3D게임프로그래밍1
기말고사 대체 과제 설명

제출일: 2022.06.24

전공: 게임공학과

학번: 2019184020

성명: 윤은지

목차

1. 조작법(키)에 대한 설명

```
            break;
            //22.06.21
        case '1':
            m_pScene->choose = false;
            //choose = false;
            break;
        case '2':
            m_pScene->choose = true;
            //choose = false;
            break;
            //
        case VK_F1:
        case VK_F2:
        case VK_F3:
            m_pCamera = m_pPlayer->ChangeCamera((DWORD)(wParam - VK_F1 + 1), m_GameTimer.GetTimeElapsed());
            break;
            //22.06.20
        case 0x52://'R' 'r'
            //m_pCamera = m_pPlayer->ChangeCamera((DWORD)(wParam - VK_F1 + 1), m_GameTimer.GetTimeElapsed());
            //m_pCamera = m_pPlayer->ChangeCamera(FIRST_PERSON_CAMERA, m_GameTimer.GetTimeElapsed());
            //PostQuitMessage(0);

            if (true == lookPoint)
            {
                m_pCamera = m_pPlayer->ChangeCamera(SPACESHIP_CAMERA, m_GameTimer.GetTimeElapsed());
                lookPoint = false;
            }
            else
            {
                m_pCamera = m_pPlayer->ChangeCamera(FIRST_PERSON_CAMERA, m_GameTimer.GetTimeElapsed());
                lookPoint = true;
            }

            break;
            //
```

1 : 첫 번째 씬으로 전환

2 : 두 번째 씬으로 전환

F1 : 1인칭 시점 카메라

F2 : 스페이스쉽 카메라

F3 : 3인칭 시점 카메라

R, r : 스페이스쉽 카메라일 때 1인칭 시점으로 전환 혹은 그 반대

```
HDC hdc;

switch (message)
{
    //22.06.11
//p를 누르면 종료
case WM_CHAR:
    if (wParam == 'P' || wParam == 'p')
        PostQuitMessage(0);
    else if (wParam == 'N' || wParam == 'n')
    {
        if (true == gGameFramework.wakeUp)
            gGameFramework.wakeUp = false;
        else
            gGameFramework.wakeUp = true;

        /*gGameFramework.pfClearColor[0] = 0.0525f;
        gGameFramework.pfClearColor[1] =  0.0525f;
        gGameFramework.pfClearColor[2] = 0.0525f;
        gGameFramework.pfClearColor[3] = 1.0f ;*/
    }
    break;
    //
```

P, p : 종료

N, n : 야간모드로 전환 혹은 야간에서 아침으로 전환

```
void CGameFramework::ProcessInput()
{
    static UCHAR pKeysBuffer[256];
    DWORD dwDirection = 0;
    if (::GetKeyboardState(pKeysBuffer))
    {
        if (pKeysBuffer[0x57] & 0xF0) dwDirection |= DIR_FORWARD;//w
        if (pKeysBuffer[0x53] & 0xF0) dwDirection |= DIR_BACKWARD;//s
        if (pKeysBuffer[0x41] & 0xF0) dwDirection |= DIR_LEFT;//a
        if (pKeysBuffer[0x44] & 0xF0) dwDirection |= DIR_RIGHT;//d
        if (pKeysBuffer[0x58] & 0xF0) dwDirection |= DIR_UP;//x
        if (pKeysBuffer[0x43] & 0xF0) dwDirection |= DIR_DOWN;//c
    }

    float cxDelta = 0.0f, cyDelta = 0.0f;
```

W : 앞으로 이동

S : 뒤로 이동

A : 왼쪽으로 이동

D : 오른쪽으로 이동

X : 위로 이동

C : 아래로 이동

2.씬(Scene)을 구성하여 '1', '2' 키로 씬을 전환

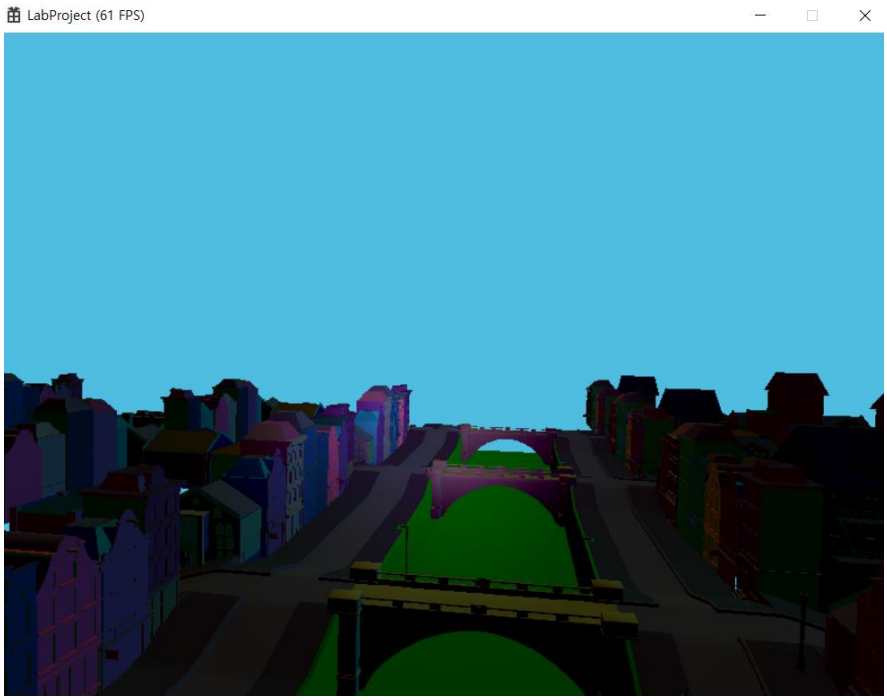
```
//if(false==choose)
mpObjVec = pObjectShader->BuildObjects(pd3dDevice, pd3dCommandList, "Models/Scene.bin");
//else
mpObjVec2 = pObjectShader2->BuildObjects(pd3dDevice, pd3dCommandList, "Models/Scene8.bin");

m_ppShaders[0] = pObjectShader;
m_ppShaders[1] = pObjectShader2;
```

m_ppShaders[0]에 첫 번째 씬을 저장하고 m_ppShaders[1]에 두 번째 씬을 저장하였습니다.

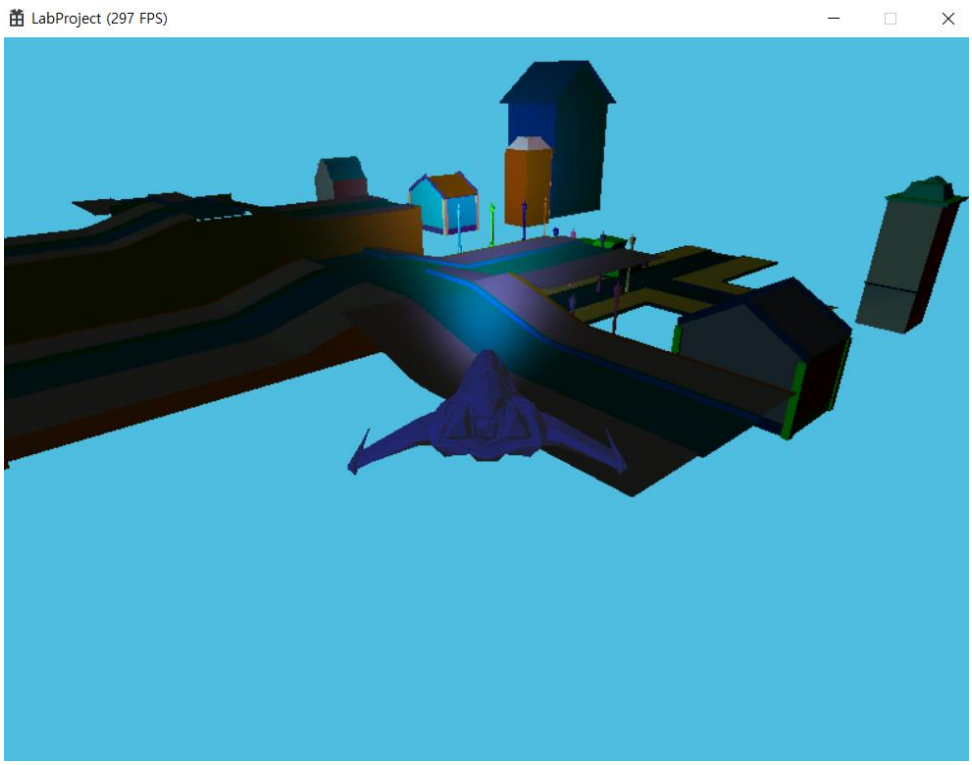
```
void CScene::Render(ID3D12GraphicsCommandList* pd3dCommandList, CCamera* pCamera)
{
    pd3dCommandList->SetGraphicsRootSignature(m_pd3dGraphicsRootSignature);

    pCamera->SetViewportsAndScissorRects(pd3dCommandList);
    pCamera->UpdateShaderVariables(pd3dCommandList);

    UpdateShaderVariables(pd3dCommandList);

    D3D12_GPU_VIRTUAL_ADDRESS d3dcbMaterialsGpuVirtualAddress = m_pd3dcbMaterials->GetGPUVirtualAddress();
    pd3dCommandList->SetGraphicsRootConstantBufferView(2, d3dcbMaterialsGpuVirtualAddress); //Materials

    D3D12_GPU_VIRTUAL_ADDRESS d3dcbLightsGpuVirtualAddress = m_pd3dcbLights->GetGPUVirtualAddress();
    pd3dCommandList->SetGraphicsRootConstantBufferView(3, d3dcbLightsGpuVirtualAddress); //Lights

    if (choose == false)
        m_ppShaders[0]->Render(pd3dCommandList, pCamera);
    else
        m_ppShaders[1]->Render(pd3dCommandList, pCamera);
```

1을 누르면 choose변수가 false로 설정되고 2를 누르면 choose변수가 true로 설정되게 하여서 그에 따라 다른 씬을 렌더했습니다.

씬1

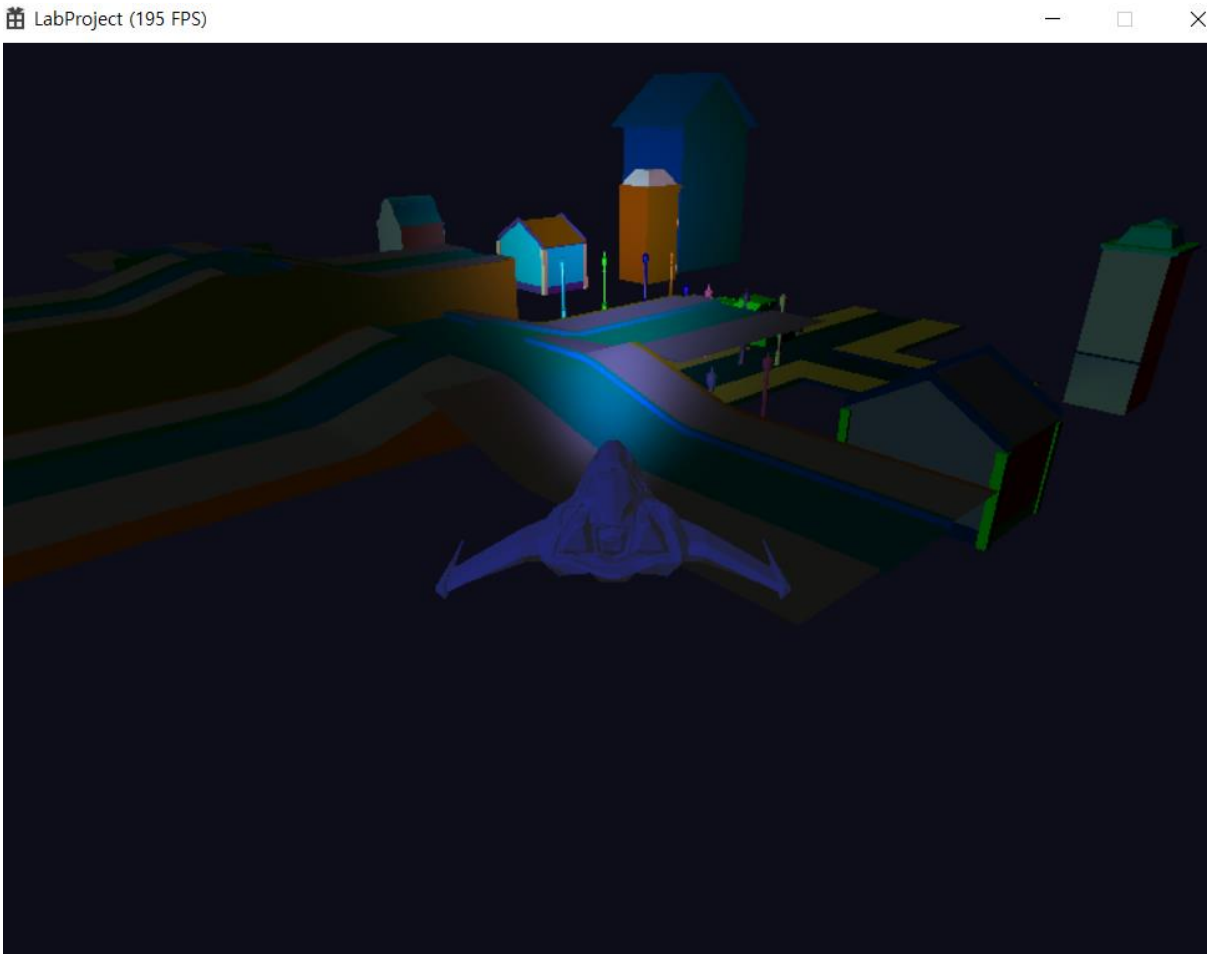


씬2



3.두 번째 씬은 제공된 건물, 도로 등을 자유롭게 배치

씬2는 유니티를 사용해 제공된 건물, 도로 등을 자유롭게 배치하여 만들었습니다.



4.조명 처리

```
vector<XMFLOAT3> CObjectsShader::BuildObjects(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList *pd3dCommandList, char *pstrFileName, void *pContext)
{
    int nSceneTextures = 0;
    m_ppObjects = ::LoadGameObjectsFromFile(pd3dDevice, pd3dCommandList, pstrFileName, &m_nObjects);

    int num = 0;

    if (0 == strcmp("Models/Scene.bin", pstrFileName))
    {
        for (int i = 0; i < m_nObjects; ++i)
        {
            //canale
            if (0 == strcmp(m_ppObjects[i]->m_pstrName, "C4_lamp"))
                //if (0 == strcmp(m_ppObjects[i]->m_pstrName, "C4_pillar"))
                //if (0 == strcmp(m_ppObjects[i]->m_pstrName, "canale"))
                {
                tmp = m_ppObjects[i]->GetPosition();
                mpObjVec.push_back(tmp);



                ++num;
                //cout << "여기 가로등 들어있어요m_ppObjects[i] : " << i << endl;
            }
```

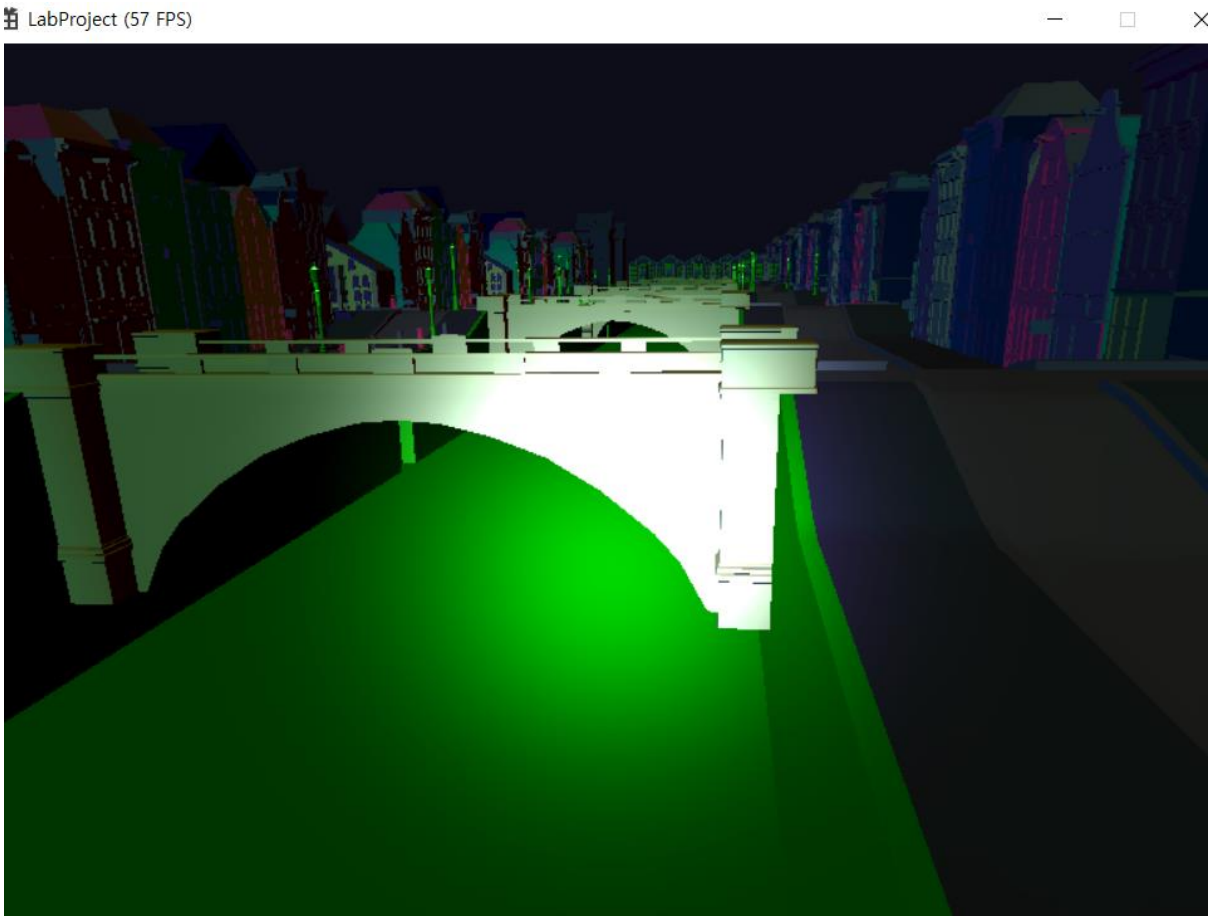
다음 함수에서 c4_lamp라는 이름을 가진 오브젝트의 좌표를 벡터에 저장하여 그 벡터를 반환했습니다.

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

    m_nShaders = 2;
    m_ppShaders = new CShader * [m_nShaders];

    CObjectsShader* pObjectShader = new CObjectsShader();
    pObjectShader->CreateShader(pd3dDevice, m_pd3dGraphicsRootSignature);
    CObjectsShader* pObjectShader2 = new CObjectsShader();
    pObjectShader2->CreateShader(pd3dDevice, m_pd3dGraphicsRootSignature);

    //if(false==choose)
    mpObjVec = pObjectShader->BuildObjects(pd3dDevice, pd3dCommandList, "Models/Scene.bin");
    //else
```

그 반환 값을 다음 함수에서 가져왔습니다.

```
for (int i = 5; i < 24; ++i)
{
    m_pLights->m_pLights[i].m_bEnable = false;
    //m_pLights->m_pLights[i].m_bEnable =wakeUp;//
    //m_pLights->m_pLights[5].m_nType = SPOT_LIGHT;
    m_pLights->m_pLights[i].m_nType = POINT_LIGHT;
    m_pLights->m_pLights[i].m_fRange = 2.0f;

    m_pLights->m_pLights[i].m_xmf4Ambient = XMFLOAT4(0.5f, 0.5f, 0.5f, 5.0f);
    m_pLights->m_pLights[i].m_xmf4Diffuse = XMFLOAT4(0.7f, 0.7f, 0.7f, 7.0f);
    m_pLights->m_pLights[i].m_xmf4Specular = XMFLOAT4(0.7f, 0.7f, 0.7f, 0.0f);
    //m_pLights->m_pLights[5].m_xmf3Position = XMFLOAT3(0.0f, 0.0f, -5.0f);
    m_pLights->m_pLights[i].m_xmf3Direction = XMFLOAT3(0.0f, 0.0f, 1.0f);
    m_pLights->m_pLights[i].m_xmf3Attenuation = XMFLOAT3(1.0f, 0.01f, 0.0001f);
    m_pLights->m_pLights[i].m_fFalloff = 8.0f;
    m_pLights->m_pLights[i].m_fPhi = (float)cos(XMConvertToRadians(40.0f));
    m_pLights->m_pLights[i].m_fTheta = (float)cos(XMConvertToRadians(20.0f));

    m_pLights->m_pLights[i].m_xmf3Position = XMFLOAT3(mpObjVec[i - 5].x, mpObjVec[i - 5].y + 5, mpObjVec[i - 5].z);
    //m_pLights->m_pLights[5].m_xmf3Position = pos;
}
```

위와 같이 벡터에 저장된 좌표로 조명의 좌표를 설정해주었습니다.



N을 누르면 하늘 색깔을 바꾸고 가로등의 조명을 on/off 할 수 있습니다.

```
void CScene::AnimateObjects(float fTimeElapsed)
```

다음 함수에서

```
    for (int i = 5; i < 24; ++i)
        m_pLights->m_pLights[i].m_bEnable = wakeUp;
```

wakeUp 변수 값에 따라서 가로등의 조명을 on/off 했습니다.

```
void CGameFramework::FrameAdvance()
{
```

다음 함수에서

```
    //float pfClearColor[4] = { 0.0525f, 0.0525f, 0.0525f, 1.0f };

    //22.06.14
    if (true == wakeUp)
    {
        /*pfClearColor[0] =1.0f;
        pfClearColor[1] = 1.0f;
        pfClearColor[2] = 1.0f;
        pfClearColor[3] = 1.0f;*/

        //float pfClearColor[4] = { 0.31f, 0.74f, 0.88f, 1.0f };// 하늘 색깔

        pfClearColor[0] = 0.31f;
        pfClearColor[1] = 0.74f;
        pfClearColor[2] = 0.88f;
        pfClearColor[3] = 1.0f;

        m_pScene->wakeUp = false;
    }
    else
    {
        pfClearColor[0] = 0.0525f;
        pfClearColor[1] = 0.0525f;
        //pfClearColor[2] = 0.0525f;
        pfClearColor[2] = 0.1f;
        pfClearColor[3] = 1.0f;

        m_pScene->wakeUp = true;
    }
    //
    m_pd3dCommandList->ClearRenderTargetView(d3dRtvCPUDescriptorHandle, pfClearColo
```

wakeUp 변수 값에 따라서 하늘 색깔을 바꿨습니다.

5.카메라를 이동

```
}

void CCamera::ReleaseShaderVariables()
{
    if (m_pd3dcbCamera)
    {
        m_pd3dcbCamera->Unmap(0, NULL);
    //  m_pd3dcbCamera->Release();
    }
}
```

다음 함수에서 m_pd3dcbCamera->Release();에 주석을 쳐서 카메라의 시점 변환이 가능하게 했습니다.

```
case 0x52://R r
    //m_pCamera = m_pPlayer->ChangeCamera((DWORD)(wParam - VK_F1 + 1), m_GameTimer.GetTimeElapsed());
    //m_pCamera = m_pPlayer->ChangeCamera(FIRST_PERSON_CAMERA, m_GameTimer.GetTimeElapsed());
    //PostQuitMessage(0);

    if (true == lookPoint)
    {
        m_pCamera = m_pPlayer->ChangeCamera(SPACESHIP_CAMERA, m_GameTimer.GetTimeElapsed());
        lookPoint = false;
    }
    else
    {
        m_pCamera = m_pPlayer->ChangeCamera(FIRST_PERSON_CAMERA, m_GameTimer.GetTimeElapsed());
        lookPoint = true;
    }

    break;
    //
```

R을 누를 때마다 1인칭시점 카메라일 때는 스페이스쉽 카메라로 혹은 그 반대로 바뀌게 했습니다.

```
CAirplanePlayer::CAirplanePlayer(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList *pd3dCommandList, ID3D12RootSignature *pd3dGraphicsRootSignature, void *pContext)
{
    LoadGameObjectFromFile(pd3dDevice, pd3dCommandList, "Models/FlyerPlayershipObject.bin");

    CPlayerShader *pShader = new CPlayerShader();
    pShader->CreateShader(pd3dDevice, pd3dGraphicsRootSignature);
    pShader->CreateShaderVariables(pd3dDevice, pd3dCommandList);

    //m_pCamera = ChangeCamera(THIRD_PERSON_CAMERA, 0.0f);
    m_pCamera = ChangeCamera(FIRST_PERSON_CAMERA, 0.0f);

    //SetPosition(XMFLOAT3(-3.944041f, 7.696952f, -2.04254f));
    SetPosition(XMFLOAT3(-3.944041f, 130, -2.04254f));
    //m_xmOOBB = BoundingOrientedBox(XMFLOAT3(-3.944041f, 130, -2.04254f), XMFLOAT3(fHalfWidth, fHalfHeight, fHalfDepth + 0.05f), XMFLOAT4(0.0f, 0.0f, 0.0f, 1.0f));
```

프로그램을 실행했을 때 1인칭 시점 카메라로 볼 수 있게, 그리고 전체 씬을 보기 위해 카메라의 높이를 높였습니다.

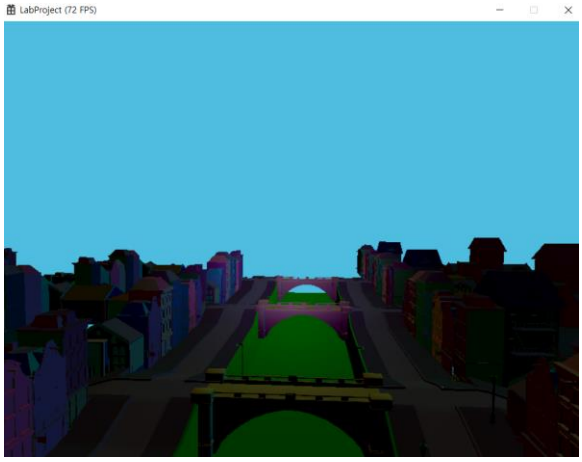
```
    break;
case SPACESHIP_CAMERA:
    SetFriction(125.0f);
    SetGravity(XMFLOAT3(0.0f, 0.0f, 0.0f));
    SetMaxVelocityXZ(400.0f);
    SetMaxVelocityY(400.0f);
    //SetPosition(m_xmf3Position.x, m_xmf3Position.y-122, m_xmf3Position.z);
    m_xmf3Position.y = 8;
    m_pCamera->Rotate(-90, 0, 0);
    //
    m_pCamera = OnChangeCamera(SPACESHIP_CAMERA, nCurrentCameraMode);
    m_pCamera->SetTimeLag(0.0f);
    m_pCamera->SetOffset(XMFLOAT3(0.0f, 0.0f, 0.0f));
    m_pCamera->GenerateProjectionMatrix(1.01f, 5000.0f, ASPECT_RATIO, 60.0f);
    m_pCamera->SetViewport(0, 0, FRAME_BUFFER_WIDTH, FRAME_BUFFER_HEIGHT, 0.0f, 1.0f);
    m_pCamera->SetScissorRect(0, 0, FRAME_BUFFER_WIDTH, FRAME_BUFFER_HEIGHT);
    break;
case THIRD_PERSON_CAMERA:
    //22.06.20
    //카메라 시점
    m_xmf3Position.y = 8;
    //
    SetFriction(250.0f);
    SetGravity(XMFLOAT3(0.0f, 0.0f, 0.0f));
    SetMaxVelocityXZ(125.0f);
    SetMaxVelocityY(400.0f);
```

스페이스쉽 카메라 혹은 3인칭 시점 카메라로 전환될 때에 적절한 높이를 맞추기 위해 m_xmf3Position.y=8;로 설정했습니다.

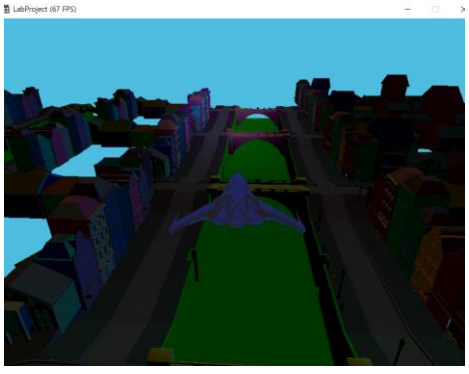
그리고 스페이스쉽 카메라로 전환될 때 1인칭 시점 카메라에서 각도를 로테이트 했기에 m_pCamera->Rotate(-90, 0, 0)을 해서 각도를 맞춰주었습니다.

1인칭 시점 카메라

스페이스쉽 카메라



3인칭 시점 카메라



6.충돌처리

```
CAirplanePlayer::CAirplanePlayer(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList *pd3dCommandList, ID3D12RootSignature *pd3dGraphicsRootSignature, void *pContext)
{
    LoadGameObjectFromFile(pd3dDevice, pd3dCommandList, "Models/FlyerPlayershipObject.bin");

    CPlayerShader *pShader = new CPlayerShader();
    pShader->CreateShader(pd3dDevice, pd3dGraphicsRootSignature);
    pShader->CreateShaderVariables(pd3dDevice, pd3dCommandList);

    //m_pCamera = ChangeCamera(THIRD_PERSON_CAMERA, 0.0f);
    m_pCamera = ChangeCamera(FIRST_PERSON_CAMERA, 0.0f);

    //SetPosition(XMFLOAT3(-3.944041f, 7.696952f, -2.04254f));
    SetPosition(XMFLOAT3(-3.944041f, 130, -2.04254f));
    //m_xmOOBB =BoundingOrientedBox(XMFLOAT3(-3.944041f, 130, -2.04254f), XMFLOAT3(fHalfWidth, fHalfHeight, fHalfDepth * 0.05f), XMFLOAT4(0.0f, 0.0f, 0.0f, 1.0f));
    m_xmOOBB = BoundingBox(XMFLOAT3(-3.944041f, 130, -2.04254f), XMFLOAT3(10, 10, 10));
```

다음과 같이 플레이어의 바운딩박스의 초기값을 설정했습니다.

```
void CPlayer::UpdateBoundingBox()
{
    //if (m_pMesh)
    //{
    //  m_pMesh->m_xmBoundingBox.Transform(m_xmOOBB, XMLoadFloat4x4(&m_xmf4x4World));
    //  //XMStoreFloat4(&m_xmOOBB.Orientation, XMQuaternionNormalize(XMLoadFloat4(&m_xmOOBB.Orientation)));
    //}

    m_xmOOBB.Center = m_xmf3Position;
}
//
```

다음 함수에서 플레이어의 바운딩박스의 센터값을 업데이트하고

```
void CPlayer::Animate(float fElapsedTime)
{

    OnUpdateTransform();
    UpdateBoundingBox();

    //CGameObject::Animate(fElapsedTime);
}
```

다음 함수에서 위의 함수를 호출했습니다.

```
void CGameFramework::AnimateObjects()
{
    if (m_pScene)
        m_pScene->AnimateObjects(m_GameTimer.GetTimeElapsed());

    //22.06.23
    if (m_pPlayer)
        m_pPlayer->Animate(m_GameTimer.GetTimeElapsed());
    //
}
```

그리고 다음 함수에서 위의 함수를 호출했습니다.

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

    m_nShaders = 2;
    m_ppShaders = new CShader * [m_nShaders];

    CObjectsShader* pObjectShader = new CObjectsShader();
    pObjectShader->CreateShader(pd3dDevice, m_pd3dGraphicsRootSignature);
    CObjectsShader* pObjectShader2 = new CObjectsShader();
    pObjectShader2->CreateShader(pd3dDevice, m_pd3dGraphicsRootSignature);

    //if(false==choose)
    mpObjVec = pObjectShader->BuildObjects(pd3dDevice, pd3dCommandList, "Models/Scene.bin");
    //else
    mpObjVec2 = pObjectShader2->BuildObjects(pd3dDevice, pd3dCommandList, "Models/Scene8.bin");

    m_ppShaders[0] = pObjectShader;
    m_ppShaders[1] = pObjectShader2;

    for (int i = 0; i < m_ppShaders[0]->m_nObjects; ++i)
    {
        //m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center = m_ppShaders[0]->m_ppObjects[i]->GetPosition();

        m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center.x = m_ppShaders[0]->m_ppObjects[i]->GetPosition().x;
        m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center.y = m_ppShaders[0]->m_ppObjects[i]->GetPosition().y + 6;
        m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center.z = m_ppShaders[0]->m_ppObjects[i]->GetPosition().z;
    }
```

위와 같이 m_ppObjects[i]->GetPosition()으로 m_ppObjects[i]의 바운딩 박스의 센터값을 초기화 했습니다.

```
void CScene::CheckObjectByObjectCollisions()
{
    if (false == choose)
    {
        for (int i = 0; i < m_ppShaders[0]->m_nObjects - 1; i++)
        {
            //m_ppShaders[0]->m_ppObjects[i]->m_pMesh->m_xmBoundingBox
            if (m_pPlayer->m_xmOOBB.Intersects(m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB))
                //if (m_pPlayer->m_xmOOBB.Intersects(m_ppShaders[0]->m_ppObjects[i]->m_pMesh->m_xmBoundingBox))
            {
                //if(m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center==XMFLOAT3(0,0,0))
                if (!(m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center.x == 0 && m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center.y == 0 &&
                    m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB.Center.z == 0))
                {
                    m_pPlayer->m_pObjectCollided = m_ppShaders[0]->m_ppObjects[i];
                    //m_ppShaders[i]->m_pObjectCollided = m_pPlayer;

                    XMFLOAT3 xmfsub = m_ppShaders[0]->m_ppObjects[i]->GetPosition();
                    xmfsub = Vector3::Subtract(m_pPlayer->GetPosition(), xmfsub);

                    xmfsub = Vector3::Normalize(xmfsub);
                    m_pPlayer->SetPosition(XMFLOAT3(m_pPlayer->GetPosition().x + xmfsub.x, m_pPlayer->GetPosition().y + xmfsub.y, m_pPlayer->GetPosition().z + xmfsub.z));
                }
            }
            //}
        }
    }
```

다음과 같이 충돌함수를 구현하고

```
    //m_ppShaders[0]->m_ppObjects[i]->m_pMesh->m_xmBoundingBox
    if (m_pPlayer->m_xmOOBB.Intersects(m_ppShaders[0]->m_ppObjects[i]->m_xmOOBB))
        //if (m_pPlayer->m_xmOOBB.Intersects(m_ppShaders[0]->m_ppObjects[i]->m_pMesh->m_xmBoun
    {
```

충돌했을 때

```
    //m_ppShaders[i]->m_pObjectCollided = m_pPlayer;

    XMFLOAT3 xmfsub = m_ppShaders[0]->m_ppObjects[i]->GetPosition();
    xmfsub = Vector3::Subtract(m_pPlayer->GetPosition(), xmfsub);
```

오브젝트에서 플레이어 쪽으로 향하는 벡터를 구하고

```
xmfsub = Vector3::Normalize(xmfsub);
m_pPlayer->SetPosition(XMFLOAT3(m_pPlayer->GetPosition().x + xmfsub.x, m_pPlayer->GetPosition().y + xmfsub.y, m_pPlayer->GetPosition().z + xmfsub.z));
```

그 벡터를 정규화하여 플레이어의 좌표에 더했습니다.

```
void CScene::AnimateObjects(float fTimeElapsed)
{
```

충돌함수를 다음 함수에서 호출하였습니다.