Project : Kirby Discovery (3D 팀 프로젝트 최성희) _ 검광효과

코드 목적 :

검이 이동한 궤적에 따라 잔상을 보여주기 위하여 구현했습니다.

코드 설명 :

1. Effect_Trail 클래스는 검광 효과를 사용하고자 하는 객체가 소유하도록 구현했습니다.
2. 해당 객체는 Effect_Trail 클래스의 Add_TrailPoint() 함수를 호출하여 월드 스페이스 상의 두 점 위치를 넘겨줍니다.
3. 두 점을 넘겨받았을 때, 궤적을 보관하고 있는 컨테이너가 비어있으면 넘겨받은 좌표로 컨테이너를 채워줍니다.
4. 컨테이너가 비어있지 않으면 컨테이너 가장 앞쪽에 추가하고, 가장 뒤에 있는 좌표를 삭제해줍니다.
5. XMVectorCatmullRom() 보간 함수를 사용하여 점의 위치를 다시 갱신하였습니다. (부드러운 곡선을 만들어주기 위하여 사용했습니다.)
6. Trail_Update() 함수에서는 CVIBuffer_Trail 클래스의 Update()를 호출하여 갱신될 정점의 위치 정보를 넘겨줍니다.
7. 넘겨받은 월드 상의 위치정보로 버퍼의 로컬위치를 변환시켜 그려지게 했습니다.

```
void CEffect_Trail::Add_TrailPoint(_float4 fTop, _float4 fBottom)
{
    if (m_pTrailTopPoint.size() == 0)
    {
        for (_uint i = 0; i < m_iNumInstance + 1; ++i)
        {
            m_pTrailTopPoint.push_back(fTop);
            m_pTrailBottomPoint.push_back(fBottom);
        }
    }
    else
    {
        auto& iterbegin = m_pTrailTopPoint.begin();
        m_pTrailTopPoint.insert(iterbegin, fTop);
        auto& iterend = m_pTrailTopPoint.end();
        m_pTrailTopPoint.erase(--iterend);


        auto& iterbegin1 = m_pTrailBottomPoint.begin();
        m_pTrailBottomPoint.insert(iterbegin1, fBottom);
        auto& iterend1 = m_pTrailBottomPoint.end();
        m_pTrailBottomPoint.erase(--iterend1);

        CatmullRom();
    }
    CreateTrail();
    Trail_Update();
}
```

```
void CEffect_Trail::CatmullRom()
{
    _uint i = 0;
    while (true)
    {
        _vector vTrailTop = XMVectorCatmullRom(XMLoadFloat4(&m_pTrailTopPoint[i]),
            XMLoadFloat4(&m_pTrailTopPoint[i + 1]),XMLoadFloat4(&m_pTrailTopPoint[i + 2]),
            XMLoadFloat4(&m_pTrailTopPoint[i + 3]), 0.1f);

        XMStoreFloat4(&m_pTrailTopPoint[i + 1], vTrailTop);


        _vector vTrailBottom = XMVectorCatmullRom(XMLoadFloat4(&m_pTrailBottomPoint[i]),
            XMLoadFloat4(&m_pTrailBottomPoint[i + 1]),XMLoadFloat4(&m_pTrailBottomPoint[i + 2]),
            XMLoadFloat4(&m_pTrailBottomPoint[i + 3]), 0.1f);

        XMStoreFloat4(&m_pTrailBottomPoint[i + 1], vTrailBottom);

        ++i;
        if (i + 3 == m_iNumInstance + 1)
            break;
    }
}
```

Project : Kirby Discovery (3D 팀 프로젝트 최성희) _ SSD

코드 목적 :

두 개의 폴리곤이 같은 깊이 값을 가지고 있으면 z-fighting이 일어나 깜박거리는 현상이 발생하는데 이 현상을 해결하고, 오브젝트 굴곡에 따라 원하는 텍스쳐를 붙이고자 구현했습니다.

코드 설명 :

1. 랜더링 파이프라인의 Deferred Rendering 시스템에서 오브젝트의 깊이 값이 기록되어있는 깊이 랜더타겟과 법선벡터가 기록되어있는 노말 랜더타겟을 이용합니다.

2. 따라서, 모든 오브젝트가 전부 그려진 후 마지막으로 그려지도록 했습니다. (SSD는 Cube 버퍼를 사용하여 그립니다)

3. 붙여지는 오브젝트의 법선벡터와 붙이고자 하는 오브젝트의 방향을 내적 하여 특정 각도 이상이면 그려지지 않도록 했습니다.

4. 깊이 랜더타겟을 이용하여 상자(Cube 버퍼) 밖에 위치한 경우 해당 픽셀은 그리지 않도록 했습니다.

5. 데칼 박스의 버퍼가 -0.5 ~ 0.5 사이이므로 0.5를 더해줘서 UV 좌표를 만들고 해당 텍스쳐를 그려주었습니다.

6. 블러 마스크를 사용하여 SSD에도 블러가 적용되도록 했습니다.

```
PS_OUT_BLUR PS_SSD_Blur(PS_IN_DECAL In)
{
    PS_OUT_BLUR     Out = (PS_OUT_BLUR)0;

    /* Decal Box Renderering */
    float2 vTexUV;
    vTexUV.x = (In.vProjPos.x / In.vProjPos.w) * 0.5f + 0.5f;
    vTexUV.y = (In.vProjPos.y / In.vProjPos.w) * -0.5f + 0.5f;
    vector vDepthDesc = g_DepthTexture.Sample(DefaultSampler, vTexUV);
    vector vNormalDesc = g_NormalTexture.Sample(DefaultSampler, vTexUV);

    clip(dot(vNormalDesc, In.vDecalDir) - cos(radians(60.f)));

    if (vDepthDesc.z == 1.f)
        discard;

    float fViewZ = vDepthDesc.x * g_fFar;
    vector vDecalLocalPos;
    vDecalLocalPos.x = (vTexUV.x * 2.f - 1.f) * fViewZ;
    vDecalLocalPos.y = (vTexUV.y * -2.f + 1.f) * fViewZ;
    vDecalLocalPos.z = vDepthDesc.y * fViewZ;
    vDecalLocalPos.w = 1.f * fViewZ;

    /* Object in Decal Local Space */
    vDecalLocalPos = mul(vDecalLocalPos, g_ProjMatrixInv);
    vDecalLocalPos = mul(vDecalLocalPos, g_ViewMatrixInv);
    vDecalLocalPos = mul(vDecalLocalPos, g_WorldMatrixInv);

    clip(0.5 - abs(vDecalLocalPos.xyz));

    float2 decalUV;
    decalUV.x = vDecalLocalPos.x + 0.5f;
    decalUV.y = vDecalLocalPos.z + 0.5f;
    vector vDecalDesc = g_DecalTexture.Sample(DefaultSampler, decalUV);

    vDecalDesc = vector(vDecalDesc.rgb * g_vColorMul + g_vColorAdd, vDecalDesc.r * g_vColorAlpha);
    vDecalDesc.a = saturate(vDecalDesc.a);

    if(vDecalDesc.a <= 0.f)
        discard;

    Out.vColor = vDecalDesc;
    Out.vBlurMask = vector(1.f, g_fBlurPower, 1.f, 1.f);

    return Out;
}
```

Project : Kirby Discovery (3D 팀 프로젝트 최성희) _ Fog

코드 목적 :

눈보라로 인한 안개효과를 표현하고자 했습니다. 또한, Stage에 따라 안개위치, 세기 등 다르게 주고자 했습니다.

코드 설명 :

1. 해당 코드는 HLSL프로그래밍 책을 참고했습니다.
2. 3D 개인프로젝트에서 구현했던 월드 스페이스 Y값 기준의 안개가 아닌 카메라 시선을 기준으로 합니다.
3. fogFactor 값을 이용하여 월드 스페이스상 Y값에 따라 색 변화를 자연스럽게 변화시켰습니다.

```
PS_OUT PS_MAIN_BLEND(PS_IN In)
{
    PS_OUT Out = (PS_OUT) 0;
    vector vDiffuseDesc = g_DiffuseTexture.Sample(DefaultSampler, In.vTexUV);
    vector vShadeDesc = g_ShadeTexture.Sample(WrapLinearSampler, In.vTexUV);
    vector vSpecularDesc = g_SpecularTexture.Sample(DefaultSampler, In.vTexUV);
    vector vDepthDesc = g_DepthTexture.Sample(DefaultSampler, In.vTexUV);
    vector vDirectLightDesc = g_DirectLightTexture.Sample(DefaultSampler, In.vTexUV);
    float fViewZ = vDepthDesc.x * g_fFar;
    vector vWorldPos;
    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;
    vWorldPos = mul(vWorldPos, g_ProjMatrixInv);
    vWorldPos = mul(vWorldPos, g_ViewMatrixInv);
    if (vDepthDesc.r != 1.f)
        vDiffuseDesc = pow(vDiffuseDesc, 2.2f);

    Out.vColor = vDiffuseDesc * vShadeDesc + vDirectLightDesc + vSpecularDesc;
    Out.vColor.a = g_DiffuseTexture.Sample(DefaultSampler, In.vTexUV).a;

    /* 안개 적용  */
    float3 eyeToPixel = float3(vWorldPos.xyz - g_vCamPosition.xyz);
    Out.vColor.xyz = ApplyFog(Out.vColor.xyz, g_vCamPosition.y, eyeToPixel);

    if (Out.vColor.a <= 0.f)
        discard;
    return Out;
}
```

```
/* fog */
float3 ApplyFog(float3 originalColor, float eyePosY, float3 eyeToPixel)
{
    /* 카메라와의 거리 Depth 값*/
    float pixelDist = length(eyeToPixel);
    /* 카메라에서부터 방향벡터 */
    float3 eyeToPixelNorm = eyeToPixel / pixelDist;

    /* 안개 시작 지점에서 해당공간이 얼마나 떨어져있는지를 나타냄 */
    float fogDist = max(pixelDist - FogStartDepth, 0.f);

    //안개 세기에 대해 거리 계산
    float fogHeightDensityAtViewer = exp(-FogHeightFalloff * eyePosY); /* 높이 소멸값 : 클수록 사라지는 높이가 낮아진다.*/
    float fogDistInt = fogDist * fogHeightDensityAtViewer;

    //안개 세기에 대한 높이 계산
    float eyeToPixelY = eyeToPixel.y * (fogDist / pixelDist);
    float t = FogHeightFalloff * eyeToPixelY;
    const float thresholdT = 0.01f;
    float fogHeightInt = abs(t) > thresholdT ? (1.f - exp(-t)) / t : 1.f;

    //위 계산값을 합해 최종인수 계산
    float fogFinalFactor = exp(-FogGlobalDensity * fogDistInt * fogHeightInt);

    //태양 하이라이트 계산 및 안개 색상
    float sunHighlightFactor = saturate(dot(eyeToPixelNorm, FogSunDir));

    sunHighlightFactor = pow(sunHighlightFactor, 8.0);
    float3 fogFinalColor = lerp(FogColor, FogHighlightColor, sunHighlightFactor);

    return lerp(fogFinalColor, originalColor, fogFinalFactor);
}
```