파티클 시스템

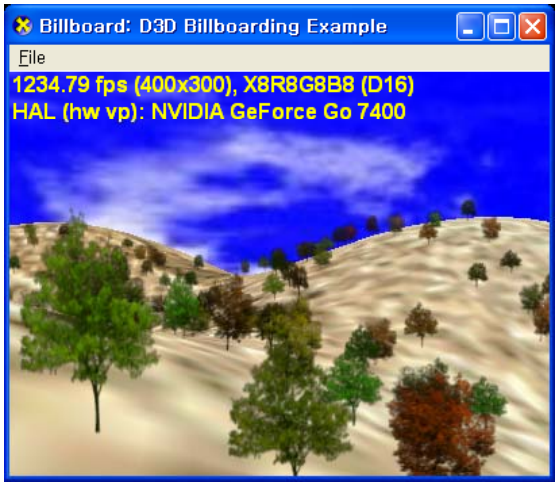
목 차

파티클과 포인트 스프라이트

- **파티클**
  - 입자, 매우 작은 물체, 하나의 포인트
  - 포인트 기본형(D3DPT_POINTLIST)
    - 단일 픽셀로 래스터라이즈 됨
    - 다양한 크기의 파티클 이용 불가
    - 텍스처를 입힌 파티클 구현 불가

파티클과 포인트 스프라이트

- **파티클**
  - 빌보드(Bill Board)
    - Direct3D 8.0 이전
    - 항상 카메라 방향을 바라보는 사각형
  - 포인트 스프라이트
    - Direct3D 8.0 이후
    - 텍스처 적용 가능, 크기 변경 가능, 하나의 포인트로 표현



파티클과 포인트 스프라이트

- **구조체 포맷**

```
struct Particle
{
        D3DXVECTOR3     _position;      // 파티클의 위치
        D3DCOLOR        _color;         // 파티클의 색상
        float           _size;          // 파티클의 크기
        static const DWORD FVF;
};
const DWORD Particale::FVF=D3DFVF_XYZ | D3DFVF_DIFFUSE | D3DFVF_PSIZE;
```

파티클과 포인트 스프라이트

- **포인트 스프라이트 렌더 상태**
  - D3DRS_POINTSPRITEENABLE
    - True : 현재 지정된 텍스처를 포인트 스프라이트의 텍스처 매핑에 이용
    - False : 포인트 스프라이트의 텍스처 좌표로 지정한 텍셀을 이용해 포인트 스프라이트의 텍스처를 입히도록 지정
  - D3DRS_POINTSCALEENABLE
    - True : 포인트 크기를 뷰 스페이스 단위로 해석
    - False : 스크린 스페이스 단위로 해석

파티클과 포인트 스프라이트

- **포인트 스프라이트 렌더 상태**
  - D3DRS_POINTSIZE
    - 포인트 스프라이트의 크기 지정
    - D3DRS_POINTSCALEENABLE값에 따라 변함
  - D3DRS_POINTSIZE_MIN
    - 포인트 스프라이트의 최소 크기
  - D3DRS_POINTSIZE_MAX
    - 포인트 스프라이트의 최대 크기

파티클과 포인트 스프라이트

- **포인트 스프라이트 렌더 상태**
  - D3DRS_POINTSCALE_A, D3DRS_POINTSCALE_B, D3DRS_POINTSCALE_C
    - 거리에 따른 변화량

$$FinalSize = ViewportHeight \times Size \times \sqrt{\frac{1}{A + B(D) + C(D^2)}}$$

FinalSize	: 거리 계산을 마치고 얻은 포인트 스프라이트의 최종 크기
ViewportHeight	: 뷰포트의 높이
Size	: D3DRS_POINTSIZE 렌더 상태에 지정된 값
A, B, C	: D3DRS_POINSTSCALE_A, B, C
D	: 뷰 스페이스 내의 포인트 스프라이트와 카메라 위치와의 거리

파티클과 포인트 스프라이트

- **파티클과 파티클의 속성들**
  - 부가적인 속성들
  - 위치, 색상, 속도 등

```
struct Attribute
{
        D3DXVECTOR3     _position;      // 월드 스페이스 내의 위치
        D3DXVECTOR3     _velocity;      // 속도
        D3DXVECTOR3     _acceleration;  // 가속
        float           _lifeTime;      // 소멸 시간
        float           _age;           // 현재 나이
        D3DXCOLOR       _color;         // 색상
        D3DXCOLOR       _colorFade;     // 색상의 시간의 흐름에 따른 변환
        bool            _isAlive;       // 생존 여부
};
```

파티클 시스템의 요소들

- **파티클 시스템**
  - 파티클들의 모임
  - 파티클의 관리
    - 파티클 갱신, 디스플레이, 소멸, 생성 등을 관장
  - PSystem
    - 파티클 시스템의 공통된 기능의 일반화

파티클 시스템의 요소들

- **PSystem**

데이터 멤버	역할
_origin	시스템의 원천, 시스템 내에서 파티클이 시작하는 곳
_boundingBox	파티클이 이동할 수 있는 부피 제한
_emitRate	시스템에 새로운 파티클이 추가되는 비율
_size	시스템 내 모든 파티클의 크기
_particles	시스템 내 파티클 속성의 리스트
_maxParticles	주어진 시간동안 시스템이 가질 수 있는 최대 파티클의 수
_vbSize	정점 버퍼가 보관할 수 있는 파티클의 수
_vbOffset	정점 버퍼에서 복사를 시작할 파티클 내 다음 단계로의 오프셋
_vbBatchSize	하나의 단계에 정의된 파티클의 수

파티클 시스템의 요소들

- **PSystem**

함수	역할
init	정점 버퍼 생성, 텍스처 생성
reset	모든 파티클 속성의 리셋
resetParticle	한 파티클의 속성 리셋
addParticle	시스템에 파티클 추가
update	시스템 내의 모든 파티클 갱신
render	시스템 내의 모든 파티클 렌더링
preRender	렌더링에 앞서 지정해야 할 초기 렌더 상태 지정
postRender	특정 파티클 시스템이 지정했을 수 있는 렌더 상태를 복구
isEmpty	파티클의 존재 여부
isDead	모든 파티클의 생존 여부
removeDeadParticles	죽은 파티클을 리스트에서 제거

파티클 시스템의 요소들

- **파티클 시스템의 드로잉**
  - 렌더링 1

    최대 수의 파티클을 보관할 수 있을만한 정점 버퍼 생성
    각 프레임에 다음 작업 수행
    모든 파티클을 갱신
    모든 살아있는 파티클들을 정점 버퍼로 복사
    정점 버퍼 그리기

  - 문제점
    - 정점 버퍼의 크기
    - 그래픽 카드의 활용도

파티클 시스템의 요소들

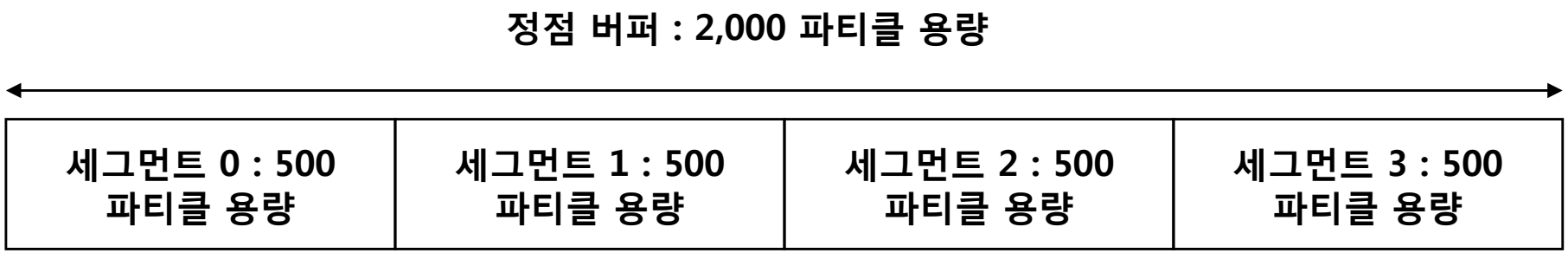
- **파티클 시스템의 드로잉**
  - 개선된 방법

```
전역변수 i=0
각 프레임에 다음과 같은 작업 수행
   모든 파티클 갱신
   모든 생존 파티클이 렌더링 될 때까지
            만약 정점 버퍼가 가득 차지 않았다면
               D3DLOCK_NOOVER-WRITE 플래그로 세그먼트 i 잠금
               세그먼트 i로 500 파티클을 복사
            만약 정점 버퍼가 가득 찼다면
               정점 버퍼의 처음부터 시작, i=0
               D3DLOCK_DISCARD 플래그로 세그먼트 i를 잠금
               세그먼트 i로 500 파티클을 복사
            세그먼트 i를 렌더링
            i++
```

파티클 시스템의 요소들

▪ 파티클 시스템의 드로잉

- 개선된 방법



- 장점
  - 정점 버퍼의 크기 감소
  - CPU와 그래픽 카드가 동시 작업

파티클 시스템의 요소들

- **무작위성**
  - 눈이 내리는 모습의 다양함이 사실감 증대
  - 랜덤 함수를 이용한 무작위성 만들기

```
float d3d::GetRandomFloat(float lowBound, float highBound)
{
        if( lowBound >= highBound )
                return lowBound;

        // [0, 1] 범위의 임의 float을 얻는다.
        float f = (rand() % 10000) * 0.0001f;

        // [lowBound, highBound] 범위의 float을 리턴한다.
        return (f * (highBound - lowBound)) + lowBound;
}
```

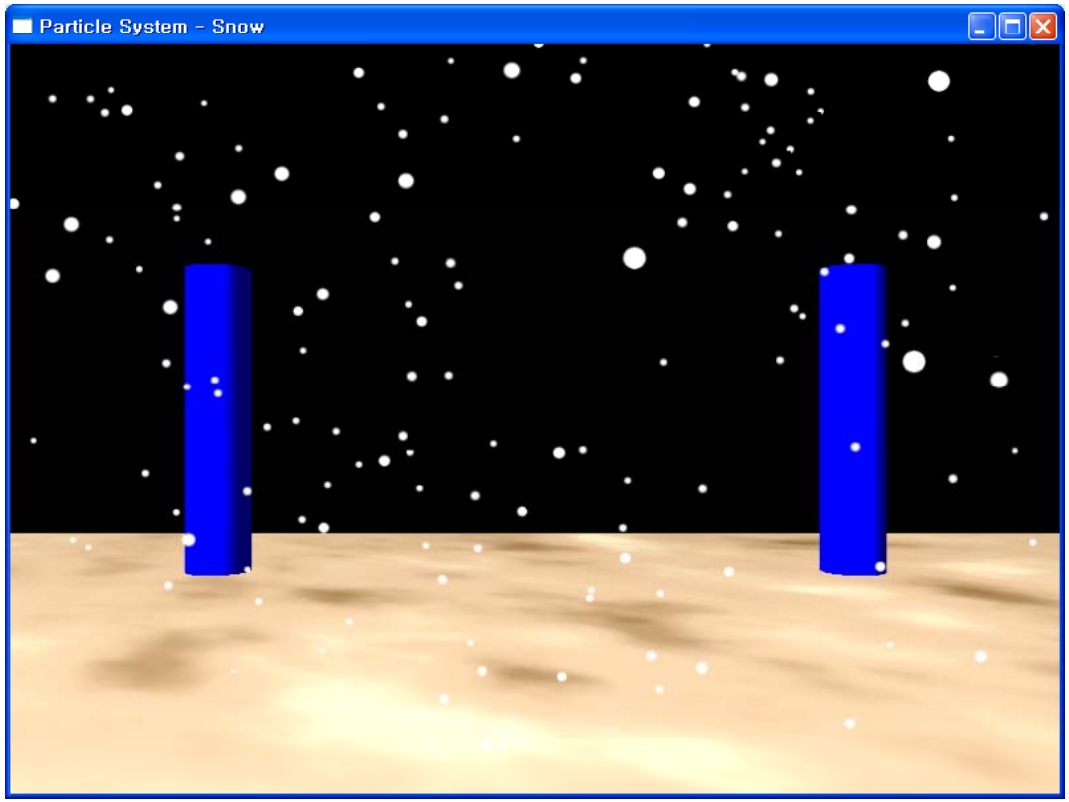
파티클 시스템의 요소들

- **무작위성**
  - 눈이 내리는 모습의 다양함이 사실감 증대
  - 랜덤 함수를 이용한 무작위성 만들기

```
void d3d::GetRandomVector( D3DXVECTOR3* out,
                           D3DXVECTOR3* min,
                           D3DXVECTOR3* max)
{
        out->x = GetRandomFloat(min->x, max->x);
        out->y = GetRandomFloat(min->y, max->y);
        out->z = GetRandomFloat(min->z, max->z);
}
```

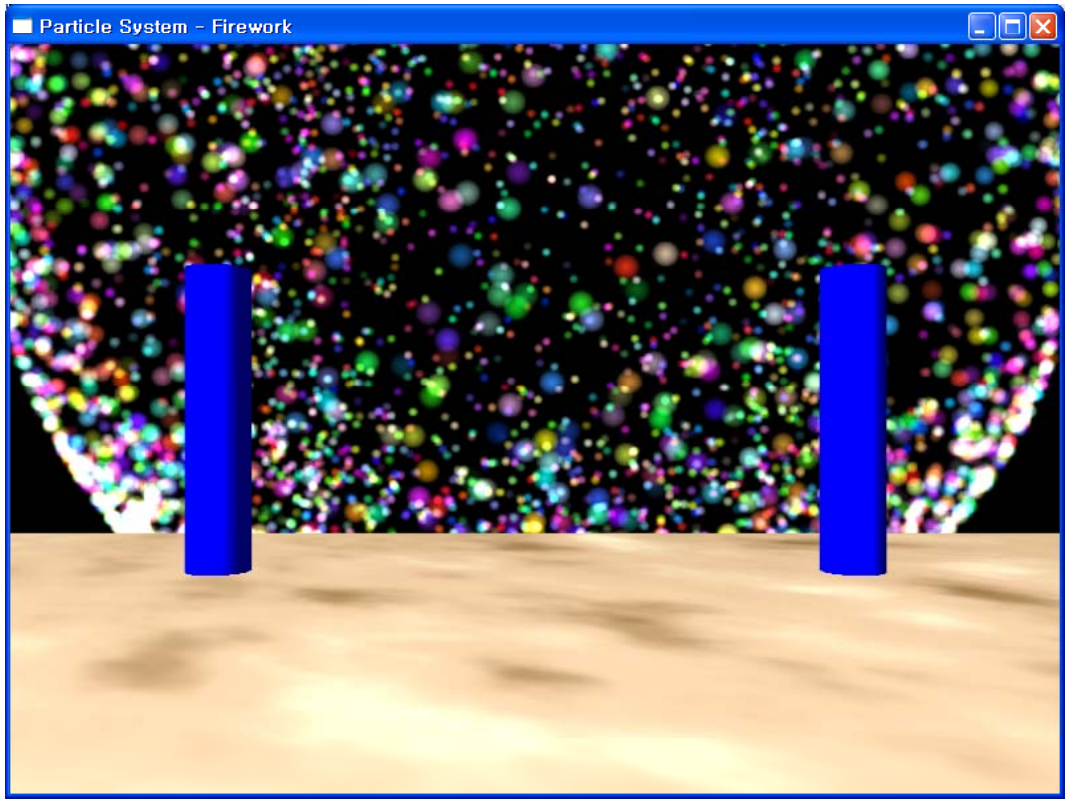
전형적인 파티클 시스템

- **예제 : 눈**



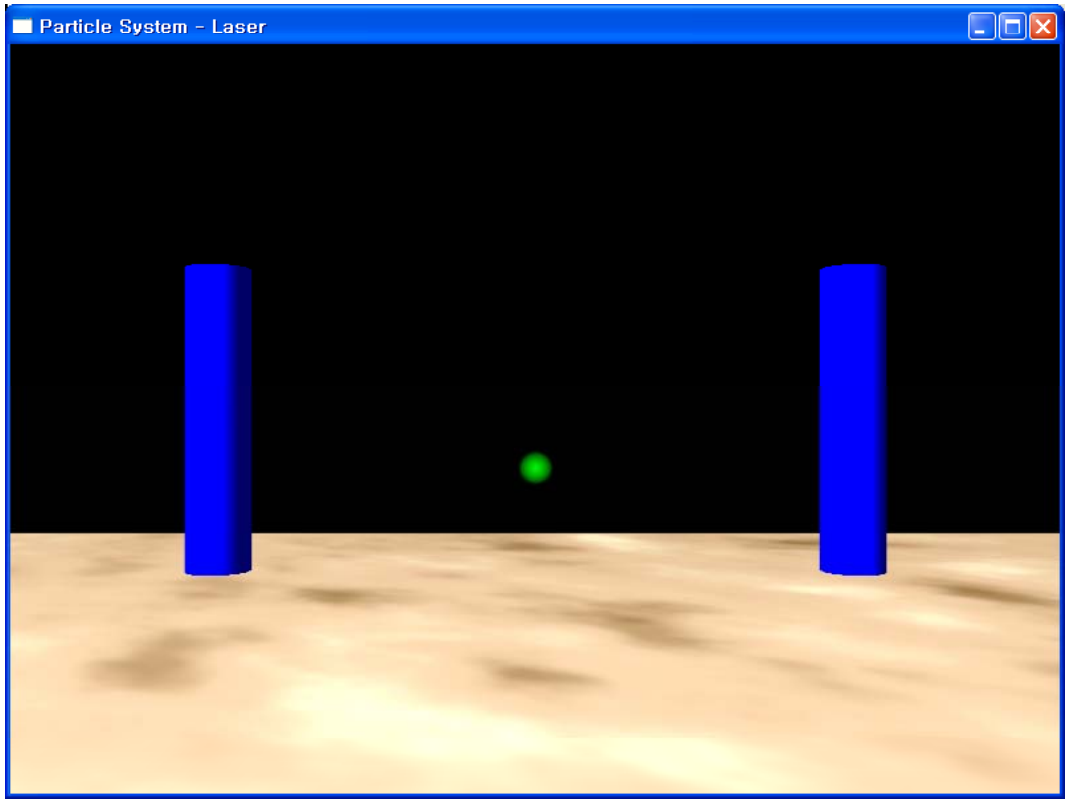
전형적인 파티클 시스템

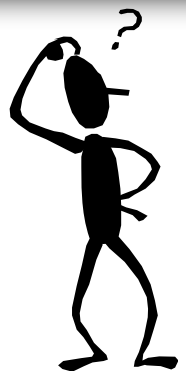
- **예제 : 불꽃놀이**



전형적인 파티클 시스템

- **예제 : 입자총**





Question?