3D게임프로그래밍
-CHAPTER6-

SOULSEEK

목차

LIGHT

1. LIGHT

- 정점이 스스로 색깔을 가지지 않을 경우 기본이 검정색
- 재질이 가지는 종류에 따라 광원이 필요로 한다.

재질

- 메시(혹은 오브젝트)의 표면 상태를 말하는 것으로 빛이 물체의 표면에서 반사되어 변화된 뒤 사람의 눈에 들어오기까지의 과정을 수학적으로 모델링 하기 위해서 사용하는 값.

재질의 종류

종류	기능
주변광(**ambient**)	최저 평균 밝기를 말한다. 똑같은 양으로 모든 면에서 나오는 빛
확산광(**diffuse**)	표면의 모든 점에 균일하게 비춰지는 빛
반사광(**specular**)	특정한 방향으로만 반사하는 빛. 광원의 위치와 카메라의 위치에 따라서 달라진다.
방출광(**emissive**)	메시 표면에서 자체적으로 방출되는 빛. 이 빛이 다른 메시에 영향을 주지는 못한다.

실제적인 재질을 표현하기 위해서는 수학적인 표현에 의한 광원의 존재가 필요!

1. LIGHT

D3DX 라이브러리에서 지원하는 광원의 종류

종류	기능	사용 예
주변 광원 **(ambient light)**	**3**차원 공간 내에서 메시의 배치나 위치와는 전혀 상관없이 똑같은 양으로 모든 곳을 비추는 빛의 강도를 말한다. 방향과 위치가 없으며, 색깔과 강도만이 존재한다.	**pd3dDevice-> SetRenderState (D3DRS_AMBIENT, 0x00202020);**
점 광원 **(point light)**	직관적으로 가장 쉽게 생각할 수 있는 빛이다. 예를 들면, 백열전수를 생각하면 된다. 광원의 위치, 방향에 따라 빛의 강도가 달라진다.	**D3DLIGHT9 light; Light.Type = D3DLIGHT_POINT Light.Diffuse.r=1.0f; Light.Diffuse.g=1.0f; Light.Diffuse.b=1.0f; pd3dDevice->SetLight(0,&light); pd3dDevice->LightEnable(0, TRUE);**
방향성 광원 **(directional light)**	모든 광원의 방향이 하나의 방향을 갖는 것으로 완벽하지는 않지만 태양을 예로 들 수 있다. 광원의 위치는 상관없고, 방향이 가장 중요한 요소다.	**D3DLIGHT9 light; Light.Type = D3DLIGHT_DIRECTIONAL; Light.Diffuse.r=1.0f; Light.Diffuse.g=1.0f; Light.Diffuse.b=1.0f; pd3dDevice->SetLight(0,&light); pd3dDevice->LightEnable(0, TRUE);**
점적 광원 **(spot light)**	정해진 위치와 범위에만 비추는 특수한 조명을 말한다. 무대에서 사용하는 점적 광원 조명을 생각하면 된다.	**D3DLIGHT9 light; Light.Type = D3DLIGHT_SPOT; Light.Diffuse.r=1.0f; Light.Diffuse.g=1.0f; Light.Diffuse.b=1.0f; pd3dDevice->SetLight(0,&light); pd3dDevice->LightEnable(0, TRUE);**

1. LIGHT

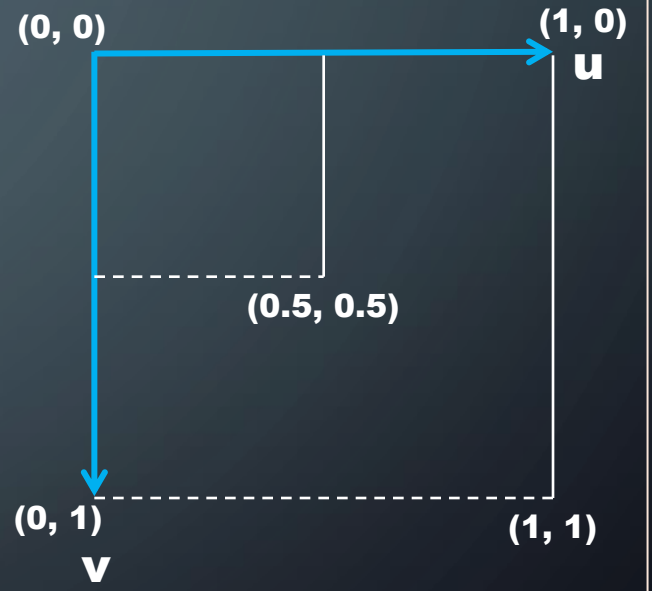
```
//광원을 사용하기 위해서는 법선 벡터의 정보가 추가 되어야 한다.
struct CUSTOMVERTEX
{
	D3DXVECTOR3 position;	//정점의 3차원 좌표
	D3DXVECTOR3 nomarl;	//정점의 법선 벡터
};

//Light 설정
void SetupLights()
{
	//재질 설정
	//재질은 디바이스에 단 하나만 설정될 수 있다.
	D3DMATERIAL9 mtrl;
	ZeroMemory(&mtrl, sizeof(D3DMATERIAL9));
	mtrl.Diffuse.r  = mtrl.Ambient.r = 1.0f;
	mtrl.Diffuse.g = mtrl.Ambient.g = 1.0f;
	mtrl.Diffuse.b = mtrl.Ambient.b = 0.0f;
	mtrl.Diffuse.a = mtrl.Ambient.a = 1.0f;
	g_pd3dDevice->SetMaterial(&mtrl);

	//광원 설정
	D3DXVECTOR3 vecDir;
	//방향성 광원(directional light)이 향한 빛의 방향
	//광원 구조체
	D3DLIGHT9 light;
	ZeroMemory(&light, sizeof(D3DLIGHT9));

	//광원의 확산광 색깔의 밝기를 지정한다.
	//광원의 종류를 설정한다(포인트 라이트, 다이렉션 라이트, 스포트 라이트)
	light.Type = D3DLIGHT_DIRECTIONAL;
	light.Diffuse.r = 1.0f;
	light.Diffuse.g = 1.0f;
	light.Diffuse.b = 1.0f;

	//광원의 방향 설정
	vecDir = D3DXVECTOR3(cosf(timeGetTime() / 350.0f), 1.0f,
	sinf(timeGetTime() / 350.0f));

	//광원의 방향을 단위 벡터로 만든다.
	D3DXVec3Normalize((D3DXVECTOR3*)&light.Direction, &vecDir);

	light.Range = 1000.0f; // 광원이 다다를 수 있는 최대거리
	g_pd3dDevice->SetLight(0, &light); // 디바이스에 광원 0번을 설치
	g_pd3dDevice->LightEnable(0, TRUE); // 광원 0번을 활성화 한다.
	g_pd3dDevice->SetRenderState(D3DRS_LIGHTING, TRUE);// 광원
	설정을 활성화 한다.

	//환경 광원의 값 설정
	g_pd3dDevice->SetRenderState(D3DRS_AMBIENT, 0x00202020);
}
```

TEXTURE

2. TEXTURE

텍스처 좌표계 – UV좌표

- 텍스처를 오브젝트에 사용하는 것을 맵핑이라고 한다.
- 맵핑을 위해서는 텍스처가 오브젝트에 어떤 모양으로 그려지는지 알아야 한다.
- 2D인 텍스처를 3D 오브젝트에 맞추는데 필요한 텍스처 스스로의 좌표를 가지고 있다.



D3D라이브러리에서 텍스처 사용..

```
//텍스처 인터페이스를 선언.
LPDIRECT3DTEXTURE9 g_pTexture= NULL;

//텍스처 좌표를 갖는 정점 선언.
struct CUSTOMVERTEX
{
      D3DXVECTOR3 position;
      D3DCOLOR color;
      FLOAT tu, tv;
};

//텍스처를 사용한다고 FVF 옵션에 추가한다.
#define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZ | D3DFVF_DIFFUSE | D3DFVF_TEX1)

//텍스처 파일을 로드한다.
D3DXCreateTextureFromFile(g_pd3dDevice, “nayeon.jpg”, &g__pTexture)
```

2. TEXTURE

```
// 텍스처 컬러 스테이지를 설정
g_pd3dDevice->SetTextureStageState(0, D3DTSS_COLOROP, D3DTOP_MOCULATE);

// 그려질 텍스처 인터페이스를 선택한다.
g_pd3dDevice->SetTexture(0, g_pTexture);

//정점 버퍼를 설정할때 텍스처도 같이 불러온다.
if (FAILED(D3DXCreateTextureFromFile(g_pd3dDevice, L"nayeon.jpg", &g_pTexture)))
{
	if (FAILED(D3DXCreateTextureFromFile(g_pd3dDevice, L"..\\nayeon.jpg", &g_pTexture)))
	{
		MessageBox(NULL, L"Could not find nayeon.jpg", L"Texture.exe", MB_OK);

		return E_FAIL;
	}
}
```

2. TEXTURE

```
//원통을 그리는 정점 셋팅
for (DWORD i = 0; i < 50; i++)
{
	FLOAT theta = (2 * D3DX_PI * i) / (50 - 1);

	pVertices[2 * i + 0].position = D3DXVECTOR3(sinf(theta), -1.0f, cosf(theta));
	pVertices[2 * i + 0].color = 0xffffffff;

	pVertices[2 * i + 1].position = D3DXVECTOR3(sinf(theta), 1.0f, cosf(theta));
	pVertices[2 * i + 1].color = 0xff808080;
}

// 0번 텍스쳐 스테이지에 텍스쳐 고정
g_pd3dDevice->SetTexture(0, g_pTexture);

// MODULATE 연산으로 색깔을 섞음
g_pd3dDevice->SetTextureStageState(0, D3DTSS_COLOROP, D3DTOP_MODULATE);

// 첫 번째 섞을 색은 텍스쳐의 색
g_pd3dDevice->SetTextureStageState(0, D3DTSS_COLORARG1, D3DTA_TEXTURE);

// 두 번째 섞을 색은 정점의 색
g_pd3dDevice->SetTextureStageState(0, D3DTSS_COLORARG2, D3DTA_DIFFUSE);

// alpha 연산을 사용하지 않음 설정
g_pd3dDevice->SetTextureStageState(0, D3DTSS_ALPHAOP, D3DTOP_DISABLE);
```