**3d게임프로그래밍 과제 1**

**2018182017 송주석**

**1.조작법**

기존

방향키 – 전후좌우 움직임

‘A’키 – 모든 적 폭파

‘ctrl’키 – 플레이어 미사일 발사

숫자 키 – 해당 번호에 해당하는 객체 폭파 (최대 9번까지 현재는 객체가 2개이므로 2번까지만 활성화)

마우스 움직임 – 플레이어 화면 이동 및 플레이어 객체 방향 전환

추가 및 수정

마우스 좌클릭 – start를 누르면 게임 시작, 플레이중 적 객체를 누르면 미사일 발사시

적 객체를 추적함

‘S’키 – 플레이어 실드 사용

**2.과제에 대한 목표**

1. Player.h에 있는 CPlayer객체에 game\_start 변수를 추가하여 false면 시작화면, true면 게임을 시작하는 용도로 사용한다. bool game\_start = false;

Scene.h에 m\_nObjects와 같이 객체의 개수를 나타내는 m\_nText, m\_ppObjects와 같이 모든 객체 유형을 수용할 수 있도록 CGameObject 클래스의 포인터 m\_ppText를 선언한다

int m\_nText = 0;

CGameObject \*\*m\_ppText = NULL;

Scene.cpp에 start글씨를 만드는 작업을 한다.

CCubeMesh\* cTextMesh = new CCubeMesh(1.0, 1.0, 1.0);

#define size 76 // 배열 크기

m\_nText = size;

m\_ppText = new CGameObject \* [m\_nText];

// 시작버튼 위치 배열

std::array<float, size> x\_list = {…}; std::array<float, size> y\_list = {…};

start를 만들기 위한 객체에 대한 x좌표, y좌표 리스트를 넘겨 오브젝트를 생성한다

for (int i = 0; i < m\_nText; ++i)

make\_start(cTextMesh, m\_ppText, i, x\_list[i] + 9.0, y\_list[i] - 5.0);

오브젝트 생성은 make\_start함수를 통해 이루어진다.

void make\_start(CCubeMesh\* mesh, CGameObject\*\* obj, int num, float x, float y) {

CText\* pText = new CText();

pText->SetMesh(mesh);

pText->SetColor(RGB(255, 125, 0));

pText->SetPosition(x, y, 40.0f);

obj[num] = pText;

}

그리고 CScene::Animate함수와 CScene::Render함수에 시작화면에 대한 코드를 추가한다

void CScene::Animate(float fElapsedTime)

{

m\_pWallsObject->Animate(fElapsedTime);

// 게임시작 추가

if (m\_pPlayer->game\_start)

for (int i = 0; i < m\_nObjects; i++)

m\_ppObjects[i]->Animate(fElapsedTime);

CheckPlayerByWallCollision();

CheckObjectByWallCollisions();

CheckObjectByObjectCollisions();

CheckObjectByBulletCollisions();

}

void CScene::Render(HDC hDCFrameBuffer, CCamera\* pCamera)

{

CGraphicsPipeline::SetViewport(&pCamera->m\_Viewport);

CGraphicsPipeline::SetViewPerspectiveProjectTransform(&pCamera->m\_xmf4x4ViewPerspectiveProject);

m\_pWallsObject->Render(hDCFrameBuffer, pCamera);

// 시작화면 추가

if (!m\_pPlayer->game\_start)

for (int i = 0; i < m\_nText; ++i) {

m\_ppText[i]->Render(hDCFrameBuffer, pCamera);

}

// 플레이 화면

else {

for (int i = 0; i < m\_nObjects; i++)

m\_ppObjects[i]->Render(hDCFrameBuffer, pCamera);

}

if (m\_pPlayer) {

m\_pPlayer->Render(hDCFrameBuffer, pCamera);

}

//UI

#ifdef \_WITH\_DRAW\_AXIS

CGraphicsPipeline::SetViewOrthographicProjectTransform(&pCamera->m\_xmf4x4ViewOrthographicProject);

m\_pWorldAxis->SetRotationTransform(&m\_pPlayer->m\_xmf4x4World);

m\_pWorldAxis->Render(hDCFrameBuffer, pCamera);

#endif

}

그리고 이 오브젝트를 클릭하면 game\_start가 true가 되어 게임이 시작되게 한다.

Scene.cpp에 있는 CScene::PickObjectPointedByCursor함수에 이전에 큐브 오브젝트를 클릭하던 코드를 수정하여 추가하였다.

if (!m\_pPlayer->game\_start) {

for (int i = 0; i < m\_nText; i++)

{

float fHitDistance = FLT\_MAX;

nIntersected = m\_ppText[i]->PickObjectByRayIntersection(xmvPickPosition, xmmtxView, &fHitDistance);

if ((nIntersected > 0) && (fHitDistance < fNearestHitDistance))

{

fNearestHitDistance = fHitDistance;

pNearestObject = m\_ppText[i];

}

}

}

이렇게 해서 start오브젝트를 클릭했을 때 반응하게 만들고 GameFramework.cpp에 있는

CGameFramework::OnProcessingMouseMessage함수에 클릭이 되면 게임이 시작되게 하는 코드를 추가하였다.

void CGameFramework::OnProcessingMouseMessage(HWND hWnd, UINT nMessageID, WPARAM wParam, LPARAM lParam)

{

if (m\_pScene) m\_pScene->OnProcessingMouseMessage(hWnd, nMessageID, wParam, lParam);

switch (nMessageID)

{

case WM\_RBUTTONDOWN:

case WM\_LBUTTONDOWN:

::SetCapture(hWnd);

::GetCursorPos(&m\_ptOldCursorPos);

if (nMessageID == WM\_LBUTTONDOWN)

{

m\_pLockedObject = m\_pScene->PickObjectPointedByCursor(LOWORD(lParam), HIWORD(lParam), m\_pPlayer->m\_pCamera);

if (m\_pLockedObject) {

// 게임 시작

if (!m\_pPlayer->game\_start)

m\_pPlayer->game\_start = true;

// 플레이 중

else

m\_pLockedObject->SetColor(RGB(0, 0, 0));

}

}

break;

case WM\_LBUTTONUP:

::ReleaseCapture();

break;

case WM\_RBUTTONUP:

case WM\_MOUSEMOVE:

break;

default:

break;

}

}

텍스트, 도표, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

이렇게 해서 스타트 버튼을 누르면 게임이 시작되게 한다

텍스트, 스케치, 평행, 도표이(가) 표시된 사진

자동 생성된 설명

2. 적기들이 미사일 발사

기존에 큐브가 날아 다니던 것을 비행기로 대체하면서 기존 코드를 수정하였다.

CAirplaneMesh\* pAirplaneMesh = new CAirplaneMesh(6.0f, 6.0f, 1.0f);

m\_nObjects = 2;

m\_ppObjects = new CGameObject \* [m\_nObjects];

CExplosiveObject \*pExplosiveObject = new CExplosiveObject();

pExplosiveObject->SetMesh(pAirplaneMesh);

pExplosiveObject->SetColor(RGB(255, 0, 0));

pExplosiveObject->SetPosition(40.0f, 0.0f, 40.0f);

pExplosiveObject->SetMovingDirection(XMFLOAT3(1.0f, 0.0f, 0.0f));

pExplosiveObject->SetMovingSpeed(10.5f);

pExplosiveObject->Rotate(90.0f, 190.0f);

m\_ppObjects[0] = pExplosiveObject;

pExplosiveObject = new CExplosiveObject();

pExplosiveObject->SetMesh(pAirplaneMesh);

pExplosiveObject->SetColor(RGB(255, 0, 0));

pExplosiveObject->SetPosition(-40.0f, -10.0f, 40.0f);

pExplosiveObject->SetMovingDirection(XMFLOAT3(1.0f, 0.0f, 0.0f));

pExplosiveObject->SetMovingSpeed(10.5f);

pExplosiveObject->Rotate(90.0f, 190.0f);

m\_ppObjects[1] = pExplosiveObject;

Scene.cpp에있는 CScene::BuildObjects함수에서 Ccubemesh 포인터를 CAirplanemesh포인터로 바꾸고

오브젝트 개수를 2개로 줄이고, SetRotationAxis, SetRotationSpeed를 없애고, 위치를 수정하고 적 비행기가 플레이어를 바라보게 Rotate함수를 사용했다.

적 비행기가 미사일을 발사할 수 있게 GameObject.h에 CBulletObject클래스를 생성하고

class CBulletObject : public CRotatingObject

{

public:

CBulletObject(float fEffectiveRange);

virtual ~CBulletObject();

public:

virtual void Animate(float fElapsedTime);

float m\_fBulletEffectiveRange = 50.0f;

float m\_fMovingDistance = 0.0f;

float m\_fRotationAngle = 0.0f;

XMFLOAT3 m\_xmf3FirePosition = XMFLOAT3(0.0f, 0.0f, 1.0f);

float m\_fElapsedTimeAfterFire = 0.0f;

float m\_fLockingDelayTime = 0.3f;

float m\_fLockingTime = 4.0f;

CGameObject\* m\_pLockedObject = NULL;

void SetFirePosition(XMFLOAT3 xmf3FirePosition);

void Reset();

};

CExplosiveObject클래스에 Player클래스에 있는 미사일에 관련된 함수와 변수들을 선언한다.

float m\_fBulletEffectiveRange = 150.0f;

CBulletObject\* m\_ppBullets[BULLETS];

void FireBullet(CGameObject\* pLockedObject);

FireBullet함수 및 미사일 관련된 함수들은 CAirplanePlayer클래스에 있는 CAirplanePlayer클래스에 있는 함수들을 참고하여 만들었다.

CExplosiveObject::CExplosiveObject()

{

CCubeMesh\* pBulletMesh = new CCubeMesh(1.0f, 4.0f, 1.0f);

for (int i = 0; i < BULLETS; i++)

{

m\_ppBullets[i] = new CBulletObject(m\_fBulletEffectiveRange);

m\_ppBullets[i]->SetMesh(pBulletMesh);

m\_ppBullets[i]->SetRotationAxis(XMFLOAT3(0.0f, 1.0f, 0.0f));

m\_ppBullets[i]->SetRotationSpeed(360.0f);

m\_ppBullets[i]->SetMovingSpeed(50.0f);

m\_ppBullets[i]->SetActive(false);

}

}코드 수정

CExplosiveObject::~CExplosiveObject()

{

for (int i = 0; i < BULLETS; ++i)

if (m\_ppBullets[i])

delete m\_ppBullets[i];

} 코드 수정

void CExplosiveObject::FireBullet(CGameObject\* pLockedObject)

{

CBulletObject\* pBulletObject = NULL;

if (!m\_bBlowingUp) {

for (int i = 0; i < BULLETS; i++)

{

if (!m\_ppBullets[i]->m\_bActive)

{

pBulletObject = m\_ppBullets[i];

break;

}

}

}

if (pBulletObject)

{

XMFLOAT3 xmf3Position = GetPosition();

XMFLOAT3 xmf3Direction = GetUp();

XMFLOAT3 xmf3FirePosition = Vector3::Add(xmf3Position, Vector3::ScalarProduct(xmf3Direction, 6.0f, false));

pBulletObject->m\_xmf4x4World = m\_xmf4x4World;

pBulletObject->SetFirePosition(xmf3FirePosition);

pBulletObject->SetMovingDirection(xmf3Direction);

pBulletObject->SetColor(RGB(255, 0, 0));

pBulletObject->SetActive(true);

if (pLockedObject)

{

pBulletObject->m\_pLockedObject = pLockedObject;

pBulletObject->SetColor(RGB(0, 0, 255));

}

}

} 함수 추가

Animate함수와 Render함수에 각각

if (!m\_bBlowingUp) {

for (int i = 0; i < BULLETS; ++i)

if (m\_ppBullets[i]->m\_bActive)

m\_ppBullets[i]->Animate(fElapsedTime);

}

if (!m\_bBlowingUp) {

for (int i = 0; i < BULLETS; ++i)

if (m\_ppBullets[i]->m\_bActive)

m\_ppBullets[i]->Render(hDCFrameBuffer, pCamera);

}

을 추가하였다.

CExplosiveObject클래스에 delay라는 변수를 추가하고 Scene.cpp에 CScene::Animate함수에

if (m\_pPlayer->game\_start) {

for (int i = 0; i < m\_nObjects; i++) {

((CExplosiveObject\*)m\_ppObjects[i])->delay += fElapsedTime \* 60;

if (((CExplosiveObject\*)m\_ppObjects[i])->delay > 120 + i\*10) {

((CExplosiveObject\*)m\_ppObjects[i])->FireBullet(m\_pPlayer);

((CExplosiveObject\*)m\_ppObjects[i])->delay = 0;

}

}

}

을 추가하여 일정시간마다 적이 미사일을 객체가 생성된 순서대로 순차적으로 미사일을 발사하게 하였다.

이후 플레이어 충돌 판정을 위해 Scene.h와 Scene.cpp에 CheckPlayerByBulletCollisions함수 추가

3.플레이어가 눈에 보이는 실드 사용