파티클을 이용한 로고 애니메이션 제작 사례 연구

A Case Study of Making Logo Animation Using Particles

정재민, 오규환, 석혜정
아주대학교 미디어학과

Jai-Min Jung(sinclair@ajou.ac.kr), Gyu-Hwan Oh(drghoh@ajou.ac.kr),
Hae-Jung Suk(dbdip@ajou.ac.kr)

요약

본 논문에서는 파티클을 이용하여 시각적으로 화려하고 역동적인 느낌을 주는 로고 애니메이션의 제작 과정을 사례 연구를 통해 정리한다. 2004년 토리노 동계 올림픽 홍보 동영상을 소스로 하여 3D 컴퓨터 그래픽스 툴인 마야에서 다양한 기법을 적용하여 이와 비슷한 비주얼을 주는 영상을 제작한다. 스케이트를 타는 동작을 역동적으로 표현한 이 애니메이션의 효과를 내기 위해 본 논문에서는 인체 모델을 큐브로 잘게 나누고, 잘게 나뉜 큐브가 모델의 움직임에 따라 역동적으로 움직이도록 구현한다. MEL 스크립트로 구현된 UI에서 옵션 값을 조절함으로서 다양한 비주얼적인 효과를 낼 수 있다.

■ 중심어 : | 마야 | 파티클 | 인스턴서 |

Abstract

In this paper, we present a case of making LOGO animation using particles with MAYA, a 3D computer graphics packages of Autodesk Inc. We composite a visual which shows a similar effects with a information movie of Torino 2006 Winter Olympic Games. By analysing the movie, we model a human body part as a set of cubes and animated the cubes to have dynamic visuals which shows similar visuals with the movie. All the system is implemented with MAYA MEL scripts. The system shows various visual effects by controlling options available in the designed UI.

■ keyword : | MAYA | Particle | Instancer |

I. 서 론

1. 연구개요

오늘날 영상과 관련된 기술이 발전하고 이에 따른 영상물 수용자들의 수준이 높아짐에 따라, 단순히 현실세계에 존재하는 모습을 화면에 담아내는 것만으로는 충분한 시각적 표현을 해낼 수 없게 되었다. 그래서 최근 광고나 영화 등의 영상물에서는 3D 컴퓨터 그래픽스 툴을 이용하여 현실세계에 존재할 수 없는 시각적으로 화려하고 역동적인 느낌을 주는 영상을 많이 제작하고 있다.

본 논문에서는 파티클(Particle)을 이용하여 시각적으로 화려하고 역동적인 느낌을 주는 로고 애니메이션의

* 이 논문은 2005학년도 1학기 아주대학교 정착연구비 지원에 의하여 연구되었습니다.

접수번호 : #060901-001
접수일자 : 2006년09월01일
심사완료일 : 2006년09월19일
교신저자 : 오규환, e-mail : drghoh@ajou.ac.kr

제작 과정을 사례 연구를 통해 정리한다. 2004년 토리노 동계 올림픽 홍보 동영상을 소스로 하여 3D 컴퓨터 그래픽스 툴인 '마야(MAYA)[1,2]'에서 다양한 기법을 적용하여 홍보 동영상과 비슷한 비주얼(Visual)을 주는 영상을 제작 한다. 스케이트를 타는 동작을 역동적으로 표현한 이 애니메이션의 효과를 내기 위해 본 논문에서는 인체 모델을 큐브(Cube)로 잘게 나누고, 잘게 나뉜 큐브가 모델의 움직임에 따라 역동적으로 움직이도록 구현한다. 움직임을 주고자 하는 모델에 사용자가 원하는 오브젝트(Object)를 배치하고, 파티클과 인스턴서(Instancer)를 이용하여 역동성을 구현하도록 한다. MEL(Maya Embedded Language) 스크립트[3]로 구현이 이루어지고 제작된 UI(User Interface)에서 다양한 옵션 값을 조절함으로써 비주얼적인 효과를 제어하고 선택할 수 있도록 구현한다.

2. 소스(Source)영상

[그림 1]은 본 논문에서 참고한 영상의 이미지 시퀀스(sequence)를 보여준다. 이 영상은 인간의 형상을 한 모델이 스케이트를 타고 질주하는 동작에서 큐브 모양의 물체가 흩날리도록 함으로써 동작의 역동성과 화려함을 보여 준다.

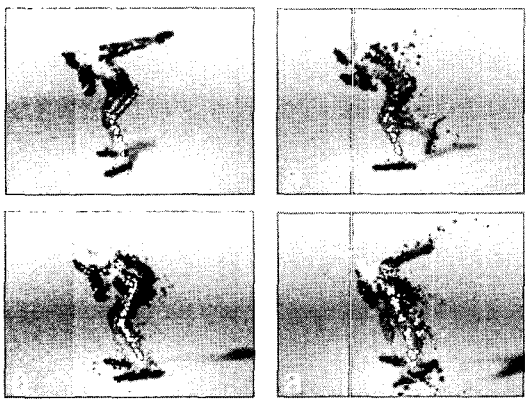

그림 1. 소스 동영상 시퀀스

II. 접근 방법

소스 동영상에서 알 수 있듯이 인체 모델에 있는 큐브는 인체 모델 표면에 사람의 형상을 보여주는 큐브와 인체 표면에서 달리는 방향과 반대의 방향으로 흩날리는 큐브로 구성된다. 본 논문에서는 이 두 종류의 큐브를 구분하고 각각을 모델링, 애니메이션, 렌더링 하는 방법을 제안한다.

이를 구현하기 위해 본 논문에서는 소스 동영상을 바탕으로 이와 유사한 형태와 애니메이션을 나타내는 3D 인체 모델 데이터를 만든다. 만들어진 인체 데이터의 버텍스(Vertex)에 큐브를 배치한다. 또한, 흩날리는 모습을 표현하기 위해 모델의 표면에서 분출되는 오브젝트를 대응시키고, 애니메이션이 될 때 모델의 버텍스 상의 큐브의 움직임과, 큐브의 모양으로 흩날리는 모습을 역동적으로 보여줄 수 있도록 구현한다. 마지막으로 3D 인체 모델을 렌더링 하고, 인체 모델위의 큐브들과 흩날리는 큐브들을 인체 모델의 칼라를 기반으로 렌더링을 함으로써 사실적이면서 역동적인 애니메이션을 만들도록 한다.

III. 3D 인체 모델링 & 애니메이션

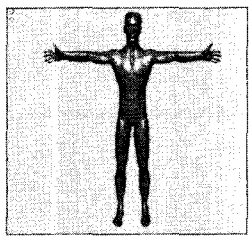

그림 2. 사용된 3D 인체 모델링 소스 데이터

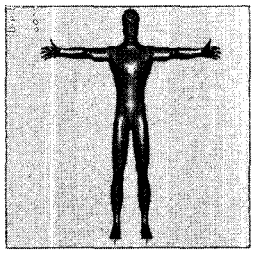

그림 3. 수정된 인체 모델링 데이터

제작에 필요한 3D 인체 모델 데이터는 뷰포인트(ViewPoint) 사의 DataShop Premier[4]에서 적합한 소스 인체 모델을 찾은 후, 모델의 버텍스의 개수를 줄이고 모델의 외형을 단순화 시켜 애니메이션을 생성하는데 효과적이 되도록 모델 데이터를 단순화한다. 스케이팅을 하는 동작 애니메이션은 수작업을 통해 '마야'에서 이루어 졌다. [그림 2]와 [그림 3]은 각각 제작에 사용한 3D 인체 모델링 소스 데이터와 이 작업을 위해 수정한 인체 모델링 데이터를 각각 보여준다. [그림 4]는 애니메이션 작업을 완료한 후 스케이팅을 하는 동작을 보여준다.

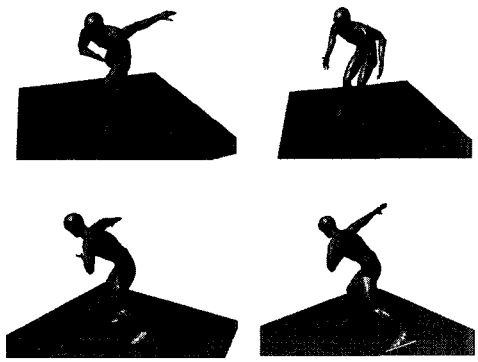

그림 4. 스케이팅 동작 애니메이션

IV. 인체 모델에 큐브 배치

인체 모델의 큐브의 배치는 인체 모델 표면에 사람의 형상을 유지하는 큐브와 인체 표면에서 달리는 방향과 반대의 방향으로 흩날리는 큐브로 구분이 가능하다. 이 장에서는 이 두 종류의 큐브를 모델링 하는 방법은 설명한다.

1. 모델 표면에 큐브 배치

인체 모델 표면에 큐브의 배치는 인체 모델을 구성하는 다면체의 모든 버텍스에 큐브의 인스턴스를 배치하는 방식으로 구현된다. 다음은 모델 표면에 큐브를 배치하는 MEL 스크립트이다.

```
string $obj[]= `ls -sl`;
int $vertexCount[] = `polyEvaluate -v $obj[0]`;
int $i;
float $vert[];
for( $i=0;$i<$vertexCount[0];$i++) {
    $vert = `pointPosition ($obj[0]+".vtx["+$i+"]")`;
    select source1;
    instance -n ("target"+$i);
    move $vert[0] $vert[1] $vert[2];
}
```

[그림 5]는 이러한 방식으로 인체 모델에 큐브 배치한 모습을 보여준다.

그림 5. 인체 모델에 큐브 대응

2. 모델 표면에서 흩날리는 큐브 세팅

인체 모델의 표면에서 큐브가 흩날리는 효과는 인체 모델 표면을 구성하는 다면체의 버텍스에 에미터(Emitter)를 배치하고, 에미터에서 분출되는 파티클에 큐브를 인스턴서(Instancer)에 매핑함으로써 이루어진다. 자연스러운 흩날림을 표현하기 위해 인체 모델의 어떤 버텍스에 에미터를 링크할 것인지에 대한 결정은 확률값을 통해 결정한다(뒷부분 UI 메뉴 참조). [그림 6]은 인체 모델의 확률로 정해진 일부 버텍스에 놓인 에미팅 파티클을 보여준다.

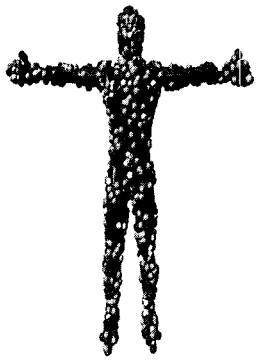

그림 6. 인체 모델의 일부 버텍스의 에미팅 파티클

'마야'에서 제공하는 인스턴서는 파티클을 특정 오브젝트로 매핑하는 명령으로 이 명령을 사용하여 파티클을 큐브로 대응하도록 한다. [그림 7]은 삼차원 좌표에서 원점에 에미터 생성 후, 파티클에 큐브 인스턴서를 대응한 결과를 보여준다. 이러한 방식으로 모델 표면에서 큐브가 흩날리는 효과를 내기 위해 에미터의 배치가 결정된 인체 모델의 모든 버텍스에 큐브 인스턴서를 대응하게 된다.

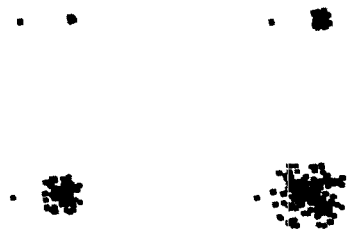

그림 7. 에미터 생성 후, 파티클에 큐브 인스턴서 적용

V. 애니메이션

1. 모델위의 큐브 애니메이션

모델위의 큐브애니메이션은 인체 모델의 애니메이션마다 인체 모델을 구성하는 다면체의 모든 버텍스에 대응하는 큐브의 위치를 재계산하여 그려주는 방식으로 이루어진다. 위치의 재계산을 위해 두 가지 방법을 제안한다.

첫 번째는 동작이 이루어지는 모든 프레임에서 매 프레임마다 인체 모델을 구성하는 다면체의 버텍스의 위치 정보를 미리 계산 한 후, 해당 프레임을 그릴 때 계산해 놓은 좌표를 이용하는 방식이다. 이 방식은 적은 수의 다면체로 만들어진 모델의 경우에는 빠른 결과물을 만드는 것을 보장하지만 미리 모든 위치 정보를 계산하는 과정에서 CPU시간 및 메모리를 많이 차지하는 단점을 가진다. 이러한 방식을 Intel 1.83GHz Dual Core CPU, 1G의 메모리를 가진 컴퓨터에서 버텍스 2500개, 150 프레임 인체 모델에 적용할 경우, 1.65GBytes의 메모리를 소비하면서 최종 결과물을 만드는데 약 30분 정도 걸린다.

두 번째는 현재 프레임에서 그 다음에 그려져야 할 프레임을 계산하는 방식이다. 이러한 방식은 매 프레임이 진행될 때마다 계산을 하기 때문에 계산에 걸리는 부하를 분산 할 수 있고 '마야'에서 익스프레션(expression)을 이용하면 다음과 같이 구현된다. 여기서 사용되는 인체 모델을 'model'로 가정한다.

```
string $mName = "model";
int $vertexCount[] = `polyEvaluate -v $mName`;
string $targetNamePrefix = "target";
for( $i=0;$i<$vertexCount[0];$i++)
{
    $vert = `pointPosition ($mName+".vtx["+$i+"]")`;
    if(`objExists ($targetNamePrefix+$i)`)
    {
        select ($targetNamePrefix+$i);
        move $vert[0] $vert[1] $vert[2];
    }
}
```

[그림 8]은 위와 같은 방식으로 계산하여 얻은 인체 모델위의 큐브 애니메이션을 보여준다.

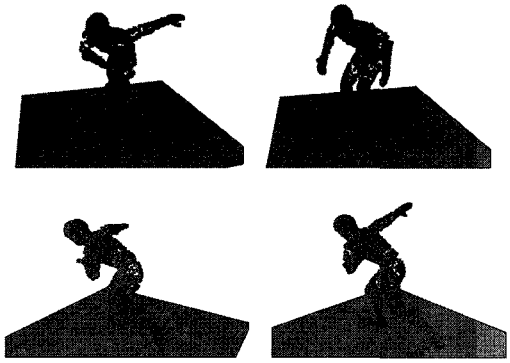

그림 8. 인체 모델위의 큐브 애니메이션

2. 모델 표면에서 흩날리는 큐브 애니메이션

모델 표면에서 흩날리는 큐브의 애니메이션은 재 프레임을 기준으로 이전 프레임과 이후 프레임 간의 내적을 계산해 속도 벡터를 산출하고 이 값으로 큐브 인스턴서의 운동을 결정하게 된다. [그림 9]는 이러한 방식으로 모델 표면에서 흩날리는 큐브의 애니메이션을 계산한 결과를 보여준다.

그림 9. 인체 모델 표면에서 흩날리는 큐브 애니메이션

3. 애니메이션 중간 결과물

앞에서 설명한 방식으로 인체 모델 상의 큐브들의 애니메이션과 인체 모델위에서 흩날리는 큐브 애니메이션을 동시에 계산하여 보여주면 인체 모델이 스케이팅을 하면서 큐브들이 움직이고 큐브들이 역동적으로 흩날리는 애니메이션의 생성이 가능하다. [그림 10]은 이러한 방식으로 구현이 이루어진 애니메이션 결과를 보여준다.

그림 10. 애니메이션 중간 결과

VI. 컬러링(Coloring)

애니메이션의 색깔을 결정하기 위해 먼저, 인체 모델의 다면체에 매핑이 되는 텍스쳐(Texture)를 UV공간에서 생성한다. [그림 11]은 컬러링을 위해 만들어진 UV 공간에서의 텍스쳐를 보여준다. 생성된 텍스쳐를 이용하여 칼라를 결정하기 위해 본 논문에서는 단순화된 칼라에 적합하도록 RGB값을 정량화(Quantization)하여 27개의 쉐이더(Shader)를 생성 한 후, 이 쉐이더를 이용하여 인체 표면의 큐브를 컬러링 한다. 인체 표면에서 흩날리는 큐브의 컬러링을 위해 27개에 해당하는 쉐이더를 이용하여 27개의 인스턴서를 생성하고 이를 이용하여 파티클의 색에 해당하는 큐브 인스턴서를 대응시켜 준다.

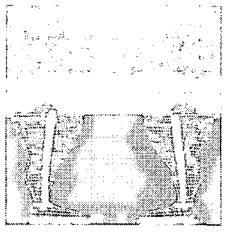

그림 11. 컬러링을 위해 uv 공간에서 정의된 텍스쳐

1. 쉐이더 생성

본 논문에서는 0.0에서 1.0 사이의 값을 가지는 RGB 에 대해 다음의 정량화를 통해 27개의 쉐이더를 생성한다.

l 0.0~0.3 의 값은 0.0 에 대응.

l 0.3~0.7 의 값은 0.6 에 대응.

l 0.7~1.0 의 값은 1.0 에 대응.

이러한 정량화는 모델의 색 결정시, 텍스쳐 상의 색을 가져와서 어두운 부분은 더 어둡게 하고, 밝은 부분은 더 밝게 해주어 소스 동영상의 느낌을 주도록 한다. [그림 12]는 만들어진 27개의 쉐이더를 보여준다.

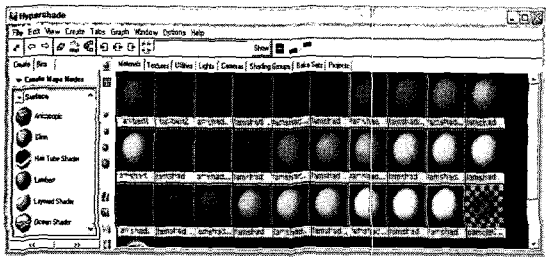

그림 12. 27개의 쉐이더

2. 모델 표면의 큐브 컬러링

인체 모델의 모든 버텍스에 큐브를 배치하고, 모델의 버텍스 색을 텍스쳐가 정의되는 해당 UV 좌표에서 가져온다. [그림 13]은 인체 모델에 배치된 큐브에 맞는 색을 적용한 결과이다.

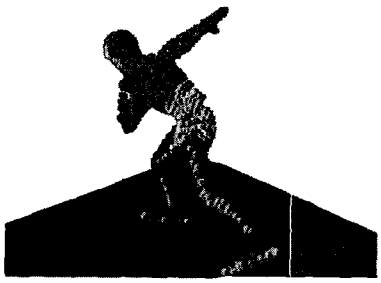

그림 13. 모델 표면의 큐브 컬러링

2. 모델 표면에서 흩날리는 큐브 컬러링

인체 모델 표면에서 흩날리는 큐브의 경우, 먼저 27개의 인스턴서가 생성되고, 각 에미터에서 뿜어져 나올 파티클의 색깔에 따라 각 파티클과 어떤 인스턴서가 연결될지 결정해 준다. 다음은 모델 표면에서 분출되는 큐브의 컬러링 부분의 코드이다. [그림 14]는 모델 표면에서 흩날리는 큐브의 컬러링 결과이다.

그림 14. 모델 표면에서 흩날리는 큐브 컬러링

VII. UI 제작

본 논문에서는 UI를 '마야'상에서 제작함으로서 구현된 내용을 효과적으로 제어하도록 한다. 만들어진 UI에서는 대상이 되는 모델의 이름, 모델 위에 배치되는 오브젝트의 이름, 모델이 갖는 텍스쳐의 이름을 입력으로 받아 여러 가지 선택 사양을 정한 후, 결과 애니메이션을 바로 만들어 결과에 대한 품질을 확인함으로써 대화방식으로 만들어지는 결과에 대한 수정이 가능하다. 제작된 UI는 [그림 15]와 같고, UI상에 있는 기능을 정리하면 다음과 같다.

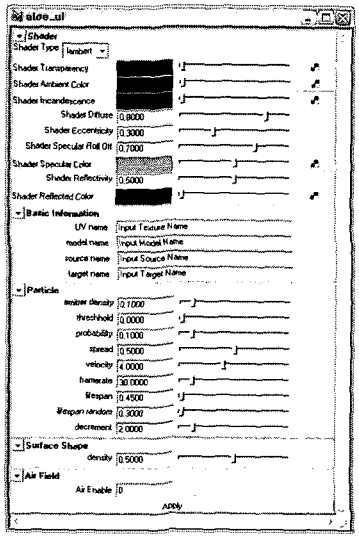

그림 15. 제작된 UI

1. Shader

생성될 쉐이더에 대한 내용을 제어한다. 생성되는 쉐이더는 총 27개로써, 일종의 팔레트를 생성하게 된다.

1.1 Shader Type

쉐이더의 종류를 결정한다. lambert와 blinn 중 하나를 선택한다.

1.2 Shader Option

일반적인 마야에서 지원하는 쉐이더의 옵션을 그대로 적용가능하다. Lambert의 경우 Diffuse까지만 적용되며, 그 이후의 옵션 값은 무시된다.

2. Basic Information

효과를 적용할 모델의 이름, 모델에 적용되어 있는 텍스쳐 이름(UV name), 결과를 적용시킬 모델 이름(model name), 모델의 vertex에 배치시킬 object의 이름(source name), 생성된 모델의 instance들의 이름(target name)을 입력 받는다.

3. Particle

모델의 움직임, 잔상 등의 표현의 옵션을 입력하는 부분이다.

3.1 emitter density

모델의 버텍스마다 생성할 에미터의 확률을 입력하는 부분이다. 0을 입력할 경우 에미터는 전혀 생성되지 않으며, 1을 입력할 경우 모든 버텍스에 에미터를 생성한다.

3.2 threshold

모델의 움직임의 정도에 따라 분출되지 않는 정도를 입력하는 부분이다. 속도 벡터가 threshold 보다 작을 경우 파티클은 분출되지 않는다.

3.3 probability

생성된 에미터가 어느 정도 확률로 분출될지 입력하는 부분이다. 0을 입력할 경우 분출되지 않으며, 1을 입력할 경우 무조건 분출된다.

3.4 spread

에미터에서 뿌려져 나온 파티클의 뿌려지는 각도를 결정한다. 0을 입력할 경우 0도, 1을 입력할 경우 90도의 각도를 갖는다.

3.5 velocity

파티클이 갖는 자체 속도를 입력한다.

3.6 framerate

작업 환경의 프레임 레이트(비율)를 입력한다.

3.7 lifespan

파티클의 지속 시간을 입력한다.

3.8 lifespanrandom

파티클의 지속시간에 랜덤값을 추가한다. lifespan이 0.5이고 lifespanrandom이 0.2일 경우, 파티클의 지속시간은 0.3~0.7이 된다.

3.9 decrement

파티클의 최종 크기를 결정한다. 1을 입력할 경우, 최종 크기는 0이 되며 1이상의 값을 입력하면 파티클의 최종 크기가 어느 정도 크기를 유지한 상태에서 사라지게 된다.

4. Surface Shape

모델의 버텍스에 어느 정도 큐브를 배치할지 결정한다. 0을 입력할 경우 전혀 배치하지 않으며, 1을 입력할 경우 모델의 모든 버텍스에 큐브를 배치한다.

5. Air Field

모델의 움직임이 미세하여, 파티클의 움직임이 명확하지 않을 경우 air field를 생성할 수 있다.

5.1 Air Enable

0을 입력할 경우 air field를 생성하지 않으며, 1을 입력할 경우 air field를 생성한다.

VIII. 연구 결과

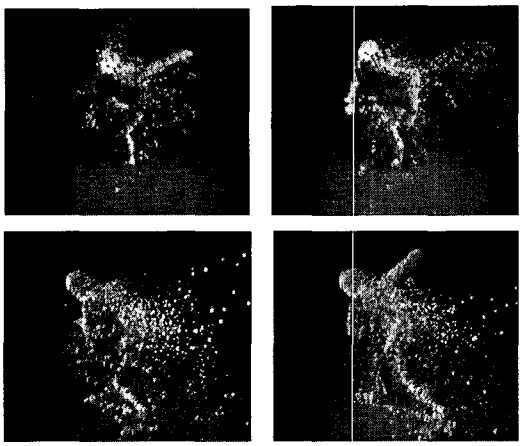

그림 16. 결과 동영상 시퀀스

[그림 16]은 본논문의 최종적인 결과물을 보여준다. 또한, 제작된 UI상에서 파티클의 흩날림의 정도, 흩날림의 지속 시간 뿐 아니라 흩날리는 오브젝트를 큐브뿐 아니라 임의의 다면체로 바꾸어 애니메이션이 가능하다. [그림 17]은 흩날림의 정도를 변화 시켜 만들어진 동영상 시퀀스를 보여준다. [그림 18]은 큐브의 흩날림을 토러스(torus)로 바꾸어 만든 애니메이션 시퀀스이다.

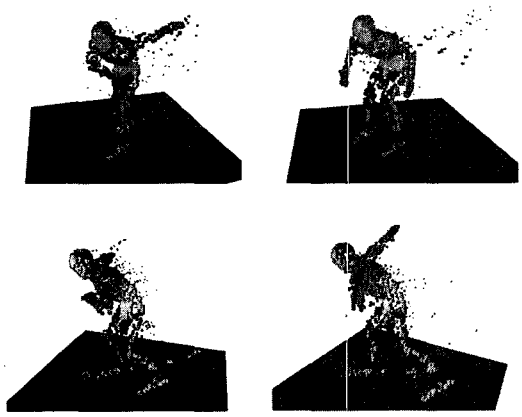

그림 17. 결과 동영상 시퀀스(흩날림양 변화)

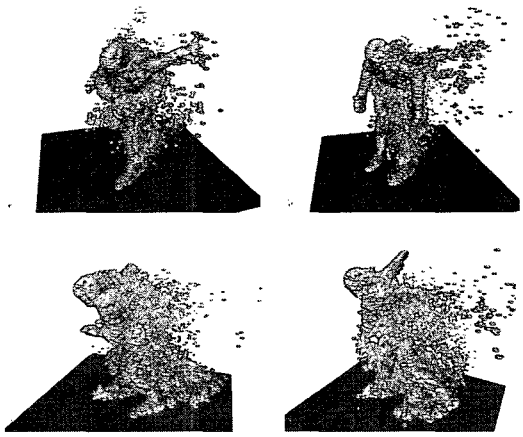

그림 18. 결과 동영상 시퀀스(토러스 적용)

IX. 결론 및 앞으로의 연구 방향

본 논문에서는 파티클을 이용하여 시각적으로 화려하고 역동적인 느낌을 주는 로고 애니메이션의 제작 과정을 사례 연구를 통해 정리하였다. 2004년 토리노 동계 올림픽 홍보 동영상을 소스로 하여 3D 컴퓨터 그래픽스 툴인 마야에서 다양한 기법을 적용하여 이와 비슷한 비주얼을 주는 영상을 제작해 보았다. MEL 스크립트로 구현이 이루어졌으며, 또한, 제작된 UI에서 다양한 옵션 값을 조절함으로서 비주얼적인 효과를 제어하고 선택할 수 있도록 구현되었다.

본 논문에서 사용한 방법은 인체 모델의 버텍스가 조밀하거나 균일하지 않을 경우 좋지 않은 결과를 낼 수 있다. 이 문제를 해결하기 위해 모델의 버텍스에 오브젝트를 배치하는 방법이 아닌 보다 효율적인 방법을 시도하여 보아야 할 것으로 판단된다.

참 고 문 헌

[1] 박진기, *Maya 5 Unlimited*, 비비컴, 2003.

[2] 이승엽, 정재환, *MAYA 7.0 - 50일 완성*, 가메, 2006.

[3] David A. D. GOULD, *COMPLETE MAYA PROGRAMMING*, Morgan Kaufmann, 2005.

[4] http://www.viewpoint.com/

저 자 소 개

정 재 민(Jai-Min Jung) 준회원

- 2005년 2월 : 아주대학교 미디어학과(미디어학사)
- 2005년 3월~현재 : 아주대학교 미디어학과 석사과정

<관심분야> : 게임 디자인

오 규 환(Gyu-Hwan Oh) 정회원

- 1991년 2월 :한국과학기술원 전산학과(학사)
- 1993년 2월 : 한국과학기술원 전산학과(석사)
- 1998년 8월 : 한국과학기술원 전산학과(박사)
- 2000년 12월~2004년 2월 : ㈜넥슨 게임개발실장
- 2005년 3월~현재 : 아주대학교 미디어학부 조교수

<관심분야> : 게임 디자인, 실시간 컴퓨터 그래픽스 기술

석 혜 정(Hae-Jung Suk) 정회원

- 1995년 2월 : 서울여자대학교 산업디자인학과(학사)
- 2001년 8월 : 홍익대학교 산업미술대학원 사진학과(석사)
- 2005년 3월~현재 : 아주대학교 미디어학부 전임강사

<관심분야> : 3D VFX