[ 게임 클라이언트 응용프로그래밍 ]

**게임 캐릭터 프로그래밍**

인하대학교 미래인재개발원 문화콘텐츠프로그래밍

김예슬

**사용자 입출력에 대한 캐릭터 조작 및 관리 소스 코드**

if (playerState != eFocus)

{

if (!isJump && (GetAsyncKeyState(VK\_DOWN) & 0x8000 || GetAsyncKeyState(VK\_DOWN) & 0x8001))

{

moveDirection = eMoveDown;

playerState = eFall;

MovePlayer(playerPos, moveDirection, eMoveSpeed, 1, 0);

int diffNum = 0;

if (CheckBtmGround(diffNum))

MovePlayer(playerPos, moveDirection, diffNum, 1, dCorrection);

else

MovePlayer(playerPos, moveDirection, diffNum, -1, dCorrection);

CheckOut(playerPos, moveDirection);

}

if (GetAsyncKeyState(VK\_RIGHT) & 0x8000 || GetAsyncKeyState(VK\_RIGHT) & 0x8001)

{

moveDirection = eMoveRight;

// playerState = eMoveRight;

isRightSight = true;

MovePlayer(playerPos, moveDirection, eMoveSpeed, 1, 0);

int diffNum = 0;

if (CollisionMap(eMoveRight, diffNum))

MovePlayer(playerPos, moveDirection, diffNum, -1, -dCorrection);

else

MovePlayer(playerPos, moveDirection, diffNum, 1, -dCorrection);

CheckOut(playerPos, moveDirection);

}

if (GetAsyncKeyState(VK\_LEFT) & 0x8000 || GetAsyncKeyState(VK\_LEFT) & 0x8001)

{

moveDirection = eMoveLeft;

// playerState = eMoveLeft;

isRightSight = false;

MovePlayer(playerPos, moveDirection, eMoveSpeed, -1, 0);

int diffNum = 0;

if (CollisionMap(eMoveLeft, diffNum))

MovePlayer(playerPos, moveDirection, diffNum, -1, 0);

else

MovePlayer(playerPos, moveDirection, diffNum, 1, 0);

CheckOut(playerPos, moveDirection);

}

if (!isJump && (GetAsyncKeyState(VK\_SPACE) & 0x8000 || GetAsyncKeyState(VK\_SPACE) & 0x8001) // || GetAsyncKeyState(VK\_UP) & 0x8000) && GetKeyState(0x41) >= 0)

{

isJump = true;

playerState = eJump;

jumpPower = eJumpPower;

}

else if (isJump && playerState == eJump)

{

jumpPower -= eGravity \* dTimeSec;

POINT checkRect[4];

checkRect[0] = playerPos[0];

checkRect[1] = playerPos[1];

checkRect[2] = playerPos[2];

checkRect[3] = playerPos[3];

int diffNum = 0;

int underLineNum = 0;

for (int i = 1; i < jumpPower; i++)

{

MovePlayer(checkRect, playerState, 1, -1, 0);

if (CollisionMap(playerState, diffNum))

{

underLineNum++;

if (isGetItem == true)

{

dMap->PlayerGetItem();

isGetItem = false;

break;

}

}

else

underLineNum--;

CheckOut(playerPos, eMoveUp);

}

if (underLineNum > 0)

{

MovePlayer(playerPos, playerState, underLineNum + 8, -1, 0);

}

else

{

playerState = eFall;

MovePlayer(playerPos, playerState, 0, 1, 1);

}

}

else

{

if (GetKeyState(VK\_SPACE) < 0) // || (GetKeyState(VK\_UP) < 0)))

isJump = true;

else if (isBtmGround)

{

isJump = false;

jumpPower = eJumpPower;

}

}

if (((GetAsyncKeyState(0x41) & 0x8000 || GetAsyncKeyState(0x41) & 0x8001)) && !isCharging)

{

moveSpeed = 0;

jumpPower = 0;

gravity = 0;

playerState = eFocus;

CalcCenterPos();

lastMoveCenter.x = centerPos.x;

lastMoveCenter.y = centerPos.y;

lastPlayerPos[0] = playerPos[0];

lastPlayerPos[1] = playerPos[1];

lastPlayerPos[2] = playerPos[2];

lastPlayerPos[3] = playerPos[3];

SetPos(fMovePos, centerPos.x, centerPos.y, ePlayerSize);

dSoundSys->PlayFocusSound();

}

else

{

moveSpeed = eMoveSpeed;

gravity = eGravity;

if (focusGauge < maxFocusGauge)

focusGauge += dAddFocusP;

if (focusGauge <= eSmallFocus || GetKeyState(0x41) < 0) isCharging = true;

else

isCharging = false;

CalcCenterPos();

SetPos(focusPos, centerPos.x, centerPos.y, focusGauge);

fCenterPos.x = centerPos.x;

fCenterPos.y = centerPos.y;

}

}

else

{

if (((GetAsyncKeyState(0x41) & 0x8000) || (GetAsyncKeyState(0x61) & 0x8000)))

{

if (focusGauge > eSmallFocus)

{

focusGauge -= dMinusFocusP;

SetPos(focusPos, centerPos.x, centerPos.y, focusGauge);

}

else

{

isCharging = true;

playerState = eIdle;

}

}

else

{

CalcFCenterPos();

SetPos(playerPos, fCenterPos.x, fCenterPos.y, ePlayerSize);

CheckBlockMap();

playerState = eIdle;

}

RECT area;

RECT rcMovepos = ConversionRect(fMovePos);

RECT rcFPos = ConversionRect(focusPos);

POINT checkCenter;

checkCenter.x = (fMovePos[0].x + fMovePos[2].x) / 2;

checkCenter.y = (fMovePos[0].y + fMovePos[2].y) / 2;

if ( (GetKeyState(VK\_UP) >= 0 && GetKeyState(VK\_DOWN) >= 0 && GetKeyState(VK\_LEFT) >= 0 && GetKeyState(VK\_RIGHT) >= 0)

|| !IntersectRect(&area, &rcMovepos, &rcFPos) )

{

SetPos(fMovePos, centerPos.x, centerPos.y, ePlayerSize);

fCenterPos.x = centerPos.x;

fCenterPos.y = centerPos.y;

}

if (GetAsyncKeyState(VK\_UP) & 0x8000)

{

moveDirection = eMoveUp;

int underLineNum = CheckFocusRange(moveDirection, -1);

if (underLineNum > 0)

MovePlayer(fMovePos, moveDirection, underLineNum, -1, 0);

else

MovePlayer(fMovePos, moveDirection, 0, 1, 1);

CheckOut(fMovePos, moveDirection);

CalcFCenterPos();

}

if (GetAsyncKeyState(VK\_DOWN) & 0x8000)

{

moveDirection = eMoveDown;

int underLineNum = CheckFocusRange(moveDirection, 1);

if (underLineNum > 0)

MovePlayer(fMovePos, moveDirection, underLineNum, 1, 0);

else

MovePlayer(fMovePos, moveDirection, 0, 1, -1);

CheckOut(fMovePos, moveDirection);

CalcFCenterPos();

}

if (GetAsyncKeyState(VK\_LEFT) & 0x8000)

{

moveDirection = eMoveLeft;

isRightSight = false;

int underLineNum = CheckFocusRange(moveDirection, -1);

if (underLineNum > 0)

MovePlayer(fMovePos, moveDirection, underLineNum, -1, 0);

else

MovePlayer(fMovePos, moveDirection, 0, 1, 1);

CheckOut(fMovePos, moveDirection);

CalcFCenterPos();

}

if (GetAsyncKeyState(VK\_RIGHT) & 0x8000)

{

moveDirection = eMoveRight;

isRightSight = true;

int underLineNum = CheckFocusRange(moveDirection, 1);

if (underLineNum > 0)

MovePlayer(fMovePos, moveDirection, underLineNum, 1, 0);

else

MovePlayer(fMovePos, moveDirection, 0, 1, -1);

CheckOut(fMovePos, moveDirection);

CalcFCenterPos();

}

}