3차원 가상공간의 저작을 위한 VRML 파서와 Scene Graph 생성기의 설계 및 구현

○

전영훈*, 허도영*, 이종석**, 황대훈*

*경원대학교 전자계산학과

**한국통신기술(주)

Design and Implementation of the VRML Parser and Scene Graph Generator for Authoring of 3D Virtual Space

○

Young-Hoon Jun*, Do-Yeong Heo*, Jong-Suk Lee**, Dae-Hoon Hwang*

*Dept. of Computer Science, Kyungwon Univ.

**Korea Telecom International

요 약

VRML 파서는 VRML(.wrl) 파일을 읽기 위하여 반드시 필요한 부분으로서, 이미 공개되어 있는 것으로는 C/C++ 코드의 QvLib1.0과 Yacc & Lex로 개발한 VRML2.0 파서가 있다. 그러나 QvLib1.0의 경우 VRML2.0 파일에 적용할 수 없으며, Yacc & Lex로 개발한 파서는 생성된 소스 코드를 수정하기 어려울 뿐만 아니라 이를 수정하기 위해서는 자동 생성기인 Yacc & Lex의 사용 방법을 잘 알고 있어야 한다.

이에 본 논문에서는 C/C++ 코드의 VRML2.0 파서를 설계 및 구현하고 파싱의 결과로 생성된 parse tree를 이용하여 VRML 파일을 scene graph로 표현할 수 있도록 하는 Scene Graph 생성기의 구현과정을 소개하였다. 또한 본 연구의 파서와 Scene Graph 생성기를 기 개발된 VRML 저작도구에 적용 및 테스트하였다.

1. 서 론

WWW(웹)의 등장으로 인하여 인터넷 상에서 다양한 멀티미디어 정보가 제공되고 있으나 이는 2차원 정보만을 기반으로 하고 있다. 이에 인터넷 상에서 3차원 입체화면을 통하여 시각적 효과를 높이고 가상현실이 제공하는 몰입감과 임장감을 체험할 수 있도록 지원하기 위한 표준언어인 VRML이 등장하였다.

VRML(Virtual Reality Modeling Language)은 웹에서 3차원 가상공간과 객체들을 기술하기 위한 표준언어이다. 이러한 VRML은 인터넷상에서 가상현실이나 작품 등과 같은 복잡한 장면(scene)들을 3차원으로 생성하기 위해 사용된다. 초기의 VRML 1.0이 비교적 단순한 애니메이션만 지원하는 반면, 현재 공개된 VRML 2.0은 복잡한 3차원 애니메이션, 시뮬레이션, Java와 JavaScript 등을 지원한다.

VRML 2.0의 경우 보통의 사용자가 HTML과 같이 쉽게 사용하기에는 그렇게 쉽지 않다. 이에 VRML로 기술된 .wrl 파일을 쉽게 제작할 수 있도록 다양한 저작도구(authoring tool)와 브라우저들이 개발되었다. 이들 저작도구와 브라우저를 구성하는 중요 모듈로는 .wrl 파일을 읽고 해석하기 위한 VRML 파서 부분과 VRML 요소들을 3D API (Application Programming Interface)와 매핑하여 렌더링하는 부분 등이 있다.

3D API는 기존의 OpenGL이나 DirectX를 이용할 수 있으며, VRML 파서는 쉐어웨어로 공개되어 있는 QvLib1.0과 Yacc&Lex를 이용한 VRML 2.0 파서 등을 이용할 수 있다. 그러나 Silicon Graphics사의 QvLib1.0은 C++ 코드로 작성되어 있어 개발자들이 이를 이용하여 자신의 저작도구나 브라우저에 맞게 부분 수정하여 사용할 수 있으나 현재 VRML 2.0을 지원하지 못한다. 또한 Yacc&Lex를 이용하여 작성된 파서는 VRML 2.0을 지원하나 자동생성된 코드를 개발자가 쉽게 수정하여 사용하기 어려우며

수정시 Yacc과 Lex의 사용 방법을 잘 알고 있어야 한다.

이에 본 논문에서는 VRML 2.0 파서를 C++ 코드로 작성하고 이와 연계하여 3차원 가상세계의 장면들을 계층적 구조(hierarchical structure)로 표현하는 Scene Graph 생성기를 설계 및 구현하였다.

2. 관련 연구

2.1 VRML 개요

VRML은 WWW에 관한 제1회 국제회의에서 처음 출발한 뒤 발전을 거듭하여 지금은 VRML 2.0이 등장하게 되었다. VRML1.0은 상대적으로 단순한 애니메이션만 지원하는 반면, 현재 개발된 VRML2.0은 복잡한 3차원 애니메이션, 시뮬레이션, Java 및 JavaScript 등을 지원한다.

다음절에 소개될 내용은 VRML의 구조적 특징과 기능에 대하여 기술한 뒤 이를 어떻게 Scene Graph로 적용 및 표현되는지 기술한다.

2.2 VRML의 구성

VRML 파일을 구성하는 요소는 크게 노드(node)와 필드(field)로 나눌 수 있으며, 이외에도 Event와 Script 및 키워드로 구성된다. (그림 1)은 노드와 필드로 구성된 VRML 파일의 예를 나타낸 것이다.

(1) 노드와 필드

VRML 2.0에서는 다양한 종류의 노드를 지원하고 있는데, 노드는 HTML의 태그(tag)에 비유할 수 있으며 필드는 이러한 노드의 특성을 나타내기 위해 사용된다.

```
#VRML V2.0 utf8          // VRML 헤더
Transform {   // Transform 노드로 Grouping node에 해당
    translation -1 .5 1
    // 변환정보를 가지는 Transform 노드의 필드
    children [
    // Grouping node 또는 차일드 노드를 갖는다.
        shape {
        // 차일드 노드로 Common node에 해당
          geometry Sphere {
          // geometry 필드는 Geometry node를 갖는다.
            radius 0.5
            // Geometry node인 Sphere의 필드로 radius값
              을 나타낸다.
          }
        }
    ]
}
```

(그림 1) VRML 2.0 파일 구조

노드는 기능에 따라 그룹노드(group node), 특수노드(special node), 일반노드(common node), 센서노드(sensor node), 구조노드(geometry node), 구조특성노드(geometry property node), 외양노드(appearance node), Interpolator node, Bindable node로 묶을 수 있으며 필드는 크게 MF(Multiple Field)와 SF(Single Field)로 나눌 수 있으며 이로는 MFString, MFFloat, SFString, SFFloat 등이다.

(2) Event

Event 노드는 VRML 2.0에 새롭게 추가된 필드로 대부분의 노드들이 이벤트를 주고받을 수 있도록 정의되어 있으며, 이는 노드들 사이에 변화된 상태를 서로 주고받기 위해서 사용된다.

(3) Script

Script 노드 역시 VRML 2.0에서 새롭게 추가된 노드이다. 이 노드는 이벤트에 어떤 효과를 주기 위해서 종종 사용되며, 이를 이용하여 새로운 이벤트를 선언할 수도 있다.

(4) 키워드

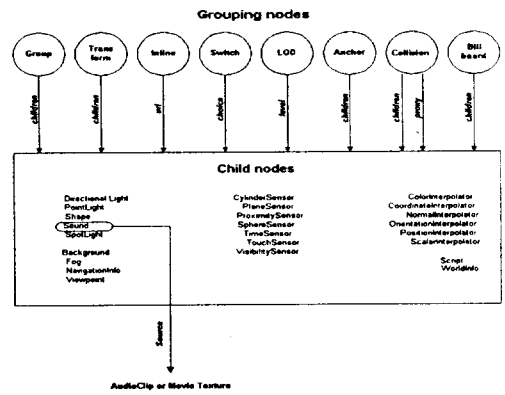
VRML 2.0에는 DEF, ROUTE, USE와 같은 키워드를 사용할 수 있으며 DEF는 노드의 이름을 작성자가 임의로 정의, ROUTE는 노드와 노드간의 이벤트 경로를 설정, 마지막으로 USE는 이미 앞에서 DEF로 정의된 노드를 다시 사용할 경우 이를 이용하면 된다.

2.3 VRML의 구조

VRML 2.0의 구조는 노드와 노드의 관계가 계층적으로 이루어져 있으며 이를 위해 Grouping node와 Child node로 구분하여 구성된다.

(1) Grouping node의 구조

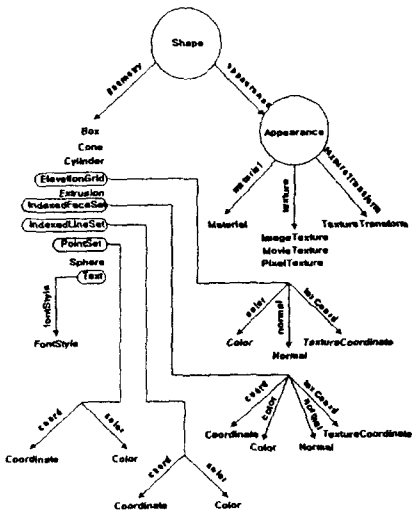
(그림 2)는 Grouping node와 Child node의 관계를 나타낸 것으로 각 Grouping node들은 자신의 필드를 통해 Child node또는 Grouping node들을 포함할 수 있다. 이때 필드의 형(type)은 MFNode로 하나 이상의 노드를 하위에 포함할 수 있다.



(그림 2) Grouping node의 구조

(2) Child node의 구조

(그림 3)은 Child node중에 필드를 통해 단지 하나의 노드만을 하위에 포함하는 구조를 나타낸 것으로 이때 필드의 형은 SFNode이다. (그림 3)에서 Shape 노드의 필드인 geometry 필드를 통해 하위에 하나의 geometry 노드가 올 수 있다.

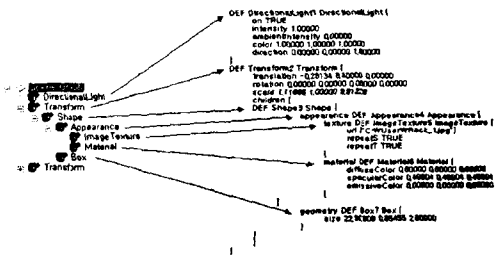


(그림 3) Child node의 구조

2.4 Scene Graph의 구조

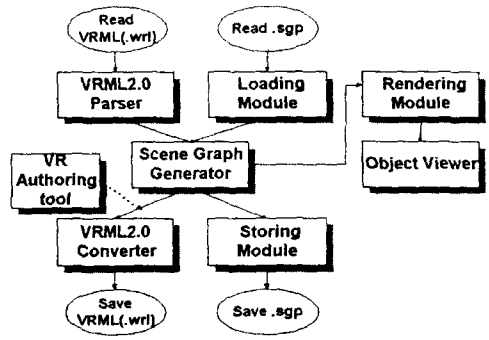
Scene graph는 장면의 구조를 계층적으로 표현하여 사용자에게 보여주며 이는 사용자가 가상세계를 저작할 때 각 노드가 어떤 관계로 구성되어 있는지 알 수 있을 뿐만 아니라 어떻게 이를 구성해야 할지 보여준다.

(그림 4)는 실제 .wrl 파일이 scene graph의 계층구조로 표현된 예를 나타낸 것으로서, 'Scene Graph'로 표시된 부분인 최상단의 루트 노드로부터 출발하여, 위에서 아래로, 좌에서 우로 해당 노드를 찾아 가면서 그 기능을 실행해 간다.



(그림 4) Scene graph의 표현

3. VRML 파서



(그림 5) 전체 시스템의 구성도

(그림 5)의 전체 시스템 구성도는 VRML Parser와 Scene Graph Generator를 중심으로 적용한 저작도구의 내부 구성도 이다.

그림에서 VRML Parser는 .wrl 파일에 대한 문법검사를 수행한 뒤 parse tree를 생성한다. Scene Graph Generator Module는 생성된 parse tree를 이용하여 scene graph를 생성한다. Rendering Module은 scene graph의 노드 정보를 순차적으로 읽어 각 노드에 해당하는 렌더링을 수행한다.

파일의 저장 역시 생성된 scene graph의 노드 정보를 이용하여 저장하게 되며, 이때 저장 가능한 파일 형식은 .sgp 형식의 직렬화(serialize) 방식과 .wrl 형식의 텍스트 방식으로 저장할 수 있다. 여기서 .sgp 파일은 scene graph의 정보를 통해 직렬화 방식으로 저장된 자체 저작도구의 포맷으로서, 한번 생성된 .sgp 파일은 Loading Module을 통해 노드를 생성하고 필드 값들을 직렬화 방식으로 읽어온다.

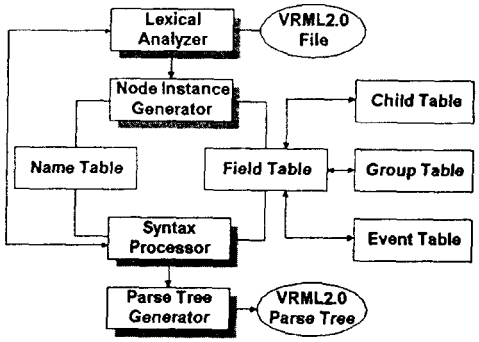
3.1 파서의 구조

VRML Parser는 크게 (그림 6)와 같이 Parser 부분과 Lexical Analyzer로 구성되어 있다.

Lexical Analyzer는 VRML 2.0 파일을 읽어 토큰을 생성한다. Syntax Processor는 생성된 토큰이 노드에 해당하는 토큰인지 필드에 해당하는 토큰인지를 구별하며, 노드인 경우 Node Instance Generator가 해당하는 노드의 인스턴스를 생성하게 된다.

생성된 인스턴스는 자신의 필드 타입을 포함하게 되며 이것이 Field Table이다. Child Table은 다음에 생성된 토큰이 차일드 노드인 경우 이를 보관하기 위해 필요한 테이블이며, 다음에 입력되는 노드가 그룹노드인 경우 Group Table에 해당 노드를 등록한다. Name Table은 'DEF'로 정의된 노드의 이름을 저장하기 위해 필요한 테이블이다. 마지막으로

노드의 필드중에 이벤트에 해당하는 필드인 경우 Event Table에 해당 이벤트를 등록한다.



(그림 6) VRML Parser의 상세도

예를 들어 'USE'혹은 'ROUTE'키워드를 통해 이전에 정의된 노드의 이름을 참조할 경우 해당 노드는 Name Table에 등록되어 있을 것이다. 그렇지 않다면 키워드에 사용된 노드의 이름이 잘못된 것임을 알 수 있다.

이러한 테이블들을 관리하고 토큰을 검사하기 위해 필요한 것이 Syntax Processor이며, 이를 통하여 VRML2.0 파일의 문법검사가 모두 끝나면 마지막으로 VRML2.0 파스 트리가 생성된다.

3.2 파싱 과정

다음은 이상과 같은 일련의 과정을 위해 필요한 토큰 생성, 문법검사 그리고 파스 트리에 대하여 자세히 기술한다.

(1) 토큰 생성

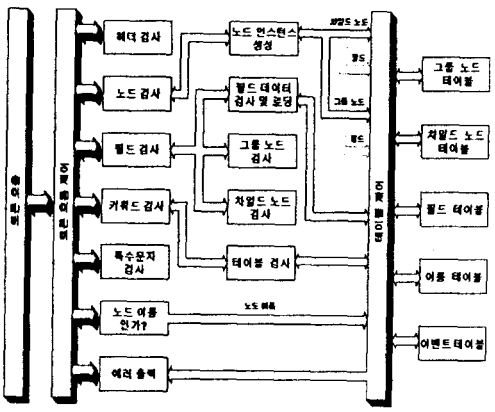
Lexical Analyzer는 VRML 2.0 화일을 읽어 문법의 최소 단위인 토큰으로 분리하여 생성한다. 토큰은 노드, 필드, 키워드, {, }, [,], #, comma 등이며, 각 각의 필드 타입에 따른 필드 데이터가 있을 수 있다. 이를 위하여 Lexical Analyser는 우선 입력 화일에서 헤더를 읽고 공백(space)를 제거한다. 그후 노드 타입과 각 필드 및 필드 데이터를 읽으며 이때 필드 데이터는 공백에 의해 구분되어 진다.

(2) 문법 검사

(그림 7)은 문법검사에 따른 전체 흐름과 테이블간의 관계를 도식화한 것이다. 이는 (그림 6)의 Syntax Processor를 보다 자세히 나타낸 것이다.

입력되는 토큰이 필드, 노드, 키워드 등에 따라 검사되고 자신에 해당하는 테이블에 등록하는 과정을 나타내고 있다. Parser는 문법 검사를 위해 여러 테이블을 생성하게된다.

우선 토큰이 노드 또는 필드인가를 구별하기 위한 방법으로 노드인 경우 다음에 반드시 '{'가 나오게 되며 이때 Node Instance Generator가 해당 노드를 생성한다. 이때 해당 노드가 없으면 토큰의 노드 타입은 잘못된 것이며, 노드가 생성되면 각각의 노드들은 자신의 필드명과 필드 타입을 Field Table에 등록을 한다. 그리고 각 필드에 기본 값을 설정하게 되는데, 이는 문서에 필드가 생략될 수 있기 때문이다. 필드 생략시 필드가 가질 수 있는 기본 값을 설정해 주어야만 랜더링시 오류 없이 작업을 수행할 수 있다.



(그림 7) 문법검사 흐름도

생성된 노드는 자신의 필드 타입과 값을 읽기 위해 토큰 호출모듈에서 Lexical Analyzer에게 토큰을 요구하게 되며 토큰이 필드명인 경우 생성된 노드의 필드명과 일치하는지 검사한다. 이 결과가 일치하면 다음 토큰을 요구하게 되는데, 다음에 입력된 토큰은 해당 필드의 값이며 값의 속성과 개수는 Field Table에 등록된 필드 타입에 의해 체크된다. 체크된 결과가 이상이 없으면 테이블의 해당 필드에 값이 등록되는데, 필드가 차일드 노드인 경우 차일드 노드 테이블에 등록을 한다. 등록된 차일드 노드는 위에서 기술한 것과 같은 행동을 하게된다. 이외에 입력된 필드가 그룹노드인 경우 그룹노드 테이블에 이를 등록하며 역시 위와 같은 행위를 반복하게 된다.

이외에 입력된 토큰이 키워드인 경우 키워드를 검사하여 키워드에 사용된 노드명이나 노드의 이벤트 필드가 해당 테이블에 등록되어 있는지 검사하면 된다. 또한 토큰이 'DEF'로 정의된 노드명인 경우 노드를 해당 테이블에 등록한다. 에러출력은 각 모듈의 문법처리과정에서 발생하는 메시지를 출력하는 부분이다.

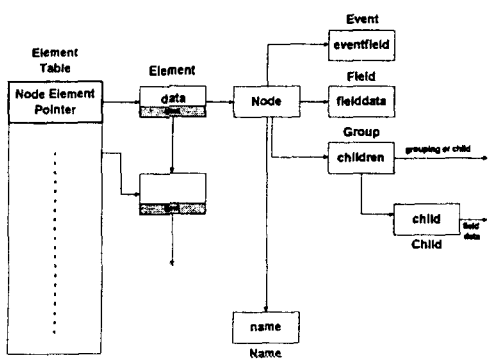
(3) Parse Tree

이와같이 문법검사를 모두 마치고 이상이 없으면 파스 트리를 구성하기 위하여 노드를 논리적인 구조로 스택에 push한다.

(그림 8)은 파스 트리의 구성도를 나타낸 것으로 일반적인 트리의 형태를 나타내고 있다. 이는 입력되는 노드의 형에 따라 트리의 형태가 다를 수 있음을 의미한다.

Parse Tree Generator가 생성된 root 노드를 스택에 pointer 정보로 push작업을 수행하며, 최초의 root 노드부터 차례로 traverse하면서 노드에 대한 포인터와 레벨 정보를 지정한다. 이렇게 생성된 파스 트리는 나중에 scene graph를 생성하기 위한 준비 단계라 할 수 있다.

Traverse는 state 정보를 이용하여 현재 스택의 상태를 파악한 뒤 노드의 pointer 정보와 현재 노드의 레벨 정보를 Element에 할당하고 스택에 추가한다. 이때 새롭게 추가되는 Element는 이전의 Element를 next로 받는다. 이런 방식으로 모든 노드들의 타입에 따라 traverse를 수행하여 파스 트리를 구성한다. 여기서 Element는 root 노드로서 VRML 2.0의 root 노드로 여러 개가 올 수 있으므로 이러한 Element가 여러 개 올 수 있다.



(그림 8) 파스 트리 구성도

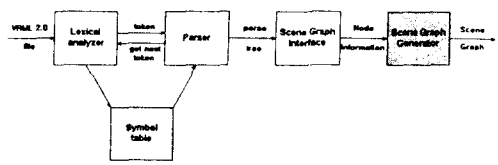
4. Scene Graph 생성기

Scene Graph 생성기는 파서를 통해 문법 검사를 마치고 최종적으로 생성된 파스 트리를 이용하여 TreeView에 계층적으로 .wrl 파일의 구조를 표현하는 기능을 가진다. 이외에 생성기를 통한 .wrl 파일의 편집도 가능하다.

본 절에는 파서와 scene graph 생성기간의 인터페이스를 살펴보고 이를 기반으로 생성기의 구현 과정을 기술한다.

4.1 파서와 생성기 간의 인터페이스

Scene Graph Interface는 Lexical Analyzer와 Parser를 통해 생성된 파스 트리 정보를 받아 스택에 저장된 노드의 정보를 pop하면서 Scene Graph Generator에 해당 노드의 필드 값 및 레벨 정보를 넘겨준다. Scene Graph Generator는 해당 노드를 Scene Graph로 생성하고 렌더링 모듈을 호출하여 해당 노드를 렌더링한다. (그림 9)



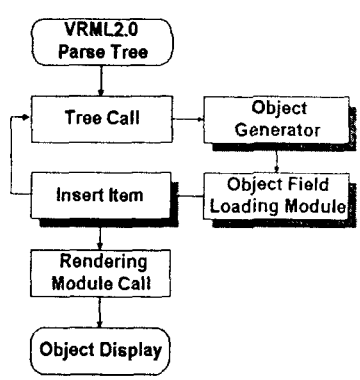
(그림 9) 파서와 생성기간의 인터페이스

4.2 생성기의 구조

(그림 10)은 Parser와 Scene Graph 생성 모듈을 연결하기 위한 부분을 나타낸 것이다. 여기에서 Scene Graph는 생성된 객체들의 계층적 구조와 객체 정보를 담고 있어, 이를 통해 장면의 구조를 알 수 있으며 또한 편집이 가능하다.

Object Generator는 파스 트리 정보를 이용하여 파싱된 노드의 객체를 생성한다. 그리고 생성된 객체는 자신에 해당하는 필드정보를 Object Field Loading Module에서 읽어온다.

이렇게 하나의 노드가 완벽한 정보를 갖게 되면 이 노드의 레벨 정보를 갖고 Scene Graph Generator에 적용된다. 최종적으로 생성된 scene graph는 렌더링 모듈을 호출하며 호출된 렌더링 모듈은 Scene Graph의 정보를 순차적으로 검색하여 각 노드들을 디스플레이한다.



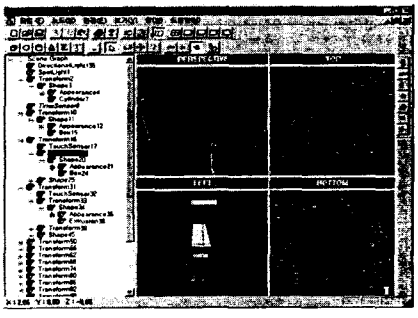
(그림 10) Scene Graph 생성기의 처리과정

4.3 파서와 생성기의 적용

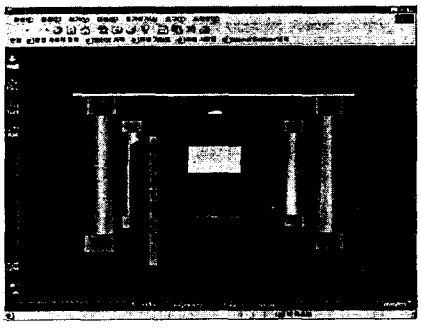
(그림 11)은 본 논문의 파서와 scene graph 생성기를 본 연구실에서 기 개발한 VRML 저작도구에 적용한 결과이다. 본 연구에서 지원하는 파일 형식은 생성된 노드의 정보를 직렬화(serialize)하여 저장하고 로딩하는 파일 형식과 파서를 통한 VRML 2.0 문서의 파일 형식이다.

(그림 11)에서와 같이 왼쪽 창에는 장면을 scene graph로 나타내고 있으며 오른쪽 창에는 이를 렌더링하여 디스플레이한 결과를 나타낸 것이다.

(그림 12)는 (그림 11)을 통하여 생성한 .wrl 파일을 VRML 2.0 브라우저인 WorldView에서 실행시킨 모습으로 저작도구에서 작성된 .wrl 파일을 브라우저를 통해 실행시킬 수 있다.



(그림 11) 저작도구에의 적용



(그림 12) WorldView의 실행

5. 결 론

본 논문에서는 VRML 2.0 파서와 scene graph 생성기를 구현하기 위해 필요한 VRML 2.0 Spec과 문법 그리고 구현에 필요한 요소들과 이들의 관계를 알아보았다.

현재 개발된 파서를 저작도구에 적용하게 되면 기존에 작성되어 있는 VRML 2.0 파일을 저작도구로 로딩할 수 있으므로 재사용성을 높일 수 있으며, 이는 사용자에게 제작 시간을 단축 할 수 있는 편리성을 제공하게 된다.

Scene Graph 생성기는 사용자가 각 객체의 연관관계를 파악하고 있지 않아도 수행이 가능한 브라우저의 경우 적용할 필요가 없지만 저작도구와 같은 응용프로그램에 적용할 경우 가상공간을 구축시 각 객체들 간의 상호 연관관계를 파악할 수 있도록 도와주며 또한 이를 통한 객체의 추가, 삭제 및 수정 기능을 제공하여 가상공간구축을 보다 쉽고 빠르게 할 수 있다.

현재 파서는 Script 노드를 지원하지 못한다. 이를 지원하기 위해서는 별도의 파서가 필요하며 앞으로 script 언어를 지원하는 파서를 추가하여 보다 개선된 VRML 2.0 파서를 개발하고자 한다. 이러한 VRML 2.0 파서는 브라우저를 개발할 경우 핵심 기본 모듈이 될 것이며 이외에 VRML과 관련하여 개발되는 모든 소프트웨어에 유용하게 적용할 수 있을 것이다.

참 고 문 헌

[1] Jed Hartman, Josie Wernecke *"The VRML 2.0 Handbook"*, Addison-Wesley, 1996

[2] 吳 世萬, *"컴파일러 입문"*, 정익사, 1994

[3] Lemay, Murdock, Couch, *"3D Graphics and VRML 2"*, Sams, 1996

[4] Bernie Roehl, Justin Couch, Tim Rohaly, Cindy Reed-Ballreich, Geoff Brown *"Late Night VRML 2.0 with Java"*, Ziff-Davis, 1997

[5] Alfred V.Aho, Ravi Sethi, Jeffrey D. Ullman, *"Compilers Principles, Techniques, and Tools"*, Addison-Wesley, 1986

[6] Rikk Carey, Gavin Bell, *"The Annotated VRML 2.0 Reference Book"*, Addison-Wesley, 1997

[7] Thomas Pittman, James Petters, *"The Art of Compiler Design : Theory and Practice"*, Prentice-Hall, 1992

[8] Martin McCarthy, Alligator Descartes, *"Reality Architecture: Building 3D Worlds In Java and VRML"*, Prentice-Hall, 1997

[9] Thomas Pittman, James Petters, *"The Art of Compiler Design : Theory and Practice"*, Prentice-Hall, 1992

[10] Rory O'Neill, Eden Muir, *"Web Developer. Com Guide to Creating 3d Worlds"*, Wiley & Sons, 1996