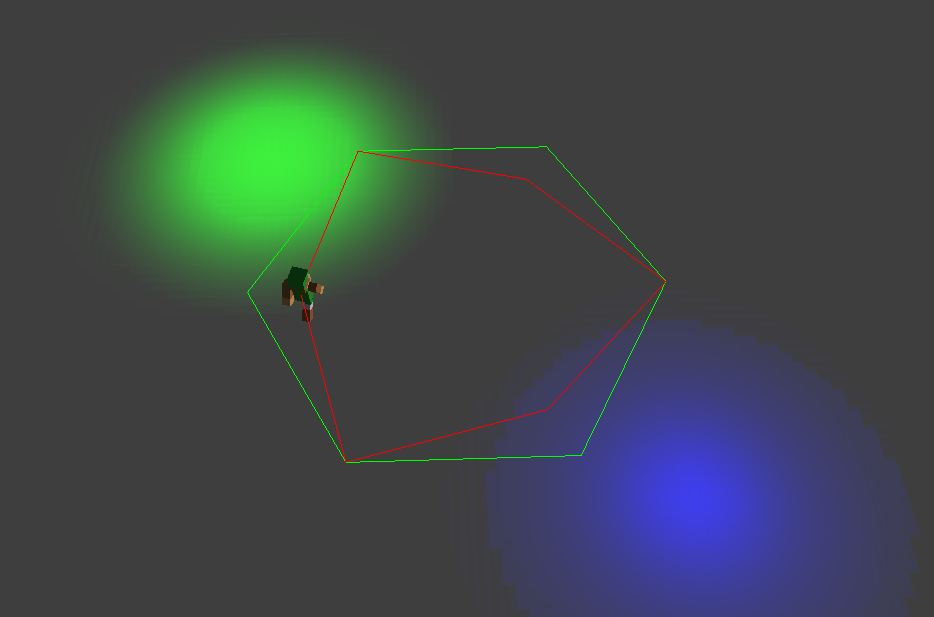
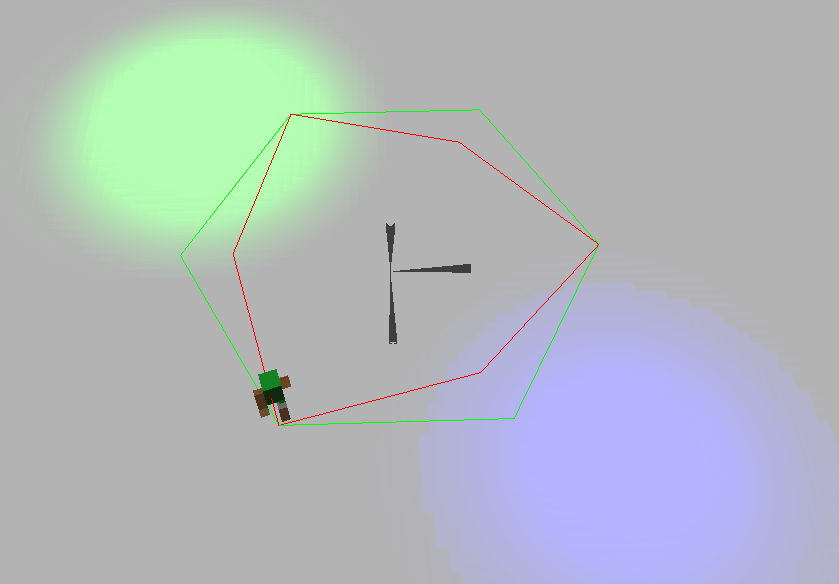
[ 게임 인공지능 프로그래밍 ]

**게임에 인공지능 프로그램 적용**

인하대학교 미래인재개발원 문화콘텐츠프로그래밍

김예슬

**인공지능을 적용한 프로그램 스크린샷**



**테스트를 통해 완성된 인공지능 프로그램 코드**

#include "stdafx.h"

#include "cPath.h"

cPath::cPath()

{

}

cPath::~cPath()

{

}

void cPath::SetUp(int spot)

{

float radian = ShapeDegreeToRadian(spot);

ST\_PC\_VERTEX addSpot;

addSpot.c = D3DXCOLOR(0.0f, 255.0f, 0.0f, 1.0f);

addSpot.p = D3DXVECTOR3(0.0f, 1.0f, 5.0f);

m\_vecVertex.push\_back(addSpot);

for (int i = 0; i < spot; i++)

{

D3DXMATRIXA16 matR;

D3DXMatrixRotationY(&matR, radian);

D3DXVec3TransformCoord(&addSpot.p, &addSpot.p, &matR);

m\_vecVertex.push\_back(addSpot);

}

}

void cPath::Render()

{

g\_pD3DDevice->SetRenderState(D3DRS\_LIGHTING, false);

D3DXMATRIXA16 matWorld;

D3DXMatrixIdentity(&matWorld);

g\_pD3DDevice->SetTransform(D3DTS\_WORLD, &matWorld);

g\_pD3DDevice->SetFVF(ST\_PC\_VERTEX::FVF);

g\_pD3DDevice->DrawPrimitiveUP(D3DPT\_LINESTRIP, m\_vecVertex.size() - 1, &m\_vecVertex[0], sizeof(ST\_PC\_VERTEX));

}

float cPath::ShapeDegreeToRadian(int spot)

{

float degree = 180 - (180 \* (spot - 2) / spot);

// ※ : 정n각형의 내각 공식 => 180 \* (spot - 2) / spot

return degree \* D3DX\_PI / 180;

}

vector<ST\_PC\_VERTEX> cPath::GetVertex()

{

return m\_vecVertex;

}

// ==============================================================

void cCharacter::Update\_Sub(vector<ST\_PC\_VERTEX> path, int &destNum)

{

D3DXMATRIXA16 matR, matT;

D3DXMatrixIdentity(&matR);

if (sqrt(pow(path[destNum].p.x - m\_vPosition.x, 2) + pow(path[destNum].p.z - m\_vPosition.z, 2)) < 0.1f)

{

destNum = destNum + 1 < path.size() ? destNum+=1 : 0;

D3DXVECTOR3 vUp(0, 1, 0);

D3DXMatrixLookAtLH(&matR, &m\_vPosition, &path[destNum].p, &vUp);

D3DXMatrixTranspose(&matR, &matR);

D3DXVECTOR3 direction = D3DXVECTOR3(0, 0, 1);

D3DXVec3TransformNormal(&direction, &direction, &matR);

D3DXVec3Normalize(&direction, &direction);

m\_fRotY += abs(acosf(D3DXVec3Dot(&direction, &m\_vDirection)));

m\_vDirection = direction;

}

else

{

m\_vPosition += m\_vDirection \* dSpeed \* 0.5f \* dTimer->DeltaTime();

}

D3DXMatrixTranslation(&matT, m\_vPosition.x, m\_vPosition.y, m\_vPosition.z);

D3DXMatrixRotationY(&matR, m\_fRotY);

m\_matWorld = matR \* matT;

}