파티클 시스템 API를 이용한 특수효과 구현

김세준, 송승헌, 김웅곤
순천대학교 컴퓨터과학과

Implementation of Special Effect Using Particle System API

Se-Joon Kim, Swung-Heon Song, Eung-Gon Kim
Dept. of Computer Science, Sunchon National University

요 약

하드웨어의 성능 및 처리속도의 급격한 증가와 그래픽 개발 환경의 다양한 발전은 사용자들에게 화려한 영상을 제공하고 있으며 영화, 게임, 가상현실, 의료영상 등에서 제공되는 3차원 그래픽스 애니메이션은 그 현실감을 날로 더해가고 있는 실정이다. 파티클 시스템이란 아주 많은 입자들을 생성하여 운동법칙에 따라 이동하도록 하여 불꽃, 폭발, 분수 등의 여러 가지 현상을 나타내는데 이용하는 기술이다.

본 논문에서는 입자를 다루기 위해 파티클 시스템 API를 이용하여 메모리의 절약, 계산 시간의 단축 등을 통해 하드웨어적인 제약을 극복할 수 있으며, 효과적으로 여러 가지 자연현상 즉, 특수효과를 구현한다.

1. 서론

하드웨어의 성능 및 처리속도의 급격한 증가와 그래픽 개발 환경의 다양한 발전은 사용자들에게 화려한 영상을 제공하고 있다. 특히 영화, 게임, 가상현실, 의료영상 등에서 제공되는 3차원 그래픽스 애니메이션은 그 현실감을 날로 더해가고 있는 실정이다.

그래픽스 프로그램을 빠르게 개발하기 위해 개발된 그래픽스 API(Application Programming Interface) 즉, 그래픽스 라이브러리는 특정 플랫폼에 상관없이 모든 시스템에 호환되어야 하며, 그래픽스 하드웨어의 성능을 최대한 이끌어내야만 한다. 이러한 그래픽스 API는 가상세계 및 역동적인 3차원 그래픽스 애니메이션을 현실세계와 연결하여 표현하는 데, 가장 중요한 요소이며 다양한 기법을 사용하여 그 효과를 극대화하고 있다.

3차원 그래픽스 애니메이션에서 표현되는 효과 중에는 자연현상이나 물의 흐름 등과 같은 임의적인 요소가 자주 표현된다. 하지만, 이러한 현상을 표현하기 위한 각각의 구성 입자들에 대해 크기, 무게, 모양, 색상, 수명, 소유자 등의 속성의 부여와 입자들의 생성 및 움직임의 조절에 있어서 많은 메모리와 계산 시간이 요구되며 그 속성의 조절에 있어서 많은 어려움이 따른다.

본 논문에서는 여러 가지 자연현상과 같은 특수 효과를 표현하기 위해 파티클 시스템 API를 이용하였으며, 이를 통하여 하드웨어의 메모리 절약, 계산 시간의 단축 등을 통하여 성능의 제약을 극복할 수 있었으며, 특히 각 입자들의 상태변화에 따라서 임의적으로 움직이도록 하였다.

2장에서는 파티클 시스템과 파티클 시스템 API를 설명하고 3장에서는 파티클 시스템 API를 이용한 특수효과의 구현과 장점에 대하여 기술하고 4장에서는 결론을 맺으며 향후 연구과제를 제시한다.

본 논문은 정보통신부 2003년도 기초기술 연구지원사업의 지원에 의한 것입니다.

2. 파티클 시스템 API

파티클 시스템은 여러 개의 개별적인 조각(즉 입자)

들의 집합이다. 각 입자는 개별적인 특성들을(속도, 색상, 수명 등등) 가지며, 다른 입자와는 독립적인 방식으로 움직인다. 주어진 한 파티클 시스템의 입자들은 일반적으로 공통의 특성들을 공유하므로, 개별 입자가 독립적으로 움직인다고 해도 입자들 전체는 하나의 공통적인 효과를 만들어 내게 된다.

컴퓨터 그래픽스에서의 동적 파티클 시스템은 1983년 Reeves[2]에 의해 처음으로 표현되었다.

파티클이 가지고 있는 속성을 파티클 시스템에 어떻게 연산되어야 하는가에 대한 것은 Reeves[2]가 발표한 관련 연구를 통하여 설명되고 있고, 그가 작성한 파티클은 구형, 사각형 또는 원형과 같은 일반적인 형태 내에서 산출되었다.

원래의 파티클 시스템에서의 역학은 매우 간단한데, 실제 입자 생성후에 가해지는 힘은 중력이고, 이러한 중력은 모든 파티클에 유연한 포물선의 경로를 가지고 있다.

이러한 복잡성(중력에 따른 입자들의 경로)은 파티클 시스템에 의해 이루어졌는데, 이러한 파티클 시스템은 물리학 같은 복잡한 역학이 아닌 초기의 Random 값으로 전개된다.

OpenGL에서 초기 데이터를 입자로 변환시키기 위한 여러 가지 작업을 수행하고 이러한 입자들을 소프트웨어 루틴이나 API를 이용하여 구현함으로써, 간단한 랜더링이나 고성능의 랜더링을 요구하는 플랫폼에서도 하드웨어 및 소프트웨어 표준을 제공하고 있는 OpenGL을 사용하여 적용할 수 있다.

다음 그림 1은 OpenGL을 이용한 파티클 시스템의 적재과정을 보여준 것이다.

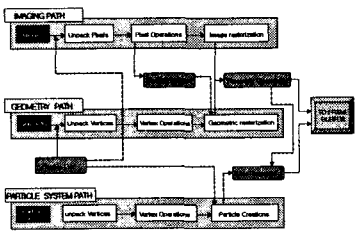


그림 1. OpenGL을 이용한 파티클 시스템의 적재과정

파티클 시스템 API는 다음의 4가지 종류의 함수를 포함한다.

Particle group들을 운영하고 관리하여 호출하는 함수, library의 위치를 지정해 주는 State setting call, 실행된 Particle group들을 호출하는 action 함수, 그리고 Action List들을 생성하고 운영하는 함수가 있다.

◆ 파티클 시스템 API의 구성요소

(1) 모든 입자들이 가지고 있는 공통적인 속성인 크기, 위치, 그리고 색상은 기본적으로 state setting call에서 지정이 된다.

(2) 모든 particle은 particle group에서 존재하고 particle들의 집합은 같은 힘으로 적용된다. Particle group은 최초에는 pGenParticleGroup들을 사용하여 생성되고, 그것은 일반화된 particle group의 수를 확인하여 반환한다.

(3) Action들은 Current Particle group에서 particle의 속성을 변경하는 API 함수이며 대부분은 pGravity와 같이 물리적인 힘의 구현을 나타낸다.

(4) Action List는 Action들을 컴파일하며 모든 기능들의 캡슐화는 특수효과 발생을 촉진시킨다

위의 요소들을 가지고 입자들을 구현할 수 있다.

3. API를 이용한 특수효과 구현

파티클 시스템의 특수효과를 생성하기 위한 흐름도는 다음 그림 2와 같다.

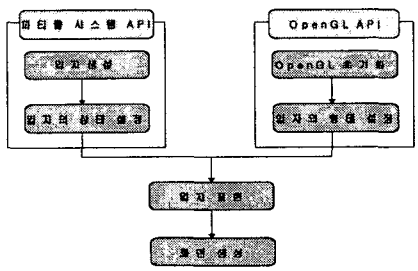


그림 2. 특수효과의 생성 흐름도

그림 2 특수효과의 생성 흐름도를 보면 파티클 시스템 API와 OpenGL API가 독립되어 작동한다는 것을 알 수 있다. OpenGL은 기본적으로 랜더링을 하면서 입자들의 모양을 설정하고 있고 파티클 시스템 API는 주로 입자들의 상태, 즉 가속도, 중력, 위치, 수명 등을 부여하여 자유로운 입자들의 움직임을 구현한다.

파티클 시스템 API의 상태 설정과 OpenGL의 형태 설정을 알아보도록 한다.

다음 그림 3은 Action에 대한 중력, 가속도, 위치갱신, 입자의 소멸 등 입자들의 기본적인 상태 설정을 해 놓은 부분이다.

```
void ComputeParticles() {
//기본설정 (속도, 색상, 크기 등)
pVelocityD(PDCylinder, 0.01, 0.0, 0.35, 0.01, 1.0,
    1.37, 0.091, 0.719);
pColorD(3.0, PDLine, 0.3, 0.3, 0.0, 1.0, 0.0, 0.0);
pSize(10.5);
pSource(400, PDLine,0.0, 0.0, 3.401, 0.0, 1.0,
 0.205);
// 중력설정
pGravity(0.0, 0.0, -0.04);
pBounce(-0.05, 0.1, 0, PDDisc, 0, 0, 0,  0, 0, 1,5);
pSink(false, PDPlane, 0,0,-3, 0,0,1);
// 위치 갱신
pMove();     }
```

그림3. 입자들의 상태 설정(메인코드)

한가지 주목할 부분은 PDCylinder와 PDDisc, PD Plane, PDLine이라고 되어 있는 부분이 있는데 이것은 중력과 속도의 작용 등 기본적인 형태를 지정해 놓은 곳이다.

pVelocityD()에서는 속도에 대한 parameter 값으로 PDCylinder이라고 연산을 할 때 동작하는 범위 내에서 실행을 한다.

그림4는 각 입자들의 상태나 모양들을 어떤 형태로 나타낼 것인가를 정의하는 부분이다. 가장 기본적인 점이나, 선, 삼각형, 원통모양 등이 정의되어 있는데 많은 입자들이 정의되어 있다면 point나 line을 쓰는 것이 바람직하다.

```
PARTICLEDLL_API enum PDomainEnum
{
PDPoint = 0, // Single point
PDLine = 1, // Line segment.
PDTriangle = 2, // Triangle
PDPlane = 3, // Arbitrarily-oriented plane
PDBox = 4, // Axis-aligned box
PDSphere = 5, // Sphere
PDCylinder = 6, // Cylinder
PDCone = 7, // Cone
PDBlob = 8, // Gaussian blob
PDDisc = 9, // Arbitrarily-oriented disc
PDRectangle = 10 // Rhombus-shaped planar
region
```

그림4. 입자들의 상태 설정(파티클 시스템 API)

그림5는 GL.h의 OpenGL의 Begin Mode이다. 이것의 사용으로 OpenGL을 이용하여 particle group에서의 입자들의 형태를 지정할 수 있게 한다.
입자들이 간격이 클 때는 TRIANGLE을 쓰는데, 이것은 삼각형의 형태로 빈 간격을 메우는 역할을 한다. 입자들의 간격이 작을때는 POINT나 LINE을 쓴다.

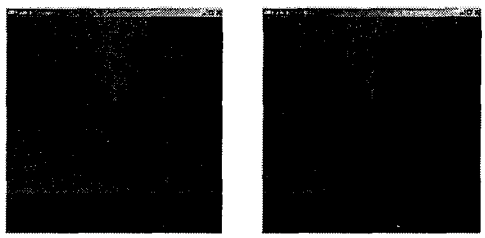
```
#define GL_POINTS                0x0000
#define GL_LINES                 0x0001
#define GL_LINE_LOOP             0x0002
#define GL_LINE_STRIP            0x0003
#define GL_TRIANGLES             0x0004
#define GL_TRIANGLE_STRIP        0x0005
#define GL_TRIANGLE_FAN          0x0006
#define GL_QUADS                 0x0007
#define GL_QUAD_STRIP            0x0008
#define GL_POLYGON               0x0009
```

그림5. 입자들의 모양 설정(OpenGL API)

기존의 파티클 시스템에서는 클래스나 OpenGL의 함수로 일일이 지면, 입자 등의 상태를 코드 상으로 지정을 해 주기 때문에 코드가 길어진다는 단점이 있고, 입자들의 상태나 위치 동작방법들을 파악하는데 어려움이 있다.

이에 비해 본 파티클 시스템 API를 이용한 효과구현에서는 API에서 입자들의 상태나 위치 동작 등을 관리하고 메인코드에서는 이것을 상황에 맞게 바꿀 수 있고, 사용자가 파티클 시스템 API에 입자들의 상태 변경과 새로운 요소를 추가할 수 있다는 장점을 가지고 있다.

그림6은 이러한 코드들을 적용시킨 구현화면이다.



가. 지표면 도달 전　　나. 지표면 도달 후

그림6. 불꽃 효과

그림 6의 가는 파티클 시스템 API를 이용하여 임의적으로 뿌려지는 입자들을 일정한 패턴을 가지고 이동을 하는 것이고, [그림 17]의 나는 이러한 입자들에 파티클 시스템 API의 속성(중력, 가속도)등을 적용하여 지면에 닿으면 튕겨져 나가는 효과를 구현하였다.

다음은 표 1은 상기와 같이 구현한 내용을 토대로 성능과 편의성에 있어서 파티클 시스템 API의 장점을 표현하였다.

표 1. 파티클 시스템 API로 구현 시 장점

	파티클 시스템 API
성 능	OpenGL과 파티클 시스템 API의 독립적인 제어로 인한 입자들의 속성과 모양 분류(중력, 가속도, 위치, 수명 등) - 성능향상 효과
편의성	실행코드에서 정의 해 놓은 API에서 제어하기 때문에 사용자가 이해하기 쉬움

파티클 시스템 API는 OpenGL과 파티클 시스템 API의 독립적인 제어로 어느 정도의 성능향상과 중력, 가속도 등의 속성을 세분화하기 때문에 보다 사실적인 장면을 연출할 수 있다.

4. 결론 및 향후 연구 과제

본 논문에서는 불꽃, 폭발 등과 같은 특수효과를 효과적으로 표현하기 위하여 각각의 입자들에 움직임 등을 주어 입자들이 더욱 자연스럽게 표현되도록 초점을 맞추었고 폴리곤이 아닌 점이나 선으로 구현을 하여 임의적으로 충돌 회피나, 물체와 부딪혀서 나타나는 효과 등을 파티클 시스템 API를 이용하여 구현하였다.

OpenGL과 파티클 시스템의 독립적인 제어로 인하여 중력, 가속도, 위치, 수명 등과 같은 일반적인 입자들의 상태설정에는 파티클 시스템 API를, 입자들의 모양이나 윈도우 창 생성 등 기본적인 형태 설정에는 OpenGL을 이용하였다.

기존에 사용하던 여러 방법들(Class 사용, 구조체 사용)을 사용하면 코딩 내의 물리적인 계산식으로 입자들의 연산을 하여 이해하기 어려운 부분이 많고 코딩 수가 길어졌는데, 파티클 시스템 API를 사용하면 API에서 중력, 가속도, 위치, 수명 등의 속성들을 관리하면서 임의적으로 입자들을 움직이게 하여 약간의 성능향상과 사용자 편의성을 제공하여 기존 시스템의 단점을 보완하였다.

이러한 파티클 시스템 API의 구현으로 입자들의 상태나 동작 등을 쉽게 파악할 수 있고 적용할 수 있게 되었다.

앞으로의 과제는 실제 현상과 같은 효과 등을 구현하기 위하여 거기에 맞는 모델들(폭발, 분수, 폭포 등)의 연산되는 값들을 패턴에 맞게 동작하도록 파티클 시스템 API에 추가하여 실제와 비슷한 효과들을 구현하는 것이다. 그리고 실제와 비슷한 효과들을 구현하기 위하여 실제 입자들의 패턴 분석과 구현할 입자들의 패턴 분석이 일치하도록 하는 연구가 요구된다.

[참고문헌]

[1] McAllister, D. K. "The Design of an API for Particle Systems" http://cs.unc.edu/~davemc/Particle, 1999

[2] Reeves, W. T. "ParticleSystems - A Techinigue for Modeling A Class of Fuzzy Objects". Proc. of SIGGRAPH '83, Detroit, Michigan, July, 1983.

[3] Reeves, W. T. and R. Blau "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems" *Proc of SIGGRAPH '85*, San Francisco, California, July, 1985

[4] http://www.opengl.org

[5] Allen, M. B. Flow - a particle animation application. http://www.dnai.com/~mba/sortware/flow/, 1999

[6] Neider, J. T. Davis, et al. OpenGL Programming Guide. Adison Wesley, 1993

[7] Leech, J. P. and R. M. Taylor. "Intractive Modeling Using Particle Systems" . *Proc of 2nd Conference on Discrete Element Methods*, MIT, 1993

[8] Sims, K. "Particle Animation and Rendering Using Data Parrel Computation". *Proc of SIGGRAPH '90* Dalls, Texas, August, 1990.

[9] 최윤철, 임순범, 고견, "컴퓨터그래픽스 배움터" 생능 출판사, 2003.

[10] Conway, M S, Audia, et al. " Alice: Lessons Learned from Building a 3D System for Novices", *Proc. of CHI 2000*, The Hague, The Netherlands, April 2000.