VRML에서 복잡한 루트에 대한 이벤트 처리

이성태, 김이선, 김판구, 이윤배, *염순자
조선대학교 대학원 전자계산학과
*호주 타스마니아 대학교 계산학부
e-mail : adaondal@hanmail.net

Event Processing for Complicated Routes in VRML

Sung-Tae Lee, Yi-Sun Kim, Pan-Koo Kim, Yun-Bae Lee, Soon-Ja Yeom*
Dept. of Computer Science, Graduated School, Chosun University
* School of Computing, University of Tasmania, Australia

요 약

VRML(Virtual Reality Modeling Language)에서 객체를 구성하는 자는 VRML 노드들 사이에 놓인 이벤트들의 루트를 맡힘으로써 가상 세계에서 동적으로 상태가 변화하는 것을 허용한다. 그런데 VRML에 정의된 개념적인 실행 모델에서 이벤트는 즉각적으로 노드들의 예정지로 전달되어야만 한다. 그러나 다양한 분야에서 동시에 일어나는 이벤트나 노드들 사이에서의 주기적 의존성과 같은 복잡한 요구를 수반하는 노드들 사이의 관계에서는 브라우저 실행을 어렵게 하는 문제가 있다. 따라서 본 논문에서는 VRML 부라우저에서 이벤트를 처리하는 기법과 복잡한 형태 루트들에 대한 이벤트 처리 방법을 제시하고 시뮬레이션을 통해 타당성을 검증한다.

1. 서론

VRML의 가장 중요한 특징 중 하나는 객체가 동적으로 변화하는 장면 그래프의 상태에 대해 이벤트의 루트를 제공하는 능력이라고 할 수 있다. 객체가 어떤 대상과 상호작용을 하고자 할 때, 또는 어떤 특정한 시간이 경과할 때 열거될 수 있는 최초의 이벤트는 루트 연결을 통해 다음 노드들로 전송된다. 이벤트를 받는 노드들은 지시 받은 그래프에 의해 표현되는데, 이상적인 이벤트 처리를 위해서 가상세계에서 노드들을 실행 할 경우 다음과 같은 사항들을 우선적으로 고려해야 한다.

① 노드에 의한 이벤트 발생의 불예측성
② 다중 이벤트 투입
③ 루프 방지
④ 주기적 의존성
⑤ 직접 출력

위와 같은 사항들을 고려하여 VRML 브라우저에 의해 사용하기 위한 이벤트 처리 방법을 표현해야 한다. 그리고 어떤 브라우저가 루트 그래프에 걸친 이벤트들을 처리하기 위해 취하는 기본적 접근 방법들이 있는데 이벤트 유출량의 방향에 따라 이벤트를 보급시키는 것으로 이것은 ROUTE에 의해 정의된다.

본 논문에서의 제안하는 이벤트 처리 방법은 먼저 이벤트를 발생시키고, 다음 노드의 순서를 결정하고, 이벤트들을 그 순서대로 발생시킨다. 이러한 접근을 위해 복잡한 루트 연결과 함께 효과적으로 사용될 수 있는 몇 가지의 근본적인 그래프 알고리즘을 제안하고자 한다.

2. 노드의 이벤트 설계

VRML에서 노드는 기본적 구성 요소로써 각각의 노드는 특정한 수의 필드들을 가지고 있는데 그것은 특징이나 속성을 표현한다.

이벤트들을 받을 수 있는 분야를 이벤트 투입(eventIns)이라고 하고, 반면에 이벤트들을 보낼 수 있는 분야를 이벤트 출력(eventOuts)이라고 한다. 어떤 루트는 한 이벤트 출력(eventOut) 필드에서 이벤트 입력(eventIn) 필드로 이어지고 그 필드 유형 둘 모두 정확히 매치 되면 그 값은 다른 이벤트와 같은 루트를 통해 보내진다. 그것을 받는 노드에 의해 값이 평가되고 다른 이벤트들을 구성하여 전송한다. 이상적인 이벤트를 전송하기 위한 처리하기 위한 방법은 다음과 같다.

첫 번째는 노드의 이벤트 발생을 예측하지 못하는 경우로 노드에 의해 받는 이벤트들은 그 노드의 상태를 변화시키거나 혹은 부가적 이벤트를 생성시키기 위해서 평가가 된다.

두 번째는 다중 이벤트를 투입하는 것으로 모든 이벤트들은 즉시 전송되는데, 한 노드가 같은 이벤트 열 내에서 동시에 다른 종류의 이벤트들을 받는 것을 허용한다. 예를 들면, 세 개의 A, B, C 노드가 있다고 가정했을 때, 루트 연결들은 다음과 같이 만들어진다.

ROUTE A.out1 TO C.in1
ROUTE A.out2 TO B.in
ROUTE B.out TO C.in2.

만약 노드 A가 동시에 eventOut out1과 eventOut out2에서 이벤트를 발생시킨다면 노드 C는 eventIn1 in1과 eventIn in2에서 다중 이벤트를 받게 된다. 이 경우 노드 C는 다중 eventIns로부터의 값에 기초를 두어 만들어wu야 한다. 따라서 노드 B는 노드 C보다 우수하다고 평가되어야 한다.
그림 1은 이러한 노드의 이벤트 발생 모형을 그래프로 표현한 것이다.

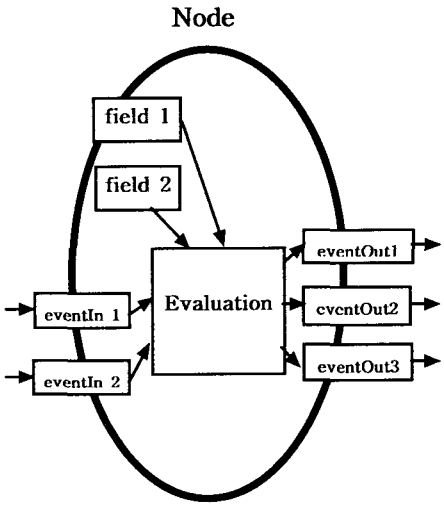

그림 1. 노드의 이벤트 발생 모형

세 번째는 루프 방지로 루프의 발생을 위한 조건을 정의하고 브라우저가 어떻게 루프를 깰 수 있는지를 설명한다. 브라우저는 필드들이 똑 같은 시간을 가지고 이벤트를 보내고 있지 않다는 것을 확실히 하기 위해 필드들을 점검함으로써 루프를 보호한다. 이것은 한 이벤트 루프가 존재하거나 루트 그래프 위상으로부터 확인여부에 관계없이 한 노드가 그것이 받는 이벤트들에 반응하기 때문이다. 그리고 브라우저는 운영 시간에 루프를 탐지하도록 요구를 받게 된다.

네 번째는 주기적 의존성으로서 루트들은 어떤 주기적 의존성이 노드들 사이에서 발생되는 방식으로 연결한다.

루프 보호 규칙 노드는 다른 이벤트 출력(eventOut)을 통해 다시 평가받지 못하게 한다. 예를 들면, 노드 A, B, C는 다음과 같은 루트 연결을 갖는다.

ROUTE A.out1 TO C.in
ROUTE A.out2 TO B.in
ROUTE B.out TO A.in2.

만약 어떤 이벤트가 이러한 순서로 처리된다면 진행 순서는 다음과 같다.
A.in1 -> A.out2 -> B.in -> B.out -> A.in2 -> A.out1 -> C.in -> C.out

이 경우 노드 A와 노드 B 사이에 주기적 의존성이 발생한다. 왜냐하면 노드 A에 대한 평가는 노드 B의 평가에 의존하고 노드 B의 평가는 노드 A에 의존하기 때문이다. 그러나,
주기적 의존성은 다중 이벤트 입력(eventIns)과 일치하지 않는다. 이전의 예에서 노드 A는 eventIn2.에서 한 이벤트를 받기 전에 eventOut Out2를 통해 이벤트를 보내는 것이 필요하다. 따라서 다중 이벤트 입력(eventIns)은 불가능하다.

다섯 번째는 직접 출력하는 방식으로 한 스크립트 노드는 그것의 직접 출력 필드가 참(True)일 때 직접 출력을 하게 된다. 그때 그것은 다른 노드들에 접근할 수 있고 루트 연결을 만들거나 파괴 할 수도 있으며 루트를 생성하지 않고 그들의 이벤트 입력(eventIns)으로 직접 이벤트들을 보낼 수 있다. 그러나, 직접 출력을 포함하는 이벤트를 처리하는 것은 어렵다. 왜냐하면 이벤트 전파 스테이지에서 동적인 이벤트 열 변경을 초래할 수 있기 때문이다. 그러한 변경은 예기치 못한 것들로써 미리 이벤트 평가 순서를 결정하는 것이 불가능하다. 따라서 특별한 취급들이 직접 출력을 하기 위해서 필요하다.

3. 이벤트 처리 기법

VRML 브라우저에서 공급한 이벤트 처리 기법은 그것들이 이벤트 열을 통해 유연하게 이벤트를 생성한다. 이것은 VRML의 이벤트 수행 디자인과 일치하지 않을 뿐만 아니라 루트 그래프 전반에 일어난 모든 이벤트를 전파시키는데 효율적이다. 그러나 모든 이벤트를 처리하는 것은 비효율적 일 수도 있다. 이벤트 처리의 중요한 목적은 필드들의 가치를 새롭게 하기 위해서이다.

브라우저 실행으로 Movie Texture와 Audio Clip 같은 눈에 보이지 않는 것에 대해 시간을 소비하는 것은 원기둥 중 몸체 부분을 검색해 냄으로써 피할 수 있다. 그러나 모든 이벤트 네트워크가 활동적이 되지 않을 것이라고 예상할 수도 있다. 불필요하게 생기는 이벤트들을 피하는 방법들을 예를 들면, ProximitySensor 노드는 사물들이 어떤 거리에서 보일 때 이벤트가 일어나는 것을 막는데 사용한다. TouchSensor 노드는 보는 사람으로 하여금 자극 될 때까지 이벤트가 일어나는 것을 피하기 위해 사용된다. 결과적으로 만약 이벤트가 처리되고 있다면 아무런 생각 없이도 처리 비용이 높지 않을 것이라는 것을 예상할 수 있다.

처리될 이벤트들을 위해 루트 네트 워크내의 노드들은 평가할 필요가 있다. 사용자가 동시에 한 노드를 평가할 수 있기 때문에 본 논문에서는 루트 연결 의미론과 일치하는 방식으로 노드에 대한 평가를 하지는 않는다. 예를 들면, 특정한 노드는 다른 노드 앞에 평가 받아야한다. 이미 설명한 예에서 노드B는 노드C 후에 평가 되서는 안 된다. 왜냐하면 노드 B에서 발생한 이벤트는 만약 노드 C가 다시 평가받지 않는다면 다루어지지 않을 것이기 때문이다.

이러한 상황을 피하기 위해서, 이벤트 처리는 미리 평가순서를 주의 깊게 결정하여 행하고 그 다음 즉각적인 이벤트 전파를 결정함으로써 행해져야 한다.

3.1 평가 순서 결정

만약 어떤 그래프가 루프나 주기적 의존성을 포함하지 않는다면 그것은 직접적인 주기적 그래프, 데카그램이라고 한다. 루트 그래프가 데카그램이라고 가정하면 지세학적 분류

에 의해 평가 순서를 얻을 수 있다. 그 결과는 모든 노드들이 루트 그래프로부터 모든 이벤트 입력(eventIn)을 받은 후에 평가되는 방법들로 이루어 진다. 그러나 불행히도 루트 그래프들은 일반적으로 데카그램이 아니다. 왜냐하면 루트 그래프들은 데카그램으로 다루기 위해서 또 다른 허용하는 기술을 필요로 하기 때문이다. 강력하게 연결된 구성 요소들은 하나의 세트에 있는 모든 노드들이 상호 접근하기 쉬운 적절성을 가진 직접적인 그래프에 있는 일련의 노드들을 의미한다.

그림 2는 어떻게 루트 그래프에 있는 주기적 의존성이 그래프를 구성 요소로 나눔으로써 제거될 수 있는지를 보여 주고 있다. 노드 A와 노드 B가 주기적 의존성을 갖기 때문에, 그들은 똑같은 구성 요소 내에 있다. 노드 그 자체가 하나의 구성 요소라는 것을 주목하여야 하는데, 노드 C는 구성요소 2의 유일한 요소이다. 따라서 구성 요소 1이 구성 요소 2 전에 평가되어야 한다는 것을 알 수 있다.

평가의 순서는

Node A -> Node B -> Node A -> Node C.

그러나 만약 구성 요소로서 노드들을 나누는 것 없이 지세학적 분류를 적용한다면, 평가 순서는 다음과 같이 또한 유효한 지세학적 순서가 될 것이다.

Node A -> Node C -> Node B -> Node A

이 경우에 하나의 이벤트는 적절히 전파되지 않을 것이다.

평가순서를 얻기 위해서 루트 그래프는 먼저 강력하게 연결된 구성 요소를 탐지함으로써 분류시킬 수 있다. 먼저 구성 요소 사이의 파생된 일련의 루트 연결은 하나의 데카그램으로 표현될 수 있다. 다음 구성요소들이 처리된 순서는 지세학적 분류에 의해 얻어질 수 있으며 마지막으로 구성 요소내의 평가 순서가 결정되게 된다.

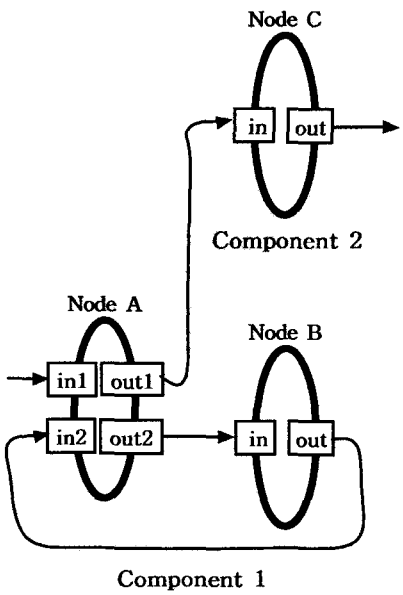

그림 2. Route Graph에 있는 주기적인 종속 관계의 Node의 구성요소

(1) 데카그램 생성

평가 순서를 결정하는 첫 번째 단계는 강하게 연결된 구성요소로 루트 그래프를 나누는 것이다. 구성요소에 의해 형성된 루트 그래프가 데카그램이다. 루트 그래프에서 강하게 연결된 구성요소들은 1972년에 R. E. Tarjan 에 의해 제안된 것으로 알고리즘은 심도 있는 탐색의 변형이기 때문에 계산 비용은 루트그래프에 있는 루트들과 많은 노드들 사이에 1차적으로 비례한다.

(2) 성분 분류

루트 그래프에서 강하게 연결된 성분들은 강하게 연결된 성분에 의해 제한된 알고리즘에 의해 얻어진다. 지세학적 분류 알고리즘은 또한 심도있는 탐색의 변형이라고 할 수 있다. 계산 비용은 성분들을 연결한 수많은 루트들과 수많은 성분들에 1차적으로 비례한다.

(3) 한 노드 성분 내의 분류

평가 순서를 결정하는 마지막 단계는 성분이 평가될 때 순서가 올바른 결과를 산출하도록 하기 위해 한 성분 내의 노드들은 분류하는 것이다. 앞에서 설명한 것처럼, 하나의 노드는 이벤트 발생에 관해서는 예측할 수 없다. 따라서 일반적으로 많은 가능한 평가 순서들이 있다. 그림 3 은 그림 2에 있는 예보다 더 복잡한 경우를 보여준다. 여기에는 두 가지의 노드 평가 순서가 있다. 대부분의 경우에 노드 A는 eventIn in1에서 이벤트를 받을 때 eventOut out2와 eventOut out3으로부터 이벤트를 발생시킨다. 다음 노드 B는 *eventOut out1*과 *eventOut out*에서 이벤트들로부터 이벤트를 발생시킨다. 마지막으로 노드 A는 eventOut out1으로부터 이벤트를 발생시킨다. 성분 1 내에서의 노드 평가 순서는

A -> B -> A

그러나, 어떤 경우에 노드 A는 단지 eventOut out2로부터 이벤트를 발생시킬 것이다. 이러한 경우에 예상되는 이벤트 흐름은 다음과 같다.

A.out2 -> B.in1 -> B.out1 -> A.in2 -> A.out3 -> B.in2 -> B.out2 -> A.in3 -> A.out1 -> C.in

노드 평가 순서는

A -> B -> A -> B -> A

일반적으로 그 순서가 경우마다 다르기 때문에 노드를 평가하기 전에 하나의 순서를 결정하는 것은 불가능하다. 이전의 경우에서 한 노드 성분 내에서 노드의 지세학적 분류는 다음과 같다.

A -> B.

이러한 과정은 노드 A와 B에서 일어난 이벤트들에 의존하여 두 세 번 반복될 것이다.

3.2 이벤트 처리 결과

만약 루트 그래프가 강하게 연결된 성분으로 분류된다면 이벤트는 그것의 성분들을 평가함으로써 조직화된 방법에 따라 전송된다.

성분들과 노드들이 평가되는 순서는 이미 결정된 후 노드가 평가된다. 무한 루프는 두 가지의 이벤트 처리 전략에 의해 전송되는데, 하나는 이벤트 전략에 있어서 루프를 피하기 위해 같은 규칙을 적용하는 것이다. 노드는 똑같은 이벤트 출력(*eventOut*)을 통해 한 이벤트 이상을 보낼 수 없다. 두 번째 전략은 만약 노드가 어떠한 이벤트 입력(eventIns)를 받

지 않는다면 노드 평가를 빠뜨리는 것이다.

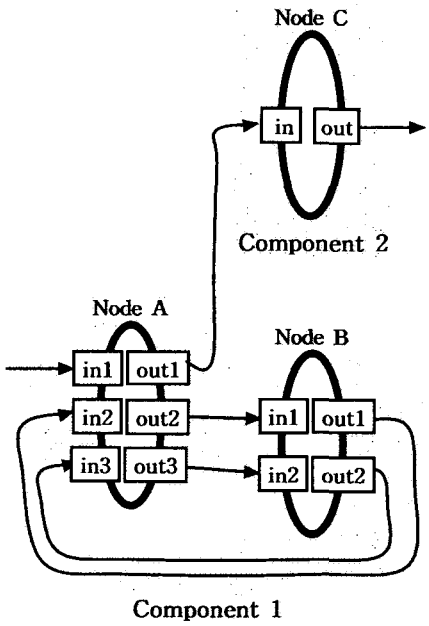

그림 3. 복잡한 Node들간의 종속 관계의 예

4. 실험

성과를 개선시키기 위해 평가 순서는 루트 그래프의 첫 번째 평가하고, 다음 변화의 상태가 그래프로 만들어 질 때까지 반복적으로 사용된다.

평가 순서는 브라우저가 실제로 이벤트를 평가하기 전에 발생된다. 순서는 노드가 이벤트를 받을 때 모든 이벤트 출력(eventOut)을 생성시킨다는 가정 하에서 만들어진다. 그러나 이 경우 순서대로 포함된 불필요한 노드들이 생성될 수 있다. 불필요한 평가를 최소화하기 위해서, 모든 노드들은 실제로 이벤트를 받는지를 알기 위해서 변경 플랙을도록 하였다..

그리고 루트 그래프 평가가 줄을 서서 저장되는 동안 만들어진 동적으로 변형된 루트연결을 다루지 않기 때문에 직접 출력이 동적으로 루트 네트워크를 변경하고 그리고 나서 그 평가가 끝난 후에 처리되도록 하였다. 유사한 방법으로 ,·이벤트들은 직접적으로 스크립트가 다른 이벤트 열로써 취급되도록 다른 노드들에게 전송하였다.

시간 특정뿐만 아니라, 최초의 이벤트에 있어서의 독특한 숫자인 이벤트 번호는 루트 그래프 이상으로 전송한다. 시간 스탬프를 사용하는 것 대신에, 그 번호는 루프 탐지를 위해 사용된다. 왜냐하면 운영 체제로부터 얻은 시간 정보는 다른 이벤트 전파 사이에서 구별 할 만큼 정확하지 않기 때문이다.

실험결과, 이벤트 포인터가 가리키는 것에 따라 이벤트 처리가 가능했고 표본의 필드는 가치들을 저장하기 위해 사용되었다. 루트 네트워크가 표본을 포함할 때 평가 순서는 표본 내에 노드들을 포함하는 방식으로 결정하였다.

5. 결론

본 논문에서 제안된 방법은 사용자가 복잡한 루트 연결을 만들도록 허용하는 실행 가능한 이벤트 처리 방법을 제공한다. 먼저 평가 순서를 결정하고 나서 이벤트를 전송시킴으로써 적절히 다중 이벤트의 투입과 주기적 의존성을 포함하는 루트 연결을 다룬다.

문제점은 루트 연결의 제거나 증가, 노드를 만들거나 삭제하는 것과 같은 루트 그래프의 수정이 필요할 때 어떻게 이벤트가 전송되어야 하는지였는데 이의 해결 방법은 이벤트 열 평가에 포함될 수 없는 일을 처리하는 것에 대한 순서를 정의하도록 한 것이다. 결과적으로, 루트 그래프는 그래프 평가동안 변경되지 않는다 왜냐하면 루트 연결의 증가와 제거는 그 그래프가 평가된 후에 이루어지기 때문이다. 이것은 제안한 이벤트 처리 실행과 일치한다.

앞에서 언급했듯이, 본 논문에서 제안한 방법은 이벤트 전파의 시작보다 우선인 노드 평가의 순서를 결정하는 것이다. 따라서 루트 그래프의 변화들은 그래프 평가동안 다루어지지 않을 것이다. 그 방법은 그래프 평가가 끝난 후에 그러한 요구사항들을 다룰 수 있다. 이것은 루트 그래프의 변경을 다루는 해결 방법과 일치한다.

모든 이벤트들이 노드 평가로써 사용되거나 혹은 한 개의 실행에서 다른 실행으로 변화하지 않는다. 브라우저가 모든 이벤트를 처리하기 위해 필요하다는 것이다.

따라서 본 논문에서 제안한 기법은 루트 그래프에 있는 노드들 사이에서의 의존성을 해결할 수 있으며 노드가 평가되기 전에 모든 이벤트들이 전송될 수 있다는 결과를 얻었다.

참 고 문 헌

[1] "The Virtual Reality Modeling Language Specification", ISO/IEC DIS 14772-1 APRIL 4, 1997. Available as "http://www.vrml.org/Specification/VRML97/DIS/".

[2] "VRML Script Working Group Clarifications," Availableas "http://www.vlc.com.au/~justin/vrml/script-wg/".

[3] Robert Tarjan, "Depth-First Search And Linear Graph Algorithms," *SIAM Journal on Computing*, Vol. 1, No. 2, June 1997, pp. 146~160

[4] Daniel J. Woods, Alan Norton, and Gavin Bell, "Wired For Speed: Efficient Routes In VRML ." in *proceedings of VRML 1997*, pp.133~138.

[5 W.W. Armstrong, M. Green, R. Lake, "Near-realtime control of human figure models", IEEE Comput. Graphics Apl. Vol.7, No.6 1987, pp.52~61.

[6] M.F. Cohen," Interactive spacetime control for animation, Computer Graphics Proceeding", Annual Conference Series (SIGGRPH 92 Proceedings), vol. 26, 1992, pp. 293~302.