3D게임프로그래밍
-CHAPTER2-

SOULSEEK

목차

VERTEX

1. VERTEX

정점의 초기화 – InitVB()

```
HRESULT InitVB()
{
	// 삼각형을 랜더링하기 위해 3개의 정점 선언
	CUSTOMVERTEX vertices[] =
	{
		{ 150.0f, 50.0f, 0.5f, 1.0f, 0xffff0000, }, // x, y, z, rhw, color
		{ 250.0f, 250.0f, 0.5f, 1.0f, 0xff00ff00, },
		{ 50.0f, 250.0f, 0.5f, 1.0f, 0xff00ffff, },
	};

	// 정점 버퍼를 생성한다.
	// 정점 구조체 3개를 저장할 메모리를 할당한다.
	// FVF를 지정하여 보관할 데이터의 형식을 지정한다.
	if (FAILED(g_pd3dDevice->CreateVertexBuffer(3 * sizeof(CUSTOMVERTEX),
			0, D3DFVF_CUSTOMVERTEX,
	D3DPOOL_DEFAULT, &g_pVB, NULL)))
	{
		return E_FAIL;
	}

	//정점의 버퍼를 값으로 채운다.
	//정점 버퍼의 Lock() 함수를 호출하여 포인터를 얻어온다.
	void* pVertices;

	if (FAILED(g_pVB->Lock(0, sizeof(vertices), (void**)&pVertices, 0)))
	{
		return E_FAIL;
	}

	memcpy(pVertices, vertices, sizeof(vertices));

	g_pVB->Unlock();

	return S_OK;
}
```

- 정점 버퍼란 기본적으로 정점 정보를 갖고 있는 메모리 블록이다.
- 정점 버퍼를 생성한 다음에는 반드시 **Lock()**과 **Unlock()**으로 포인터를 얻어내서 정점 정보를 정점 버퍼에 써넣어야 한다.
- **D3D**는 인덱스 버퍼도 사용 가능하다는 것을 명심하자.
- 정점 버퍼나 인덱스 버퍼는 기본 시스템 메모리 외에 디바이스 메모리(비디오카드 메모리)에
- 생성될 수 있는데, 대부분의 비디오카드에서는 이렇게 할 경우 엄청난 속도향상을 얻을 수 있다.

1. VERTEX

LPDDIRECT3DVERTEXBUFFER9 g_pVB // 정점을 보관할 버퍼

- **D3D**에서는 수많은 정점포맷이 있지만 모든 플래그를 다 선언할 필요가 없다.
- 플래그가 어떤 일을 하는지 이해하고 사용하는 것이 중요.
- 사용할 **FVF** 플래그와 플래그에 맞는 **vertex** 정보를 구성하는 구조체를 구성해야 한다.

명칭	용도
정점의 좌표	정점의 **3**차원 좌표
RHW	동차 좌표계의 **w**값. 이 값이 존재하면 변환이 완료된 정점이다.
결합 가중치	스키닝에 사용된다.
법선 벡터	정점의 법선 벡터를 나타낸다. 주로 광원 처리 시 사용
Diffuse	**RGBA(r, g, b, a)** 매크로 값이며, 정점의 확산광 색깔을 나타낸다.
Reflection	**RGBA(r, g, b, a)** 매크로 값이며, 정점의 반사광 색깔을 나타낸다.
텍스쳐 좌표	텍스쳐 좌표값을 나타낸다. **D3D**는 **8**개까지 텍스쳐를 동시에 겹쳐서 사용될 수 있다.

```
struct CUSTOMVERTEX
{
        FLOAT           x, y, z, rhw; //정점의 변환된 좌표(rhw 값이 있으면 변환이 완료된 정점이다.)
        DWORD           color // 정점의 색깔
}

//x, y, z, RHW값과 Diffuse 색깔 값으로 이루어져 있음을 알 수 있다.
#define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZRHW | D3DFVF_DIFFUSE)
```

1. VERTEX

정점 버퍼

- 정점처리만을 위해 모아두는 일종의 메모리
- 비디오 메모리와 시스템 메모리 두 가지 메모리를 사용한다.
- 비디오 메모리에 생성된 정점 버퍼는 비디오카드의 GPU에 의해서 강력한 하드웨어 가속을 사용할 수 있지만, 비디오 카드 용량을 벗어 날 수 없다.
- 시스템 메모리에서 생성된 정점 버퍼는 하드웨어 가속을 사용할 수 없지만 풍부한 메모리를 가지고 있어 많은 양의 정점 버퍼를 관리 할 수 있다.

정점 버퍼에 들어갈 Data

```
CUSTOMVERTEX vertices[] =
{
    { 150.0f, 50.0f, 0.5f, 1.0f, 0xffff0000, },

    { 250.0f, 250.0f, 0.5f, 1.0f, 0xff00ff00, },

    { 50.0f, 250.0f, 0.5f, 1.0f, 0xff00ffff, },
}
```

1. VERTEX

정점 버퍼 생성

HRESULT CreateVertexBuffer(UINT Length, DWORD Usage, DWORD FVF, D3DPOOL Pool, IDirect3DVertexBuffer ppVertexBuffer, HANDLE* pSharedHandle)**

Length : 생성할 정점 버퍼의 바이트 단위 크기
Usage : 정점 버퍼의 종류 혹은 처리 방식**(SW, HW)**지정
FVF : 정점 정보 구조체에 따라 선언된 **FVF** 플래그 값
Pool : 정점 버퍼가 저장될 메모리의 위치**(**비디오 카드**,** 시스템 메모리**)**와 관리 방식 지정
ppVertexBuffer : 반환될 정점 버퍼의 인터페이스

정점 버퍼의 Data를 추가

생성한 정점 버퍼에는 아직 쓰레기 값들로 채워져 있기 때문에 **Lock()**걸고 **Data**를 넣기 위해 생성한 버퍼 메모리의 포인터를 **Lock()**로 얻어내야 한다**.** 사용 후 **UnLock()**로 풀어줘야 **Render**를 할 수 있다**.**

HRESULT Lock(UINT OffsetToLock, UINT SizeToLock, VOID ppbData, DWORD Flags);**
OffsetToLock : Lock을 할 버퍼의 시작점**, SizeToLock**과 함께 양쪽 모두 **0**이면 버퍼 전체
SizeToLock : Lock을 할 버퍼의 크기**, OffsetToLock**과 함께 양쪽 모두 **0**이면 버퍼 전체
ppbData : 읽고 쓸 수 있게 된 메모리 영역의 포인터
Flags : Lock을 수행할 때 함께 사용하는 플래그

1. VERTEX

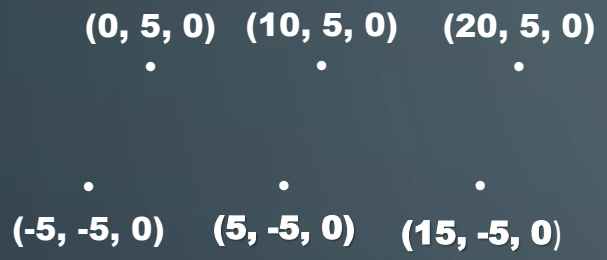
준비된 정점 버퍼로 그리기

```
void Render()
{
	if (NULL == g_pd3dDevice)
		return;

	g_pd3dDevice->Clear(0, NULL, D3DCLEAR_TARGET, D3DCOLOR_XRGB(0, 0, 255), 1.0f, 0);

	if (SUCCEEDED(g_pd3dDevice->BeginScene()))
	{
		// 정점 버퍼로 삼각형을 그린다.
		// 정점 정보가 담겨 있는 정점 버퍼를 출력 스트림으로 할당한다.
		g_pd3dDevice->SetStreamSource(0, g_pVB, 0, sizeof(CUSTOMVERTEX));
		// D3D에게 정점 셰이터 정보를 지정한다. 대부분의 경우 FVF만 지정한다.
		g_pd3dDevice->SetFVF(D3DFVF_CUSTOMVERTEX);
		// 기하 정보를 출력하기 위한 DrawPrimitive() 함수 호출
		g_pd3dDevice->DrawPrimitive(D3DPT_TRIANGLELIST, 0, 1);

		g_pd3dDevice->EndScene();
	}

	g_pd3dDevice->Present(NULL, NULL, NULL, NULL);
}
```

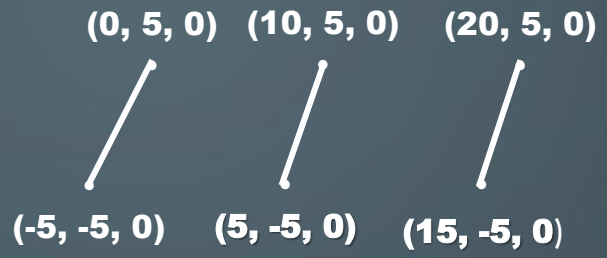
1. VERTEX

DrawPrimitive의 옵션

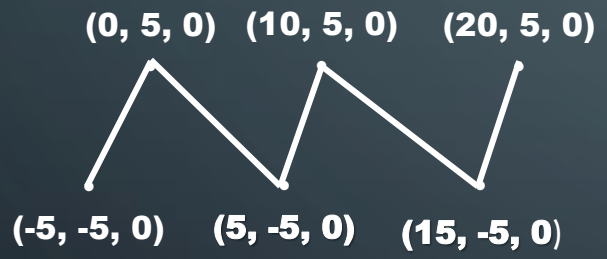
D3DPT_POINTLIST



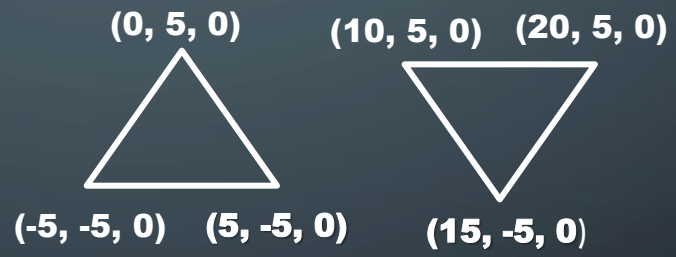
D3DPT_LINELIST



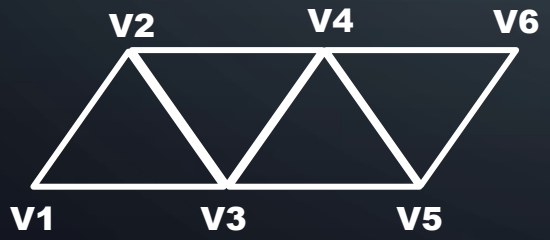
D3DPT_LINESTRIP



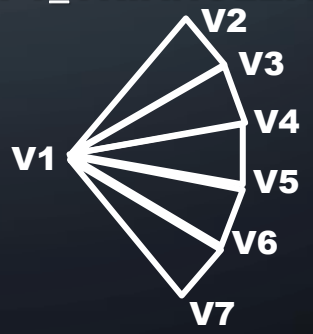
D3DPT_TRIANGLELIST



D3DPT_TRIANGLESTRIP



D3DPT_TRIANGLEPAN



1. VERTEX

학습과제

- 정점들의 정보를 변경해 보자
- 삼각형 그리기를 이용해서 사각형을 그려보자