Goal-Directed Human Animation of Multiple Movements

Claudia L. Morawetz
Thomas W. Calvert

School of Computing Science
Simon Fraser University
Burnaby, British Columbia, Canada V5A 1S6

Abstract

The complexity of animating articulated bodies often results in unrealistic animations produced using cumbersome methods of movement specification. This paper presents a method of obtaining detailed and life-like human movement from a high-level specification. In the GESTURE system, the movement language script uses a powerful language to describe an actor's actions. Movements are carried out realistically using gesture specification functions which make use of various specialized algorithms for different movements. Movement continuity, and the ability for movements to interrupt one another, is obtained by the use of a movement control graph. We also show how this system can be based in a framework providing a straight-forward interface for the animator to produce realistic human animation.

Resume

La complexité dans l'animation des corps articulés souvent a pour résultat des animations pas réalistiques, produites par l'utilisation des méthodes encombrantes pour spécifier le mouvement. Cet article présente une méthode pour obtenir du mouvement detaillé et semblable au mouvement humain, à partir d'une specification de haut niveau. Dans le système GESTURE, la spécification du langage de mouvement emploie un puissant langage pour décrire les actions d'un acteur. Par le moyen des fonctions de spécification des gestes, le mouvement se réalise réalistiquement, à l'aide de plusieurs algorithmes spécialisés pour les mouvements divers. On obtient continuité de mouvement, ainsi bien que la capacité d'un mouvement donné d'interrompre un autre mouvement, par l'utilisation d'un graphe de control du mouvement. On démontre aussi comment ce système peut être basé dans un cadre qui fournit une interface directe pour la production d'animation humaine réalistique par l'animateur.

Keywords: Human animation, movement language script, gesture specification functions, graph, movement specification algorithms, secondary movement.

1. Introduction

The animal world is often considered the most difficult domain to animate realistically. This is due to the large number of joints which must be controlled in articulated bodies [5]. Depending on the level of detail, a model of the human body can have from 40 to 200 degrees of freedom [12]. Specifying values for each of these parameters over time is an unwieldy and unreasonable task for animators. In order to reduce the amount of work, goal-directed systems have been proposed. In these systems, the animator assigns animated figures high-level goals such as "grasp object" or "walk" and the system computes the joint angles over time that will produce a realistic execution of the specified goal.

These systems have tended to deal with only a few types of movements at a time [7, 8]. Although they may produce realistic animation of one particular movement, the animator is limited by the type of movement implemented. Conversely, conventional animation systems that give animators complete freedom over the actors' movements provide little support in the realistic execution of these movements.

GESTURE combines both of these advantages by requiring only a high-level description of the movements to be executed as input, and using algorithms from various systems to produce realistic movement. The algorithms or data for different movements are integrated into the system through *gesture specification functions*. A graph representation enables movements to follow and interrupt one another. This feature implies that the animator can specify any order and timing of desired movements for an actor. The animator can now play the role of a stage director, who gives directions indicating the desired movements, and the actors execute them convincingly [14]. The resulting system is goal-directed, providing many high-level commands to the animator, and can apply the most suitable movement specification algorithms to each goal to attain as realistic an execution of each movement as possible.

2. The GESTURE System

The objective of the GESTURE system is to read a high-level movement language script for each actor and to produce an animation script containing a complete description of all joint angles for each actor for every frame of the animation. Figure 1 shows the various components of GESTURE.

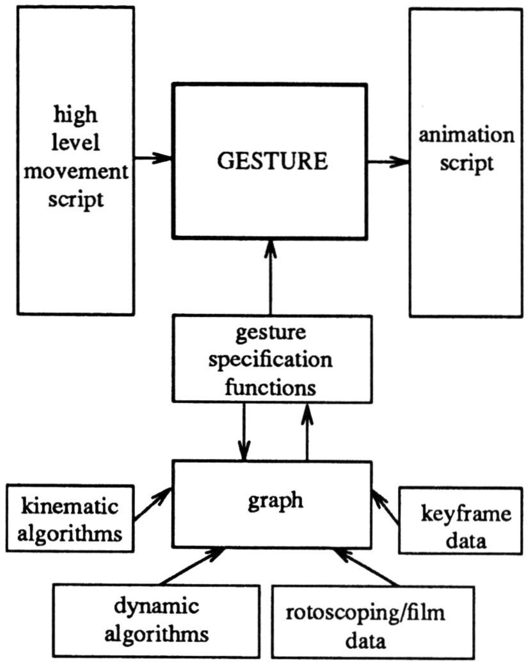

Figure 1: Components of the GESTURE system

GESTURE produces realistic animation of the specified movements in the high-level movement script by consulting *gesture specification functions*. Each of these functions is responsible for executing only one particular movement, and can therefore make use of any specific knowledge about how that gesture is performed by humans. As each of these functions is independent of the others, existing algorithms and/or data or new specialized algorithms for particular movements can be incorporated into the animation system as appropriate.

The movements produced by the gesture specification functions are coordinated using a representation similar to Zeltzer's frame-based approach [18]. Movements are encoded in a graph by storing joint angles in the nodes, which can be looked upon as key positions. Executing a sequence of movements is equivalent to traversing the graph. Arcs are labelled with names of different movements, and a number indicating the number of frames between key positions, i.e. the surrounding nodes. This graph representation is a natural way of encoding keyframed movement, or any movement that is represented by data at key times, such as film or rotoscoping data. It will also be shown that the gesture specification functions can make use of the graph to generate any kind of movement that uses an algorithm producing joint angle data, such as kinematic [10, 11] or dynamic [1, 4, 16] simulations.

To demonstrate the flexibility in combining and interrupting movements using the graph representation, most of the movements available to the animator in GESTURE have been chosen from a set of gestures called *secondary movement*. Unlike *primary movement* which consists of actions that are carried out to accomplish a specific goal such as grasping an object, these gestures are carried out largely sub-consciously, and are closely correlated to personality traits [3, 15]. Examples of secondary movement include lowering or scratching the head, or brushing hair out of the eyes. These types of movements can be useful for animating a crowd scene in which all actors are given roughly the same primary goals, but individual actors' personalities dictate differing secondary movement. These gestures may occur in many combinations and may have to be interrupted if the actor's environment changes. Thus, these movements are suitable to exploit the capabilities of using the graph representation in GESTURE. The next four sections present details of the GESTURE system.

3. The Movement Language Script

The movement language script is the high-level description of actor movements which is used as input to GESTURE. The movement script contains a chronological list of movements. There are two types of movements which can be found in a movement language script: *action* and *destination* movements. A destination movement involves placing a part of the body in a certain end position. Action movements involve positioning a part of the body, and then continuing to cycle through several body part positions for a duration of time. Examples of destination movements in GESTURE are a confirming nod, making a fist, or putting the hands on the waist. Examples of action movements are scratching, waving or walking.

The distinction between action and destination 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 are generated until that movement has been completed. However, if the movement script requests the execution of an action movement, 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 involved in the action movement, first to take them to the starting position for the movement and then to cycle through the series of positions which characterize the movement. The end of an action movement is indicated by the beginning of another movement that conflicts with the execution of the action movement. When an action movement is ended, the object corresponding to that movement is removed.

The termination of an action movement occurs only when it has been interrupted by another conflicting movement. This ability to interrupt gestures provides greater flexibility in specifying the timing of movements. It is also critical to implementing secondary movement which consists of random, sub-conscious movement. In Zeltzer's system [18], actors can switch between several completed movements: walking, sitting, and lying. However, his system is not able to have the actor begin to sit up and then lie down again, for example. In GESTURE, interruptions are not limited to action movements which must be interrupted in order to terminate. In fact, the graph allows interruptions to occur between any two movements.

As with commands in goal-directed systems, the movement language script consists of English movement commands, such as "scratch" or "nod". The format of the movement language script is a series of movement command lines. Each line specifies a time (frame number) at which a movement should begin, the movement, and an optional collection of words which qualify the movement. Command lines must be sorted in chronological order by the specified starting time of the movement. Movements are specified by words uniquely defining the movement. Each movement can be qualified by words which alter the style of the movement produced. In the absence of qualifying words, defaults are chosen. The qualifying words which may be used vary for different movements. The command line

```
15    look up
```

causes 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" are 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 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 2: A sample movement language script

Figure 2 shows an example of a movement script that can be used by GESTURE to produce an animation. Explanations of the movements are in italics. Note that the head scratch which was begun at frame 20 will stop when interrupted by the head turn at frame 50. If the command to place the left hand on the waist had been omitted, 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.

GESTURE also ensures that the transitions between movements occur in a natural way. 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 it is physically impossible. It is up to the gesture specification function dealing with head scratches in conjunction with the graph to specify a realistic path such as moving the hand in front of the body and then up to the head. Thus, the movement script in figure 2 can be produced without concern for how natural transitions between the movements will occur.

4. Producing the Animation Script

The animation script produced by GESTURE contains the joint angles and body location for each actor for every frame of the animation. The control structure for reading the movement script and producing the animation script is relatively straightforward and is based on channel tables. In our case, each channel stores angle values for one joint in the body model at key times. This control structure is event driven, i.e. the joint angles for the animation script are computed in sequential order, and 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.

The gesture specification functions provide the basis for calculating the joint angles at the keyframes specified in the channel tables. Values for every frame can then be obtained by interpolating between these keyframes. 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 movements that occurred previously, although on-going gestures can be interrupted.

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. 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 is encountered in the script which uses the same joints as the active action movement. The creation and deletion of action objects will be elaborated upon in the discussion about gesture specification functions.

The overall control structure of GESTURE can be summarized with the pseudo-code shown in figure 3. 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 (line 3). The processing of each command line in the movement script initiates a new event. The clock is pushed forward to the time

```
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 corres-
               ponding 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 3: Control Structure for GESTURE

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 current movement 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 system, and requires no knowledge about how particular movements are executed. The following two sections will discuss what happens at line 8 of the algorithm: the achievement of realistic movement from a descriptive movement command line.

5. A Graphical Representation for Movement Specification

The notion of a graph, in which the nodes represent key body positions and the arcs represent movements between these key positions has been developed. This builds on ideas presented by Zeltzer [18]. In implementing this approach a number of problems had to be addressed, particularly the situation where one movement interrupts another.

When a gesture specification function is activated, the current position of the body must be known before keyframes for the movement can be generated. The problem can be made tractable if sets of joint angle values are grouped together and associated with a small number of states representing body positions. These states can be represented by the nodes of a graph. 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 needed to make a 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. If an intermediary state is introduced between these two states where the hand is slightly distanced from the side of the body, then the movement of the arms from behind the back to the front of the body appears more realistic.

In figure 4, the named nodes are those labelled *rest*, *on_waist* and *wave*, and the intermediary node is unlabelled. The labels on the arcs in the graph are the names of the 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 itself.

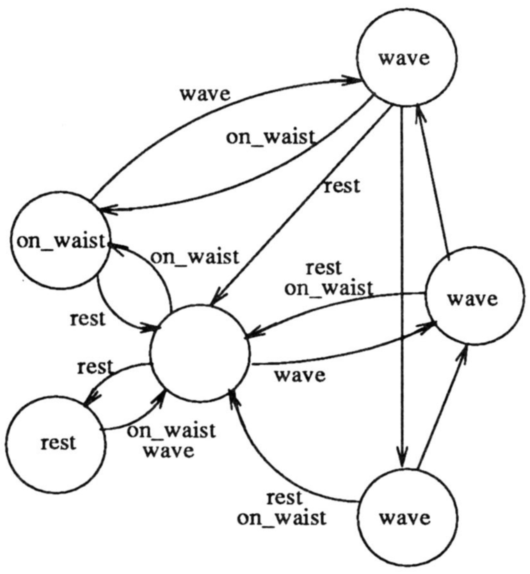

Figure 4: Graph with the action movement: "wave"

Each arc must also carry some measure of the temporal distance between nodes so that interpolation can be generated appropriately. Since nodes represent arbitrarily defined body positions, the time to travel between any two nodes will depend on the physical distance and the nature of the gesture. Thus, 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 instead of reaching one body position, cycle through several positions; for example, a head scratch or a wave. In the case of an action movement, the same name is assigned to several nodes, and an ordering is defined for cycling between them. In figure 4, the wave gesture is an example of an action movement.

Since the nodes in the graph correspond to body positions, the graph would become unmanageably large if all possible body positions were included in one graph. One way in which the number of nodes can be reduced is to take advantage of the fact that most gestures do not involve the whole body; usually a gesture is performed by one arm, the head or the torso or some simple combination. For example, a wave is done by an arm, a nod by the head, and a slouch by the torso, while a head scratch uses an arm and the head. To reduce the number of nodes and to 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 figure 4 - rest, on-waist, wave - are in a graph that records only joint angles for the arm. Execution of some body movements requires the traversal of several graphs concurrently, with each of the graphs controlling the movement of one set of joints.

If one movement interrupts another when the body is "at" a node, it suffices to follow the arc emanating from the node labelled by the new movement. However, some time is required to travel between nodes, and thus it is likely that movements will be interrupted while an arc is being traversed. In order to capture the complex timing of secondary movement, GESTURE has been designed to allow movements to be interrupted and followed by other movements at any time. To achieve this, a procedure is needed to go from an arbitrary body position between two states to a known state, without the need to numerically analyze the body position.

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 is not accessible in the graph after that temporary node has been passed. 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 additionally, the distance measures which 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 5 shows an example of a movement of the arm from rest position to the hand on the waist position. (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 adopts an arc with the label "back" and distance measure 2. Once this arc has been traversed, the movement can continue by following the appropriate arc leading from the intermediary node B. 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.

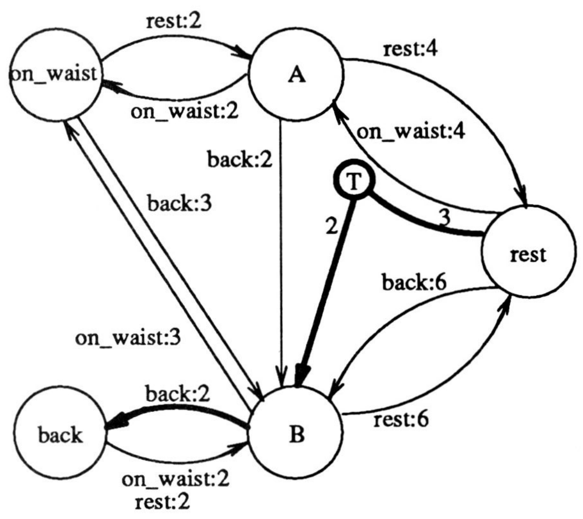

Figure 5: Graph with a temporary node caused by an interrupting gesture

6. 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 a different number of joints in each limb, and constraints on the physically possible rotations of the joints vary. Human animation must therefore appeal to specialized algorithms which reflect the particular characteristics of each body part. In GESTURE these are the gesture specification functions. 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 previously specified keyframes after 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 interpolating 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, but 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, 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.

Consider the following example of a gesture specification function which controls the movement of a head scratch. During a head scratch, the movements of the arm and hand must be coordinated with the movement of the head. Specifically, 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 reaches the position to scratch before the head, it is important that the scratching cycle does not begin until the head has been properly positioned. Thus 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 head scratch 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.

Most movements can be generated by the gesture specification functions using just the two tools *anchor* and *traverse_graph*. However, the flexibility of the graph is demonstrated by the fact that not only can it represent movement that has been specified by keyframing, but it can also handle movement generated other ways. The primary movement *walk* serves as an example of movement produced from a dynamic simulation and incorporated into the graph [4]. Joint angles produced from the simulation are assigned 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. In general, this is a way to incorporate the data from any movement generation algorithm that supplies values for every frame. Thus the graph could also represent movement from film data or rotoscoping, for example.

Human walking also involves swinging each arm in synchrony with the opposite leg. An arm swing is initiated by the *walk* gesture specification function using a kinematic description of the movement of the arm as a function of the movement of the leg. 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. In order to meet this requirement, the gesture specification function modifies the arm graphs by introducing 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. The arc lengths to and from these nodes are calculated based on the number of frames in the walk cycle that would synchronize the arm swing with the legs. Thus the graph representation can also incorporate kinematically computed movements.

Assigning responsibility for the execution of single gestures to individual functions has proved to be an appropriate method for controlling movement. Since each gesture requires special knowledge about how it is to be executed, gesture specification functions are a good way to organize the knowledge about 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. This approach to human animation provides a flexible and general way to produce and coordinate different types of movements.

7. Framework for a Comprehensive Human Animation System

A major difficulty in designing human animation systems is to strike a balance between obtaining realistic movement from a complex articulated body and providing a relatively straightforward but non-tedious interface to the animator. GESTURE contributes much towards this goal, as its input is a high-level language, and it provides a mechanism to incorporate algorithms or data from other systems, increasing the number of realistic movements that can be executed in combination. Ongoing research in computer graphics is examining ways to animate human figures realistically [2, 4, 16]. In this section, a front-end to GESTURE is proposed in which the animator would need to provide minimal specification.

In GESTURE, except for the movement of walking and swinging the arms, all other gestures are examples of secondary movement. This class of movements was chosen as they best demonstrated the capabilities of the graph to

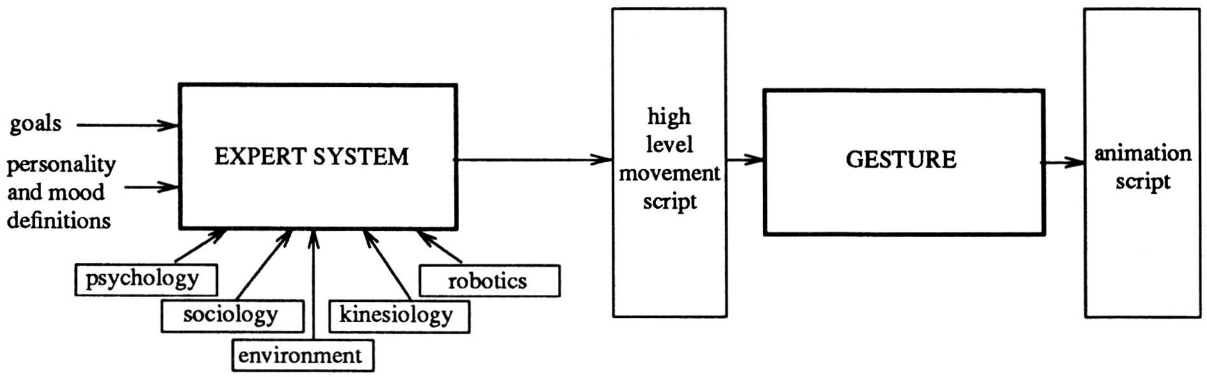

Figure 6: Framework for a human animation system

combine and interrupt movements. They were also chosen because they represent a very different type of human movement that has not been addressed so far in the graphics animation literature. Without secondary movement, a person's movements look stiff and robot-like. An ideal human animation system would automatically provide the secondary movement consistent with the high-level goals assigned by the animator to each of the actors.

The field of artificial intelligence investigates ways in which a computer can aid humans in tasks requiring human intelligence. An expert system which would select appropriate secondary movement for actors would be valuable to a human animation system. Figure 6 suggests a complete framework for a goal-directed human animation system that produces life-like human movement. Initially, the animator supplies information pertaining to each actor's personality and moods. This could include how nervous or impatient, or how assertive or domineering an actor is supposed to be. Then, high-level goals will be assigned to the actors, and the expert system will assign secondary movement to be executed in conjunction with the desired movements. This expert system produces a high-level script which can be used as input to GESTURE to produce life-like animations.

A human animation system based on this framework, and incorporating a wide variety of realistic human movements would be a very powerful tool for an animator. In our implementation, the complete framework has been implemented except for the expert system. This demonstrates that this is a feasible approach to a comprehensive human animation system.

8. System Evaluation

A new approach to character animation has been presented in which secondary movement is added to animated actors performing primary movement. The GESTURE system is fully implemented and provides straight-forward ways in which new movements can be added to the system. The remainder of the framework has been simulated so that the full effect of the animation system can be experienced. The way in which this environment is simulated will now be described.

Our system has only two actors, Simon and Sally. Walking is the sole primary goal they may be assigned. A mock expert system has been implemented to replace the expert system in the framework. This mock expert system applies very simple rules about how personality and moods govern human behaviour to choose secondary movement for an actor. In GESTURE, a set of sliders 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).

Our scripting language consists of a maximum of twenty movements, with up to three qualifying adjectives per movement. The language is expressive, using a natural description of the movements and their style, and is succinct, so that translating the many styles of movements into an animation is not too complex. Adding a new gesture to our system is not difficult, and one could quickly develop a large selection of movements from which an expert system could choose. To add a new gesture, a corresponding gesture specification function must be created as well as new nodes and arcs. It would be quite feasible to implement a simple program which would allow the user to interactively build new gestures.

The incorporation of secondary movement gives each of the actors their own identity, and it is interesting to see how their movements evoke the personality and moods they were assigned. GESTURE has been used to generate numerous animations, and its use has proved how easy it is to alter the actors' personality and moods, and quickly obtain new, appropriate secondary movement [13].

A major contribution of this work has been the development of the graphical representation for movement. This representation has proven to be an effective way to coordinate sequences of movements. The graph also allows interruptions of gestures and the continuation of movement to other gestures. One great strength of this representation is that the graph can incorporate motor programs or data generated independently. GESTURE uses keyframe data collected from a keyframing system, COMPOSE [6], and data produced from a dynamic simulation of a walk [4]. Other types of movement generating algorithms could easily be used.

In the GESTURE system, we have explored only a small set of the gestures people perform. The existing program can serve as an experimental tool for animators in trying out new types and combinations of gestures. This program could also be beneficial to researchers in sociology who study body language. Sociologists and psychologists could work with a knowledge engineer to develop the expert system suggested in the framework. The GESTURE system can be readily used to evaluate the resulting scripts produced by such an expert system. Many such ideas have emerged during the implementation of GESTURE, which could lead to interesting new topics of research.

9. Conclusion

In the last few years, many ways have been proposed for controlling purposeful movement of specific skills such as walking, sitting or grasping objects [8, 11, 17]. Algorithms that generate these movements in very realistic ways have also been developed [4, 9, 16]. In order for much of the human animation research to be beneficial to animators, these algorithms need to be combined in a cohesive system in which different combinations of movements can define an animation. GESTURE's graph representation of movement suggests a method for controlling human movement via gesture specification functions, which can be tailored for each incorporated movement algorithm or collection of data. A large enough set of test movements has been included into our system to demonstrate the success achieved in combining and interrupting different gestures.

The decision to use secondary movement as our representative set of movements in GESTURE has proved to be very interesting. By incorporating secondary movement in GESTURE we have produced animations in which the figures evoke markedly different personalities while executing the same goal (walking) with very little guidance by the animator. These results show that life-like human figure animation is possible without dramatically increasing the complexity of the animator's task.

Evaluation of the output of the GESTURE system suggests that the approaches presented here are useful. However, additional work is still required to develop a powerful human animation system which will give a richer and more varied set of movements capable of responding to subtle changes in actors' moods and the environment. With the combined efforts of psychologists and sociologists encoding their observations of body language into an expert system, and researchers in kinesiology and robotics developing realistic movement motor programs, convincingly life-like animations could be produced in such an animation system.

References

1. W.W. Armstrong and M. Green. The Dynamics of Articulated Rigid Bodies for Purposes of Animation. Graphics Interface '85, 1985, pp. 407-415.

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

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

4. Armin Bruderlin and Thomas W. Calvert. "Goal-Directed, Dynamic Animation of Human Walking". *Computer Graphics (Proc. Siggraph '89) 23* (1989), 233-242.

5. Thomas W. Calvert. The Challenge of Human Figure Animation. Graphics Interface '88, 1988, pp. 203-210.

6. Thomas W. Calvert, Chris Welman, Severin Gaudet and Catherine Lee. Composition of multiple figure sequences for dance and animation. In *New Advances in Computer Graphics, R.A. Earnshow and B.Wyvill (eds)*, Springer, Verlag, Tokyo, 1989, pp. 245-255.

7. Charles Csuri. Goal-Directed Movement Simulation. CMCCS '81/ACCHO '81, 1981, pp. 271-280.

8. Karin Drewery and John Tsotsos. Goal-Directed Animation using English Motion Commands. Graphics Interface '86, 1986, pp. 131-135.

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

10. James U. Korein. Using Reach Descriptions to Position Kinematic Chains. Proceedings CSCSI/SCEIO, Saskatoon, 1982, pp. 79-84.

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

12. R.B. McGhee. Robot Locomotion. In *Neural Control of Locomotion*, Plenum Press, New York, 1976, pp. 237-264.

13. Claudia L. Morawetz. A High-Level Approach to the Animation of Human Secondary Movement. Master Th., School of Computing Science, Simon Fraser University,1989.

14. Gary Ridsdale. *The Director's Apprentice: Animating Figures in a Constrained Environment.* Ph.D. Th., School of Computing Science, Simon Fraser University, 1987.

15. Albert E. Scheflen. *Body Language and the Social Order.* Prentice-Hall Inc., Englewood Cliffs, NJ, 1972.

16. Jane Wilhelms. "Using Dynamic Analysis for Realistic Animation of Articulated Bodies". *IEEE Computer Graphics and Applications 7*, 6 (1987), 12-27.

17. David Zeltzer. "Motor Control Techniques of Figure Animation". *IEEE Computer Graphics and Applications 2*, 9 (1982), 53-59.

18. David Zeltzer. Knowledge-Based Animation. Workshop on Motion, ACM SIGGRAPH/SIGART, 1983, pp. 187-192.