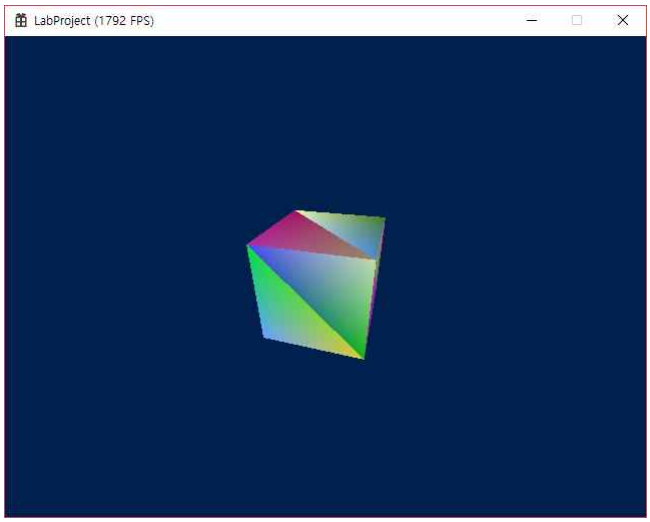
□ 예제 프로그램 9: LabProject08(회전하는 직육면체 그리기)

LabProject07 프로젝트를 기반으로 다음과 같이 회전하는 직육면체를 그려본다.



① 새로운 프로젝트의 생성

먼저 새로운 프로젝트 LabProject08를 생성한다. “LabProjects" 솔루션을 열고 솔루션 탐색기에서 마우스 오른쪽 버튼으로 『솔루션 LabProjects』를 선택하고 메뉴에서 『추가』, 『새 프로젝트』를 차례로 선택한다. 그러면 『새 프로젝트 대화상자』가 나타난다. 그러면 프로젝트 이름 “LabProject08”를 입력하고 『확인』을 선택한다.

❶ 파일 탐색기에서 프로젝트 “LabProject07” 폴더의 다음 파일을 선택하여 프로젝트 “LabProject08” 폴더에 복사한다.

- GameFramework.h
- GameFramework.cpp
- Mesh.h
- Mesh.cpp
- Camera.h
- Camera.cpp
- Object.h
- Object.cpp
- Scene.h
- Scene.cpp
- Shader.h
- Shader.cpp

- stdafx.h
- Timer.h
- Timer.cpp
- Shaders.hlsl

❷ 위에서 복사한 파일을 Visual Studio 솔루션 탐색기에서 프로젝트 “LabProject08”에 추가한다. 오른쪽 마우스 버튼으로 『LabProject08』을 선택하고 『추가』, 『기존 항목』를 차례로 선택한다. 그러면 “기존 항목 추가” 대화 상자가 나타난다. 다음 파일들을 마우스로 선택(Ctrl+선택)하여 『추가』를 누르면 선택된 파일들이 프로젝트 “LabProject08”에 추가된다.

② LabProject08.cpp 파일 수정하기

이제 “LabProject08.cpp" 파일의 내용을 “LabProject07.cpp" 파일의 내용으로 바꾸도록 하자. “LabProject07.cpp" 파일의 내용 전체를 “LabProject08.cpp" 파일로 복사한다. 이제 “LabProject08.cpp" 파일에서 “LabProject07”을 “LabProject08”로 모두 바꾼다.
그리고 “LABPROJECT07”을 “LABPROJECT08”로 모두 바꾼다.

③ “Stdafx.h” 파일 수정하기

“Stdafx.h” 파일에 다음을 추가한다.

```
/*정점의 색상을 무작위로(Random) 설정하기 위해 사용한다. 각 정점의 색상은 난수(Random Number)를 생성하여 지정한다.*/
#define RANDOM_COLOR  XMFLOAT4(rand() / float(RAND_MAX), rand() / float(RAND_MAX), rand() / float(RAND_MAX), rand() / float(RAND_MAX))
```

④ “Mesh.h” 파일 수정하기

“Mesh.h” 파일에 직육면체 메쉬를 위한 클래스 “CCubeMeshDiffused”를 다음과 같이 선언한다.

```
class CCubeMeshDiffused : public CMesh
{
public:
//직육면체의 가로, 세로, 깊이의 길이를 지정하여 직육면체 메쉬를 생성한다.
   CCubeMeshDiffused(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList *pd3dCommandList, float fWidth = 2.0f, float fHeight = 2.0f, float fDepth = 2.0f);
   virtual ~CCubeMeshDiffused();
};
```

⑤ “Mesh.cpp” 파일 수정하기

❶ “Mesh.cpp” 파일에 직육면체 메쉬를 표현하기 위한 클래스 “CCubeMeshDiffused”의 생성자를 다음

과 같이 정의한다.

직육면체는 다음 그림과 같이 6개의 면(사각형)을 가지고 각 면은 2개의 직각삼각형으로 구성된다. 직육면체를 삼각형 리스트(D3D_PRIMITIVE_TOPOLOGY_TRIANGLELIST)로 표현하려면 총 36개의 정점 데이터가 필요하다(6x2x3). 즉, 직육면체는 6개의 사각형이 필요하고 각 사각형은 2개의 삼각형, 그리고 각 삼각형은 3개의 정점 데이터가 필요하다.

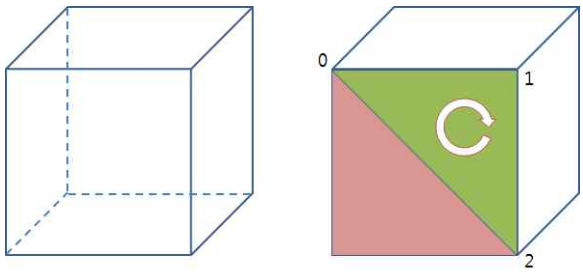
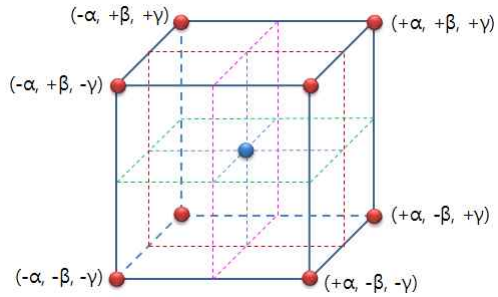


그림 2. 직육면체

다음 그림과 같이 직육면체의 정가운데가 모델좌표계의 원점이 되도록 정점의 좌표를 구성한다. 직육면체의 가로(x-축) 길이 2α, 세로(y-축) 길이 2β, 깊이(z-축)의 길이 2γ가 주어지면 직육면체의 8개의 정점 좌표는 다음과 같다.

(-α, -β, -γ) (-α, +β, -γ) (+α, +β, -γ) (+α, -β, -γ)
(-α, -β, +γ) (-α, +β, +γ) (+α, +β, +γ) (+α, -β, +γ)



직육면체의 정점 위치 벡터

```
CCubeMeshDiffused::CCubeMeshDiffused(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList
*pd3dCommandList, float fWidth, float fHeight, float fDepth) : CMesh(pd3dDevice,
pd3dCommandList)
{
//직육면체는 6개의 면 가로(x-축) 길이
   m_nVertices = 36;
   m_nStride = sizeof(CDiffusedVertex);
   m_nOffset = 0;
   m_nSlot = 0;
   m_d3dPrimitiveTopology = D3D_PRIMITIVE_TOPOLOGY_TRIANGLELIST;

//fWidth: 직육면체 가로(x-축) 길이, fHeight: 직육면체 세로(y-축) 길이, fDepth: 직육면체 깊이(z-축) 길이
```

```
   float fx = fWidth*0.5f, fy = fHeight*0.5f, fz = fDepth*0.5f;

   CDiffusedVertex pVertices[36];
   int i = 0;

//정점 버퍼 데이터는 삼각형 리스트이므로 36개의 정점 데이터를 준비한다.
//ⓐ 앞면(Front) 사각형의 위쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, -fz), RANDOM_COLOR);

//ⓑ 앞면(Front) 사각형의 아래쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, -fz), RANDOM_COLOR);

//ⓒ 윗면(Top) 사각형의 위쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, -fz), RANDOM_COLOR);

//ⓓ 윗면(Top) 사각형의 아래쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, -fz), RANDOM_COLOR);

//ⓔ 뒷면(Back) 사각형의 위쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, +fz), RANDOM_COLOR);

//ⓕ 뒷면(Back) 사각형의 아래쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, +fz), RANDOM_COLOR);

//ⓖ 아래면(Bottom) 사각형의 위쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, +fz), RANDOM_COLOR);

//ⓗ 아래면(Bottom) 사각형의 아래쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, +fz), RANDOM_COLOR);

//ⓘ 옆면(Left) 사각형의 위쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, +fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, -fz), RANDOM_COLOR);
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, -fz), RANDOM_COLOR);

//ⓙ 옆면(Left) 사각형의 아래쪽 삼각형
   pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, +fy, +fz), RANDOM_COLOR);
```

```
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, -fz), RANDOM_COLOR);
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(-fx, -fy, +fz), RANDOM_COLOR);

//ⓚ 옆면(Right) 사각형의 위쪽 삼각형
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, -fz), RANDOM_COLOR);
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, +fz), RANDOM_COLOR);
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, +fz), RANDOM_COLOR);

//① 옆면(Right) 사각형의 아래쪽 삼각형
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, +fy, -fz), RANDOM_COLOR);
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, +fz), RANDOM_COLOR);
	pVertices[i++] = CDiffusedVertex(XMFLOAT3(+fx, -fy, -fz), RANDOM_COLOR);

	m_pd3dVertexBuffer = ::CreateBufferResource(pd3dDevice, pd3dCommandList, pVertices,
m_nStride * m_nVertices, D3D12_HEAP_TYPE_DEFAULT,
D3D12_RESOURCE_STATE_VERTEX_AND_CONSTANT_BUFFER, &m_pd3dVertexUploadBuffer);

//정점 버퍼 뷰를 생성한다.
	m_d3dVertexBufferView.BufferLocation = m_pd3dVertexBuffer->GetGPUVirtualAddress();
	m_d3dVertexBufferView.StrideInBytes = m_nStride;
	m_d3dVertexBufferView.SizeInBytes = m_nStride * m_nVertices;
}
```

❷ 클래스 “CCubeMeshDiffused”의 소멸자를 다음과 같이 정의한다.

```
CCubeMeshDiffused::~CCubeMeshDiffused()
{
}
```

⑥ ”Scene.cpp” 파일 수정하기

“Scene.cpp” 파일을 다음과 같이 수정한다.

❶ “CScene” 클래스의 BuildObjects() 함수를 다음과 같이 수정한다.

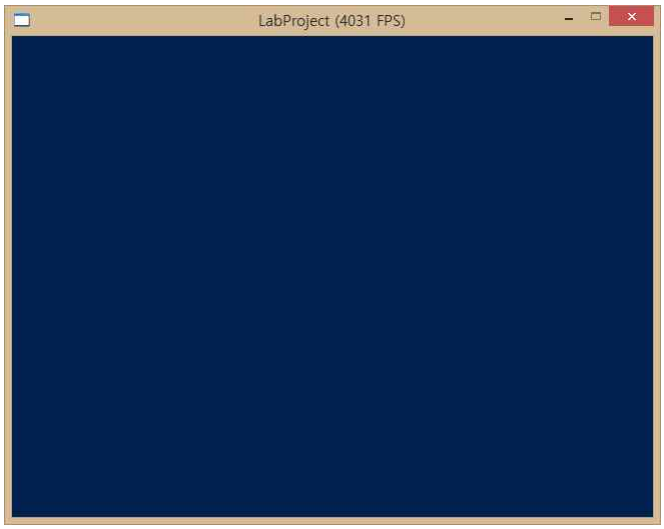
```
void CScene::BuildObjects(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList
*pd3dCommandList)
{
	m_pd3dGraphicsRootSignature = CreateGraphicsRootSignature(pd3dDevice);

//가로x세로x깊이가 12x12x12인 정육면체 메쉬를 생성한다.
	CCubeMeshDiffused *pCubeMesh = new CCubeMeshDiffused(pd3dDevice, pd3dCommandList,
12.0f, 12.0f, 12.0f);

	m_nObjects = 1;
	m_ppObjects = new CGameObject*[m_nObjects];

	CRotatingObject *pRotatingObject = new CRotatingObject();
	pRotatingObject->SetMesh(pCubeMesh);

	CDiffusedShader *pShader = new CDiffusedShader();
```

```
    pShader->CreateShader(pd3dDevice, m_pd3dGraphicsRootSignature);
    pShader->CreateShaderVariables(pd3dDevice, pd3dCommandList);

    pRotatingObject->SetShader(pShader);

    m_ppObjects[0] = pRotatingObject;
}
```

⑦ 프로젝트 빌드하여 실행하기

❶ 프로젝트를 빌드하여 실행

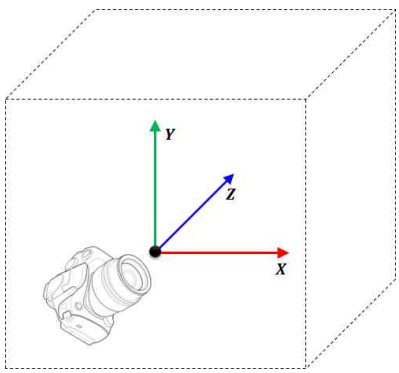
프로젝트를 빌드하여 실행하면 다음 그림과 같이 화면에 아무것도 보이지 않게 된다.



실행 결과(카메라의 위치: (0, 0, -2))

이것은 직육면체의 크기가 (12x12x12)이고 직육면체 게임 객체의 위치가 (0.0f, 0.0f, 0.0f)이며 카메라의 위치가 (0.0f, 0.0f, -2.0f)이기 때문이다. 즉, 카메라가 직육면체의 내부에 존재한다.

```
    m_pCamera->GenerateViewMatrix(XMFLOAT3(0.0f, 0.0f, -2.0f), XMFLOAT3(0.0f, 0.0f, 0.0f),
XMFLOAT3(0.0f, 1.0f, 0.0f));
```



카메라가 직육면체의 내부에 있음

직육면체 게임 객체의 위치가 (0.0f, 0.0f, 0.0f)인 것은 월드 변환 행렬(m_xmf4x4World)을 단위행렬로 초기화하였기 때문이다.

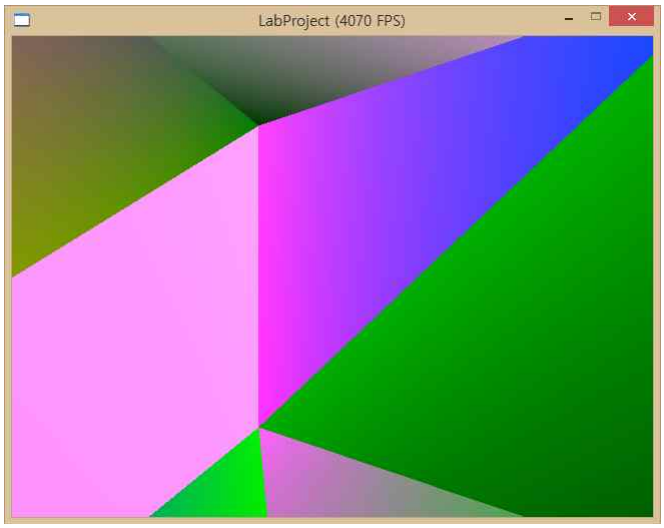
```
m_xmf4x4World = Matrix4x4::Identity();
```

그리고 직육면체 게임 객체를 회전만 시키고 있으므로 게임 객체의 위치(행렬의 _41, _42, _43)가 변하지 않기 때문이다. 그리고 다음 그림과 같이 카메라가 직육면체의 내부에 위치하기 때문에 직육면체의 모든 면은 카메라에 보이지 않게 된다. 왜냐하면 직육면체를 그릴 때 래스터라이저 상태의 컬링 모드가 다음과 같이 은면을 제거하도록 설정되어 있기 때문이다.

```
d3dRasterizerDesc.CullMode = D3D12_CULL_MODE_BACK;
d3dRasterizerDesc.FrontCounterClockwise = FALSE;
```

만약 직육면체를 그릴 때 은면을 컬링하지 않으면 다음 그림과 같이 직육면체의 안쪽 면들이 그려질 것이다. 이 경우 "Mesh.cpp" 파일의 CShader::CreateRasterizerState() 함수에서 래스터라이저 상태를 다음과 같이 설정한다.

```
d3dRasterizerDesc.CullMode = D3D12_CULL_MODE_NONE;
```



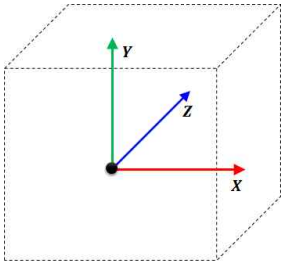
카메라가 객체 내부에 있을 때 은면을 컬링하지 않는 경우

❷ "GameFramework.cpp" 파일 변경하기

"CGameFramework" 클래스의 BuildObjects() 멤버 함수에서 카메라의 위치를 직육면체 바깥쪽에 위치하도록 다음과 같이 변경한다. 직육면체의 가로, 세로, 높이가 12이므로 카메라의 위치를 (0.0f, 15.0f, -25.0f)로 변경하여 카메라와 게임 객체 사이의 거리를 조금 늘리도록 한다.

```
m_pCamera->GenerateViewMatrix(XMFLOAT3(0.0f, 15.0f, -25.0f), XMFLOAT3(0.0f, 0.0f, 0.0f), XMFLOAT3(0.0f, 1.0f, 0.0f));
```

다시 빌드하여 프로젝트를 실행해 보면 회전하는 직육면체를 제대로 볼 수 있을 것이다.



카메라가 직육면체 바깥쪽에 위치