시뮬레이션의 계층적 애니메이션 환경

조대호, 이미라

Hierarchical animation Environment for Simulation

Tae Ho Cho, Mi Ra Yi

< 개 요 >

DEVS(Discrete EVent system Specification)형식론은 계층적이고 모듈화 된 형식으로 시스템을 설계함으로써 신뢰성 있는 모델링이 가능하도록 이론적으로 잘 정립된 시뮬레이션 방법론이다. 시뮬레이션의 진행과정 및 결과를 표현하기 위한 애니메이션 개발 환경에 있어서도 DEVS 모델의 구조를 반영하여 계층적 애니메이션 환경을 구현 할 수 있다. 계층적 애니메이션은 시뮬레이션 관찰자가 시스템의 특정 레벨에 맞추어 애니메이션 진행 상황을 볼 수 있도록 한다. 이는 일반적으로 시뮬레이션 애니메이션이 갖는 장점인 시스템 이해 및 모델의 신뢰성 검증의 향상 뿐 아니라, 다양한 관점에서 시스템의 변화를 확인 할 수 있다는데 그 필요성이 요구된다. 본 논문에서는 계층적인 애니메이션과 그래픽 표현이 가능하도록 하기위해 DEVS 모델의 구조를 반영한 계층성을 갖는 애니메이터(시뮬레이션과 애니메이션의 연결 프로세서)를 설계하였다. 이러한 애니메이터는 모델과 같은 구조로 각 모델에 필요한 애니메이션 객체들을 유지·관리함으로써 계층적 애니메이션을 가능하게 하며, 시뮬레이션의 이산적인 시간 경과와 애니메이션의 연속적인 시간 경과 사이에 동기화를 용이하게 한다.

1. 서론

애니메이션은 컴퓨터에서 그래픽 객체를 움직이는 출력 형태로써 여러 분야에서 많이 사용되고 있으며, 컴퓨터 시뮬레이션에서도 시스템의 상태를 이해하기 쉽도록 표현하기 위해 애니메이션을 이용하고 있다[1].

시뮬레이션이 시스템에 대한 변화를 예측 또는 평가하기 위한 것인 만큼, 그 수행 과정 및 결과를 이해하기 쉽도록 애니메이션으로 표현하는 것은 중요하다. 이러한 이해의 용이성은 시뮬레이션에 익숙하지 않은 시스템 전문가(시뮬레이션 요구자)가 시뮬레이션 모델링과정에 참여할 수 있도록 도와주어 시뮬레이션의 신뢰성을 향상시키는 역할을 하고, 시뮬레이션 개발자측면에서 애니메이션은 시뮬레이션 모델에 대한 검증을 좀더 쉽게 한다[1,7]. 실제로 많은 시뮬레이션 툴 개발자들이 그래픽적인 표현에 관심을 보이고 있으며, 상용으로 개발된 이산사건 시뮬레이션 개발 환경들의 대부분이 애니메이션 출력형태를 갖는다[1,10]. 본 연구는 그 중 DEVS 시뮬레이션 애니메이션에 관한 것이다.

DEVS(Discrete EVent System Specification) 형식론은 연속적인 시간상에서 이산사건을 발생시키는 시스템을 시뮬레이션하기 위해 이론적으로 잘 정립된 모델링 방법론이다. 이는 모델의 구조와 행동을 시뮬레이션 수행으로부터 추상화 시키기 위해 모델을 집합이론적 방법을 이용하여 시스템을 계층적(hierarchical)이고 모듈화(modular) 된 형식으로 기술한다. 이러한 특징은 복잡한 시스템에 대해 보다 신뢰성 있는 모델링을 가능하게 한다[2].

본 논문에서는 DEVS 모델링 방법론을 기반으로 하는 시뮬레이션의 계층적 애니메이션 개발 환경을 설계한 것이다. 다음 2 장에서는 DEVS 시뮬레이션의 특징을 통해 계층적 애니메이션의 필요성을 확인하고, 3 장에서는 계층적 애니메이션을 위한 시뮬레이터와 애니메이션 연결 프로세서인 애니메이터를 설

계하며, 4 장에서는 계층적 애니메이션을 위한 구성요소, 실제 적용 예 및 기대효과를 서술한다.

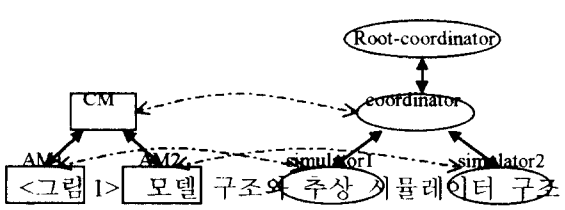
2. DEVS 시뮬레이션과 계층적 애니메이션

2-1. DEVS 모델구조와 시뮬레이터

DEVS 형식론에서 시스템을 기술하기 위해 제안한 모델 형태에는 Atomic 모델, Coupled 모델이 있다. Atomic 모델은 시스템의 최하위 구성 요소들을 표현하기 위한 것이고, Coupled 모델은 시스템의 구성요소 간에 상호작용을 표현하기 위한 것이다. Coupled 모델은 다른(상위레벨) Coupled 모델을 위한 구성 모델로 사용될 수 있다. 따라서 모델링이 완성되면 시스템은 Atomic 모델과 Coupled 모델로 구성된 하나의 Coupled 모델로 표현 될 수 있으며, 시스템의 각 레벨별로 계층성이 있는 모델구조를 갖게 된다[2].

현재 DEVS 모델링 방법을 기반으로 만들어진 시뮬레이션 환경들(DEVS-Scheme, DEVS-C++,....)은 모델을 실행(이벤트 스케줄링)하기 위한 프로세서(시뮬레이터)를 제공한다. 이들은 DEVS 모델의 모듈화 특성에 맞게 객체 지향적인 시뮬레이션이 가능하도록 모델클래스와 프로세서클래스가 정의되어 있지만, 환경에 따라 모델과 프로세서가 하나의 클래스로 묶여있기도 한다[2,9]. 이러한 환경들은 모델을 정의하는데 있어서는 DEVS 이론을 따르므로 차이가 없으나, 프로세서에서 이벤트 스케줄링을 하는데 다소 차이가 있다. 본 논문에서는 프로세서의 모듈화가 잘 되어 있는 시뮬레이션 환경(DEVS-Scheme[2])을 기준으로 용어 및 관련 개념을 설명한다.

<그림 1>은 모델구조와 프로세서(추상 시뮬레이터) 구조를 나타낸 것이다. Atomic 모델,Coupled 모델에 해당하는 프로세서가 각각 simulator, coordinator 이고, 최상위에는 시뮬레이션 시간을 제어하기 위한 root-coordinator 가 있다.

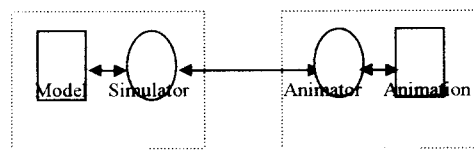


<그림 1> 모델 구조와 추상 시뮬레이터 구조

simulator 와 coordinator 는 모델의 상태변화(transition)이 일어날 때마다 상위 프로세서에게 다음 이벤트가 일어날 시간 정보를 done-메시지에 싣어 올려보낸다. 마지막으로 done-메시지를 받은 Root-coordinator 는 해당 시간만큼 시간을 갱신하고 다음 이벤트를 *-메시지를 보냄으로써 지시하게 된다. simulator 에서 나오는 output 은 상위 coordinator 에 의해 y-메시지에 싣어 전달되고, coordinator 는 이 output 정보를 x-메시지에 싣어 연관된 다른 프로세서에게 전달한다. 즉, y-메시지를 받았을 때 외부상태변화(ext-transition)를, *-메시지를 받았을 때 내부상태변화(int-transition)를 일으킨다.

2-2. 시뮬레이션과 애니메이션 관계

애니메이션으로 표현하는 시뮬레이션 개발 환경에서는 모델링 할 때 시스템의 이벤트 스케줄을 위한 정보와 각 이벤트에 의한 상태변화를 애니메이션하기 위한 정보를 정의해야 한다. 정의된 모델은 이벤트 스케줄링을 위한 프로세서인 시뮬레이터와 애니메이션을 위한 프로세서인 애니메이터를 갖는다.<그림 2>



<그림 2> 시뮬레이션과 애니메이션 관계

제안한 시뮬레이션 애니메이션 환경은 시뮬레이터가 모델구조를 반영하여 계층성을 이루는 것과 같이 계층적인 애니메이터를 매개로 계층적 애니메이션이 가능하다.

2-3. 계층적 애니메이션의 필요성

계층적 애니메이션이란 DEVS 시뮬레이션의 계층성을 애니메이션에 반영한 것을 말한다. 일반적인 애니메이션이 애니메이션 객체들을 일괄적으로 관리하면서 화면에 출력하는 것과 달리, 시뮬레이션 모델마다 각 애니메이션 객체들을 갖고 이들을 제어하는 프로세서(애니메이터)를 갖는다. 이러한 계층성을 애니메이션에 담아내는 것은 다음과 같은 이유로 중요한 의미를 갖는다.

첫째, 시뮬레이션과 애니메이션간의 동기화를 용이하게 한다. 이산적으로 시간이 경과되는 시뮬레이터는 연관된 모델의 상태변화에 따라 애니메이션 지시를 내리고, 시뮬레이터가 다음 이벤트를 처리하기 전에 애니메이터로부터 이전 이벤트에 대한 애니메이션 처리가 끝났음을 알리는 메시지를 받게 함으로써 시뮬레이션과 애니메이션 상호간 이벤트의 동기화를 이룰 수 있다. 이러한 동기화의 대상이 모델과 같은 계층성을 갖는 시뮬레이터이므로 애니메이터 또한 그 계층성을 따르면 동기화를 쉽게 할 수 있다.

둘째, 시스템의 계층별 명세작성을 용이하게 한다. 모델을 애니메이션을 통해 계층별로 확인이 가능하기 때문에, 시뮬레이션을 모델링 할 때 시스템의 각 계층에 맞게 기술 할 수 있도록 도와준다.

셋째, model 을 재구성 할 경우 애니메이션의 재구성이 용이하다. DEVS 형식론에 의한 시뮬레이션은 모듈성이 좋게 모델링을 할 수 있으므로, 만들어 놓은 모델들을 재구성 하기 쉽다. 이러한 장점을 애니메이션에도 적용이 되게 하려면 DEVS 모델구조를 애니메이션에 그대로 반영해야 한다.

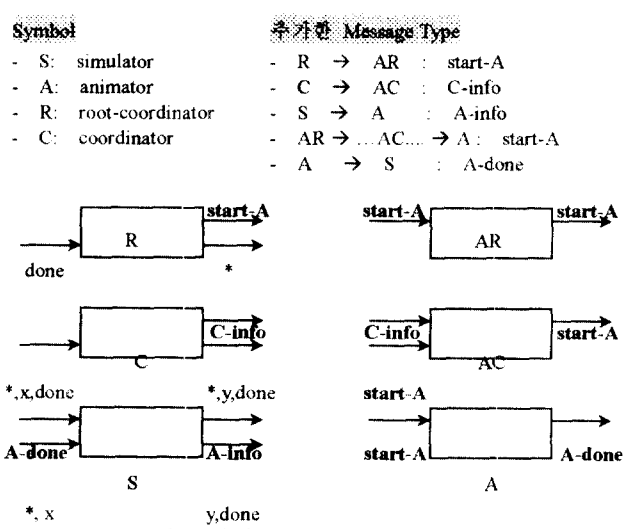
이 외에도 DEVS 시뮬레이션 모델링의 계층적이고

객체 지향적인 설계가 지닌 장점[2,4,7]들이 애니메이션 모델링에도 반영될 수 있다.

3. 계층적 애니메이터 설계

3-1. 애니메이터 스케줄링

앞에서 설명하였듯이 DEVS 모델마다 해당 애니메이터가 생기므로 모델과 같은 구조로 애니메이터가 생성된다. 하지만, 이는 모델이 아닌 시뮬레이터와 상호작용을 하면서 애니메이션 스케줄링을 한다. 그리기 위해서는 기존의 시뮬레이터에도 수정을 가해야 한다. 다음은 그들의 입·출력(그림 3)과 스케줄링을 설명한 것이다.



<그림 3> 시뮬레이터와 애니메이터의 입·출력

R(Root-coordinator)는 C 에게 *-메시지를 보낼 때 AR 에게 start-A-메시지를 보낸다. 이는 시뮬레이션 시간이 갱신된다는 것을 애니메이션에게 알리기 위해 elapsed –time 정보를 메시지에 실어 보낸다.

C(Coordinator)는 상위 프로세서에게 done-메시지를 보낼 때 AC 에게 C-info-메시지를 보낸다. 이는 C 의 수정된 다음 이벤트에 대한 시간 정보를 AC 에게 알리기 위함이다. 또, 하위 프로세서들이 화면에 보일 애니메이션 단계인지에 대한 정보를 전달한다.

S(Simulator)는 *-메시지를 받으면 대기하고 있다가 A-done-메시지를 받은 후 내부 상태변화 처리를 한다. A-done-메시지는 A 로부터 이전 이벤트에 대한 애니메이션이 끝났다는 의미를 갖는다. 즉, 시뮬레이터와 애니메이터 간에 동기화가 이루어지게 된다. 그러나, x-메시지에 의한 외부상태변화 처리는 A-done-메시지를 받지 않고도 처리한다. 또, 상태변화 처리를 반영할 애니메이션이 있으면 애니메이션 객체와 필요한 정보를 A-info 에 실어 A 에게 보낸다.

AR(Animator's Root-coordinator)는 시뮬레이션 상에서 R 이 시뮬레이션 시간의 흐름을 제어하듯, 애니메이션상에서 시간의 흐름을 제어한다. R 로부터 다음 이벤트의 시간 정보를 갖는 start-A-메시지를 AC 에게 보낸다.

AC(Animator's Coordinator)는 상위 프로세서로부터 start- A-메시지를 받으면 애니메이션 동작의 남은 시간 값을 갱신한다. 그리고, C 로부터 C-info-메시지를 통해 받은 화면출력 단계 정보를 확인한 후 하위 프로세서들이 화면에 보일 애니메이션 단계이면 하위 프로세서들에게 start-A-메시지를 전달한다.

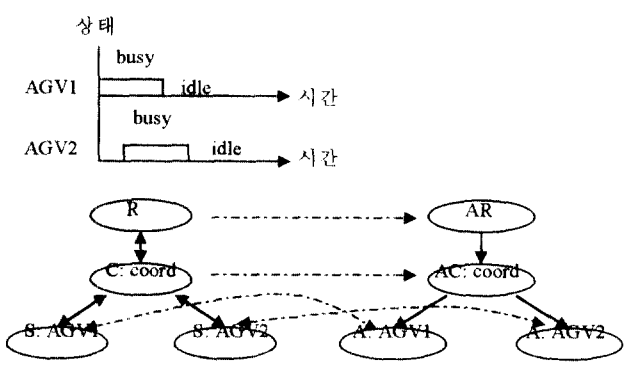
A(Animator)는 S 로부터 A-info-메시지를 통해 받은 애니메이션 정보로 애니메이션 할 준비를 하고 있다가 AR 로부터 start-A-메시지를 받으면 해당 elapsed-time 동안 애니메이션 동작을 수행한다. 동작이 끝난 후 남은 애니메이션 동작 시간이 0 이면 S 에게 A-done-메시지를 보낸다.

A, AC 는 화면 출력에 대한 플래그 정보를 갖고서 해당 계층에서 애니메이션을 화면에 보일 것인지를 판단하여 계층적 애니메이션을 할 수 있도록 한다.

3-2. 시뮬레이터와의 동기화

설명한 애니메이터 스케줄링을 간단한 시스템을 예로 살펴보자.

<그림 4>는 두 개의 AGV(atomic 모델)와 그들을 제어할 하나의 coordinator(coupled 모델)를 갖는 시스템에 대한 시뮬레이터와 애니메이터의 구조를 그래프로 나타낸 것이고, <표 1>은 실제 시뮬레이션에서 한



<그림 4> 시뮬레이터와 애니메이터의 상호작용

사이클이 실행되는 동안의 스케줄링을 나타낸 것이다. AGV1 과 AGV2 는 모두 processing time 이 30 이고, AGV2 는 AGV1 이 작동한 후 시뮬레이션 시간 10 만큼 지난 후 수행된다.

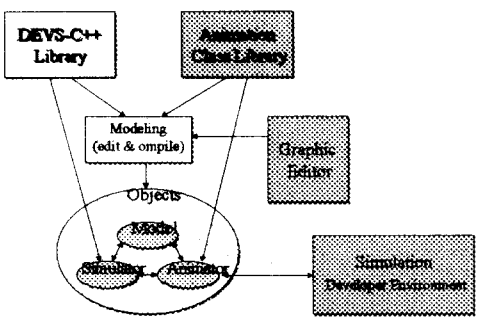
<표 1> 시뮬레이션 스케줄링

Sender	Message	Receiver	
R	* 0	S: AGV1	
R	Start-A 0	AR	
AR	Start-A 0	A: AGV1	
A: AGV1	A-done	S: AGV1	AGV1:화면출력 완료
S: AGV1	A-info	S: AGV1	
S: AGV1	Done 30	C: coord	
C: coord	C-info	AC: coord	

R	* 10	S: AGV2	
R	Start-A 10	AR	
AR	start-A 10	A: AGV1	AGV1:시간 10 동안움직임
A: AGV2	A-done	S: AGV2	AGV2:화면출력 완료
S: AGV2	A-info	A: AGV2	
S: AGV2	done 30	C: coord	
C: coord	C-info	AC: coord	

4. 계층적 시뮬레이션 애니메이션 개발 환경

이 장에서는 계층적 애니메이션 개발 환경을 포함했을 때의 전체적인 시뮬레이션 개발 환경에 대해 살펴본다. <그림 5>는 그의 구성도이다.

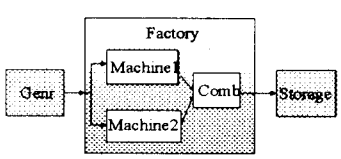


<그림5> 시뮬레이션 개발 환경 구성요소

모델링을 할 때 시뮬레이션과 애니메이션에 관련된 것들은 각각 DEVS-C++ 라이브러리, Animation Class 라이브러리를 이용하여 작성하고, 애니메이션 관련 모델링을 할 때 필요한 그래픽 관련 파일들은 Graphic Editor로 만든다. 이렇게 하여 생성된 시뮬레이션 모델이 컴파일되면, 사용자가 작성한 모델의 정보를 유지하는 프로세서인 Model, 이벤트 스케줄링을 하는 프로세서인 Simulator, 이벤트 스케줄링을 가시화하기 위한 프로세서인 Animator 들이 생성된다. 실제로 시뮬레이션 진행은 이러한 프로세서들 간의 상호작용을 통해 이루어진다.

다음은 계층적 애니메이션을 간단한 시스템을 예로 살펴보겠다. <그림 6>은 두 대의 기계와 이들에서 만들어진 것들을 결합하는 하나의 combiner가 있는 생산 공장 시스템의 구성도 이다. <그림 7>은 최상위(Factory) 계층에서 공장의 입출력을 통해 전체 시스템의 동적특성을 관찰하는 출력 화면 이고, <그림 8>은 공장의 내부구조를 나타내고 그 구성 요소의 동적 특성을 관찰하는 두 번째 계층의 출력 화면이다.

이러한 계층적 애니메이션은 일반적으로 시뮬레이션 애니메이션이 갖는 장점인 시스템 이해 및 모델

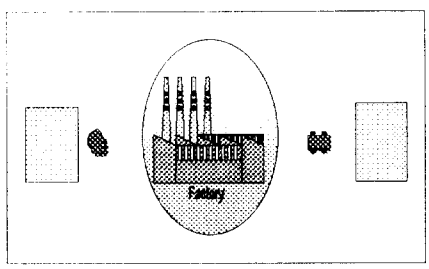


<그림6> 자동차 생산 시스템

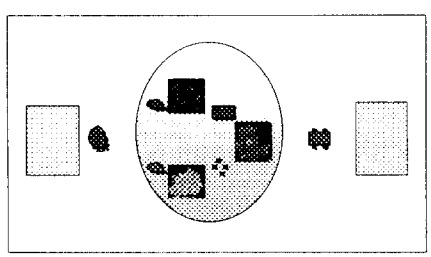
의 신뢰성 검증을 향상시킬 수 있을 뿐 아니라, 다양한 관점에서 시스템의 변화를 확인 할 수 있다. 위의 예에서 볼 수 있듯이 시스템 관찰자는 원하는 계층에서의 시뮬레이션 진행과정을 확인 할 수 있다.

5. 결론

본 논문에서는 시뮬레이션에서 애니메이션이 갖는 의미와 DEVS 모델링 방법을 기반으로 하는 시뮬레이션의 계층적 애니메이션이 갖는 의미를 살펴보고, 이를 위한 계층적 애니메이터와 시뮬레이션 개발환경를 설계하였다. 계층적 애니메이션은 실제 애니메이션의 프로세서인 애니메이터가 모델구조와 같이 계층적으로 돼 있어 모델별로 필요한 애니메이션 객체들을 유지·관리하고, 시뮬레이션의 이산적인 시간 경과와 애니메이션의 연속적인 시간 경과 사이에 동기화를 용이하게 한다.



<그림7> Factory Level 출력화면



<그림8> Machine Level 출력화면

계층적 애니메이션은 시뮬레이션 관찰자가 시스템을 다양한 관점에서 애니메이션 진행 상황을 확인함으로써 시스템 이해 및 모델의 신뢰성 검증을 향상시킬 수 있다.

참고문헌

[1] David RC Hill, Object-Oriented Analysis and Simulation, Addison-Wesley, 1996.

[2] Bernard P. Zeigler, Object-Oriented Simulation with Hierarchical, Modular Models, Academic Press, 1990.

[3] Averill M. Law and W. David Kelton, Simulaiton Modeling and Analysis, Second Edition, McGraw-Hill, 1991.

[4] Bernard P. Zeigler, Object and Systems, Springer-Verlag,1997.

[5] Bernard P. Zeigler, Theory of Modeling and Simulation , John Wiley, 1976, reissued by Krieger, Malabar,1985..
[6] Bernard P. Zeigler, "Multifacetted Modeling and Discrete Event Simulation, Academic, 1984
[7] Lan Sommerville, Software Engineering, Fifth Edition, Addison-Wesley, 1995.
[8] Paul A. Fishwick, Simulation Model Design and Execution,Prentice Hall, 1995.
[9] Hyup J. Cho, DEVS-C++ Reference Guide, 1997.
[10] CACI Company , MODSIM III Manual, 1997.
[11] David J. Kruglinski, Inside Visual C++, Microdoft Press, 1996.
[12] 조대호, 이질기, "계층의 구조를 갖는 시뮬레이션 모델에 있어서 단계적 접근을 위한 모델연결 방법론과 그 적용 예", 한국시뮬레이션학회 논문지, 제 5 권,제 2 호, 1996.

저자소개

조대호

1983 성균관대학교 전자공학과 학사
1987 알라바마대 전자공학과 석사
1993 아리조나대 전자 및 컴퓨터공학 박사
1993~1995 경남대학교 전자계산학과 전임강사
1995~현재 성균관대학교 전기전자 및 컴퓨터공학부 조교수
관심분야: 시뮬레이션 모델링 방법론, 지능형 시스템, 공장 자동화, 인공 지능.

이미라

1998 성균관대학교 정보공학과 학사
현재 성균관대학교 전기전자 및 컴퓨터공학부 석사과정
관심분야: 이산사건 시뮬레이션, 시뮬레이션 개발 환경.