

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

## 코드 목적 :

It takes two(rpg)에서는 **탐 뷰** 시점으로 게임이 진행됩니다.  
맵 크기에 비해 뷰포트 상에 보이는 오브젝트의 수가 적다는 장점을 활용하여 **최적화**를 하기 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) \_ Skill

### 코드 목적 :

특정 스킬 사용 시 해당 스킬을 편리하게 사용하고자 구현했습니다.

### 코드 설명 :

1. 우측 사진은 플레이어 클래스에서 키 입력을 받아 스킬 사용하는 함수의 일부입니다.
2. 스킬은 콤보 공격 3단계, 대쉬, 공중에서 내려찍는 공격, 필살기가 있습니다. 스킬은 기본 2~3개의 이펙트를 가지고 있고, 이펙트마다 지속시간이 다르며, 지속시간에 따라 공격 지속시간도 영향을 받습니다.
3. 따라서 플레이어의 경우 **이펙트 클래스**를 따로 만들어 주었고, 이펙트의 생성과 소멸은 이펙트 클래스에서 그 기능을 구현했습니다.
4. 플레이어는 이펙트 객체를 소유(m\_MayEffectSkill)함으로써 멤버 변수를 통하여 해당 스킬의 **Start 함수 또는 Stop 함수**를 통해 제어할 수 있도록 구현했습니다.

```
if (false == m_bDashStart && false == m_bGrandSmashStart && false == m_bComboING ) /*Dash, Groundsmash 애니메이션의 끝*/
{
    /*기본 움직임*/
    BasicMove(&pGameInstance);

    /* SKILL : Dash */
    if (pGameInstance->Get_DIKeyState(DIK_SPACE) && true == m_bDashCoolTimeEnd && false == m_bWhirlWindStart)
    {
        m_eDynamicState = CDynamicObject::DYNAMICSTATE_ATTACKING;
        m_pCamera_dynamic->RandomShake(TimeDelta * 15.f);
        m_bDashCoolTimeEnd = false;
        m_bDashStart = true;
        m_pModelCom->Setup_AnimIndex(DASH, true);
        CSoundMgr::GetInstance()->PlaySound(TEXT("May_Dash.mp3"), CSoundMgr::PLAYER_MAY_EFFECT, 1.0f);
    }

    /* SKILL : GROUND SMASH */
    else if (pGameInstance->Get_DIKeyState(DIK_Y) && true == m_bGrandSmashCoolTimeEnd && false == m_bWhirlWindStart)
    {
        m_eDynamicState = CDynamicObject::DYNAMICSTATE_ATTACKING;
        m_pCamera_dynamic->RandomShake(TimeDelta * 15.f);
        m_bGrandSmashCoolTimeEnd = false;
        m_bGrandSmashStart = true;
        m_pModelCom->Setup_AnimIndex(GROUND_SMASH, true);
        m_pMayEffectSkill->Start_GroundSmash();
        CSoundMgr::GetInstance()->PlaySound(TEXT("May_GroundSkill.mp3"), CSoundMgr::PLAYER_MAY_EFFECT, 1.0f);
    }

    /* SKILL : 필살기 */
    else if (pGameInstance->Get_DIKeyState(DIK_U) && 100.f <= m_CurrentSkillBar && false == m_bWhirlWindStart)
    {
        m_eDynamicState = CDynamicObject::DYNAMICSTATE_ATTACKING;
        m_pCamera_dynamic->RandomShake(TimeDelta * 15.f);
        m_bWhirlWindStart = true;
        m_pModelCom->Setup_AnimIndex(ULTIMATEWHIRLWIND_MH, false, 0.1f);
        /* Effect : 필살기 */
        m_pMayEffectSkill->Start_Whirlwind();
        m_pMayEffectSkill->Stop_SkillFullGauge();
        CSoundMgr::GetInstance()->PlaySound(TEXT("May_WhirlWind.mp3"), CSoundMgr::PLAYER_MAY_EFFECT, 1.0f);

        /* for. SkillSound */
        m_FullSkillTimer = 0.f;
        m_bFullSkillGaugeSound = false;
    }
}

else if(true == m_bDashStart)
{
    m_bDashSkillEffectTimer += TimeDelta * 2.f;
    m_pTransform->Go_Straight(TimeDelta * 4.0f, m_pNavigationCom);
    m_eDynamicState = CDynamicObject::DYNAMICSTATE_ATTACKING;
    m_pMayEffectSkill->Start_Dash();
}
```

## Project : It takes two (3D 개인프로젝트 최성희) \_ Skill UI

### 코드 목적 :

필살기 스킬 사용을 위한 **게이지**를 나타내기 위하여 구현했습니다.

### 코드 설명 :

1. 위 사진은 필살기 사용을 위한 게이지 UI의 **버텍스 셰이더**입니다.
2. `g_vStartPosX`는 UI 버퍼의 시작 지점 `float4(-0.5f, 0.5f, 0.f, 1.f)`  
`g_vEndPosX`는 UI 버퍼의 끝 지점 `float4(0.5f, 0.5f, 0.f, 1.f)`을 나타냅니다.
3. 시작과 끝 지점의 좌표를 월드, 뷰, 투영 행렬을 통하여 뷰포트 상의 좌표로 변환해 주었습니다.
4. 셰이더 전역변수로 현재 게이지(`g_vGauge`)를 받아와 비율(`GaugeRatio`)로 변환해 주었습니다.
5. 시작 지점(`StartX`)에서 게이지 비율(`GaugeRatio`)만큼 더해주었으며, 그 결과를 픽셀 셰이더로 넘겨주어 게이지 비율만큼 다른 색으로 표현해 주었습니다.

```
VS_OUT VS_MAIN_GAUGE(VS_IN In)
{
    VS_OUT Out = (VS_OUT)0;

    matrix matWV, matWVP, matVP;
    matWV = mul(g_worldMatrix, g_ViewMatrix);
    matWVP = mul(matWV, g_ProjMatrix);
    Out.vPosition = mul(vector(In.vPosition, 1.f), matWVP);

    vector StartPosition = mul(g_vStartPosX, matWVP);
    vector EndPosition = mul(g_vEndPosX, matWVP);

    float StartX = StartPosition.x / Out.vPosition.w;
    float EndX = EndPosition.x / Out.vPosition.w;
    StartX = (1 + StartX) * 1280.f * 0.5f;
    EndX = (1 + EndX) * 1280.f * 0.5f;

    float GaugeRatio = (EndX - StartX) * (g_vGauge * 0.01f);
    StartX += GaugeRatio;

    /* PosforDir : gage의 위치값 */
    Out.vPosforDir = StartX;
    Out.vTexUV = In.vTexUV;
    return Out;
}
```



# Project : It takes two (3D 개인프로젝트 최성희) \_ Skill UI

## 코드 목적 :

스킬 쿨타임을 나타내기 위하여 구현했습니다.

## 코드 설명 :

1. 아래 사진은 스킬 쿨타임이나 리스폰 쿨타임 UI의 픽셀 셰이더의 일부입니다.
2. UI 버퍼의 중점을 기준으로 현재 픽셀의 방향 벡터를 구하였고, 내적과 외적을 통하여 각도를 계산했습니다.
3. 셰이더 전역변수로 현재 각도를 받아와 그 각도 보다 크거나 작을 경우 다르게 표현해 주었습니다.

```
PS_OUT PS_MAIN_RESPAWN(PS_IN In)
{
    PS_OUT Out = (PS_OUT)0;
    Out.vColor = g_DiffuseTexture.Sample(DefaultSampler, In.vTexUV);

    if (Out.vColor.r <= 0.1f && Out.vColor.b <= 0.01f)
        discard;

    vector vStandardVector = vector(0.f, -1.0f, 0.f, 0.f);

    vector vDir1 = vector(In.vPosition) - vector(In.vPosforDir, 1.f);
    vector vNormalizeDir = normalize(vDir1);
    /* 내적결과 */
    float Dot = dot(vStandardVector, vNormalizeDir);
    float Radian = acos(Dot);
    /* 외적 후 내적 */
    float3 vCross = cross(vStandardVector, vDir1);
    float3 vZ = float3(0.f, 0.f, 1.f);
    float vRightorLeft = dot(vCross, vZ);

    float Degree = degrees(Radian);

    /* 0 보다 크면 공행 : 예각, 작으면 둔각 */
    if (vRightorLeft <= 0)
    {
        Degree = 360.f - Degree;
    }
}
```



# Project : It takes two (3D 개인프로젝트 최성희) \_ 용암

## 코드 목적 :

평소 불을 내뿜는 용암이 남자 캐릭터(코디)의 특정 스킬에만 얼도록 했습니다. 용암이 얼었을 때는 두 캐릭터 모두 얼음 위를 지나갈 수 있고, 얼음은 시간이 지나면서 점점 녹아내리기 시작합니다. 녹아내리는 도중에 다시 스킬을 쓰면 초기 상태로 하얗게 얼게 됩니다.

## 코드 설명 :

- 1. 용암은 2가지 타입을 구분했습니다. LAVA\_GENERAL(일정한 시간텀을 두고 솟아 오르는 용암형태), LAVA\_SWITCH(스위치와 충돌 시 솟아 오르는 용암 형태)로 enum으로 구분하여 서로 다른 로직을 실행시켰습니다.
- 2. PROP\_FIRSTSTART 상태에서는 용암이 올라오는 상태이며 어떤 공격에도 충돌되지 않습니다.
- 3. PROP\_FIRSTSTART 상태에서 Y값이 일정 높이만큼 올라오면 PROP\_FIRE로 충돌할 수 있는 용암 상태가 됩니다.
- 4. 용암이 솟아오르는 PROP\_FIRE 상태가 일정 시간 지속되면 PROP\_NONE 상태로 용암도 얼음도 없는 대기상태가 되도록 구현했습니다.
- 5. PROP\_FIRE 상태일 때 남자 캐릭터의 기본 스킬과 충돌 시 PROP\_ICE 상태로 변합니다.
- 6. PROP\_ICE 상태일 경우 시간이 지날수록 점점 빨갱게 됩니다. 일정 시간이 지나면 얼음은 깨지면서 다시 PROP\_NONE 상태로 돌아옵니다
- 7. 또한, PROP\_ICE 상태일 경우에는 재충돌이 일어날 경우에는 다시 하얗게 초기 얼음 상태로 초기화하도록 구현했습니다.

```
class CLava final : public CStaticObject
{
public:
    /* fire 상태에서 충돌 -> Ice 상태로 변환 , PROP_NONE : 불도 물도 없는 용암 상태 */
    enum PROP { PROP_FIRESTART, PROP_FIRE, PROP_ICE, PROP_NONE, PROP_END };
    enum TYPE {LAVA_GENERAL, LAVA_SWITCH, LAVA_END };

    /* Lava Fire Start */
    if (true == m_bLavaSwitchOn && (m_pFire == PROP_FIRESTART || m_pFire == PROP_FIRE))
    {
        _float fPosY = XMVectorGetY(m_pRectTopTransform->Get_State(CTransform::STATE_POSITION));

        m_pRectTopTransform->Go_Straight_SpecificDir(XMVectorSet(0.f, 1.f, 0.f, 0.f), TimeDelta);
        m_pRectTopTransform->Scaling(TimeDelta);
        m_pRectMidTransform->Go_Straight_SpecificDir(XMVectorSet(0.f, 1.f, 0.f, 0.f), TimeDelta);
        m_pRectMidTransform->Scaling(TimeDelta);
        CSoundMgr::GetInstance()->PlaySound(TEXT("LavaUp.MP3"), (CSoundMgr::CHANNELID)m_iChannel, 0.3f);
        if (fPosY >= 0.3f && m_pFire != PROP_FIRE)
        {
            m_pFire = PROP_FIRE;
            m_fAlpha = 0.5f;
        }
        m_fAlpha += TimeDelta ;
    }

    /* Lava Fire 충돌 가능 상태 */
    if (m_pFire == PROP_FIRE)
    {
        m_LavaFireTimer += TimeDelta;
        if (m_LavaFireTimer >= 2.5f)
        {
            m_pFire = PROP_NONE; /* 그냥 비어있는 상태 */
            m_LavaFireTimer = 0.f;
            m_bLavaSwitchOn = false;
        }
    }
}
```





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

## 코드 목적 :

용암으로 인한 안개 효과를 표현했습니다.

## 코드 설명 :

1. 게임 내 월드 스페이스 상 플레이어의 Y축 이동범위가 고정되어 있고, 용암은 플레이어의 Y값에서 8 만큼 아래에 있는 특징을 이용했습니다.
2. 렌더링 파이프라인의 Deferred Rendering 시스템에서 빛 연산 시 사용되는 깊이 랜더타겟은 월드 스페이스 상의 좌표로 변환해줍니다. 따라서 최적화를 위해 빛 연산 시 안개 효과를 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_vMatlAmbient);

    Out.vShade.a = 1.f;

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

    /* 로컬위치 * 월드맵셀 * 뷰맵셀 * 투영맵셀 * 비나누기. */
    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 * 월드맵셀 * 뷰맵셀 * 투영맵셀 */
    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 * 월드맵셀 * 뷰맵셀 */
    vWorldPos = mul(vWorldPos, g_ProjMatrixInv); /* 뷰스페이스 */

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

    vector          vLook = vWorldPos - g_vCamPosition;

    Out.vSpecular = (g_vLightSpecular * g_vMatlSpecular) * 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;
}
```



## Project : It takes two (3D 개인프로젝트 최성희) \_ 네비메쉬&디졸브

### 코드 목적 :

용암에 빠진 플레이어의 죽는 상태를 나타내기 위해 구현했습니다.

### 코드 설명 :

1. 플레이어는 Dash 모드가 아니고, 얼음 상태의 용암과 충돌하지 않은 상태일 경우에 NeviOptionCheck() 함수를 통해 현재 위치해 있는 네비메쉬의 인덱스가 용암 인덱스 인지를 매 프레임 체크합니다.
2. 조건을 만족할 경우 플레이어는 디졸브 효과로 타들어 가는 느낌을 주어 사라지도록 하였습니다.
3. 디졸브 효과는 디졸브 텍스처를 이용하여 색상 값에서 증가량만큼 빼 주어 부분마다 사라지도록 구현했습니다.



```
int CMay::LateTick(_double TimeDelta)
{
    if (0 > __super::LateTick(TimeDelta))
        return -1;

    Damage(TimeDelta);

    /* Nevi Option Check : Dash 모드가 아니고 ICE에 충돌하지 않았을 경우 검사 */
    if (false == m_bDashStart && false == m_bIceLava_Collision)
    {
        NeviOptionCheck(TimeDelta);
    }

    m_fAfterSkillGauge += m_fSkillGauge * 0.3f;

    /* Skill Value Update */
    SkillUIValue(TimeDelta);
}
```

```
PS_PLAYEROUT PS_MAIN_PLAYER(PS_IN In)
{
    PS_PLAYEROUT Out = (PS_PLAYEROUT)0;

    vector vDiffuse = g_DiffuseTexture.Sample(DefaultSampler, In.vTexUV);
    vector vNormalDesc = g_NormalTexture.Sample(DefaultSampler, In.vTexUV);
    vector vDisolveDesc = g_DisolveTexture.Sample(DefaultSampler, In.vTexUV);

    float ClipAmount = vDisolveDesc.r - g_ColorDelta;

    if (ClipAmount < 0.f)
        discard;

    float3 Normal = vNormalDesc.xyz * 2.f - 1.f;
    float3x3 WorldMatrix = float3x3(In.vTangent.xyz, In.vBinormal.xyz, In.vNormal.xyz);
    vNormal = mul(vNormal, WorldMatrix);
    Out.vDiffuse = vDiffuse - g_ColorDelta;
    Out.vNormal = vector(vNormal.xyz * 0.5f + 0.5f, 0.f);
    Out.vDepth = vector(In.vProjPos.w / 300.0f, In.vProjPos.z / In.vProjPos.w, 0.f, 0.f);
    Out.vPlayerDepth = vector(In.vProjPos.w / 300.0f, In.vProjPos.z / In.vProjPos.w, 0.f, 0.f);

    if (Out.vDiffuse.a < 0.1f)
        discard;

    return Out;
}
```