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

<table>
<thead>
<tr>
<th>Discrete Space</th>
<th>Continuous Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete Time</strong></td>
<td><strong>Continuous Time</strong></td>
</tr>
<tr>
<td>Difference Equations (with integer states)</td>
<td>Difference Equations (with real states)</td>
</tr>
<tr>
<td>Cellular Automata</td>
<td>Queuing Models</td>
</tr>
<tr>
<td>Finite State Automata</td>
<td>Digital Logic Models</td>
</tr>
<tr>
<td><strong>Continuous Time</strong></td>
<td><strong>Discrete Event</strong></td>
</tr>
<tr>
<td><strong>Continuous</strong></td>
<td><strong>Continuous</strong></td>
</tr>
<tr>
<td><strong>Differential Equations</strong></td>
<td></td>
</tr>
</tbody>
</table>
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.

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 modeling. 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. $P = \{p_0, \ldots, p_4\}$
3. $T = \{t_0, \ldots, t_{19}\}$
4. $I : T \rightarrow P^\infty$
5. $O : T \rightarrow P^\infty$
6. $\mu : P \rightarrow Z_+^\infty \land \mu(p_i) = 1$ 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. \( O(t_{15}) = \{p_{1}, p_{1}, p_{2}\} \)
29. \( O(t_{16}) = \{p_{2}, p_{2}, p_{3}, p_{4}\} \), \( O(t_{17}) = \{p_{4}, p_{5}, p_{6}\} \)
30. \( O(t_{18}) = \{p_{6}, p_{7}, p_{8}\} \), \( O(t_{19}) = \{p_{8}, p_{9}, p_{10}\} \)

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](image)

![Figure 4: Level 2: The Eating Process Sub-Network](image)

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 \( \Delta T \). 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.
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 sub-objects 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
4.4 Dynamics and Video

Often, the animation produced by keyframing methods is not very realistic looking because it is nonphysical 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 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


Effective Computer Animation Control


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

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.