VRML 렌더링을 위한 WRL 분석기 구현

김세현[+], 류승택[+], 최정단[++], 윤경현[+]
(+) 중앙대학교 첨단영상전문대학원 영상공학과 :
{astrix, bluelancer, khyoon}@cglab.cse.cau.ac.kr
(++) 한국 전자통신연구원 : jdchoi@etri.re.kr

WRL loader implementation for VRML rendering

Sehyun Kim[O], Seungtaek Ryoo[+], Jungdan Choi[++], Kyunghyun Yoon[+]
+ : Graduate School of Advanced Imaging Science, Multimedia, and Film, Chung-ang University
++ : Electronics and Telecommunications Research Institute

요 약

VRML 데이터를 지원하는 3차원 기반의 응용프로그램 제작을 위하여 VRML 데이터 파일인 WRL을 분석하고, WRL을 읽어들일 수 있는 파서(parser)를 제작하였다. 제작된 파서는 VRML의 구조를 계층적으로 보여주는 프로그램과, 읽어들인 데이터를 바탕으로 가상공간을 네비게이션(navigation)하는 프로그램을 통해 검증하였다. 본 논문에서는 응용프로그램의 요구에 수준에 따라 WRL의 정보를 효율적으로 분석하는 방법과, 읽어들인 정보를 가공하여 렌더링에 적용하는 방법을 설명하였다.

1. 서론

웹 환경의 비중이 높아지면서 VRML로 작성된 모델링 데이터의 수가 기하급수적으로 늘어나고 있다. 비단 웹 서비스뿐만 아니라 웹과 자원을 공유하는 시스템의 설계에 있어서 VRML의 지원 가능 여부는 중요하다. 모델링 능력이 없는 개발자들이 손쉽게 VRML 자료를 사용하기 위해서도 VRML 데이터인 WRL을 분석하여 사용하는 기능은 필요하다. 그러나 VRML 데이터는 다른 파일과 달리 문법의 자유도가 높기 때문에 분석이 쉽지 않다. 본 논문에서는 WRL을 효과적으로 분석할 수 있는 WRL 파싱(parsing)과 파싱된 내용을 메모리에 적재시키는 제반 과정을 실제 시스템 구현 과정을 통해 설명하고자 한다.

2. 관련연구

VRML은 웹 환경을 기반으로 하여 다양한 플랫폼에서 동작하여야 하기 때문에 이진(binary) 데이터를 사용하지 않고 아스키(ASCII) 데이터를 사용한다. 아스키 데이터는 일반적으로 같은 양의 정보를 표현하기 위하여 이진 데이터보다 많은 용량을 사용해야 하지만, 사람이 직접 의미를 이해할 수 있고 쉽게 다룰 수 있다는 장점을 가진다. VRML은 일반 텍스트 에디터를 통해 적성이 가능하며, 저작 도구를 통해 만들어진 데이터 역시 텍스트 에디터를 통해 편집이 가능하다.

VRML에서 하나의 가상세계를 장면(Scene)이라고 한다. 장면은 다양한 형태를 가진 복수의 노드로 구성되고, 여러 개의 노드가 모여 하나의 객체를 구성하게 된다. 이런 노드들은 장면 그래프(Scene graph)라고 하는 계층적 구조 속에서 조정된다. 장면 그래프가 노드를 위한 순서를 정의하는 것이다. 따라서, 장면의 앞에 있는 노드가 뒤에 나타나는 노드에 영향을 미칠 수도 있다. VRML에는 60개의 노드 형태가 있으며 노드와 노드를 이었을 때 이를 그룹핑 노드(Grouping node)라고 하고, 위쪽의 노드를 부모 노드(Parent node) 그리고 아래쪽 노드를 자식 노드(Child node)라고 한다. 노드들이 어떤 기능을 수행하는 부분이라고 한다면 각각의 노드에는 일련의 필드(Field)를 포함하며 이것은 기능적 요소 값을 가진다. WRL 파일 상에서 계층 구조는 노드와 노드 간의 종속관계로 표현된다. 노드의 각 필드는 기정의(default) 값에 의해 생략될 수도 있고, 사용자에 의해 값이 지정될 수도 있다. WRL 파서는 이러한 특성을 고려하여 다양한 노드에 대한 계층 구조 구성 및 기정의 값의 지정이 가능하도록 설계되어야 한다.

3. VRML 적재기

VRML 데이터를 메모리에 적재하기 위해서는 WRL 파일을 파싱하는 것뿐만 아니라 그 정보를 손실하지 않고 메모리로 적재시키는 과정이 필요하다.

VRML 문법에 맞게 VRML 각 키워드는 각각의 파싱 루틴을 필요로 한다. 메모리 자료형은 VRML 구조를 바탕으로 고안되었으며 렌더링에 사용될 추가 자료도 포함된다.

3.1. 파서의 구조

제작한 VRML 파서는 기본적으로 노드를 선언하는 키워드를 인식하여 노드를 읽어 들이는 함수를 부르는 콜백(call back) 형태를 취한다. 프로그램 내에 선언된 기본적인 함수 이외에 새로 추가될 노드의 확장성을 위해 이와 같은 형태로 설계하였다. 노드를 읽어 들이는 함수는 최종적으로 데이터를 메모리에 적재 시켜 현재 노드를 읽어 들이는 과정을 마친다.

노드를 판단하는 기준은 노드의 키워드와 노드를 읽어 들일 함수가 구조체 형태로 묶인 배열이다. 노드를 판단하는 함수는 배열을 순차적으로 검색하여 해당 함수를 부른다. 배열에 등록되지 않은 노드는 무시하고 다음 토큰으로 진행한다. 이러한 검색 과정은 파일의 끝을 나타내는 EOF를 만날 때까지 계속된다.

노드 키워드와 노드를 읽어 들일 함수를 묶은 구조체

```
struct key2func_t {
  char keywrd[32];
  void (*func)(void);
}
typedef struct key2func_t key2func_t;
typedef struct key2func_t * pkey2func_t;
```

VRML은 각 노드를 DEF 문으로 선언할 수 있고, USE 문으로 선언된 내용을 불러 쓸 수 있다. 이는 노드를 판단하는 키워드 토큰을 읽어 들이는 시점에서 판단할 수가 없기 때문에 노드를 읽어 들이는 함수는 각 경우에 따라 다른 코드를 실행하야야 한다. 이를 위하여 노드를 읽어 들이는 함수의 뼈대는 USE와 DEF를 인식하여 분기하는 구조를 띠고 있다. USE로 분기될 경우, 기존에 등록된 노드를 탐색하여 현재 노드와 링크하는 과정이 수행되어야 하고, DEF로 분기될 경우 새로운 데이터를 가진 노드를 생성하는 과정이 필요하다. DEF로 선언된 데이터를 가진 노드는 USE에 의해 검색되므로 검색에 필요한 데이터가 추가되어야 한다. 아스키 파일로 저장되는 VRML의 경우, 노드를 판단하는 기준은 해당 노드의 이름이다.

3.2. VRML 데이터의 메모리 적재

VRML을 렌더링하기 위해서는 메모리에 적재하고 렌더링할 수 있는 자료구조가 필요하다. 파싱 모듈상의 노드를 읽어 들이는 함수는 결국 데이터 적재 모듈의 해당 함수를 불러 메모리 적재과정을 마치게 된다. 각 모듈간의 함수는 1 대 1 대응관계를 갖는다.

키워드를 검색하여 해당 함수를 실행하는 함수

```
Int readChunk(void)
{
  int lpp, res;
  res = getToken(paramString);
  if(res!=EOF)
    for(lpp=0;lpp<sizeof(chunkKf) /
sizeof(key2func_t);lpp++)
      if(!strcmp(paramString,
chunkKf[lpp].keyword)) {
        chunkKf[lpp].func();
        break;
      }
    return res;
}
```

데이터 적재 모듈은 일종의 매개자료로서 VRML과 동일한 구조를 갖도록 설계하여 파서로부터 읽어 들인 내용을 손실 없이 저장한다. 이 구조를 프로그램에서 그대로 사용할 수도 있지만 보다 간략화 하여 저장하면 보다 빠른 렌더링이 가능하다. 간략화란 필요 없는 데이터를 삭제하여 메모리 사용량을 줄이는 것과 연결 리스트 형태의 자료구조를 배열형태로 바꾸어 데이터를 사용할 때 발생할 수 있는 캐시 미스(cache miss)를 줄이는 것을 뜻한다. 이러한 기능을 별도의 렌더링용 데이터 모듈을 통해 구현하였다. <그림1>에서 PRCONV 모듈이 이에 해당한다. 실제 렌더링은 이 모듈에 의해 변환된 자료구조를 사용한다.

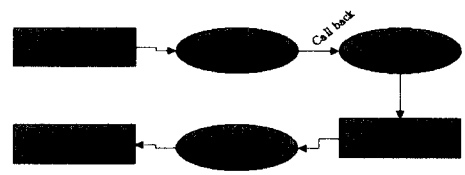

그림 1 렌더링을 위한 VRML 데이터 흐름도

<그림1>에서 VCCONV로 표현된 데이터 적재 모듈의 목적은 VRML과 동일한 구조로 메모리를 동적으로 할당하여 VRML 파일을 한번만 읽어 메모리상의 데이터 자료를 완성하는데 있다. 따라서 모든 노드는 이중 연결리스트를 활용한 트리 형태로 관리된다. 트리상의 부모 노드는 이중 연결리스트로 구성된 복수의 자식 노드를 갖고, 각각의 노드는 이중 연결리스트로 구성된 자기 자신

의 데이터를 갖는다. 자기 자신의 데이터란 정점 리스트, 면 리스트, 재질 정보와 같은 각 노드가 고유하게 갖는 정보를 의미한다.

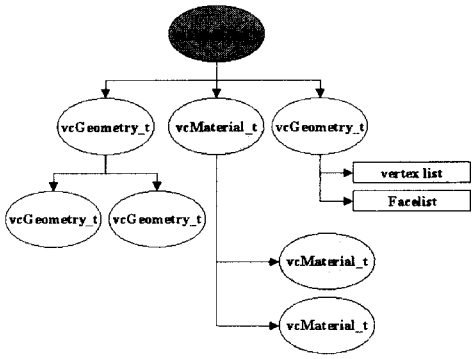

그림 2 VRML 데이터의 메모리 적재를 위한 계층적 데이터 구조

3.3. 구현

본 시스템은 마이크로 소프트 윈도우 95/98/NT와 SGI 에서 사용할 수 있고, 렌더링으로서 OpenGL 그래픽 라이브러리를 사용하여 구현하였다. 아래 그림은 실행 예를 보인다.

<그림3>의 프로그램은 파싱이 성공적으로 이루어졌는지 여부를 보여주기 위하여 VRML의 계층 구조를 나타내었고, <그림4>의 프로그램은 읽어들인 내용을 바탕으로 가상세계를 네비게이션 할 수 있도록 만들었다.

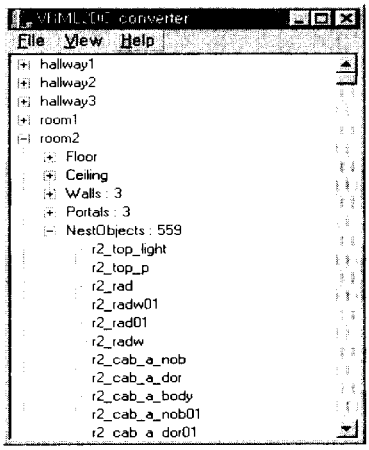

그림 3 Microsoft Windows 95/98/NT용으로 작성된 VRML 로더

4. 결론

웹 환경은 그 폭발적인 인기로 어느 누구에게도 낯설지 않은 환경이 되었다. 이는 웹 구축에 사용되는 정보량이 그만큼 늘었다는 것을 뜻한다. 이미 구축된 웹은 개발자들에게 좋은 자료가 되기도 한다. 웹이라는 풍부한 자원을 활용하기 위해서 웹형의 자원을 활용할 수 있는 능력은 중요하다. 웹과 연동하여 동작하는 응용 프로그램은 개발에도 웹형의 자원을 처리할 수 있는 능력은 기본이 되어야 한다.

3차원 렌더링에 있어서 VRML 자료의 처리는 이제 더욱더 보편화되어 갈 것이다. 이러한 요구를 수용하기 위해서는 응용프로그램이 WRL 파일을 처리할 수 있어야 한다. 따라서 본 논문에서는 WRL 파일을 파싱하고 파싱한 내용을 메모리에 적재하여 사용할 수 있는 방법을 설명하였다. 파서는 응용프로그램의 정보 요구 수준에 맞게 파서 자체를 재설계할 수 있도록 유연하게 설계되었다.

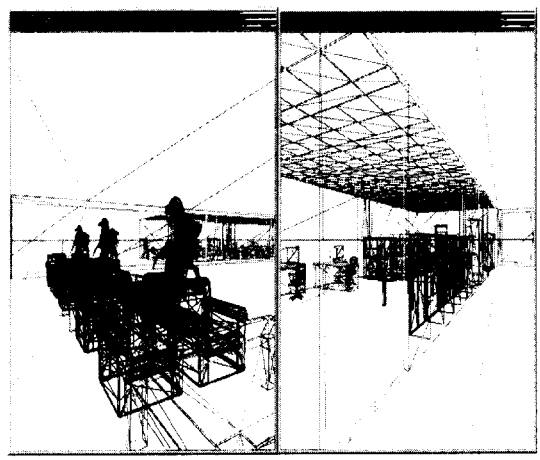

그림 4 VRML 데이터(WRL 파일)을 읽어 렌더링한 화면

5. 참고문헌

[1] Jed Hartman, VRML 2.0 Handbook, Addison-Wesley

[2] Richard S, OpenGL Super Bible, Waite Group

[3] Mason woo, OpenGL Programming Guide, Addison-Wesley

[4] Graphics Library Programming Guide Volume II, Silicon Graphics