자동 동작 생성 엔진을 이용한 게임 개발

송지훈*, 박창준**, 강영민**, 옥수열**
*동명대학교 컴퓨터공학과
**동명대학교 게임공학과
czgolmae@naver.com, nom00su@nate.com, sooyol@tu.ac.kr, ymkang@tu.ac.kr

Game Development with Automated Motion Generation Engine

JiHun Song*, ChangJun Park**, SooYol Ok**, YoungMinKang**
*Dept. Computer Engineering, Tongmyoung Univ.
**Dept. Game Engineering, Tongmyoung Univ.

요 약

인간형 캐릭터의 보행 동작에 대하여 고수준 개념의 매개변수 제어를 통하여 다양한 스타일의 특성을 얻을 수 있는 자동 동작 생성 엔진과 Steve 'sinbad' Streeting이 시작한 장면 기반의 오픈 소스 그래픽 렌더링 엔진인 OGRE[1]를 이용한 게임 제작 과정에서 나타나는 여러 가지 문제들을 해결하는 방법을 제안하고 있다. 아울러 본 논문에서는 게임 제작에 활용된 자동 동작 생성 엔진과 렌더링 엔진 들 사이의 데이터 변환 문제를 해결하고 효과적으로 게임을 제작할 수 있는 기술을 소개하고 있다.

1. 서론

게임 엔진은 컴퓨터·비디오 게임이나 실시간 그래픽 표시 기능을 갖춘 상호작용 어플리케이션을 구현하는 핵심 소프트웨어 구성 요소를 말한다. 또한 개발에 있어서 공통적이고 자주 쓰이는 하부 작업들을 대신해 줄 뿐 아니라 고급 게임 기술들을 손쉽게 사용할 수 있도록 해주기 때문에 게임 개발 시 많은 시간과 노력을 절감시켜 주는 역할을 한다[2].

일반적으로 모든 엔진에는 자체의 모델링 데이터 형식이 존재 하며 그 엔진에 맞는 모델링 데이터만을 사용할 수 있다. 본 논문에서는 현재 가장 많이 사용하는 오픈 소스 그래픽 렌더링 엔진[3]인 OGRE에 자동 동작 생성 엔진으로 생성된 모델링 데이터를 변환하여 적용하는 방법을 다룬다.

2. 자동 동작 생성 엔진의 적용

본 논문의 게임 제작 환경에서 사용된 동작 생성 엔진은 본 연구진이 자체 개발한 캐릭터 동작 자동 생성 엔진으로서, 게임 캐릭터의 보행 동작을 나이, 성별, 피로도와 같은 다양한 고수준 제어 파라미터로 생성할 수 있는 엔진이다. 이 자동 동작 생성 엔진은 FBX 파일 형식을 기반으로 동작하고 있다. 그렇기 때문에 FBX 파일 형식을 바로 사용할 수 없는 환경에서는 자동 동작 생성 엔진의 결과물을 목표 환경에 맞게 변환할 필요가 있다.

변환의 방법에는 크게 두 가지 방법이 있다. 첫 번째 방법으로는 자동 동작 생성 엔진 자체를 목표로 하는 환경에서 요구하는 파일 형식에 맞춰 변환

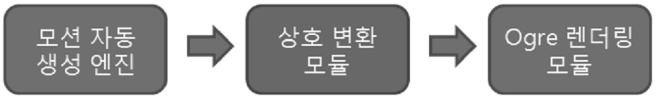
을 하는 것이다. 하지만 이 방법은 엔진을 새롭게 구성하고 구현하여야하는 큰 단점이 생긴다. 또한 오픈소스의 공개된 엔진이 아닌 경우 엔진의 직접적인 수정이 불가능한 경우가 많기 때문에 이 방법 자체가 불가능 할 수 있다. 반면에 엔진의 직접적인 수정이 가능하면 목표 환경에 대한 최적화를 할 수 있으므로 속도에 많은 이득을 볼 수 있다.

두 번째 방법은 자동 동작 생성 엔진은 그대로 두고 자동 동작 생성 엔진에서 나온 결과물, 즉 변경된 정점과 뼈대의 정보를 실시간으로 목표 환경의 렌더링 엔진에 맞는 정점과 뼈대 정보로 변환하여 출력하는 것이다. 이 때 자동 동작 생성 엔진에서 나온 정점을 목표 환경에 맞게 변환하는 '실시간 상호 변환 모듈'이 필요하다. 첫 번째 방법에 비해 두 번째 방법은 실시간 상호 변환 모듈이라는 단계를 한 번 더 거쳐야하기 때문에 속도에 손실을 입을 수밖에 없는 단점이 있다.

	모션 엔진 자체를 변환하는 방법	실시간 상호 변환 모듈을 통한 방법
장점	•목표 환경에 대한 최적화가 가능 •빠른 속도	•엔진 수정 불필요 •실시간 상호 변환 모듈만 필요
단점	•새롭게 엔진을 구성할 필요	•추가적인 변환 과정에 따른 속도 느려짐 현상 발생

[표 1 – 자동 동작 생성 엔진을 렌더링 엔진에 적용하는 방법에 따른 장•단점]

본 논문에서는 실시간 상호 변환 모듈의 수정만으로 자동 동작 생성 엔진을 OGRE뿐만 아니라 다른 그래픽 렌더링 엔진에도 적용할 수 있는 장점 때문에 두 번째 방법을 이용하여 목표로 하는 환경인 OGRE에 동작 자동 생성 엔진을 적용하여 게임을 제작하였다.



[그림 2 – 모션 엔진과 Ogre의 연동]

3. 자동 동작 생성 엔진과 OGRE

3.1 실시간 상호 변환 모듈

자동 동작 생성 엔진의 기반인 FBX에서 변형된(deform) 정점들을 목표 환경인 OGRE의 Mesh 정보로 변환하는 것이 실시간 상호 변환 모듈의 목적이다. 변환될 대상인 정점 하나에는 위치, 법선 벡터, UV 텍스처 좌표 정보가 하나의 조합을 이루어야 한다. 또한 각 폴리곤을 이루는 정점들의 인덱스 정보도 변환이 필요하다.

3.2 자동 동작 생성 엔진과 OGRE의 정점 데이터

자동 동작 생성 엔진에서 사용하는 FBX에서 정점이 저장되는 방식은 크게 두 가지로 볼 수 있다. 먼저 제어점(control point)마다 법선 벡터와 UV 텍스처 좌표가 1 대 1로 대응하는 방식이다. 이 방식은 제어점과 법선 벡터, UV 텍스처 좌표의 수가 모두 같으며, 제어점의 순서를 정의한 인덱스와 UV 텍스처 인덱스가 동일하다. 이러한 방식을 FBX에서는 By-Control-Point(이하 CP방식) 라고 부른다.

두 번째 방식은 UV 텍스처 좌표가 제어점에 대응하는 것이 아니라 폴리곤에 대응한다. 그래서 제어점과 UV 텍스처 좌표의 수가 동일하지 않으며, 제어점의 인덱스와 UV 텍스처의 인덱스가 달라서 각각 인덱스 버퍼를 따로 둔다. 이 방식은 제어점은 제어점대로, UV 텍스처는 UV 텍스처대로 중복되는 점들을 추려낼 수 있으므로 점의 수를 줄일 수 있다. 이러한 방식을 FBX에서는 By-Polygon-Vertex(이하 PV방식) 라고 부른다.

By-Control-Point	By-Polygon-Vertex
•제어점과 UV가 1:1 대응 •제어점과 UV의 수가 같음 •제어점의 인덱스와 UV의 인덱스 동일	•UV가 폴리곤에 대응 •제어점과 UV의 수가 다름 •제어점의 인덱스와 UV의 인덱스 버퍼를 각각 따로 둠

[표 2 – FBX에서의 정점이 저장되는 방식 비교]

OGRE에서는 제어점과 법선벡터, UV 텍스처 좌표가 하나의 조합을 이루어서 저장이 된다. 그렇기 때문에 인덱스 버퍼는 하나만 있으면 된다. 즉 FBX의 CP 방식과 동일하다.

앞에서 살펴본 바와 같이 FBX와 OGRE에는 동일한 정점 데이터 저장 방식이 있다. 이러한 점을 이용하여 실시간 상호 변환 모듈을 구성할 수 있다.

4. 정점 데이터의 변환 과정

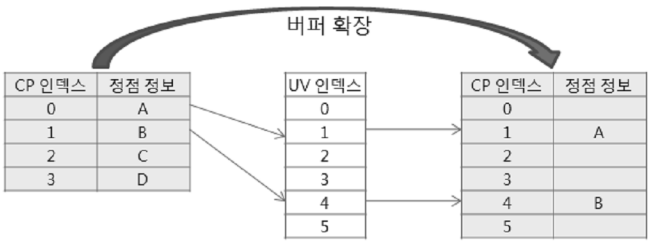
MAX 또는 MAYA에서 제작된 캐릭터를 FBX

로 변환하게 되면 위에서 살펴본 대로 두 가지 방식 중 최적화된 방식으로 저장된다.

실시간 상호 변환 모듈에서 이 두 가지 방법을 모두 지원하게 되면, PV 방식인 경우에는 OGRE에서 CP 방식만을 사용하므로, FBX에서 PV 방식으로 저장된 정점 데이터들은 CP 방식으로 변환하는 단계가 추가적으로 필요하게 된다. 변환 모듈이 실시간 변환임을 생각한다면 매 프레임마다 또 다른 추가적인 변환을 거치게끔 하는 것은 속도에 많은 악영향을 끼치게 된다. 그러므로 PV 방식의 FBX 파일은 CP 방식으로, 게임상의 메인 루프가 아닌 로딩 단계에서 먼저 바꾼 후에 사용하여야 한다. 이렇게 바꾼 CP 방식의 FBX 파일을 실시간 상호 변환 모듈에서 OGRE의 정점 데이터로 변환하여 사용한다.

4.1 PV 방식을 CP 방식으로 변환

통상적으로 PV 방식은 제어점의 수보다 UV 텍스처 좌표의 수가 많거나 같다. 이 사실을 이용하면 PV 방식에서 CP 방식으로의 변환은 제어점을 UV의 인덱스에 대응시키는 것으로 쉽게 해결할 수 있다.



[그림 3 – PV 방식을 CP 방식으로 변환]

4.2 OGRE의 ManualObject 생성

FBX에서 변형된 점을 OGRE에서 사용할 공간인 ManualObject를 생성한다. ManualObject의 생성은 OGRE의 SceneManager를 통해 가능하다. ManualObject는 내부적으로 하드웨어 버퍼를 이용하여 정점과 인덱스 데이터를 관리한다.

```
// create manual object
ManualObject* pManualObject =
      m_pOgreSceneManager->createManualObject(name);

pManualObject->begin(materialName.Buffer(),
                     RenderOperation::OT_TRIANGLE_LIST);

(정점 정보 및 인덱스 정보를 저장하는 부분)

pManualObject->end();
```

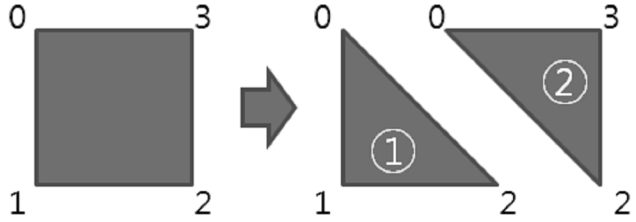
[표 3 – OGRE의 ManualObject를 생성하는 소스]

4.3 정점 정보 변환

정점 정보는 FBX와 OGRE가 동일한 저장 방식을 사용하므로 매우 간단하게 변환이 가능하다. FBX에서 변형된 정점 데이터의 버퍼를 얻어 와서 버퍼에 저장된 순서대로 Ogre의 ManualObject에 그대로 저장하면 된다.

4.4 인덱스 정보 변환

FBX에서의 폴리곤 하나는 3개의 정점으로만 이루어진 것이 아니다. 즉 삼각형 또는 사각형으로 폴리곤이 구성되어 있다. 하지만 Ogre에서는 삼각형으로만 폴리곤을 구성하므로 FBX의 폴리곤을 삼각형으로 나누는 작업이 필요하다.



[그림 2 – 4각 폴리곤을 3각 폴리곤으로 나눔]

```
void ConvertTriangleStripToTriangleList(KFbxMesh* pMesh,
  int nPolygonIndex, unsigned short faces[4][3],
  int& nFacesCounts)
{
  int nVerticeCount = pMesh->GetPolygonSize(nPolygonIndex);
  nFacesCounts = nVerticeCount - 2;

  for (int nVerticeIndex = 1;
       nVerticeIndex <= nVerticeCount - 2; ++nVerticeIndex)
  {
    faces[nVerticeIndex - 1][0] =
      pMesh->GetPolygonVertex(nPolygonIndex, 0);
    faces[nVerticeIndex - 1][1] =
      pMesh->GetPolygonVertex(nPolygonIndex, nVerticeIndex);
    faces[nVerticeIndex - 1][2] =
      pMesh->GetPolygonVertex(nPolygonIndex, nVerticeIndex+1);
  }
}
```

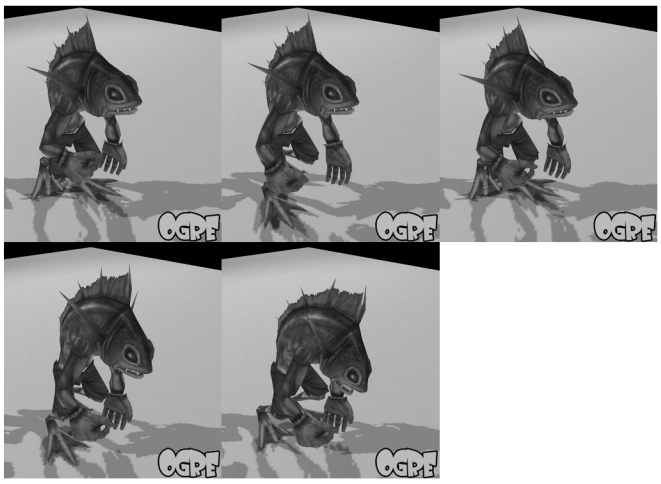
[표 4 – 삼각형 두 개로 나누는 소스]

표 4의 알고리즘으로 새롭게 구성된 인덱스를 OGRE의 하드웨어 인덱스 버퍼에 차례대로 저장하면 된다.

5. 자동 동작 생성 엔진을 OGRE에 적용한 결과

다음은 자동 동작 생성 엔진과 OGRE를 이용하여 3인칭 액션 게임을 제작한 결과 화면이다. 주인공

의 체력 수치에 따라 자동 동작 생성 엔진에서 사용하는 고수준 매개변수의 하나인 '피로도' 수치의 변화로 동작이 변화하는 것을 볼 수 있다.



[그림 3 – '피로도'에 따른 동작 변화]

[그림 4 – 자동 동작 생성 엔진의 게임 적용 결과]

6. 결론

본 논문에서는 자동 동작 생성 엔진과 OGRE를 이용한 게임 제작에 있어서 자동 동작 생성 엔진의 결과를 목표 환경인 OGRE에 적용하는 기술을 제안하고 있으며, 개발된 실시간 상호 변환 모듈을 소개하고 있다.

현재의 실시간 상호 변환 모듈은 변환될 대상을 각각의 정점으로 하고 있으므로 한 번에 변환될 데이터의 양이 많다. 향후 변환될 데이터의 양을 줄임으로써 속도를 빠르게 할 필요성이 있다.

한 가지 방법으로 변환될 데이터를 각각의 정점이 아닌 뼈대의 회전 및 이동 값으로 하면 변환될 데이터가 상당히 많이 줄어들게 된다. 뼈대에 속한 정점들을 일일이 변환하지 않고 뼈대 하나만 변환하면 되기 때문이다. 하지만 이 방법을 사용하기 위해서는 자동 동작 생성 엔진에서 사용하는 뼈대와 OGRE의 뼈대를 동일하게 조작 할 수 있는 기술이 필요하다. 우리는 이 기술을 사용하여 실시간 상호 변환 모듈의 개선을 계획하고 있다.

본 논문은 2007년도 지역문화산업연구센터(CRC)지원 사업에 의하여 이루진 것입니다.

[참고문헌]

[1] OGRE 공식 홈페이지, "About - What Is OGRE?", http://www.ogre3d.org

[2] 김혜선, "플랫폼 간 경계 허물기" DIGITAR CONTENT, APR 2006.

[3] 게임엔진데이터베이스, http://www.devmaster.net/engines/