kinesics*. Duncan and Fiske [Duncan 77] classify kinesics as one of the seven categories of communication in face-to-face interaction. The other six are paralanguage (the speed of utterances, and the number of pauses or interjections such as 'um' or 'er'), proxemics (the social and personal space), scent, haptics (body contact between people), use of artifacts such as a pipe or kleenex, and of course the main mode of communication, language. Psychologists have adopted a notation for recording these many forms of communication and have correlated this to character traits [Bull 83, Duncan 77, Scheflen 72, Birdwhistell 70]. The results of these studies provide a suitable basis for deciding what secondary movement to attribute to the different actors in an animation system.

Using previous work on human animation, and drawing knowledge from fields such as psychology, kinesiology, robotics and physics, an ideal framework for human animation could be created. Such a system would include a high-level user-interface where the animator defines characters, and specifies goal-directed movements for the actors. The resulting movement would be life-like. An expert system could apply knowledge from the field of robotics about path planning and manoeuvring in a constrained environment to solve for the movements required to accomplish goals specified by the animator. Expertise from psychology would be used to determine the secondary movement. Depending on the types of movement, different methods could be used for generating the movement.

The system, GESTURE, implements a portion of this ideal animation system. It has two actors, Simon and Sally, whose sole primary goal is to walk past each other as if they were passing on the street. The user can define the actors' personalities, such as how extroverted or introverted each actor is, and moods, such as the degree of boredom each actor feels. A mock expert system applies a set of rules to the values selected for the actors' personalities and moods and decides on the secondary movement. A high-level specification for these movements is defined, and from this description, an animation of the two actors is produced.



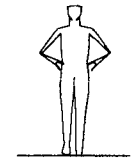
There are two primary objectives of this thesis. The first is to introduce a new type of movement to human animation, namely secondary movement, which will make the animated actors more believable. The second goal is to demonstrate that this type of movement can be incorporated into actors' movements with very little interaction required from the animator. These two objectives are met in the implementation of GESTURE as will be described in this thesis.

1.3. Motivation

At the time when hand-drawn pictures were the only way to create an animation, finding techniques for obtaining realism was an art. With the advent of computer graphics, animators could produce very realistic scenes, a task which was extremely difficult, if not impossible, with traditional animation. In particular, computer animation systems which make use of sophisticated modellers and rendering algorithms can produce convincingly real-world pictures which take less time to produce and modify than hand-drawn pictures.

One area where computer animation still cannot compete with traditional animation is when animating articulated bodies. Whether the animator spends painstaking time with a keyframing system, or uses a high-level control language, the movement seems stilted and unnatural. The film *Rendez-vous à Montréal* [Thalmann 87] combines many state-of-the-art techniques for computer animation, and yet the actors' movements still do not seem truly life-like.

Making actors' movements believable involves not only concentrating on the actor as an individual, but also on how he or she interacts with other actors in the animation. Ridsdale explored the types of movement that result when "like" and "hate" relations exist between the different actors [Ridsdale 87]. Reynolds extended this concept to flocks of birds, claiming that each bird's movements are a consequence not only of its own goals but also of the movements of the birds around it [Reynolds 87]. Based on this observation, he has produced a successful animation of a flock in motion. Extending the same idea to people, a convincing crowd scene can result where each actor's secondary movement has been considered and the influence of the actors around him or her has been taken into consideration.

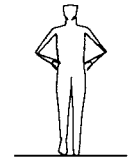


The main purpose of this thesis is to demonstrate that by incorporating body language into the actors' movements, a transformation will be seen from robot-like animations to life-like personalities. Although the GESTURE program does not consider all possible types of secondary movement, the small diverse set implemented is sufficient to show the value of this approach. Section 5.2 summarizes the variety of techniques used to produce the secondary movement.

In chapter 2, a survey of the recent literature in computer animation will demonstrate the difficulties in solving problems in human animation. Chapter 3 will present a general framework for human animation, and chapter 4 will discuss the implementation of a component of this suggested framework. The implementation will be evaluated for its success in addressing an area in human animation in chapter 5. The concluding chapter will summarize the results of the implementation and will suggest other directions to be explored in human animation.

Chapter 2
Related Research

Animated films are popular because of their ability to create fantasy worlds and situations that cannot be captured using actors and places in the real world. Talking animals, shrubs that grow to trees in seconds or a journey into someone's wild imagination become reality as thousands of masterfully drawn pictures flip before the viewer's eyes. Animating live figures is the greatest challenge as this type of motion is very complex, and the human eye is very sensitive to "mistakes" in these movements. The Walt Disney era produced some of the best animated characters because animators would spend long hours studying the movements of humans and animals that they would eventually assign to their figures [Thomas 81]. Techniques were applied afterwards to exaggerate movements which could convey humour or draw the focus to objects in a scene. For example, characters' movements can be accentuated by anticipation or follow-through at the beginning and end of their motion, or, applying squash and stretch to a ball can make it appear more bouncy.

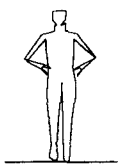


An animation sequence is usually displayed at a minimum of 24 frames/second so that the eye perceives the series of still frames as continuous motion. This requires thousands of frames to produce a few minutes of animation. In practice, the artist draws only the keyframes - the frames in which a significant change of motion or mood occurs in the sequence. The inbetween frames, the series of frames which make the transition between the keyframes, can be filled in by other painters or can be generated by a computer. This process is called keyframe animation.

Early computer keyframing efforts were based on 2-dimensional drawings. The inbetweening process was automated by having the animator specify the correspondence between lines in successive keyframes, and the computer would interpolate between the lines [Burtnyk 71]. However the results of interpolating from flat images were not always successful because there is no depth information. This led to the introduction of 3-D animation systems, in which objects and

figures in the animation are represented in a 3-D world, and a 2-D image is produced by projecting this world onto a plane [Sturman 86]. Movement in the animation is achieved by a series of rotations and translations on objects in the 3-D world. The inbetween images are more realistic because the interpolations are applied in the 3-D world before collapsing by a dimension to 2-D.

With the advantages of increased realism when moving from 2-D to 3-D come a few other problems which must be addressed. These are object modelling, movement specification and image rendering. Animation is mostly concerned with the second issue, however specifying movement for an object is dependent on how that object is represented. Animating human figures is particularly difficult because the body is a very complex structure. The human model can be thought of as consisting of a set of rigid links connected at joints, and organized in a tree-like hierarchy [Zeltzer 82a]. Movement is attained by applying 3-D transformation matrices at a joint. A matrix that is applied at a joint in the hierarchy will be applied to all joints nested deeper in the hierarchy. Connectivity of the model is assured if the transformations are all rotation matrices. Thus if a hand is to be positioned in space, the arm can never be disconnected from the body with a translation matrix; instead, joint rotations are applied at the pelvis, back joints, shoulder, elbow or whatever combination of joints required in order that the hand reach the desired position.

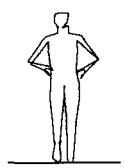


Specifying movement for the human model involves assigning values to all joint angles for every frame of the animation script. The human body has over 200 degrees of freedom [McGhee 76], so that even if a model represents only 40 or 50 of these, an enormous amount of data must still be specified. This would be an overwhelming task for an animator to determine manually, and consequently many ways have been examined for representing movement and automating some of the animator's task [Badler 79, Ridsdale 86, Calvert 88]. Animation systems can be categorized at three levels where at each higher level, the animator can obtain more complex movement with less specification. These three types of systems are guiding systems, systems which specify movement algorithmically, and task level systems.

Guiding systems are those for which there is no easy way to generalize movement specification. For example, creating a dance sequence cannot be generalized so that many dancers could be animated using the same set of moves but facing in different directions. They are synonymous

with kinematic systems, in which the position of the body and all joint angles must be specified. The film *Brilliance* is a successful example of an animation in which a robot's motion was obtained from a live actor using rotoscoping to record her movements [Abel 85]. Another way to record human movement is by using an electrogoniometer [Calvert 82]. However, this technique may cause the person to move unnaturally as he or she may feel restricted by the equipment attached to the body.

Keyframing systems represent another type of guiding system which allows animators to very quickly specify the movement for an animation sequence using intermittently spaced key poses. While the computer can save animators time by calculating position and joint angles for the inbetween frames, the animators may wish to be more involved in the interpolation process. Steketee and Badler describe a method for obtaining control over the nature of the motion that is produced from the interpolation process [Steketee 85]. Although these guiding systems give animators complete freedom over a figure's motion, this can often be a hindrance since specifying the motion at this level of detail can be very cumbersome.



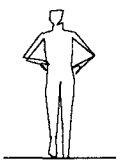
At a second level, systems can describe motion algorithmically by providing data abstraction in which graphical objects are manipulated as program types or by providing adaptive motion where objects' motion can be altered by a changing environment. ASAS [Reynolds 82] and MIRA [Thalmann 83], are examples of this type of system. The ASAS system is an extension to LISP which allows the user to define graphical objects and apply operators such as **shrink** or **local-move** to them. An object's motion can be adapted to a changing environment by the use of message passing. In MIRA data abstraction is achieved through Pascal-like objects whose values can be examined to decide on movements.

In both of these types of systems, the animator's creativity may be stifled because of the greater amount of time involved in realizing ideas instead of developing them. With human animation it is especially difficult to transfer the animator's ideas to the system quickly and easily because of the complexity of the model.

A task-level animation system helps to correct this situation because it allows the animator to

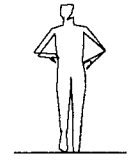
specify the actors' movements by task descriptions, and the appropriate motor programs are invoked which will produce the desired movements. This type of movement specification opens up new research areas in human animation. Given a task description, how does the computer select a reasonable set of motor programs that will perform that task in a realistic way? Now that the animator's skill in motion specification is replaced by a set of motor programs, how is believable movement attainable? Finally, what will be the trade-off between how much control the animator has on the resulting movement, and how much is automatically generated by the system?

The advantage of task-level animation systems is that they free the animator from specifying how a task is accomplished and allow the animator to concentrate on specifying what the task is. Such systems are labelled goal-directed because the movement generated is guided by user-specified goals. Badler identifies some problems which must be addressed in goal-directed human movement simulation [Badler 80]. One problem is to define a schedule for the execution of movements in the different parts of the body. This is difficult because in the hierarchical model of the body, transformations applied to joints to execute one movement could affect the transformations applied to accomplish another movement. A second problem is that of positioning the end of a limb in space. For example, a goal could be to grasp an object, which would require the hand to move to a particular location. Without any other information to constrain the problem, such as an instruction specifying whether the elbow should be held high or low, there are an infinite number of solutions to attaining this goal. This is the general inverse kinematics problem which is difficult to solve for realistic movement of multi-link structures because in general there is no knowledge about the constraints on even simple movements executed by humans. A large problem in positioning the end of a linked system is to decide what constraints to place on the system so that natural-looking movement of a limb will result. A suggested method is to provide a reach description for a linked chain [Korein 82b, Korein 82a]. In this solution, each joint in the chain is rotated by the minimum amount required for the end of the chain to reach its goal position. This solution, however, will not necessarily yield the most natural way a human would move a limb. Other problems that must be examined in goal-directed systems are situations in which parts of the body make contact with other parts of the body, maintaining balance while in locomotion, and the orientation of the body segments.



One goal-oriented system is POSIT [Badler 87]. In this system the user defines positional goals for different parts of the body. Each goal is assigned a numerical value which represents the "strength" or likelihood in attaining that goal in relation to the weights assigned to other goals. If two goals conflict, then the figure will lean towards achieving the goal with the higher strength value. One situation that was modelled was that of a figure constrained by a car seatbelt and reaching for an object. A large strength value restricts the body in the seat, and a smaller value is assigned to the goal of the hand reaching. The figure in the simulation will reach forward until the larger strength value restricts further reaching. This system demonstrates a method of moving a body in a multiply constrained environment.

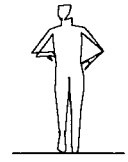
The concept of goal-directed systems has led to attempts to understanding how humans execute movements [Csuri 81]. In natural movement systems, a collection of joints and muscles controls a class of motions. Csuri describes a goal-directed system which has abstracted this idea by identifying three levels at which motions are processed. At the top level, or the task level, motions are broken down and assigned to appropriate motor programs. The motor programs oversee the execution of the motion using local motor programs, which use information about the current state of the body to produce the movements. The Skeleton Animation (SA) system implements these three levels of control to produce the movement of walking [Zeltzer 82b]. At the top level, the animator interacts with the system by providing task descriptions. At a lower level, the motor programs co-ordinate the execution of the local motor programs to produce the desired type of walk, for example the velocity, or a pattern such as a limp. At the lowest level, local motor programs control different stages of the walking cycle, for example those for a left leg swing and a left leg stance. Decisions have to be made by the task processor about which motor programs to invoke and in what order. For example, the motor program for standing must be invoked before invoking the one for walking if the figure is sitting down. The task processor organizes a schedule of motor programs by representing each movement by a frame in which there are preconditions which must be satisfied before the movement can be executed [Zeltzer 83]. This may involve activating other frames which will cause other movements to be executed. In order to produce realistic movement, the preconditions should address two issues. The feasibility of a movement is dictated by which set of actions must be invoked to correctly execute the movement. For example,



standing up, walking across the room, and reaching must all occur before grasping an object across the room. Preconditions also ensure that movements follow each other in a natural way, for example determining what distance to stop in front of a chair before sitting down. The movement frames select the preconditions that need to be satisfied by consulting a "blackboard" which stores the current state of the body.

Motor programs to execute realistic movement can be scheduled by a planning system. Drewery and Tsotsos have proposed a frame-based system which invokes a planner to choose a sequence of motions that will achieve a goal [Drewery 86]. In this system the tasks are motion verbs which act on objects. Both the objects in the world and the movements are represented by frames. The movement frames have internally defined procedures which generate the motion.

The approaches in these goal-directed systems present potential solutions for animators where a minimum amount of specification can produce life-like movement. However, it is crucial that the motor programs used by goal-directed systems do not produce movement of quality inferior to that which the animator could have achieved using some other method. Recently movement simulation using dynamics has become very popular because the results are much more life-like than kinematic simulations [Miller 88, Witkin 88]. This is because the laws of physics are applied to the moving objects. The increased realism is particularly noticeable when the objects are involved in collisions with each other, or with the ground, because the mass of the object and forces involved are considered in the movement simulation.

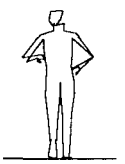


In human animation, dynamics is particularly suitable for forms of locomotion: walking and running. In the PODA system, Girard and Maciejewski explore how dynamics can be used to control the motion of multi-legged figures [Girard 85]. Others examine how dynamics can be applied to human locomotion by specifying forces which are applied to different parts of the body [Wilhelms 85, Wilhelms 87, Armstrong 85]. Human walking has also been dynamically simulated using a goal-directed approach [Bruderlin 88, Bruderlin 89]. From the results of these systems, it can be concluded that dynamic simulations can be successfully used as motor programs.

Several approaches to human animation systems have been presented. The goal of all of these

systems is to provide the animator with maximum control over the movement while requiring the least amount of interaction. Ideally the results will be so realistic that the viewer will not be able to determine if the actors are real people or were produced from an animation system. This is an ambitious goal. As well as the problem of producing realistic motion, other areas in human animation will need to be explored, such as the joint problem - how to model the body surface at a joint [Chung 87] or how to model free-hanging material such as clothing [Weil 86], in order to obtain completely realistic human animation.

The film *Rendez-vous à Montréal* [Thalmann 87] represented a major accomplishment in drawing together many areas in human animation. Using the Human Factory system, synthetic actors were modelled by a digitization process, and various animation methods were used to make the actors move, grasp objects and speak. One of the successes of the system was its capability to attribute an individual personality to each of the actors. This is a necessary requirement if an animation system is to produce believable people, as no two people are alike. Animating convincing facial expressions will help to transform robot-like figures into expressive actors [Parke 82, Pearce 86]. The rest of the body can also perform subtle movements which define a unique character for the actor.

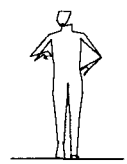


Human animation systems to date support a variety of methods for making actors accomplish primary goals. Earlier systems, such as keyframing systems, are cumbersome and tedious to use, however the animator also has control over actors' secondary movement. As the interface for animation systems has been abstracted to the level where the animator need only specify tasks or goals, animations can be created more quickly and easily with minimum specification. However, in all of these high-level systems, there is no provision for secondary movement. Each actor will perform the specified tasks and goals in exactly the same manner, regardless of their individual personalities or moods. This thesis examines a method for attributing unique body language to actors, while keeping in mind the idea of goal-directed systems where the animator will not be required to specify the movements manually.

Chapter 3
A Conceptual Model for a Human Animation System

A number of approaches to human animation were presented in chapter 2. Each method addresses one or more specific problems in human movement specification. A fully integrated human animation system could combine many of these approaches. This chapter will describe a framework for such a human animation system, and will then focus on the component dealing with the generation of secondary movement. This latter component is the main contribution of this thesis.

3.1. A General Framework

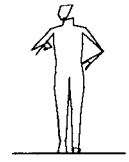


Animating articulated bodies is much more complex than animating solid objects. The human model used in our implementation has forty-four rotational joints. Considering that animations are generally played at 24 frames/second, and that two or three actors may be on the stage at one time, manually specifying all joint angles for all actors over time to produce natural-looking human movement would be a near-impossible task. Human animation systems attempt to draw the focus away from the lower level task of manually specifying angles to a higher level conceptual control, where animators can describe the desired motion in English rather than numerically.

When one shifts from a low-level to a high-level movement specification, much of the skill of the animator is replaced by the system. This could mean that many smaller movements that an animator might specify for an actor may be lost, unless the animation system pays attention to this kind of detail. It also implies that the animation system must now understand concepts which the animator would consider intuitive, such as bumping into other actors, walking through objects, or that actors must stand before walking. It is therefore natural that computer graphics has turned to the field of artificial intelligence, where research problems include understanding and reasoning about one's world. Expert systems are artificial intelligence programs which have expertise in a

particular area, such as medicine or engineering, and can solve problems at a comparable level of competence as an expert in that area. Many of these systems are used commercially to aid the human experts in their work [Bachant 84, Winston 84]. Expert systems can be applied to computer animation by incorporating expertise in areas related to human movement.

Path planning, displacing objects and interacting with a changing environment are examples of skills which humans put into practice every day without giving any thought to how they are done. While performing these tasks is considered intuitive, understanding how we identify individual movements which combine to execute tasks is an extremely difficult problem. This is one of the main problems in a human animation system that aims to provide the animator with high-level control. To animate humans realistically, animation systems become multi-disciplinary programs, calling on knowledge of human behaviour and human movement from fields such as kinesiology, robotics, psychology and sociology. The body of knowledge applicable to human animation is fragmented across these various disciplines, with each fragment contributing some constraints towards the choice of movements. Expert systems can apply these constraints to the selection of appropriate human movement.



Choosing proper secondary movement for a character requires knowledge from psychology and data about the actor's personality and moods. Researchers in the field of psychology have acquired a lot of expertise about the correlation between people's character traits and the movements they perform [Bull 83, Duncan 77, Scheflen 72, Birdwhistell 70]. However this knowledge is not well structured, and there is no systematic method for converting this knowledge into deductions about the types of secondary movement people perform. Also, specifying how the movements are to be done requires knowledge about what movements are likely to follow one another, or occur together. For example, it would not be likely for a person to shyly look away from a stranger and then wave. An expert system embodying this knowledge from psychology could assign appropriate secondary movement to characters.

Given that an expert system can help to solve problems in human animation, the animator can be presented with a very simple, yet powerful user-interface. In essence, the animator should play the role of a stage director who assigns a character and a set of motives to the actors. A good actor

should then be able to perform his or her actions while portraying the character assigned to him/her without any further direction.

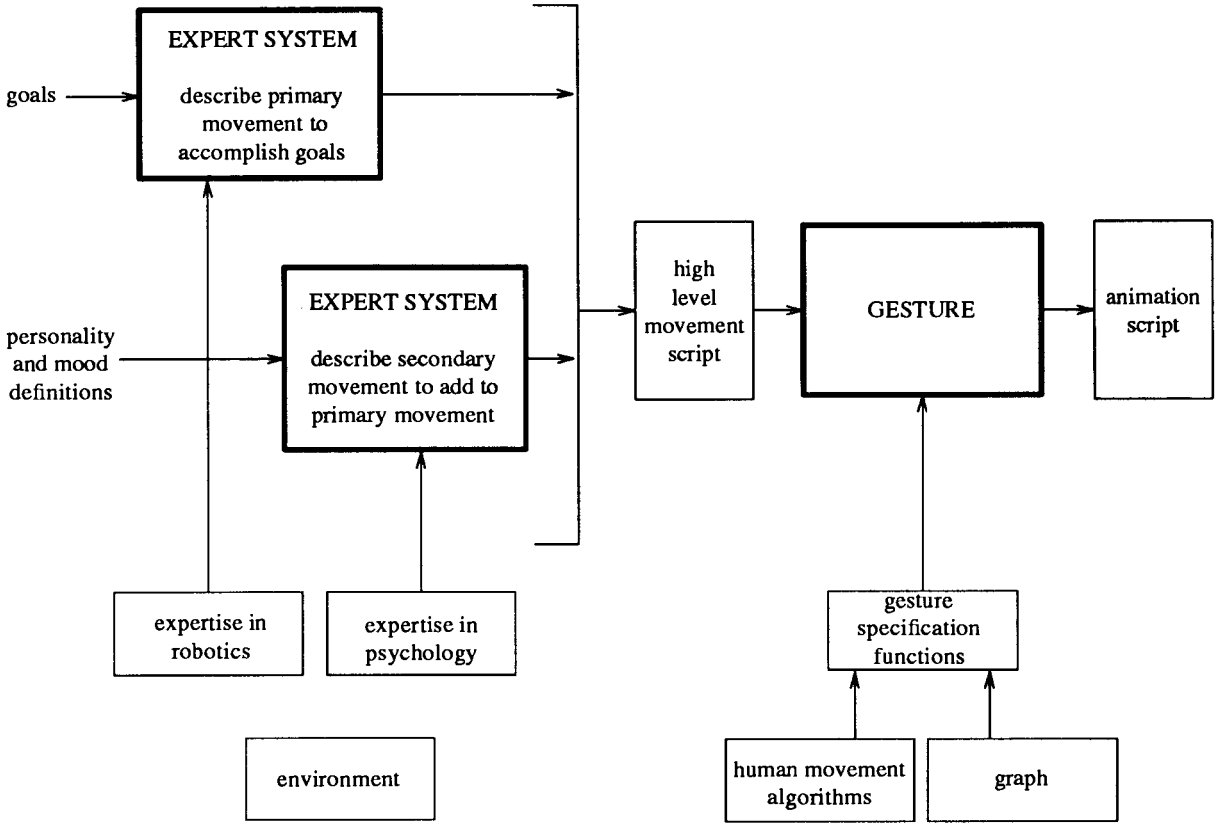


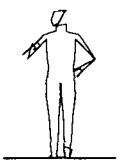
Figure 3-1: Framework for a human animation system

Figure 3-1 suggests a framework for an animation system where the animator chooses goals and defines personalities and moods for actors, and from this minimal specification a life-like animation is created. An expert system with expertise in robotics can be used to plan a series of actions in order to accomplish the high-level movement goals. Using expertise from studies in psychology which show the correlation between people's movements and their character traits, another expert system can specify the secondary movement for the actors. These expert systems will need to communicate with each other, and also use knowledge from the environment, such as where objects are placed and what are the purposes of these objects. After considering the animator's requests, the expert systems can produce a high-level movement script, which specifies primary and secondary movement for an actor. From this description, an animation script is produced by consulting gesture specification functions which have information on how the

specified movements are executed. The animation script is a complete description of all joint angles for the actor for every frame of the animation. The portion of the animation system that transforms the movement script into an animation script has been implemented for this thesis.

A clearly defined interface exists between the expert system level and the animation system level. This interface is a *movement script* for each actor written in a high-level movement language. The script contains a list of completely qualified movements to be carried out, and at what time during the animation these should be carried out.

When creating a movement script, the expert system is responsible for several tasks. One is co-ordinating the different movements that the body can perform simultaneously. For example, if the expert system requests a head scratch, and then later instructs the head to look in another direction, it is up to this system to resolve what to do with the arm. The expert system must also have some knowledge about how movements are executed. For example, if two actors walking towards each other are to stop and talk, the time to decelerate and come to a stop must be taken into consideration when specifying the time to halt in the movement script. Otherwise the actors may pass each other before stopping. Multiple actors in an animation introduce a variety of other challenges to specifying movement. Ridsdale has explored how actors' likes and dislikes for one another affect the shape of their motion paths [Ridsdale 87]. Secondary movement should be modified to account for the approach of another actor, and should consider the relationship between the two actors. The expert system must also ensure that requested movements do not conflict either with each other, other actors' movements, or obstacles in the actor's world.



Once the movement script has been created, the remaining task of the animation system is to create the animation script. This involves determining joint angles over time using movement generation algorithms to produce the movement specified by the movement script in as realistic a manner as possible. Each section of the animation system operates as an independent module, and together they produce convincing human animation.

The GESTURE program is an implementation of the component in the animation system that produces an animation from the movement script. As this thesis is concerned mainly with

secondary movement, only this type of movement is recognized in the movement script. The one exception is the primary movement of walking. Incorporating one primary goal in the implementation is helpful in justifying the secondary movement that is displayed. It is also beneficial to create a mock front-end for GESTURE as a replacement of the expert system. The user can prepare movement language scripts for GESTURE. Alternatively the user can assign personalities and moods to the actors and an appropriate movement script for the animation will be made available for GESTURE by this mock front-end.

3.2. Issues in Secondary Movement

3.2.1. Characters and Movements

In order for an actor to portray a certain character convincingly, appropriate secondary movement for that character must be displayed. Psychologists have studied people's movements, and drawn conclusions about their characters. They have also made the converse correlations which can be used in human animation systems: given a character description, an expert system can supply appropriate secondary movement [Birdwhistell 70, Bull 83].

In GESTURE, the front-end of the system allows the user to select values for different personality traits (how extroverted or introverted, cheerful or gloomy, assertive or passive, and domineering or submissive a person is) and moods (the degree of boredom, nervousness, tiredness, impatience and fear). Depending on the values chosen, actors will display varying types of secondary movement. An expert system has not been implemented to carry out the task of selecting the movements. Thus no claim is made in the implementation as to the psychological validity of the movements chosen with respect to the personality and mood definitions. However, a survey of some of the psychology literature supports some of the decisions made in selecting secondary movement in GESTURE.

One study of human behaviour was conducted by Scheflen [Scheflen 72]. He observed how different types of people interacted socially. Many of the movements implemented in GESTURE are drawn from the results of these studies. Strangers passing in the street, for example, observe a ritual of "civil inattention". At 12-15', each person will glance at the other, longer glances indicating a more sociable person, a short glance characterizing introvertedness or even hostility.

When passing someone one is familiar with, an introverted person will nod at a distance and set the head position as they pass. A less solitary person will salute or wave at a distance, and perhaps stop for a quick hello.

More subtle movements, if noted, can reveal interesting character traits. A more passive person will droop their head and hide their hands, their feet may be turned in, and they will tend to slouch. A more assertive person will have better body balance with feet slightly apart, they will be relaxed, and their body and head will be erect. A cheerful or gloomy disposition can affect the speed and bounciness in a walk. Hunched shoulders, a bowed head and hands clasped in front of the body indicate a weaker character, whereas a more aggressive posture would include hands on hips, raised head and perhaps clenched fists. The more domineering person will often step towards people, whereas the more submissive person will give way to strangers.

Some types of movements appear only as the mood of a person varies. For example fatigue will cause a person to slouch, and rub their eyes more. Nervousness introduces a variety of grooming gestures, such as rubbing one's hands together, rubbing one's chin, scratching one's head or brushing one's clothes off. Increased frequency of these gestures can signify impatience. Foot tapping and posture shifts indicate boredom. Fear is displayed by hunching the shoulders, and keeping the head up.

There are certainly many other types of secondary movement. Furthermore, the correlation between character and movements can vary in interpretation. Gestures performed in one society can have a new significance in a different ethnic group and vary between age groups [Birdwhistell 70]. The secondary movement presented in this section will form the basis of the movement implemented in GESTURE.

3.2.2. Action and Destination Movements

Movements can appear in two forms: either the movement reaches an end, such as drooping one's head, or the movement can continue for a duration of time, such as scratching one's head. These two types of movements have been classified into the two categories, *action* and *destination movements*. A destination movement involves placing a part of the body in a certain position.

Action movements involve positioning a part of the body, and then continuing to cycle through several body part positions. Examples of destination movements in GESTURE are a confirming nod, making a fist or putting one's hands on one's waist. Examples of action movements are scratching, waving or walking.

The distinction between these two types of movements is important, as they must be treated differently in the animation system. When the movement script requests the execution of a destination movement, the animation frames can be generated until that movement has been satisfied. However, if the movement script requests the execution of an action movement, frames can be generated for the body parts involved in that movement to reach the beginning of the cycle. But then there is no way of determining how many times to cycle the body positions without looking ahead in the movement script. To avoid having to look ahead in the movement script, another method must be found for deciding how many times to cycle through the movement.

In GESTURE, when an action movement is encountered, an object containing critical information about the movement is created. As the system proceeds through time in the movement script, joint angles are computed for all body parts engaged in an action movement. The end of an action movement is indicated by the beginning of another movement that would conflict with the execution of the action movement. When the movement script signals the end of an action movement, the object corresponding to that movement is removed.

3.2.3. Interrupting Movement

Algorithms have been developed for generating various types of human movement. For example, kinematics has been used to produce reach descriptions for rigid links with joints [Korein 82b, Korein 82a], and locomotion has been controlled by dynamics algorithms [Armstrong 85, Bruderlin 88, Wilhelms 87]. These algorithms can be used to provide a high-level front-end to an animation system. In a high-level animation system, an animator can specify the desired movement, and supply a qualitative description of the style in which this movement should be executed. A good algorithm will have many parameters affecting the quality of the movement, and will then compute the joint angles over time for the articulated figure, producing realistic movement.

These algorithms, although they can produce many styles of movement given their starting conditions, usually do not easily allow for a change of style, or a complete change of movement, once the movement begins. In Zeltzer's system [Zeltzer 83], actors can switch between several completed movements - walking, sitting, lying. However, the system is not able to have the actor begin to sit up and then lie down again. In this system, actors are always known to be in a certain posture after completing a movement. Determining how to proceed to another movement mid-way between two known body positions is a non-trivial task. In general this may entail solving the inverse kinematics problem, which involves examining the joint angles and then computing the most natural way to resume a known posture.

Interrupting a movement may not be such a critical requirement for primary movement, where an actor has a set of goals to achieve. However it is critical for secondary movement which usually is not prompted by well-defined objectives. Rather, an actor's character is revealed by the manner in which secondary movement is carried out including the frequency with which gestures are started but not completed. This can reflect on a person's character as much as the types of gestures that are carried out. Therefore the ability to interrupt movements and begin other movements at any time is an important requirement in animating secondary movement.

A solution will be proposed that is similar to that proposed by Zeltzer [Zeltzer 83], in that certain body positions represent known states. However new states can be created when a movement is interrupted to represent the body position at the time when the interruption occurred. Although there are limitations with this method, there is great flexibility in specifying the exact secondary movement desired and when it will begin and end. Details of how interruptions are handled using this method are discussed in section 4.3.

3.3. Secondary Movement in GESTURE

A general framework for a human animation system was presented in section 3.1. Within this framework, the problem of producing the movement specified in the movement script was identified. The GESTURE system has been implemented as a proposed solution to this problem. This section will describe the components of GESTURE that support the generation of secondary movement.

The main objective of GESTURE is to transform a high-level description of movement into realistic execution of these movements by the actors. The implementation is not concerned with whether an appropriate set of movements is selected, but primarily with executing whatever movements have been included in the script as realistically as possible. One of the issues that was addressed was the flexibility of being able to interrupt movements with other movements at any time. Thus a representation that supports this capability of interrupting movements at the same time as executing these movements in the most natural way is required.

Chapter 2 discussed research in human animation, which has produced a variety of successful algorithms for controlling the movements of an articulated body doing various specific tasks. These *motor programs* usually excel in producing life-like movement for one particular task by using specific knowledge about that task. An animation system concerned with being able to handle a variety of movements should be able to incorporate these successes.

In GESTURE, a set of *gesture specification functions* have been implemented, each one corresponding to a particular movement that can be specified in the movement script. Since each of these functions is concerned with producing the movement for only one particular gesture, they can make use of any knowledge about how that gesture is performed by humans. As each of these functions is independent of the others, existing motor programs or new specialized algorithms for particular movements can be incorporated into the animation system. Also, the technique for producing movement that is most appropriate to that movement can be used. In GESTURE, most of the movements have been generated using a keyframing system [Calvert 89], however the sole primary movement that has been implemented - walking - uses dynamics to generate the walk [Bruderlin 88] and kinematics to produce the arm swing.

The gesture specification functions fulfill the requirement of ensuring that the animation system will produce realistic motion. However, the method with which movements will be able to proceed from one to another, or interrupt each other must be resolved. A representation of movement shared by all gesture specification functions has been suggested. Movement will be encoded in a graph, and the execution of a movement will be equivalent to traversing the graph. Nodes in the graph contain joint angle values for a collection of joints. These can be looked upon as key

positions for a set of joints in the body model. Arcs are labelled with names of different movements, and a number indicating the number of frames between key positions represented by the surrounding nodes. A path along arcs of the same label represents the series of key positions to assume in order to execute a particular movement. Continuing along a new arc with a different label signifies beginning the execution of a new movement. The graph representation is a very natural way to encode movement that has been generated by keyframing, where joint angle values are not defined at every frame. Movement from an algorithm that generates joint angles at every frame can be stored in the nodes of the graph, and 0 "inbetweens" can be assigned to the arcs for that movement. In this way, movements produced from a kinematic algorithm or a dynamic simulation can also use this underlying graphical representation of movement. If a movement based on an algorithm is repeated, yet with different qualities which may alter the parameters to the algorithm, the graph may be altered (the number of nodes, and/or values of joint angles at the nodes) before the graph is traversed the second time.

The top-level control of GESTURE reads movements from the movement script and activates the appropriate gesture specification functions. The specific knowledge about how to perform movements is all contained in the gesture specification functions. These functions all use an underlying graphical representation of movement. In chapter 4, the components of the animation system will be examined in more detail, and examples from the implementation will be supplied.

Chapter 4
GESTURE Implementation

GESTURE is a stand-alone program which produces a computer animation from a description in a movement script. The movement script uses a qualitative language to describe the gestures of an actor. However there are no details as to how these gestures are to be translated into joint angles. This chapter describes how gesture specification functions are consulted about each gesture, and how a graph is used as a foundation on which to generate movement. Since GESTURE also represents a module that would receive its input from an expert system, the movement language script which is the interface between these two systems is described, and a description of how the expert system is simulated is presented.

4.1. The Movement Language Script

The movement language script is the dividing line between the responsibilities of the expert system and GESTURE, which generates the movement. This script is produced after considering the movements which an actor should do, and the order in which they should be done. A chronological list of movements is generated by the expert system which creates the movement script. The gesture specification functions can apply specialized algorithms for producing the movements from this script. Recall that the gesture specification functions each know how to execute one movement very well. It is up to the expert system to ensure that it makes sense for the movement to be executed at that time.

With the absence of an expert system in our implementation, it is up to the user of GESTURE to guarantee that movement scripts are created with a logical selection of movements. In order to fulfill this requirement, it is important to understand the syntax and semantics of the language used for the movement script.

An animation script is some form of movement specification over time. The format of the movement language script is a series of movement command lines. Each line specifies a time at which a movement should begin, the movement and then an optional collection of words which qualify the movement. The command lines must be sorted in chronological order by the specified starting time of the movement. The times on the command line represent frame numbers. Choosing another measure of time, such as seconds, was considered, however it was decided that the movement specification should have the same time granularity as the animation it is scripting. This implies that the expert system be aware of the frame rate of the final animation in order to be able to control the speed of movements.

Movements are specified by words which uniquely define the movement. Each movement can be qualified by words which will alter the way in which the movement is produced. In the absence of qualifying words, defaults are chosen. Qualifying words that may be used, vary between movements. The command line

```
15   look up
```

will cause the head to begin looking upward at frame 15. Because the movement "look" can be qualified by a direction (left, right, straight) and speed (fast, average, slow) as well as height (up, down, ahead), the defaults "straight" and "average" will be chosen for the direction and speed of the movement respectively.

```
0    walk                    walk forward
0    in_back                 place arms behind body
20   scratch left            scratch head with left hand
50   look right ahead fast   turn head quickly to the right
50   on_waist left           put left hand on waist
60   halt                    stop walking
100  nod                     nod head
120  walk                    walk forward
```

Figure 4-1: A sample movement language script

Figure 4-1 shows an example of a movement script that can be used by GESTURE to produce an animation. Explanations of the movements are in italics. A complete description of the movement script language can be found in appendix B. The example script shown in figure 4-1 is designed to portray some of the issues that were discussed with respect to the responsibilities of the expert

system. Note that the head scratch which was begun at frame 20 will stop when interrupted by the head turn at frame 50. However if the expert system had not specified the command to place the hand that was scratching on the waist, the arm would remain in mid-air. This script was intended for an actor who will stop and nod at another actor before continuing on. Although the halting process begins at frame 60, the actor will not stop moving for a while, and so the nod is scheduled to begin at frame 100, after the actor has come to a stop.

The expert system is responsible for specifying the order and timing of movements, but not for specifying the most natural way to produce each movement. Beginning at frame 20, the left arm will move from behind the back to behind the head in preparation for a head scratch. The most direct way to make this transition would be for the arm to move up along the back until the hand was behind the head. This movement would not only appear very unnatural, but is physically impossible. It is up to the gesture specification function dealing with head scratches, in conjunction with the graph, to specify a path, such as moving the hand in front of the body and then up to the head. The movement script in figure 4-1 can be logically produced without concern for how natural transitions between the movements will occur.

4.2. Producing the Animation Script

Creating an animation script from a given movement specification is the process of specifying stage position and joint angles for all the actors for a number of frames. GESTURE carries out this process, producing an animation script from the movement language script.

The body model used in our implementation is represented as a hierarchy of rotational joints between limbs. Applying a rotation to a joint higher up in the hierarchy affects the position of, and joints between, all links lower in the hierarchy. So, for example, a rotation of the whole body will displace every part of the body, whereas a rotation of the elbow joint will be applied only to the elbow, wrist and metacarpal joints of the same arm. The hierarchy of joints used in our model is shown in figure 4-2.

The animation script produced by GESTURE contains joint angles, and the body location for an actor for a determined number of frames. The control structure for reading the movement script

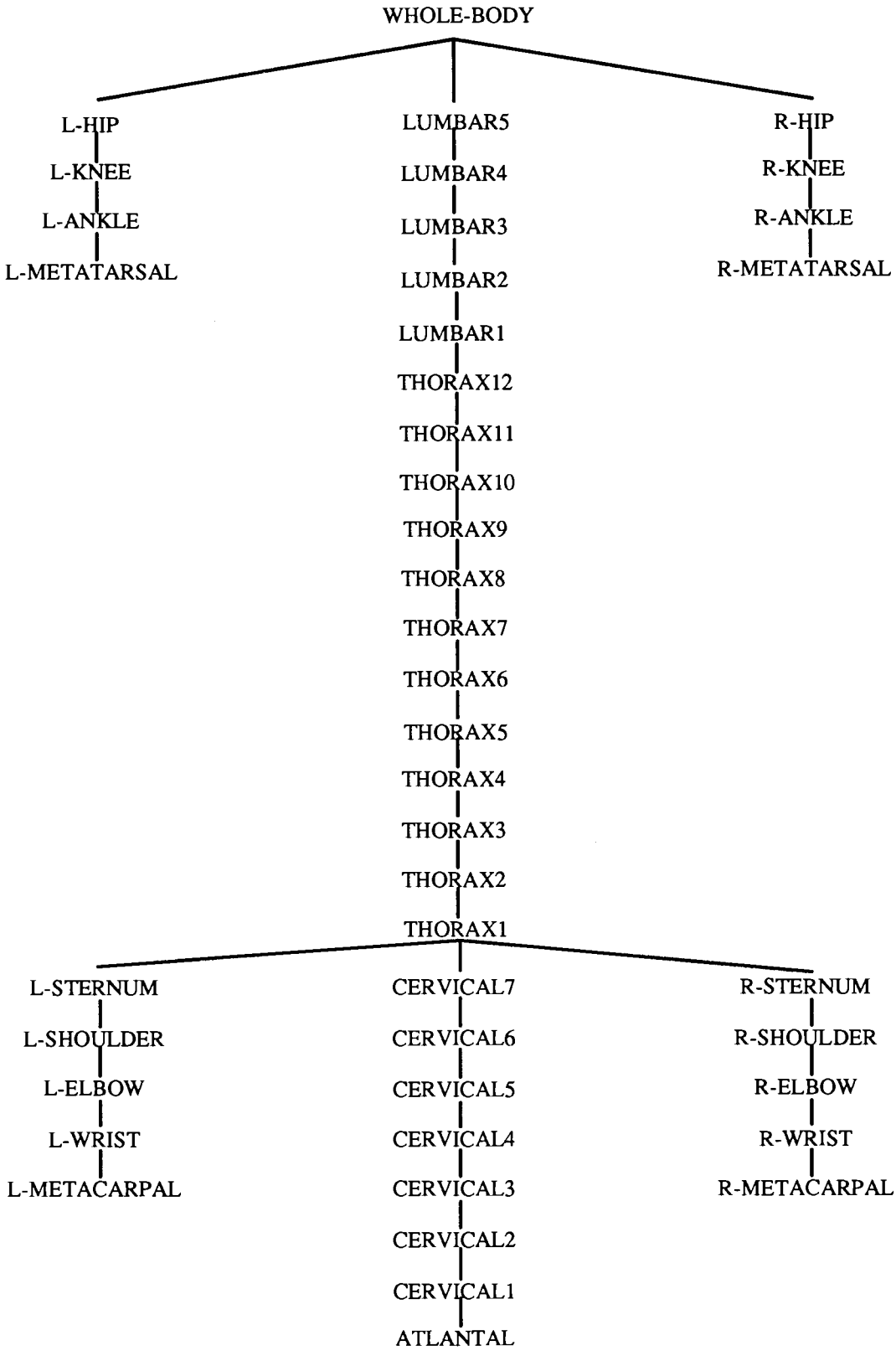


Figure 4-2: Joint hierarchy for model of body used in GESTURE

and producing the animation script is relatively straightforward as most of the work in determining what angles will produce the desired movements is performed by the gesture specification functions. However a few concepts must be explained in order to fully understand the control structure.

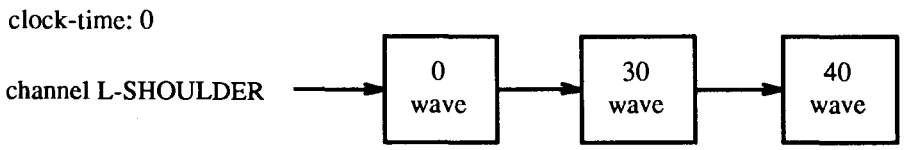
Due to the nature of the movement language script, GESTURE's control is event driven. Joint angles for the animation script are computed in sequential order through frame numbers. However time does not advance in equal units, but rather is controlled by the times specified in the movement script. These times are used to update a global clock, which represents the current clock time. This is an important reason for ensuring that the times in the movement script are in chronological order, as stated in the previous section.

Before the animation script is produced, the data must be stored in a temporary location while it is being computed. A standard approach is to use a channel table to store joint values at key frame numbers (keyframes) and then to interpolate between these keyframes to obtain values at every frame. There is one channel per joint, and three channels for the (x, y, z) location of the actor on the stage. Entries into the channel table are made by the gesture specification functions as movements are requested by the movement script. When a movement event is processed, channels may only be altered for the current clock time and future times. This means that an event cannot change the movements that occurred previous to it. It does however allow for interruptions of gestures.

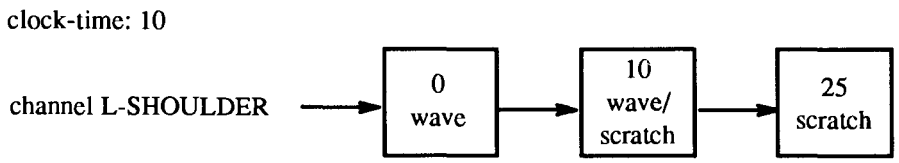
Consider the case where we wish an actor to begin a wave from frames 0 to 10, and continue with a head scratch after frame 10. The generation of a wave involves producing several keyframes for several channels. For example, if the current clock time is t = 0, then a wave could cause the placing of keyframes in the shoulder, elbow and wrist joints of one arm at t = 0, t = 30 and t = 40. When at t = 10, the movement script calls for a head scratch, keyframes after the current clock time (t = 30 and t = 40) would be removed, and new keyframes at t = 10 and t = 25, for example, could be inserted. This example is illustrated in figure 4-3. Manipulating the channel table in this way will cause an interruption of the wave by a scratch. This example demonstrates how the channel table is used in general. A gesture specification function determines how to execute a movement,

0 wave
10 scratch

(a) Movement script for wave followed by scratch



(b) Movement "wave" involves placing keyframes at current clocktime, (clocktime + 30) and (clocktime + 40)



(c) Movement "scratch" involves placing keyframes at current clocktime and (clocktime + 15)

Figure 4-3: Interrupting gestures in the channel table

and then produces keyframes in the appropriate channels at and after the current clock time, removing existing keyframes when necessary.

If the designated movement is a destination movement, control can proceed to the next movement in the script after keyframes have been generated. However, recall that for action movements, keyframes are only generated until the beginning of the cyclic part of the movement (see section 3.2.2). After this, an action object corresponding to the cyclic part of the movement is created before control proceeds to the next movement. The action object specifies how keyframes will be generated at a later time to produce the cyclic movement. These keyframes are generated as control proceeds through the movement script. At each new event, the clock time is advanced to the starting time of that event, and keyframes for all active action objects are generated extending

at least to the current clock time. An action movement ends when a movement in the script is specified which uses the same joints as the active action movement. The creation and deletion of action objects will be discussed further in section 4.4.

After the whole movement script has been processed, the channel tables contain a representation of all the movement specified in this script. One last step must be completed before an animation script can be created: values for channels at the times when there are no keyframes must be evaluated. These values will be determined by applying interpolation routines to each of the channels.

Interpolating to obtain values for inbetween frames can be looked upon as a curve-fitting problem, given a set of points. The choice of interpolation routines can affect the values computed. Interpolation algorithms based on Hermite or Bezier matrix forms guarantee first order continuity and that the curve will pass through the given points. B-spline curves may only approximately pass through the given points, but first and second order continuity is guaranteed [Foley 82]. Careful thought must be given to the method used to interpolate joint rotations. If joint rotations are represented in the Euclidean co-ordinate system, then an interpolation is done on rotations around each of the axes separately. A rotation is applied to a joint by defining an order of rotation around the axes, for example, X then Y then Z. Unfortunately, rotations represented in the Euclidean geometry can introduce Gimbal lock. This phenomenon occurs when the angle of rotation around Y is 90 °. This rotation is applied before the rotation around Z, but after the rotation around X, causing these two axes to be superimposed, and therefore losing one degree of freedom.

To avoid this problem, joint rotations are represented as quaternions and a quaternion interpolation is applied to the joint channels [Shoemake 85]. Quaternions specify an axis and an angle of rotation around the axis. Since no arbitrary axes of rotation are specified as in the Euclidean space, Gimbal lock is avoided. Also, only one channel per joint is required instead of three since the interpolation treats a quaternion as one entity.

Whereas quaternions are a natural way of representing rotations, they are not applicable to translations. Since translations are independent of order if applied along orthogonal axes, stage

location is best represented in terms of an (x, y, z) position. Thus three channels are required for specifying the location of the actor in the animation.

The interpolation routine applied to the three channels for specifying location is a spline interpolation with local tension, continuity and bias control [Kochanek 84]. This interpolation method allows for much control over how the interpolation is carried out. The tension parameter controls how sharply the curve bends at a key position. The bias controls the direction of the path of the curve as it passes through keyframes. The continuity parameter varies the amount of discontinuity at a keyframe. In GESTURE, these three parameters are not exploited to their full extent. Experimenting with values for the parameters could produce varying styles of executing movement. For example, introducing a discontinuity could produce a sudden change of direction in the actor's movement. An animator may desire this effect to show a change of mind in the actor's objectives. This area could be explored further by finding ways to relate the control parameters to adjectives an animator may use to qualify movement.

The overall control structure of GESTURE can now be explained using the concepts presented thus far in this section. Consider the pseudo-code for the control structure shown in figure 4-4.

GESTURE first initializes the channel tables. This essentially empties them of any keyframes in preparation for processing a new movement script. A keyframe is then placed in each channel at frame 0 with the body assuming a resting stance and an initial stage position (line 2). The global clock is reset to 0 in preparation for advancing forward by event. The processing of each command line in the movement script initiates a new event. The clock is pushed forward to the time of this new event (line 6) and then keyframes are generated for the active action movements up to the new current clock time (line 7). Control then proceeds to the step where frames for the movements are generated (line 8). This is done by activating the gesture specification function corresponding to the movement. After the channel table contains keyframes for all movements in the script, the interpolation is applied to each channel, producing joint angles and stage position for all frames. These values are then stored in an animation script which can be used to play back the animation.

The high-level control for GESTURE is general and quite straightforward. It is an event-driven

```
1    initialize channel tables
2    set first frame to rest position
3    clock_time ← 0
4    while (there is still another line in the movement language script)
5    {
6        clock_time ← time specified by new movement command
7        produce frames for active action movements to current clock time
8        activate gesture specification function corresponding to gesture
             in movement command
9    }
10   interpolate channels in channel table
11   place interpolated values for stage position and all joints in animation script
```

Figure 4-4: Control Structure for GESTURE

system, and requires no knowledge about how particular movements are executed. The remainder of this chapter will discuss what happens at line 8 of the algorithm: how realistic movement is achieved from a descriptive movement command line.

4.3. A Graphical Representation for Movement Specification

In the previous chapter, the notion of a representation for movement accessed by the gesture specification functions was introduced. It was suggested that key body positions could be recorded using nodes in a graph, and movement could be produced by traversing the graph. This section will discuss the problems that arise when determining how to generate movement, and will present the graph as a solution.

When a gesture specification function is activated, it has to know the current position of the body before it can produce keyframes for the movement. For example, to scratch one's head the arm must follow a completely different path from a stretching position, with both arms high above one's head, than if both arms were hanging by one's side. The problem in producing keyframes for

a movement is to ensure that a sensible (i.e. a physically possible, and most likely) path is chosen, and that the keyframes are placed at times that will produce the desired velocity. Therefore to determine how and when the keyframes are to be placed, the current body position has to be assessed for position and orientation.

An analysis of the body position can take on many different forms. At one end of the spectrum, one could do a thorough mathematical analysis of all the joint angles, determine where all the joints and end-effectors are in space, and compute a path. This method could produce very realistic results, but would be very expensive computationally. The algorithm would also be dependent on the limb that is being moved, and the movement that it is executing. This approach is very low-level, and would be inappropriate for gesture specification functions which access information at a higher level. At the other end of the spectrum, a set of states could be maintained which would hold true for the body position at given times. This set would be updated as new movements are executed. Then instead of examining 10 or 15 joint angles to determine where the right arm is, we could retrieve information on the state of the right arm such as "right-arm-at-rest" or "right-arm-stretching". Then it would be relatively straightforward to determine which set of keyframes to produce in order to execute the next movement. The disadvantage of this method is that it is coarse-grained. While a movement to a new state is in progress, there are no states corresponding to the body positions during the transition period. This means that there is no information about how to begin a new movement during this transition. Therefore complete movements will have to be generated.

Secondary movement is a reflection on a person's character and emotions. As a person reacts to his or her environment, this motion constantly changes. To make the motion realistic, it is important to be able to switch quickly and smoothly between movements at any time. This involves being able to evaluate very quickly what position different body parts are in, and determining what path to follow to continue the smooth execution of another movement. We would like to choose a representation of the body posture that allows for quick evaluation of the pose the body is assuming and that easily conveys how to move from that pose to another one. The suggested representation is a graph in which the nodes represent body positions and the arcs are labelled with names of movements. If the body is in a position represented by a node in the graph,

then to execute a movement the graph is traversed along the arcs labelled with that movement. Since each node represents a key position for joints, the nodes in a path through the graph are equivalent to the keyframes in a channel; the keyframes can be interpolated to produce a smooth animation of a set of movements. Let us examine this graph representation in more detail.

The nodes in the graph represent *states* of the body, or, body positions. States can either be *named states* which correspond to actual poses, for example, "hands-on-waist", or *intermediary states* between the poses. Intermediary states are not necessary if there is a direct path between two named states. However intermediary states will have to be introduced to make the movement more realistic. For example, if no intermediary state existed between the position of the arms behind the back and the position of the arms in front of the body, the movement would be executed by pushing the hands through the body until they were in front of the body. If an intermediary state is introduced between these two states where the hand is slightly distanced from the body, the movement of the arms from behind the back to in front of the body will appear more realistic.

Two types of states have been identified. The nodes can be named similarly to reflect the states they represent: a named node represents the final pose of a movement, and an intermediary node represents a transition state along a path between two (or more) named nodes. Figure 4-5 shows a graph containing three nodes: two are named nodes, and the third node is an intermediary node between two body positions.

The labels on the arcs in the graph are the names of gestures. A gesture in this context is a movement that will achieve a certain body pose. Gestures, and thus arc labels, use the same names as named states. This means that to attain the body position "on-waist" from any node in the graph, the arcs with the gesture label "on-waist" should be traversed. Thus every node should have an arc for all gestures that do not have the same name as the name of the node. For example, the node named "rest" does not need an arc labelled "rest" since the body has already attained that position.

The other information that an arc must carry is some sort of distance measure between nodes. Since nodes represent arbitrarily defined body positions, there is no reason that the time to travel between any two pairs of nodes should be the same. When the final interpolation is applied to the

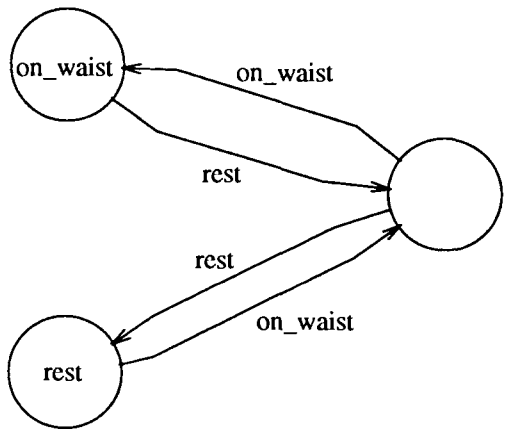
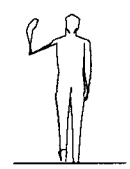


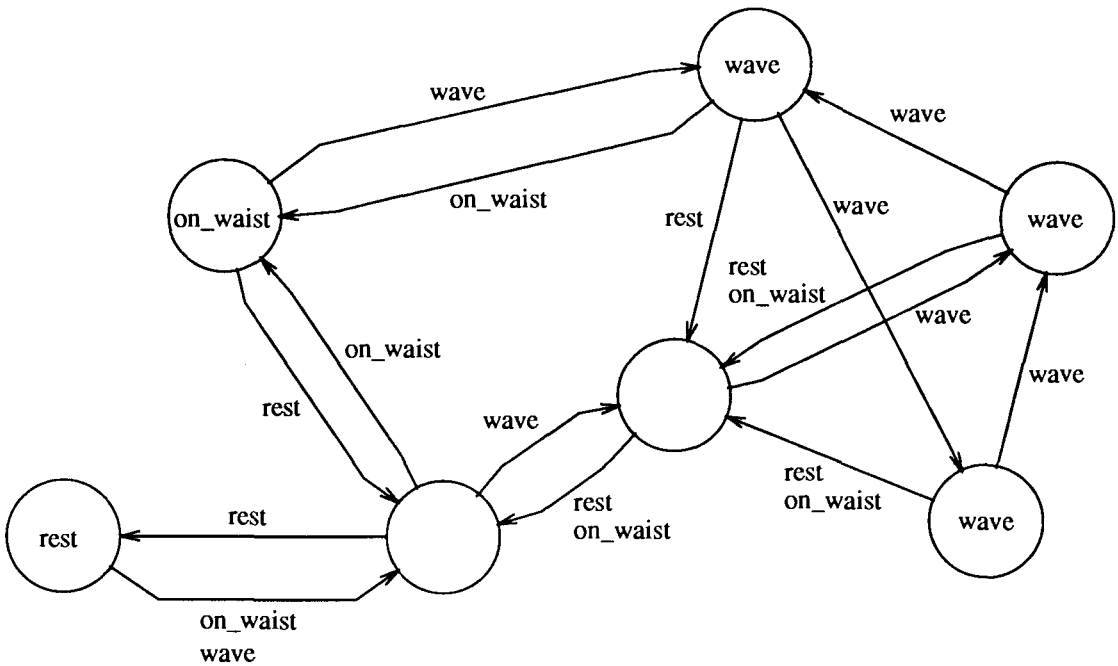
Figure 4-5: Graph with two named states and one intermediary state

ordered set of nodes, representing the keyframes, there must be information about how many frames to place between each of the keyframes. Thus the arcs are labelled with the number of inbetween frames that should be placed between adjacent nodes in a path.



The idea of singling out some nodes as named states corresponding to the final body positions of gestures works well for destination movements. But recall that there are also action movements which do not reach one body position, but rather, cycle through several body positions; for example, a head scratch or a wave. In the case of an action movement, we will assign the same name to several nodes, and define an ordering of these nodes through which to cycle. An expanded version of the above figure incorporating the wave gesture is shown in figure 4-6.

The nodes in the graph correspond to body positions. If we were to consider all the body positions that could be assumed, the graph would certainly be very large. One way in which the number of nodes can be reduced is to take into consideration the nature of a gesture. A gesture does not normally involve the whole body. Usually a gesture is performed by the arm, the head or the torso: in each case, some specific limb or part of the body is involved. For example, a wave is done by an arm, a nod by the head, or a slouch by the torso. Two gestures that are performed by different parts of the body are often independent of one another, and there is no reason they could not be done in conjunction with each other, such as nodding while waving. If the nodes were to



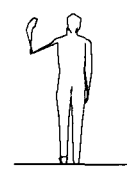


Figure 4-6: Graph with the action movement: "wave"

define complete body positions, then for every movement of one body part there will have to be a set of nodes with all possible movements of other body parts. This approach would yield a massive graph containing much duplicated information. To reduce the number of nodes and avoid repeating information, nodes are grouped into separate graphs, each graph pertaining to a unique set of joints which form one body part. For example, the three arm gestures presented in figures 4-5 and 4-6 - rest, on-waist, wave - are in a graph that records only joint angles for the arm. Execution of body movements can proceed by traversing several graphs concurrently, with each of the graphs controlling the movement of one set of joints.

This method of localizing the joints involved in body positions for a graph works well because gestures tend to correspond to local body parts. In fact, gestures are most naturally specified in terms of the movements of end-effectors, for example "place hand on waist" as oppose to "rotate shoulder and elbow joints such that ...". Graphs can naturally divide the body such that the joints from the centre of the body to an end-effector form a graph. In GESTURE there are seven graphs: two arms, two hands, head, torso and legs. The arm and hand have been separated for finer control

over movements of these limbs. Both legs have been grouped into one graph because of the nature of the algorithm used for the only movement executable by the legs: walking. This will be discussed in the next section.

Occasionally movements of different body parts will have to be co-ordinated. A head scratch, for example, not only involves traversing the arm and hand graphs to position the hand against the head, but also requires traversing the head graph to ensure that the head is positioned properly to receive the scratch. It would not do if the hand was scratching in mid-air because the head had turned away to look at something. Thus, some gestures may be common to two graphs, and the graphs must be traversed concurrently along arcs with the same gesture name.

A node in the graph represents a body position in space assumed by a set of joints. Continuous movement is achieved by smooth interpolation through a defined ordering of these nodes. So far in the discussion, if one movement interrupts another, we have assumed that we are "at" a node (i.e. the body is assuming a position defined by one of the nodes), and to continue with the next movement, it suffices to follow the arc emanating from the current node labelled by this new movement. However, recall that there is some distance between nodes, and therefore some time is required to travel between nodes. Thus, it is possible that a movement will be interrupted while the arc is being traversed, rather than when the body part is assuming a position at a node. In this case we would not be able to follow the arc for the new gesture until we had reached the next node. If this were the solution to interrupting gestures, it is likely that very few interruptions would be observed in the movement, especially if there were very few intermediary nodes for a movement. Thus the animation would result in a series of uninterrupted movements. This solution might be acceptable for primary movement, where actors have clearly defined objectives. However secondary movement is governed by sub-conscious thoughts, environmental changes or body discomfort. So, for body language to convey anything, secondary movement should be able to switch easily from one movement to another at any time.

In GESTURE, we would like the flexibility that would allow for movements to be interrupted and continued with other movements, and perhaps for those to be interrupted in turn. This means that when a new movement takes over after interrupting the body in the midst of executing a previous

movement, the body will not necessarily be in a predetermined state, but could be somewhere between two states. A procedure is needed for going from this inbetween body position to a known state, without having to digress to numerically analyzing this body position.

In order to implement this instantaneous fluctuation between gestures in the graph, it should be possible to create new positions at any time. Thus if a movement is in the process of traversing an arc when it is interrupted, a temporary node is created. This node is a special node which can act as a keyframe for the final interpolation, but which will not be accessible in the graph as soon as it is no longer the current node. The reason for this is that although the joint angles for that node can be computed easily (by interpolating between the joint values of the two nodes connected by the arc being traversed when the interruption occurred), it would be a non-trivial problem to determine where all the arcs emanating from this new node should lead, and furthermore, what distance measure should be associated with each of these new arcs. However this temporary node must at least have an arc to the most appropriate node in the graph for beginning the execution of the new movement. Therefore, the temporary node adopts the arcs and arc lengths of either the most recent node or the destination node for the original gesture before the interruption, depending on which node is closer (by the distance measure). Figure 4-7 shows an example of a movement of the arm from rest position to putting the hand on the waist. (The intermediary nodes have been labelled A and B in this figure to distinguish them.) After 3 time units, this movement is interrupted to put the arm behind the back. A temporary node (labelled T in the figure) is created to represent the body position at the time of the interruption. Since the body position was closer to the intermediary node A than to the "rest" node when the interruption occurred, the temporary node adopts the arcs of the intermediary node. This means that the temporary node will adopt an arc with the label "back" and distance measure 2. Once this arc has been traversed, the intermediary node B has its own arcs for all the gestures, and can continue the movement. From this point, the temporary node is inaccessible until it is required in the final interpolation. The thicker solid lines in the figure indicate the final path traversal and the node labelled T indicates the temporary node created when the interruption occurred.

A method has been presented for representing movement. The graphical representation allows for the continuation of any movement with any other movement by defining paths for the gestures

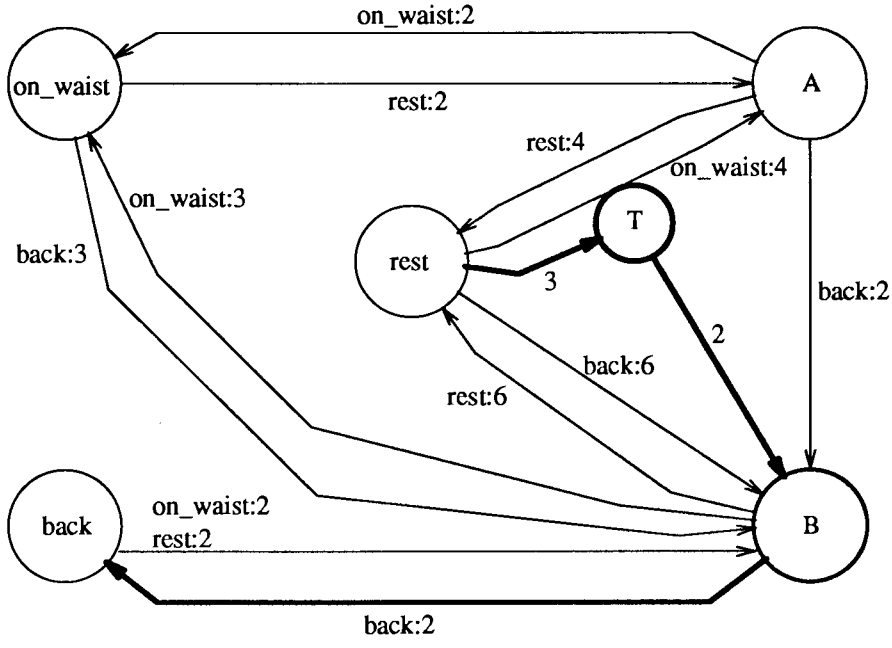
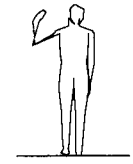


Figure 4-7: Graph with a temporary node caused by an interrupting gesture



from every node in the graph. This solution presents a way of interrupting movements at any time and recovering easily to continue with another movement. The flexibility of this representation is well-suited for animating secondary movement.

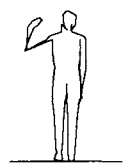
4.4. The Gesture Specification Functions

One difficulty in devising algorithms for human animation is that the human body is not a very general structure. An algorithm that generates movement for one arm could be used to control the motion of the other arm by applying a reflection operation. However the same algorithm would not be suitable for controlling the motion of the legs, and even less appropriate for the head. Although all these limbs are links connected at joints, the links are different sizes, there are different numbers of joints in each limb, and constraints on the humanly possible rotations of the joints vary between the joints. Human animation must therefore appeal to specialized algorithms with data about the composition of the body.

Animation systems which incorporate knowledge focusing on particular types of human movement can produce realistic motion, because different knowledge is required to animate

different parts of the body. GESTURE encodes knowledge about movements of different body parts into gesture specification functions. In particular, each function generates keyframes to perform a particular movement using the graph and two special-purpose tools, *anchor* and *traverse_graph.*

Recall that a request for the execution of a movement initiates a new event at a certain clock time. The gesture specification functions can alter the channel table from the current clock time onwards to interrupt previous movements and generate keyframes for the new movement. Anchoring is a mechanism for placing an appropriate keyframe at the current clock time, and then removing all keyframes in the future of that time. The new keyframe that is created corresponds to a temporary node in the graph. The joint values at this time are computed by applying an interpolation to the segment defined by the keyframes surrounding the current clock time. The effect of anchoring a channel at a particular time is to ensure that movements that were executing up to that time continue to do so in the same way, and that the new (interrupting) movement can begin immediately at the current clock time.

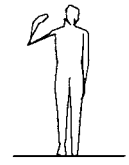


The other tool available to the gesture specification functions, *traverse_graph*, generates the keyframes for the specified movement. The type of gesture desired is specified, along with a qualitative description of how fast the movement should be executed (this will affect the inbetween values on the arcs), and one keyframe is placed in the channel table for every node encountered in the graph traversal. The *traverse_graph* tool also handles the creation and deletion of action movement objects. Before the graph traversal begins, all active action objects are examined to see if the joints used in executing their movement coincide with the current movement being processed. If so, the action movement will no longer be able to continue its cycle and so the object is removed. At the end of a graph traversal, the last node reached is examined to see if it represents the last posture in a destination movement, or if it is one node in a cycle for an action movement. If the latter is true, an action object is created for this new movement.

Most movements can be generated by the gesture specification functions using just the two tools *anchor* and *traverse_graph.* Movements that are generated using predefined keyframes mostly fall into this category, since the graph is a natural representation for keyframing. However,

occasionally a gesture specification function may generate movement by applying an algorithm such as a dynamic or a kinematic simulation. Using the results of such an algorithm could involve altering body positions for nodes in the graph, or adding or removing nodes from the graph. In this case, the gesture specification functions will require additional control besides simply using the two tools. This situation will be discussed further in the latter part of this section.

Each of the gesture specification functions is invoked with four parameters. The *clock_time* is the time at which the movement should begin execution. The other three parameters are adjectives which describe specific ways the movements can be executed. The *speed_factor* describes how quickly the movement should be executed. The *hemisphere* indicates if the left, right or both sides of the body are involved in the movement, or in the case of the head, if the movement should be done looking left, right or straight ahead. The *height* is used by head movements to signify if the head is looking up or down, or in the case of a wave, if the gesture is large or small. If the movement script does not qualify the movements, default values are used. The speed of the gesture always defaults to an average speed, however the defaults of the other qualifiers depend on the movement.



The *speed_factor* for a gesture requires further explanation. The sole primary movement in the GESTURE implementation is *walk*. The velocity of a walk can be measured quite accurately, and a slow, average and fast walk could be defined with respect to walking velocities of people. However measuring the velocity of a head scratch, or of turning the head to look in another direction is not so easily done. The secondary movement in GESTURE is generated from keyframe data. This data was obtained by using a keyframing system COMPOSE [Calvert 89]. The joint angles for each of the positions created in this system were recorded in the graph nodes. However a decision had to be made as to the measure attached to each of the arcs. Recall that this measure represents the number of "inbetween" frames between key positions and that this value will control the speed of the movement. The choice of measures on the arc was based on applying the same kind of intuition an animator uses in a keyframing system. For each movement, a value was selected that would produce a natural, average-looking velocity of that movement. The qualifiers "slow" and "fast" simply multiply or divide these values by 2, to decrease or increase the speed of the movements respectively.

```
/*
*************************************************************************
*
*   Gesture Specification Function:     gs_on_waist
*
*   Gesture Description:
*       Put one or both hands on waist.
*
*   Qualities Used (defaults in < > brackets):
*       speed_factor    : SLOW_SPEED, <AVERAGE_SPEED>, FAST_SPEED
*       hemisphere      : LEFT_SIDE, RIGHT_SIDE, <BOTH_SIDES>
*
*************************************************************************
*/
void gs_on_waist(clock_time, speed_factor, hemisphere, height)
int clock_time;
int speed_factor;
int hemisphere;
int height;
{
#ifdef lint
    height = height;
#endif

    /*
     * set defaults
     */
    if (speed_factor == NO_QUALITY)
        speed_factor = AVERAGE_SPEED;
    if (hemisphere == NO_QUALITY)
        hemisphere = BOTH_SIDES;

    /*
     * produce keyframes
     */
    if (hemisphere == LEFT_SIDE ||
        hemisphere == BOTH_SIDES)
    {
        anchor(GRAPH_LARM, clock_time);
        traverse_graph(GRAPH_LARM, LARM_ON_WAIST, speed_factor);
        anchor(GRAPH_LHAND, clock_time);
        traverse_graph(GRAPH_LHAND, LHAND_WAIST, speed_factor);
    }

    if (hemisphere == RIGHT_SIDE ||
        hemisphere == BOTH_SIDES)
    {
        anchor(GRAPH_RARM, clock_time);
        traverse_graph(GRAPH_RARM, RARM_ON_WAIST, speed_factor);
        anchor(GRAPH_RHAND, clock_time);
        traverse_graph(GRAPH_RHAND, RHAND_WAIST, speed_factor);
    }
}
```

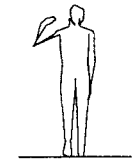


Figure 4-8: Gesture Specification Function for gesture: on_waist

The gesture specification function which controls the movement of placing one or both hands on the waist is shown in figure 4-8. This function demonstrates the most basic use of the *anchor* and *traverse_graph* tools to produce movement from the keyframing data in the graph. First the

```
/*
*************************************************************************
*
*   Gesture Specification Function:     gs_scratch
*
*   Gesture Description:
*       Scratch back of head with left or right hand
*
*   Qualities Used (defaults in < > brackets):
*       speed_factor    : SLOW_SPEED, <AVERAGE_SPEED>, FAST_SPEED
*       hemisphere      : LEFT_SIDE, <RIGHT_SIDE>
*
*************************************************************************
*/
void gs_scratch(clock_time, speed_factor, hemisphere, height)
int clock_time;
int speed_factor;
int hemisphere;
int height;
{
    ACTION *action_movt;
    CHAN_ENTRY *last_head, *last_arm;

#ifdef lint
    height = height;
#endif

    /*
     * set defaults
     */
    if (speed_factor == NO_QUALITY)
        speed_factor = AVERAGE_SPEED;
    if (hemisphere == NO_QUALITY)
        hemisphere = RIGHT_SIDE;

    /*
     * produce keyframes
     */
    if (hemisphere == LEFT_SIDE)
    {
        anchor(GRAPH_HEAD, clock_time);
        traverse_graph(GRAPH_HEAD, HEAD_L_SCRATCH, speed_factor);

        anchor(GRAPH_LHAND, clock_time);
        traverse_graph(GRAPH_LHAND, LHAND_SCRATCH, speed_factor);

        anchor(GRAPH_LARM, clock_time);
        action_movt =
            traverse_graph(GRAPH_LARM, LARM_SCRATCH, speed_factor);
        action_movt->graphs_involved[GRAPH_HEAD] = TRUE;
        action_movt->graphs_involved[GRAPH_LHAND] = TRUE;
    }
    else
    {
        anchor(GRAPH_HEAD, clock_time);
        traverse_graph(GRAPH_HEAD, HEAD_R_SCRATCH, speed_factor);

        anchor(GRAPH_RHAND, clock_time);
        traverse_graph(GRAPH_RHAND, RHAND_SCRATCH, speed_factor);

        anchor(GRAPH_RARM, clock_time);
        action_movt =
            traverse_graph(GRAPH_RARM, RARM_SCRATCH, speed_factor);
        action_movt->graphs_involved[GRAPH_HEAD] = TRUE;
```

```
        action_movt->graphs_involved[GRAPH_RHAND] = TRUE;
    }
    /*
     * if the arm gets in position before the head, we don't want
     * to be scratching in mid-air, so anchor the arm to the time
     * when the head is in position, so that it won't begin its
     * cycle 'till after that time
     */
    last_head = chan_head;
    while(last_head->next != NULL)
        last_head = last_head->next;
    last_arm = (hemisphere == LEFT_SIDE ? chan_larm : chan_rarm);
    while(last_arm->next != NULL)
        last_arm = last_arm->next;

    if (last_arm->time < last_head->time)
    {
        if (hemisphere == LEFT_SIDE)
            anchor(GRAPH_LARM, last_head->time);
        else
            anchor(GRAPH_RARM, last_head->time);
    }

}
```

Figure 4-9: Gesture Specification Function for gesture: scratch

defaults are set if the movement qualities were not specified in the script. This movement can only be qualified by the speed and hemisphere, and so the height qualifier is ignored. By default both hands are placed on the waist. Then, the channel table is anchored at the current clock time, and the graphs are traversed; this places the keyframes required to put the hands on the waist in the channel table.

The discussion of the graph in the last section raised the possibility of having to traverse two graphs concurrently in order to execute some movements. One example is the head scratch which must co-ordinate the movements of the arm and hand on one side of the body with the head. The solution in GESTURE is shown in figure 4-9. This gesture has the added complication of being an action movement, which means that an action movement object is created in the graph traversal for the arm. Since a head scratch involves the joints controlled by three graphs, the action object created stores this information so that if a movement involving any of these joints occurs at a later time, this object will be removed, and the head scratching will cease. The gesture specification function for head scratch must deal with one other point. When the head and arm graphs are traversed, depending on where these limbs were positioned before the interruption occurred, one limb might arrive at its destination before the other. If the arm is in position to scratch before the head, it is important that the scratching cycle does not begin until the head has been properly

```
/*
*************************************************************************
*
*   Gesture Specification Function:     gs_walk
*
*   Gesture Description:
*       Start up a walk.
*
*   Qualities Used (defaults in < > brackets):
*       speed_factor    : SLOW_SPEED, <AVERAGE_SPEED>, FAST_SPEED
*
*************************************************************************
*/
void gs_walk(clock_time, speed_factor, hemisphere, height)
int clock_time;
int speed_factor;
int hemisphere;
int height;
{
    static void create_legs_graph();
    static void set_swing();

    CHAN_ENTRY *last_arm;

#ifdef lint
    hemisphere = hemisphere;
    height = height;
#endif

    /*
     * set defaults
     */
    if (speed_factor == NO_QUALITY)
        speed_factor = AVERAGE_SPEED;

    /*
     * If this is the first time we have been asked to walk, we must
     * create the legs graph and the location graph.
     * We can tell if they haven't been created yet, because the
     * arc labelled LEGS_WALK from the first node, will just point
     * to itself (the first node), and not another node which we
     * would traverse to get to the walk cycle nodes
     */
    if (legs_states[0].next_state[LEGS_WALK] == 0)
        create_legs_graph(speed_factor, hemisphere, height);

    /*
     * produce keyframes
     */
    anchor(GRAPH_LEGS, clock_time);
    traverse_graph(GRAPH_LEGS, LEGS_WALK, speed_factor);

    /*
     * if the arm is *not* doing something else right now (ie. is at rest),
     * start the arm swinging with the legs
     */
    last_arm = chan_larm;
    while(last_arm->next != NULL)
        last_arm = last_arm->next;
    if (last_arm->state == 0)
    {
        set_swing(GRAPH_LARM);
        traverse_graph(GRAPH_LARM, LARM_SWING, AVERAGE_SPEED);
    }
```

```
        last_arm = chan_rarm;
        while(last_arm->next != NULL)
            last_arm = last_arm->next;
        if (last_arm->state == 0)
        {
            set_swing(GRAPH_RARM);
            traverse_graph(GRAPH_RARM, RARM_SWING, AVERAGE_SPEED);
        }
    }
```

Figure 4-10: Gesture Specification Function for gesture: walk

positioned. Therefore the channel table for the arm is anchored in the destination position for scratching at the time when the head has reached its destination position.

The flexibility of the graph is demonstrated by the fact that it can not only represent movement that has been specified by keyframing, but also by any other form of movement generation. The movement *walk* serves not only as an example of primary movement in GESTURE, but also as an example of movement generated using non-keyframing techniques. The gesture specification function that generates the walking movement uses the graph to incorporate data produced from a dynamic simulation of a walk [Bruderlin 88]. Human walking also involves swinging each arm in synchrony with the opposite leg. An arm swing is initiated by this same gesture specification function using a kinematic description of the movement of the arm as a function of the movement of the leg.

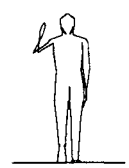
Figure 4-10 shows how the *walk* gesture specification function uses the graph to incorporate dynamically and kinematically simulated movement. The subsidiary functions *create_legs_graph* and *set_swing* are invoked which alter the graph by changing joint values at the nodes, and modifying arcs and arc lengths. When the graph is traversed after this, animation frames will be generated that control movement according to each of these simulations. The method for incorporating each of these dynamic and kinematic algorithms into the graph will now be examined.

Animating a human walk is a particularly complicated type of control over articulated bodies, as it involves addressing such issues as balance and co-ordination as well as impact with the ground, an external object to the body model. One way of producing very realistic motion for complex human movement is to set up a dynamic simulation of the movement, which considers the forces and torques applied to the body while executing this movement. To avoid the user having to

supply these forces and torques to produce the desired movement, a high-level interface is appropriate. Bruderlin has implemented a goal-directed system where the movement specification for the walk includes supplying parameters such as the velocity, step length or step frequency, pelvic list and lateral displacement [Bruderlin 88]. The result of his simulation is a frame by frame specification of joint angles in the legs for the duration of the walk. These results are used to control the walk in GESTURE.

The style of an actor's walk depends on the personality and moods defined for the actor. A gesture specification function can use the adjectives that qualify the walk in the movement script to supply appropriate parameters to the walking algorithm. Once a walk is generated, the joint values over time must be incorporated into the graph.

The function *create_legs_graph* assigns joint angles produced from the simulation to one node in the legs graph for each frame of the walking sequence. Since the simulation produces every frame in the walk, an interpolation is not required. To enforce this, the arcs that are created between the nodes all have arc length 0. This is a general way to incorporate the data from any movement generation algorithm that defines values for every frame. Thus the graph could also represent movement from film data or rotoscoping, for example.



A walking sequence generated by this algorithm consists of three cycles: the develop, the rhythmic and the decay cycles. The body accelerates to the specified velocity during the develop cycle, walks at that velocity during the rhythmic cycle and then decelerates to a stop during the decay cycle. The rhythmic cycle is represented by nodes in the graph which generate the movement of two steps, one on the left foot and one on the right. The walk is therefore an action movement, and the cycle through the nodes in the rhythmic cycle continues until the movement script specifies a *halt* command.

An example of the nodes and arcs in the legs graph after the *create_legs_graph* function has been executed is shown in figure 4-11. In general there will be of the order of 50 nodes in each of the three walking cycles.

When people walk, they usually swing their arms. When a walk command is specified in the

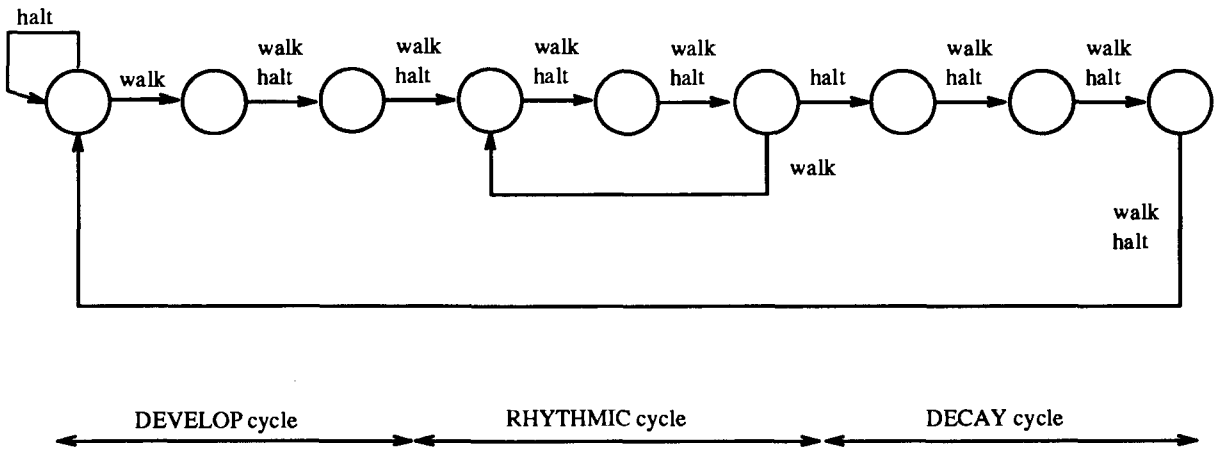
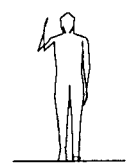


Figure 4-11: The legs graph representing the *walk* and *halt* movements

movement script, the gesture specification function that knows about walking also generates an arm swing, if appropriate. There are a few issues to be addressed with respect to this movement. First of all, the style of arm swing is affected by the walk: a bouncy, light-hearted walk may cause a more vigorous arm swing. Therefore the nodes in the arm graph that control the arm swing will be modified accordingly. Secondly, frames for controlling the arm swing cannot be generated independently of those controlling the walk. The gesture specification function must ensure that the frame generating the left heel strike occurs at exactly the same time as when the right arm is in its furthest forward position. Lastly, since arms can also be engaged in secondary movement, the arm swing should not occur unless the arm was at rest next to the body. Furthermore, if an arm was executing some secondary movement during the walk, and then is dropped to the side, the gesture specification function that deals with placing the arm at rest must know to begin an arm swing if the actor is walking. This involves creating arcs to the arm swing nodes from nodes representing any other arm position, and determining the arc lengths that will put the arm swing in synchrony with the walking cycle.

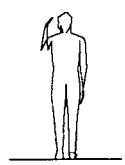


The function *set_swing* modifies the arm graphs so that traversing those graphs will produce an arm swing meeting the three requirements stated above. The arm swing is generated kinematically with the use of two keyframes representing the forward and back positions of the arm in the swing. The joint angles for the arms at these keyframes are determined as a percentage of the joint angles

in the legs at heel strike and toe off. If the arm is to travel from an arbitrary node to one of the arm swing nodes, the gesture specification function determines if it is more appropriate to go to the forward or back position of the arm swing, and then calculates the arc length based on the number of frames in the walk cycle that would synchronize the arm with the legs. The arm swing is then represented by an action object which is removed only when the walk comes to a halt.

Assigning individual functions to be responsible for the execution of a single gesture has proved to be an appropriate way to oversee the control of secondary movement. Since each gesture requires special knowledge about how it is to be executed, gesture specification functions are a good way to give individual attention to each of these movements. Furthermore, with the use of the graph, the gesture specification functions can apply the most suitable movement generating algorithm for each gesture. The approach to human animation presented here provides a flexible and general way to produce and co-ordinate different types of movements.

4.5. Presentation



GESTURE is an interactive graphics program in which the user communicates through the mouse and keyboard and the output is displayed on a colour monitor. The system is written in 'C' and runs on an IRIS-2400. The IRIS geometry engines, which perform matrix operations in hardware, are used to increase the speed of matrix multiplications that are required to determine the rotation of each joint in the model of the body.

The purpose of GESTURE is to demonstrate that secondary movement can be produced from a description in a movement script. In implementing GESTURE, it was considered beneficial if the user-interface simulated the environment of the ideal framework as presented in chapter 3. In this ideal situation, the user selects the number of actors for the animation, assigns them all primary goals, and defines the personality and mood of each actor which then determines their secondary movement. The ideal framework has been restricted in the following way. The animator can work with exactly two actors who are introduced as Simon and Sally. These actors have their own fixed starting position, facing each other at opposite ends of the stage. Since GESTURE is concerned only with secondary movement, the sole primary goal assumed for Simon and Sally is to walk to the other end of the stage. This initial set-up is pictured in figure 4-12.

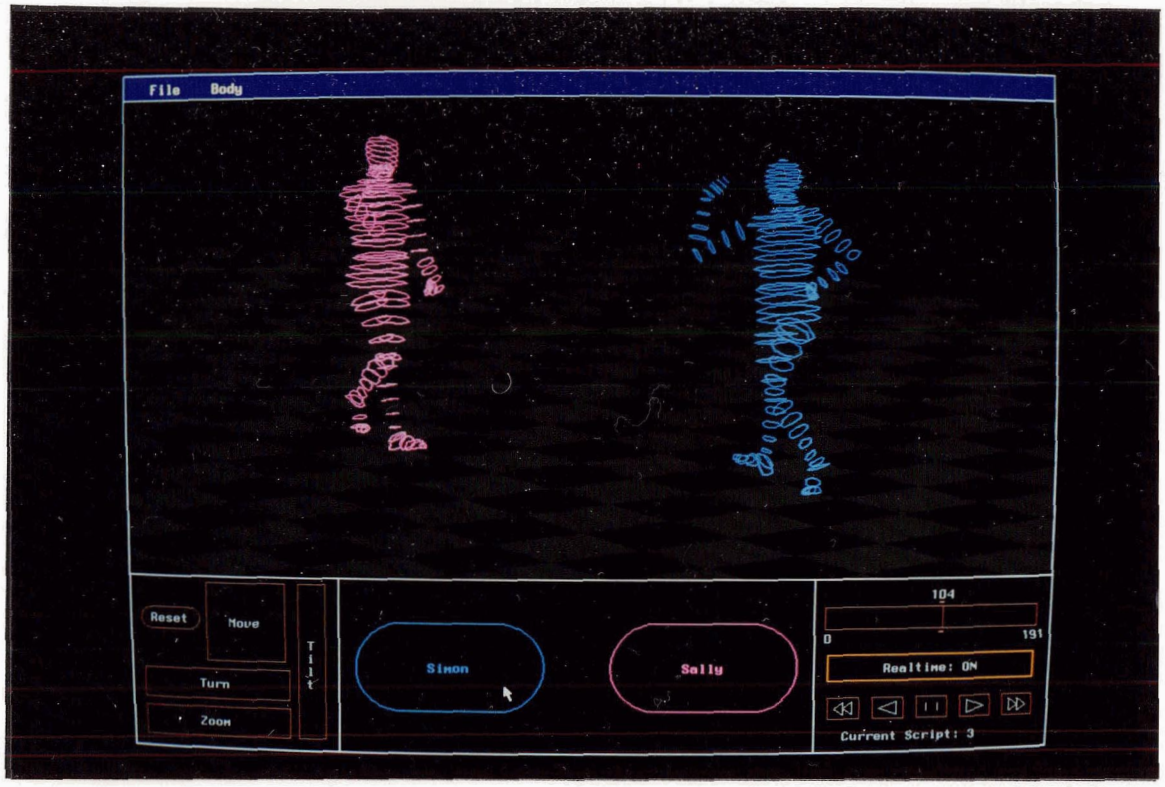


Figure 4-12: Main screen of GESTURE

In GESTURE, the animator has high-level control over the actors' secondary movement by defining each of the actors' personality and moods. Figure 4-12 shows two buttons on the screen labelled "Simon" and "Sally". If one of these buttons is selected with the mouse, the character definition screen for the selected actor is displayed, and the animator can give this actor a character (figure 4-13). In the upper half of the screen, the actor's personality is defined by adjusting a set of two-ended sliders. For example, an actor can be either extroverted or introverted on a scale of 0 to 10. The other possible opposing personality traits to select from are a cheerful or gloomy disposition, assertive or passive and domineering or submissive. The lower half of the screen determines the mood of the actor. An actor can exhibit degrees of boredom, nervousness, tiredness, impatience and fear on a scale of 0 to 10. Depending on the values selected on this screen, the actors will display different types of secondary movement as they walk across the stage. For example, if the value of "tired" is very high, the actor will rub his or her eye.

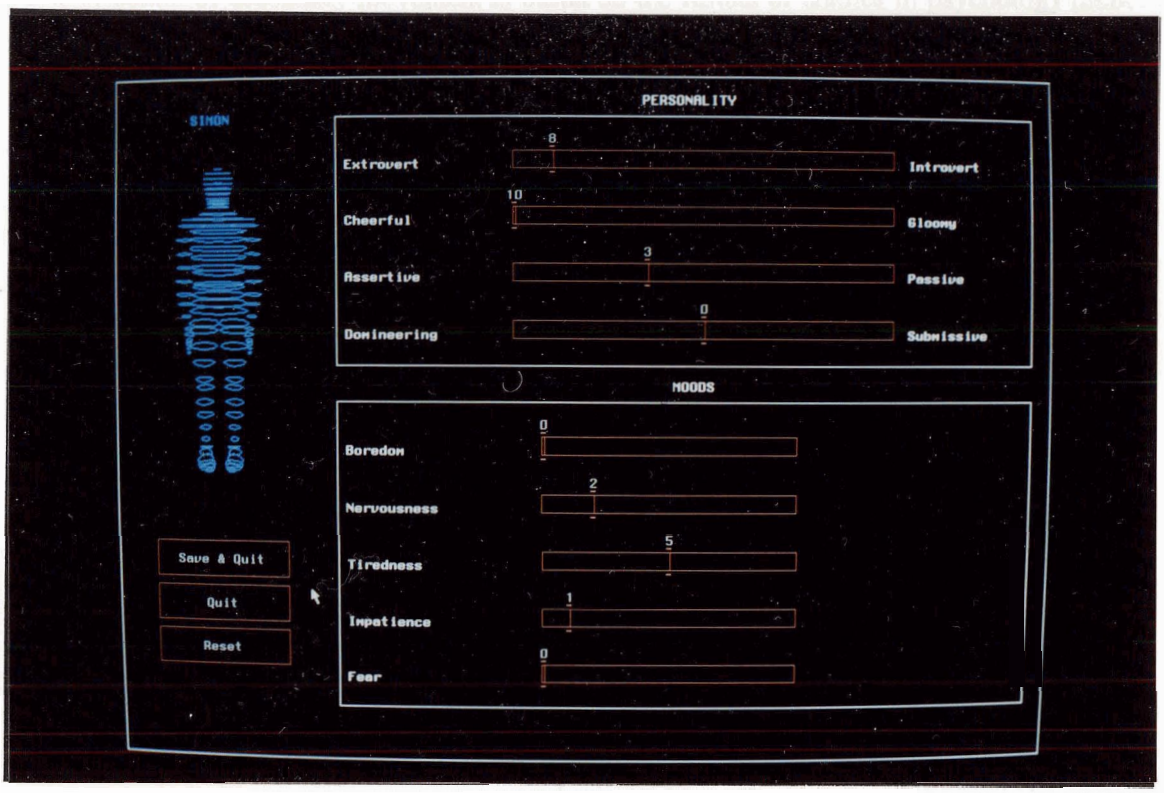
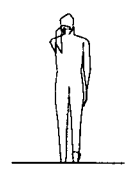


Figure 4-13: Character definition screen of GESTURE

The presentation of GESTURE demonstrates how an animator can interact with a goal-directed system and achieve realistic human movement. GESTURE does not include an expert system which would base its selection of secondary movement to display on the character descriptions. However, a function has been written to simulate the behaviour of the expert system. This *mock expert system* creates a movement script by selecting appropriate gestures at fixed intervals of time. Since the primary goal in GESTURE is to walk, then "walk" is the first command that is entered in the movement script. The speed of the walk is selected depending on how cheerful or gloomy the actor is, and on his or her degree of boredom and tiredness. Secondary movement is entered depending on characteristics related to that gesture. For example a scratch will be entered into the movement script more often if the actor is very nervous. The scratch will be qualified by "fast" if the actor is very impatient or fearful and "slow" if the actor is passive. The "wave" gesture will occur at most once as the actors are about to pass each other, only if the actor is not introverted. The movement script will terminate with a "halt" command. The rules used in the mock expert system can be found in appendix C.

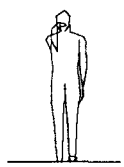
The selection of secondary movement is based on the results of studies in psychology [Scheflen 72]. In these studies the frequency and duration of movements were not specified. Frequency and duration of gestures were chosen to adequately demonstrate secondary movement in GESTURE. The results may not be psychologically sound, however they demonstrate the effectiveness in making the actors' motion believable. Movement scripts may also be specified manually and processed by GESTURE in the same way as the ones produced by the mock expert system described above. In this way, movement scripts can be specially constructed to contain realistic combinations of secondary movement.



Chapter 5
System Evaluation

This thesis has presented a general framework for a human animation system, with the focus on a high-level approach to incorporating secondary movement into actors' motion. GESTURE implements the part of the framework that produces the secondary movement specified in a movement script. In the first section, the success of the implementation of the graphs and gesture specification functions will be examined. In the following section, the system will be evaluated against various measures. The chapter will conclude by comparing the results of this system with other research that has been conducted in similar areas.

5.1. Analysis



In the last chapter, the use of gesture specification functions along with a graph was presented as a way to represent and generate movement, with the flexibility of interrupting movements at any time. An advantage of using these functions is that knowledge about movements is represented in an organized way, which is easy to understand and modify. Thus new gestures can be easily introduced into the system. Many types of gestures were incorporated into the system, and it was also demonstrated how movements simulated by various types of algorithms could be incorporated.

Adding a new gesture to the system requires the creation of a corresponding gesture specification function. If the gesture can be produced by using nodes currently in the graph, writing this function can be easily done using the two tools *anchor* and *traverse_graph* presented in section 4.4. If body positions not currently represented in the graph must be used, new nodes and arcs must be added to the graph in the following way. If the node is an intermediate node, then an arc must be created from that node for every gesture. This is done so that any movement can be continued from this intermediate node. If the node is a named node, as well as the arcs described above, an arc must be added to every other node in the graph labelled with the name of the new node. This is

done so that the gesture represented by this new node can be performed from any position represented by all other nodes in the graph. The last step is to label all the arcs with the distance measure.

Let us examine the complexity of representing the graph in terms of storage space. Assume the graph has n nodes and g gestures. In general g < n because there can also be intermediate nodes. Each named node must have an arc for every other gesture but the one labelling its node, thus g(g - 1) arcs. The intermediate nodes must have an arc for every gesture, thus (n - g)g arcs. A graph must therefore store O(ng) arcs. Adding a node to the graph requires adding up to g arcs to the new node, and possibly n arcs for the other nodes if the new node is named by a new gesture. So adding a new node can be done by adding O(n) arcs.

```
struct graph_head {
        QUATERNION joint_value[NUM_HEAD_JOINTS];
        short next_state[NUM_HEAD_GESTURES];
        short arc_length[NUM_HEAD_GESTURES];
};
```

Figure 5-1: Structure representing a node in the head graph

Consider for example the space used by the head graph in the GESTURE implementation. An arc can be represented by two integers, one representing the node it is pointing to and the other the distance measure. The storage for a node must also contain the quaternion values for each joint at that node. A quaternion is represented by four floating point numbers. Figure 5-1 shows the structure representing one node in the head graph. The number of head joints (NUM_HEAD_JOINTS) is 8, one for the ATLANTAL and 7 CERVICAL joints in the neck. The head graph has 13 (NUM_HEAD_GESTURES) gestures and there are 13 nodes in the graph. The total storage space used for the head graph is 2340 bytes. This is not an unreasonable size for storing a graph that represents the movements of one body part. The storage required for representing an animation script of 126 frames is 90937 bytes, and so in comparison, the storage for the graphs is not significant.

The number of arcs that must be added to the graph when introducing a new gesture must also be gauged with respect to the amount of work involved in deciding which nodes to connect, and what measure to assign to the arcs. Creating arcs can be a tedious process as a minimum of g (the

number of gestures) arcs must be added for each new node. However, once the arcs have been created, the graph provides a foundation for any combination of movements to be executed. The arc creating process could be made easier by the use of a graph generating program. This program would display nodes as body positions, and show the different arcs connecting them. The user could modify arcs and arc lengths, and then specify a path through the graph which would define a series of movements to be executed by a figure. This mini-script could be previewed, and then the graph modified to adjust body positions or timing between the positions. A graph generating program was not implemented since a keyframing tool was available which could be used to preview movements, and adjust timing which would alter arc lengths [Calvert 89].

One point should be mentioned about the distance measure on the arcs. In general it is safe to say that the further the physical distance between two body positions, the larger the distance measure on the arc between the two nodes representing the two positions. However the arcs length should not be viewed in this way. Instead, this measure, which is attached to an arc labelled by a gesture, should be an indication of how long it would be natural to take when moving between the two positions to perform the particular gesture. For example, consider the two body positions represented by the hand at the side of the body, and the hand next to the head, and arcs for rubbing an eye and saluting which connect the nodes representing these positions. The distance measure on the arc with the eye rubbing gesture will be larger than the measure on the arc with the salute gesture, since the former gesture will probably be executed in a slower, lazier manner than the latter. In fact, in GESTURE, the arc length of the eye rubbing arc is twice as large as the arc length of the salute arc.

From this analysis, one can see that a great deal of time and care may be taken in creating the graphs and producing gesture specification functions. However this time must only be invested once. The time involved in giving studious attention to the specification of the various gestures pays off in the endless number of possible scripts that can be created from the movements. This process is analogous to developing an expert system: a knowledge engineer can spend many months embodying an expert's knowledge into a system, but once it is complete, the expert system can be used by many people in varying situations.

5.2. General Evaluation

One of the objectives of this thesis is to show that incorporating secondary movement into a human animation system can transform a robot-like figure into a very life-like actor. The success of an animation is best judged by how believable the viewer considers the actors' movements to be. In order to view the resulting animation produced from a movement script, a sequence has been printed on the margin of the pages of this thesis. The animation can be viewed by flipping through the pages in reverse order. The movement script that generated this sequence is shown in figure 5-2. This script generated 126 frames, however only frames 26 through to 101 are printed.

```
0    walk
0    rub
50   wave friendly
50   look left
60   look right up
70   on_waist both slow
70   clench both
80   look left up
80   halt
100  look straight ahead
```

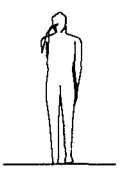
Figure 5-2: Movement script used to generate animation printed in margin

As well as the quality of the final animation, the system should also be assessed on its usability. GESTURE implements only the part of the framework that reads movement scripts and produces an animation script. However the whole framework setting has been simulated in order to provide a more user-friendly environment for the system. GESTURE also has a well-defined interface which could be used if an expert system were to be implemented and incorporated into the framework. The movement script uses a language which is powerful and flexible. Movements can be initiated at any clock time and without regard to the state of the body at that time. If a movement is not completely qualified in the script, default qualities will be assumed. The movement script completely defines the animation script that will be produced by GESTURE, and is an elegant interface for other modules to specify the movements for GESTURE.

The gestures that have been implemented were chosen to cover a variety of different types of movements and to demonstrate different techniques for generating movements. Movements such

as "on_waist" or "in_front" simply use named nodes to keyframe their positions. An intermediary node was introduced when the "in_back" gesture was added, to ensure that movements would not force the arm through the body on the way to the back. The "scratch" gesture demonstrates how movements in two graphs (the head and the arm) must be co-ordinated. Non-keyframed movements can be generated by applying algorithms in the gesture specification functions and modifying the graph to incorporate the results. The "walk" movement is an example of this type. A complete list of the gestures implemented is found in appendix B. The number of gestures and the variety of methods used to implement them supports the ease with which other secondary movement could be incorporated.

Graphics programs are inherently time-consuming and resource-intensive because of the amount of information that must be stored and displayed. For each frame in the animation, a hierarchy of matrix transformations must be stored which corresponds to the joint rotations applied in the hierarchical representation of the body for that frame. This is not only a storage problem, but will also involve many matrix multiplications to display each frame. One of the options in GESTURE is to select playback speed as real-time. This feature is made possible by using two techniques. First of all, the animation script is preprocessed and all matrix multiplications are performed for every frame. The script is then represented as a series of 3-D polygons which when drawn will display the figure in the body position represented by the hierarchy of joint transformation matrices. This optimization process increases the display speed significantly, however it is possible that real-time will still not been attained. In these cases, real-time display is achieved by skipping frames in the script so that only some of the frames are displayed. With this capability to display animations in real-time, it is possible for animators to get a sense of the overall movement. Providing this option in GESTURE gives immediate feedback on the resulting movement.



The actual process of creating an animation script from the movement script also requires some time. The movement script shown in figure 5-2 required 27 seconds of processing time to compute the animation script. However, when the "walk" and corresponding "halt" commands were removed, creating the animation script took only 3 seconds. This wide difference in time is due to the fact that processing the "walk" command involves reading the data for the legs graph from a file. In either case, the processing time is not very significant.

5.3. Comparative Results

Animating human figures is such a complex task that research in human animation has been divided into smaller sub-areas. There seems to be general agreement that goal-directed systems are the most appropriate method for animating live figures because of their ability to produce complex realistic motion with minimum input from the animator. However providing the kind of high-level control a goal-directed system offers requires amalgamating solutions to the many problems of specifying human movement.

In the last few years, many ways have been proposed for controlling the movement of specific skills such as walking, sitting or grasping objects [Zeltzer 82b, Drewery 86, Korein 82a]. Algorithms that generate these movements in very realistic ways have also been developed [Girard 85, Wilhelms 87, Bruderlin 88]. However very little work has been done on character animation, where movement generation would concentrate on developing an individual character for an actor rather than on making the actor execute specific tasks. One area from which character animation can benefit greatly is facial animation. Apart from being able to simulate speech, animated faces can also register emotions [Pearce 86]. These results would blend nicely into a system where the animator would specify the actors' disposition, and the actors' faces would express their moods.

An actor's moods can also be expressed in the way the whole body executes movements. In Ridsdale's system [Ridsdale 87], an actor's path is chosen while considering the position of other actors in the room whom the first actor may like or dislike. However in this system the movements of the body are not considered. In the film *Rendez-vous à Montréal* [Thalmann 87], Humphrey Bogart and Marilyn Monroe evoke their own personalities, however the movements that display each of their characters had to be specified by the animator. In GESTURE, a system has been developed where actors will perform movements in their own characteristic way. The use of gesture specification functions implies that animators will have high-level control over secondary movement and need not specify it manually.

Chapter 6
Conclusion

The two main objectives of this thesis were to introduce a new type of motion to animation systems, secondary movement, for making animated figures more life-like, and to demonstrate that this movement can be produced through the use of a high-level specification. The GESTURE system demonstrates this by allowing the animator to assign personality traits and set moods for the actors, and then determining the secondary movement that the actors display. With a minimum amount of user interaction, animations of any length can be produced in which actors will carry out gestures appropriate to their character for the duration of the sequence.

The benefit of describing actors' characters rather than directly assigning them their secondary movement is that this process only needs to be done once. After the initial character defining stage, all primary movements will be executed by the actors while displaying secondary movement appropriate to their character. This means that the animator could specify the same goal for two actors with different character traits, and the goals would be accomplished in slightly different ways by each of the actors. GESTURE has demonstrated this by assigning the primary goal of walking to each of the actors Simon and Sally, and producing varying types of animations for each of them depending on the personality and moods that were given to them.

An important requirement in the design of GESTURE was that movements could be initiated and interrupted at any time. This feature is important because secondary movement is not planned motion and can vary with changing primary goals. The ability to interrupt movements at any time is also very useful if actors are to be able to react immediately to a new situation. In GESTURE there is complete flexibility in starting any movement at any time in the animation, even if the actor was previously carrying out another movement.

Since a model of the human body is an irregular structure, localizing knowledge about different

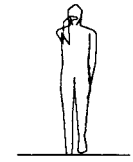
body parts seems most appropriate for animating human movement. The approach used in GESTURE allows specialized movement generating algorithms to be written for each movement supported in the system. The animation system therefore does not rely on one method for producing gestures. Each gesture specification function can apply the most suitable algorithm for executing the movement it is responsible for as realistically as possible. The graph serves as an underlying representation of all movements. While each gesture specification function handles movements on an individual basis, the graph is used to represent the results of any algorithm applied in the gesture specification functions.

The philosophy that was followed in designing GESTURE was to assume that the animator would like to concentrate on specifying primary goals for actors, and have the secondary movement incorporated without explicitly specifying this motion. However sometimes the animator may wish to intervene with the automatic selection of the secondary movement by the system. It would be interesting to investigate a way to compromise between animator and system specified secondary movement. A system addressing this problem should automatically generate movements from the animator's high-level specification, but should also allow the animator to interact at a lower level of movement specification for part or all of the animation sequence.

The graph representation in GESTURE can be used to store movement that has been specified from keyframing. The inbetween frames are automatically generated by the system to produce smooth movement. The automatic interpolation between keyframes is another example where the animator may desire finer control over the process. In section 4.2, the interpolation method used for the stage location of the actor was discussed. In this interpolation method three parameters affecting the continuity, tension and bias of the curve at key points can be used to control the result of the interpolation. In GESTURE these parameters use only default values in all the interpolations. However, varying the values of these parameters could result in the keyframed movements being performed in different styles. For example if an animator was interpreting the phrase "Suddenly he looked to the right" into an actor's motion, a discontinuity in the interpolation could be introduced at the moment the actor looks right to make the movement appear abrupt. An actor's movements could appear stiffer by increasing the tension parameter. An interesting area to explore would be to map a set of adjectives to values for continuity, tension and bias providing the

animator with a high-level language to control the interpolation process, and thus the style of the movements.

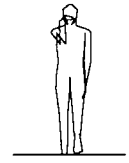
A direct extension of the work done in this thesis would be to incorporate other types of secondary movement. This would not only include other "standard" forms of secondary movement, but could also include movements that would be performed by old people or children, or people from different ethnic backgrounds. There is also a whole different set of movements that occur when people interact with one another, for example, hand signals people use while they express certain ideas [Duncan 77]. This research would be cross-disciplinary with the field of psychology. It would also be interesting to work with a psychologist to develop the expert system proposed in the ideal framework. Ultimately it would be very rewarding if many of these ideas and other research in human animation could be combined to produce a powerful and comprehensive goal-directed human animation system.



Appendix A
Glossary

Animation

Animation is a technique for creating the illusion of movement. This effect is achieved by displaying pictures at a rate above the flicker fusion frequency (about 20 frames per second), so that the viewer perceives them as one scene with moving components. Consecutive pictures must be very similar if the motion is to appear smooth. A computer animation system provides tools for increasing the speed at which an animation can be made and for improving the realism.



Articulated Body

An articulated body is a structure made up of links connected at spherical joints. The constraints on such a body are that the links are rigid (ie. not bendable) and that they remain connected at the joints. Such a structure therefore has 3 degrees of freedom, which specify the location of some fixed point in the structure, plus 3 degrees of freedom for each joint in the structure.

Articulated bodies can be represented as a tree structure where each node in the tree represents a joint connecting two or more links or segments. To ensure that the segments remain connected, the hierarchy of the tree structure can be used to dictate the transformations that are applied at a joint: for any node in the tree, the transformation matrices at all its ancestor nodes will be applied before the rotation specified at that node. The human body is generally represented as an articulated body.

Channel Table

In a keyframing system, the position and orientation of an object is specified for various frames in the animation sequence, and then an interpolation through these keyframes defines the position and orientation of the object for all the other frames. Before the interpolation is carried out, keyframes are stored in a channel table which is made up of many channels. A channel is a place to store the values of keyframes for one degree of freedom in the model of the object. So, for a rigid object which has 6 degrees of freedom, the movement of the object can be represented in a channel table using 6 channels.

Degrees of Freedom

The position and orientation of a structure in space can be completely defined using a combination of (x, y, z) co-ordinates and (θ, Φ, Ψ) angles of rotation around each of the axes. One of these parameters represents a degree of freedom of the structure. The number of degrees of freedom decreases as the structure is constrained by specifying values for these parameters. A rigid object has 6 degrees of freedom, 3 to specify its position in space, and 3 to define its orientation.

Dynamics

Dynamics is a method for specifying movement by supplying the forces and torques that act on parts of the body. The resulting motion is realistic because the laws of physics are applied to the body to compute the motion. In general, it is not intuitively obvious what motion will result when forces and torques are specified.

The inverse dynamics problem is concerned with determining the forces and torques to be applied in order to produce a given motion. For example, if the animator specifies the motion of kicking a ball, the system will compute the forces and torques to apply to produce this movement.

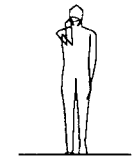
Event Driven

A program that simulates the occurrence of events over a period of time can be classified as a time-driven or an event-driven system. A time-driven system carries out the simulation by incrementing the clock by fixed intervals in time, and checking at each time if there are any events to process. Event-driven systems process a queue of events that have arrived in chronological order, and which have a time stamp identifying the time that the event occurred. At each step in the simulation, the clock is advanced to the time associated with the next event to be processed.

Frame

In this thesis, a frame is used in two contexts.

An animation frame is one picture in a sequence of pictures which when viewed in rapid succession give the illusion of movement.



A frame representation is a method used in artificial intelligence to store knowledge about a concept [Minsky 75]. A frame consists of slots, each of which describe an attribute of the concept, and relationships between the slots. The slots can form a PART-OF hierarchy and the frames are organized in an IS-A hierarchy. For example, a car IS-A motorized vehicle. The concept motorized vehicle could have a slot for engine, and a piston is PART-OF an engine.

Gesture

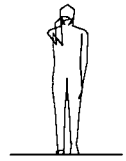
In the context of this thesis, a gesture is a movement which a person performs sub-consciously. This can be in the form of an action that begins and ends in the same position, such as scratching one's head, or can involve repositioning a part of the body, such as placing one's hands on one's waist. The collection of gestures is called secondary movement.

Goal-Directed

A goal-directed system is an animation system in which the animator specifies movement for the actors by assigning them goals. Goals can take on various forms. One type of goal is an action such as walk, sit or grasp. Goals can also be assigned to the ends of limbs in the human model to position the body, such as in reaching for an object while seated. The development of human animation systems is tending towards goal-directed approaches as it will simplify the task of specifying the movement for such a complex structure as the human body.

Interpolation

Interpolation is the process of determining a set of missing values between two known values. For example, in a curve-fitting program, a random set of points is given, and an interpolation will determine the points that will make up a curve passing through the given points. In keyframe animation, an interpolation between key positions of an object can help produce the 24 frames/second required for animation, relieving the animator from this tedious task. A more flexible interpolation routine will provide some form of control on the shape of the resulting curve.



Keyframe

In 2-dimensional animation, a keyframe is one picture that will be part of an animation sequence. One way for animators to quickly conceptualize what the movement will look like is to draw frames at intervals in the animation. These are the keyframes. When the animator is satisfied with this outline of the animation, the "inbetween" frames can be drawn. Producing animations through this method is called keyframing. In 3-dimensional animation, instead of keyframing the completed picture, positions and orientation of objects in the scene are defined, and the "inbetweening" process computes the transitions the objects make between the key positions.

Kinematics

Kinematics is a method for specifying motion by supplying position, velocities and accelerations for bodies without any regard to forces or torques that could be acting on the bodies. For example, rotoscoping and keyframe animation describe movement in terms of positions of the body over time.

In the inverse kinematics problem, the joint angles of a multi-link structure must be found such that the end of the structure will be in a specified position in space. Since this problem is under-constrained for a many-segmented structure, there are an infinite number of solutions. If the structure represents a human limb, such as an arm, the problem becomes one of choosing constraints that will determine a solution producing natural movement for the limb. This is a very difficult problem.

Kinesiology

Kinesiology is the study of human movement and human performance. These areas are studied with respect to the anatomical, physiological, mechanical, developmental, psychological and sociological aspects of movement. A typical application of kinesiology is in sports science where human movement would be studied to find ways in which to improve the efficiency in performance. In computer animation, kinesiology can be helpful by providing information about the way that people move naturally.

Knowledge Base

A knowledge base is a representation scheme for information which is structured in a way that is easy to understand and modify. Representation schemes are often classified into three categories: logical schemes (schemes that use logical formulae to represent knowledge), network schemes (schemes that use nodes and edges in a semantic network) and procedural schemes (schemes whose knowledge is retrieved by activating procedures) (see [Mylopoulos 83]). The information generally embodies some domain of knowledge, such as symptoms and cures for a class of

diseases. A knowledge base is created and maintained by a knowledge engineer who will work with an expert in the domain to capture his or her expertise in the knowledge base.

Motion Specification

The purpose of an animation system is to provide a means by which an animator can communicate to a computer how actors should move. The process of describing the actors' movements is called motion specification. Motion specification can take on many different forms. At a low level, actor's movements can be described by assigning values to all the actor's joint angles in every frame of the animation. This method is tedious, and so higher level ways of specifying movement are being explored. Ideally motion should be specified in a language familiar to the animator, and the animation system will produce the movements based on this motion specification.

Rigid Body

A rigid body is one which has no flexible parts or movable joints. The positions and orientation of a rigid object can be completely defined with 6 degrees of freedom. Examples of rigid objects are a box or a ball.

Robotics

Robotics is the study of machines (robots) that can simulate human activities. Many robots can perform complex activities that can aid humans in their work, such as handling hot or radioactive items. Techniques developed for robotics can be applied to human animation if the machines are modelled as humanoid figures. Algorithms that are used to control a robot's walking, lifting of objects or planning a path in a cluttered environment can be applied to animating figures.

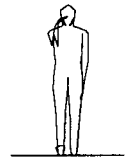
Appendix B
The Movement Script Language

The interface between the expert system and GESTURE is a script defined by a movement language. The format of the scripts is a series of lines each containing movement commands. Each line must be of the format:

clock_time gesture gesture_quality [gesture_quality]

The clock times on each line must be in increasing chronological order. These clock times represent frame numbers. Thus, approximately 24 frames will be equivalent to 1 second of animation. On the following page is a complete description of the movement language. The gesture qualities in bold face represent the default value used for that quality if none is specified.

Gesture	Qualities	Description
drop_arm	slow, **average**, fast left, right, **both**	drop the arm(s) to the rest position next to the body
on_waist	slow, **average**, fast left, right, **both**	place hand(s) on waist
in_front	slow, **average**, fast left, right, **both**	hold hand(s) in front of body
in_back	slow, **average**, fast left, right, **both**	hold hand(s) behind back
scratch	slow, **average**, fast left, **right**	scratch the head with a hand
wave	slow, **average**, fast left, **right**, both emphatically, **friendly**, shyly	wave with arm (emphatically or friendly) or with the fingers (shyly)
salute	slow, **average**, fast **right**	raise hand to head in a salute
rub	slow, **average**, fast left, **right**	rub the left/right eye with the left/right hand
relax_hand	slow, **average**, fast left, right, **both**	open hand(s) in a relaxed position
clench	slow, **average**, fast left, right, **both**	make hand(s) into a fist
flex	slow, **average**, fast left, right, **both**	extend hand(s) in a flexed position
look	slow, **average**, fast left, right, **straight** up, down, **ahead**	turn head in a given direction
nod	slow, **average**, fast left, right, straight	lower and raise head in a confirmative nod
straighten	slow, **average**, fast	straighten back into upright posture
slouch	slow, **average**, fast	slouch back in a bent posture
walk	slow, **average**, fast	accelerate and then begin walking
halt		decelerate from a walk and come to a stop



Appendix C
The Mock Expert System

In the ideal framework presented in chapter 3, an expert system is used to produce the movement language script. The expert system has not been implemented for this thesis. In its place, a "mock expert system" produces movement scripts based on the personality and mood definitions assigned to actors by the animator. The rules used to produce these movement scripts are shown below.

```
if (gloomy >= 8  OR  boredom >= 8  OR tiredness >= 9)
        then "0 walk slow", walk_speed is SLOW.
else if (cheerful >= 7)
        then "0 walk fast", walk_speed is FAST.
else
         "0 walk average", walk_speed is AVERAGE.

if (tiredness >= 7  OR  passive >= 9)
        then "0 slouch".

if (fear >= 5)
        then "0 slouch fast".

if (cheerful >= 2  OR  fear >= 5)
        then "0 look up".
        if (cheerful >= 5)
                then "0 in_back both".
else if (gloomy >= 2)
        then "0 look down".

if (domineering <= 5  AND  domineering > 0)
        then "0 clench both".
else if (domineering >= 6)
        then "0 on_waist".
else if (submissive >= 4)
        then "0 in_front".
else if (passive >= 5)
        then "0 flex both".

if (nervousness >= 2  OR  impatience >= 2)
        then "0 scratch left fast".
else if (tiredness >= 2)
```

```
        then "0 rub left".

if (nervousness >= 2 OR impatience >= 2 OR tiredness >= 2)
        then "45 drop_arm left", "45 relax_hand left".

"45 look left".

"45 straighten".

if (introvert >= 6)
        then "45 nod".
else if (introvert >= 1)
        then "45 wave shyly".
else if (extrovert <= 5)
        then "45 wave friendly".
else if (domineering >= 8)
        then "45 salute".
else
        "45 wave emphatically".

"90 drop_arm right".

if (domineering >= 5)
        then "90 clench both".
else if (passive >= 5)
        then "90 flex both".
else
        "90 relax_hand both".

if (cheerful >= 4)
        then "90 look right up".
else if (gloomy >= 2)
        then "90 look down".
else
        "90 look straight ahead".

if (nervousness >= 5 OR impatience >= 5)
        then "90 scratch right fast".
else if (tiredness >= 5)
        then "90 rub right fast".

if (fear >= 1)
        then "130 look straight up".
else if (cheerful >= 1)
        then "130 look left up".
else
        "130 look right".

if (tiredness >= 5)
        then "130 rub right".
```

```
else if (nervousness >= 5 OR impatience >= 5)
        then "130 scratch right".
else if (domineering >= 2)
        then "130 on_waist right".
else if (submissive > 5 OR passive > 5)
        then "130 in_front both".
else
        "130 drop_arm right", "130 relax_hand right".

if (walk_speed is FAST)
        then "150 look straight ahead", "150 halt".

if (walk_speed is not FAST)
        then
        if (cheerful >= 1)
                then "160 look right".
        else if (gloomy >= 1)
                then "160 look down".
        else
                "160 look ahead".

        if (nervousness >= 5 OR tiredness >= 5 OR impatience >= 5)
                then "160 drop_arm right fast", "160 relax_hand right".

        if (nervousness >= 1  OR impatience >= 1)
                then "160 scratch left fast".

        if (tiredness >= 5)
                then "160 slouch", "160 look down".

if (walk_speed is AVERAGE)
        then "200 drop_arm both", "200 relax_hand both", "200 halt".

if (walk_speed is SLOW)
        then "300 drop_arm both", "300 relax_hand both", "300 halt".
```

References

[Abel 85] Robert Abel and Associates.
SIGGRAPH Video Review, issue 20: The Making of Brilliance.
ACM, N.Y., 1985.

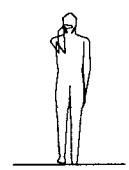
[Alias] Alias Research Inc.
110 Richmond St. East, Suite 500, Toronto, Ont., Canada, M5C 1P1.

[Armstrong 85] W.W. Armstrong and M. Green.
The Dynamics of Articulated Rigid Bodies for Purposes of Animation.
In *Graphics Interface '85*, pages 407-415. 1985.

[Bachant 84] Judith Bachant and John McDermott.
R1 Revisited: Four Years in the Trenches.
The AI Magazine 5(3):21-32, 1984.

[Badler 79] N.I. Badler and S.W. Smoliar.
Digital Representation of Human Movement.
Computing Surveys 11(1):19-38, 1979.

[Badler 80] N.I. Badler, J. O'Rourke and B. Kaufman.
Special Problems in Human Movement Simulation.
Computer Graphics (Proc. Siggraph '80) 14(3):189-197, 1980.



[Badler 87] Norman I. Badler, Kamran H. Manoochehri and Graham Walters.
Articulated Figure Positioning by Multiple Constraints.
IEEE Computer Graphics and Applications 7(6):28-38, 1987.

[Birdwhistell 70] Ray L. Birdwhistell.
Kinesics and Context: essays on body motion communication.
University of Pennsylvania Press, Philadelphia, 1970.

[Bruderlin 88] Armin Bruderlin.
Goal-Directed, Dynamic Animation of Bipedal Locomotion.
Master's thesis, School of Computing Science, Simon Fraser University, 1988.

[Bruderlin 89] Armin Bruderlin and Tom W. Calvert.
Goal-Directed, Dynamic Animation of Bipedal Locomotion.
Computer Graphics (Proc. Siggraph '89), to appear , 1989.

[Bull 83] Peter Bull.
Body Movement and Interpersonal Communication.
John Wiley & Sons Ltd., New York, 1983.

[Burtnyk 71] N. Burtnyk and M. Wein.
Computer-Generated Key-Frame Animation.
J. of SMPTE 80:149-153, 1971.

[Calvert 82] T.W. Calvert, J. Chapman and A. Patla.
Aspects of the Kinematic Simulation of Human Movement.
IEEE Computer Graphics and Applications 2(9):41-50, Nov. 1982.

[Calvert 88] Tom W. Calvert.
The Challenge of Human Figure Animation.
In *Graphics Interface '88*, pages 203-210. 1988.

[Calvert 89] Tom W. Calvert, Chris Welman, Severin Gaudet and Catherine Lee.
Composition of multiple figure sequences for dance and animation.
In *Proceedings of Computer Graphics International Conference*. Springer Verlag, June 1989.

[Chung 87] Chin Wah Tony Chung.
An Approach to Human Surface Modelling Using Cardinal Splines.
Master's thesis, School of Computing Science, Simon Fraser University, 1987.

[Csuri 81] Charles Csuri.
Goal-Directed Movement Simulation.
In *CMCCS '81/ACCHO '81*, pages 271-280. 1981.

[Cubicomp] Cubicomp Canada.
1550 Alberni St., Suite 450, Vancouver, B.C., Canada, V6G 1A5.

[Drewery 86] Karin Drewery and John Tsotsos.
Goal-Directed Animation using English Motion Commands.
In *Graphics Interface '86*, pages 131-135. 1986.

[Duncan 77] Starkey Duncan Jr. and Donald W. Fiske.
Face-to-Face Interaction: Research, Methods and Theory.
Lawrence Erlbaum Associates, Hillsdale, N.J., 1977.

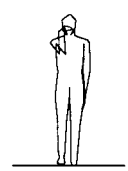
[Foley 82] James D. Foley and Andries Van Dam.
Fundamentals of Interactive Computer Graphics.
Addison-Wesley Publishing Co., Reading, Massachusetts, 1982.

[Girard 85] Michael Girard and A.A. Maciejewski.
Computational Modeling for the Computer Animation of Legged Figures.
Computer Graphics (Proc. Siggraph '85) 19(3):263-270, 1985.

[Kochanek 84] Doris H.U. Kochanek.
Interpolating Splines with Local Tension, Continuity and Bias Control.
Computer Graphics (Proc. Siggraph '84) 18(3):33-41, 1984.

[Korein 82a] James U. Korein and Norman I. Badler.
Techniques for Generating the Goal-Directed Motion of Articulated Structures.
IEEE Computer Graphics and Applications 2(9):71-81, Nov. 1982.

[Korein 82b] James U. Korein.
Using Reach Descriptions to Position Kinematic Chains.
In *Proceedings CSCSI/SCEIO, Saskatoon*, pages 79-84. 1982.



[McGhee 76] R.B. McGhee.
Robot Locomotion.
Neural Control of Locomotion.
Plenum Press, New York, 1976, pages 237-264.

[Miller 88] Gavin S.P. Miller.
The Motion Dynamics of Snakes and Worms.
Computer Graphics (Proc. Siggraph '88) 22(4):169-173, 1988.

[Minsky 75] Marvin Minsky.
The Psychology of Computer Vision.
A Framework for Representing Knowledge.
McGraw-Hill, New York, 1975.

[Mylopoulos 83] John Mylopoulos and Hector Levesque.
An Overview of Knowledge Representation.
On Computational Modelling: Perspectives from AI, Databases and Programming Languages.
Springer-Verlag, New York, 1983, pages 5-12.

[Parke 82] Frederic I. Parke.
Parameterized Models for Facial Animation.
IEEE Computer Graphics and Applications 2(9):61-68, 1982.

[Pearce 86] Andrew Pearce, Brian Wyvill, Geoff Wyvill and David Hill.
Speech and Expression: A Computer Solution to Face Animation.
In *Graphics Interface '86*, pages 136-140. 1986.

[Reynolds 82] Craig W. Reynolds.
Computer Animation with Scripts and Actors.
Computer Graphics (Proc. Siggraph '82) 16(3):289-296, 1982.

[Reynolds 87] Craig W. Reynolds.
Flocks, Herds, and Schools: A Distributed Behaviour Model.
Computer Graphics (Proc. Siggraph '87) 21(4):25-34, 1987.

[Ridsdale 86] Gary Ridsdale, S. Hewitt and Tom W. Calvert.
The Interactive Specification of Human Animation.
In *Graphics Interface '86*, pages 121-130. 1986.

[Ridsdale 87] Gary Ridsdale.
The Director's Apprentice: Animating Figures in a Constrained Environment.
PhD thesis, School of Computing Science, Simon Fraser University, 1987.

[Scheflen 72] Albert E. Scheflen.
Theory Body Language and the Social Order.
Prentice-Hall Inc., Englewood Cliffs, NJ, 1972.

[Shoemake 85] Ken Shoemake.
Animating Rotation with Quaternion Curves.
Computer Graphics (Proc. Siggraph '85) 19(3):245-254, 1985.

[Steketee 85] Scott N. Steketee and Norman I. Badler.
Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control.
Computer Graphics (Proc. Siggraph '85) 19(3):255-262, 1985.

[Sturman 86] David Sturman.
Interactive Keyframe Animation of 3-D Articulated Models.
In *Graphics Interface '86, Tutorial on Computer Animation.* 1986.

[Thalmann 83] Nadia Magnenat-Thalmann and Daniel Thalmann.
The Use of High-Level 3-D Graphical Types in the Mira Animation System.
IEEE Computer Graphics and Applications 3(9):9-16, 1983.

[Thalmann 87] Nadia Magnenat-Thalmann and Daniel Thalmann.
The Direction of Synthetic Actors in the *Film Rendez-vous a Montreal.*
IEEE Computer Graphics and Applications 7(12):9-19, 1987.

[Thomas 81] Frank Thomas and Ollie Johnston.
Disney Animation: The Illusion of Life.
Abbeville Press, N.Y., 1981.

[Wavefront] Wavefront Technologies.
530 East Montecito, Santa Barbara, CA, 93101.

[Weil 86] Jerry Weil.
The Synthesis of Cloth Objects.
Computer Graphics (Proc. Siggraph '86) 20(4):49-54, 1986.

[Wilhelms 85] Jane Wilhelms.
Using Dynamic Analysis to Animate Articulated Bodies such as Humans and Robots.
In *Graphics Interface '85*, pages 97-104. 1985.

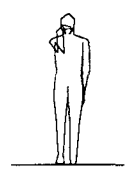
[Wilhelms 87] Jane Wilhelms.
Using Dynamic Analysis for Realistic Animation of Articulated Bodies.
IEEE Computer Graphics and Applications 7(6):12-27, 1987.

[Winston 84] Patrick H. Winston and Karen A. Prendergast, e.d.
Theory of Knowledge.
MIT Press, Cambridge, Massachusetts, 1984.

[Witkin 88] Andrew Witkin and Michael Kass.
Spacetime Constraints.
Computer Graphics (Proc. Siggraph '88) 22(4):159-168, 1988.

[Zeltzer 82a] David Zeltzer.
Representation of Complex Animated Figures.
In *Graphics Interface '82*, pages 205-211. 1982.

[Zeltzer 82b] David Zeltzer.
Motor Control Techniques of Figure Animation.
IEEE Computer Graphics and Applications 2(9):53-59, 1982.



[Zeltzer 83] David Zeltzer.
Knowledge-Based Animation.
In *Workshop on Motion*, pages 187-192. ACM SIGGRAPH/SIGART, 1983.