3차원 대화형 시뮬레이션 모델기술언어로서의 VRML

VRML as a Modeling Language for 3D Visual Interactive Simulation

김형도*
Kim, Hyoung Do

Abstract

VRML (Virtual Reality Modeling Language) is an Web-based standard for modeling 3D spaces and provides applications with 3D interactive interfaces. With its recent upgrade, it supports events, routes, scripts, and other behavior medeling constructs. This paper approaches VRML as a simulation modeling language. This approach promotes the sharing and distribution of simulation results and demonstration among distributed users as well as efficient modeling of systems through the direct mapping of 3D objects and behaviors. This paper analyzes the behavior modeling constructs of VRML, presents effective modeling alternatives through the modeling of a simple material processing system, and discusses the upgrade direction of VRML as a foundation for distributed interactive simulation system.

Keyword : Simulation, VRML, 3D Interactive Interface, Behavior Modeling

1. 서 론

시뮬레이션 대상 시스템의 내부 상태/현상을 표현하여 사용자의 이해를 돕거나 사용자와의 대화를 통한 결과를 보여주는 경우, 시뮬레이션 패키지가 제공하는 방식을 사용하거나 일반 언어 또는 그래픽 라이브러리와 같은 패키지를 사용하여 개발하는 경우가 대부분이다. 비행 시뮬레이터 등과 같은 교육/훈련용 시뮬레이션에서 사용자의 입력이 대상 시스템에 미치는 영향을 실시간으로 보여주는 경우가 여기에 해당한다. 이런 경우 시뮬레이션 시스템 개발이 시뮬레이션 언어/패키지 중심으로 진행되어 개발생산성이 떨어지고, 대화형 인터페이스를 따로 개발해야 하는 부담이 있다. 생산시스템 분야 [5], 통신망 설계분야 [3] 등 특정 분야에서는 그 분야의 지식을 활용하여 그래픽 모델링을 사용한 경우도 있으나 사용자간 유통성이 부족하고 시뮬레이션 시스템간 통합도 어렵다. 특히 네트워크를 통한 시뮬레이션 시스템의 유통과 공유에 많은 제약이 따른다.

VRML(Virtual Reality Modeling Language) [1, 4]은 웹을 위한 3차원 장면 기술용 표준 언어로서 3차원 대화

* (주)데이콤 종합연구소, 멀티미디어 기술팀

형 인터페이스를 제공해 줄 뿐 아니라 최근에는 사건(Event), 스크립트(Script)등을 제공하여 3차원 시뮬레이션을 위한 기반을 제공하고 있다. VRML을 사용하여 3차원 대화형 시뮬레이션이 지원되면 3차원 인터페이스 측면의 장점뿐 아니라 인터넷을 통한 원격 데모가 가능하며 원격 시뮬레이션 시스템간의 상호 접속성도 높아져 분산 대화형 시뮬레이션[6]도 쉽게 구현될 수 있다. 본 논문에서는 VRML의 기능을 시뮬레이션의 관점에서 분석/검토하고, 3차원 대화형 시뮬레이션의 효과적인 모형화 방법과 이를 위한 향후의 VRML개선방향을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 VRML의 전반적인 내용을 소개하고 시뮬레이션에 관계된 구성요소들을 자세히 검토한다. 3장에서는 간단한 생산시스템 시뮬레이션 예제를 VRML을 사용하여 모형화해 보고, 이를 바탕으로 4장에서는 이러한 3차원 시뮬레이션 모형 개발의 문제점과 이에 대한 대안을 논한다. 마지막으로 5장에서는 논문을 정리하고 향후 연구방향을 제시한다.

2. VRML

VRML은 인터넷상에서 3차원 데이타를 표현하고 검색하기 위한 표준으로서 95년도에 1.0 표준[1]이 제정되어 3차원 정보검색의 새장을 열었으며 이후 표현력을 보강하고 애니메이션, 사건등 동적인 능력을 보강한 2.0표준[4]이 96년 8월에 완료되었다.

2.1 VRML개념

개념적으로 VRML은 노드(Node)와 필드(Field)로 구성되어 있다. 노드는 3차원 공간상의 물체, 카메라, 빛등 직접 보이는 물체와 색, 위치, 방향등 물체의 속성등이 여기에 해당된다. 노드는 노드타입명과 함께 필드를 정의하기 위한 블럭으로 구성된다. 노드는 다른 노드를 포함할 수 있다. 다른 노드를 포함하는 노드는 부모노드, 포함되는 노드는 자식노드라고 부른다. 최상위 노드를 제외한 모든 노드는 다른 노드의 자식노드이다.

필드란 노드의 블럭안에 정의된 노드의 속성을 말한다. 대부분의 노드는 하나 이상의 필드를 가지고 있다. 필드는 3차원 그래픽에 필요한 수치체계를 표현하기 위한 필드타입중 어느 한 종류에 속하는 값을 가지게 된다. 이러한 필드타입으로는 하나의 값을 가지는 SF(Single Field)와 두개 이상의 값을 가지는 MF(Multiple Field)가 있다. 필드가 생략된 경우 기본값을 가지게 된다. 예를 들면 육면체 노드 Cube는 width, height, depth라는 세개의 필드를 가질 수 있다. 필드는 필드의 이름과 필드값으로 구성된다.

VRML 2.0표준은 정적인 VRML 1.0표준에 사실적인 표현능력을 향상시키고 사용자와의 상호작용, 애니메이션(Animation), 스크립트(Script)등 동적인 기능을 추가하고 있다. 안개, 땅과 하늘, 3차원 음향등 다양한 사실적 표현능력을 제공한다. 사용자와의 상호작용을 위해 지형 센서들, 시간의 흐름을 추적하는 센서, 물체간 충돌을 알아내는 충돌검색등의 기능을 제공하고 있다. 또한 애니메이션을 위한 다양한 인터폴레이터(Interpolator)등이 제공된다. 스크립트는 각종 물체들을 움직이게 만들기 위한 논리처리등 지능적인 측면을 지원할 수 있도록 한다. 이상의 관련 노드들은 사건(Event)을 발생시키거나 받아들일 수 있으며 사건간을 라우트(Route)로 연결하여 동적인 기능을 구현할 수 있다.

또한 VRML 2.0은 추가적인 확장을 쉽게 할 수 있는 토대를 제공하고 있다. 프로토타입을 통해 여러노드를 모아서 하나의 새로운 노드로 만들 수 있고 이를 일반 노드와 같이 필드값을 사용하여 다양한 인스턴스를 만들 수 있다.

2.2 행위 기술 관련 노드들

2.2.1 사건

VRML 2.0에서 시뮬레이션과 관련된 가장 중요한 개념은 사건(Event)으로서 한 노드로부터 다른 노드로 전달되는 메시지를 말한다. 각 노드는 노드의 종류에 따라서 받을 수 있는 사건(입력사건)과 보낼 수 있는 사건(출력사건)을 가지고 있다. 사건은 명시적으로 정의되거나 개방된 필드와 관련되어 묵시적으로 정의된 사건이 있다. 개방된 필드(exposedField)란 초기값으로 고정되어 사용되는 일반 필드와 달리 초기값이 다른 노드에 의해 바뀔 수 있는 필드를 말한

다. 이러한 필드들은 "set_개방필드명"으로 된 입력사건(eventIn)을 받아 들일 수 있고, "개방필드명_changed"로 된 출력사건(eventOut)을 내보낼 수 있다. 물체간의 충돌을 테스트하는 그룹 노드인 Collision노드의 프로토타입인 〈그림 1〉을 가지고 구체적으로 설명해 보면 다음과 같다.

```
Collision {
eventIn       MFNode addChildren
eventIn       MFNode removeChildren
exposedField  MFNode children       [ ]
exposedField  SFBOOL collide        TRUE
field         SFVec3f bboxCenter    0 0 0
field         SFVec3f bboxSize      -1 -1 -1
field         SFNode proxy          NULL
eventOut      SFTime collide Time
}
```

〈그림 1〉 Collision노드 프로토타입

addChildren과 removeChildren은 모든 그룹노드가 가지는 입력사건으로서 children필드에 노드를 추가하거나 삭제할 경우 입력되는 사건을 말한다. SFBOOL유형의 데이타를 가지는 collide는 set_collide입력사건과 isCollide출력사건을 가진다. collideTime은 충돌이 일어난 시간을 알려주는 출력사건이다. 외부노드는 "get_collideTime"을 사용하여 출력사건과 관련된 값을 임의의 시점에 확인할 수도 있다. 〈그림 1〉의 프로토타입에서 굵은 글씨가 실제로 Collision노드를 정의할 때 사용되는 부분이다.

2.2.2 라우트

사건을 발생시키는 노드와 사건을 받아들이는 노드사이의 연결을 라우트(Route)라고 하는데 출력사건으로부터 입력사건으로의 라우트만이 허용된다. 또한 입력사건과 출력사건간의 자료유형은 일치해야만 한다. 최초의 사건이 발생한 이후에는 라우트를 통하여 다른 노드들에 사건이 전파된다. 이런 노드들은 추가적인 사건을 발생시킬 수 있는데, 이렇게 연결된 사건들은 동일한 사건 발생시간(Timestamp)을 가지게 된다. 두개 이상의 출력사건으로 부터 한 입력사건으로 라우트가 연결된 경우 Fan-in이라고 하는데 출력사건들이 동일한 발생시간을 가진 경우에 수행 순서가 상황에 따라 다르므로 주의해야 한다. 하나의 출력사건이 2개 이상의 입력사건으로 라우드가 연결된 경우 Fan-out이라고 하는데 이 때는 각각의 라우트를 통하여 동일한 발생시간을 가지는 출력사건을 내보내게 된다.

2.2.3 센서 노드

센서노드는 사용자의 입력 또는 시간의 변화에 따라서 사건을 발생시키는 노드이다. 사용자의 입력에 따라서 사건을 발생시키는 센서(Sensor)로는, 마우스의 지형적인 접근에 따라서 사건을 발생시키는 ProximitySensor, 사용자가 센서를 마우스로 누름으로서 사건이 발생하는 TouchSensor, 노드가 화면에 보이기 시작하면 사건을 발생시키는 VisibilitySensor등과 마우스의 이동에 따라서 다른 종류의 사건을 발생시키는 CylinderSensor, PlaneSensor, SphereSensor 등(이 세 가지 센서를 드래그 센서라고 함)이 있다. 시간에 따라서 사건을 발생시키는 센서로는 TimeSensor가 있으며 주어진 시간에 또는 일정한 시간을 주기로 하여 반복적으로 사건을 발생시킨다. 시간은 1970년 1월 1일 이후의 시간을 초로 표시하는 것으로 시뮬레이션의 논리적 시간이 아닌 실제시간임을 유의해야 한다. 따라서 정의된 시간만큼 시간을 소비하게 되며 시간을 단축하기 위해서는 일정한 비율로 시간을 확대 또는 축소해야 한다.

2.2.4 스크립트 노드

사건이 어떤 효과를 미치는 지를 결정하기 위해서는 논리 구현과 상태관리가 필요하며 스크립트 노드가 이러한 목적에 사용된다. 스크립트 노드는 다른 노드의 사건을 받아들여 처리하고 다른 노드에 사건을 내보내는 역할을 수행한다. 사건을 받아들여 스크립트 노드가 활성화되면 url필드에 정의되어 있는 프로그램 또는 스크립트가 실행되어 사건을 처리하게 된다. 개념적으로 스크립트 노드가 사건을 받아들여 처리하는데는 시간이 경과되지 않은 것으로 처리된다. 스크립트 노드에 의해 받아들여진 사건은 사건값과 사건발생시간을 인수로 하여 스크립트 내의 적당한 함수로 전달

된다. 스크립트는 지정된 노드의 노출된 필드, 입력사건, 출력사건등을 접근할 수 있고 스크립트 노드의 출력사건을 통하여 값을 전달할 수 있다. 지정된 노드는 스크립트 노드의 필드 또는 입력사건을 통해 접근 가능한 노드를 말한다.

2.2.5 프로토타입 노드

프로토타입 노드는 기존의 노드유형을 바탕으로 새로운 노드유형을 정의할 수 있도록 해준다. 이 노드는 〈그림 2〉와 같이 크게 프로토타입 선언부와 정의부로 구성된다. [ ]로 감싸진 부분이 선언부이며 { }로 감싸진 부분이 정의부이다.

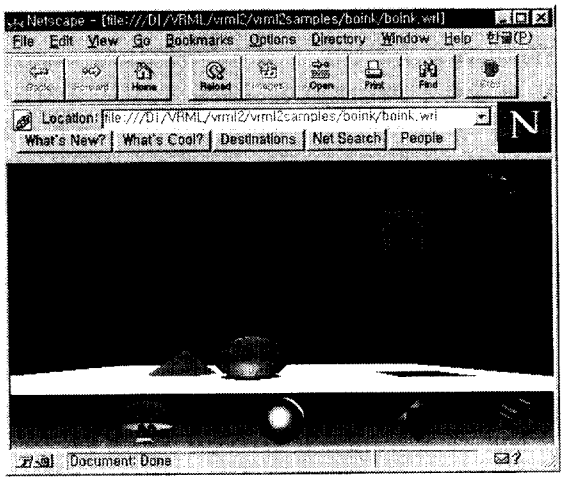
```
PROTO prototypename [ eventIn     eventtypename   name
        eventOut      eventtypename   name
        exposedField  fieldtypename   name  defaultvalue
        field         fieldtypename   name  defaultvalue
        … ]
{
    라우트와 프로토타입
    최초의 노드
    노드, 라우트, 프로토타입
}
```

〈그림 2〉 프로토타입 노드의 프로토타입

프로토타입 노드의 노드유형은 최초의 노드의 유형을 따른다. 따라서 프로토타입 노드가 어느곳에 사용될 수 있는가 하는 점은 최초의 노드에 의해 좌우된다. 프로토타입 선언부의 모든 입력사건은 프로토타입 정의부의 입력사건 또는 노출된 필드와 IS를 사용하여 연계되며, 모든 출력사건은 프로토타입 정의부의 출력사건 또는 노출된 필드와 연계된다. 프로토타입 정의부의 필드들은 IS를 통하여 선언부의 필드들과 연계될 수 있다.

2.3 행위기술 예

이제 〈그림 3〉과 같이 육면체가 공처럼 네번 튀어서 원점으로 되돌아 오는 과정을 무한히 반복하는 행위를 VRML로 기술하면 〈그림 4〉와 같다.



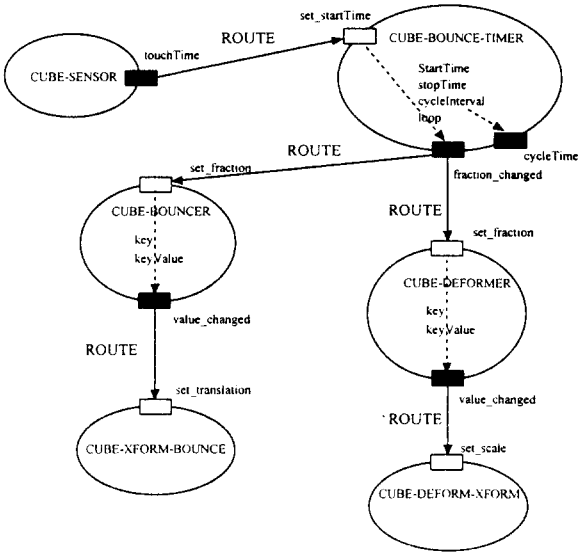
〈그림 3〉 3차원상에서의 육면체의 이동

변환 노드인 CUBE-XFORM-BOUNCE는 접촉센서노드 CUBE-SENSOR, 시간센서노드 CUBE-BOUNCE-TIMER, 변환노드, 위치 인터폴레이터 CUBE-BOUNCER와CUBE-DEFORMER 등을 자식노드로 가지고 있다. 먼저 사용자가 CUBE-SENSOR를 마우스로 누르면 CUBE-SENSOR노드가 출력사건인 touchTime을 발생시키면 이 사건은 첫번째 ROUTE에 정의된 대로 CUBE-BOUNCE-TIMER에 입력사건 set_startTime을 전달한다. 이제 이 시간센서는 2.5초마다 반복하여 출력사건인 cycleTime을 발생시키고, 시간의 변화에 따라서 현재시간이 주기내에서 차지하는 비율을 나타내는 fraction_changed사건을 지속적으로 내보낸다. 이 사건의 값은 0이상 1이하의 값을 가진다. 이제 2번째와 3번째 ROUTE 정의에 의하여 이 출력사건은 CUBE-BOUNCER와CUBE-DEFORMER의 입력사건인 set_fraction에 각각 전달되고 위치 인터폴레이터는 이 값을 키(key)로 사용하여 키값(keyValue)을 결정하고 출력사건인 value_changed를 발생시킨다. 이렇게 발생된 출력사건은 4번째와 5번째의 ROUTE정의에 의하여 변환노드인 CUBE-XFORM-BOUNCE, CUBE-DEFORM-XFORM에 set_translation, set_scale입력사건을 전달하게 되며 이 사건을 전달받은 변환노드는 각각 해당 변환을 수행한다. CUBE-XFORM-BOUNCE는 육면체의 위치이동을, CUBE-DEFORM-XFORM는 육면체의 모양변화를 수행한다.

```
...
DEF CUBE-XFORM-BOUNCE Transform {
  children [
    DEF CUBE-SENSOR TouchSensor{ },
    DEF CUBE-BOUNCE-TIMER TimeSensor {
      loop TRUE
      stopTime 1
      cycleInterval 2.5
      },
      DEF CUBE-DEFORM-XFORM Transform { children
        DEF CUBE Inline { url "cube.wrl" }
        scale 1.3 .7 1.3
      },
      DEF CUBE-BOUNCER PositionInterpolator {
        keys [ 0, .04, .1, .2, .3, .4, .5, .6, .7, .8, .9, .96, 1 ]
        values [ 0 0 0, 0 .66 0,
             0 1.664 0, 0 3.036 0,
             0 4.016 0, 0 4.604 0,
             0 4.8 0,
             0 4.604 0, 0 4.016 0,
             0 3.036 1, 0 1.664 0,
             0 .66 0, 0 0 0
        ]
      },
      DEF CUBE-DEFORMER PositionInterpolator {
        keys [ 0, .04, .1, .2, .3, .4, .5, .6, .7, .8, .9, .96, 1 ]
        values [ 1.3 .7 1.3, .895 1.105 .895,
             .92 1.08 .92, .955 1.045 .955,
             .98 1.02 .98, .995 1.005 .995,
             1 1 1, .995 1.005 .995,
             .98 1.02 .98, .955 1.045 .955,
             .92 1.08 .92, .895 1.105 .895,
             1.3 .7 1.3
        ]
      }
    }
  ],
  ...
  ROUTE CUBE-SENSOR.touchTime TO CUBE-BOUNCE-TIMER.set_startTime
  ROUTE CUB-BOUNCE-TIMER.fraction_changed TO CUBE-BOUNCER.set_fraction
  ROUTE CUBE-BOUNCE-TIMER.fraction_changed TO CUBE-DEFORMER.set_fraction
  ROUTE CUBE-BOUNCER.value_changed TO CUBE-XFORM-BOUNCE.set_translation
  ROUTE CUBE-DEFORMER.value_changed TO CUBE-DEFORM-XFORM.set_scale
  ...
```

〈그림 4〉 육면체의 이동에 관한 행위기술

이상의 과정을 도식화하면 〈그림 5〉와 같이 표현할 수 있다. 이 그림에서 원은 노드를, 사각형은 사건을, 실선 화살표는 ROUTE에 의한 사건 전달을, 점선 화살표는 노드안에서의 사건 전달 과정을 나타낸다. 특히 검정색 사각형은 출력사건을, 흰색 사각형은 입력사건을 표시한다.



〈그림 5〉 육면체 이동에 관한 사건 전달/처리 과정

3. VRML을 사용한 시뮬레이션

이 장에서는 간단한 생산 시스템을 VRML 2.0을 사용해 모형화하고 VRML을 사용한 3차원 대화형 시뮬레이션의 가능성을 제시한다.

3.1 대상 시스템

모형화 대상 시스템은 육면체로 된 원재료을 가공하여 구형으로 만드는 공정을 가진 간단한 생산 시스템이다. 먼저 육면체는 [10,12]초 사이의 일양분포(Uniform Distribution)에 따라서 생성되며 가공기계까지는 5초에 걸쳐서 이동한다. 기계에 도착하면 [5,8]초 사이의 일양분포에 따르는 가공과정을 거쳐 구형으로 변화된 제품은 5초 동안 이동한 뒤 사라진다.

3.2 시뮬레이션의 시작

시뮬레이션은 접촉센서(START-BUTTON-SENSOR)에 의하여 시작되며 출력사건인 startTime이 스크립트(START-BUTTON-SCRIPT)에 전달된다. 이 스크립트는 2가지의 출력사건값을 생성하는데 먼저 modifiedTime을 계산하고 라우트로 음악소스(START-BUTTON-SOUND)의 startTime을 설정한다. 또한 일양분포에 따른 최초의 육면체 도착을 구현하기 위해서 arrivalTime을 계산하고 라우트로 시간센서(CUBE-PRODUCT-TIMER)의 cycleInterval값을 수정한다. 이상은 〈그림 6〉과 같이 정의할 수 있다.

3.3 확률분포에 따른 육면체의 도착과 이동

시뮬레이션이 시작된 후에는 이 시간센서의 cycleTime출력사건을 스크립트(CUBE-PRODUCT-SCRIPT)의 startTime으로 연결하여 실행한다. 이 스크립트는 육면체가 정의된 VRML파일을 읽어들이고 앞에서와 같이 일양분포를 따르는 다음번 도착시간을 계산하여 라우트로 CUBE-PRODUCT-TIMER의 cycleInterval값을 설정한다. 도착한 육면체는 2장에서의 예와 같은 방법으로 MACHINE를 향하여 이동하게 된다.

3.4 육면체의 이동과 큐관리

육면체의 이동이 끝나는 지점에서 QUEUE-SCRIPT를 작동시켜 육면체 큐를 점검하고 큐에 대기중인 육면체가 없으면 바로 가공을 위한 시간센서 MACHINE-TIMER를 작동시킨다. 큐가 있으면 큐에 등록하고 대기중인 육면체수를 증가시킨다. 이러한 육면체의 이동과 큐관리는 〈그림 7〉과 같이 정의할 수 있다.

3.5 육면체의 가공과 구의 생성

가공을 위한 시간센서가 작동되면 스크립트를 작동시켜 일양분포에 따른 이후의 가공종료시간을 계산하여 cycleInterval을 설정하고 작동시킨다. 이 시간센서의 cycleTime출력사건은 MACHINE-SCRIPT의 endProcess 사건으로 연결되어 육면체를 소거하고 구를 생성하며 이동을 위한 시간센서를 작동시킨다. 또한 QUEUE-SCRIPT의 checkOut로 연결되어 큐를 검사하여 대기중인 육면체가 있는 경우 이 과정을 반복한다. 이러한 과정은 〈그림 7〉과 같이 정의될 수 있다.

```
DEF START-BUTTON-ROOT Transform {
  children [
    DEF START-BUTTON transform {
      children [
        DEF START-BUTTON-SENSOR TouchSensor{ },
        Transform { children
          DEF CUBE Inline {
            url "cube.wrl"
          }
          scale 1.3 .7 1.3
        },

        DEF START-BUTTON-SCRIPT Script {
          eventIn SFTime startTime
          eventOut SFTime modifiedTime
          eventOut SFTime arrivalTime
          url "vrmlscript:
            function startTime (time){
                modifiedTime = time - .2;
                arrivalTime = 10 + rand()*(12-10);
            }
          "
        },

        Sound {
          minFront 0.1
          minBack 0.1
          maxFront 1000
          maxBack 1000
          source DEF START-BUTTON-SOUND AudioClip {
            url "cube.wav"
            loop FALSE
          }
        },
      ]
    },
  ]
} # End of START-BOTTON-ROOT
ROUTE START-BUTTON-SENSOR.touchTime TO START-BUTTON-SCRIPT.startTime
ROUTE START-BUTTON-SCRIPT.modifiedTime TO START-BUTTON-SOUND.startTime
ROUTE START-BUTTON-SCRIPT.arrivalTime TO CUBE-PRODUCT-TIMER.cycleInterval
ROUTE START-BUTTON-SCRIPT.modifiedTime TO CUBE-PRODUCT-TIMER.startTime
```

〈그림 6〉 시뮬레이션 시작

```
DEF QUEUE Group { },
DEF QUEUE-SCRIPT Script {
  eventIn MFNode checkIn;
  eventIn MFNode checkOut;
  field SFInt32 num;
  field SFBool bWorking;
  field MFNode queue USE QUEUE;
  eventOut MFNode startMachine;
  url vrmlscript::
    function checkIn(value) {
      if (bWorking) {
        num = num + 1;
        queue.addChildren = value;
      }
      else {
        startMachine = value;
      }
    }
    function checkOut(value) {
      if (num>0) {
        MFNode node;
        node = queue.get1(0);
        queue.removeChildren = node;
        num = num - 1;
        startMachine = node;
      }
    }
}
DEF CUBE-TO-PROCESS Group { }
DEF MACHINE-TIMER TimeSensor { }
DEF MACHINE-SCRIPT Script {
  eventIn MFNode startProcess;
  eventIn SFTime endProcess;
  field MFNode cube USE CUBE-TO-PROCESS;
  eventOut SFTime endTime;
  url vrmlscript::
      function startProcess(value,time) {
        cube.addChildren = value;
        endTime = time + 5 + rand() * (8-5);
      }
      function endProcess(value) {
        MFNode node;
        node = cube.get1(0);
        cube.removeChildren = node;
      }
}
ROUTE QUEUE-SCRIPT.startMachine TO MACHINE-SCRIPT.startProcess
ROUTE MACHINE-SCRIPT.endTime TO MACHINE-TIMER.cycleInterval
ROUTE MACHINE-SCRIPT.endTime TO MACHINE-TIMER.startTime
ROUTE MACHINE-TIME.cycleTime TO MACHINE-SCRIPT.endProcess
ROUTE MACHINE-TIME.cycleTime TO QUEUE-SCRIPT.checkOut
```

〈그림 7〉 육면체의 이동과 큐관리, 육면체의 가공 과정에 대한 기술

3.6 구의 이동과 소멸

구는 육면체와 동일한 방법으로 이동하게 되며 이동이 끝나면 소멸된다.

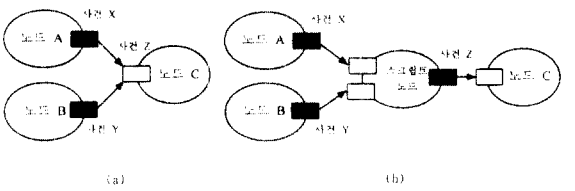
4. 토의

VRML을 사용하여 시뮬레이션을 수행할 수 있게 되면 시뮬레이션 진행상태를 3차원으로 손쉽게 표현할 수 있어서 교육 또는 홈쇼핑등과 같이 이해가 중요한 시뮬레이션에 있어서 많은 도움을 준다. 또한 인터넷을 사용하여 원격지에서의 시뮬레이션 수행이 손쉬워 시뮬레이션의 공유가 용이하며 다자간 분산 시뮬레이션으로의 확장이 용이하다. 시뮬레이션 기술상으로 볼 때, 기존 시뮬레이션에서의 그래픽 인터페이스는 대부분 내부상태를 보여주는데 그치고 있으나 VRML에서는 그래픽 인터페이스상의 각종 사건들이 상호작용 하면서 내부상태를 만들어 가도록 지원할 수 있어서 시뮬레이션 모형화와 관리에 직관적인 일치성을 제공한다.

앞장의 생산 시스템 모형을 보면 시간센서, 사건, 스크립트, 라우트가 시스템의 동적인 특성을 모형화하는데 반복적으로 사용됨을 알 수 있는데 프로토타입노드를 사용하여 새로운 노드를 형성하면 모형화 효율성을 개선할 수 있다. 예를 들면 시간센서와 스크립트를 묶어서 자발적으로 시작하여 통계적인 특성을 보이는 노드를 형성하면 매우 유용할 것이다. 또한 일양분포, 정규분포, 기하분포등 다양한 확률분포를 따르는 사건발생을 다루기 위한 노드들을 체계적으로 제공하는 것이 필요하다.

사건관리 측면에서 보면 분산된 사건들이 동일 시간에 발생할 경우 우선순위를 조정하기 어려운 점이 있다. 이러한 동시발생 사건간 우선순위는 구현된 VRML 브라우져마다 다를 수 있으므로 주의가 요망된다. 이러한 모호성을 없애기 위해서는 사용자가 사건에 우선순위(Priority)를 명시적으로 부여하여 동시발생시 VRML브라우져가 이를 바탕으로 사건처리를 수행하는 것이 필요하나 현재 VRML2.0에서는 지원하지 않으므로 향후 이런 방향에서의 개선이 모색되어야 한다.

하나의 출력사건이 하나의 입력사건에 전달되면 해당노드가 작동되므로, 둘 이상의 출력사건이 동시적 또는 순차적으로 발생할 때 입력사건이 작동하도록 하기 위해서는 중간에 스크립트 노드를 두어 사건발생에 관한 내용을 필드에 저장하여 관리하고 각각의 함수에서 조건을 점검하여 만족하면 출력사건을 입력사건으로 내보내는 것이 필요하다. 이것을 그림으로 도식하면 〈그림 8〉과 같다. 이 그림의 (a)와 같이 라우트가 설정된 경우 사건 X 또는 사건 Y가 발생하면 입력사건 Z로 전달되어 각각 반응한다. 사건 X와 사건 Y가 (임의의 순서로) 모두 발생할 경우에만 입력사건 Z로 전달되어 반응해야 한다면 이 그림의 (b)와 같이 중간에 스크립트노드를 두어서 사건X와 사건Y가 발생할 때마다 스크립트를 실행하여 상태를 관리하고 조건을 만족할 경우 입력사건 Z로 사건발생을 전달하여 반응하여야 한다. 물론 노드C가 스크립트 노드일 경우에는 직접 이러한 상태관리를 수행해도 된다.



〈그림 8〉 복수의 사건 연계

또한 사건간 연결은 라우트에 의하여 모형화되고 의사결정부분은 스크립트로 처리되도록 하고 있는데 이러한 의사결정 법칙을 라우트처럼 외형적으로 모형화하는 것이 동적인 모형화 과정에서 중요하다. 이것은 VRML 2.0을 벗어나는 것으로 이후의 확장에서 반드시 반영되어야 할 것이다.

5. 결 론

본 논문은 3차원 표준 가상공간 기술언어인 VRML 2.0을 사용하여 3차원 시뮬레이션 모형화의 가능성을 점검하고 개선점을 제시하기 위한 것으로서 먼저 시뮬레이션에 관련된 노드들을 살펴보고, 간단한 생산 시스템을 모형화하여 그 가능성을 알아보고 개선점을 토론하였다. 먼저 VRML을 사용하여 생산 시스템의 모

형화가 가능하였으나 모형기술 내용이 많아져 3차원 모형화 지원도구의 도움이 절대적으로 필요하였다. 다양한 통계분포를 따르는 사건처리를 위해 시간센서 스크립트, 사건등을 효과적으로 결합한 새로운 프로토타입 노드의 생성이 필요하다. 동적인 특성은 라우트에 의한 사건 연결과 스크립트에 의한 의사결정부분으로 모형화되는데 의사결정 법칙을 라우트처럼 일반적으로 모형화하는 것이 모형의 직관성을 높여주고 효율성을 증가시킬 것으로 판단된다.

이상의 개선점이 반영된다면 음악, 이미지, 비디오, 그래픽 등이 결합된 3차원 가상공간 개발/공유 언어로서 뿐 아니라 3차원 대화형 시뮬레이션 환경으로 자리매김 할 수 있을 것이다. 또한 다수의 사용자를 동시에 지원하기 위한 분산 시뮬레이션 도구로서의 발전이 기대된다[2].

참고문헌

[1] Bell, G., A. Parisi, M. Pesce, "The Virtual Reality Modeling Language Version 1.0 Specification," http://wintermute.gmd.de:8000/vrml/spec/vrml10-3.html, May 26, 1995.

[2] Broll, W. and D. England, "Bringing Worlds Together: Adding Multi-User Support to VRML," *Proceedings of the 1995 Symposium on the VRML*, Dec. 1995.

[3] Hamilton, J.A., G.R. Ratterree, P.C. Brutch and U.W. Pooch, "Public Domain Tools for Modeling and Simulating Computer Networks," *Simulation*, September 1996, pp. 161-169.

[4] ISO/IEC CD 14772, "The Virtual Reality Modeling Language Specification Version 2.0," http://webspace.sgi.com/moving-worlds/spec/index.html, August 4, 1996.

[5] Kamigaki, T. and N. Nakamura, "An Object-Oriented Visual Model-Building and Simulation System for FMS Control," *Simulation*, December 1996, pp. 375-385.

[6] Tarr, R.W. and J.W. Jacobs, "Distributed Interactive Simulation (DIS)," *Proceedings of the 1994 Summer Computer Simulation Conference*, July 1994.

● 저자소개 ●

김형도

1988년	1985년 서울대학교 산업공학과 학사
1987년	KAIST 경영과학과 석사
1992년	KAIST 경영과학과 박사
1993년~현재	(주)데이콤 종합연구소 멀티미디어 연구팀 재직
관심분야	의사결정지원, MIS, Systems Modeling & Analysis, 시스템 개발 방법론, 멀티미디어 서비스 등