[ 게임 플랫폼 응용프로그래밍 ]

**PC 환경에서 작동 가능한 게임프로그램 제작**

인하대학교 미래인재개발원 문화콘텐츠프로그래밍

김예슬

**PC환경에서 작동 가능한 게임 프로그램 제작**

1) 게임 개발 문서

- 입력은 키보드와 마우스로 이루어진다

- 마우스 처리

마우스 움직임에 따라 카메라가 이동될 수 있어야 함

- 키보드 처리

W, A, S, D : 플레이어 이동

Ctrl : UI 출력

**2) 게임 프로그램 코드**

void CCharacter::ReceiveEvent(ST\_EVENT eventMsg)

{

float speed = 0.005f;

float rotation = -1.0f;

if (!g\_gameManager->GetUImode())

{

// todo : 상태에 따른 애니메이션 출력

if (eventMsg.message == WM\_LBUTTONDOWN)

{

ST\_EVENT msg;

msg.eventType = EventType::eColorChangeEvent;

msg.ptrMessage = &m\_color;

g\_pEventManager->CheckEvent(msg);

return;

}

if (eventMsg.eventType == EventType::eChangedColorEvent)

{

eventMsg.playerInput = PlayerInputType::eUp;

}

switch (eventMsg.playerInput)

{

case PlayerInputType::eUp:

rotation = 0.0f;

break;

case PlayerInputType::eLeftUp:

rotation = D3DX\_PI / 4.0f \* -1;

break;

case PlayerInputType::eRightUp:

rotation = D3DX\_PI / 4.0f;

break;

case PlayerInputType::eDown:

rotation = D3DX\_PI;

break;

case PlayerInputType::eLeftDown:

rotation = D3DX\_PI + D3DX\_PI / 4.0f;

break;

case PlayerInputType::eRightDown:

rotation = (D3DX\_PI + D3DX\_PI / 4.0f) \* -1;

break;

case PlayerInputType::eLeft:

rotation = -D3DX\_PI / 2.0f;

break;

case PlayerInputType::eRight:

rotation = D3DX\_PI / 2.0f;

break;

case PlayerInputType::eHold:

// todo : 잡기 구현

break;

default:

speed = -1.0f;

break;

}

if (speed > 0)

{

DoRotation(rotation);

DoMove(speed);

}

}

D3DXMATRIXA16 matT;

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

m\_matWorld = m\_matRotY \* matT;

}

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

void CCamera::ReceiveEvent(ST\_EVENT eventMsg)

{

if (eventMsg.eventType == EventType::eInputEvent)

{

switch (eventMsg.message)

{

case WM\_LBUTTONDOWN:

m\_preMousePos.x = LOWORD(eventMsg.lParam);

m\_preMousePos.y = HIWORD(eventMsg.lParam);

m\_isLBtnDown = true;

break;

case WM\_LBUTTONUP:

m\_isLBtnDown = false;

break;

case WM\_MOUSEMOVE:

{

if (g\_gameManager->GetUImode())

{

POINT ptCurMouse;

ptCurMouse.x = LOWORD(eventMsg.lParam);

ptCurMouse.y = HIWORD(eventMsg.lParam);

m\_preMousePos = ptCurMouse;

return;

}

POINT ptCurMouse;

ptCurMouse.x = LOWORD(eventMsg.lParam);

ptCurMouse.y = HIWORD(eventMsg.lParam);

float fDeltaX = (float)ptCurMouse.x - m\_preMousePos.x;

float fDeltaY = (float)ptCurMouse.y - m\_preMousePos.y;

/// 좌우 밖으로 계속 카메라 이동 처리 중

RECT rc;

GetClientRect(g\_hWnd, &rc);

if (ptCurMouse.x < rc.left + 1.0f)

{

cout << ptCurMouse.x << ' ' << ptCurMouse.y << endl;

fDeltaX -= 15.0f;

}

else if (ptCurMouse.x > rc.right - 2.0f)

{

cout << ptCurMouse.x << ' ' << ptCurMouse.y << endl;

fDeltaX += 15.0f;

}

else if (ptCurMouse.y < rc.top + 1.0f)

{

cout << ptCurMouse.x << ' ' << ptCurMouse.y << endl;

fDeltaY += 15.0f;

}

else if (ptCurMouse.y > rc.bottom - 2.0f)

{

cout << ptCurMouse.x << ' ' << ptCurMouse.y << endl;

fDeltaY -= 15.0f;

}

else

{

}

m\_vCamRotAngle.y += (fDeltaX / 150.0f);

m\_vCamRotAngle.x += (fDeltaY / 150.0f);

if (m\_vCamRotAngle.x < - D3DX\_PI / 3.0f - 0.0001f)

m\_vCamRotAngle.x = -D3DX\_PI / 3.0f;

if (m\_vCamRotAngle.x > D3DX\_PI / 2.0f - 0.0001f)

m\_vCamRotAngle.x = D3DX\_PI / 2.0f - 0.0001f;

m\_preMousePos = ptCurMouse;

}

break;

#if \_DEBUG

// DEBUG Mode

case WM\_MOUSEWHEEL:

{

m\_fCameraDistance -= (GET\_WHEEL\_DELTA\_WPARAM(eventMsg.wParam) / 30.0f);

if (m\_fCameraDistance < 5.0 + 0.0001f)

m\_fCameraDistance = 5.0f;

}

break;

#else

// RELEASE Mode

#endif

} // << : switch

} // << : if

}