OpenGL 명령어 캡쳐를 통한 VRML Scene Graph 생성

박수호*, 정기숙*, 권순일*, 정순기*
*경북대학교 컴퓨터공학과
{shpark, gsjung, sikwon}@vr.knu.ac.kr, skjung@knu.ac.kr

The VRML Scene Graph Construction from OpenGL Command Capture

Su-Ho Park*, Gi-Sook Jung*, Soon-Il Kwon*, Soon-Ki Jung*
*Dept of Computer Engineering, Kyung-Pook National University

요 약

최근 유비쿼터스 게임의 성장과 더불어 이기종의 환경에서 새로운 기법들로 보여질 수 있는 연구가 활발히 진행되고 있다. 본 논문에서는 이기종 환경에서 임의의 OpenGL 기반의 PC 게임이나 게임 뷰어가 있는데서 Java3D 형태로 재복원해서 볼 수 있는 방법과 VRML을 사용하여 캡쳐된 장면들을 네비게이션 해 봄으로써 프레임간 오브젝트들의 움직임을 비교·분석할 수 있는 방법을 제안한다. 결과적으로 구조적인 장면 그래프에서 사건을 처리하고, Parser에서 OpenGL Primitive들의 정점 정보를 해석해서 계층구조를 변화시킴으로서 보다 빠른 실시간 렌더링을 구현하고자 한다.

1. 서론

최근 유비쿼터스 컴퓨팅을 위한 프로토타입 시스템으로 이기종의 환경에서 새로운 기법들로 보여질 수 있는 연구가 많이 진행되고 있다[1]. 그러나 현재 대부분의 응용 프로그램들은 동일한 운영체제하에서 작동한다. 다양한 장치 및 플랫폼에 대한 사용자의 요구가 급상승하고 있는 시점에서, 그에 맞는 새로운 방법으로 시스템이 설계·개발되고 있다. 본 연구는 임의의 OpenGL 기반의 PC 게임을 구조적으로 검증된 Java3D로 재복원해서 볼 수 있는 방법과 더불어 VRML을 사용하여 웹브라우저에서 캡쳐된 장면들을 네비게이션하면서 프레임간 오브젝트의 움직임을 비교·분석할 수 있는 방법을 제안한다.

2. 관련 연구

기존 3D 게임에서는 동적으로 실시간 렌더링을 구현하기 위한 연구 기법들이 많이 있다. 본 논문에서는 구조적인 렌더링과 처리 시간의 단축을 동시에 향상시키는 방법으로 Java3D 장면 그래프를 통한 기법을 제안하고자 한다.

장면 그래프의 대표적인 사례로는 idx3d, Open Inventor, Open Scene Graph, Java3D 라이브러리들이 있다. 현재 많은 네트워크 게임이 OpenGL로 되어 있다. 그러나 OpenGL 같은 경우는 장면 그래프를 위한 루틴이 존재하지 않으며, 사용자가 glPush Matrix(), glPopMatrix() 및 여러 Transform 관련 함수를 사용해서 장면을 구성하는 프리미티브의 계층 구조를 나타낸다. 이는 전혀 객체적인 디자인을 할 수 없는 방식인데 반해 Java3D는 장면 그래프에 기반한 디자인 방식으로 객체를 그리므로, 계층적인 디자인을 직관적으로 할 수가 있다. 이와 같은 접근 방법을 사용한 이유는 그 클래스의 구조가 이미 검증된 것이기 때문이다. 또한 네트워크 상에서 3D 모델과 데이터간의 협조가 가능하고 플랫폼에 독립적인 특징으로 인하여 성능 확장이 쉽기 때문이다.

그림1. idx3d와 Open Scene Graph

3. 시스템 설계 및 고찰

본 논문에서 구현하고자 하는 전체 시스템의 구성도는 그림 2와 같다. 시스템은 Java3D 클래스를 사용하여 렌더링하며 장면 그래프를 통해 변형 계층구조를 이룬다. 본 논문에서는 게임에서 OpenGL 명령어를 캡처하는 부분과 캡처된 명령어들로부터 Java3D의 장면 그래프를 생성하는 부분에 초점을 맞춰 기술한다.

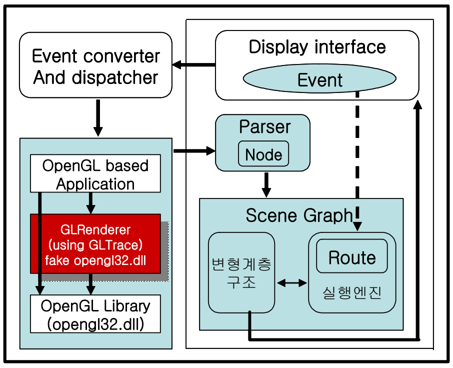


그림 2. 시스템 구성도

시스템에서는 Parser에서 기하 정보를 해석하고 장면 그래프를 만드는데, 이는 변형 계층 구조들과 Route Graph로 구성된다. 여기서 Route는 바로 이 노드들의 연결이라 할 수 있다. 장면 그래프는 또한 사건을 처리하고 Route Graph를 읽고 편집하는 실행엔진을 포함하며 계층 구조를 변화시킨다. 또한 데이터 간략화를 위해서 OpenGL 프로그래밍 유틸리티인 GLTrace[2]를 사용하였다.

4. OpenGL 명령어 캡쳐를 통한 장면 그래프 생성

본 논문에서는 OpenGL 기반의 PC 게임에서 장면을 각 프레임 별로 나누고 부분적인 결과로 장면 그래프를 구했다.

4.1 명령어 인터셉터 및 기하정보 생성

GLTrace는 어플리케이션이 실행되면서 호출되는 함수와 파라미터들을 파일로 기록한다. 실행되는 어플리케이션은 OpenGL을 기반으로 한 것이면 모두 가능하며, 중간에 명령어만을 가로채어 실제 opengl32.dll을 대신할 가짜 opengl32.dll을 생성하게 된다(그림 3). 이를 통해 기존 소스를 건드리지 않고 중간에 명령어만을 가로채어 기하 정보를 가진 새로운 자료구조를 생성한 후 다른 렌더링 스타일을 적용할 수 있다[3].

4.2 OpenGL 명령어를 장면 그래프로 재구성

본 논문에서는 간략화 된 기하 정보를 가지고 OpenGL 명령어 스트림 정보를 사용해서 한 프레임에 대한 정형화된 Java3D 형태의 장면 그래프를 만들었다(그림 4).

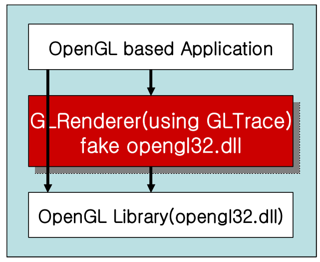


그림 3. Intercepting OpenGL Command

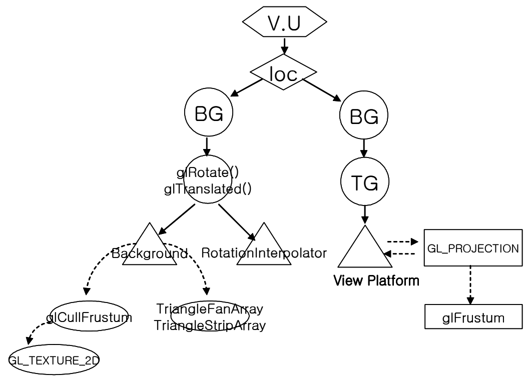


그림 4. 한 프레임에 대한 장면 그래프

4.3 OpenGL 명령어와 Java3D 클래스의 유사성

한 프레임에서 표현되는 OpenGL 명령어와 Java3D 클래스를 비교 분석해 본 결과, OpenGL에서 프리미티브 정점(vertex)들의 좌표를 명시하는 방법과 Java3D GeometryArray의 하부 클래스에서 좌표를 명시하는 방법은 유사하였다[4]. 즉, GeometryArray의 하부 클래스는 GeometryArray안의 정점들이 어떠한 도형을 나타내는지를 명시해주며, Geometry 정보를 렌더링할 때 점을 그릴지, 선을 그려야 할지, 삼각형을 그려야 할지를 알려준다. 또한 GeometryStripArray의 하부 클래스들에는 LineStripArray, TriangleFanArray가 있으며, 이것은 각각 OpenGL의 GL_LINE_STRIP, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN과 같은 기능을 한다. 그래서 우리는 OpenGL 프리미티브의 정점 정보를 가로채어 Java3D 장면 그래프의 기하 정보로 사용하였다.

4.4 정점 변환 파이프라인(Pipeline)과 외곽선 검출

우리는 glPushMatrix()와 glPopMatrix() 사이에 불려지는 프리미티브들의 집합을 하나의 오브젝트로 간주하고, 외곽선을 검출하였다. 그러나 가로챈 정점들은 모델 좌표계의 좌표들로 각 변에 대한 외곽선을 검출하기 위해서는 카메라 좌표계의 좌표들로 변환해 주어야 하고, 각 면에 대한 법선 벡터를 구해야 한다. 다음과 같이 수식(1)의 $M_c^{-1} M_o$에 해당

하는 MODELVIEW 행렬을 계산해서 카메라 좌표로 변환하였고, 삼각형 세 점의 두 벡터를 이용해서 법선 벡터를 구하였다[3].

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = M_c^{-1} M_o \begin{pmatrix} x_o \\ y_o \\ z_o \end{pmatrix} \qquad (1)$$

여기에서 (x_c, y_c, z_c)는 카메라좌표, (x_o, y_o, z_o)는 모델좌표, M_o는 모델 좌표계를 세계 좌표계로 변환하는 행렬, M_c^{-1}는 세계 좌표계로 변환된 좌표를 카메라 좌표계로 변환하는 행렬이다.

4.5 삼차원 정보 저장

정점 변환 파이프라인을 거쳐 디자인 된 삼차원 기하정보를 저장하기 위해서는 GeometryArray 객체를 이용하는데 다음과 같이 세 가지 과정을 거쳐서 Shape3D 객체의 속성으로 정의된다. 첫 번째로 빈 GeometryArray 객체를 형성한다. 여기에는 정점의 개수와 정점 형태의 두 가지 사항이 명시되어야 한다. 정점의 개수는 배열의 구성 요소에 필요한 꼭지점의 수를 나타내고, 정점 형태는 자료의 형태로서 좌표나 색상, 텍스쳐 좌표 등을 나타낸다. 두 번째는 GeometryArray 객체에 삼차원 정보를 저장한다. 즉, 명시된 정점 형태에 상응하는 배열에 값들을 할당한다. 세 번째는 Shape3D 객체가 GeometryArray 객체를 참조하도록 한다. 이와같은 구조로 삼차원 기하 정보들을 저장할 수가 있다.

5. Java3D에서 오브젝트 생성과 노드간의 연결

장면 그래프는 외형이나 변환, 재질, 조명 등을 정의하는 오브젝트들을 결합하여 만든다. 오브젝트를 만드는 가장 쉬운 방법은 기하도형 프리미티브 클래스를 사용하는 것이다. 이 클래스는 색상은 기술하지 않고 그 형태만을 기술하기 때문에 유연성이 있다. 기하도형 자체는 변경할 수 없으나 색상, 텍스쳐와 같은 겉모습을 바꿀 수 있다. 이 변경은 Appearance 객체에 Attributes를 연결함으로써 가능하다. 그래서 기하도형 프리미티브 클래스를 사용하여 오브젝트 별로 간략화 된 정점 정보를 파일로부터 읽어서 한 프레임의 Scene을 만드는데 적용하였다.

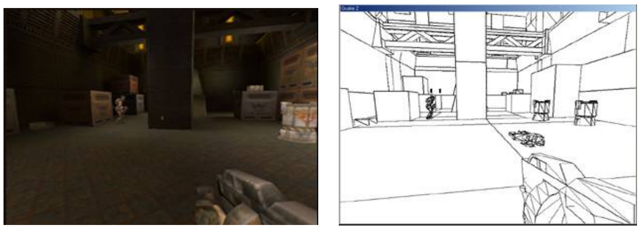
OpenGL에서 glPushMatrix()와 glPopMatrix()로 구분된 각각의 오브젝트는 Java3D의 TransformGroup에 각각 연결하고, 이 오브젝트의 움직임을 위해서는 주어질 변화를 Transform3D 객체에 연결하고, 이를 장면 그래프에 넣어 Rendering시키기 위해서 TransformGroup 객체를 이용하였다. Java3D에서 Transform3D 클래스는 그래픽 엔진에서 좌표를 변환, 투사하는데 쓰이는 함수들을 가지고 있다. 이 클래스를 이용해서 입력된 좌표나 벡터를 변형시켰다. 이와같이 사용자는 이 클래스들을 이용하여 프로그램 내에서 직관적으로 장면 그래프를 표현할 수 있게 된다. 따라서 장면 그래프가 형성되면 Java3D 렌더러(renderer)는 장면 그래프의 각 노드(node)를 Depth First Traversal하면서 객체들을 화면에 그려준다.

6. VRML 장면 그래프 생성과 네비게이션

Java3D에서 구조적인 장면 그래프가 생성되면 최상위 루트에 연결된 모든 노드들의 정보를 VRML97 라이브러리를 이용해서 VRML 파일로 저장하였다. 그리고 HTML의 EMBED 태그를 사용하여 VRML 파일을 HTML 문서 내에 포함시킴으로써 웹 브라우저를 통해 캡쳐된 장면을 네비게이션 할 수 있었다.

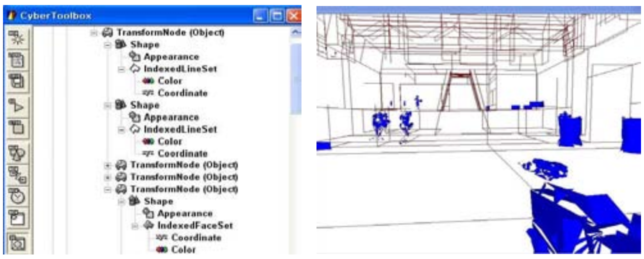
7. 실험 결과

본 논문에서는 OpenGL 기반 게임에서 데이터 간략화 기법에 의해 추출된 최소한의 기하 정보를 파일로부터 읽어 들여서 구조적으로 검증된 Java3D 장면 그래프를 구성한 후 렌더링을 하였다(그림 6). 이를 실험하기 위해서 GLTrace를 사용하였고, OpenGL 기반의 PC 게임으로는 QuakeII[6]를 사용하였다. 그리고 그림 7은 검증된 Java3D 장면 그래프의 구조를 보기 위해 VRML 뷰어로 장면 그래프 구조를 확인하고 렌더링 된 화면을 보았다[7].



(a) (b)

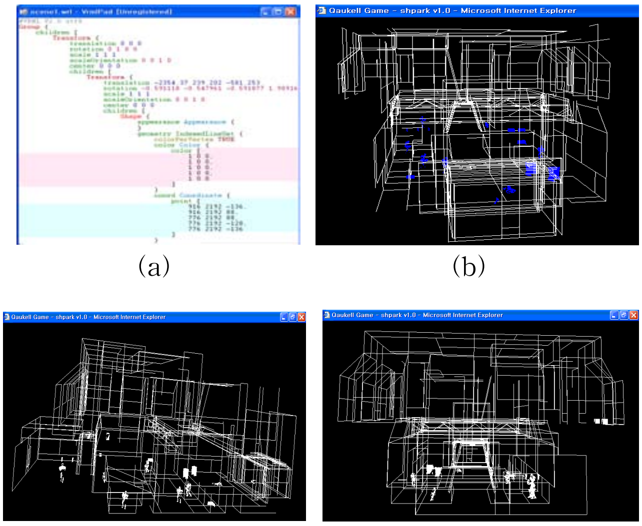
그림 6. 외곽선 검출 기법이 적용된 QuakeII 게임의 렌더링 결과 (a) OpenGL, (b) Java3D



(a) (b)

그림 7. VRML 뷰어로 본 QuakeII 장면 구조와 결과 (a) 장면 구조, (b) CyberToolbox for Java3D

그리고 그림 8은 VRML을 사용하여 QuakeII 게임에서 캡처된 장면을 네비게이션 하면서 모든 오브젝트들의 위치를 파악해 보기 위해서 와이어프레임(wireframe)으로 렌더링한 결과이다.



(a) (b) (c) (d)

그림 8. QuakeII 게임에서 캡쳐된 장면을 VRML을 이용해서 네비게이션 한 결과
(a) VRML 문서, (b) 정면, (c) 뒷면, (d) 옆면

8. 결론 및 향후 과제

본 논문에서는 임의의 OpenGL 기반의 PC 게임에서 OpenGL 명령어 스트림만을 가지고 특정한 프레임을 Java3D의 장면 그래프로 만든 후, Java3D 뷰어로 그 장면을 볼 수 있음을 확인하였다. 그리고 VRML을 이용해서 캡처된 장면을 네비게이션 해 봄으로써 전체 장면구조를 한 눈에 파악할 수 있었고, 이전 프레임과 현재 프레임간에 오브젝트들의 위치와 움직임이 어떻게 달라졌는가를 비교하고 분석할 수 있음을 보였다.

본 연구가 가지는 의의는 OpenGL 응용 프로그램의 장면이 가지는 Transform의 계층적 구조를 Java3D 혹은 VRML의 장면 그래프로 바꿀 수 있음을 보인 것이다. 향후 과제는 원격에서도 캡쳐된 장면을 실시간 공유하면서 보다 빠른 렌더링을 할 수 있는 연구를 들 수 있다.

감사의 글

이 논문은 2003년도 두뇌한국21 사업에 의하여 지원되었음.

참고문헌

[1] 정기숙, 이동훈, 정순기, “유비쿼터스 게임”, 정보처리학회지, 제10권 제4호 pp173-181, 2003년 7월.
[2] SGI and John Miles. GLTrace http://reality.sgi.com/opengl/gltrace/, 1997
[3] 최현미, 정기숙, 박찬, 박수호, 고영덕, 정순기, “OpenGL 기반 PC 게임의 PDA 적용을 위한 NPR 기법, HCI2004 학술대회, pp84-89, 2003년 2월.
[4] Java3D API Specification. http://java.sun.com/products/java-media/3D
[5] VRML97 http://www.vrml.org/
[6] ID Software http://www.idsoftware.com/games/quake/quake2
[7] CyberToolbox for Java3D http://www.cybergarage.org/vrml/ctb/java3d/