가상제조환경을 위한 형상기구학 모델링 및 시뮬레이션으로의 DEVS 확장

황문호(큐빅테크 기술연구소), 최병규(KAIST 산업공학과), 천상욱(큐빅테크 기술연구소)

Extending the DEVS formalism toward Geometrical Kinematic Modeling and Simulation for Virtual Manufacturing Environment

Moon-Ho Hwang, Byoung-Kyu Choi, Sang-Uk Choen

Abstract

Proposed in this paper is a modeling and simulation methodology for a virtual manufacturing environment. Based on DEVS formalism[Zeigler 76], the proposed model, so called GKDEVS, is designed to descript the geometrical kinematic structure as well as event-driven and continuous state dynamics. In terms of abstract simulation algorithm[Zeigler 84], the simulation method of GKDEVS is proposed for combined discrete-continuous simulation. Using the GKDEVS, and FMS model consisting of a turing machine, a 3-axis machine and a RGV-mounted robot is constructed and simulated.

Keywords: virtual manufacturing, DEVS, combined discrete-continuous simulation

1. 서론

90년대에 들어서 가상현실(Virtual Reality)의 개념을 제조시스템에 도입하여 제조시스템의 설계와 운용에 사용하고자 하는 가상제조시스템(Virtual Manufacturing system)의 개념이 소개되고 있다[Onosato 93][Jones 93]. 이와 관련하여 3차원 설비 모델의 시뮬레이션은 핵심적인 기술 중 하나로 자리 매김 되고있으며[Iwata 95][Choi 99], 애니메이션 분야에서부터 연구가 시작되어왔다[Jones 93][Kim 95]. 그러나 가상 제조시스템을 구성하는 설비의 모델들이 증가하면 증가할수록, 또 설비들이 독립적으로 존재하는 것이 아니라 상호 유기적으로 영향을 줄수록 이를 표현하는 효율적인 모델링 및 시뮬레이션 방법론에 대한 연구가 필요하며, 이를 위해 형상 기구학 모델과 이산사건 모델의 통합 시뮬레이션의 환경에 대한 연구의 필요성이 지적되고 있다[Klingstam 99]. 이에 본 연구에서는 DEVS 형식론[Zeigler 76]과 이의 추상화 시뮬레이션 방법론[Zeigler 84]을 확장하여 3차원 설비의 모델링 및 시뮬레이션을 수행하고자한다.

2. 요구사항

가상 제조시스템 구축을 위하여 시뮬레이션 모델이 지녀야할 요구사항 중 본 논문에서 다루는 것들로는

1) 3차원 공간상의 다관절 구조의 표현
2) 이벤트 발생에 따른 동적 상태변화 표현
3) 연속상태 및 이산사건 혼합 시뮬레이션
4) 사용자 참여(User Interactive) 시뮬레이션

등이다.

3. Geometrical and Kinematic DEVS: GKDEVS

형상과 기구학 정보를 갖고 내부에 회귀적(recursive)인 구조를 갖는 확장된 모델은 다음과 같이 정의된다.

GKDEVS

$= \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta, M, Z, SELECT \rangle$

where

X: 입력 이벤트 집합; Y: 출력 이벤트 집합; S: 상태변수 집합, $S=\langle GK, S^r \rangle$, $GK=\langle G, type_{link}, F, {}^UF, A, v, [v_{\min}, v_{\max}], >$ G: 볼록 다각형 집합; $F=\langle R, P \rangle$; 상위 GK.F의 상대적 프레임, R(3x3 matrix): 상대 회전정보, P ($\in R^3$): 상대 위치 정보; ${}^UF=\langle {}^UR, {}^UP \rangle$: 절대 좌표계상의 프레임, ${}^UR$(3x3 matrix): 절대 회전정보, ${}^UP(\in R^3)$: 절대 위치 정보; $type_{link} \in \{fixed, prismatic, revolute\}$ 조인트(joint) type; $A \in R^3$: 링크의 축벡터; $vin[v_{\min}, v_{\max}]$ $v, v_{\min}, v_{\max} \in R$: 축벡터 상의 값 및 이동범위; S^r: 링크정보를 제외한 자신의 상태변수 집합, w.r.t $GK \bigcap S^r = \{\}$; M: 연결된 링크 GKDEVS의 집합; $\delta_{int}: S \rightarrow S$: 내부 상태천이 함수 ; $\delta_{ext}: Q \times (X \bigcup \{\varnothing, c\}) \rightarrow S$: 외부 상태천이 함수, 단, $Q=\{(s,e) | s \in S, 0 \le e \le ta(s)\}$, 총체적 상태, $\varnothing$: 공(공) 입력 이벤트(non-event); c: 공간상의 충돌이벤트; $\lambda: S \rightarrow Y$, 외부 출력함수; $ta: S \rightarrow R_0^\infty$, 시간전진함수; $Z: Y \rightarrow X$: 출력전달함수; $SELECT$: $2^{M \bigcup \{self\}} - \{\} \rightarrow M \bigcup \{self\}$, 타이 해결함수;

여기서 프레임(F)를 4x4 동차변환행렬(homogeneous transformation matrix)로 표현하면 다음과 같다.

$$T = \begin{cases} \begin{bmatrix} R & | & P \\ 0\ 0\ 0 & | & 1 \end{bmatrix} & \text{if type= fixed} \\ \begin{bmatrix} R & | & P+vA \\ 0\ 0\ 0 & | & 1 \end{bmatrix} & \text{if type= prismatic} \\ \begin{bmatrix} RR_A^v & | & P \\ 0\ 0\ 0 & | & 1 \end{bmatrix} & \text{if type= revolute} \end{cases}$$

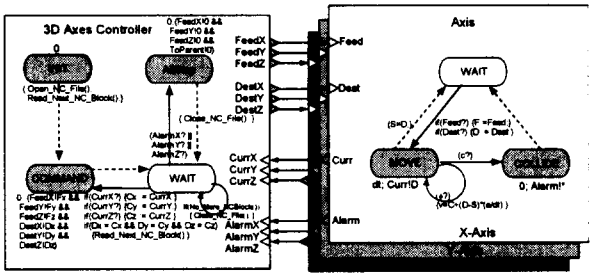
where

$$R_a^v = \begin{bmatrix} a_x a_x kv + cv & a_x a_y kv - a_z sv & a_x a_z kv + a_y sv \\ a_x a_y kv + a_z sv & a_y a_y kv + cv & a_y a_z kv - a_x sv \\ a_x a_z kv - a_y sv & a_y a_z kv + a_z sv & a_z a_z kv - cv \end{bmatrix}$$

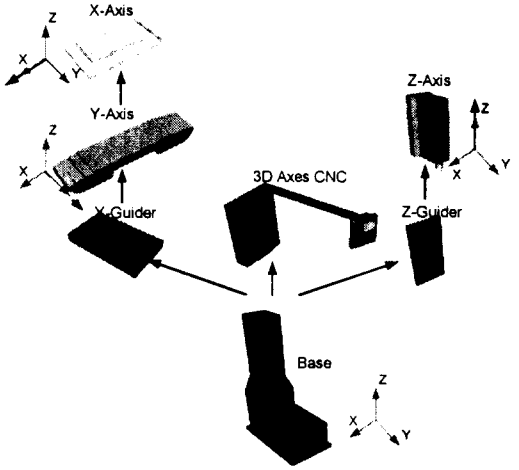
where $cv = \cos v$, $sv = \sin v$, $kv = 1 - \cos v$

$A = [a_x, a_y, a_z]$. 따라서, 어떤한 프레임 관점에서 본 형상정보를 ${}^bG$라고 한다면 ${}^bG = T \cdot G = \bigcup_{g \in G} T \cdot g$로 계산 가능하고, 이와 유사한 방법으로, 연속적인 행렬식의 곱을 이용하여 절대좌표계에서의 i번째 프레임의 정보 ${}_i^UT$와 ${}^UG$도 구할 수있다[Hwang 99].

GKDEVS를 이용하여 3축 CNC(computer numerical control) 밀링(milling) 시스템의 모델링 예에 대하여 <그림 1>은 동적인 행태를 갖는 요소와 이들간의 연결(coupling)관계를 <그림 2>에서는 사용된 GKDEVS와 이들간의 계층관계를 보여주고 있다. <그림 1>에서 알 수 있듯이, 3축 제어기(3D Axes CNC)는 NC 파일을 읽어 가면서 X,Y,Z축이 이동해야할 목적지와 이때의 속도를 각 축에 보낸다. 이때 각 축으로부터 목적지에 도달했음을 알리는 신호(Curr)를 받게되며 만약, 축 이동 중에 3차원 공간상에서 충돌사건(c)이 발생하게되면, Alarm 이벤트로 제어기에게 전달되며, 이때 제어기 모델은 각 축의 이동을 정지시키게 된다.



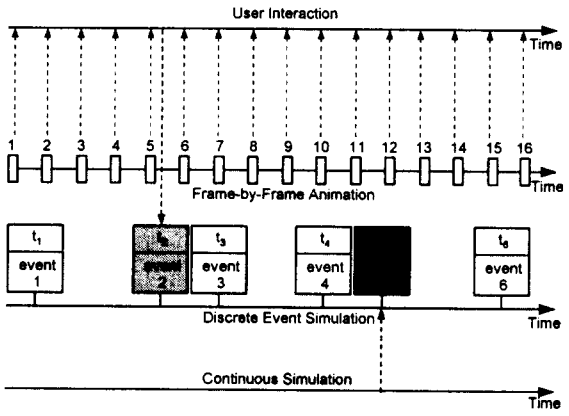
<그림 1> CNC Milling 모델의 상태변이도



<그림 2> CNC Milling 모델의 계층관계

4. 추상화 시뮬레이션

본 연구에서 다루는 시뮬레이션은 이산사건 시뮬레이션과 연속적인 운동에 의한 공간상의 충돌 등을 고려한 연속상태 시뮬레이션 그리고 이러한 상황을 사용자에게 보여주고 사용자의 입력을 처리하는 사용자 참여 시뮬레이션까지를 포함하고 있으며 <그림 3>은 이들 시뮬레이션들간의 관계를 보여주고 있다.



<그림 3> 시뮬레이션 종류와 관계

본 연구에서는 <그림 3>에서 제시한 시뮬레이션의 적용을 위해 기존의 DEVS 모델의 추상화 시뮬레이션 기법[Zeigler 84]과 유사한 방법론을 적용하였다. 즉, 모델과 데이터의 분리를 꾀하여 GKDEVS에 대한 시뮬레이션 프로세서인 GKSimulator를 설계하였는데. GKSimulator는 기존의 Coordinator에서 수행하던 계층적인 스케줄링의 기능과 기존의 Simulator에서 수행하던 기능을 포함하고 있다. 본 시뮬레이션에서 고려하고 있는 메시지의 종류는 ① (*,t): 내부변이 이벤트 메시지; ②(x,t): 외부변이 이벤트 메시지; ③(done,t): 재스케줄 메시지; ④(∅,t): 공이벤트 메시지; ⑤ (c,t): 연속상태에서 이벤트(충돌) 발생 메시지로 구성된다.

시뮬레이터의 계층구조상에서 가장상위에 존재하는 RootCoordinator는 (1)스케줄된 이산사건 이벤트(*,t), (2) 사용자의 입력이벤트(x,t) (3) 연속상태 메시지(∅,t)중에 하나를 선택하여 계층구조를 따라서 시뮬레이터에 전달하게 된다. <그림 4>는 RootCoordinator의 이러한 과정을 설명하고 있다.

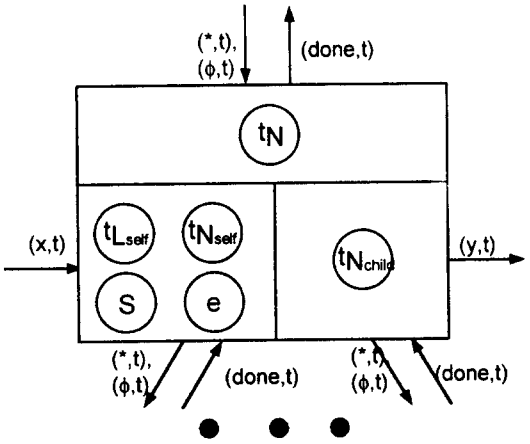
```
Procedure RootCoordinator:main(TimeType t_f,
                               TimeType t_ci, TimeType t_vi)
 1 TimeType t_ne = its child simulator's t_N;
 2 TimeType t_cc := 0; /* 현재 연속상태 시뮬레이션 진행된 시각 */
 3 TimeType t_nv := t_vi;
 4 while (buffer_user_event is not empty ||
         MIN(t_ne, t_nv, t_cc+t_ci) < t_f) begin
 5    if buffer_user_event is not empty then
 6       x_user = remove the first event from buffer_user_event;
 7       send (x_user, t_now) to its child simulator;
 8    else if t_cc < MIN(t_ne, t_nv) then
 9          TimeType t_nc := MIN(t_cc+t_ci, MIN(t_ne, t_nv));
10          t_cc := CheckCollision(t_cc, t_nc);
11          if t_cc < t_nc then /* no collisions */
12             t_ne := its child simulator's t_N;
13          end if
14    else if t_ne <= t_nv then
15          send (*, t_ne) to its child simulator;
16          send (ϕ, t_ne) to its child simulator;
17          t_ne = its child simulator's t_N;
18    else
19          update_screen(); /* draw windows */
20          t_nv := t_nv + t_vi;
21    end if
22 end_of_while
23 send (ϕ, t_f) to its child simulator;
```

<그림 4> RootCoordinator의 메인 루프

GKDEVS의 시뮬레이터인 GKSimulator는 <그림 5>에서와 같이 자신의 마지막 이벤트 발생시각(t_{Lself}), 모델의 상태변수집합(S), 상태지속시간(e), 그리고 자신의 다음 이벤트 계획시각(t_{Nself})를 관리하고 있다. 또한 기존 DEVS의 연결모델의 Coordinator가 가지고 있는 자식들의 다음 이벤트 계획시각(t_{Nchild})을 기억하고 있다.

GKSimulator가 스케줄된 이벤트를 알리는 (*,t)를 받았을 경우 자신의 스케줄에 의한 것인지 또는 하위의 GKDEVS에 스케줄인 지를 판단하여 자신의 것인 경우에는 출력함수(λ)를 이용하여 출력 이벤트를 발생하여 전달하고, 내부 상태변이 함수 (δ_{int})를 이용하여 상태 변수를 변화시키고 자신의 마지막 이벤트 발생시각(t_{Lself})과 자신의 다음 이벤트 스케줄시각(t_{Nself})을 계산한다. 만약 자신의 스케줄이 아니라

하위의 스케줄인 경우에는 하위의 모델 중에 하나를 선택하여 그것에 (*,t) 메시지를 전달하고, 후에 하위의 스케줄을 재 정렬하고 자식 중에 가장 빠른 스케줄을 t_{Nchild}에 기억한다. 자신의 스케줄 또는 하위의 스케줄 공히 다음 이벤트 스케줄시각(t_N)을 t_{Nself}와 t_{Nchild}의 최소값으로 설정한다.



<그림 5> GKSimulator의 구조와 messages

GKSimulator가 (x,t)[1]를 받은 경우에는 소요시간(e)를 갱신하고 외부상태변이 함수(δ_{ext})를 이용하여 상태변수를 갱신하고 t_{Lself}와 t_{Nself} 그리고 t_N을 갱신하고 상위의 GKSimulator에게 (done, t_N)을 보내어 자신이 속한 스케줄 리스트의 재 정렬을 알린다.

GKSimulator가 (done,t)를 받는 경우에는 하위의 GKSimulator의 스케줄을 재 정렬하고 가장 빠른 스케줄을 t_{Nchild}에 기억하며 다음 이벤트 스케줄시각(t_N)을 t_{Nself}와 t_{Nchild}의 최소값으로 설정한다. 그리고 또 다시 상위에 (done, t_N)를 보내어 알린다.

GKSimulator가 (ø,t)를 받았을 경우에는 소요시간(e)를 갱신하고 외부상태 δ_{ext}를 이용하여 ${}^{U}T_p$, ${}^{U}T$, 그리고 각 객체의 위상정보인 ${}^{U}G$를 갱신한다. 이것은 t시점에서의 위상상태를 말하며 공간상의 이동에 따른 충돌등을 계산할 때 사용된다. 자신의 상태를 갱신한 후에는 자식들에게도 (ø,t)를 전달하여 시점 t에서의 연속상태를 계산하도록 한다.

공간상의 객체의 이동시에 발생할 수 있는 충돌과 같이 미리 그 시점을 스케줄 하기 힘든 이벤트(unschedulable event)는 t 시점을 잘게 쪼개어 연속상태의 계산을 거쳐서 각 시점에서의 연속상태 이벤트를 계산하게 되는데 이때에도 (ø,t)를 이용하게 된다. GKSimulator를 이용한 충돌의 연속상태 이벤트 검출 방법은 [Hwang 99]를 참조 바라며, DEVS에 기반한 일반적인 혼합 시뮬레이션 방법론에서 연속상태 이벤트의 처리는 [Praehofer 91]을 참조바란다. <그림 6>은 GKSimulator가 메시지를 받아 처리하는 과정을 정리한 것이다.

```
Procedure GKSimulator::when_receive_(*,t)
 1 if t = t_L then
 2    if t_Nself < t_Nchild or
         t_Nself = t_Nchild and self is selected by SELECT then
 3       y := λ(s);
 4       send to (y,t) to influencee simulators;
 5       s := δ_int(s);
 6       t_Lself := t;
 7       t_Nself := t_Lself + ta(s)
 8    else
 9       find the imminent simulators;
10       select one, i*, and send the input (*, t)to it;
11       t_Nchild := minimum of component t_i*;
12       resort children by their t_i ;
13    end if
14    t_L := MIN(t_Nself,t_Nchild);
15 else
16    ERROR;
17 end if
Procedure GKSimulator::when_receive_(x,t)
 1 if t_L <= t <= t_L then
 2    e := t-t_L;
 3    s := δ_ext(s,e,x);
 4    t_Lself := t;
 5    t_Nself := t_L + ta(s);
 6    t_L := MIN(t_Nself,t_Nchild);
 7    send (done, t_L) to the parent Simulator;
 8 else
 9    ERROR;
10 end if
Procedure GKSimulator::when_receive_(done,t)
 1 if t_L <= t <= t_L then
 2    resort children by their t_i ;
 3    t_Nchild := minimum of component t_i*;
 4    t_L := MIN(t_Nself,t_Nchild);
 5    send (done, t_L) to the parent Simulator;
 6 else
 7    ERROR;
 8 end if
Procedure GKSimulator::when_receive_(ϕ,t)
 1 if t_L <= t <= t_L then
 2    e := t-t_L;
 3    s := δ_con(s,e,ϕ);
 4    for-each m in child simulators
 5       send (ϕ,t) to m;
 6    end_of_for-each
 7 else
 8    ERROR;
 9 end if
Procedure GKDEVS::δ_con(s,e,ϕ)
 1 T_p := get pareant T;
 2 T   := T_p s.T;
 3 G   := G T;
```

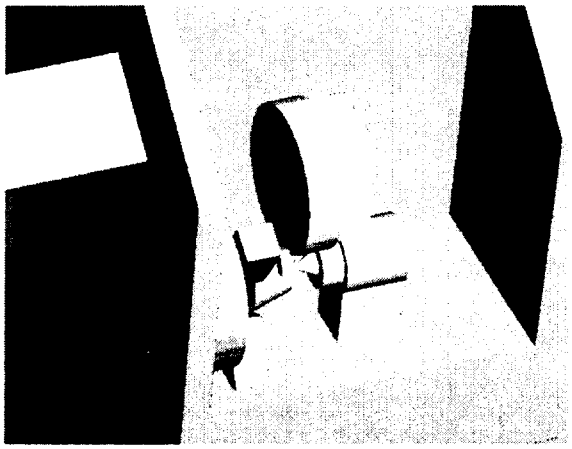
<그림 6> GKSimulator의 메시지 처리과정

5. 유연 제조시스템으로의 적용

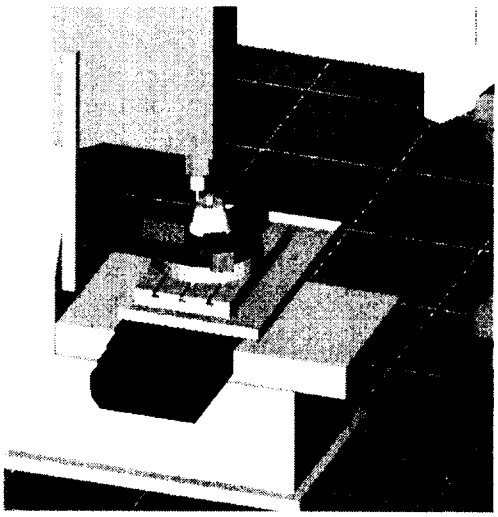
GKDEVS모델과 GKSimulator를 이용하여 유연 제조시스템(Flexible Manufacturing System)의 시뮬레이션 연구에 적용하였다. 모델링하고자 하는

1) (x,t) 메시지는 (*,t)메시지와 달리 계층적으로 전달되는 것이 아니라 output을 발생한 모델에게서 직접 전달되는 것으로 설계하였다.

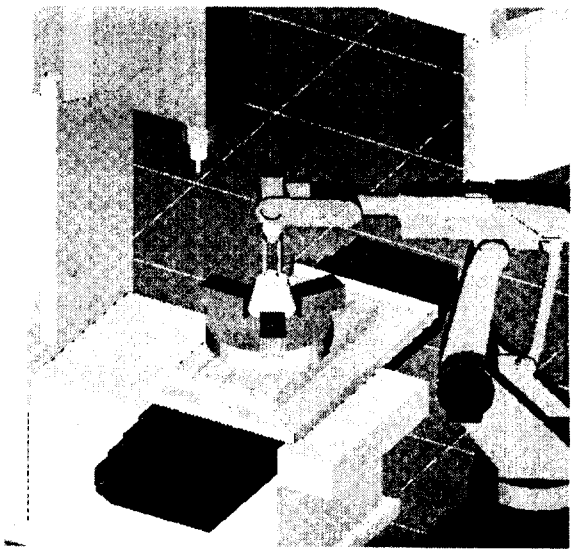
대상시스템은 한 대의 CNC 선반(Turning Machine)과 한 대의 CNC 밀링(Milling Machine)을 절삭설비로 가지고 있으며 하나의 입력 테이블과 출력 테이블을 갖는다. 중간의 물류이송을 담당하는 설비로 6관절 로봇이 사용되는데 이 로봇은 레일 상에 왕복운동을 하는 대차설비 위에 탑재되어 있다.



<그림 7> 선반의 시뮬레이션



<그림 8> 밀링 시뮬레이션



<그림 9> 로봇의 물류 공급 시뮬레이션

4장에서 설명한 시뮬레이션 방법론에 따라서, 선반 밀링 그리고 로봇의 물류 이송시의 충돌들의 발생 유무를 <그림 7><그림 8><그림 9>에서와 같이 단위 설비의 작업별로 수행할 수 있다. 이때에는 충돌 등의 발생과 이의 제거에 관심을 가지고 시뮬레이션을 수행하게 되므로 연속상태의 샘플링 간격을 조밀하게 하는 것이 유리하다. 각 단위 작업의 설계가 완료되며, 전체 시스템 관점에서의 유기적인 운영에 초점을 맞추어 <그림 10>과 같은 시스템 레벨의 시뮬레이션이 진행된다. 이때에는 단위작업에서 수행된 충돌등의 검사는 불필요하므로, 연속상태 시뮬레이션의 옵션은 선택하지 않는 것이 속도 측면에서는 바람직하겠다.

6. 결론

본 연구에서는 집합론에 기반한 이산사건 시스템의 기술 형식론 중에 하나인 DEVS를 기초로 하여 가상제조시스템을 모델링할 수 있는 GKDEVS를 제안하였다. 제안된 GKDEVS는 이산사건시스템의 특징과 공간상의 운동에 따른 계층적이고 연속적인 영향관계를 모델링할 수 있다. 뿐만 아니라, GKDEVS에 대한 추상화 시뮬레이션 방법론을, 이산사건 및 연속상태 시뮬레이션과 사용자 참여 시뮬레이션의 관점에서 제안되었고 이를 응용하여 유연제조시스템의 시뮬레이션을 단위설비들로부터 시스템 수준으로

확장하면서 실시하였다.

참고 문헌

[Choi 99] 최병규, "VMS 기술과 산업공학의 역할," IE 매거진, V6, N1, 1999, p18-21

[Hwang 99] 황문호, 자동화 제조시스템용 시뮬레이터 개발을 위한 DEVS 형식론응용에 관한 연구, 박사학위 논문, 산업공학과, 한국과학기술원, 1999

[Iwata 95] K. Iwata, M. Onosato, K. Teramoto, S. Osaki, "A Modelling and Simulation Architecture for Virtual Manufacturing Systems," Annals of the CIRP, V44, January, 1995, p399-402

[Jones 93] K.C. Jones and M.W. Cygnus, "Virtual Reality for Manufacturing Simulation," Proceedings of the 1993 Winter Simulation Conference, 1993, p882-887

[Kim 95] B.H. Kim, A Study on the Development of an Animation for AMS Graphic Simulation, MS Thesis, IE Dept., KAIST, 1995 (in Korean)

[Kim 97] T.G. Kim, Lecture Note: EE617, EE Dept, KAIST, 1997 (http://sim.kaist.ac.kr)

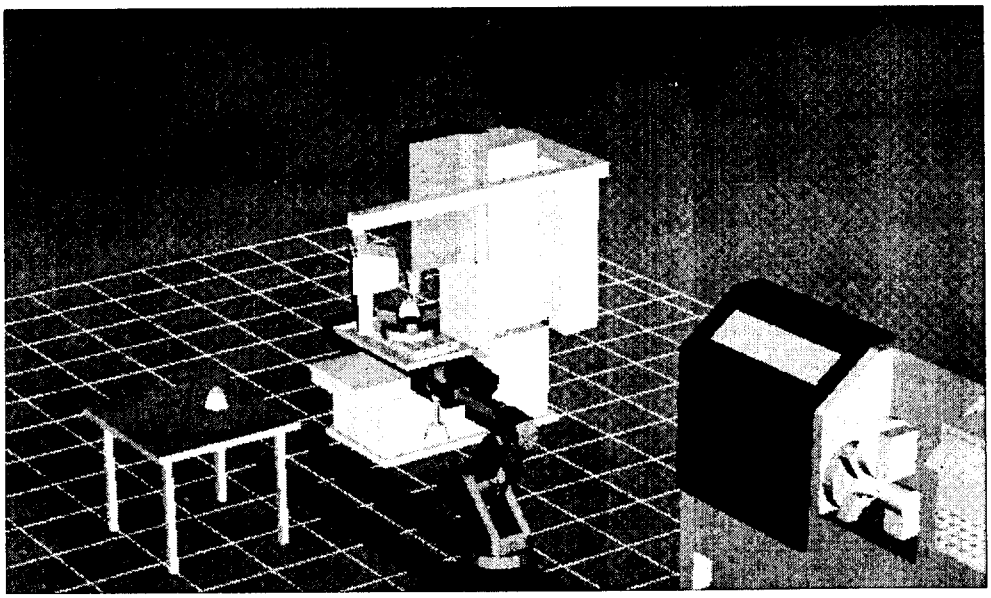
[Klingstam 99] p. Klingstam and P. Gullander, "Overview of Simulation Tools for Computer-Aided Production Engineering," Computer in Industry, V38, 1999, p173-186

[Onosato 93] M. Onosato and K. Iwata, "Development of a Virtual Manufacturing System by Integrating Product Models and Factory Models," Annals of the CIRP, V42, 1993, p475-478

[Praehofer 91] H. Praehofer, System Theoretic Foundation for Combined Discrete-Continuous System Simulation, Ph.D. Thesis , Department of Systems Theory, University of Linz, Austria, 1991

[Zeigler 76] Bernard P. Zeigler, *Theory of Modelling and Simulation*, John Wiley & Sons, New York, 1976,

[Zeigler 84] Bernard P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, Orlando, FL, 1984,



<그림 10> FMS 전체 시뮬레이션