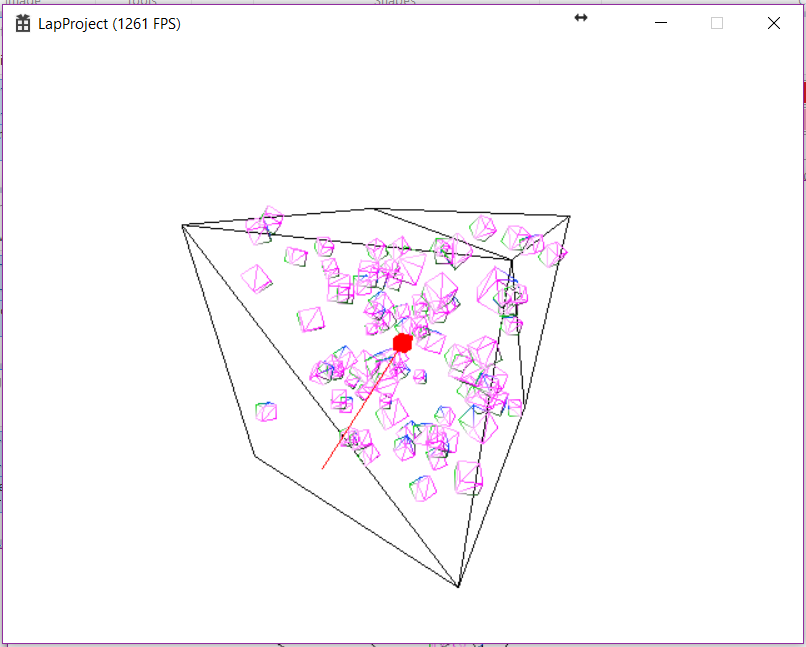
3D게임 프로그래밍 과제 2

컴퓨터 공학과

2011150041 정종선



실행 화면 – 클릭된 육면체는 사라지고 레이저만 보이는 장면

1. 과제 목표

- 과제 1에서 win32 api를 통해 렌더링을 진행하였던 것을 쉐이더 프로그래밍을 통해 렌더링 하도록 소스를 수정한다.

- 마우스로 돌아다니는 육면체를 클릭시 해당 육면체로 레이저가 발사되며 해당 육면체는 더 이상 렌더링 되지 않는다.

1. 과제 범위

* 돌아다니는 육면체의 개수는 100개로 제한한다.
* 돌아다니는 육면체는 와이어 프레임으로 그린다.
* 캐릭터 육면체는 빨간색이며 솔리드로 그린다.
* 캐릭터 육면체는 움직이지 않는다.
* 클릭시 가장 앞에 있는 (카메라와 가까운) 육면체가 소멸된다.

1. 조작법

* 돌아다니는 객체를 마우스로 클릭하면 해당 객체가 랜더 되지 않는다.
* 마우스 클릭 이외의 조작법은 없다.

1. 소스 코드 설명
2. 쉐이더를 이용하여 렌더링

기본적으로 따라하기10 (LabProject07) 을 기반으로 제작하였다.

1. 육면체의 이동 범위를 제한하는 박스 오브젝트

PLANE\_WIDTH = 250;

class CBoxObject : public CGameObject

{

public:

CBoxObject();

virtual ~CBoxObject();

D3DXPLANE m\_Plane[6];

D3DXVECTOR3 m\_Normal[6];

D3DXVECTOR3 m\_point[6];

virtual void Render(ID3D11DeviceContext \*pd3dDeviceContext);

};

// 여섯개의 평면을 가지고 있으며 평면의 노말벡터와 평면위의 점을 가지고 있다.

CCubeMesh \*pBoxMesh = new CCubeMesh(pd3dDevice, PLANE\_WIDTH, PLANE\_WIDTH , PLANE\_WIDTH , D3DXCOLOR(0.0f,0.0f,0.0f,0.0f),1);

// 박스에 사용될 메쉬는 검정색으로 와이어 프레임으로 구성된다.

// 마지막 int 형 파라미터는 솔리드인지 와이어 프레임인지를 결정 (1 = 와이어프레임)

CCubeMesh::CCubeMesh(ID3D11Device \*pd3dDevice, float fWidth, float fHeight, float fDepth, D3DXCOLOR m\_d3dxcDiffuse,int Rast) : CMesh(pd3dDevice)

{

…

if(Rast == 1) CreateRasterizerState(pd3dDevice);

else if(Rast == 2) CreateRasterizerStateSolid(pd3dDevice);

…

}

1. 플레이어 캐릭터

CCubeMesh \*pPlayerMesh = new CCubeMesh(pd3dDevice, 15.0f, 15.0f, 15.0f, D3DXCOLOR(1.0f, 0.0f, 0.0f, 0.0f),2);

// 플레이어 캐릭터는 빨간색의 솔리드로 구성되어있다.

1. 움직이는 육면체들

pObject->SetPosition((float)ur(dre), (float)ur(dre), (float)ur(dre));

pObject->m\_ObjectMoveVector = D3DXVECTOR3((float)speedur(dre), (float)speedur(dre), (float)speedur(dre));

void CRotatingObject::Animate(float fTimeElapsed)

{

Rotate(rand() % 180 \* fTimeElapsed, rand() % 270 \* fTimeElapsed, rand() % 180 \* fTimeElapsed);

Translate(m\_ObjectMoveVector\*fTimeElapsed);

}

// 움직이는 육면체도 솔리드로 구성되었으며 회전값과 이동값을 랜덤으로 가지고 있다.

// 반사 벡터를 구하는 소스는 지난 과제에 제출하였기에 적지 않겠다.

1. 픽킹

* 기본적으로 따라하기 16의 픽킹 소스를 그대로 가져왔다.

CGameObject \*CShader::PickObjectByRayIntersection(D3DXVECTOR3 \*pd3dxvPickPosition, D3DXMATRIX \*pd3dxmtxView, MESHINTERSECTINFO \*pd3dxIntersectInfo)

->

CGameObject \*CScene::PickObjectByRayIntersection(D3DXVECTOR3 \*pd3dxvPickPosition, D3DXMATRIX \*pd3dxmtxView, MESHINTERSECTINFO \*pd3dxIntersectInfo)

// 기존 소스는 쉐이더객체에서 오브젝트를 관리했지만 제 프로젝트에서는 씬에서 관리를 하기 때문에 해당 함수를 처리하는 객체가 바뀌었습니다.

int CMesh::CheckRayIntersection(D3DXVECTOR3 \*pd3dxvRayPosition, D3DXVECTOR3 \*pd3dxvRayDirection, MESHINTERSECTINFO \*pd3dxIntersectInfo)

{ …

CDiffusedVertex\* pbPositions = m\_pd3dxvPositions;

…

// 기존 소스는 포지션버퍼와 색상 버퍼를 따로쓰지만, 제 코드에서는 하나의 버퍼에서 모두 관리하기 때문에 해당 버퍼에서 포지션 데이터만 가져오도록 소스를 수정했습니다.

v0 = pbPositions[m\_pnIndices[(i\*nOffset) + 0]].GetPosition();

v1 = pbPositions[m\_pnIndices[(i\*nOffset) + 1]].GetPosition();

v2 = pbPositions[m\_pnIndices[(i\*nOffset) + 2]].GetPosition();

…

1. 픽킹된 객체에게 레이저 발사

CGameObject \*CScene::PickObjectPointedByCursor(int xClient, int yClient)

{

// 따로 함수를 만들지 않고 픽킹 함수 내부에 구현하였다.

…

if (pNearestObject)

{

m\_pLineMesh = new CLineMesh(m\_pd3dDevice, D3DXVECTOR3(0, 0, 0), D3DXVECTOR3(pNearestObject->m\_d3dxmtxWorld.\_41, pNearestObject->m\_d3dxmtxWorld.\_42, pNearestObject->m\_d3dxmtxWorld.\_43));

m\_LineObject->SetMesh(m\_pLineMesh);

onClick = true;

} …

}

// 객체가 픽킹된 경우에 원점에서 해당 객체로 레이저를 그리는 매쉬를 생성, 오브젝트에   
// 셋 한다.

void CScene::Render(ID3D11DeviceContext \*pd3dDeviceContext, CCamera \*pCamera)

{ …

if (onClick)

{

m\_LineObject->Render(pd3dDeviceContext);

lineRanderCount++;

if (lineRanderCount > 50)

{

onClick = false;

lineRanderCount = 0;

}

}

…

}

// 레이저가 한번만 렌더되면 눈에 보이지 않아서 50번 렌더하여 보이도록 했다.

class CScene

{

public:

…

ID3D11Device \*m\_pd3dDevice;

…

} // 메쉬를 어디서나 생성할 수 있도록 D3DDevice를 CScene이 가지도록 했다.

CLineMesh::CLineMesh(ID3D11Device \*pd3dDevice, D3DXVECTOR3 begin, D3DXVECTOR3 end) : CMesh(pd3dDevice)

{

…

m\_d3dPrimitiveTopology = D3D11\_PRIMITIVE\_TOPOLOGY\_LINELIST;

CDiffusedVertex pVertices[2];

pVertices[0] = CDiffusedVertex(begin, D3DXCOLOR(1.0f, 0.0f, 0.0f, 1.0f));

pVertices[1] = CDiffusedVertex(end, D3DXCOLOR(1.0f, 0.0f, 0.0f, 1.0f));

…

}

// LineMesh는 LineList Topology를 사용하였다.