Client Programmer

기술 소개서

오준석

연락처: 010-7206-0506
E-Mail: topojs8@naver.com

Game List

3D Game

언리얼 토너먼트 2004 (Team)
개발 기간 : 18.04.02~18.05.15 (6주)
개발 언어 : C / C++ **개발 환경** : Visual Studio / DirectX11 / MFC / HLSL
Page **3p ~ 8p**

블레이드 앤 소울
개발 기간 : 17.07.02~17.08.09 (5주)
개발 언어 : C / C++ **개발 환경** : Visual Studio / DirectX9 / MFC
Page **9p ~ 14p**

오버워치 (Team)
개발 기간 : 17.06.08~17.07.09 (4주)
개발 언어 : C / C++ **개발 환경** : Visual Studio / DirectX9 / TCP/IP
Page **15p ~ 19p**

2D Game

스톤에이지
개발 기간 : 17.04.03~17.05.02 (4주)
개발 언어 : C / C++ **개발 환경** : Visual Studio / DirectX9 / MFC
Page **20p ~ 27p**

메이플스토리
개발 기간 : 17.03.01~17.03.24 (3주)
개발 언어 : C / C++ **개발 환경** : Visual Studio / WIN_API
Page **28p ~ 32p**

Game List

3D Game | Team

언리얼 토너먼트 2004

개발 기간 : 18.04.02~18.5.15 **(6주)**

개발 언어 : C / C++

개발 인원 : 2명

개발 환경 : Visual Studio / DirectX11 / MFC / HLSL

개발 중점 사항 :

범프맵핑, StreamOutput단계 파티클출력,
안개, 블랜딩, 스텐실링, 쉐도우맵핑,
컴포넌트 패턴, 층단위 네비게이션메쉬

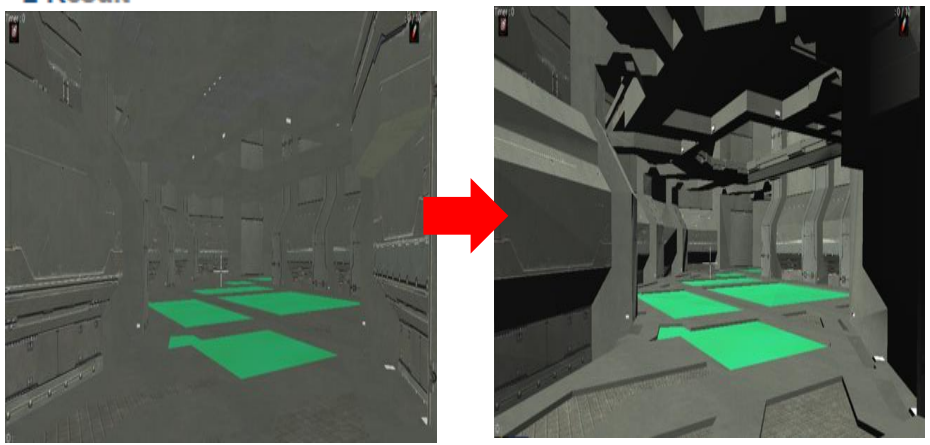
■ 1. Bump Normal Mapping

■ Code

```
float4 WorldMapPS(VertexOut pin, uniform int gLightCount) : SV_Target
{
    float3 normalMapSample = gNormalMap.Sample(samLinear, pin.Tex).rgb;
    float3 bumpedNormalW = NormalSampleToWorldSpace(normalMapSample, pin.NormalW, pin.TangentW);

    for (int i = 0; i < gLightCount; ++i)
    {
        float4 A, D, S;
        ComputeDirectionalLight(gMaterial, gDirLights[i], bumpedNormalW, toEye,
            A, D, S);

        ambient += A;
        diffuse += D;
        spec += S;
    }
    // Modulate with late add. : 변조 후 가산
    litColor = texColor * (ambient + diffuse) + spec;

    return litColor;
}
```

```
void ComputeDirectionalLight(Material mat, Light light, float3 normal, float3 toEye,
    out float4 ambient, out float4 diffuse, out float4 spec)
{
    코드 중략
    float diffuseFactor = dot(lightVec, normal);

    // Flatten to avoid dynamic branching.
    [flatten]
    if (diffuseFactor > 0.0f)
    {
        float3 v = reflect(-lightVec, normal);
        float specFactor = pow(max(dot(v, toEye), 0.0f), mat.Specular.w);

        diffuse = diffuseFactor * mat.Diffuse * light.mat.Diffuse * light.intensity.y;
        spec = specFactor * mat.Specular * light.mat.Specular * light.intensity.z;
    }
}
```

■ Result



■ Comment

정점쉐이더에서 얻어온 노말값을 이용해 폴리곤의 기복정도를 텍스처로 저장후 폴리곤에 입혀 표면을 높낮이가 있게 표현

(조명벡터와 노말값을 내적한값)반사벡터를 구해 반사벡터와 시야벡터를 내적한후 Specular(정반사광)만큼 제곱한 정도로 표현

*쉐이더 최적화: 노말값을 PS에서 연산하지 않고 VS에서 연산함으로써 계산량을 줄였다.

■ 2. Shadow Mapping

■ Code

```
void Engine::CPrimitive::DrawSkinnedShadow(const XMMATRIX & matWorld, const XMFLOAT4X4 & matView, const XMFLOAT4X4 & matProj)
{
    코드 중략
    m_pMyGDI->pTech = m_pMyGDI->pEffect->GetSkinnedTech( );
    for (UINT p = 0; p < techDesc.Passes; ++p)
    {
        ID3DX11EffectPass* pass = m_pMyGDI->pTech->GetPassByIndex(p);

        m_pMyGDI->pContext->IASetVertexBuffers(0, 1, &m_pVB, &stride, &offset);
        m_pMyGDI->pContext->IASetIndexBuffer(m_pIB, DXGI_FORMAT_R32_UINT, 0);

        XMVECTOR pos;
        XMFLOAT3 floatPos;
        pos = matWorld.r[3];
        XMStoreFloat3(&floatPos, pos);

        XMVECTOR shadowPlane = XMVectorSet(0.0f, PlaneY, 0.0f, 0.0f); // xz plane
        XMVECTOR toMainLight = -XMLoadFloat3(&mDirLights[0].Direction);
        XMMATRIX S = XMMatrixShadow(shadowPlane, toMainLight);
        XMMATRIX shadowOffsetY = XMMatrixTranslation(0.0f, 0.01f, 0.0f);
        // Set per object constants.
        XMMATRIX world = matWorld*S*shadowOffsetY;
        XMMATRIX worldInvTranspose = MathHelper::InverseTranspose(world);
        XMMATRIX worldViewProj = world*view*proj;

        m_pMyGDI->pEffect->SetWorld(world);
        m_pMyGDI->pEffect->SetWorldInvTranspose(worldInvTranspose);
        m_pMyGDI->pEffect->SetMaterial(mShadowMat);

        m_pMyGDI->pContext->OMSetDepthStencilState(RenderStates::NoDoubleBlendDSS, 0);
        pass->Apply(0, m_pMyGDI->pContext);
        m_pMyGDI->pContext->DrawIndexed(m_iNumIndices, 0, 0);

        // Restore default states.
        m_pMyGDI->pContext->OMSetBlendState(0, blendFactor, 0xffffffff);
        m_pMyGDI->pContext->OMSetDepthStencilState(0, 0);
    }
}
```

■ Result

■ Comment

Shadow행렬(Plane벡터와 빛의 방향벡터를 XMMarixShadow 범용 그림자 행렬에 대입)을 구해 월드행렬에 곱하여 구현

스텐실 버퍼를 통해 2D평면에 정점이 겹치지 않게 옵션을 설정

Write마스크 설정비트를 반전(XOR)하여 그려야할 영역만 표현

*XMMarixShadow: 물체의 각 정점을 거쳐가는 반직선(평행광방향으로 정점을 거치는 광선)들과 그림자 평면의 교점을 이용해 투영된 물체 그림자를 행렬로 만듬

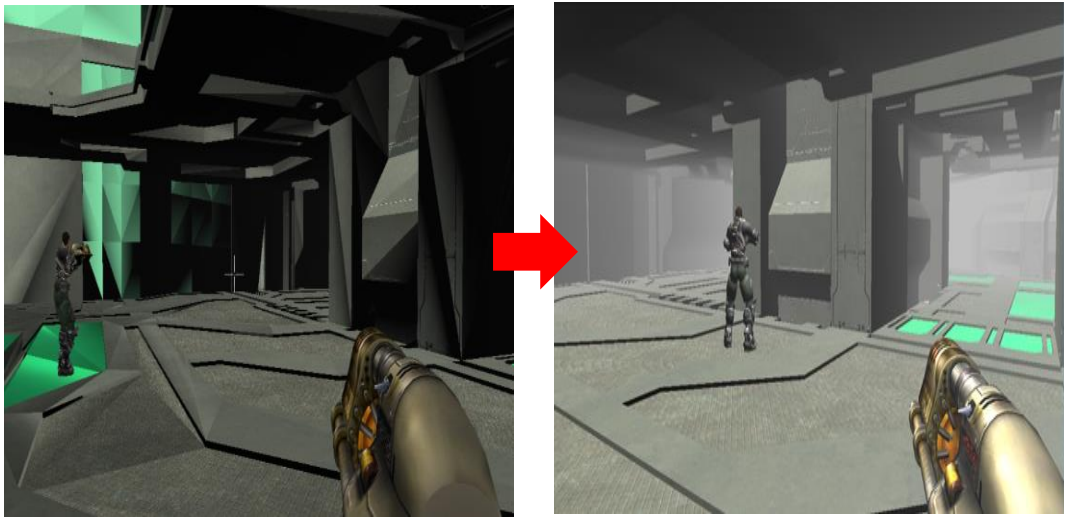
■ 3. Fog

■ Code

```
float4 PS(VertexOut pin, uniform int gLightCount, uniform bool gUseTexture, uniform bool gFogEnabled) : SV_Target
{
    코드 중략
// Fogging
if (gFogEnabled)
{
    float fogLerp = saturate((distToEye - gFogStart) / gFogRange);

    // Blend the fog color and the lit color.
    litColor = lerp(litColor, gFogColor, fogLerp);
}

// Common to take alpha from diffuse material and texture.
litColor.a = gMaterial.Diffuse.a * texColor.a;

return litColor;
}
```

```
int CLandObject::Update(void)
{
    m_pMyGDI->pEffect->SetFogColor(Colors::WhiteSmoke);
    m_pMyGDI->pEffect->SetFogStart(StartFog);
    m_pMyGDI->pEffect->SetFogRange(RangeFog);
        코드 중략
    CGameObject::Update( );
    return 0;
}
```

■ Result



■ Comment

안개 시작위치를 안개범위로 나눠 안개 보간정도 변수를 구한다. PS에서 안개를 적용할 객체의 색상과 안개색상을 안개 보간정도에 따라 선형보간 하여 구현

■ 4. Blending & Stencilling

■ Code

```
void Engine::CBoltTex::Draw(XMFLOAT4X4& rmatWorld)
{
    float blendFactor[] = { 0.0f, 0.0f, 0.0f, 0.0f };
    for (UINT p = 0; p < techDesc.Passes; ++p)
    {
        코드 중략
        m_pMyGDI->pContext->RSSetState(RenderStates::NoCullRS);
        m_pMyGDI->pContext->OMSetBlendState(RenderStates::AdditiveBS, blendFactor, 0xffffffff);
        m_pMyGDI->pContext->OMSetDepthStencilState(RenderStates::DepthWriteOffDSS, 0);

        m_pMyGDI->pTech->GetPassByIndex(p)->Apply(0, m_pMyGDI->pContext);
        m_pMyGDI->pContext->DrawIndexed(mBoltIndexCount, 0, 0);

        // Restore default render state.
        m_pMyGDI->pContext->RSSetState(0);
        m_pMyGDI->pContext->OMSetBlendState(0, blendFactor, 0xffffffff);
        m_pMyGDI->pContext->OMSetDepthStencilState(0, 0);
    }
}
```

```
BlendState AdditiveBlending
{
    AlphaToCoverageEnable = FALSE;
    BlendEnable[0] = TRUE;
    SrcBlend = SRC_ALPHA;
    DestBlend = ONE;
    BlendOp = ADD;
    SrcBlendAlpha = ZERO;
    DestBlendAlpha = ZERO;
    BlendOpAlpha = ADD;
    RenderTargetWriteMask[0] = 0x0F;
};

technique11 LightTex
{
    pass P0
    {
        SetVertexShader(CompileShader(vs_5_0, VS( )));
        SetGeometryShader(NULL);
        SetPixelShader(CompileShader(ps_5_0, PS0RIGIN(1, true)));

        SetBlendState(AdditiveBlending, float4(0.0f, 0.0f, 0.0f, 0.0f), 0xffffffff);
    }
}
```

$C = (a_s * C_{src}) + C_{dst}$: (투명도 * 덮을 color) + 대상의 color

■ Result

■ Comment

블렌드상태를 가산혼합 상태로 바꾸고 스텐실 버퍼 상태를 깊이쓰기 비활성화로 바꿔 색상을 누적되게한다. 깊이 읽기와 깊이판정은 활성화해서 비혼합 기하구조가 뒤에있는 혼합 구조를 가리게 구현 렌더후 블렌드와 스텐실버퍼 상태를 초기화한다.

■ 5. Particle System

■ Code

```
void StreamOutGS(point Particle gin[1],
    inout PointStream<Particle> ptStream)
{
    gin[0].Age += gTimeStep;
    if (gin[0].Type == PT_EMITTER)
    {
        // time to emit a new particle
        if (gin[0].Age > 0.002f)
        {
            for (int i = 0; i < 6; ++i)
            {
                // Spread rain drops out above the camera.
                float3 vRandom = 35.0f*RandVec3((float)i / 6.0f);
                vRandom.y = 20.0f;

                Particle p;
                p.InitialPosW = gEmitPosW.xyz + vRandom;
                p.InitialVelW = float3(0.0f, 0.0f, 0.0f);
                p.SizeW = float2(1.0f, 1.0f);
                p.Age = 0.0f;
                p.Type = PT_FLARE;

                ptStream.Append(p);
            }
            // reset the time to emit
            gin[0].Age = 0.0f;
        }

        // always keep emitters
        ptStream.Append(gin[0]);
    }
    else
    {
        // Specify conditions to keep particle; this may vary from system to system.
        if (gin[0].Age <= 3.0f)
            ptStream.Append(gin[0]);
    }
}
GeometryShader gsStreamOut = ConstructGSWithSO(
    CompileShader(gs_5_0, StreamOutGS()),
    "POSITION.xyz; VELOCITY.xyz; SIZE.xy; AGE.x; TYPE.x");
```

■ Result

■ Comment

렌더타켓에서 렌더링 하지않고 Stream Output 단계에서

Geometry Shader의 출력을 통해 구현 GPU연산 사용 -> Frame Up

*GPU는 스텐실버퍼(깊이),후면버퍼에 자료를 기록하며
SO단계에서 정점버퍼에 기하구조 자체를 직접기록한다

Game List

3D Game

블레이드 앤 소울

개발 기간 : 17.07.02~17.8.09 **(5주)**

개발 언어 : C / C++

개발 인원 : 1명

개발 환경 : Visual Studio / DirectX9 / MFC

개발 중점 사항 :

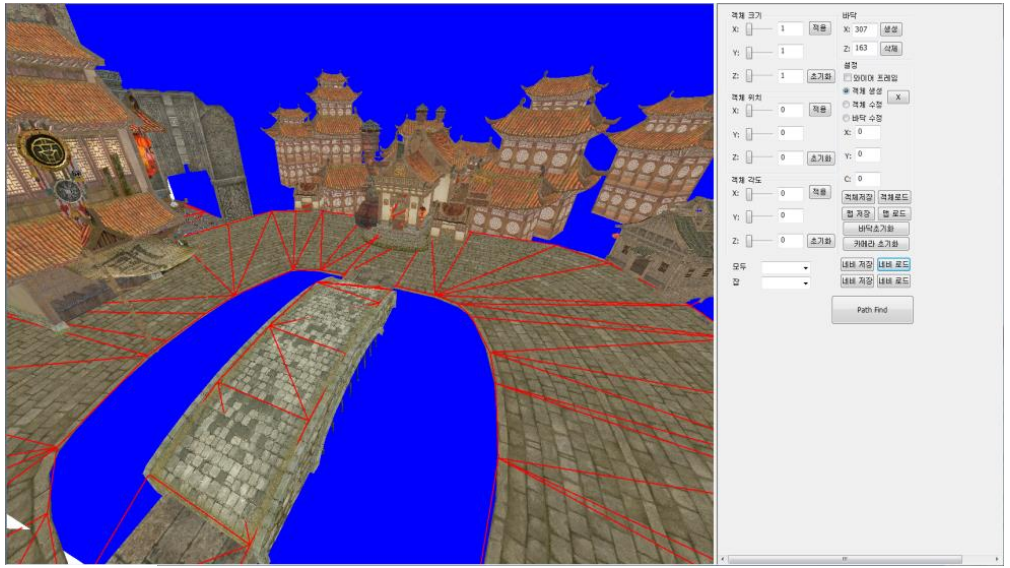
네비게이션메쉬(MFC Tool), 액션카메라, 애니메이션 적용, OBB충돌, 소드트레일, 컴포넌트 패턴, 동적라이브러리(DLL), 램버트 조명모델을 이용한 조명연산

■ 1. MFC Tool – Ray Collision

■ Code

```
bool CMouseCol::PickTerrain(D3DXVECTOR3* pOut)
{
    Translation_ViewSpace();
    D3DXMATRIX      matIdentity;
    D3DXMatrixIdentity(&matIdentity);
    Translation_Local(&matIdentity);

    const Engine::VTXTEX*   pVertex = m_pTerrainVtx;
    const Engine::INDEX32*  pIndex = m_pTerrainIdx;

    float   fU, fV, fDist;
    bool bPickTerrain = false;

    float fDistance = 10000.f;
    D3DXVECTOR3 vecTemp;

    for (int i = 0; i < m_iFaceNum; ++i)
    {
        if (D3DXIntersectTri(&(pVertex[pIndex[i]._1].vPos + 0.01f)
            , &(pVertex[pIndex[i]._2].vPos* 0.01f)
            , &(pVertex[pIndex[i]._3].vPos* 0.01f)
            , &m_vPivotPos, &m_vRayDir, &fU, &fV, &fDist))
        {
            vecTemp = (pVertex[pIndex[i]._1].vPos
                + (pVertex[pIndex[i]._2].vPos - pVertex[pIndex[i]._1].vPos) * fU
                + (pVertex[pIndex[i]._3].vPos - pVertex[pIndex[i]._1].vPos) * fV) * 0.01f;

            // 카메라와 가장 가까운 점을 찾는다.
            if (fDistance > D3DXVec3Length(&(m_vPivotPos - vecTemp)))
            {
                bPickTerrain = true;

                fDistance = D3DXVec3Length(&(m_vPivotPos - vecTemp));
                *pOut = vecTemp;
            }
        }
    }

    return bPickTerrain;
}
```

■ Result



■ Comment

D3DXIntersetTri()함수를 사용하여 카메라에서 쏜 Ray와 충돌된
지점의 버텍스를 Vector에 저장하여 Navigation Mesh를 구현하였다.

■ 2. Client - Navigation Mesh

■ Code

```
void Engine::CNaviMgr::LinkCell(void)
{
    VECCELL::iterator   iter = m_vecNaviMesh.begin();
    if(m_vecNaviMesh.end() == iter)
        return ;

    for( ; iter != m_vecNaviMesh.end(); ++iter)
    {
        VECCELL::iterator   iter_Target = m_vecNaviMesh.begin();
        while(iter_Target != m_vecNaviMesh.end())
        {
            if(iter == iter_Target)
            {
                ++iter_Target;
                continue;
            }
            if((*iter_Target)->ComparePoint((*iter)->GetPoint(POINT_A)
                , (*iter)->GetPoint(POINT_B)
                , (*iter)))
            {
                (*iter)->SetNeighbor(NEIGHBOR_AB, (*iter_Target));
            }
            else if((*iter_Target)->ComparePoint((*iter)->GetPoint(POINT_B)
                , (*iter)->GetPoint(POINT_C)
                , (*iter)))
            {
                (*iter)->SetNeighbor(NEIGHBOR_BC, (*iter_Target));
            }
            else if((*iter_Target)->ComparePoint((*iter)->GetPoint(POINT_C)
                , (*iter)->GetPoint(POINT_A)
                , (*iter)))
            {
                (*iter)->SetNeighbor(NEIGHBOR_CA, (*iter_Target));
            }
            ++iter_Target;
        }
    }
}
```

```
bool Engine::CNaviMgr::FallOnNaviMesh(D3DXVECTOR3 * pPos, const DWORD & dwCurrentIdx)
{
    if (dwCurrentIdx == -1)
        return false;

    D3DXPLANE Plane = m_vecNaviMesh[dwCurrentIdx]->GetPlane();

    float fY = (-Plane.a * pPos->x - Plane.c * pPos->z - Plane.d) / Plane.b;

    if (pPos->y <= fY) // 플레이어의 위치가 더 아래
    {
        pPos->y = fY;
        return true;
    }

    return false;
}
```

■ Comment

벡터에 저장된 정점들을 3개씩 연결하여 네비셀을
만들고 각각의 네비셀 마다 **평면정보**를 멤버변수로
갖고있고, 3차원상의 위치를 받아서 **평면정보** 와 비교하여
플레이어나 몬스터가 NaviMesh위에서 떨어지지 않게 구현

■ Result

■ 3. Attach Weapon Mesh

■ Code

```
int CSphere::Update(void)
{
    if(m_pmatPlayerInfo == NULL || m_pmatWeaponRef == NULL)
    {
        const Engine::CComponent*   pPlayerInfo =  Engine::Get_Management()->GetComponent(
            CStage::LAYER_GAMELOGIC, L"Player", L"Transform");
        const Engine::CComponent*   pPlayerMesh =  Engine::Get_Management()->GetComponent(
            CStage::LAYER_GAMELOGIC, L"Player", L"DynamicMesh");

        m_pmatPlayerInfo = &((static_cast<const Engine::CTransform*>(pPlayerInfo))->m_matWorld));

        m_pmatWeaponRef = ((Engine::CDynamicMesh*)pPlayerMesh)->FindFrame("ValveBiped_Anim_Attachment_RH");
    }
    m_pInfo->m_matWorld = matScale*m_pInfo->m_matWorld * (*m_pmatWeaponRef) * (*m_pmatPlayerInfo);


    return 0;
}
```

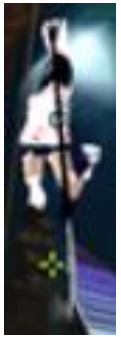
```
const D3DXMATRIX* Engine::CDynamicMesh::FindFrame( const char* pFrameName )
{
    DERIVED_FRAME*      pFrame = (DERIVED_FRAME*)D3DXFrameFind(m_pRootBone, pFrameName);

    return &pFrame->CombinedMatrix;

}
```

■ Comment

FindFrame 함수를 통해 플레이어 오른손 Bone정보의 Matrix를 찾는다.

무기의 월드행렬 = 스케일행렬 X 회전행렬 X 오른손 Bone행렬 X 플레이어 월드행렬 순으로 곱함으로써 구현하였다.

■ Result



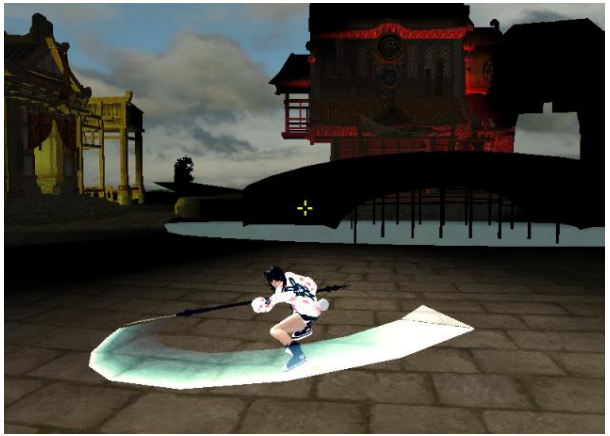
■ 4. Sword Trail

■ Code

```
void Engine::CTrailBuffer::SetVertexTrail(const D3DXVECTOR3* pPos, const DWORD& dwCnt)
{
    m_pVertex = new VTXTRAIL[dwCnt];
    ZeroMemory(m_pVertex, sizeof(VTXTRAIL) * dwCnt);

    for(DWORD i = 0; i < dwCnt; ++i)
    {
        m_pVertex[i].vPos = pPos[i];

        if(i % 2)
            m_pVertex[i].vTexUV = D3DXVECTOR2( 1 / (dwCnt - 1.f), 0.f);
        else
            m_pVertex[i].vTexUV = D3DXVECTOR2( (i - 1) / (dwCnt - 1.f), 1.f);
    }

    m_dwTriCnt = dwCnt - 2;
}
```

```
void CTrailEffect::PointMemorize(void)
{
    while(m_Pointlist.size() >= 20)
    {
        m_Pointlist.pop_front();
    }

    float       fTime = Engine::Get_TimeMgr()->GetTime();
    m_fAccTime += fTime;

    if(0.01f < m_fAccTime)
    {
        D3DXVECTOR3     vPoint[2];
        vPoint[0] = D3DXVECTOR3(50.f, 0.f, 0.f);
        vPoint[1] = D3DXVECTOR3(20.f, 0.f, 0.f);

        for(int i = 0; i < 2; ++i)
        {
            D3DXVec3TransformCoord(&vPoint[i], &vPoint[i], m_pmatWeaponRef);
            D3DXVec3TransformCoord(&vPoint[i], &vPoint[i], m_pmatPlayerInfo);
            m_Pointlist.push_back(vPoint[i]);
        }

        m_fAccTime = 0.f;
    }
}
```

■ Result



■ Comment

D3DXVec3TransformCoord를 사용하여 위치변환 연산한 정점을

List로 관리하면서 시간에 따라 정점을 이동(**보간**)하여 구현

*D3DXVec3TransformCoord:
4x4행렬을 곱하여 벡터(x,y,z,1)로 w값을 1로 만들어
위치변환연산 후 D3DXVECTOR3를 리턴

■ 5. Action Camera

■ Code

```
void CCamera::ActionCam(void)
{
    if (dynamic_cast<Engine::CTransform*>(m_PlayerTrans)->m_tPlayer.m_dwIndex == ACTIONCAMINDEX)
    {
        CSoundMgr::GetInstacne()->PlayBGMSound(L"ogre_shout_boss_01.wav");
        dynamic_cast<Engine::CTransform*>(m_MonsterTrans)->m_tInfo.m_dwState = FightState;
        m_bDynamicCamera = true;
    }

    if (m_bDynamicCamera == TRUE)
    {
        m_fDynamicCamera += Engine::Get_TimeMgr()->GetTime();
        m_pTargetInfo = dynamic_cast<Engine::CTransform*>(m_MonsterTrans);
        D3DXVECTOR3     vLook = m_vAt - m_vEye;
        D3DXVec3Normalize(&vLook, &vLook);
        m_vEye += vLook * m_fCamSpeed * fTime;
        m_vAt += vLook * m_fCamSpeed * fTime;

        D3DXVECTOR3     vRight;
        D3DXMATRIX      matCamState;
        D3DXMatrixInverse(&matCamState, NULL, &m_matView);
        memcpy(&vRight, &matCamState.m[0][0], sizeof(D3DXVECTOR3));
        D3DXVec3Normalize(&vRight, &vRight);
        m_vEye -= vRight * m_fCamSpeed * fTime;
        m_vAt -= vRight * m_fCamSpeed * fTime;
    }
    else
    {
        m_pTargetInfo = dynamic_cast<Engine::CTransform*>(m_PlayerTrans);
    }
    if (m_fDynamicCamera < 3.f)
    {
        Initialize();
        m_bDynamicCamera = false;
    }

    Engine::CCamera::Update();
    Engine::Get_InfoSubject()->Notify(L"Matrix_StaticCamera");
}
```

■ Result

■ Comment

플레이어가 특정 네비게이션 메쉬 인덱스(네비툴에서 저장한정보)에 있을때
카메라의 타겟을 몬스터로 바꾼후 카메라의 Eye값과 At값을 변경하여 구현

Game List

3D Game | Team

오버워치

개발 기간 : 17.06.08~17.7.09 (4주)

개발 언어 : C / C++

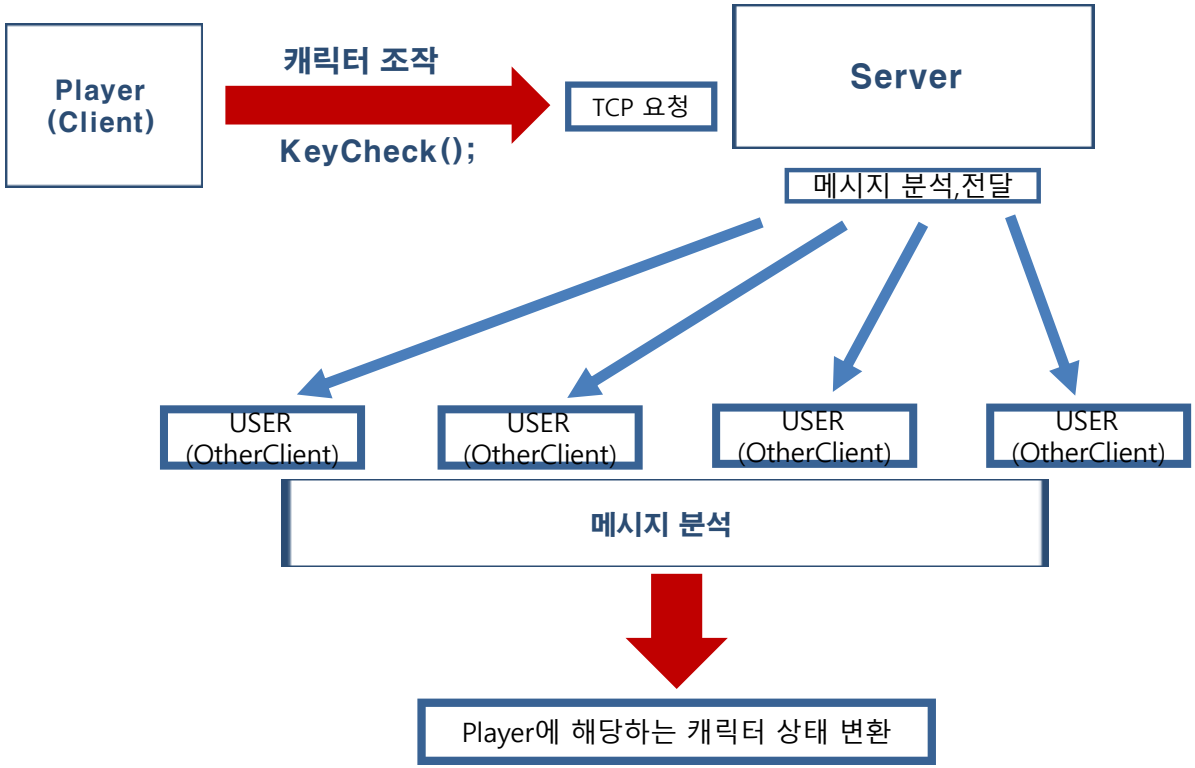
개발 인원 : 4명

개발 환경 : Visual Studio / DirectX9 / TCP/IP

개발 중점 사항 :

TCP/IP소켓 프로그래밍을 이용한 4인용 서버구축,
비트연산을 이용한 KeyManager, AABB,OBB,구충돌,
Scene단위 Layer관리, 컴포넌트 패턴, 빌보드

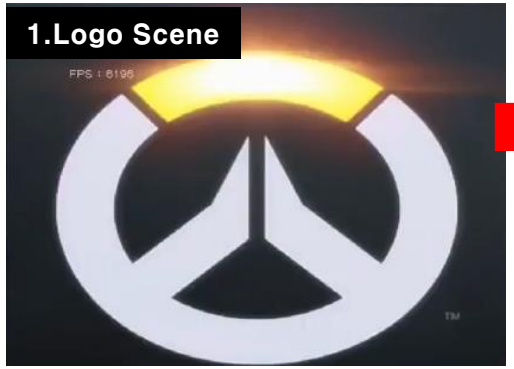
■ 1. TCP/IP Server



Player 1

Player 2

■ 2. Scene Logic & Client Server

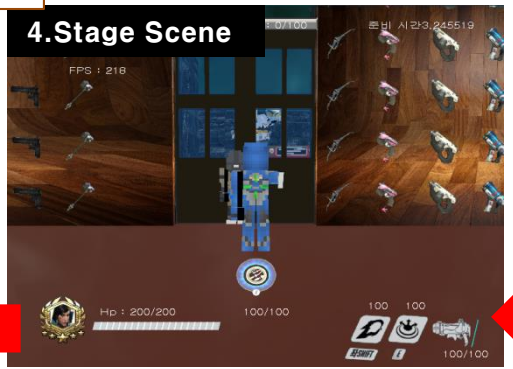


-Select Scene Data 다운로드, Socket();Connect();

-캐릭터 정보 동기화를 위한 Scene

-Stage를 구성하는 Data 다운로드

Send();
recv();



-미션 플레이를 하는 Scene

-미션 완료 시 엔딩 Scene

게임종료시
close();

1. 기분 구성을 위한 데이터는 첫 로딩 시 Load 되며 Socket을 생성하고 connect한다.

2. 모든 리소스 데이터는 다운로드 후 Component로 추가된다.
   *Component 디자인패턴

■ 3. KeyManager(Bit Operation) & Sending Packets

■ Code

```
void CPlayer::KeyCheck(void)
{
    if (GetAsyncKeyState(VK_RBUTTON) && m_tPlayer.paraRightButton > 0.f)
    {
        g_iMsg.iMessage.bNumber |= MOVE_MOVE;
        m_pInfo->m_vPos.y += m_fSpeed * fTime;
        m_tPlayer.paraRightButton -= Engine::Get_
    }
    if (GetAsyncKeyState('W'))
    {
        g_iMsg.iMessage.bNumber |= MOVE_MOVE; // W
        m_pInfo->m_vPos += m_pInfo->m_vDir * m_fSpeed * fTime;
    }
    if (GetAsyncKeyState('S'))
    {
        g_iMsg.iMessage.bNumber |= MOVE_MOVE; // S
        m_pInfo->m_vPos -= m_pInfo->m_vDir * m_fSpeed * fTime;
    }
    if (GetAsyncKeyState('A'))
    {
        g_iMsg.iMessage.bNumber |= MOVE_MOVE; // A
        D3DXVECTOR3 RightVec;
        D3DXVec3Cross(&RightVec, &m_pInfo->m_vDir, &D3DXVECTOR3(0.f, 1.f, 0.f));
        m_pInfo->m_vPos += RightVec * m_fSpeed * fTime;
    }
    if (GetAsyncKeyState('D'))
    {
        g_iMsg.iMessage.bNumber |= MOVE_MOVE; // D
        D3DXVECTOR3 RightVec;
        D3DXVec3Cross(&RightVec, &m_pInfo->m_vDir, &D3DXVECTOR3(0.f, 1.f, 0.f));
        m_pInfo->m_vPos -= RightVec * m_fSpeed * fTime;
    }
}
```

```
#define MOVE_FIRE       0x00000001
#define MOVE_UP         0x00000002
#define MOVE_DOWN       0x00000004
#define MOVE_LEFT       0x00000008
#define MOVE_RIGHT      0x00000010
#define MOVE_JUMP       0x00000020
#define MOVE_RIGHTA     0x00000040
#define MOVE_LEFTA      0x00000080
#define MOVE_ALL        0x000000FF
```

Server

```
DWORD WINAPI ProcessClient(LPVOID arg)
{
    코드 중략
    while(iClientCount > 0)
    {
        msg.iClientNumber = 0;
        msg.iMessage.iNumber = 0;
        msg.iMessage.vDir = D3DXVECTOR3(0,0,0);
        msg.iMessage.vPos = D3DXVECTOR3(0,0,0);
        msg.iMessage.bNumber=0;

        if(m_ESCENE==SCENE_STAGE)
        {
            retval = recv(client, (char*)&msg, sizeof(NET_MSG), 0);
            if( retval == SOCKET_ERROR )
            {
                cout << "수신 에러" << endl; break;
            }

            pMsg.iMessage[iClientNum - 1] = msg.iMessage;

            if( msg.iMessage.iNumber == CLIENT_EXIT )       // 클라이언트 접속 종료
                iClientCount--;

            retval = send(client, (char*)&pMsg, sizeof(P_MSG), 0);
            if( retval == SOCKET_ERROR )
            {
                cout << "송신 에러" << endl; break;
            }
        }
    }
    return 0;
}
```

Client

```
int CPlayer::Update(void)
{
    코드 중략
    g_iMsg.iClientNumber = g_iClientNumber;
    send(g_ClientSocket, (char*)&g_iMsg, sizeof(NET_MSG), 0);
    recv(g_ClientSocket, (char*)&g_iGetMsg, sizeof(P_MSG), 0);
    return 0;
}
```

문제점: 서버가 건네주는 속도보보다 데이터 도착속도가 클때 수신버퍼 TrashData 발견

```
DWORD WINAPI ProcessClient(LPVOID arg)
{
    bool opt_val = TRUE;
    int getrecv = 0;
    int recvlen = sizeof(getrecv);
    int getsend = 0;
    int sendlen = sizeof(getsend);
    setsockopt(client, IPPROTO_TCP, TCP_NODELAY, (char *)&opt_val, sizeof(opt_val));
    getsockopt(client, SOL_SOCKET, SO_RCVBUF, (char *)&getrecv, &recvlen);
    getsockopt(client, SOL_SOCKET, SO_SNDBUF, (char *)&getsend, &sendlen);
    getrecv = getrecv + buferSize;
    getsend = getsend + buferSize;
    setsockopt(client, SOL_SOCKET, SO_RCVBUF, ((char *)&getrecv), sizeof(getrecv));
    setsockopt(client, SOL_SOCKET, SO_SNDBUF, ((char *)&getsend), sizeof(getsend));

    return 0;
}
```

개선방법: 수신버퍼를 수용인원에 맞게 미리 여유공간 확보

■ Comment

Bit연산을 통해 각각의 클라이언트 메시지 정보를 서버를 통해 동기화 하였다

■ 4. AABB, OBB Collision

■ Code

```
bool Engine::CCollisionMgr::CheckCollision(const CCollision_OBB* pTarget)
{
    const OBB*  pOBB[2] = {&m_tOBB, pTarget->GetObbInfo()};

    float       fDistance[3];
    for(int i = 0; i < 2; ++i)
    {
        for(int j = 0; j < AXIS_END; ++j)
        {
            fDistance[0] = fabs(D3DXVec3Dot(&pOBB[0]->vProj[AXIS_X], &pOBB[i]->vParallel[j]))
                + fabs(D3DXVec3Dot(&pOBB[0]->vProj[AXIS_Y], &pOBB[i]->vParallel[j]))
                + fabs(D3DXVec3Dot(&pOBB[0]->vProj[AXIS_Z], &pOBB[i]->vParallel[j]));

            fDistance[1] = fabs(D3DXVec3Dot(&pOBB[1]->vProj[AXIS_X], &pOBB[i]->vParallel[j]))
                + fabs(D3DXVec3Dot(&pOBB[1]->vProj[AXIS_Y], &pOBB[i]->vParallel[j]))
                + fabs(D3DXVec3Dot(&pOBB[1]->vProj[AXIS_Z], &pOBB[i]->vParallel[j]));

            D3DXVECTOR3     vTemp = pOBB[1]->vCenter - pOBB[0]->vCenter;
            fDistance[2] = fabs(D3DXVec3Dot(&vTemp, &pOBB[i]->vParallel[j]));

            if(fDistance[2] > fDistance[0] + fDistance[1])
                return false;
        }
    }
    return true;
}
```

```
bool Engine::CCollisionMgr::Collision_AABB( const D3DXVECTOR3* pDestMin, const D3DXVECTOR3* pDestMax ,
    const D3DXVECTOR3* pSourMin, const D3DXVECTOR3* pSourMax )
{
    float       fMin = 0.f;
    float       fMax = 0.f;

    fMin = max(pDestMin->x, pSourMin->x);
    fMax = min(pDestMax->x, pSourMax->x);
    if(fMin > fMax)
        return false;

    fMin = max(pDestMin->y, pSourMin->y);
    fMax = min(pDestMax->y, pSourMax->y);
    if(fMin > fMax)
        return false;

    fMin = max(pDestMin->z, pSourMin->z);
    fMax = min(pDestMax->z, pSourMax->z);
    if(fMin > fMax)
        return false;

    return true;
}
```

문제점: 모든 회전객체에 OBB충돌 체크
->Frame Drop

➡

개선방법: 구충돌을 만족한 객체만 OBB 검사
->Frame UP

■ Comment

OBB연산은 FrameDrop이
심하므로 세세한 충돌처리가
필요할때만 사용

■ Result

Game List

2D Game

스톤에이지

개발 기간 : 17.04.03~17.5.02 **(4주)**

개발 언어 : C / C++

개발 인원 : 1명

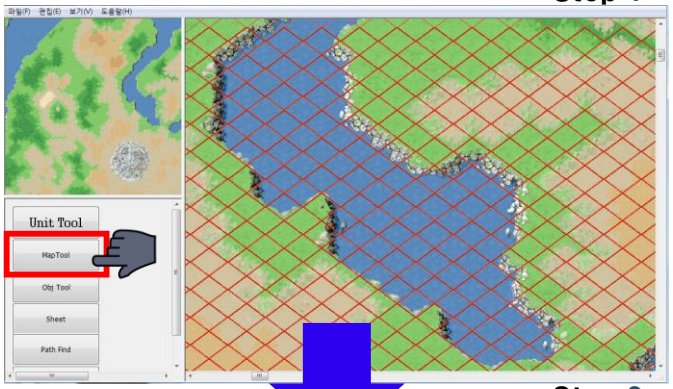
개발 환경 : Visual Studio / DirectX9 / MFC

개발 중점 사항 :

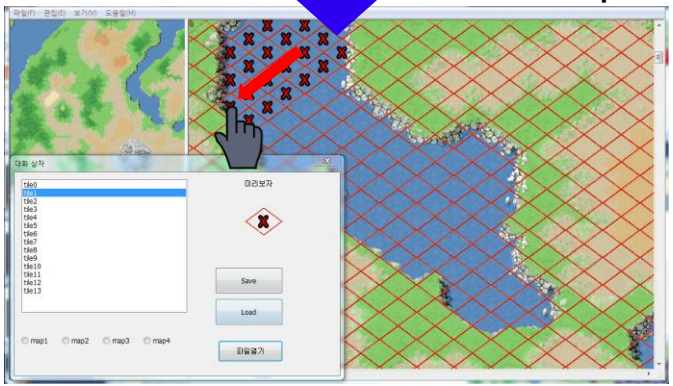
MFC Tool(타일,리소스), PathFinder,
LoadingThread, Astar 알고리즘,
턴제 전투방식, 옵저버 패턴, 브릿지 패턴

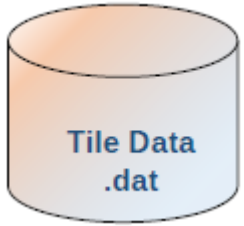
■ 1. MFC Tool

Step 1

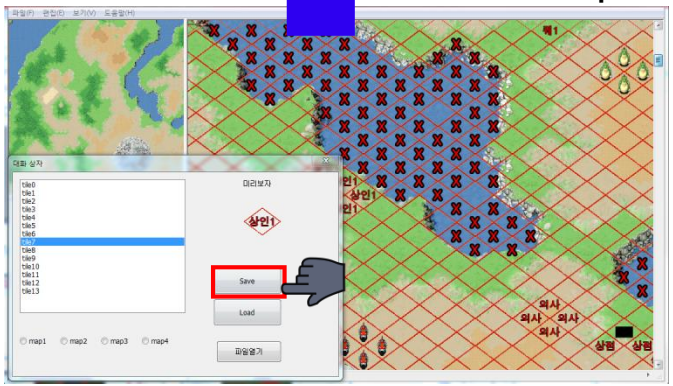


Step 2



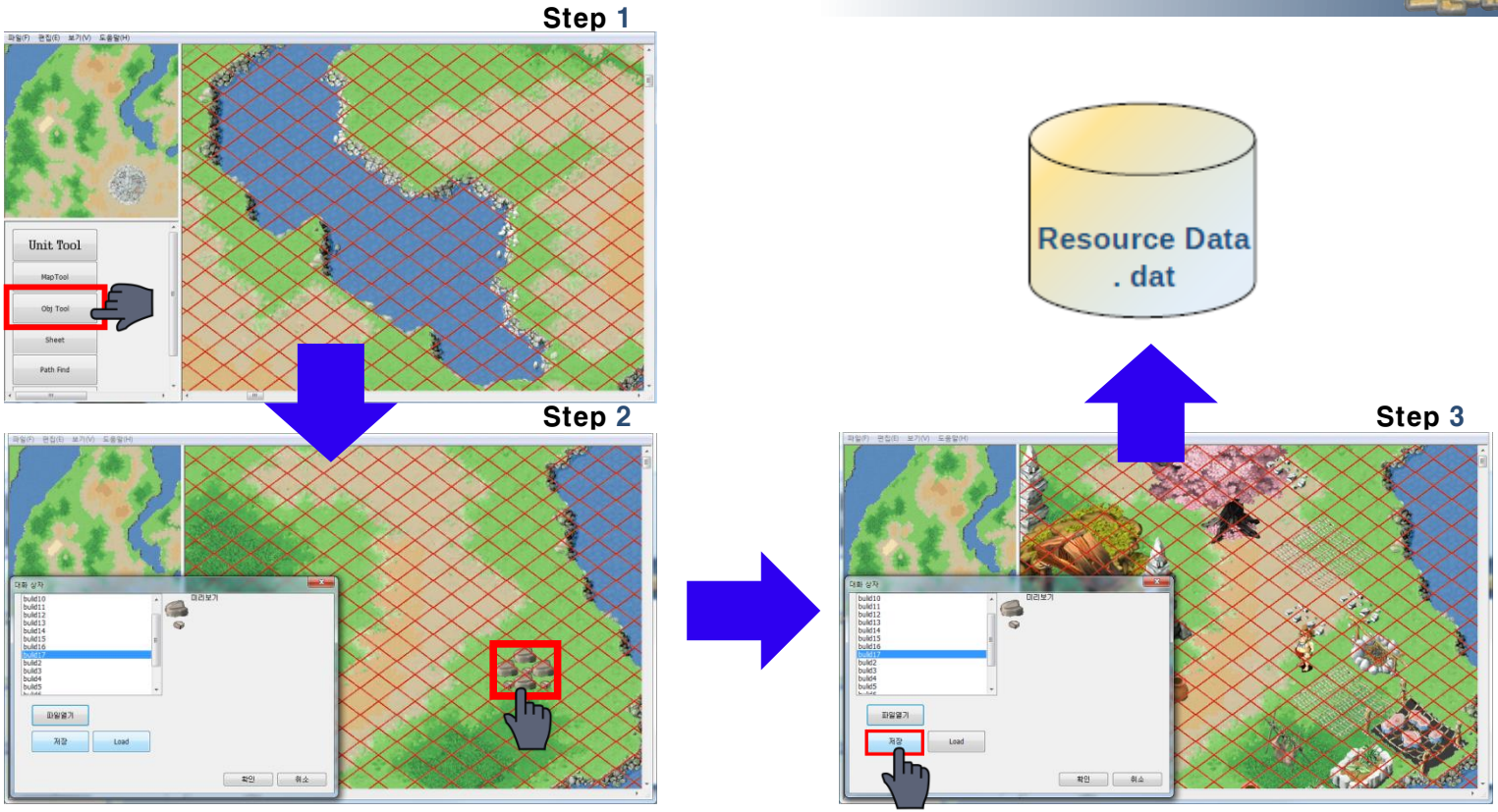


Step 3



1. Tile을 드래그 및 클릭 설치 후 저장

■ 1. MFC Tool



2. Resource를 드래그 및 클릭 설치 후 저장

■ 1. MFC Tool - Mouse Picking

■ Code

```
bool CBackGround::Picking(const D3DXVECTOR3& vPos, const int& iIndex)
{
    //포인트를 구한다.
                    코드 중략
    //방향 벡터
    D3DXVECTOR3 vDir[4] =
    {
        vPoint[1] - vPoint[0],
        vPoint[2] - vPoint[1],
        vPoint[3] - vPoint[2],
        vPoint[0] - vPoint[3]
    };

    //법선벡터를 만들어 보자.
    D3DXVECTOR3 vNormal[4] =
    {
        D3DXVECTOR3(vDir[0].y, -vDir[0].x, 0.f),
        D3DXVECTOR3(vDir[1].y, -vDir[1].x, 0.f),
        D3DXVECTOR3(vDir[2].y, -vDir[2].x, 0.f),
        D3DXVECTOR3(vDir[3].y, -vDir[3].x, 0.f),
    };

    //법선벡터들을 단위벡터로 변경을 한다.
    for(int i = 0; i < 4; ++i)
        D3DXVec3Normalize(&vNormal[i], &vNormal[i]);

    for(int i = 0; i < 4; ++i)
    {
        D3DXVECTOR3 vTemp = vPos - vPoint[i];

        float fDot = D3DXVec3Dot(&vTemp, &vNormal[i]);

        if(fDot > 0.f)
            return false;

        //내적 결과가 양수면 타일 외부이고
        //          음수면 타일 내부이다.
    }
    return true;
}
```

추출된 Tile 과 Resource 데이터

■ Comment

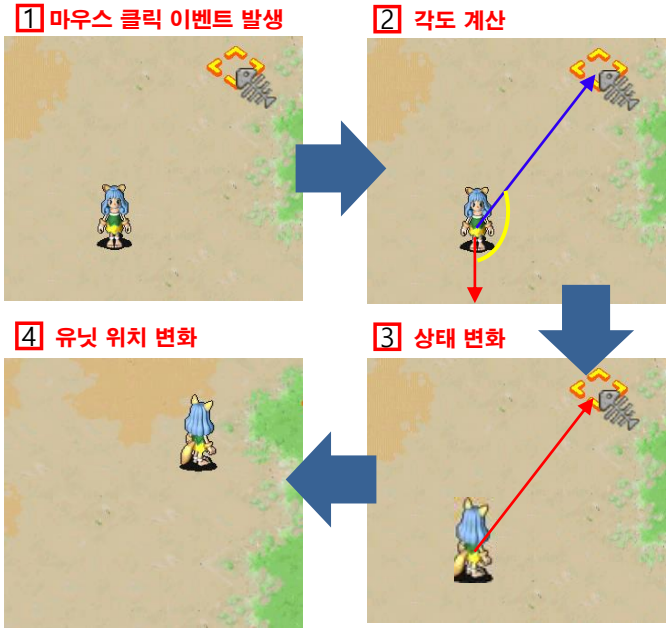
법선벡터와 방향벡터의 내적을 이용하여 타일의 내부를 판단후 해당 타일에 타일정보,리소스정보를 각각 저장하였다 그후 파일입출력을 통하여 추출한 데이터를 클라이언트로 연동하였다.

■ Result

Client에 적용된 모습

■ 2. Client – Move

■ Result



■ Code

```
void CUnit::MoveAngle(int* _iDirection)
{
    m_tInfo.vDir = m_vMousePos - m_tInfo.vPos;

    D3DXVec3Normalize(&m_tInfo.vDir, &m_tInfo.vDir);

    m_fCos = D3DXVec3Dot(&m_tInfo.vLook, &m_tInfo.vDir); //내적
    m_fAngle = D3DXToDegree(acosf((m_fCos)); //각도로 바꿈(라디안)

    // 반대 방향을 확인하기 위해 if문으로 방향체크
    if (m_tInfo.vPos.x > m_vMousePos.x)
        m_fAngle = 360 - m_fAngle;

    // 12시 방향과 6시 방향일 때 각도 틀어짐을 방지
    if (m_tInfo.vPos.x == UNIT_LOOK) // D3DXVECTOR3(0.f, -1.f, 0.f)
        m_fAngle = 0.f;
    else if (m_tInfo.vDir == UNIT_CONTRAY_LOOK) // D3DXVECTOR3(0.f, 1.f, 0.f))
        m_fAngle = 180.f;

    // 각도 구하기 완료 후 각도의 시작을 0도 부터로 맞추기 위해 5.625f를 더해줌
    m_fAngle = m_fAngle + UNIT_CALIBRATE_ANGLE; // 5.625f = 360.f * 0.016f;

    // 0이 5.625가 되었을 때 기존보다 5.625에 먼저 도달할 경우 0으로 초기화함
    if (m_fAngle > 360.f)
        m_fAngle = m_fAngle - 360.f;

    *_iDirection = int(m_fAngle / UNIT_MIN_DIR); // 11.25f = 360.f * 0.032f
}
```

■ Comment

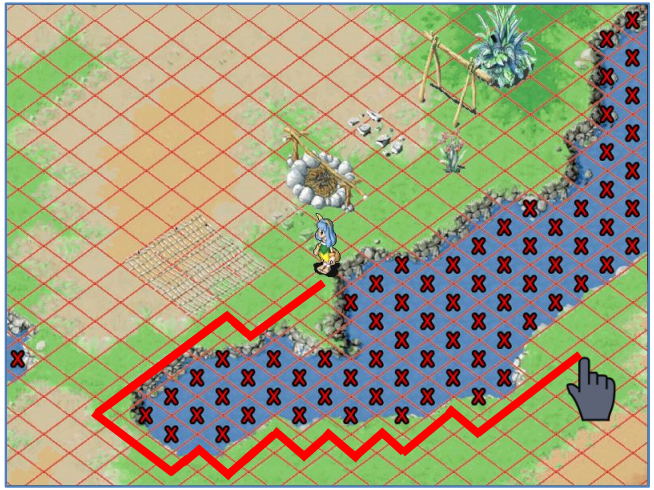
메인프레임의 Update에서 매회 유닛의 각도를
구하고 각도에 따른 이미지 변경후 이동

■ 2. A* Algorithm

Step 1

1. 마우스 클릭시 플레이어 이동 이벤트 발생

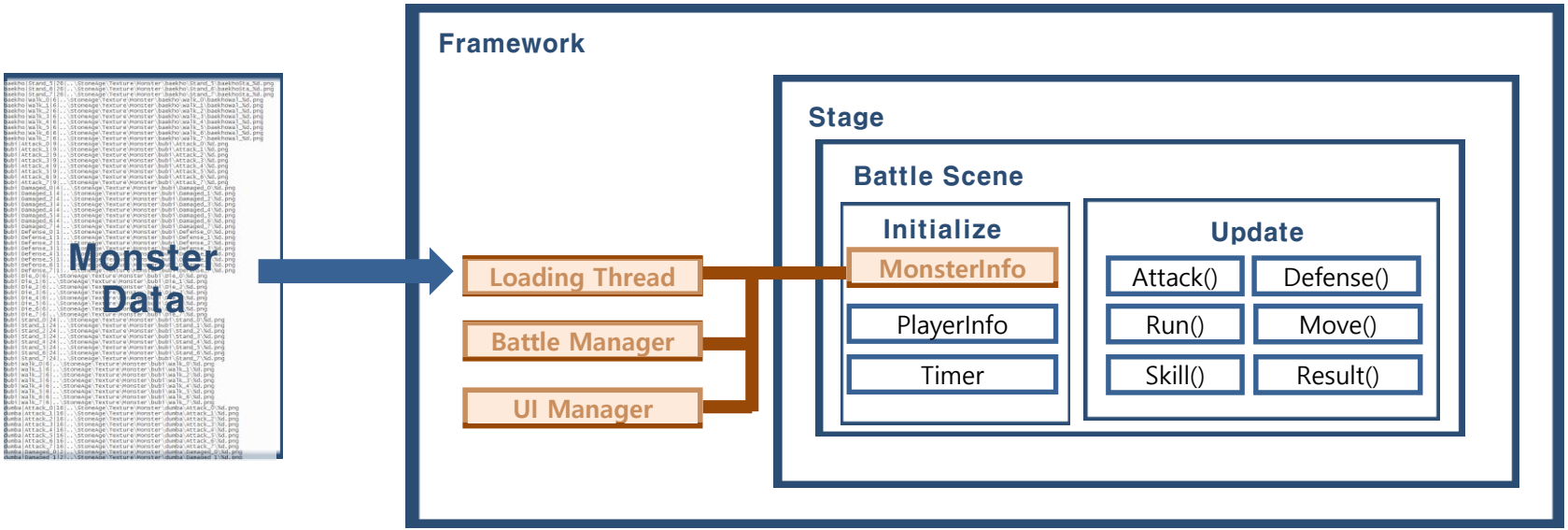
Step 2



2. Vector에 담긴 타일 인덱스를 검색하여 이동 가능
한 인덱스를 List에 저장 후 최적의 인덱스를 따라 이동

STL의 Vector,List

■ 3. Battle System



■ 3. Battle System

■ Result

■ Code

```
int CBattleScene::Update(void)
{
    if (m_pMonster->GetInfo()->stateNum == STATE_DEFENSE)
    {
        Defense();
    }
    if (m_pMonster->GetInfo()->stateNum != 0)
    {
        //ist<CObj*>::iterator iter_begin = CUIManager::GetInstance()->GetObjList()[BATTLE_UI].begin();
        //ist<CObj*>::iterator iter_end = CUIManager::GetInstance()->GetObjList()[BATTLE_UI].end();
        for (; iter_begin != iter_end; ++iter_begin)
        {
            //rtti써서 삭제
            if (typeid(*(*iter_begin)) == typeid(CBattleUI))
                (*iter_begin)->SetDead();
        }
    }
    switch (m_iTurn)
    {
    case STATE_SKILL:
        Skill();
        break;
    case STATE_ATT:
        Attack();
        break;
    }
    //ist<CObj*>::iterator iter_begin = CBattleManager::GetInstance()->GetObjList()[BATTLE_MONSTER].begin();
    //ist<CObj*>::iterator iter_end = CBattleManager::GetInstance()->GetObjList()[BATTLE_MONSTER].end();
    for (; iter_begin != iter_end; ++iter_begin)
    {
        if (iter_begin->m_bAttack == false)
            SelectMonster();
    }
    if (m_bDead == true)
    {
        return STATE_DEAD;
    }
                    코드 중략
}
```

■ Comment

턴제 전투방식으로 공격,방어,스킬,도망 등이 있다

플레이어가 특정 타일에 도착했을때 일정한 확률로

Battle Scene으로 넘어가며 제한시간 60초가 초과

되거나 행동을 취하면 몬스터의 턴으로 넘어간다

Game List

2D Game

메이플스토리

개발 기간 : 17.03.01~17.03.24 **(3주)**

개발 언어 : C / C++

개발 인원 : 1명

개발 환경 : Visual Studio / WIN_API

개발 중점 사항 :

싱글톤패턴, 추상팩토리패턴, 메디에이트패턴, 비트맵이미지 처리, Line충돌, CScene전환, 몬스터패턴, Rect충돌, 인벤토리 구현

■ 1. Line Collision

메이플스토리

■ Code

```
void CPlayer::LineCollision( void )
{
    LINE*   pLine = NULL;

    for(list<LINE*>::iterator iter = m_pLineList->begin(); iter != m_pLineList->end(); ++iter)
    {
        if ((*iter)->tLPoint.fX < m_tInfo.fX && m_tInfo.fX < (*iter)->tRPoint.fX &&
            (*iter)->tLPoint.fY < m_tInfo.fY + m_tInfo.fY / 2 &&
            m_tInfo.fY >(*iter)->tRPoint.fY - m_tInfo.fY / 2)
        {
            pLine = (*iter);
            break;
        }
    }
    float fWidth = pLine->tRPoint.fX - pLine->tLPoint.fX;
    float fHeight = pLine->tRPoint.fY - pLine->tLPoint.fY;
    float fGradient = fHeight / fWidth;
    float fY = fGradient * (m_tInfo.fX - pLine->tLPoint.fX) + pLine->tLPoint.fY;
    if(m_bJump != true&& m_bUp==false)
    {
        m_tInfo.fY = fY-m_tInfo.fCY/2 ;
    }
    else
    {
        if(m_tInfo.fY > fY)
        {
            m_bJump = false;
            m_fJumpAcc = 0.f;
            m_tInfo.fY = fY;
        }
    }
    if (pLine != NULL)
    {
        m_bUp = true;
        return;
    }
}
```

■ Result

■ Comment

- 2D게임에서 Line충돌을 통해 층을 구분함

■ 2. Alpha Blending

메이플스토리

■ Code

```
void CPlayer::AlphaBlending(HDC _dc)
{
    CMyBitmap* pBitmap = CBitmapMgr::GetInstance()->FindImage(m_pName);
    if(pBitmap == NULL) return;
                        코드 출력
    HDC dcTemp = CreateCompatibleDC(_dc);
    HBITMAP hBitmap = CreateCompatibleBitmap(_dc, int(m_tInfo.fCX), int(m_tInfo.fCY));
    HBITMAP hOld = (HBITMAP)SelectObject(dcTemp, hBitmap);

    //원본의 시작 위치는 플레이어 위치를 중점으로 사이즈와 스크롤 값을 더해서 계산된다.
    int iOriStartX = int(m_tInfo.vPos.x - m_tInfo.fCX / 2) + m_ptScroll.x+(int)g_fScrollX;
    int iOriStartY = int(m_tInfo.vPos.y - m_tInfo.fCY / 2) + m_ptScroll.y+(int)g_fScrollY;

    //대상의 시작 위치를 뒤로 당겨야 함.    //원본 시작 위치가 0보다 작으면 0으로 고정시켜야 함.
    if (iOriStartX < 0)
    {
        iDestStartX = -iOriStartX;
        iOriStartX = 0;
    }
    else if(iOriStartX > 800 - m_tInfo.fCX)
    {
        //대상의 끝 위치를 이미지가 벗어난 만큼으로 사이즈에서 빼줘야 한다.
        iDestEndX = int(m_tInfo.fCX + ((800 - m_tInfo.fCX) - iOriStartX));
    }
    //백버퍼를 잘라서 임시 dc에다가 복사를 먼저 함.
    TransparentBlt(dcTemp, iDestStartX, iDestStartY,iDestEndX, iDestEndY,
        CBitmapMgr::GetInstance()->FindImage(L"BackBuffer")->GetMemDC(),
        iOriStartX,iOriStartY,iDestEndX,iDestEndY,RGB(255, 0, 255));
    //플레이어 제외한 배경들이 플레이어 사이즈의 dc에 먼저 그려짐.
    TransparentBlt(dcTemp,  int(m_tInfo.vPos.x - m_tInfo.fCX) + (int)g_fScrollX,int(m_tInfo.vPos.y - m_tInfo.fCY + 2),
        int(m_tInfo.fCX) + 2,int(m_tInfo.fCY) + 4,pBitMap->GetMemDC(),int(m_tInfo.fCX + m_tFrame.iFrameStart + 2),
        int(m_tInfo.fCY + m_tFrame.iScene + 4),int(m_tInfo.fCX) + 2,int(m_tInfo.fCY) + 4,RGB(255, 0, 255));
    AlphaBlend(_dc,int(m_tInfo.vPos.x - m_tInfo.fCX) + (int)g_fScrollX,int(m_tInfo.vPos.y - m_tInfo.fCY + 2)
        ,int(m_tInfo.fCX) + 2,int(m_tInfo.fCY) + 4,dcTemp,0,0,int(m_tInfo.fCX) + 2,int(m_tInfo.fCY) + 4,tFunction);
    hBitmap = (HBITMAP)SelectObject(dcTemp, hOld);
    DeleteObject(hBitmap);
    DeleteDC(dcTemp);
}
```

■ Result

■ Comment

- 알파블랜딩의 원리를 이해하고 실제 게임에 적용시킴

■ 3. Item Inventory

메이플스토리

■ Code

```
void CInventory::CheckInven(void)
{
    list<CObj*>::iterator iter_begin = CObjMgr::GetInstance()->GetObjList()[OBJ_PLAYER].begin();


    if (GetAsyncKeyState(VK_LBUTTON))
    {
        if (PtInRect(&rect, ptMouse))
        {
            CSoundMgr::GetInstacne()->PlayEffectSound(L"click.MP3");
            for (int i = 0; i < MAX_INVEN; ++i)
            {
                ppInven = ((CPlayer*)m_pPlayer)->GetInven();

                while (*ppInven != NULL)
                {
                    if ((*ppInven)->GetItem()->strname == m_pItem[i]->GetItem()->strname)
                    {
                        // 아이템 중복일때 처리
                        ++(*ppInven)->GetItem()->iCount;
                        (*iter_begin)->GetInfo()->iMoney -= m_pItem[i]->GetItem()->iPrice;
                        return;
                    }

                    ++ppInven;
                }
                코드 중략
                return;
            }
        }
    }
}
```

■ Result

■ Comment

- 상점에서 아이템 구매가 가능하고
  인벤토리에 있는 아이템이 장착됨

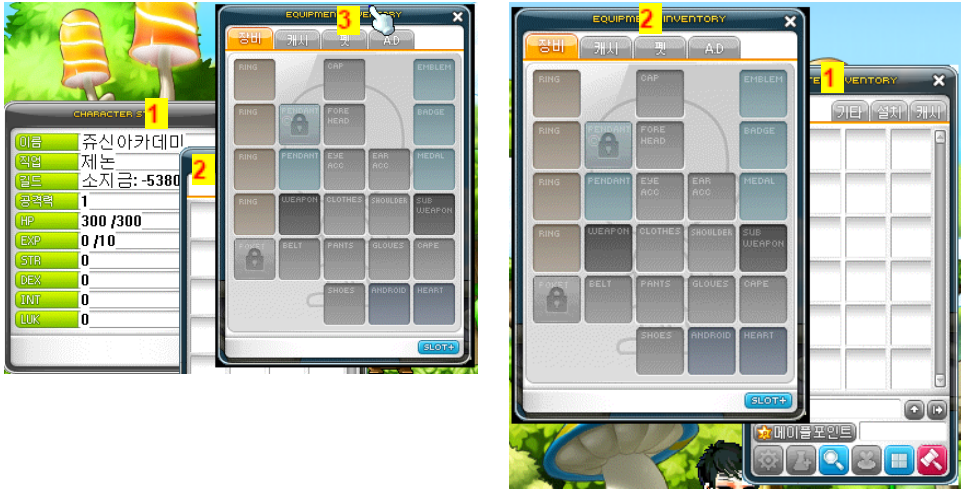
■ 4. Render (Y sorting)

메이플스토리

■ Code

```
void CRenderMgr::Render( HDC _hdc )
{
    // 객체들을 Y소팅으로 정렬한다.
    sort(m_vecRenderObj[RENDER_WORLDOBJ].begin(),
        m_vecRenderObj[RENDER_WORLDOBJ].end(),
        CRenderMgr::Compare );

    for(int i = 0; i < RENDER_end; ++i)
    {
        int iCount = m_vecRenderObj[i].size();

        for(int j = 0; j < iCount; ++j)
        {
            // 정렬된 객체를 출력 한다.
            m_vecRenderObj[i][j]->Render(_hdc);
        }
        m_vecRenderObj[i].clear();
    }
}

bool CRenderMgr::Compare( CObj* pDest, CObj* pSour )
{
    return (pDest->GetInfo()->fY + pDest->GetInfo()->fCY
        <pSour->GetInfo()->fY + pSour->GetInfo()->fCY);
}
```

■ Result



■ Comment

- 원하는 UI를 마우스 클릭시 Y소팅하여
  가장 먼저 볼수있게 됨

Refernce Book

- DirectX 11를 이용한 3D 게임 프로그래밍 입문 (한빛미디어 / 프랭크 D. 루나 지음)
- Effective C++ (PTG / 스콧 마이어스 지음)
- C++11 STL 프로그래밍 (한빛미디어 / 최흥배 지음)

Refernce Link

Fmod 참고링크
https://www.fmod.com/resources/documentation-api?page=content/generated/lowlevel_api_interfaces.html#/

다익스트라 알고리즘 참고링크
http://thrillfighter.tistory.com/235?category=399367

렌더링 파이프라인 참고링크
http://dlgnlfus.tistory.com/135

Blending 참고링크
https://m.blog.naver.com/PostView.nhn?blogId=khk6435&logNo=50185664186&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

Game Movie Link

1) 언리얼토너먼트2004 (DirectX11, HLSL, MFC, C++)

2) 블레이드 엔 소울 (C++, DirectX3D, MFC)
https://www.youtube.com/watch?v=NFvZv3yfaig&feature=youtu.be

3) 오버워치 (TCP/IP, DirectX3D, MFC, C++)
https://www.youtube.com/watch?v=Abtu_jMtIJI&feature=youtu.be

4) 스톤에이지 (C++, DirectX2D, MFC)
https://www.youtube.com/watch?v=hdrn_kycRsY

5) 메이플스토리 (C++,API)
https://www.youtube.com/watch?v=XU_GEw_vmxs&feature=youtu.be

※기타 작품

베트맨 아캄시티 (DirectX11, HLSL, MFC, C++)
https://www.youtube.com/watch?v=BASgtN9sVHk&feature=youtu.be

오버워치 (Unity, pc, mobile, C#)
https://www.youtube.com/watch?v=upW0U58b8VY

런닝 큐브 (C++,OpenGL)
https://www.youtube.com/watch?v=AqIr995VafM

식물 vs 좀비 (Python, Pico2d)
https://www.youtube.com/watch?v=pN-WZPFfpwE

마법사의하루 (C#, Unity, VR)
https://www.youtube.com/watch?v=e5Kng51ufy8&t=1s

드래곤볼서버 (C++, API, TCP/IP)
https://www.youtube.com/watch?v=Tenbp6ckip4&t=75s

Thank You

오준석

연락처: 010-7206-0506
E-Mail: topojs8@naver.com