USING DISCRETE EVENT MODELING FOR EFFECTIVE COMPUTER ANIMATION CONTROL

Paul A. Fishwick
Hanns-Oskar A. Porr

Dept. of Computer & Information Science
University of Florida
Bldg. CSE, Room 301
Gainesville, FL 32611

ABSTRACT

Computer animation is a discipline that has traditional roots within the computer graphics community. Our work shows that discrete event methods within computer simulation can play an important role in helping to organize animations. We discuss an animation of several articulated figures —the dining philosophers scenario— via the control afforded by discrete event modeling and simulation. We have found that multiple models consisting of discrete event and continuous components can provide an easier to understand description of a complex system.

1 INTRODUCTION

With the advent of faster scientific workstations, researchers may now obtain real time animation of simple systems with good graphics rendering for the objects involved. This represents a significant advance for scientists since they now can expect excellent visualization capabilities for physical systems under study.

Within the computer animation and graphics communities, there has been a great deal of interest in *physically based modeling* (Badler, Barsky and Zeltzer 1991; Armstrong and Green 1985) for more realistic motion control of objects. While physically based modeling has improved the realism associated with dynamical system renderings at a physical level, there remains much to be done with respect to controlling, not only low level motion, but also the high-level interactions associated with complex systems. To take an example, consider the internal operation of a bank teller services a line of customers. Physical methods can be used to control the motions of the customers and the teller; however, it is useful to have a higher level model that reflects the dynamic global characteristics of the bank system. A queuing network model or Petri net model can easily serve in this capacity. For instance, by linking together a queuing network model with physical object models, we may simulate and animate the system at more than than the lowest abstraction level. Each model level serves to describe the system at some given level of abstraction.

Badler et al. (1985; 1987) have produced a system that integrates AI, simulation and animation concepts. Tasks are specified in natural language (Gangel 1984; Esakov and Badler 1991; Kalita 1990) and are used to construct a model for simulation and animation. Our work most closely resembles Badler's approach since we are interested in a hierarchical methodology for computer animation — from natural language task description down to video frames. Our current concentration is focused on the use of multiple mathematical models to drive the animation of complex systems. We have found that a flexible and comprehensive *multi-model* (Fishwick 1988; Fishwick 1991a; Fishwick 1991b) representation is necessary to control the animation of systems with articulated objects in a detailed environment.

2 DISCRETE EVENT SIMULATION

Table 1 displays a wide variety of simulation modeling types available for experimentation. As can be seen from table 1, discrete event models are dynamical models whose state variables take on a discrete number of values. While time is continuous in discrete event models, there are also a discrete number of time changes corresponding to the state changes. Discrete event modeling is useful for representing a system's dynamics at a fairly high abstraction level, and therefore complex systems can be efficiently represented as a network or hierarchy of discrete event and continuous models (Fishwick 1991a; Fishwick 1991b).

Key components of systems are well defined in the systems literature (Singh 1987) and include *state*,

Table 1: Simulation Model Types

	Discrete Space	Continuous Space
Discrete Time	*Discrete Time*	*Discrete Time*
	Difference Equations (with integer states) Cellular Automata Finite State Automata	Difference Equations (with real states)
Continuous Time	*Discrete Event*	*Continuous*
	Queuing Models Digital Logic Models	Differential Equations

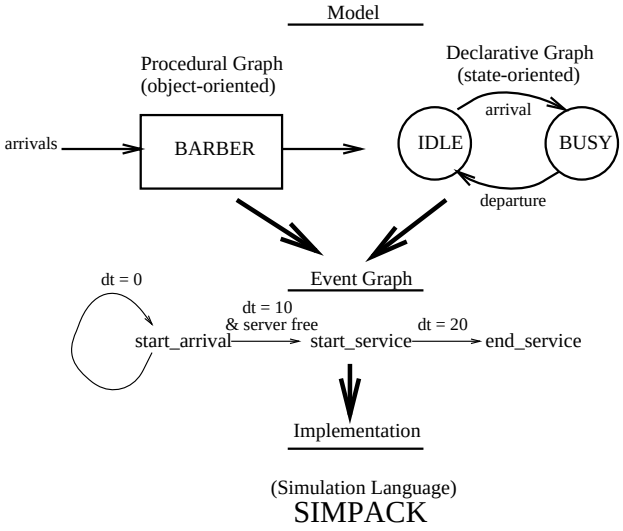

Figure 1: Different Levels of Barbershop Modeling

event and *time*. The *state* of the system represents the condition of the system objects and varies over *time*. *Events* are specific points in time where the state changes. Often, events have cognitive associations; that is, certain state/time pairs denote specific actions that have natural language equivalents. For instance, the familiar barbershop model contains entities (people) that move through the system (shop) using two events: "arrival" and "departure." Figure 1 displays the process of modeling and simulating a barbershop. In fig. 1, The input to a simulation system such as SimPack (Fishwick 1990) is the graph model specification. Models such as the barber shop model are constructed from queuing networks, and queuing networks are a form of procedural, data flow modeling where the barber (server) accepts entities (data) that flow through the system. Declarative, state-oriented modeling is also possible by emphasizing the current state of the barber or entities. The top two models in fig. 1 display these two alternative modeling strategies: data flow vs. state transition.

Given that events demarcate changes in state, we may form an event graph (fig. 1) that displays the event precedence in the barbershop. As people arrive into the stop every ten time units (i.e., $dt = 10$), they either (1) begin servicing immediately (if the barber is not busy), or (2) wait in a first-come first-serve queue until the barber is free. Servicing takes 20 time units.

3 COMPUTER ANIMATION

A computer animation system enables a user to change a graphical scene from one instance of time to the next. The computer then renders (i.e., paints) the scene and displays all changes in rapid succession (30 frames/second for video), thereby creating the illusion of motion. The problem of specifying these "changes" to a computer is called "motion control." There exist a number of different ways of achieving control — all of which are capable of creating the same complex animation. However, some methods require less user interaction than others and are, therefore, more powerful.

The first animation systems were scripted (also called Actor/Scriptor animation systems), such as ASAS (Reynolds 1978; Reynolds 1982) or CINEMIRA (Thalmann and Magnenat-Thalmann 1985). Although different in implementation, they are all basically graphical programming languages that encapsulate the following format: "From t_1 to t_2 do **action** with **object**."

The next type of animation system to be developed was the "keyframe" system, as described in (Burtnyk and Wein 1971; Kochaneck 1982; Gomez 1985). The keyframe technique is an extension of how traditional cell animation (White 1988) is done in 2D animation. In cell animation, the most talented artists draw the figures at "key" positions (ie., start and finish),

and other animators fill in the in-between pictures. The same principle holds true in a 3D keyframe animation system, where the user interactively positions the figure at key times in the system, and the computer calculates all in-betweens. Usually this is done by interpolating between the configurations with a spline curve algorithm. The most popular splines in computer animation are the Catmul-Rom (1974), the Cardinal (Smith 1983), and the Hermite Spline (Doris and Kochanek 1984) because, unlike other types, they actually pass through the control points. At the present time, keyframe systems are the most widely used type of animation system.

Since the mid-80s there has been a trend in computer animation to incorporate physical based modeling –as it has been used in robotics and other system based disciplines– into the animation to reduce the burden on the animator. The objects are given mass and inertia, then forces and torques are applied to achieve a certain motion, which usually looks very realistic. While these systems certainly represent the future, they are still in their experimental stages. It is often awkward for a human to specify the animation in terms of physical parameters (Witkin and Kass 1988; Armstrong and Green 1985). In other instances, a system may have been designed only for a special case (Bruderlein 1988). Thus, there still remains more research to be done in generating an easy-to-use, general animation system.

4 INTEGRATING SIMULATION AND ANIMATION

Figure 2 displays the order in which we proceed in integrating our computer simulation with the traditional keyframe animation system.

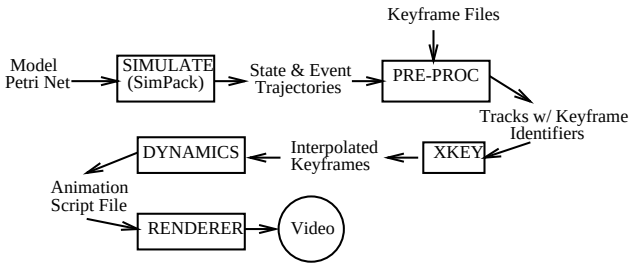

Figure 2: Simulation to Animation Pipeline

4.1 Modeling

First we begin with the simulator (stage 1). We use a set of tools called *SimPack*. SimPack (Fishwick 1990) permits the following types of modeling:

- Finite state automaton with timed states.
- Markov chain modeling.
- Queuing networks.
- Differential, difference equation and delay differential equation modeling.
- Pulse processes.
- Stochastic Petri networks.
- Bi-directional message passing networks.
- Parallel network simulation with the Linda parallel computation model (Gelernter 1985).

For our example, we chose the Petri net modeler. Petri nets can be thought of as a hybrid between procedural and declarative modeling as depicted in fig. 1. In our simulation studies, we have created a two level timed Petri net to model the behavior of 5 articulated figures which comprise the dining philosopher's (**DP**) scenario.

A Petri net model of **DP** is described by Peterson (1981). This model represents the concurrency of "eating" and the resource dependencies for each philosopher. Note that the places and transitions are labeled counter-clockwise (using concentric passes) starting with p_0 and t_0 respectively. We define **DP** as a 4-tuple containing places (P), transitions (T), inputs (I) and outputs (O) as follows:

1. $S =< P, T, I, O >$
2. $P = \{p_0, \ldots, p_{24}\}$
3. $T = \{t_0, \ldots, t_{19}\}$
4. $I : T \rightarrow P^{\infty}$
5. $O : T \rightarrow P^{\infty}$
6. $\mu : P \rightarrow Z_0^+, \mu(p_i) = 1 \; for \; i \in \{0, \ldots, 9\} \; else \; \mu(p_i) = 0$
7. $I(t_0) = \{p_0, p_1, p_2\}, I(t_1) = \{p_2, p_3, p_4\}$
8. $I(t_2) = \{p_4, p_5, p_6\}, I(t_3) = \{p_6, p_7, p_8\}$
9. $I(t_4) = \{p_8, p_9, p_0\}$
10. $I(t_5) = \{p_{10}\}$
11. $I(t_6) = \{p_{11}\}, I(t_7) = \{p_{12}\}$
12. $I(t_8) = \{p_{13}\}, I(t_9) = \{p_{14}\}$
13. $I(t_{10}) = \{p_{15}\}$
14. $I(t_{11}) = \{p_{16}\}, I(t_{12}) = \{p_{17}\}$
15. $I(t_{13}) = \{p_{18}\}, I(t_{14}) = \{p_{19}\}$
16. $I(t_{15}) = \{p_{20}\}$
17. $I(t_{16}) = \{p_{21}\}, I(t_{17}) = \{p_{22}\}$
18. $I(t_{18}) = \{p_{23}\}, I(t_{19}) = \{p_{24}\}$
19. $O(t_0) = \{p_{10}\}, O(t_1) = \{p_{11}\}$
20. $O(t_2) = \{p_{12}\}, O(t_3) = \{p_{13}\}$
21. $O(t_4) = \{p_{14}\}$
22. $O(t_5) = \{p_{15}\}, O(t_6) = \{p_{16}\}$

23. $O(t_7) = \{p_{17}\}, O(t_8) = \{p_{18}\}$
24. $O(t_9) = \{p_{19}\}$
25. $O(t_{10}) = \{p_{20}\}, O(t_{11}) = \{p_{21}\}$
26. $O(t_{12}) = \{p_{22}\}, O(t_{13}) = \{p_{23}\}$
27. $O(t_{14}) = \{p_{24}\}$
28. $O(t_{15}) = \{p_0, p_1, p_2\}$
29. $O(t_{16}) = \{p_2, p_3, p_4\}, O(t_{17}) = \{p_4, p_5, p_6\}$
30. $O(t_{18}) = \{p_6, p_7, p_8\}, O(t_{19}) = \{p_8, p_9, p_0\}$

Two Petri nets are shown in figures 3 and 4. Fig. 3 displays the higher level net while fig. 4 displays just the "eating" sub-net that replaces the shaded place in fig. 3.

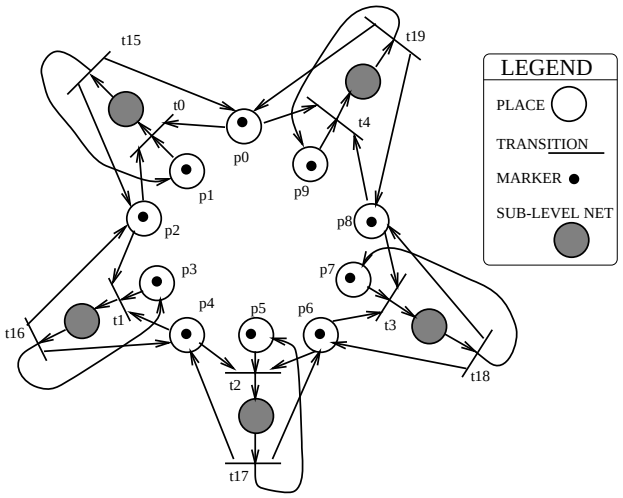

Figure 3: Level 1: Five Synchronized Figures

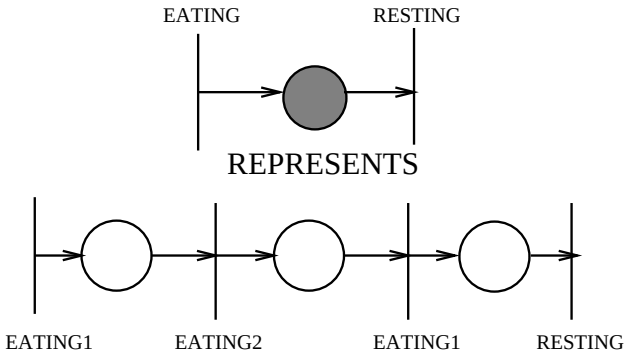

Figure 4: Level 2: The Eating Process Sub-Network

Figures 3 and 4 can be compressed into one net, although it is useful to represent two levels of aggregation as shown. This model represents the well known "dining philosophers" *problem* in operating systems literature, although we are using it as a basis for modeling, and not to solve analytic problems in concurrency. Each philosopher requires two forks to eat (no two adjacent philosophers, therefore, may eat simultaneously). The tokens within the Petri net represent the fork resource and general flow of control. Initially all tokens (or "markers") start out in the center of of the net in fig. 3. Then, two philosophers grab their respective fork pairs the proceed to eat at some rate defined by the modeler.

The total action of any single philosopher is a period of eating followed by a period of rest. The *eating process* is shown in the first three transitions and places of fig. 4. If we consider the subprocess for the first philosopher (starting with t_0) then the transitions fired while eating for level 2 (ref. fig. 4) are: t_0, t_5 and t_{10}. The eating process is described by three actions:

1. EATING1: The right hand is half way between the table and mouth. See figure 6. Example: transition t_0.
2. EATING2: The right hand is at the mouth. See figure 7. Example: transition t_5.
3. EATING1: See fig. 6. Example: transition t_{10}.

The *resting process* (RESTING) is represented by a single net transition (example: transition t_{15}). The resting configuration is shown in figure 5.

4.2 State and Event Trajectories

In fig. 2 we note that the model is input to the appropriate simulator in *SimPack*. Our Petri net model is simulated for some period of time; each transition can be arbitrarily set to some ΔT. The output from the simulation is a single file consisting of a sequence of 3-tuples, as the following example illustrates:

```
[... tuples deleted ...]
  40  man3 RESTING
  45  man1 RESTING
  46  man3 EATING1
  65  man1 EATING1
  74  man3 EATING2
  82  man3 EATING1
  85  man1 EATING2
  89  man1 EATING1
  90  man3 RESTING
[... tuples deleted ...]
```

The meaning of the individual fields in a tuple is as follows:

1. The time when the respective event (node firing) is to take place. An integer time is used here, as it corresponds directly to the frame number in the animation.

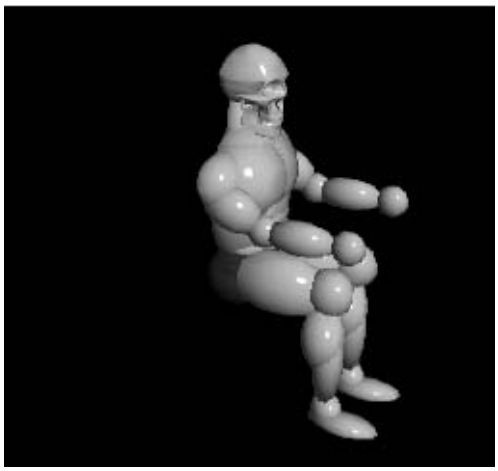

Figure 5: Keyframe # 1 - Resting (RESTING)

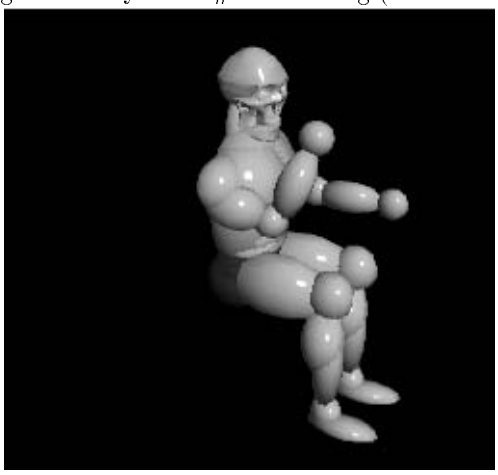

Figure 6: Keyframe # 2 - Eating Stage 1 (EATING1)

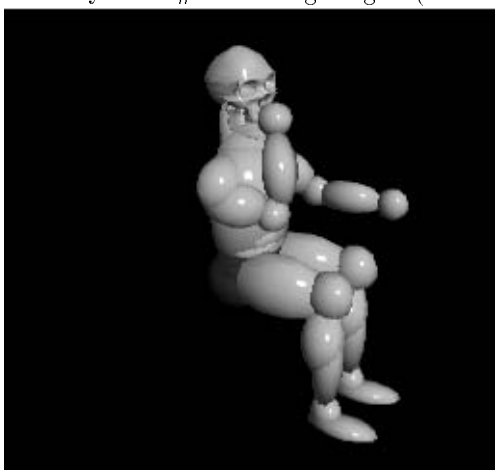

Figure 7: Keyframe # 3 - Eating Stage 2 (EATING2)

2. The animation object associated with the event. In this example, the object is one of the five philosophers (man1 - man5).

3. The event to take place. This event corresponds to a keyframe as it was defined previously in the animation program. For this particular animation, we used three keyframes to represent the eating sequence (EATING1 → EATING2 → EATING1) and one keyframe for the period of rest (RESTING).

Note that we do not have to create keyframes for each of the five philosophers, which would be a total of 20 keyframes. Rather, we define the sequence in the *local* coordinate frame of one generic object. Then, during the simulation we translate the keyframe to the *global* world coordinate of the respective philosopher.

The output from the pre-processor is a file containing the now *absolute* keyframe descriptions for the whole animation as it is needed by the animation program. In the next stage, we use the keyframe animation program to produce the finished animation.

4.3 Geometric Keyframe Modeling

At some point in the modeling process we must create geometric models for the philosophers, and a formal specification for the constraints on the articulated figures. This is done in the XKEY system. XKEY provides an easy interactive environment for specifying an object's low level positions at each keyframe. Specifically, XKEY provides:

- A capability for loading, storing and creating keyframes.
- Denavit-Hartenberg notation (1955) which is used for setting up the articulation linkage and their associated degrees of freedom (DOF).
- An interactive method for moving objects or subobjects with respect to their DOF.
- A multi-track (Gomez 1985) coordinator that includes cut, copy and paste of frames.
- Cubic spline and linear interpolation of keyframe DOF variables.

Figure 8 displays the X-window screen for XKEY.

The models used are polygon based, and were created by using a CSG (Constructive Solid Geometry) approach where spheres are combined to form the limbs. The skull is the same as used by Zeltzer (1982b; 1982a). The renderer supports Phong shading, texture/bump/reflection mapping, metal and spot light effects.

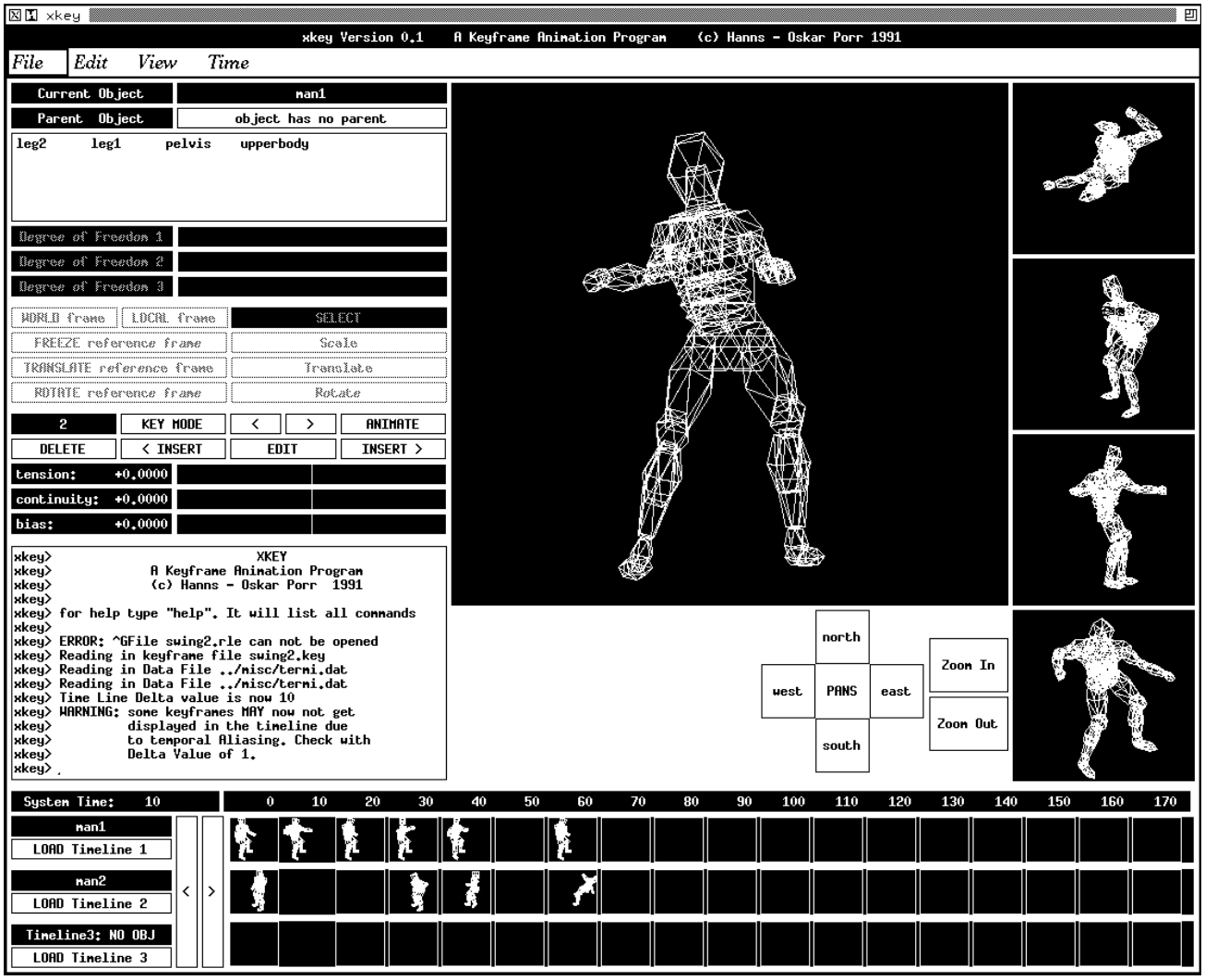

Figure 8: XKEY Interface

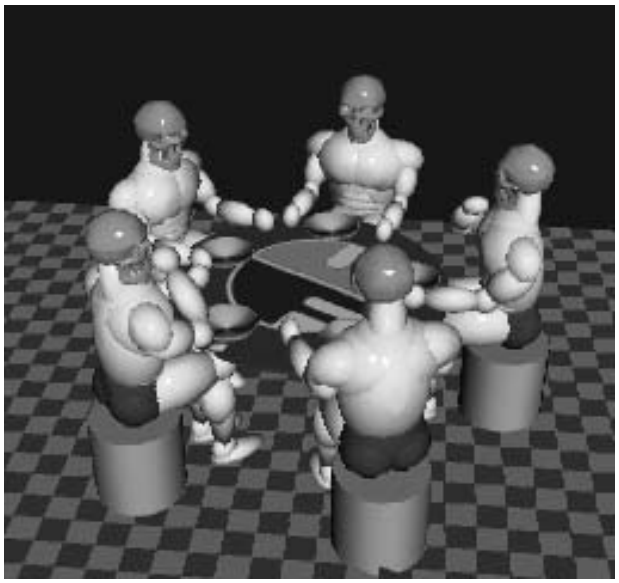

Figure 9: Sample **DP** Video Frame

4.4 Dynamics and Video

Often, the animation produced by keyframing methods is not very realistic looking because it is non-physical in nature. We are currently experimenting with the implementation of dynamic "filters" to enhance the realism. The objects are assigned masses and inertias, and the algorithm then computes accelerations on the spline curve. We are currently investigating the following three methods:

- Constrain the dynamics to follow the spline path exactly (Girard 1991; Wilhelms 1987).
- Constrain the dynamics to follow a path to some ϵ tolerance. This is often achieved through the use of spring-like forces attached to the path(Witkin, Fleischer and Barr 1987).
- Constrain dynamics to pass through or near the keyframe points and use a method such as optimal control to obtain the path achieved by minimizing potential energy.

Obtaining fast physical object responses within an easy-to-use animation system is an active research problem.

The result of our efforts is video footage on a professional 3/4" video recorder where each frame is stored one at a time. Figure 9 displays a sample frame from our **DP** video footage.

5 CONCLUSIONS

We have demonstrated a method for combining discrete event modeling methods with keyframe computer animation. Our focus has been to study existing methods in computer simulation that can be used to aid the graphics community in their search for better mathematical modeling approaches for complicated systems. Such systems often contain discrete as well as continuous components and multiple models must be used to coordinate or control the animation. With the increased amount of physically based modeling research in the graphics field, and the tendency toward more graphical realizations of model execution in the simulation field, we see a need to integrate methodologies from both fields. This research is a small step in that direction.

One immediate extension of the system that we are currently working on is to combine the individual keyframes into a "motion library." Thus, during the simulation, we can refer to the process simply as "eating" instead of having to list all the individual keyframes.

For the future, we plan on building better high-end simulation modeling tools for combined discrete event/continuous models, and better low-end tools allowing the analyst to specify kinematic and dynamic constraints of the systems under study.

REFERENCES

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

Badler, N. I., Barsky, B. A., and Zeltzer, D. 1991. *Making Them Move: Mechanics, Control and Animation of Articulated Figures.* Morgan Kaufmann, San Mateo, CA.

Badler, N. I., Korein, J., Korein, J., Radack, G., and Brotman, L. 1985. Positioning and Animating Human Figures in a Task-Oriented Environment. *The Visual Computer*, 4(1):212 – 220.

Badler, N. I., Manoochehri, K., and Walters, G. 1987. Articulated Figure Positioning by Multiple Constraints. *IEEE Computer Graphics and Applications*, 7(6).

Bruderlein, A. 1988. Goal-Directed, Dynamic Animation of Bipedal Locomotion. Master's thesis, Simon Fraser University.

Burtnyk, N. and Wein, W. 1971. Computer Generated Key Frame Animation. *Journal of the SMPTE 80*, pages 149 – 143.

Catmul, E. and Rom, R. 1974. *A Class of Local Interpolating Spline.* Academic Press, San Franciso.

Denavit, J. and Hartenberg, R. S. 1955. A Kinematic Notation for Lower Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics*, 22:215 – 221.

Doris, H. and Kochanek, U. 1984. Interpolating Splines with Local Tension, Continuity, and Bias Control. In *SIGGRAPH '84, Computer Graphics*, volume 18, pages 33 – 41.

Esakov, J. and Badler, N. I. 1991. An Architecture for High Level Human Task Animation Control. In Fishwick, P. and Modjeski, R., editors, *Knowledge Based Simulation: Methodology and Application*, pages 162 – 199. Springer Verlag.

Fishwick, Paul A. and Zeigler, B. P. 1991a. A Multimodel Methodology for Qualitative Model Engineering. *ACM Transactions on Modeling and Computer Simulation.* (submitted for review).

Fishwick, P. A. 1988. The Role of Process Abstraction in Simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):18 – 39.

Fishwick, P. A. 1990. Computer Simulation Modeling and Analysis: Methodology and Algorithms. unpublished simulation course notes.

Fishwick, P. A. 1991b. Heterogeneous Decomposition and Inter-Level Coupling for Combined Modeling. In *Winter Simulation Conference.* To be presented.

Gangel, J. 1984. A Motion Verb Interface to a Task Animation System. Master's thesis, University of Pennsylvania.

Gelernter, D. H. 1985. Generative Communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80 – 112.

Girard, M. 1991. Constrained Optimization of Articulated Animal Movement in Computer Animation. In Badler, N. I., Barsky, B. A., and Zeltzer, D., editors, *Making Them Move: Mechanics, Control and Animation of Articulated Figures.* Morgan Kaufmann.

Gomez, J. 1985. TWIXT: A 3-D Animation System. *Computer & Graphics*, 9(3):291 – 298.

Kalita, J. 1990. *Analysis of a Class of Action Verbs and Synthesis of Underlying Tasks in an Animation Environment.* PhD thesis, University of Pennsylvania.

Kochaneck, D. 1982. A Computer System for Smooth Keyframe Animation. Master's thesis, University of Waterloo. Technical Report CS-82-42.

Peterson, J. L. 1981. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Inc., Englewood Cliffs, N.J.

Reynolds, C. 1978. Computer Animation in the World of Actors and Scripts. Master's thesis, MIT.

Reynolds, C. 1982. Computer Animation with Scripts and Actors. In *SIGGRAPH '82, Computer Graphics*, volume 16, pages 289 – 296.

Singh, M. 1987. *Systems and Control Encyclopedia: Theory, Technology, and Applications.* Pergammon Press. (8 Volume set).

Smith, A. 1983. *Spline Tutorial Notes - Technical Memo No. 77.* ACM Press.

Thalmann, D. and Magnenat-Thalmann, N. 1985. *Computer Animation.* Springer Verlag.

White, T. 1988. *The Animator's Workbook.* Billboard Publications, Inc.

Wilhelms, J. 1987. Using Dynamic Analysis for Realistic Animation of Articulated Bodies. *IEEE Computer Graphics and Applications*, 7:12 – 27.

Witkin, A., Fleischer, K., and Barr, A. 1987. Energy Constraints on Parameterized Models. In *SIGGRAPH '87, Computer Graphics*, volume 24, pages 225 – 232.

Witkin, A. and Kass, M. 1988. Spacetime Constraints. In *SIGGRAPH '88, Computer Graphics*, volume 22, pages 159 – 168.

Zeltzer, D. 1982a. Motor Control Techniques for Figure Animation. *IEEE Computer Graphics and Applications*, 2(9):53 – 59.

Zeltzer, D. 1982b. Representation of Complex Animated Figures. In *Graphics Interface '82*, pages 205 – 211.

AUTHOR BIOGRAPHIES

PAUL A. FISHWICK is an associate professor in the Department of Computer and Information Sciences at the University of Florida. He received the BS in Mathematics from the Pennsylvania State University, MS in Applied Science from the College of William and Mary, and PhD in Computer and Information Science from the University of Pennsylvania in 1986. He also has six years of industrial/government production and research experience working at Newport News Shipbuilding and Dry Dock Co. (doing CAD/CAM parts definition research) and at NASA Langley Research Center (studying engineering data base models for structural engineering). His research

interests are in computer simulation modeling and analysis methods for complex systems. He is a member of IEEE, IEEE Society for Systems, Man and Cybernetics, IEEE Computer Society, The Society for Computer Simulation, ACM and AAAI. Dr. Fishwick was chairman of the IEEE Computer Society technical committee on simulation (TCSIM) for two years (1988-1990) and he is on the editorial boards of several journals including the *ACM Transactions on Modeling and Computer Simulation*, *The Transactions of the Society for Computer Simulation*, *International Journal of Computer Simulation*, and the *Journal of Systems Engineering*.

HANNS-OSKAR A. PORR is a graduate student in the Department of Computer and Information Sciences at the University of Florida where he is working on his Masters Degree. He received the BS in Computer Science from the University of Utah in 1989. Prior to that he worked as a Computer Graphics Operator for the Electronic Graphics Group, Munich, West Germany. His research interests are in Computer Graphics (advanced rendering and texturing algorithms), Computer Animation (high level motion control) and Robotics. Mr. Porr is a native of Trier, West Germany.