UberShader 소개와 구현

김성익(noerror@hitel.net)
2009.5.17

Spirit of Flame
3D RealTime Graphics Programming Study
Kasa

셰이더 프로그래밍의 어려움

- 조합의 폭발
  - RimLight 가 있는 경우 없는 경우
  - 텍스처를 사용하는 경우 안하는 경우
  - 노말맵을 사용하는 경우 안하는 경우
  - 그림자가 드리우는 경우 드리우지 않는 경우
- 조합에 따른 셰이더 파일의 폭발적 증가
- 툴화의 어려움
  - 아티스트가 손쉽게 다룰 수 있는 환경 구축이 어려움
  - 기능 추가를 손쉽게 하면서 툴에 기능 추가 하기의 번거로움

셰이더 연동

- **그래프 편집**
  - 다양한 조합이 가능
  - 거꾸로 조합의 다양성으로 인한 아티스트의 어려움
  - 툴 구현 작업과 연동의 난이도
  - 대표적인 케이스 : Unreal3, 초기 ProjectOffset, 엔트리브 왕리얼 차기 엔진 (^^)
- **Uber Shader**
  - 많은 inhouse 엔진에서 사용 중
  - 특별히 셰이더를 제너럴하게 만들 필요가 없는 경우
  - 대표적인 케이스 : Crysis, ProjectOffset, 대부분의 inhouse 엔진

Uber Shader

- 하나의 셰이더로 모든 기능을 구현
- super shader 라고도 불림
- 그래프 편집과 비교
  - 몇 가지 기능 위주로 선택적으로 사용할 경우 그래프 편집보다도 편리
  - 아티스트의 셰이더 조합에 대한 스트레스 없이 기능을 적용
  - 지속적인 기능 추가를 위해선 전담 셰이더 프로그래머가 필요
  - 셰이더 종류가 한정적이어서 최적화에 유리

기본 아이디어

- 셰이더의 전처리기를 이용해서 셰이더 작성

```
float4 p(VS_OUT In) : COLOR
{
        float4 Out(1, 1, 1, 1);
#ifdef _DIFFUSE_TEXTURE
        Out = Out * tex2D(texSampler0, In.texCoord.xy );
#endif
#ifdef _DIFFUSE_COLOR
        Out = Out * In.diffuse;
#endif
#ifdef _CONST_COLOR
        Out = Out * constcolor;
#endif
        return Out;
}
```

- 셰이더 빌드시 셰이더 앞에 아래 코드를 선택적으로 삽입

```
#define _DIFFUSE_COLOR
```

- 기본 구현부는 **동일한 uber shader를 사용**하고 선택적으로 기능이 포함되도록 작성

```
comb = comb + "#define _DIFFUSE_COLOR₩r₩n";

shadercode = comb + g_shader;
D3DXCreateEffect(g_d3ddev, shadercode.c_str(), strlen(shadercode.c_str()), NULL, ...
```

- (텍스처, 텍스처+상수색, 텍스처+버텍스 칼라등 쉽게 조합해서 사용 가능)

셰이더 변수 연동(1)

- 셰이더 코드에 필요한 변수들을 선택적으로 선언

```
#ifdef _DIFFUSE_TEXTURE
sampler2D texSampler0 : BASETEXTURE;
#endif
```

- 코드에서는 GetParameterBySemantic(NULL, "BASETEXTURE") 혹은 GetParameterByName("texSampler0") 등으로 핸들 찾아서 세팅

```
CShaderSystem::ApplyShader(int mask)
{
        ...
        sh = FindShader(mask);
        if (sh == NULL)
        {
                CreateShader(mask);
                sh = FindShader(mask);
        }

        if (sh->hTexture != NULL)
                sh->effect->SetTexture(sh->hTexture, m_BaseTexture);
        if (sh->hProjMat != NULL)
                sh->effect->SetMatric(sh->hProjMat, m_ProjMat);
}

CShaderSystem::CreateShader(int mask)
{
        ...
        ID3DXEffect* effect = NULL;
        ret = D3DXCreateEffect(g_d3ddev, shadercode.c_str(), strlen(shadercode.c_str()), NULL, ...
        if (ret == D3D_OK)
        {
                sh->effect = effect;
                sh->hTexture = GetParameterBySemantic(NULL, "BASETEXTURE");
                ...
```

셰이더 변수 연동(2)

- 외부에서 세팅해서 사용

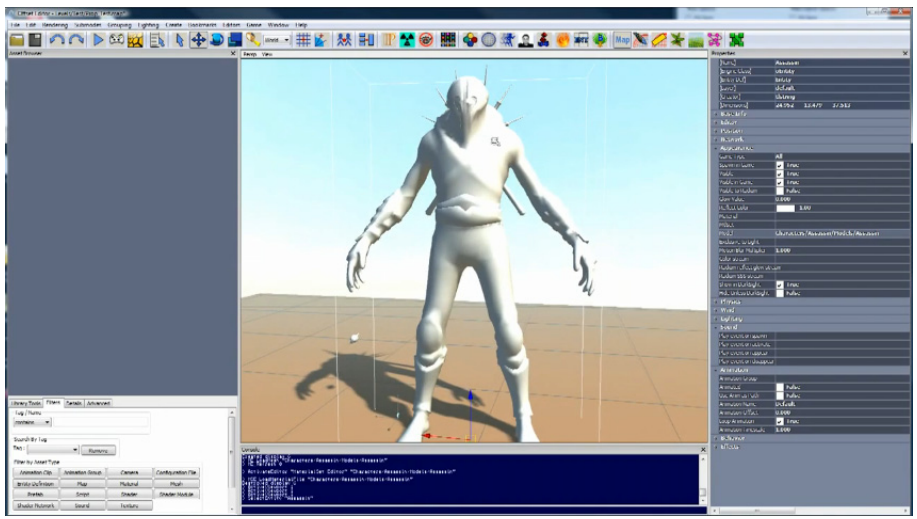
```
#define _USE_DIFFUSETEXTURE     0x01
#define _USE_VERTEXCOLOR        0x02
#define _APPLY_RIMLIGHT         0x04
#define _USE_NORMALMAP          0x08

shadersystem.SetBaseTexture("check.bmp");
shadersystem.ApplyShader(_USE_DIFFUSETEXTURE);
```

- 변수 세팅하는 방법은 다양
- 변수 찾는 부분을 자동화 할 경우 사용하지 않는 변수는 빌드에 포함되지 않도록 해야 불필요한 세팅 시간을 줄일 수 있다
- tip
  - 실제로 셰이더 코드는 하나지만 선택적으로 생성된 effect는 여러개가 되서 모든 변수를 매번 세팅
  - d3d에서 셰이더들 공통으로 사용하는 변수는 공유해서 사용가능

툴과 연동

- 기능 켜고 끄는 것을 ubershader 과 연동 (전처리 define)
- 기능을 on 했을 때 세팅을 해야 하는 파라메터들이 추가 되어 세팅 할 수 있도록
- 아티스트 친화적으로 데이터 주도형으로 구현
- (참고영상:http://www.youtube.com/watch?v=utaMQFuwNHw)



요약

- Ubershader 셰이더 시스템 구현
  - 모든 기능을 구현한 셰이더 파일 (effect)
  - 선택적으로 셰이더 기능 활성 (전처리 코드와 ubershader조합)
  - 셰이더 세팅과 변수 연계
  - 선택된 기능별 셰이더 바이너리 캐싱 (prebuild)
- 아티스트를 위한 셰이더 설정 그래픽 툴 구현