Maya 데이터와 SEDRIS STF 데이타간의 자동변환기 설계 및 구현

허 용 도+, 이 광 형++

요 약

모델링 및 시뮬레이션 분야에서 이미 모델링되어 있는 환경 데이터를 재사용하고 확장할 수 있는 방법은 매우 중요하다. 이에 환경데이터의 공유 필요성을 만족할 수 있도록 확장될 수 있는 환경 데이터 표현 및 교환 메카니즘이 절대적으로 필요하다고 할 수 있다. SEDRIS의 STF(SEDRIS Transmittal Format)는 이러한 목적을 가능케 하는 표준으로써, 환경 데이터 사용자 및 생성자에게 명료하게 정의된 교환 명세를 제공한다.

본 논문에서는 SEDRIS의 표준 교환 포맷을 이용하여 상용 포맷(Maya)을 정보 내용의 의미 손실 없이 교환 포맷으로, 교환 포맷을 상용 포맷으로 변환하기 위한 자동변환기를 설계 및 구현하였다.

키워드 : SEDRIS, DRM, Maya, 자동변환, 교환 포맷, 객체 모델링, DAG, DAG 패스

The Design and Implementation of Automatic Converter of Maya Data And SEDRIS STF Data

Her, Yong Do+, Lee, Kwang-Hyung++

ABSTRACT

The method of reusing the environmental data which is previously modelled in modeling and simulation is very important. So, we need an environmental data representation and interchange mechanism which satisfies the requirements of sharing. The SEDRIS STF(SEDRIS Transmittal Format) provides environmental data users and producers with a clearly defined interchange specification.

In this paper, We design and implement an automatic converter which converts commercial data(Maya) format to standard interchange format and vice verse without losing semantic of information content using SEDRIS standard interchange format.

1. 서론

수년간 미 국방성은 모델링 및 시뮬레이션을 통해 군대 시스템의 성능 및 군대 구조 필요사항을 분석할 수 있는 능력을 얻기 위해 노력해 왔으며, 이러한 요구에 맞는 시뮬레이션 애플리케이션은 요구된 시스템 성능을 개발, 테스트, 및 평가할 수 있어야 한다.

군사작전 모델은 자연 환경의 표현과의 상호 작용에 의존한다. 결과적으로 환경의 공통적인 표현은 이질적인 시뮬레이션의 상호 조작을 위해 없어서는 안될 필수 조건인 것이다. 이것은 지형, 해양, 대기 및 우주공간에 대한 균일하고 믿을만한 3차

+ 정회원 : 건양대학교 전산게임학과 부교수
++ 정회원 : 고려대학교 시간강사
논문접수: 2004년 10월 1일, 심사완료: 2004년 11월 16일

원 표현을 요구하게 되었으나 미 국방성은 M&S(Modeling and Simulation) 애플리케이션들 중에는 환경 데이터의 표현, 재사용 및 상호 교환을 위한 표준 메카니즘이 존재하지 않는다는 것을 인식하였다. 이러한 근거로 가상적으로 구현된 환경 데이터베이스의 효과적인 교환을 지원하기 위한 새로운 솔루션이 요구되었으며, SEDRIS (Synthetic Environment Data Representation And Interchange Specification) 프로젝트는 이러한 문제들을 해결하기 위해 시작되었다[1,3,6].

기존의 환경 데이터들은 서로 다른 환경 데이터베이스 형식으로 작성되어 있기 때문에, 각각의 변환은 개별적 자료 변환 소프트웨어 애플리케이션의 개발을 요구한다. 이러한 일대일 솔루션은 비싸고, 시간 소모적이며, 종종 신뢰성이 떨어진다. 또한 각각의 변환은 데이터 손실이나 오류에 대한 위험이 높으며, 변환 소프트웨어 모듈의 개발과 유지 관리는 매우 어려워진다.

따라서, 본 논문에서는 국제 표준으로 규격화된 SEDRIS 데이터 모델인 DRM과 교환 포맷인 STF를 이용하여 환경 데이터의 중간 교환 포맷을 정의한다. 또한 상용 그래픽 도구인 Maya 데이터 포맷의 물리적인 구조와 논리적인 구조를 중간 교환 포맷으로 변환하고, 중간 교환 포맷을 다시 Maya 데이터 포맷으로 변환할 수 있는 자동변환기를 구현한다.

2. 기존의 환경 데이타 변환기의 문제점

상용 3D 애니메이션 또는 그래픽 도구에 있어서, 계산(computation)구조는 애플리케이션이 작동하기 위한 매우 중요한 부분이며, 각각의 도구에 있어서 경쟁력과 우수성을 나타내는 차별성이 많이 반영되는 부분이다. 이러한 계산구조의 반영은, 애플리케이션마다 scene 그래프 구조의 특징적 요소나 플러그인 인터페이스를 통해 구현되는데, 여기서 나타나는 연동 및 호환성의 문제가 환경 데이터 변환기의 구현에 상당한 난점들로 작용하게 된다.

Maya[5]의 경우, 다른 애플리케이션에 Maya의 scene 그래프 개념을 적용할 수 있도록 라이브러리를 초기화 할 수 있어 문제가 없지만, 이는 예외적인 경우에 불과하다. Maya 이외의 다른 도구의 경우에는 다른 애플리케이션에서 해당 scene 그래프로의 접근이 불가능한 경우도 많이 있다. 이는 요즘의 3D 애니메이션 도구들이 라이브러리 기반이 아닌, 플러그인 기반으로 구성되는 추세와 무관하지 않은데, 이럴 경우 플러그인(plug in)을 통해 scene 그래프에 접근하는 것은 필수라고 해도, 별도의 애플리케이션을 통해 scene 그래프를 조작하는 것을 지원하는 것은 선택사항이기 때문이다. 따라서, 소스(source)와 대상(destination)의 메모리 scene 그래프를 직접 연결시키는 방식은 플러그인 아키텍처 간에는 현실적으로 이루어지기 어렵다.

그러나 플러그인을 통해서건, 아니면 별도 애플리케이션을 통해서건 어느 한쪽의 scene 그래프만을 작동시키고 대응되는 상대편은 파일 형태의 데이터로 import 혹은 export하는 차선책은 가능하다. 이 방법은 데이터 중심적인 접근이라는 면에서는 현실적으로 상당히 타당한 장점을 가지고 있다. 그러나 이 경우에는 환경 데이터 변환을 위해 너무 많은 노력이 드는 단점이 있다.

이러한 문제는 중간에 공통의 교환 포맷이 존재할 경우, 상당히 개선될 수 있다. 왜냐하면, n개의 서로 다른 포맷간의 컨버팅이 O(n)개의 export 혹은 import 플러그인으로 해결될 수 있고, 어느 한 도구의 입장에서는 공통의 호환 포맷으로 1개의 export 혹은 import 플러그인만을 지원하면 되기 때문이다.

그러나 공통 호환 포맷이 가장 좋은 것이라고는 할 수는 없다. 공통 호환 포맷은 3D 도구들 간의 교환 가능한 3D-semantic을 모두 수용하거나, 앞으로라도 수용할 가능성을 갖추고 있어야 하는데 이 조건을 만족하는 것이 현실적으로 쉽지 않기 때문이다.

이상에서 설명한 바와 같이 기존의 환경 데이터 변환기들의 문제점을 정리하면 아래와 같다.

1) 근본적인 계산구조 차이와 플러그인 인터페이스의 비호환성
2) 상호 변환 문제를 기술적으로 심도 있게 다루기 어려운 상호 경쟁적인 시장 상황
3) 데이터 포맷간 exporter 혹은 importer의 개발
4) 호환 포맷이 없을 경우 발생하는 변환기 개발의 비효율성
5) 호환 포맷 요건 자체의 난이도, 즉 특정 계산구조에 독립적인 3D semantic 구조
6) SEDRIS나 BDF를 통한 해결책은 Semantic

지향적인 호환 포맷으로의 exporter 혹은 importer 개발 필요

3. SEDRIS를 이용한 데이터 표현 방법

3.1 SEDRIS 데이터 표현 모델의 필요성

데이터 표현 모델이란 각기 속성을 가진 데이터 요소 및 그 데이터 양식들 간의 논리적인 관계에 대한 설명이다. 데이터 모델의 장점은 이런 데이터 관계들을 완벽하고 명백하게 정의하는 능력이다.

실제 데이터와 통신하는 것 보다 데이터의 메타 모델로서 제공되는 데이터 모델과 통신하는 것이 대개는 더 수월하다. 이것은 메타 모델이 데이터를 기술할 때 그들의 저장 형식을 통해서가 아니라 그들의 속성을 통해서 기술하기 때문이다. 데이터 모델은 환경 데이터의 모든 양식이 포착되어 있고 교차되는 표현(예를 들면, feature 와 geometry)간의 관계가 정의되어 있음을 보증함으로써 모호성을 제거한다. 여러 가지의 접근 방법을 사용함으로써 다수의 표현 뷰(view)가 제공될 수 있다.

데이터 모델은 데이터 접근 시에 사용하기 위한 API 의 개발을 지원한다. 그런 까닭에 데이터 모델은 가상 구현된 환경 데이터의 재사용 및 교환을 가능하게 한다. 왜냐하면 데이터 생성자와 사용자가 그들의 원래 데이터 표현을 데이터 모델로 변환하기 위해 API를 이용함으로써 쉽게 소프트웨어 툴을 개발할 수 있기 때문이다.

3.2 SEDRIS 데이터 표현 모델

SEDRIS는 모든 시뮬레이션 응용 형태에 사용될 수 있는 데이터 표현모델(DRM)을 개발하였다. SEDRIS DRM에 포함되어 있는 데이터는 모든 환경 데이터(영토, 지형, 대양, 대기, 공간)를 포함한다. SEDRIS DRM은 데이터의 명백한 묘사를 제공할 뿐만 아니라 사용자가 해석을 제대로 할 수 있기 위해 중요한 데이터간의 관계도 정의한다[2].

SEDRIS DRM은 그들의 속성이나 관계와 더불어 데이터 형태의 식별을 제공하는데 이용되는 도식 기반 디자인 툴(tool)이다. DRM은 데이터베이스의 구현이 아니라, 왜 데이터가 필요한가의 정당성 뿐만 아니라 시스템내의 데이터의 형태를 묘사하는 것이다. DRM은 각각의 데이터 형태, 속성, 관계 등에 대한 추가적인 문자 정보를 제공하는 데이터 사전을 지원한다. DRM의 이용은 데이터에 대한 모호하지 않는 명료한 표현 정보를 위한 소프트웨어 공학 기술이다.

DRM의 중요한 이점은 모든 시스템 데이터 형태는 시스템 내의 다른 데이터 형태와 가지는 관계뿐만 아니라 속성의 완전하고 모호하지 않은 정의를 포함함을 보장하는 방법을 제공한다. DRM이 기술적 표현이기 때문에, 시스템의 완전한 표현 보장을 위해 쉽게 검사되고, 도전되며, 개선되어질 수 있다.

DRM은 프리미티브(primitive) 데이터 형태 뿐만 아니라 데이터 클래스도 식별한다. 데이터 클래스는 데이터의 조직적 구조를 묘사한다. 프리미티브 데이터 형태는 데이터 클래스를 기술하는데 이용되는 기본적인 데이터 요소들을 정의한다. 데이터 클래스의 식별은 또한 클래스 사이의 관계 기술의 능력을 제공한다.

3.3 SEDRIS 데이터 표현 모델링 기술

객체 지향 데이터 표현 모델을 묘사하는 데이터 클래스는 문제 영역에서 발견되는 실세계 "실물"의 추상화이다. 이러한 개념을 이용하면, "나무", "탱크", "건물", "벼랑길" 등과 같은 데이터 클래스를 SEDRIS DRM 내에서 표현할 수 있다. 이 표현은 종합 환경의 시각적이고 비시각적인 표현을 나타내기 위한 필수적인 데이터를 포함한 데이터베이스를 이용하여 실행된다. 그러므로 SEDRIS DRM 내에 발견되는 데이터 클래스의 한 집합은 시각적 장면을 묘사하기 위해 사용되는 기하학적 데이터 형태들이다. 이런 클래스들은 다각형, 선, 정점, 색, 촉감등과 같은 데이터 형태들이다. SEDRIS DRM은 또한 기하학적 데이터 형태들에 포함된 정보의 선택적인 표현을 제공하기 위한 데이터 클래스 집합을 포함한다. 이런 클래스들은 점, 선형, 그리고 지면 데이터 클래스를 표현하기 위한 노드, 에지, 그리고 면의 프리미티브 데이터 추상화를 이용하는 모양 데이터 클래스들이다.

예를 들어, 다각형 데이터 클래스는 종합 환경에서 "나무" 또는 다른 객체의 시각적 표현을 생성하는데 사용된 데이터를 포함할 수 있다. 그러나 데이터는 시각적 장면에서 "나무"의 이미지를 표현하는 다각형을 정의하기 위해 사용된 데이터는 나무

를 보기 위한 컴퓨터 생성력(CGF) 모델에 필요한 데이터의 종류일 수도 있고, 환경에서 "나무"의 존재에 근거하여 결정을 내리는 것일 수도 있다. 시뮬레이션 응용의 비시각적 부분을 지원하기 위해, SEDRIS DRM은 환경내의 "실물"의 선택적인 표현을 제공한다. 이 대안은 모양 데이터 클래스의 이용을 통해 구현된다. 환경중에 "나무" 위치에서, 점 모양의 경우는 "나무"와 동일시 된다. 이 점 모양은 "나무" 모양 모델의 표현을 정의하기 위해 특정한 방법으로 조직된 노드, 에지, 면의 데이터 클래스에 의해 기술된 어떤 속성을 가진다. 이 정보는 CGF 모델에 의해 "나무"에 대한 결정을 위해 직접적으로 이용될 수 있다. 따라서 "나무"의 시각적 이미지를 생성하는 기하학적 데이터와 "나무"에 대한 추론을 위한 모양 데이터 사이의 선택적 표현 관계 SEDRIS DRM에 포함된다.

시뮬레이션 응용의 비시각적인 구성요소의 추론 모델을 잘 보조하기 위해서, SEDRIS DRM은 또한 특별한 종합 환경에 포함된 "실물"에 대한 위상적인 정보도 포함한다. 위상적 정보는 기하학적 모양 데이터를 제공한다. 위상적 정보는 환경 내의 공간적 "실물"에 대한 연결성과 인접 정보를 제공한다. 위상적 정보는 이것을 기하학적 모양 데이터 클래스들로 연결하는 관계를 가진 SEDRIS DRM 내의 다른 특정한 데이터 클래스들도 포함한다. 위상적 관계는 정보 유추를 위한 계산의 실행대신에 명백한 정보를 제공받음으로 추론 모델을 보조한다.

3.4 SEDRIS 내부 객체 구조 및 연관관계

SEDRIS DRM 즉, SDRM은 현상적으로 존재하는 모든 환경 데이터를 표현하기 위해 만들어진 모델과 모델링 방법에 대한 규칙이다. 각각의 DRM 클래스는 추상적 혹은 실제적인 하나의 객체를 표현하기 위한 객체로서 정의할 수 있으며, 각각의 객체들은 서로간의 연관관계를 표현할 수 있는 다양한 연관 방법을 제시하고 있다. 또한 각 객체에는 객체만이 가지는 고유의 속성(Attribute)들을 표현할 수 있다.

SEDRIS의 DRM 클래스중에서 3D 그래픽 데이터의 변환을 위해 사용될 수 있는 가장 중요한 DRM 클래스는 <GEOMETRY>이다. Maya와 같은 3D 그래픽 에디터의 데이터 변환에는 이들 클래스중에서, <Union of Geometry>, <Union of Geometry Hierarchy>, <Union of Primitive Geometry> 등이 주요한 클래스들로서 사용된다. 또한 SEDRIS DRM의 가장 큰 특징중의 하나인 <Model> 클래스는 자체의 지역 좌표계(Local Coordinate)와 변환 정보를 가진 하나의 객체로서, 주어진 환경 영역에서 한번이상 존재할 수 있다. 즉 <Model>은 언제나 재사용이 가능하다. 또한 <Model> 내에서도 다른 하위 <Model>을 컴포넌트 개념으로서 가지고 있을 수 있어, 다중 복합 <Model>이 존재할 수 있다. 이것은 Maya에서 사용되는 하나의 캐릭터의 복합적인 구성 요소들을 표현할 수 있다.

4. Maya 데이터의 표현 방법

4.1 Maya의 Scene 그래프와 3D Semantic 구조

3D분야에서 scene을 표현하는 자료구조, 즉 scene 그래프의 일반적인 형태는 주로 트리 구조나 DAG 구조인데, Maya는 DAG를 채택하고 있다.

Maya가 채택한 scene 그래프 구조는, 엄밀히 말해 순수 DAG구조라고는 할 수 없다. 전체적인 구조는 오히려 디렉티드(directed) 그래프에 가깝고, 이 중 일부에 DAG구조를 적용한 형태이다. 단지, 이 일부의 DAG구조가 Shape이나, Shape의 <위치, 방향, 크기>를 나타내는 Transform으로 이루어져 있으며, 이 부분이 전체 그래프 구조에서 3D 특성을 부여하는 중추적인 역할을 하기 때문에, Maya에서의 scene 그래프의 구조를 DAG 구조라고 할 수 있다.

(1) 노드

Maya에서 DAG는 geometry의 위치, 방향, 크기 요소들을 정의하는 transform 노드와 shape 노드로 구성된다.

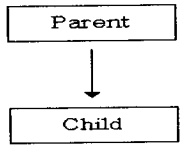
transform 노드는 위치, 회전, 크기와 같은 transformation 정보 및 DAG 계층상에서의 부모 정보 (parenting information)를 관리한다. 예를 들어, "손바닥"과 "손가락"을 모델링하고 이 둘을 하나의 "손"으로서 같이 회전시키고자 한다면, "손바닥"과 "손가락"을 공통의 parent transformation에

연결시키고, 이것에 rotation을 적용하면 된다. transform 노드중에는 LOD나 애니메이션 등과 관련되어 특화된 노드들(LodGroup, Joint 등)도 있다.

shape 노드는 polygon, NURBS, sub-division등의 geometry를 나타내는데, transformation 정보나 구조와 관련된 parenting정보를 포함하지 않는다. 따라서, shape 노드는 DAG 계층에서 항상 terminal로만 나타난다.

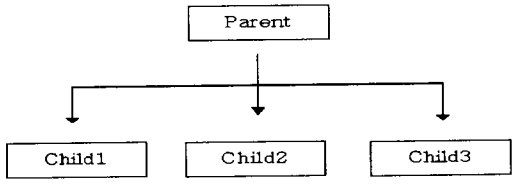
(2) Hierarchy(parenting, grouping, instancing)

DAG 노드들은 <그림 1>에서 보는 바와 같이 DAG 계층상에서 parent, sibling, children을 포함할 수도 있고, 포함될 수도 있는데, 이러한 parenting information은 연결된 노드들간의 관계에서 정의된다.



<그림 1> Parenting

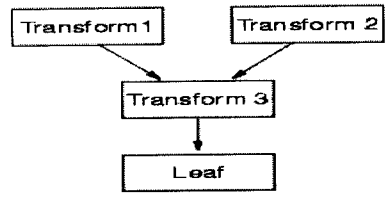
<그림 2>에서 보는 바와 같이 transform 노드들은 다수의 자식 노드들을 가질 수 있는데, 이 경우 자식 노드들은 transform 노드에 그룹화 되고, 이러한 노드 그룹핑(grouping)은 자식 노드들로 하여금 transformation 정보를 공유하여 하나의 단위로서 다루어 질 수 있게끔 한다.



<그림 2> Grouping

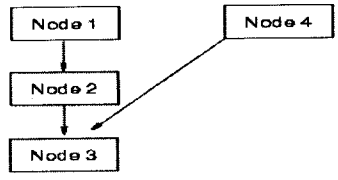
또한, <그림 3>에서 보는 바와 같이 하나의 DAG 노드가 다수의 부모 노드를 가질 수 있는데, 이 경우 부모 노드들은 자식 노드에 대한 인스턴스(instance)들이 된다.

인스턴싱(Instancing)은 어떤 모델에 대한 geometry 정보의 양을 줄이는데 유용하다. 예를 들어, 나무가 수 천 개의 leaf들로 구성 된 경우, 각각의 리프(leaf)를 별개의 NURBS나 polygon 데이터로 만든다면, 트리는 전체적으로 상당히 많은 양의 데이터로 이루어져야 한다. 반면, 하나의 leaf를 수 천 번 인스턴싱을 한다면, 하나의 모델을 공유하면서 데이터 양을 효율적으로 관리할 수 있는데, 자식노드의 geometry가 복잡할수록 인스턴싱의 효율은 높아진다.



<그림 3> Instancing

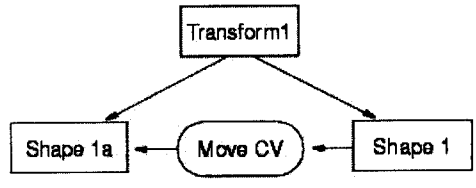
또한, <그림 4>에서 보는 바와 같이 Maya는 generalized instancing을 지원한다. generalized instancing은, 다른 노드를 인스턴싱하는 노드들끼리 sibling 관계일 필요가 없다는 의미이다.



<그림 4> Generalized instancing

<그림 4>에서 Node2와 Node4는 각각 Node3을 인스턴싱 하고 있지만, sibling 관계는 아니다. 이러한 인스턴싱을 통해 매우 복잡한 계층을 만드는 것도 가능하다.

또한, transform 노드는 임의개수의 transform 노드를 자식으로 가질 수 있고, 하나 이하의 shape 노드를 자식으로 갖는다. 그러나 <그림 5>에서 보는 바와 같이 transform 노드가 다수의 shape 노드를 자식으로 갖는 경우가 있는데, 이것은 원래의 shape이 dependency graph에 의해 수정된 경우에 발생한다.



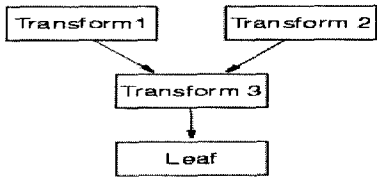
<그림 5> Transform with multiple shapes

이 경우 수정된 shape의 transformation을 관리하

기 위해, 이 shape을 원래 shape과 같은 transform 노드 밑에 위치시킨다. 그러면, 원래의 shape과 동일한 transform이 적용되며, 어떤 방식으로든 (위의 경우, shape의 CV들의 이동) 수정이 가능하게 된다.

(3) DAG 패스(path)

하나의 패스는 DAG 계층에서 특정 노드나 인스턴스를 유일하게 식별할 수 있는 노드들의 집합 혹은 열로 나타내며, 루트 노드로부터 패스가 식별하는 모든 노드까지의 계승관계를 표현한다. 인스턴스 노드에 관해서는 여러 개의 path가 존재할 수 있는데, 루트 노드부터 인스턴스 노드까지 각각의 인스턴스에 대해 하나의 패스가 주어진다.



<그림 6> DAG 패스 예

<그림 6>에서 leaf의 두 개의 인스턴스에 대한 패스는 다음과 같다.

{|Transform1|Transform3|Leaf}
{|Transform2|Transform3|Leaf}

노드의 이름과 달리 패스는 DAG 계층에서 노드나 인스턴스를 유일하게 식별하므로, 특히 API 수준의 접근에서는 매우 중요하다.

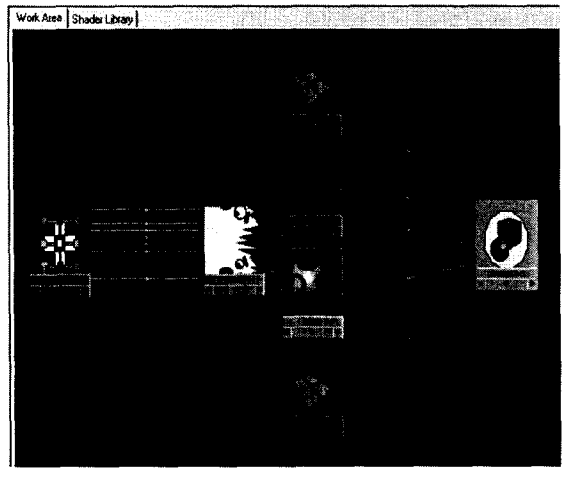
4.2 Maya의 Scene 그래프와 computation 구조

Maya의 scene 그래프에서 모든 노드들은 DG 노드이다. DG 노드는 Maya에서 특정한 계산을 표현하는 역할을 하는데, 이 DG 노드와 연관된 플러그인에 의해 계산된다. 계산 구조의 입력과 출력은 서로 연결될 수 있는데, 이렇게 나타난 연결망은 Maya에서의 복잡한 계산들을 가시적으로 보여준다. <그림 7>은 Maya에서 대표적인 DAG와 DG의 조합인 shading 망의 예이다. DAG 노드인 3개의 shape 노드들이 shading 노드를 통해 texture material(ballM)과 결합되고 있다. 이러한 연결망은, Maya에서 3D semantic의 구조일 뿐만 아니라, visualization등을 위해 실제 수행되는 계산들의 모델이다.

(1) DG(Dependency Graph) 노드

DG는 서로 연결된 개체들의 집합이다. DAG와 달리 연결은 cyclic일 수도 있으며 parenting 관계를 나타내지 않는다. 대신 그래프상의 연결은 하나의 개체로부터 다른 개체로 데이터가 이동하는 역할을 한다. 그래프에서 데이터를 받거나 보내는 개체를 dependency 노드라고 한다.

DG 노드들은 DG에서 계산을 수행하는 부분이다. 하나의 DG 노드는 다른 노드와의 연결을 통해 받은 데이터들이나 자체로 가지고 있는 데이터들을 입력으로 이용하여 출력 데이터를 생성한다.



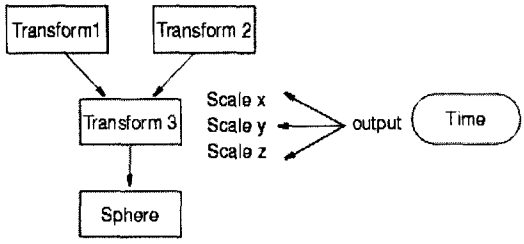
<그림 7> DG node의 예 : Shading Network

Maya에서 대부분의 객체들은 DG 노드이거나 이들의 연결 망이다. 예를 들어, DAG 노드는 DG 노드이며, shader는 노드간의 연결 망이다. DG 노드가 연결되어 DAG 노드에 영향을 줄 수 있기 때문에 화면 렌더링에 실제적인 영향을 미친다.

<그림 8>은 DG와 DAG 계층의 결합을 보여준다. Transform1, Transform2, Transform3, Sphere는 DG 노드이면서 DAG 노드이고, Time은 단지 DG 노드이다. Transform3 노드의 x, y, z scale 파라메타들은 Time 노드에 의해 구동되는데, Time 노드의 출력이 Transform3 노드의 x, y, z scale 연결점들로 플러그인 되고 있다고 생각할 수 있다.

위의 그래프 구조가 실제 애니메이션으로 플레

이되면, 두 개의 Sphere 인스턴스들의 크기가 변화한다.



<그림 8> DG와 DAG 계층의 결합

(2) 속성(Attribute)과 플러그(Plug)

변경시킬 수 있는 방법이 제공되지 않는 노드는 그 자체로는 의미가 없다. 노드에 대한 변경은, 속성과 플러그에 의해 이루어진다.

노드의 속성은 그래프에서 다른 노드에 대한 데이터 인터페이스를 정의한다. 데이터들은 속성을 통해 평가 시에만 입출력된다. 속성은 입력받는 데이터의 형식이 무엇인지, 데이터의 이름이 무엇인지, 데이터를 하나의 리스트로 만들 수 있는지 , 어떤 접근 방식으로 데이터 이동할지 등을 나타낸다. 데이터의 형식은 비교적 간단한 정수형이나 실수형을 나타내며, 약간 복잡한 형식인 polygon mesh, NURBS 표면 등을 포함한다.

노드의 플러그는 주어진 속성에 대한 데이터를 실제적으로 포함하는 구조체이다. 따라서, 속성에 대한 모든 접근은 플러그를 통해 이루어진다. 속성 자신은 오직 데이터의 형식과 이름만을 정의한다. 따라서 플러그는 노드간의 연결을 만드는 연결점들이기도 하다. 속성 이름은 DG 노드의 계층구조에서 유일해야 하지만, 서로 상관이 없는 노드들 간에서는 재사용이 가능하다.

(3) 데이터 블록과 데이터 핸들

데이터 블록은 노드의 데이터에 대한 저장 객체로 노드가 보내고 받는 데이터들을 관리하는데 사용된다.

Maya에서 렌더링(rendering)시 노드는 하나의 픽셀에 대해서 여러 번 계산 메쏘드(compute method)를 수행할 수 있다. 이것을 주의하지 않으면, 의존 그래프(dependency graph)는 렌더링시의 병목점이 될 가능성이 크다. 보내거나 받는 데이터를 얻어내는 것이 처음에는 다소 복잡해 보일 수 있지만, 전체적으로 빠른 메커니즘을 구성하는데 꼭 필요한 부분이다.

데이터 핸들은 데이터 블럭 안을 참조하는 것으로, 노드 데이터 중 속성이나 플러그로 구별되는 특정 부분을 참조한다. 즉, 속성의 데이터를 가져오거나 세팅할 때 데이터 핸들을 이용한다.

(4) 계산 메쏘드(Compute method)

의존 그래프 노드를 구현하는 것은, 속성을 할당하고 계산 메쏘드를 코딩하는 것이다. 계산 메쏘드는 노드 인스턴스의 속성과 플러그로부터 입력을 받는다. 입력은 속성에 할당된 플러그가 없을 경우 해당 속성의 기본 값이 되고, 속성에 할당된 플러그가 있을 경우 플러그의 값이 된다. 플러그가 해당 정보에 대한 요청을 받으면, 플러그는 요청에 대해 응답할 수 있는 적절한 상태인지를 검사한다. 만약, 적절하지 않은 상태라면, 플러그에 연결된 다른 노드에게 자신의 값을 갱신하도록 요청한다. 이러한 요청은 전체 DG를 통해 전파될 수 있다. 일단, 모든 플러그들이 clean 상태가 되면, 노드는 플러그들로부터 데이터를 취해 결과를 계산하고 출력 속성으로 보낸다.

하나의 노드는 그 자신의 속성과 플러그 외의 어떠한 것에 대해서도 알아서 안 된다. 예를 들어 그것이 DAG나 다른 DG 노드를 바꾸게 되면, 새로운 DG 평가가 발생하고 전파를 통해 그 자신의 노드를 재평가 하게 된다. 이것은 평가가 무한 루프에 빠지는 것을 의미한다. 혹시 어떤 이유로 노드가 그 자신의 주변 환경에 대해 알아야할 필요가 있다면, 그런 지식은 연결될 수 있는 속성의 형태로 제공받아야 한다.

5. Maya 데이터와 STF의 자동 변환기 구현

5.1 SDRM을 이용한 Maya DAG 계층 표현

SDRM의 3D semantic 부분은 시뮬레이션 분야의 교환 포맷으로 엄격하게 구조화되고 정리되어 있다. 따라서 Maya같은 상용 툴의 3D semantic을 수용 하는 데에도 전혀 무리가 없다. Maya의 3D semantic의 핵심인 DAG 계층은 SDRM의 <Union Of Geometry Hierarchy>, <Union Of Primitive

Geometry> 클래스를 통해 shape 정보(polygon, mesh등), 부모-자식 관계, 그룹핑을 표현할 수 있고, <Geometry Model>과 <Geometry Model Instance> 클래스를 이용하여 인스턴싱 관계를 표현 할 수 있다.

(1) Maya transform의 변환

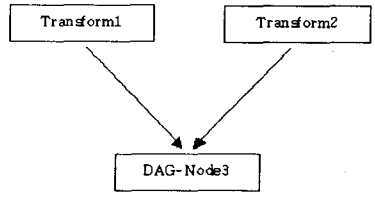
Maya DAG 계층에서 transform 노드는 <translate, rotate, scale>등의 transformation 정보 뿐 만이 아니라, 부모-자식 관계, 그룹핑 및 인스턴싱 관계를 나타내기 때문에, 계층 특성과 밀접한 관련이 있는 노드이다. Maya의 transform 노드는 SDRM에서 <Transformation> 클래스의 인스턴스를 구성요소로 <Union Of Geometry Hierarchy>와 일치한다. SDRM에서 <Union Of Geometry Hierarchy>는 부모-자식 관계나 그룹핑을 직접적으로 포함하며, <Transformation> 구성요소를 통해 Maya transform 노드의 transformation 정보를 옮길 수 있다.

(2) Maya shape의 변환

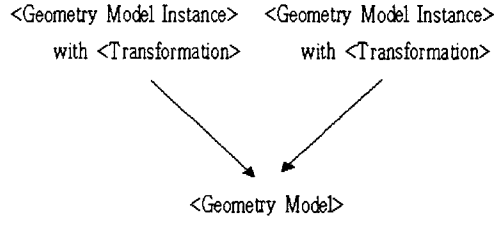
Maya의 shape 노드는 Mesh나 NURBS 등의 geometry 정보만을 포함한다는 의미에서 SDRM의 <Transformation> 구성요소가 존재하지 않는 <Union Of Primitive Geometry>와 일치한다. Maya의 shape 노드는 다른 DAG 노드를 자식으로 가질 수 없고, SDRM의 <Union Of Primitive Geometry> 역시 <Union Of Geometry Hierarchy>나 <Union Of Primitive Geometry>를 포함 할 수 없으므로, 두 가지 모두 계층상에서 terminal 노드에 해당된다.

(3) Maya 인스턴스의 변환

Maya의 인스턴스는 DRM의 <Geometry Model>과 <Geometry Model Instance>를 통해 표현될 수 있는데, Maya의 인스턴스 관계에 포함된 transform 노드와 SDRM의 <Geometry Model Instance>와 약간 다르기 때문에, 적용상에 약간의 주의가 필요하다.
<그림 9>에서 DAG 노드3은 단순한 shape 노드가 될 수도 있고, 서브트리의 루트노드일 수도 있는데, 부모 노드의 개수가 2 이상이므로 <Geometry Model>에 해당한다. 따라서, 위의 경우를 SDRM으로 표현하면 <그림 10>과 같다.

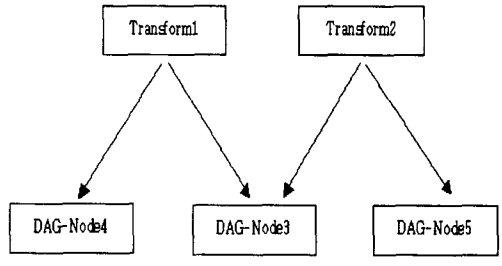


<그림 9> 인스턴싱



<그림 10> Maya 인스턴스의 SDRM

<그림 10>의 SDRM표현은 <그림 11>과 의미론적으로 일치하지만, Maya의 transform 노드를 <Geometry Model Instance>로 표현하는 것이 항상 가능하지는 않다.

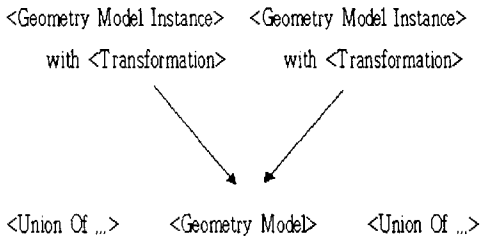


<그림 11> 별도의 DAG 노드를 포함하는 경우

<그림 11>은 인스턴싱에 관계된 transform 노드들이 각각 별도의 DAG 노드를 포함하고 있는 경우이다. <그림 11>의 transform 노드를 <Geometry Model Instance>로 바꾸면 <그림 12>와 같다.

그러나 <그림 12>의 경우 <Geometry Model Instance>의 정의상 DAG 노드4와 DAG 노드5에 해당되는 SDRM의 구성요소가 <Geometry Model

Instance>의 구성요소로 될 수 없다.

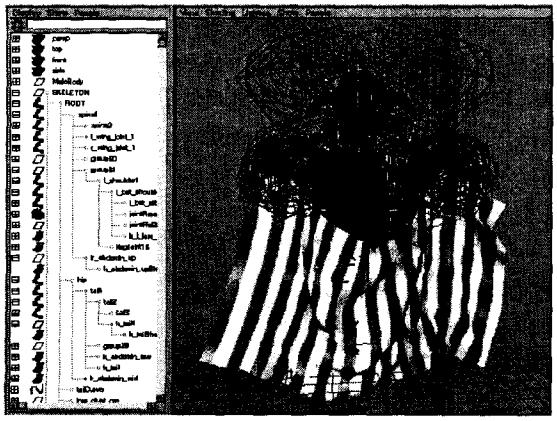


<그림 12> <그림 11>표현에 대한 SDRM

왜냐하면, <Geometry Model Instance>는 하나의 <Geometry Model>만을 지시할 뿐, 별도의 <Union Of ...>를 포함하지 않기 때문이다. 따라서, SDRM의 <Geometry Model Instance>는 Maya DAG상에서 인스턴스와 관계된 transform 노드에 적용하는 것이 아니라, 부모 노드가 2이상인 자식 노드와 이를 자식으로 갖는 transform 노드와의 관계에 적용해야 한다. 인스턴스에 관계된 링크는<transformation>이 없는 <Geometry Model Instance>과 일치한다.

5.2 Maya와 SEDRIS STF간의 자동변환기

Maya와 SEDRIS STF간의 자동변환기는 Maya 어플리케이션의 규약에 의하여 작성했으며, 기능은 Maya의 scene에 있는 그래픽 객체중 기하 객체들의 내용과 부모 자식 계층구조의 의미(semantic)의 변환에 중점을 두었으며, 외관(material)에 관해서는 기본적인 기능만 구현했다. 즉, 변환기에서의 외관은 객체의 와이어 프레임 칼라와 비트맵 형식(JPEG)의 텍스춰 맵만 이용하였다.



(1) Maya 로 그림 그리기

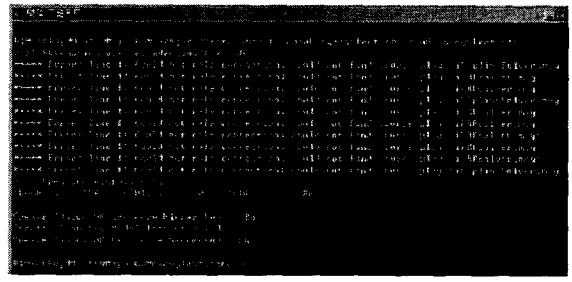
이 단계에서는 디자이너가 Maya 기능을 이용하여 <그림 13>과 같이 캔버스상에서 그림을 직접 그리거나, 기존의 Maya 파일을 화면으로 로드한다.

(2) STF 형태로 변환 수행

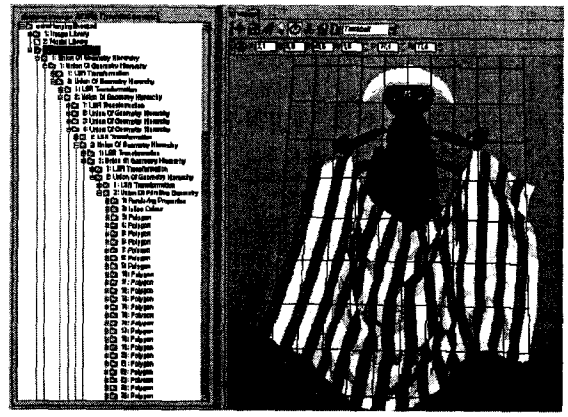
디자이너가 Maya의 캔버스에 그림을 다 그린 후 Maya 파일을 STF로 변환하기 위해서 <그림 14>와 같이 ma2stf 프로그램을 실행시킨다. 프로그램이 정상적으로 실행되면 STF 형태로의 변환이 시작되고 지정한 디렉토리에 STF 파일이 생성된다.

변환이 성공적으로 이루어졌는지를 확인하기 위해서 변환된 ednaHangingSheet.stf 파일을 side by side 브라우저를 이용하면 <그림 15>와 같이 정상적으로 변환된 것을 확인할 수 있다.

< 그림 13> Maya의 canvas에 그림을 그린 화면



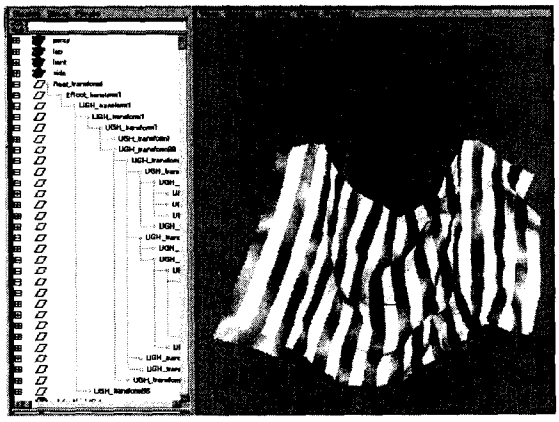
< 그림 14 > STF 변환 실행화면



< 그림 15 > sbs 브라우저의 결과 화면

(3) STF 파일을 Maya에서 확인

변환된 stf 파일은 사용자가 지정한 디렉토리와 파일명에 따라 STF 형태로 저장이 되어있는데, 이것을 다시 Maya에서 로드하여 확인하는 과정이 <그림 16>에 잘 나타나 있다.



<그림 16> Maya에서 stf 파일 열기

6. 결론

Maya는 기본적으로 플러그인 형태로 소프트웨어가 구성되어 있고 exporter 나 importer를 플러그인으로 만들수도 있고, 라이브러리의 모드를 플러그인이나 스탠드얼론(stand-alone) 애플리케이션 중에서 선택할 수 있다.

본 논문에서는 자동변환기를 애플리케이션 형태로 구현하였지만 플러그인으로의 전환도 쉽게 가능하다. 서로간에 다소간의 장단점은 있지만 근본적으로 변환에 문제가 발생하지는 않는다. 플러그인의 형태는 사용자 인터페이스에서 export 대상으로 선택하는데 유리하고, 애플리케이션 형태는 배치 작업이나 다른 3D 애니메이션과의 연동에 유리한 특징이 있다.

본 논문 이후, 현재 개별적으로 구현되어 있는 3D MAX 변환기와 Maya 변환기를 통합하여 사용할 수 있는 통합 변환기가 구현되면 기존의 다양한 환경 데이터를 다양한 3D 저작 도구 툴에서 데이터의 손실없이 사용하는 것이 가능할 것이다.

참고 문헌

[1] Defense Modeling and Simulation Office, "Synthetic Environment 데이터 Representation and Interchange Specification Overview,", Mar. 1998.

[2] Michael R. Welch, "SEDRIS As An Interchange Medium," Orion Development Group, Inc. Mar. 1998.

[3] John E. Carswell,"Fundamentally SEDRIS: he Technology Components," Aug. 2002.

[4] Boris Kulagin, "3ds max 4 : from objects to animation", Charles River Media, 1998

[5] Mark Adams, Erick Miller, Max Sims "Inside Maya 5" , New Riders, 2001

[6] ISO/IEC JTC1/SC24, "SEDRIS Environmental Data Representation and Interchange Specification (SEDRIS)-Part 1: Functional specification." Jun. 2002.

허 용도

1986년 고려대학교 수학과(이학사)
1988년 고려대학교 대학원 이학석사(전산학 전공)
1993년 고려대학교 대학원 이학박사(전산학 전공)
1993년~현재 건양대학교 전산게임학과 부교수
관심분야 : 분산시스템, 컴퓨터 네트워크 보안, 데이터마이닝, 컴퓨터그래픽스

이 광형

1995 고려대학교 전산과학과 졸업(이학박사)
1996.2 ~ 1997.1 한국표준협회 선임연구원
1997.3~ 2000.2 동서대학교 컴퓨터공학과 전임강사
2000.3 ~ 2002. 3 (주)ECO 시스템개발실 부장
2002.4~ 003. 6 (주)코딕커뮤니케이션즈 책임연구원
현재 : 고려대학교 강사
관심분야: 컴퓨터 그래픽스, CBD, 병렬/분산시스템