Project : It takes two (3D 개인프로젝트 최성희) _ 시스템 구조

코드 목적 :

It takes two(rpg)에서는 탑 뷰 시점으로 게임이 진행됩니다.
맵 크기에 비해 뷰포트 상에 보이는 오브젝트의 수가 적다는 장점을 활용하기 위해 구현했습니다.

코드 설명 :

1. 절두체 컬링 여부를 검사해주는 함수 (Culling_Tick())를 모든 오브젝트의 최상위 부모 클래스에 추가했습니다.
2. 검사 결과를 저장하는 변수(m_bCulling)도 최상위 부모클래스에 추가했습니다.
3. 저장된 변수를 사용하여 화면에 보이지 않는 물체는 Tick(), LateTick() 함수 진입을 막아 갱신되지 않도록 했습니다.
4. 또한, 플레이어가 두 명이고 오브젝트와의 상호작용이 많은 게임 특성상 충돌검사 대상이 많았기 때문에 위와 비슷한 방식으로 충돌검사를 진행했습니다.
5. 최상위 부모 클래스에 충돌검사리스트에 넣어주는 함수(Add_Collider())를 추가했습니다.
6. 오브젝트 타입에 따라 서로 다른 충돌 리스트에 추가가 이루어져야 하고, 충돌검사를 하지 않는 오브젝트들도 있기 때문에 해당 함수는 오버라이딩하여 각 클래스에서 재정의해주었습니다.
7. 함수(Add_Collider()) 재정의 시 화면에 보이는 않는 물체는 충돌 리스트에 추가되지 않습니다.

```
/* for.Culling */
void Culling_Tick();
/*for. Add Culling Object */
virtual void Add_Collider() {}
```

```
void CGameObject::Culling_Tick()
{
    CGameInstance* pGameInstance = GET_INSTANCE(CGameInstance);

    if (pGameInstance->isInCamera(m_pTransform->Get_State(CTransform::STATE_POSITION), m_fCullingRadius))
    {
        m_bCulling = true;
    }
    else
    {
        m_bCulling = false;
    }

    RELEASE_INSTANCE(CGameInstance);
}
```

```
void CToyMage::Add_Collider()
{
    if (nullptr == m_pAABBCollisionCom || true == m_bCulling)
        return;

    CGameInstance* pGameInstance = GET_INSTANCE(CGameInstance);
    pGameInstance->Add_ColliderGroup(CCollider_Manager::COLLIDER_MONSTER, m_pAABBCollisionCom);
    RELEASE_INSTANCE(CGameInstance);
}
```

Project : It takes two (3D 개인프로젝트 최성희) _ Fog

코드 목적 :

용암으로 인한 안개효과를 표현했습니다.

코드 설명 :

1. 게임 내 월드 스페이스 상 플레이어의 Y축 이동범위가 고정되어 있고, 용암은 플레이어의 Y값에서 8 만큼 아래에 있는 특징을 이용했습니다.

2. 랜더링 파이프라인의 Deferred Rendering 시스템에서 빛 연산 시 사용되는 깊이 랜더타겟은 월드 스페이스 상의 좌표로 변환해줍니다. 따라서 최적화를 위해 빛 연산 시 안개효과를 같이 적용했습니다.

3. fogFactor 값을 이용하여 월드 스페이스상 Y값에 따라 색 변화를 자연스럽게 변화시켰습니다.

```
PS_OUT_LIGHT PS_MAIN_DIRECTIONAL(PS_IN In)
{
    PS_OUT_LIGHT    Out = (PS_OUT_LIGHT)0;

    vector      vNormalDesc = g_NormalTexture.Sample(DefaultSampler, In.vTexUV);
    vector      vDepthDesc = g_DepthTexture.Sample(DefaultSampler, In.vTexUV);
    float       fViewZ = vDepthDesc.x * 300.f;

    vector      vNormal = vector(vNormalDesc.xyz * 2.f - 1.f, 0.f);

    Out.vShade = g_vLightDiffuse * saturate(dot(normalize(g_vLightDir) * -1.f, vNormal)) + (g_vLightAmbient * g_vMtrlAmbient);

    Out.vShade.a = 1.f;

    vector      vReflect = reflect(normalize(g_vLightDir), vNormal);
    vector      vWorldPos;

    /* 로컬위치 * 월드행렬 * 뷰행렬 * 투영행렬 * w나누기. */
    vWorldPos.x = (In.vTexUV.x * 2.f - 1.f);
    vWorldPos.y = (In.vTexUV.y * -2.f + 1.f);
    vWorldPos.z = vDepthDesc.y; /* 0 ~ 1 */
    vWorldPos.w = 1.f;

    /* 로컬위치 * 월드행렬 * 뷰행렬 * 투영행렬 */
    vWorldPos.x = (In.vTexUV.x * 2.f - 1.f) * fViewZ;
    vWorldPos.y = (In.vTexUV.y * -2.f + 1.f) * fViewZ;
    vWorldPos.z = vDepthDesc.y * fViewZ; /* 0 ~ f */
    vWorldPos.w = 1.f * fViewZ; /* near ~ far */

    /* 로컬위치 * 월드행렬 * 뷰행렬 */
    vWorldPos = mul(vWorldPos, g_ProjMatrixInv);  /* 뷰스페이스 */


    /* 로컬위치 * 월드행렬 */
    vWorldPos = mul(vWorldPos, g_ViewMatrixInv); /* 월드 스페이스 */

    vector      vLook = vWorldPos - g_vCamPosition;

    Out.vSpecular = (g_vLightSpecular * g_vMtrlSpecular) * pow(saturate(dot(normalize(vReflect) * -1.f, normalize(vLook))), 30.f);
    Out.vSpecular.a = 0.f;


    float fogFactor = saturate((vWorldPos.y + 8.f) / (8.f));
    float4 fogColor = float4(1.0f, 0.2f, 0.1f, 1.0f);
    Out.vFog = (1.0 - fogFactor) * fogColor;

    return Out;
}
```