Towards an Integrated View of 3-D Computer Character Animation

David Zeltzer
The Media Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

ABSTRACT

To automate character animation and extend it to 3-D we need to create and manipulate three-dimensional models of articulated figures. Abstraction and "adaptive motion are key mechanisms for dealing with the degrees of freedom problem, which refers to the sheer volume of control information necessary for coordinating the motion of an articulated figure when the number of links is large. A three level hierarchy of control modes for animation is proposed: guiding, animator-level, and task-level systems. Guiding is best suited for specifying fine details but unsuited for controlling complex motion. Animator-level programming is powerful but difficult. Task-level systems give us facile control over complex motions and tasks by trading off explicit control over the details of motion. The integration of these three control levels is discussed.

1. Animation as Simulation

Currently there is much controversy about the nature of 3-D computer animation. Should such systems by based on simulation, keyframing, or an animation programming language? Should the interface be through graphical input devices, or through the keyboard? It is my purpose here to provide a conceptual framework for 3-D computer animation in general, and character animation in particular.

Automatic inbetweening has been the focus of attention in moving from conventional to computer-assisted 2-D animation [6,9]. From this point of view, it seems a natural extension to apply automatic inbetweening to 3-D animation, and in fact, a number of such systems have been developed [10,18,25]. Here I will argue that 3-D keyframing belongs to the lowest level of a three level hierarchy of control regimes for animating articulated figures.

In order to produce convincing character animation, conventional 2-D animators refer constantly to living models, or study motion pictures of living models, on draw directly from still frames of such motion pictures (rotoscoping). That is, character animation is not a process of transforming lines and shapes on 2-D surface, nor is it simply the art of squashing, stretching, or otherwise exaggerating and caricaturing motion. Thomas and Johnston [41] make quite clear that the great success of the Disney animators was due in large part to the long hours they devoted to studying and observing the movements of humans and animals in preparing for a particular sequence. A character was successful precisely in proportion to how well the animator understood the kinematics of the figure, the structure and timing of a movement, and the effects of a movement on soft tissue and clothing. Once these elements were mastered, only then could the animator develop a character's personality by the judicious exaggeration or de-emphasis of particular attributes.

As long as animation requires the generation of many drawings by hand, simplicity and economy will be essential elements. 3-D computer animation, however, is an entirely different medium. The animator's energy is no longer invested in drawing and the tedium of inbetweening. Instead, the focus is on creating an environment -- designing microworlds and populating them with interesting characters. 3-D computer animation is thus a process of simulation in its most general sense: the specification of objects and transformations on objects. In this paper we will look at the specification and coordination of the behavior of the objects we wish to animate. There are three basic approaches.

(1) We can explicitly describe the behaviors we are interested in. This is the guiding mode.

(2) We can describe behaviors algorithmically, in some programming notaion. This is the animator level.

(3) Lastly, we can describe behavior implicitly, in terms of events and relationships. This is the task level.

For our purposes we can consider the domain of 3-D computer character animation to be the control and coordination of the motion of articulated structures made up of rigid links. Differential scaling and other shape transformations. In the next section I examine the fundamental problem we face in trying to coordinate the motion of articulated figures, and we will look at mechanisms for dealing with the complexities of figure animation. Later we will see that the graded implementation of these mechanisms gives the above three-tiered hierarchy of control modes.

2. The Degrees of Freedom Problem

The essential problem of coordinating the motion of an articulated figure is to generate appropriate values of the joint variables that control the positions and orientation of each link. Joints may be modeled as lower pairs [13], such as rotary or sliding joints, or they may be more complex, as in a detailed model of the human knee. For a figure with n joints, we can think of an n-dimensional pose space, where we assign a coordinate axis to each of n degrees of freedom; and an n-component pose vector, which completely specifies a particular configuration. To animate the motion of a jointed figure, a pose vector must be specified for each frame of the sequence. To animate a minute of complex motion for a reasonably detailed figure, say, with 30 links, tens of thousands of values will need to be specified for the joint variables to display a new configuration each frame. Even if the sequence is keyframed, with a keyframe every two seconds, 30 pose vecors -- nearly a thousand values -- need to be specified. This is an example of the degrees of freedom (DOF) problem [42], which refers to the sheer volume of control information necessary for coordinating the motion of an articulated figure when the number of links is large, as in a human figure. It is the reason why animators find 3-D character animation so tedious.

Of course we are not interested in random motion; the movements of a figure must be "correct" in some sense to be of any use -- the robot programmer may wish to optimize energy expenditure, for example, and the animator will want the figure to move in some expressive manner. Viewed in this way, character animation is problem of search. Not only do we have to generate pose vectors, we need to find a particular set of variables out of an immense pose space -- if each of 30 joints has only 3 possible positions, there are over a billion potential configurations!

To complicate the problem, many figures of interest are kinematically redundant, possessing "extra" degrees of freedom that allow multiple solutions -- perhaps an infinity of pose vectors -- all of which satisfy a particular movement problem. The human arm, for example is redundant -- you can reach for an object with the elbow held high, low, or in between. That is, there are many arm configurations that will position the hand at some fixed location in space. This redundancy gives us the extra flexibility we need to reach around and over objects, and, in general, to maneuver in a cluttered environment. And it is why individuals can develop characteristic and expressive "styles" of movement.

For animation, of course, we are not interested in a single configuration, but a sequence of pose vectors -- a "hyper-path" through a many-dimensional pose space. So it is not surprising that even when the main features of a motion are known, it may take an animator many iterations to get the movement "just right".

In the next sections we look at two important techniques for feeling with the degrees of freedom problem.

3. Adaptive Motion

By adaptive motion I mean the ability of a figure controllor to use information about the environment and the figure itself in the control process. That is, feedback can be used to guide the search through the huge space of potential configurations. To do this, at least the location and orientation of objects and their surfaces must be available to the animation software, and not just to the rendering programs, as is usually the case. Physical interactions between figures and objects are so ubiquitous in the real world -- touching, grasping, pushing, not to mention locomotion over a wide variety of surfaces -- that automatic collision detection, long of interest in CAD/CAM and robotics, should become an integral part of the animation environment.

Adaptive motion makes possible goal-directed and constrained behavior, since it allows the animator to describe movement in terms of relations among objects and figures. It lends generality to animation sequences, since animation software can adjust motion sequences for different scenes. This helps to hide unnecessary detail from the animator, since the burden of generating much of the control information can be left to the animation software.

Reynolds [32] has suggested that it would be desirable if an animator could establish "rules of behavior" for objects and characters in some imagined microworld. After establishing the initial conditions of this simulated universe, the animator would sit back and let the animation system generate the sequence. This amounts to a 3-D, computerized extension of the <u>straight ahead</u> style of 2-D animation [4]. Adaptive motion makes possible the extension of this technique to 3-D computer animation.

<u>4. Abstraction</u>

The importance of abstraction in dealing with the intellectual complexity of computer programming is well-known [36] and it is a basic tool for dealing with the kinematic and behavioral complexities of articulated motion as well.

There are five kinds of abstraction useful for controlling character animation: structural, procedural, functional, character and world modeling.

<u>4.1. Structural Abstraction</u>

A structural abstraction describes the kinematic properties of a figure, i.e., the transformation hierarchy, the nature of the allowable joint motions, and whether links are rigid or non-rigid (although we will deal only with rigid motion here). The notion of a transformation hierarchy is a generalization to 3-D of the familiar 2-D instancing systems described in graphics text. Most 3-D animation sytems provide some means of represent ing transformation hierarchies, e.g., Crow's <u>scn assmblr</u> [12], Blinn's <u>artic</u> [5], and Reynold's ASAS [32]. In these systems, joint transformations are represented as simple rotations and translations, sometimes including scaling, although more general representations for articulated motion have long been used in the field of mechanism design [13], and more recently, robotics [29]. <u>sdl</u>, the skeleton description language, is the tool for specifying structural abstractions for use in <u>sa</u>, an articulated motion system described in [46]. At the M.I.T. Arts and Media Laboratory we are developing a set of graphical tools, implemented on a Symbolics 3600 Lisp machine, for designing and editing kinematic descriptions of jointed figures.

<u>4.2. World Modeling</u>

In the physical world objects and figures interact in complex ways at many levels of detail. Just as we need mechanisms for representing figures, we need some means of describing the physical and functional properties of objects in the environment. Adaptive motion requires at least efficient geometric representations for collision testing and path planning; goal-directed animation control requires in addition sophisticated machanisms for knowledge representation.

Part of the problem involves structuring high-density graphical data bases to avoid exhaustive searches through long lists of surface elements to do, say, collision testing. Rather, we want to consider only those objects in the scene that are "near" the figure. This means that the data base must be carefully organized spatially so that searches always proceed at the appropriate level of detail. Various hierarchical methods of structuring data to speed up occlusion testing have been reported, e.g., [11,16,35]. Franklin [15] describes a set of algorithms which are useful for intersection testing as well.

Another problem is to represent attributes, functionality, and relationships of objects in a scene, so that the animation system can plan motion sequences in complicated environments. If a script specifies, say, that a character is to "leave the room", the animation system should be able to infer, among other things, that

> To leave a room, use the door.
> The current room has a door at (x,y,z).
> To open the door, move to (x,y,z) and
> invoke skill S.

The problem of knowledge representation is currently a major area of research in artificial intelligence. At The Media Laboratory we are developing a set of design goals and specifications for a knowledge-based animation system capable of common sense planning and problem solving in simple environments. For a discussion of knowledge representation issues in computer animation, see [47].

<u>4.3. Procedural Abstraction</u>

A procedural abstraction [39] is the representation of a movement algorithm <u>independent</u> of the structure of the figure it controls.

For example, the DOF problem is not so severe in the case of a robot manipulator with six or seven joints. Even so, humans are not good at calculating the ncessary joint angles for controlling even a simple manipulaor, and resolved motion [20,44] is generally used. That is, the position and orientation of a target location are input, and the manipulator controller automatically computes the pose vector necessary to reach it.

Resolved motion control is an example of the use of procedural abstraction in the solution of the DOF problem: a computation is specified that will transform the input parameter, i.e., the position and orientation of the target, into the output object, here the set of joint angles that will position and orient the end effect or at the desired location in the workspace, if possible. Resolved motion control is independent of a particular kinematic structure, and can be applied to figures of 6, 8 or more links, and for, say, a human figure, can be applied equally well to control either arms or legs [33]. Other examples of procedural abstraction are the computation of trajectories for falling objects, the computation of the paths of colliding objects, or the use of spline curves for generating smooth motion. Such facilities are often provided for the animator ready-made, but may be constructed by the animator in animation systems embedded in high-level programmming languages.

4.4. Functional Abstraction

For the robot arm the number of links is small and the arm is treated as a single kinematic entity. But for a figure with many links, we want to be able to group together both the structural elements and the procedures that are necessary to effect a particular class of motions. Alternatively, we can impose constraints on the movements of a set of joints. We call such a grouping a functional abstraction.

Functional abstractions are important because they allow the animator to factor the pose space into motor skills. If we already know the general "shape" of a motion, we need only consider a subregion of the total pose space. Say we want a figure's hand to grasp an object -- we already know which joints need to move, roughly how they should move, and moreover, we know this is a useful motion that we want to repeat often. We can cluster this group of joint movements around the task "to grasp", and attach one or more procedures to implement it (perhaps resolved motion). Once this motor skill has been defined, the details of its execution can be suppressed. That is, we need only supply the appropriate parameters, e.g., target location, fast or slow, hard or soft, to the motor program for the grasping skill. By specifying functional abstractions for grasping and other tasks, the animator is spared the burden of generating pose vectors and can instead think of the figure motion at a higher level -- in terms of the tasks and events that are to be performed.

Functional abstractions allow us to attach implicit goals to figure motion. By decomposing a figure's potential movements into a repertoire of skills we can associate the events and relationships the animator specifies with the skills (implemented as functional abstractions) that the figure controller "knows" about. Moreover, if we allow functional abstractions to refer to other functional abstractions, it is possible to construct behaviors as compositions of simplier movements.

4.5. Character Abstraction

We would like to abstract a step further from these structural and functional descriptions, so that we could define, say, a generic human figure and a repertoire of "standard" skills. The animator could then instance on or more "customized" human figures appropriate to a particular animated sequence. At the same time, we need to instance the skills defined for the generic figure so that we can customize the behaviors of the instantiated characters as well. Object-oriented programming systems such as Smalltalk [40] and Loops [37] explicitly support the creation and manipulation of classes of objects and their instances. For example, a class of human figures could be specified with a particular structural description and a set of default dimensions specifying limb lengths and rotational constraints at the joints. By altering these defaults the user could instantiate a particular human character, or even create subclasses of special figures, e.g., dwarves or giants. Motor skills also need to have a uniform representation, including preconditions that must be true prior to execution, pointers to other skills to invoke in order to satisfy necessary preconditions, rules for executing under special circumstances, postconditions that will be true after execution, and so on. I have oulined one representation scheme for motor skills in [47]; the development of techniques for representing and manipulating characters and behaviors is an ongoing area of research.

4.6. Text-Mediated and Device-Mediated Interaction

The power of an animation system derives ultimately from its available abstraction mechanisms and the implementation of adaptive movement, not simply be providing the animator with joysticks, knobs and dials. Much has been made of device-mediated interaction in computer graphics, especially in early work (e.g. [3,38]), begun at a time when Fortran or assembly code may have been the only alternative means of human-machine communication. However, language will probably remain the medium of choice for describing algorithms and complicated spatial, temporal, and behavioral relationships. Much of the objection to text-mediated interaction really is an objection to typing. Progress in improving the ergonomics of the typewriter keyboard and ultimately, developments in speech recognition will go a long way towards ameliorating this this aspect of the human-machine interface.

At the same time, there are many functions, e.g., picking, locating, and sketching, for which the graphical gesture clearly is the preferred mode of interaction. Perhaps the ultimate example of graphical interaction is that of flying a simulated airplane, or steering the six-legged walk of a science fiction robot ant with a joystick. But these are large simulation programs built on a complex set of procedures. The user interacts with the top level of a hierarchy of abstractions, and it is this organization that allows small movements of the operator's hand on a joystick to be amplified into a complex of meaningful control signals with such a powerful result.

In the following sections, we will see how three levels of control result from the graded implementation of adaptive motion and abstraction mechanisms.

5. A Three Level Hierarchy for Character Animation

We can classify animation systems as being either guiding, animator level, or task level systems. (For a similar classification of robot programming systems, see [23]).

5.1. Guiding

Guiding systems are those with no mechanisms for user-defined abstraction or adaptive motion. There are a wide range of guiding systems, including motion recording [7,17], shape interpolation [18], key-transformation systems [10,18,45] and notation-based systems [8,43].

In motion recording, various devices are used to acquire kinematic data from a moving figure. The kinematic data is then used to control an animated figure. Such systems are usually limited to measurements of a restricted range of human movement in a laboratory setting, but offer a potentially rich source of data on human motion. Shape-interpolation (also known as "metamorphosis") is the 3-D analog of 2-D keyframing. Where there is a one-to-one correspondnce between the points and faces of separate objects, inbetween frames can be computed by interpolating between the data points of the two objects. In key-transformation systems, whole objects are manipulated by affine transformations. Inbetween frames are generated by interpolating the transformation parameters and transforming the objects. Such systems usually allow the specification of transformation hierarchies, making articulated motion possible. In such key pose systems, e.g., BBOP, a p-curve facility [3] is provided so that the user can graphically specify velocities. Notation-based systems are an example of text-mediated guiding in which the user describes a movement in a choreographic notation or an alphanumeric equivalent (i.e. [7]).

5.1.1. Limitations of Guiding Systems

In guiding systems, the animator must specify in advance the details of motion. This is reasonable only in a relatively featureless environment. Suppose a human character is to walk over rough terrain. Walk cycles are not difficult to generate using keyframing or shape interpolation, but in this case, the walk cycle changes with each step, requiring a large number of intermediate configurations to ensure that the motion looks right. This is because the inbetween frames are computed without regard for other objects in the scene. If a foot goes through the floor, or the figure walks right through a wall, so be it. What is worse, if the character is to walk in another direction over different terrain, none of the earlier key configurations can be used.

In guiding, the animator has nearly complete control over the motion of a figure. Because of the nature of the DOF problem, this is both a blessing and a curse. The animator is free to design an expressive motion sequence in toto, but for complicated figures or intricate mechanisms this is a demanding or perhaps an impossible task, even with a well-designed device-mediated interface [24].

Most guiding systems include predefined procedural abstractions for smoothing motion based on one or several spline techniques [34]. Often these tools allow the animator to interactively adjust the spline parameters until some desired trajectory is achieved. Splining allow the animator to more closely simulate the dynamics of rigid bodies, e.g., acceleration and deceleration due to inertia, friction, or gravity, since motion that is linear and jerky doesn't look right and is often unpleasant to view. (In conventional animation acceleration and deceleration are referred to as <u>ease in</u> and <u>ease out</u> respectively, and must be calculated by hand and from tables). In general, splining provides convenient control over the velocity of many kinds of transformations, including changes in size, shape and color, in addition to changes in positions and orientation. The value of using parameterized curves to control animation was recognized early on [3] and the refinement of these techniques remains an active area of interest (see e.g., [21]). While the use of spline curves is a powerful simulation mechanism, spline techniques alone are not a general solution to the DOF problem, since the control of many transformations requires the generation and refinement of many splines.

To date, a number of interesting animation sequences have been produced using guiding systems at various commercial production houses and university laboratories. However, since powerful abstraction mechanisms are not provided, and because adaptive motion is not possible at all, guiding systems do not scale up well for use with complicated figures, and their utility for controlling animation in complicated environments is limited.

5.2. Animator-Level Systems

A number of animator level systems have been designed to allow the animator to specify motion algorithmically. A few of these systems, while not specifically designed as character animation systems, do provide some measure of one or both adaptive motion and abstraction.

5.2.1. GRAMPS, ASAS, and MIRA

GRAMPS [27] has no facility for adaptive motion, but does allow the construction of motion macros based on functional abstraction. Joints can be grouped together and their input derived from dials, and the motion at the joints can be explicitly constrained to lie within some range of values. This is a good example of the interaction of a guiding mechanism (dials) and a functional abstraction (motion macros). While not designed as a character animation system, GRAMPS has been used to generate interesting animation of a human figure.

Craig Reynolds's ASAS [32] provides a set of low-level mechanisms for both abstraction and adaptive motion. The actor paradigm explicitly provides a general abstraction mechanism allowing the definition of transformation hierarchies (structural abstraction) and behaviors (procedural and functional abstractions). The message passing mechanism makes it possible to implement adaptive motion, since animated entities can report aspects of their physical attributes or their internal states.

Another animation system, MIRA [25], is based on a programming paradigm closely related to actor-based systems, namely, the data abstraction [36] MIRA provides a set of important abstraction facilities nearly identical to those of ASAS. While MIRA is not a message-passing system, the animator can set and examine the values of variables (of various data types), so that attributes of figures and objects can be used to influence the generation of movement.

5.2.2. TEMPUS

The group led by Norman Badler at the University of Pennsylvania has long been involved in research on representing and portraying human movement. They have developed TEMPUS [2,22], a system for analyzing and displaying the movements of realistic human figures in a workspace. While not a general-purpose animation system, TEMPUS has sophisticated features for defining and modifying human figures, and for resolved motion control.

Because the domain of TEMPUS is restricted, unlike MIRA and ASAS, to positioning and orienting human figures, TEMPUS can be largely device-mediated. Users pick actions from a graphically displayed menu, and control motions using displays of simulated potentiometers. Available movements are rotation and translation of the whole figure, rotations at selected joints, and resolved motion of the limbs.

TEMPUS has no facilities for adaptive motion, and abstraction mechanisms available to the animator are limited to a parameterless macro facility which allows the user to group movement commands. The implementation of a flexible resolved motion algorithm for positioning the limbs of a human figure is an important step towards task-level animation.

5.2.3. Discussion

Because it is possible to implement adaptive motion, and to define structural, functional and procedural abstractions, animator level systems provide significant improvements over guiding in terms of the DOF problem. But as usual, there is a trade-off. Guiding systems are relatively easy to learn and use, but lack the power to control complicated animation. Animator level systems, on the other hand, provide the computational power of a general programming language but at the same time saddle the user with all the problems so closely associated with software development. Thalman et al. [25] note that "it took 14 months to produce [a] 13-minute film," certainly highlighting the problem. That is, while it is possible to develop complex motion in either ASAS or MIRA, it is not necessarily easy, since neither language provides explicit, high level support for developing functional abstractions or adaptive motion. Interestingly, Thalman et al. note that they found it necessary to integrate a guiding system, MUTAN [14], into their production scheme. I will have more to say about integrating control modes later on.

5.3. Task Level Animation

At the task level, the animation system must schedule the execution of motor programs to control characters, and the motor programs themselves must generate the necessary pose vectors. To do this, a knowledge base of objects in the environment is necessary, containing information about their position, physical attributes, and functionality.

In [47] I outline one approach to task level animation in which motor behavior is generated by traversing a hierarchy of skills (represented as frames [26] or actors [19] in an object-oriented system) selected by rules which map the current action and context onto the next desired action. Albus of the Bureau of Standards has designed a robot control system based on a hierarchy of table-driven computing elements [1]. Powers has outlined a behavioral control hierarchy based entirely on servomechanism theory [31]. Both of these latter approaches seem to work well at the lower levels of motor control and what we might call instinct-driven behavior, but seem rather vague when it comes to behavior requiring symbolic interaction with the environment.

Task level motor control is a difficult problem under study by cognitive scientists, roboticists, and of course those interested in high level animation systems. In the near term we can expect the development of prototype systems capable of generating rather simple behaviors. How well such systems scale up depends on our understanding of the motor control problem itself.

With task level control, the animator can only specify the broad outlines of a particular movement and the animation system fills in the details. Whether this approach is appropriate depends on the particular application. A non-expert user may be satisfied with the 'default' movements and figures the system provides if he or she can produce, in a reasonable amount of time and at a reasonable cost, and animation that gets the point across. A 'high-end' user, say, in the entertainment industry may want nearly total control over every nuance of a character's movenent to make a sequence as expressive as possible. However, control over the expressive qualities of movement does not mean that the animator needs or wants a pure guiding system to generate pose vectors. The animator does need access to different levels of the control hierarchy in order to generate new motor skills and to 'tweak' the existing skills.

6. Integration of Control Modes

Guiding is the prevalent mode in most current interactive animation systems. The necessity for integrating all three modes of control stems from the inability of any one mode to provide complete yet economical control. Guiding is best suited for specifying fine details but unsuited for controlling complex motion. Animator level programming is powerful but difficult. Task level systems give us facile control over complex motions by trading off explicit control over the details of motion.

Part of the solutin lies in applying guiding techniques at appropriate points in the motion control hierarchy. The key is the ability to decompose the movement repertoire into a manageable set of hierarchically organized skills. The notion of browsers, as implemented in Smalltalk [40] or Loops [37] suggests a powerful method for attaching guiding controls to motor skills. Suppose I have on my RGB monitor a shaded display of a human character. On my terminal screen is a representation of the structure of the character and its skills. Now suppose I trace a curve on the graphics tablet. If I specify that that curve represents a particular joint rotation, -- i.e., I point to the node for the little finger on my terminal, I should immediately see on the display the little finger of my character wiggling. Suppose now I point to the node for "grasping with the left hand" -- I should see the figure's left hand open and close with the velocity I have specified. Lastly, if I pick the node labeled "walk", the figure should begin to walk across the screen, and this time, the curve I have drawn could determine, say, the speed of the gait.

This modular, hierarchical organization allows the user to identify the motion qualities that need to be adjusted, and at the same time it helps to localize the effect of such changes. This calls for a uniform representation of motor skills that incorporates, for each skill, a specification of the kinds of adjustments that are possible, and, in addition, a uniform set of mechanisms, e.g., p-curves for interacting with skills.

7. Conclusion

I have presented a conceptual analysis of the domain of three-dimensional computer animation, which is viewed as the process to simulating objects and their behaviors in a microworld specified by the animator. The degrees of freedom problem is the central issue in the coordination of articulated figures. Computer animation systems must be based on the appropriate set of domain concepts, namely adaptive motion and the five abstraction mechanisms, to enable the animator to define and manipulate interesting characters and environments in an expressive way.

The discussion of the three control modes suggest criteria for good guiding and animator level systems. Guiding systems have received the greatest attention to date -- the notion of an interactive, device-mediated interface has come to be viewed almost as a standard way of communicating with computers, as evidenced by the popularity of the "mouse-and-window" style of computing. In general, however, guiding should be seen as a mechanism for developing and controlling the behavior of complex systems, rather thatn just picking points, drawing lines, or generating scalar values for various transformation parameters. As suggested above, we want to be able to attach the output of a physical input device at arbitrary levels of a behavioral hierarchy. While the meaning of a gesture depends, of course, on the process that is viewing it, the hard question is to find an appropriate set of parameters for controlling a complex process, for example, facial expressions (cf. Parke [28] and Platt [30]). Once a natural control set has been determined, it is not hard to use input devices to generate parameter values interactively. There are two complementary design themes: How can we "plug in" guiding mechanisms to drive a given complex behavior? How can input device modules serve as standard "gesture amplifiers" that can be easily redirected to various functions of the figure control hierarchy?

An animator level language should incorporate the design features and principles we expect in a quality programming language. Concealing the programming task from the user or sugar-coating the syntax is not nearly as important as providing the expressive power needed for animation. This is not the place for a discussion of the future of automatic programming, nor of the merits of the latest programming paradigm. The point is that the algorithmic description of behavior -- "Do this, then do that" -- is an essential and fundamental way to communicate about movement. More often than not the so-called "naive user" will quickly learn the syntax of an animation language only to become frustrated because the language is not powerful enough. Animation level languages and systems should therefore combine what we know about software technology with the mechanisms appropriate to motion control -- e.g., functional abstraction and adaptive motion.

Finally, adaptive motion, in the form of collision testing, and resolved motion control should be implemented, at least in part, as basic elements of any 3-D computer animation system.

The art and science of 3-D computer animation continues to evolve towards the simulation of hypothetical worlds complete with physical laws and figures possessing behavioral repertoires. It is by learning to construct and control these simulations that we give computer animation its expressive power.

Acknowledgements

This work was supported in part by the National Science Foundation Grant, No. MCS 8304185.

References

1. J. S. Albus, Brains, Behavior and Robotics, Byte Books, Peterborough, NH (1981).

2. N. I. Badler, "Design of a Human Movement Representation Incorporating Dynamics," Course Notes, Seminar on Three-Dimensional Computer Animation, (July 27, 1982). ACM SIGGRAPH 82

3. R. M. Baecker, "Picture-driven Animation," Proc. AFIPS Spring Joint Computer Conference, Volume 34, pp.273-288 (Spring 1969).

4. N. Bernstein, The Coordination and Regulation of Movements, Pergamon Press, Oxford (1967).

5. J. F. Blinn, "Systems Aspects of Computer Image Synthesis," Course Notes, Seminar on Three Dimensional Computer Animation, (July 1982). ACM SIGGRAPH 82

6. N. Burtnyk and M. Wein, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation," Communications of the ACM, Vol. 19(10) (October 1976).

7. T. W. Calvert, J. Chapman, and A. Patla, "The Integration of Subjective and Objective Data in the Animation of Human Movement," Computer Graphics, Vol. 14(3) pp.198-203 (July 1980). Proc. ACM SIGGRAPH 80

8. T. W. Calvert, J. Chapman, and A. Patla, "Aspects of The Kinematic Simulation of Human Movement," IEEE Computer Graphics and Aplications, Vol. 2(9) pp. 41-50 (November 1982).

9. E. Catmull, "The Problems of Computer-Assisted Animation," Computer Graphics, Vol. 12(3)(July 1978). Proc. ACM SIGGRAPH 78

10. R. Chuang and G. Entis, "3-D Shaded Computer Animation -- Step-by-Step," IEEE - Computer Graphics and Applications, Vol. 3(9) pp. 18-25 (Dec 1983).

11. J. H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms," Communications of the ACM, Vol. 19(10) pp. 547-554 (October 1976).

12. F. C. Crow, "A More Flexible Image Generation Environment," Computer Graphics, Vol. 16(3) pp. 9-18 (July 1982). Proc. ACM SIGGRAPH 82

13. J. Denavit and R. B. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," Journal of Applied Mechanics, Vol. 23 pp. 215-221 (June 1955).

14. D. Fortin, J. Lamy, and D. Thalman, "A Multiple Track Animator System for Motion Synchronization,"Proc.ACM SIGGRAPH/SIGART Workshop on Motion, pp. 180-186 (April 1983).

15. W. R. Franklin, "3-D Geometric Databases Using Hierarchies of Inscribing Boxes," Proc. Conf. Canadian Society for Man-Machine Interaction, pp. 173-180 (June 1981).

16. H. Fuchs, Z. Kedem, and B. Naylor, "On Visible Surface Generation by A Priori Tree Structures," Computer Graphics, Vol. 14(3) pp. 124-133 (July 1980). Proc. ACM SIGGRAPH 80

17. C. Ginsberg and D. Maxwell, "Graphical Marionette," Proc. ACM SIGGRAPH/SIGART Workshop on Motion, pp. 172-179 (April 1983).

18. J. E. Gomez, "Twixt: A 3-D Animation System," Proc. Eurographics '84, North-Holland, (September 1984).

19. C. Hewitt, "Control Structure as Patterns of Message Passing," pp. 433-465 in Artificial Intelligence: an MIT Perspective, ed. R. H. Brown, MIT Press, Cambridge, MA (1979).

20. C. Klein and C. Huang, "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13(3) pp. 245-250 (March 1983).

21. D. H. U. Kochanek and R. H. Bartels, "Interpolating Splines with Local Tension, Continuity, and Bias Control," Computer Graphics, Vol. 18(3) pp. 33-41 (July 1984). Proc. ACM SIGGRAPH 84

22. J. Korein, J. Korein, G. Radack, and N. Badler, "TEMPUS User Manual," Unpublished, Dept. of Computer and Information Science, University of Pennsylvania, Philadelphia, PA (September 1983).

23. T. Lozano-Perez, "Robot Programming," AI Memo 698, MIT, Cambridge, MA (December 1982).

24. D. Lundin, "3-D Modeling, A Personal Orthodoxy," Course Notes, Seminar on Three-Dimensional Computer Animation, (July 27, 1982). ACM SIGGRAPH 82

25. N. Magnenat-Thalman and D. Thalman, "The Use of High-Level 3-D Graphical Types in the MIRA Animation System," IEEE Computer Graphics and Applications, Vol. 3(9) pp. 9-16 (Dec 1983).

26. M. Minsky, "A Framework for Representing Knowledge," in The Psychology of Computer Vision, ed. P. Winston, McGraw-Hill, New York (1975).

27. T. J. O'Donnel and Arthur J. Olson, "GRAMPS -- A Graphics Language Interpreter for Real-Time, Interactive, Three-Dimensional Picture Editing and Animation," Computer Graphics, Vol. 15(3)(August 1981). Proc. ACM SIGGRAPH 81

28. F. I. Parke, "Parameterized Models for Facial Animation," IEEE Computer Graphics and Applications, Vol. 2(9) pp. 61-68 (November 1982).

29. R. Paul, Robot Manipulators: Mathematics, Programming, and Control, MIT Press (1981).

30. S. M. Platt and N. I. Badler, "Animating Facial Expressions," Computer Graphics, Vol. 15(3) pp. 245-252 (August 1981). Proc. ACM SIGGRAPH 81

31. W. T. Powers, Behavior: The Control of Perception, Aldine Publishing Co., Chicago (1973)

32. C. W. Reynolds, "Computer Animation with Scripts and Actors," Computer Graphics, Vol. 16(3) pp. 289-296 (July 1982). Proc. ACM SIGGRAPH 81

33. E. A. Ribble, "Synthesis of Human Skeletal Motion and the Design of a Special-Purpose Processor for Real-Time Animation of Human and Animal Figure Motion," M.S. Thesis, The Ohio State University (June 1982).

34. D. F. Rogers and J. A. Adams, Mathematical Elements for Computer Graphics, McGraw-Hill, New York (1976)

35. S. Rubin and T. Whitted, "A 3-Dimensional Representation for Fast Rendering of Complex Scenes," Computer Graphics, Vol. 14(3) pp. 110-116 (July 1980). Proc. ACM SIGGRAPH 80

36. M. Shaw, "The Impact of Abstraction Concerns on Modern Programming Language," Proc. of the IEEE, Vol. 68(9) pp. 1119-1130 (September 1980).

37. M. Stefik, D. Bobrow, S. Mittal, and L. Conway, "Knowledge Programming in Loops: Report on an Experimental Course, "AI Magazine, Vol. 4(3) oo, 3-13 (Fall 1983).

38. I. E. Sutherland, "Sketchpad: A Man-Machine Graphical Communication System," Proc. AFIPS Spring Joint Computer Conf., Vol. 23 pp. 329-346 (Spring 1963).

39. R. D. Tennent, Principles of Programming Languages, Prentice-Hall, Englewood Cliffs, NJ (1981).

40. L. Tesler, "The Smalltalk Environment," Byte, Vol. 8(8) pp. 90-147 (August 1981).

41. F. Thomas and O. Johnston, Disney Animation: The Illusion of Life, Abbeville Press, New York (1981).

42. M. T. Turvey, H. L. Fitch, and B. Tuller, "The Problems of Degrees of Freedom and Context-Conditioned Variability," pp. 239-252 in Human Motor Behavior, ed. J.A.S. Kelso, Lawrence Erlbaum Associates, Hillsdale, Jew Jersey (1982).

43. L. Weber, S. W. Smoliar, and N. I. Badler, "An Architecure for the Simulation of Human Movement," Proc. ACM Ann. Conf., pp. 737-745 (1978).

44. D. E. Whitney, "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators," Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control, Vol. 122 pp. 303-309 (December 1972).

45. L. Williams, "BBOP," Course Notes, Seminar on Three-Dimensional Computer Animation, (July 27, 1982). ACM SIGGRAPH 82

46. D. Zeltzer, "Representation of Complex Animated Figures," Proc. Graphics Interface 82, pp. 205-211 (May 1982).

47. D. Zeltzer, "Knowledge-based Animation," Proc. ACM SIGGRAPH/SIGART Workshop on Motion, pp. 187-192 (April 1983).