Directx 10

네번째시간- 변환 자유자재로 다루기

When. 2008. 7. 16. (수)

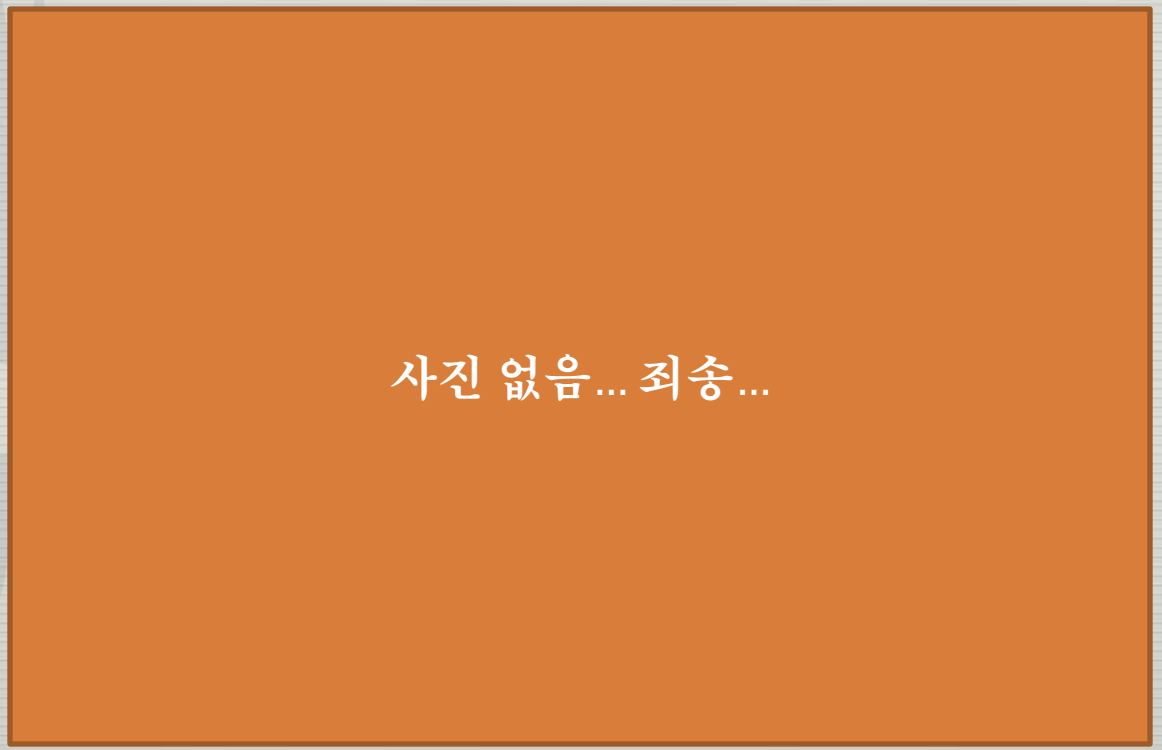
Who. 최승식

오늘의 목표

- 인덱스버퍼 사용하기.
- 변환 자유자재로 다루기.
  - 한 물체의 이동 + 회전.
  - 두 물체의 이동 + 회전.
- 실습.
  - 연관된 두 물체의 이동 + 회전.
  - 연관된 여러 물체의 이동 + 회전.

인덱스버퍼 사용하기.

- 목표 1 : 인덱스버퍼 사용하기.



인덱스버퍼 사용하기.

- (2/1) - 변수 추가 및 정점 정보 변경.

```
ID3D10Buffer*           g_pIndexBuffer = NULL;
```

← 해오던대로 변수추가

```
SimpleVertex vertices[] =
{
// 위
    {D3DXVECTOR3( 0.0f, 5.0f, 0.0f ), D3DXVECTOR4(1.0f, 1.0f, 0.0f, 1.0f)},

// 아래 1
    {D3DXVECTOR3( 0.0f, 0.0f, 5.0f ), D3DXVECTOR4(1.0f, 0.0f, 0.0f, 1.0f)},

// 아래 2
    {D3DXVECTOR3( -5.0f, 0.0f, -5.0f ), D3DXVECTOR4(0.0f, 1.0f, 0.0f, 1.0f)},

// 아래 3
    {D3DXVECTOR3( 5.0f, 0.0f, -5.0f ), D3DXVECTOR4(0.0f, 0.0f, 1.0f, 1.0f)},
};
```

← 사면체의 네 꼭지점을 입력

```
bd.ByteWidth = sizeof( SimpleVertex ) * 4;
```

← 정점 개수 수정

인덱스버퍼 사용하기.

- (2 / 2) - 인덱스버퍼 생성 및 해제.

```
UINT stride = sizeof( SimpleVertex );
UINT offset = 0;
g_pd3dDevice->IASetVertexBuffers( 0, 1, &g_pVertexBuffer, &stride, &offset );
```

기존에 있는 부분

```
DWORD indices[] =
{
    1, 2, 3,
    0, 3, 2,
    0, 2, 1,
    0, 1, 3,
};
```

요것이 핵심
숫자들을 주시하라!

```
bd.Usage = D3D10_USAGE_DEFAULT;
bd.ByteWidth = sizeof( DWORD ) * 12;
bd.BindFlags = D3D10_BIND_INDEX_BUFFER;
bd.CPUAccessFlags = 0;
bd.MiscFlags = 0;
InitData.pSysMem = indices;
hr = g_pd3dDevice->CreateBuffer( &bd, &InitData, &g_pIndexBuffer );
if( FAILED( hr ) )
    return hr;

g_pd3dDevice->IASetIndexBuffer( g_pIndexBuffer, DXGI_FORMAT_R32_UINT, 0 );
```

버퍼의 속성을 입력하고
인덱스버퍼를 생성

```
if( g_pIndexBuffer ) g_pIndexBuffer->Release();
```

CleanUpDevice() 에서 해제

인덱스버퍼 사용하기.

- (3 / 3) - 더 살펴보기.

```
SimpleVertex vertices[] =
{
// 위
    {D3DXVECTOR3( 0.0f, 5.0f, 0.0f ), D3DXVECTOR4(1.0f, 1.0f, 0.0f, 1.0f)},
// 아래 1
    {D3DXVECTOR3( 0.0f, 0.0f, 5.0f ), D3DXVECTOR4(1.0f, 0.0f, 0.0f, 1.0f)},
// 아래 2
    {D3DXVECTOR3( -5.0f, 0.0f, -5.0f ), D3DXVECTOR4(0.0f, 1.0f, 0.0f, 1.0f)},
// 아래 3
    {D3DXVECTOR3( 5.0f, 0.0f, -5.0f ), D3DXVECTOR4(0.0f, 0.0f, 1.0f, 1.0f)},
};
```

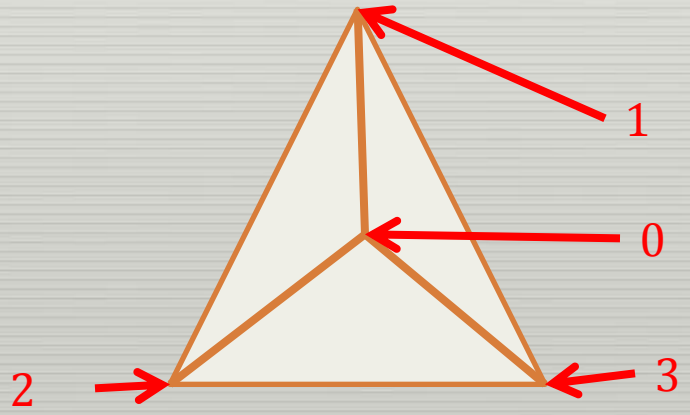
0

1

2

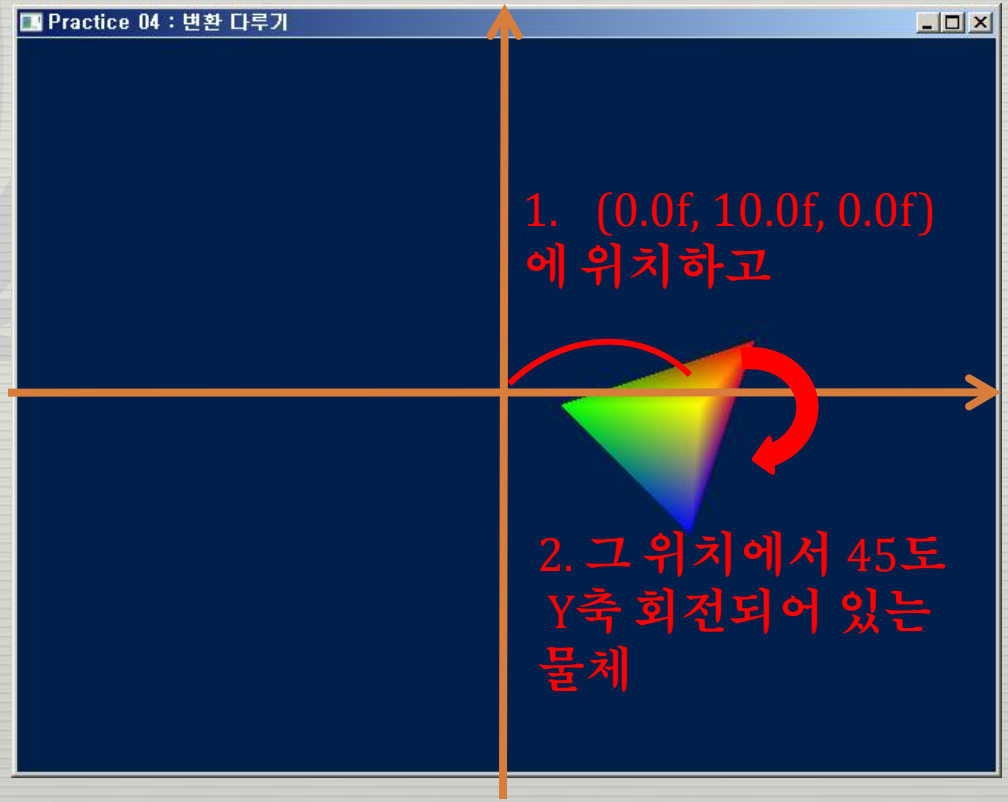
3

```
DWORD indices[] =
{
    1, 2, 3,
    0, 3, 2,
    0, 2, 1,
    0, 1, 3,
};
```



변환 자유자재로 다루기.

- 목표 2-1 : 한 물체의 이동 + 회전.



변환 자유자재로 다루기.

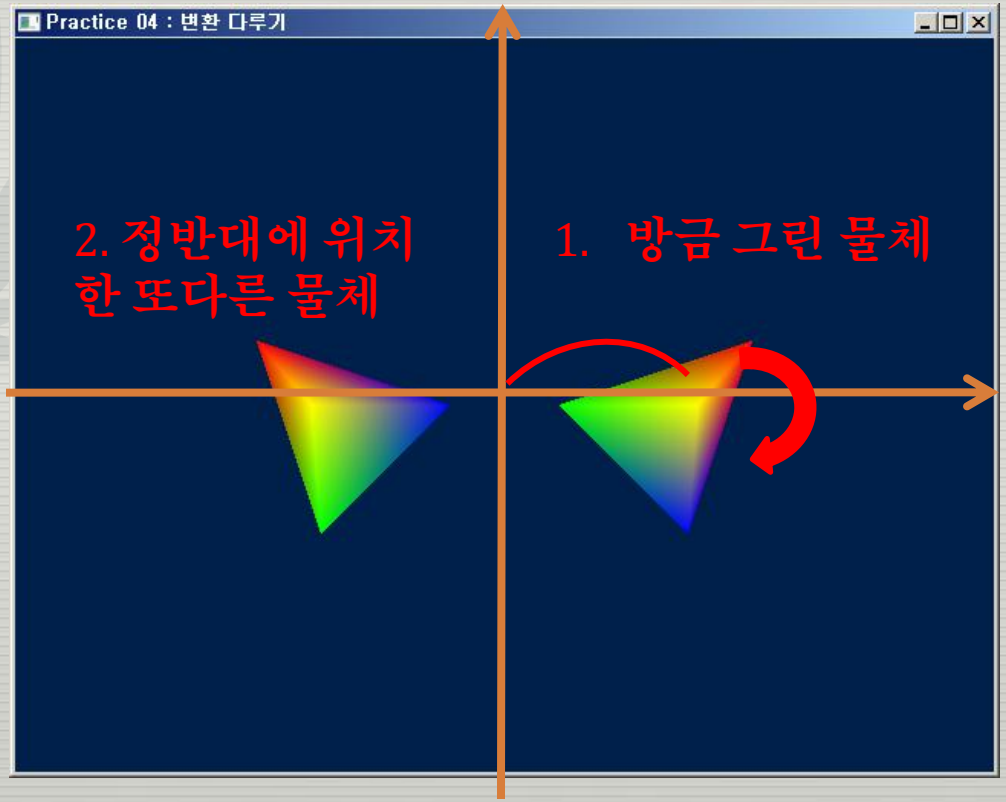
- (2 / 1) – 카메라 위치 조정.
  - vEye(0.0f, 20.0f, 0.0f)
  - vLookAt(0.0f, 0.0f, 0.0f)
  - vUp(0.0f, 0.0f, 1.0f)

변환 자유자재로 다루기.

- (2 / 2) – 행렬 계산.
  - Y축 45도 회전이므로
    - 45도 = D3DX_PI / 4
    - D3DXMatrixRotationY(&m1, D3DX_PI / 4)
  - +X방향으로 10.0f 만큼 이동이므로
    - D3DXMatrixTranslation(&m2, 10.0f, 0.0f, 0.0f)
  - 두 동작을 동시에 하기 위해
    - g_World = m1 * m2
    - 순서에 주의해야함.

변환 자유자재로 다루기.

- 목표 2-2 : 두 물체의 이동 + 회전.



변환 자유자재로 다루기.

- (2 / 1) - 두번째 사면체를 위한 준비.

```
D3DXMATRIX                  g_World2;
```

두번째 사면체를 이동,
회전 시키기 위한 행렬

```
D3DXMatrixRotationY(&m, -D3DX_PI / 4);
g_World2 *= m;

D3DXMatrixTranslation(&m, -10.0f, 0.0f, 0.0f);
g_World2 *= m;
```

행렬을 계산한다.

변환 자유자재로 다루기.

- (2 / 2) - 두번째 사면체를 그림.

```
D3D10_TECHNIQUE_DESC techDesc;
g_pTechnique->GetDesc( &techDesc );
for( UINT p = 0; p < techDesc.Passes; ++p )
{
    g_pTechnique->GetPassByIndex( p )->Apply( 0 );

    g_pWorldVariable->SetMatrix( (float*)&g_World );
    g_pd3dDevice->DrawIndexed( 12, 0, 0 );



    g_pTechnique->GetPassByIndex( p )->Apply( 0 );

    g_pWorldVariable->SetMatrix( (float*)&g_World2 );
    g_pd3dDevice->DrawIndexed( 12, 0, 0 );
}
```

두번째 사면체를 그린다.

실 습.

- 목표 2-3 : 연관된 두 물체의 이동 + 회전.