J2ME상에서 JSR-184를 이용한 모바일 3D 엔진의 설계 및 구현

조종근° 박윤희 김종민
고미드(주)
{jkdang°, multigirl, jmkim}@gomid.co.kr

Design and Implementation of Mobile 3D Engine using JSR-184 on J2ME

JongKeun Cho° YoonHee Park JongMin Kim
GOMID Inc.

요 약

본 논문에서는 J2ME상에서 JSR-184를 이용한 모바일 3D 엔진을 설계 및 구현하였다. 기존에는 모바일 표준 3D 그래픽 API(C언어 기반)인 OpenGL-ES를 사용하여 모바일 3D 엔진을 제작해, 핸드폰에 애플리케이션을 작동시켰으나, 저수준(Low-Level)의 다양한 기능만 제공함으로써, 다양한 콘텐츠제작 및 호환성에 제약이 많았다. 이에 OpenGL-ES보다 더욱더 다양한 고수준(High-Level)의 API를 제공하면서도 GSM 폰을 중심으로 J2ME상에서 자바환경에 최적화된 모바일표준 3D 표준 API(Java언어 기반)인 JSR-184로 모바일 3D 엔진을 제작하고, 스킨드 메시(Skinned Mesh) 형태를 가지는 모델의 처리속도를 향상시키는 방법을 제시한다.

1. 서 론

최근 모바일 3D 업계에서 OpenGL-ES와 더불어 주목받고 있는 것이 JSR-184이며, 자바환경에서 최적화된 모바일 표준 3D 그래픽스 API이다.

J2ME 환경에서 3차원 그래픽을 구현하기 위해서 저수준(Low-Level)의 OpenGL을 이용할 경우, 코드가 길어져 MIDlet(MIDP Application)의 덩치가 커지므로 속도가 느려질 수밖에 없고, 자바 3D API를 이용할 경우에는 스펙의 양이 너무 방대하기 때문에 역시 MIDP(Mobile Information Device Profile)에 이용하기엔 적합하지 않다.

본 논문에서는 다양한 모바일 3D 멀티콘텐츠 수용을 위해 JSR-184를 이용해서 J2ME 기반의 모바일 3D 엔진을 설계 및 구현하였다. 또한 노키아(Nokia)의 JSR-184 엔진에서 스킨드 메시(Skinned Mesh) 형태의 모델을 처리하는 것보다 좀더 빠르게 데이터를 처리할 수 있는 방법을 제시한다.

2. 관련 연구

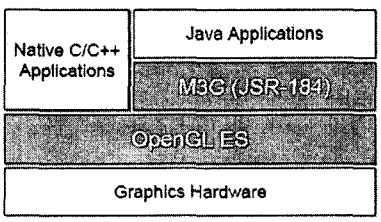

그림 1. 노키아 JSR-184 3D 엔진구성도

노키아의 JSR-184 모바일 3D 엔진 역시 그림 1과 같은 구조로 작성되었다. OpenGL-ES 단에 자바 가상머신과 M3G(JSR-184) 단에 자바 인터페이스(GUI)가 구동되어 실행되는 구조로 본 논문에서 설계한 방법과 크게 다르지 않다.

3. JSR-184 모바일 3D 엔진구성도

본 논문에서 구현한 JSR-184 모바일 3D 엔진구성도는 다음과 같다.

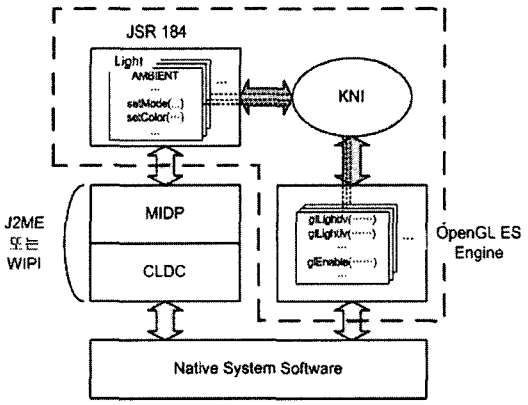

그림 2. JSR-184 3D 엔진구성도

그림 2에서 보는 것처럼 JSR-184의 30개 클래스들(Class)과 250개 메소드(Method)들로 3D 처리부분을 구현해 놓고, KNI(K-Native Interface)를 이용해서 C언어로

작성된 OpenGL-ES 함수와 매핑(Mapping)시켜 동작하게 한다.

3.1 자바언어 구현부분

JSR-184 스펙에 정의된 클래스와 메소드들은 순수 자바언어로 구현한다. 여기에 해당되는 것들은 다음과 같다.

3.1.1 3차원 그래픽스 부분

모델링(Modeling), 변환(Transformation), 카메라(Camera), 텍스쳐 매핑(Texture Mapping), 재질(Material), 라이팅(Lighting) 등의 기본적인 3D 그래픽스 표현과 관련된 부분이며, 기존의 OpenGL-ES 엔진을 최대한 활용할 수 있도록 자바 래퍼(Wrapper)를 작성하여, 3차원 그래픽스 부분을 구현하였다.

3.1.2 애니메이션 부분

키 프레임(Key Frame) 애니메이션 및 본(Bone) 애니메이션, 모션모핑 등을 의미하며 이 부분은 OpenGL-ES에 없는 부분으로써 먼저 C언어로 구현한 후, 이를 자바환경에서 사용할 수 있는 자바 래퍼를 별도로 작성하였다.

3.1.3 씬그래프(Scene Graph) 부분

계층구조(Hierarchy)지원, 오브젝트 트래버스(Traverse), 피킹(Picking), 정렬(Alignment) 기능은 자바로 구현하였다.

3.2 자바언어와 C언어 연결부분

J2ME 구현방식과 마찬가지로 JSR-184 역시 실제동작은 대부분 C언어로 작성된 부분을 통해서 동작한다. 즉, 자바에서 구현된 데이터 값들을 C언어로 작성된 부분으로 넘겨줄 때 이들 구조에 대한 적절한 변환을 수행하는 부분으로 KNI(K Native Interface)를 사용하여 자바와 C언어 부분을 연결부분을 구현하였다.

3.3 C언어 구현부분

자바로부터 KNI를 통해 넘겨받은 그래픽 관련 데이터들을 적절한 OpenGL-ES 함수와 매칭시켜 JSR-184 랜더러의 실제 동작부분을 구현하였다. 이 부분은 고미드에서 제작한 OpenGL-ES 엔진함수를 바탕으로 구현하였다.

4. 설계 및 구현

본 논문 3장의 내용들을 기반으로 설계한 JSR-184로 엔진구현을 제작하기 위해서 그림 3과 같은 절차를 따라서 작성하였다.

4.1 전체적인 설계도

본 논문에서 사용된 파일 포맷은 JSR-184에서 제공하는 M3G 파일 포맷을 사용하여 샘플 파일을 작성한 후 실험에 사용하였다.

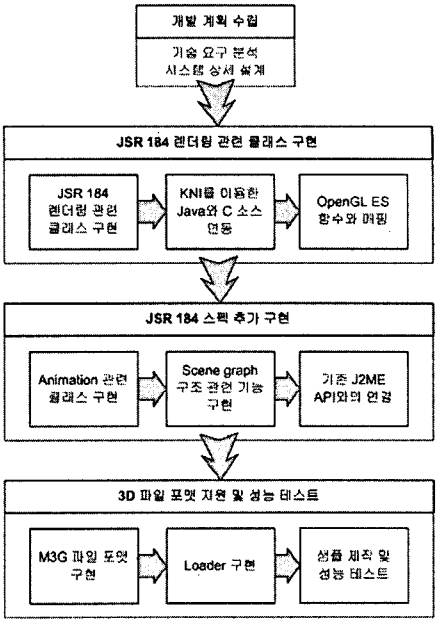

그림 3. JSR-184를 이용한 3D 엔진구현과정

4.2 스킨드 메시 모델 처리향상을 위한 방법

스킨드 애니메이션에서는 하나의 메시(Mesh)를 사용하고, 여러 번의 변환 가중치 변화에 따른 가중치 결과를 혼합하여 사용하며, 정점(Vertex)들은 영향을 받는 뼈와 가중치를 가지고 있어야 한다. 단, 가중치의 합은 1(100%)이다.

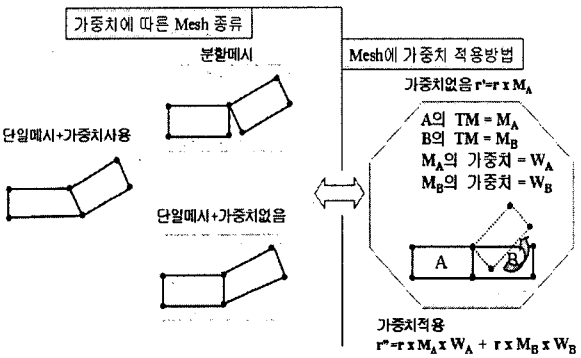

그림 4. 스킨드 애니메이션 결과값 비교

그림 4를 보면 알 수 있듯이, A와 B라는 부분의 애니메이션이 달라질 경우, 분할 메시의 경우에는 별다른 방법이 없는 것을 알 수 있다. 그러나, "단일 메시+가중치 없음"의 경우에는 비교적 그럴듯한 결과 값을 얻을 수 있으나 관절부위가 자연스럽지 못한 것을 알 수 있다. 마지막, "단일 메시+가중치 사용"의 경우에는 가장 이상

적인 형태로 A의 애니메이션 행렬 M_A와 B의 애니메이션 행렬 M_B가 각각 가중치 W_A와 W_B값에 의해서 결합된 후, 최종값을 누적해서 적용되는 것을 알 수 있다.

이와 같이 연결 부분의 정점은 두개의 좌표계를 함께 사용하며, 각각의 결과를 적절히 혼합하여 사용하며, 이를 수식으로 표현하면 다음과 같다.

$$\begin{pmatrix} Rx \\ Ry \\ Rz \end{pmatrix} = WM \begin{pmatrix} Ax \\ Ay \\ Az \end{pmatrix} + (1-W)M \begin{pmatrix} Bx \\ By \\ Bz \end{pmatrix}$$

수식 1. 스킨드 메시 적용수식

수식 1에서 W는 정점 가중치를 의미하며, 해당 정점이 양쪽 좌표계에 동일하게 의존할 경우 각각 W=0.5를 적용한다, 한편, 뼈대 다른 끝 부분의 경우 정점 가중치는 0 또는 1이 되어 하나의 좌표계에만 의존하게 된다.

4.3 스킨드 메시 적용 알고리즘

본 논문의 스킨드 메시 오브젝트에 적용한 알고리즘은 다음과 같다.

```
//Deform 관련 정보저장
int          weightList[vertex 개수][bone 개수]
int          weightSum[vertex 개수]
Transform    toBone[bone 개수]
Transform    positionTransform[bone 개수]
Transform    normalTransform[bone 개수]
Vector       boneList

addTransform()
{
    해당 위치의 weightList에 weight값 누적.
    weightSum 누적하여 새로 계산.
}
morph()
{
   for(int i = 0; i < bone 개수 i++)
   {
       positionTransform 계산.
       normalTransform 계산.
    }
   for(int i = 0; i < vertex 개수 i++)
   {
       for(int j = 0; j < bone 개수 j++)
       {
          weightSum과 transform을 이용하여
          vertex position과 normal deform.
       }
   }
}
```

기본적으로 스킨드 메시 오브젝트 처리시, 그림 3과 같은 계산이 실시간으로 처리되어야 정확한 움직임에 자연스러운 메시와 메시 사이의 연결부분을 적용할 수 있다. 즉, 각각의 정점(Vertex)들은 어느 본에 대해 얼마만큼의 가중치 영향을 받는지에 대한 정보와 각각의 정점(Vertex)별로 모든 본의 가중치값들을 실시간으로 계산하여 렌더링(Rendering)시마다 계산값을 적용하는 것이 아니라, 최종 로딩(Loading)시에 미리 계산된 값을 한번만 적용해서 처리하도록 하여, (주)노키아(Nokia) JSR-184 엔진의 스킨드 메시 적용 알고리즘에서 가중치 값을 정규화할 때 비효율적이고 메모리 사용량이 늘어나고 렌더링 시간이 늘어나는 문제점을 개선하였다.

5. 실험 결과

그림 5는 JSR-184로 구현한 엔진으로써 3D MAX 7.0을 사용해서 M3G 파일 포맷으로 스킨드 메시를 적용해서 제작한 소년이 스케이트보드를 타고 묘기부리는 모습의 실행한 화면이다.

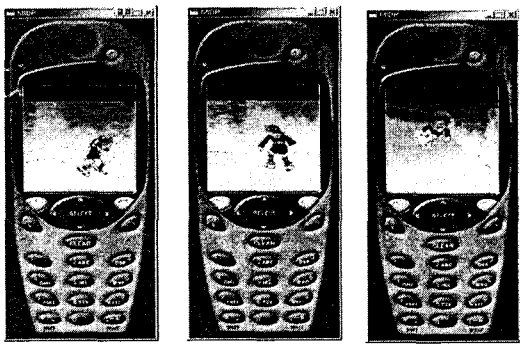

그림 5. JSR-184 엔진을 적용한 실행화면

6. 결론

본 논문에서는 J2ME 환경에서 모바일 3D 그래픽스 표준인 JSR-184를 이용하여 유럽을 중심으로 미국과 중국 등의 전 세계 이동통신 시장의 70%를 차지하고 있는 GSM 시장에서 상용화할 수 있는 모바일 3D 엔진을 설계 및 구현하였다.

향후 연구방향으로는 JSR-184에서 처리되는 모든 그래픽 처리단계에서의 최적화에 관한 기법들에 대해서 연구할 계획이다.

7. 참고문헌

[1] James.Dong.L and Twigg.Christopher.D, "Skinning Mesh Animations," ACM Trans, Vol.24, No.03, pp.399-407, 2005.7.

[2] 유소란, "모바일 게임 시장 및 개발동향," 정보처리학회논문지, 제 9권, 제 3호, pp.42-49, 2002.5.

[3] J.W.Muchow, "Core J2ME Techonology & MIDP," Prentice Hall, 2001.

[4] 김용준, "3D 게임 프로그래밍," 한빛미디어, 2004.

[5] http://www.forum.nokia.com/java/jsr184