**그림판 예제**

/////////////////////////////////////////////////////////

CLINET - MAIN

/////////////////////////////////////////////////////////

// DrawServer.cpp : 애플리케이션에 대한 진입점을 정의합니다.

//

#pragma comment(lib, "ws2\_32.lib")

#pragma warning(disable:4996)

#ifdef \_DEBUG

#pragma comment(linker, "/entry:wWinMainCRTStartup /subsystem:console")

#endif

#include "framework.h"

#include "DrawServer.h"

#include <windowsx.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include "RingBuffer.h"

#define CONNECT\_IP \_T("1.236.10.181")

#define UM\_NETWORK (WM\_USER + 1)

// 전역 변수:

HINSTANCE hInst; // 현재 인스턴스입니다.

HWND hWnd;

// 이 코드 모듈에 포함된 함수의 선언을 전달합니다:

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

struct DRAW\_PACKET

{

int StartX;

int StartY;

int EndX;

int EndY;

};

#pragma pack(push, 1)

struct DRAW\_PACKET\_TEST

{

short header;

int StartX;

int StartY;

int EndX;

int EndY;

};

#pragma pack(pop)

BOOL g\_Connect = false;

BOOL g\_Send;

int CurX;

int CurY;

int OldX;

int OldY;

void ProcRead();

void ProcWrite();

void SendPacket();

void MsgProc(DRAW\_PACKET\* Packet, int PacketSize);

void ErrorPrint(DWORD err);

SOCKET Socket;

CRingBuffer SendQ(50);

CRingBuffer RecvQ(50);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

//윈도우 클래스 등록

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_DRAWSERVER));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_DRAWSERVER);

wcex.lpszClassName = \_T("DrawServerName");

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

RegisterClassExW(&wcex);

//윈도우 생성

hInst = hInstance; // 인스턴스 핸들을 전역 변수에 저장합니다.

hWnd = CreateWindowW(\_T("DrawServerName"), \_T("DrawServerTitle"), WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

//소켓생성

WSADATA wsa;

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

return -1;

Socket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (Socket == INVALID\_SOCKET)

return -1;

SOCKADDR\_IN sockaddr\_in;

memset(&sockaddr\_in, 0, sizeof(sockaddr\_in));

sockaddr\_in.sin\_family = AF\_INET;

sockaddr\_in.sin\_port = htons(25000);

InetPton(AF\_INET, CONNECT\_IP, &sockaddr\_in.sin\_addr);

WSAAsyncSelect(Socket, hWnd, UM\_NETWORK, FD\_CONNECT | FD\_CLOSE | FD\_READ | FD\_WRITE);

int ret = connect(Socket, (SOCKADDR\*)&sockaddr\_in, sizeof(sockaddr\_in));

if (ret == SOCKET\_ERROR)

{

if (WSAGetLastError() != WSAEWOULDBLOCK)

{

OutputDebugString(\_T("CONNECT ERROR\n"));

return -1;

}

}

MSG msg;

// 기본 메시지 루프입니다:

while (true)

{

if (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE))

{

if (msg.message == WM\_QUIT)

break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

else

{

if (g\_Connect == TRUE)

{

//Update();

}

}

}

return (int)msg.wParam;

}

//

// 함수: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// 용도: 주 창의 메시지를 처리합니다.

//

// WM\_COMMAND - 애플리케이션 메뉴를 처리합니다.

// WM\_PAINT - 주 창을 그립니다.

// WM\_DESTROY - 종료 메시지를 게시하고 반환합니다.

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_LBUTTONDOWN:

{

OldX = GET\_X\_LPARAM(lParam);

OldY = GET\_Y\_LPARAM(lParam);

break;

}

case WM\_MOUSEMOVE:

{

if (wParam == MK\_LBUTTON)

{

CurX = GET\_X\_LPARAM(lParam);

CurY = GET\_Y\_LPARAM(lParam);

SendPacket();

OldX = CurX;

OldY = CurY;

}

break;

}

case UM\_NETWORK:

{

if (WSAGETSELECTERROR(lParam))

{

OutputDebugString(\_T("UM\_NETWORK : ERROR\n"));

return -1;

}

switch (WSAGETSELECTEVENT(lParam))

{

case FD\_CONNECT:

g\_Connect = true;

break;

case FD\_READ:

ProcRead();

break;

case FD\_WRITE:

g\_Send = true;

ProcWrite();

break;

case FD\_CLOSE:

break;

}

}

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// 메뉴 선택을 구문 분석합니다:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: 여기에 hdc를 사용하는 그리기 코드를 추가합니다...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// 정보 대화 상자의 메시지 처리기입니다.

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

void ProcRead()

{

char RecvBuff[5000];

//리시브할떄 최대한 많이 뽑는것이 원칙

int Recv\_ret = recv(Socket, RecvBuff, 5000, NULL);

if (Recv\_ret == SOCKET\_ERROR)

{

DWORD err = WSAGetLastError();

if (err != WSAEWOULDBLOCK)

{

OutputDebugString(\_T("RecvAPI ERROR\n"));

return;

}

}

//Recv하여 뽑은 데이터를 RecvQ.Enque()

int RecvQ\_ret = RecvQ.Enque(RecvBuff, Recv\_ret);

if (RecvQ\_ret != Recv\_ret)

{

//RecvQ가 가득차서 Enque하지못하고있음.

RecvQ.GetUseSize();

OutputDebugString(\_T("RecvQ ERROR\n"));

return;

}

int Count = 0;

while (true)

{

Count++;

USHORT header = 0;

int RecvQp\_Ret = RecvQ.peek((char\*)&header, sizeof(header));

if (RecvQp\_Ret != sizeof(header))

{

break;

}

//뽑은 헤더의 길이가 패킷의 길이가 아니라면

if (header != sizeof(DRAW\_PACKET))

{

RecvQ.peek((char\*)&header, sizeof(header));

break;

}

RecvQ.MoveFront(sizeof(header));

///RecvQ에서 headr길이만큼 데이터를 뽑아온다.

DRAW\_PACKET Packet;

RecvQp\_Ret = RecvQ.peek((char\*)&Packet, sizeof(Packet));

if (RecvQp\_Ret != sizeof(Packet))

{

OutputDebugString(\_T("RecvQ.peek RecvBuff2 Error"));

break;

}

RecvQ.MoveFront(sizeof(Packet));

MsgProc(&Packet, sizeof(DRAW\_PACKET));

}

}

void ProcWrite()

{

SendPacket();

}

void SendPacket()

{

//좌표가 변하지않았다면 return

if (OldX == CurX && OldY == CurY)

return;

DRAW\_PACKET Packet;

Packet.StartX = OldX;

Packet.StartY = OldY;

Packet.EndX = CurX;

Packet.EndY = CurY;

//먼저 헤더를 SendQ.Enque()

USHORT header = sizeof(Packet);

int SendEnQ\_ret = SendQ.Enque((char\*)&header, sizeof(header));

if (SendEnQ\_ret != sizeof(header))

{

OutputDebugString(\_T("SendQ header Enque ERROR\n"));

return;

}

//이후 패킷을 SendQ.Enque()

SendEnQ\_ret = SendQ.Enque((char\*)&Packet, sizeof(Packet));//3번째 인자 header로 바꾸기

if (SendEnQ\_ret != sizeof(Packet))

{

SendQ.GetUseSize();

OutputDebugString(\_T("SendQ packet Enque ERROR\n"));

return;

}

//지금 보낼상황이 아니라면 return

if (g\_Send == false)

return;

char Sendbuffer[5000];

int SendSize = SendQ.peek(Sendbuffer, sizeof(Sendbuffer));

int Send\_ret = send(Socket, Sendbuffer, SendSize, NULL);

if (Send\_ret == SOCKET\_ERROR)

{

DWORD err = WSAGetLastError();

if (err != WSAEWOULDBLOCK)

ErrorPrint(err);

else

g\_Send = false;

}

SendQ.MoveFront(Send\_ret);

}

void MsgProc(DRAW\_PACKET\* Packet, int PacketSize)

{

HDC hdc = GetDC(hWnd);

MoveToEx(hdc, Packet->StartX, Packet->StartY, NULL);

LineTo(hdc, Packet->EndX, Packet->EndY);

ReleaseDC(hWnd, hdc);

\_tprintf(\_T("StartX : %d\t StartY : %d\t EndX : %d\t, EndY : %d\n")

, Packet->StartX, Packet->StartY, Packet->EndX, Packet->EndY);

}

void ErrorPrint(DWORD err)

{

\_tprintf(\_T("%d"), err);

}

/////////////////////////////////////////////////////////

SERVER - MAIN

/////////////////////////////////////////////////////////

// TMP.cpp : 애플리케이션에 대한 진입점을 정의합니다.

//

#pragma comment(lib, "ws2\_32.lib")

#pragma warning(disable:4996)

#include "framework.h"

#include "DrawServer.h"

#include "RingBuffer.h"

#include <WS2tcpip.h>

#include <winsock2.h>

#include <list>

#define SERVER\_IP "1.236.10.181"

#define UM\_NETWORK WM\_USER+1

#pragma comment(linker, "/entry:wWinMainCRTStartup /subsystem:console")

// 전역 변수:

HINSTANCE hInst; // 현재 인스턴스입니다.

HWND hWnd;

SOCKET g\_ListenSocket;

BOOL g\_Connect = false;

BOOL g\_Send;

struct DRAW\_PACKET

{

int StartX;

int StartY;

int EndX;

int EndY;

};

struct SESSION

{

SOCKET Socket;

CRingBuffer RecvQ;

CRingBuffer SendQ;

SOCKADDR\_IN Addr\_in;

BOOL Sendflag = FALSE;

BOOL DeleteFlag = FALSE;

};

std::list<SESSION\*>Slist;

SESSION\* FindSession(SOCKET socket);

BOOL RemoveSessionFlag(SESSION\* Session);

void RemoveSession(SESSION\* Session);

void ProcAccept();

void ProcRead(SOCKET ClientSocket);

void ProcWrite(SESSION\* Session);

void SendBroadCast(DRAW\_PACKET\* Packet);

void SendPacket(SESSION\* Session, DRAW\_PACKET\* Packet);

void PRINT\_ERROR(DWORD err);

void MsgProc(DRAW\_PACKET\* Packet);

// 이 코드 모듈에 포함된 함수의 선언을 전달합니다:

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDC\_DRAWSERVER));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_DRAWSERVER);

wcex.lpszClassName = \_T("DrawServerName");

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

RegisterClassExW(&wcex);

hInst = hInstance; // 인스턴스 핸들을 전역 변수에 저장합니다.

hWnd = CreateWindowW(\_T("DrawServerName"), \_T("DrawServerTitle"), WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

WSADATA wsa;

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

return -1;

g\_ListenSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (g\_ListenSocket == INVALID\_SOCKET)

return -1;

SOCKADDR\_IN addr\_in;

memset(&addr\_in, 0, sizeof(addr\_in));

addr\_in.sin\_family = AF\_INET;

addr\_in.sin\_port = htons(25000);

InetPton(AF\_INET, \_T(SERVER\_IP), &addr\_in.sin\_addr);

if (bind(g\_ListenSocket, (SOCKADDR\*)&addr\_in, sizeof(addr\_in)) != 0)

return -1;

if (listen(g\_ListenSocket, SOMAXCONN) != 0)

return -1;

WSAAsyncSelect(g\_ListenSocket, hWnd, UM\_NETWORK, FD\_ACCEPT | FD\_CLOSE | FD\_READ | FD\_WRITE);

MSG msg;

// 기본 메시지 루프입니다:

while (true)

{

if (PeekMessage(&msg, NULL, 0, 0, PM\_REMOVE))

{

if (msg.message == WM\_QUIT)

break;

TranslateMessage(&msg);

DispatchMessage(&msg);

}

else

{

if (g\_Connect == TRUE)

{

}

//연결플래그 on되면 게임로직 실행

}

}

return (int)msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case UM\_NETWORK:

{

//모든 클라와 Listen소켓의 에러가 모두 이쪽으로 올것.

if (WSAGETSELECTERROR(lParam))

{

SESSION\* p = FindSession((SOCKET)wParam);

RemoveSessionFlag(p);

}

switch (WSAGETSELECTEVENT(lParam))

{

case FD\_ACCEPT:

ProcAccept();

break;

case FD\_READ:

ProcRead((SOCKET)wParam);

break;

case FD\_WRITE:

{

g\_Send = true;

SESSION\* FSession = FindSession((SOCKET)wParam);

ProcWrite(FSession);

break;

}

case FD\_CLOSE:

{

SESSION\* RSession = FindSession((SOCKET)wParam);

RemoveSessionFlag(RSession);

break;

}

}

}

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// 메뉴 선택을 구문 분석합니다:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: 여기에 hdc를 사용하는 그리기 코드를 추가합니다...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

SESSION\* FindSession(SOCKET socket)

{

for (std::list<SESSION\*>::iterator iter = Slist.begin(); iter != Slist.end(); ++iter)

{

if ((\*iter)->DeleteFlag == false && socket == (\*iter)->Socket)

return (\*iter);

}

return nullptr;

}

BOOL RemoveSessionFlag(SESSION\* Session)

{

Session->DeleteFlag = TRUE;

return TRUE;

}

void RemoveSession(SESSION\* Session)

{

for (std::list<SESSION\*>::iterator iter = Slist.begin(); iter != Slist.end(); ++iter)

{

if ((\*iter)->DeleteFlag == TRUE && Session->Socket == (\*iter)->Socket)

{

SESSION\* DSession = (\*iter);

iter = Slist.erase(iter);

delete DSession;

DSession = nullptr;

}

}

}

void ProcAccept()

{

SOCKADDR\_IN addr;

INT addrLen = sizeof(addr);

SOCKET Socket = accept(g\_ListenSocket, (SOCKADDR\*)&addr, &addrLen);

SESSION\* nSession = new SESSION;

nSession->Socket = Socket;

nSession->Addr\_in = addr;

nSession->DeleteFlag = FALSE;

nSession->Sendflag = FALSE;

TCHAR addrBuffer[sizeof(SOCKADDR\_IN) \* 2];

InetNtop(AF\_INET, &addr, addrBuffer, \_countof(addrBuffer));

Slist.push\_back(nSession);

\_tprintf(\_T("accept Success \t\t Socket : %d\tIP : %s\n"), Socket, addrBuffer);

}

void ProcRead(SOCKET ClientSocket)

{

SESSION\* p = FindSession(ClientSocket);

if (p == nullptr)

{

PRINT\_ERROR(-1);

return;

}

char Recvbuffer[5000];

int RecvAPI\_ret = recv(p->Socket, Recvbuffer, sizeof(Recvbuffer), NULL);

int RecvEQ\_ret = p->RecvQ.Enque(Recvbuffer, RecvAPI\_ret);

if (RecvEQ\_ret != RecvAPI\_ret)

{

OutputDebugString(\_T("RecvQ\_Ret ERROR"));

return;

}

int Count = 0;

while (true)

{

++Count;

USHORT header = 0;

int RecvPQ\_ret = p->RecvQ.peek((char\*)&header, sizeof(header));

if (header != sizeof(DRAW\_PACKET))

{

p->RecvQ.peek((char\*)&header, sizeof(header));

OutputDebugString(\_T("RecvPQ\_ret ERROR"));

break;

}

p->RecvQ.MoveFront(sizeof(header));

DRAW\_PACKET Packet;

int RecvDQ\_ret = p->RecvQ.peek((char\*)&Packet, sizeof(Packet));

if (RecvDQ\_ret != sizeof(Packet))

{

OutputDebugString(\_T("RecvDQ\_ret ERROR"));

break;

}

p->RecvQ.MoveFront(sizeof(Packet));

MsgProc(&Packet);

SendBroadCast(&Packet);

}

}

void SendBroadCast(DRAW\_PACKET\* Packet)

{

for (std::list<SESSION\*>::iterator iter = Slist.begin(); iter != Slist.end(); ++iter)

{

if((\*iter)->DeleteFlag == FALSE)

SendPacket((\*iter), Packet);

}

}

void SendPacket(SESSION\* Session, DRAW\_PACKET\* Packet)

{

USHORT header = sizeof(\*Packet);

int SendEQ\_ret = Session->SendQ.Enque((char\*)&header, sizeof(header));

if (SendEQ\_ret != sizeof(USHORT))

{

OutputDebugString(\_T("SendEQ\_ret ERROR"));

return;

}

SendEQ\_ret = Session->SendQ.Enque((char\*)Packet, sizeof(DRAW\_PACKET));

if (SendEQ\_ret != sizeof(DRAW\_PACKET))

{

OutputDebugString(\_T("SendEQ\_ret ERROR"));

return;

}

if(g\_Send != false)

ProcWrite(Session);

}

void ProcWrite(SESSION\* Session)

{

while (true)

{

//여기서 Session->SendQ에서 Deque해서 실제로 send. 보낼수있을만큼 보낸다.

char buffer[5000];

int SendDQ\_ret = Session->SendQ.peek(buffer, sizeof(buffer));

if (SendDQ\_ret == 0)

break;

int SendAPI\_ret = send(Session->Socket, buffer, SendDQ\_ret, NULL);

if (SendAPI\_ret == SOCKET\_ERROR)

{

DWORD err = WSAGetLastError();

if (err != WSAEWOULDBLOCK)

{

RemoveSessionFlag(Session);

return;

}

else

{

g\_Send = false;

break;

}

}

Session->SendQ.MoveFront(SendDQ\_ret);

}

}

void PRINT\_ERROR(DWORD err)

{

\_tprintf(\_T("ERROR CODE : %d"), err);

if (err == -1)

{

//일부러 Crash를 내는상황

int\* p = nullptr;

\*p = 10;

}

}

void MsgProc(DRAW\_PACKET\* Packet)

{

HDC hdc = GetDC(hWnd);

MoveToEx(hdc, Packet->StartX, Packet->StartY, NULL);

LineTo(hdc, Packet->EndX, Packet->EndY);

ReleaseDC(hWnd, hdc);

}

// 정보 대화 상자의 메시지 처리기입니다.

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

**별이동 예제**

========================================================

별이동 예제 SERVER

========================================================

// StarServer.cpp : 이 파일에는 'main' 함수가 포함됩니다. 거기서 프로그램 실행이 시작되고 종료됩니다.

//

#pragma comment(lib, "ws2\_32.lib")

#include <iostream>

#include <winsock2.h>

#include <WS2tcpip.h>

#include <tchar.h>

#include <list>

#define CONOSLE\_WIDTH 80

#define CONSOLE\_HEIGHT 23

struct \_MSG

{

UINT Type; //1. 생성 2.삭제 3.이동

UINT ID;

UINT X, Y;

};

enum TYPE\_NUM

{

T\_TAG = 0,

T\_CREATE,

T\_DELETE,

T\_MOVE

};

//클라이언트 정보

struct \_DATA

{

UINT ID;

UINT X, Y;

};

//서버가 들고있는 연결된 클라정보

struct \_SESSION

{

SOCKET socket;

\_DATA data;

SOCKADDR\_IN addr;

};

void MyAccept(SOCKET\* ListenSocket);

void MyUniCast(\_SESSION\* DstSession, \_MSG\* msg);

void Disconnect(\_SESSION\* session);

void MyBroadCast(\_SESSION\* ExpSession, \_MSG\* msg);

void MoveStar(\_MSG\* msg);

void Render();

void cs\_Initial(void);

void cs\_MoveCursor(int iPosX, int iPosY);

std::list<\_SESSION\*>Slist;

HANDLE hConsole;

UINT g\_ID = 0;

int main()

{

srand(unsigned(NULL));

WSADATA wsa;

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

return -1;

SOCKET ListenSocket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (ListenSocket == INVALID\_SOCKET)

return -1;

ULONG on = 1;

if (ioctlsocket(ListenSocket, FIONBIO, &on) == SOCKET\_ERROR)

return -1;

SOCKADDR\_IN addr\_in;

memset(&addr\_in, 0, sizeof(addr\_in));

addr\_in.sin\_family = AF\_INET;

addr\_in.sin\_port = htons(3000);

if (InetPton(AF\_INET, L"1.236.10.181", &addr\_in.sin\_addr) != 1)

return -1;

if (bind(ListenSocket, (SOCKADDR\*)&addr\_in, sizeof(addr\_in)) != 0)

return -1;

listen(ListenSocket, SOMAXCONN);

while (true)

{

//네트워크 파트

FD\_SET rset;

FD\_ZERO(&rset);

FD\_SET(ListenSocket, &rset);

//Select에 socket 세팅

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

FD\_SET((\*Pos)->socket, &rset);

//Select

timeval time;

time.tv\_sec = 0;

time.tv\_usec = 1;

int ret = select(NULL, &rset, NULL, NULL, NULL);

if (ret > 0)

{

//Listen 반응시 Accept

if (FD\_ISSET(ListenSocket, &rset) != 0)

MyAccept(&ListenSocket);

//Select를 거치고 어떤 소켓이 남았는가 확인

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

{

//반응이 있었다면

if (FD\_ISSET((\*Pos)->socket, &rset) != 0)

{

//recv

\_MSG MsgBuff = { 0, };

int ret = recv((\*Pos)->socket, (char\*)&MsgBuff, sizeof(MsgBuff), NULL);

if (WSAECONNRESET == WSAGetLastError())

{

Disconnect(\*Pos);

break;

}

while (ret == sizeof(\_MSG)) //WOULDBLOCK뜰때까지 돌리기

{

switch (MsgBuff.Type)

{

case T\_CREATE: //accept에서 생성

break;

case T\_DELETE: //Disconnect에서 삭제

break;

case T\_MOVE:

MoveStar(&MsgBuff);

}

ret = recv((\*Pos)->socket, (char\*)&MsgBuff, sizeof(MsgBuff), NULL);

}

}

}

}

Render();

}

WSACleanup();

}

//ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

//FD\_SET반응이 있을때 accept하고 새로운세션 만들어 삽입.

//ㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡㅡ

void MyAccept(SOCKET\* ListenSocket)

{

SOCKADDR\_IN addr\_in;

int addrSize = sizeof(addr\_in);

SOCKET Socket = accept(\*ListenSocket, (SOCKADDR\*)&addr\_in, &addrSize);

\_SESSION\* session = new \_SESSION;

session->socket = Socket;

session->addr = (SOCKADDR\_IN)addr\_in;

session->data.ID = g\_ID++;

session->data.X = rand() % 70 + 1; //Console Size 80\*21

session->data.Y = rand() % 15 + 1;

//새별의 생성을 해당별을 제외한 모든 클라이언트한테 전파

\_MSG msg;

msg.Type = T\_CREATE;

msg.ID = session->data.ID;

msg.X = session->data.X;

msg.Y = session->data.Y;

MyBroadCast(session, &msg);

//생성된 별의 ID전달

msg.Type = T\_TAG;

MyUniCast(session, &msg);

msg.Type = T\_CREATE;

MyUniCast(session, &msg);

//새별에게 기존별들을 전파

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

{

\_MSG msg;

msg.Type = T\_CREATE;

msg.ID = (\*Pos)->data.ID;

msg.X = (\*Pos)->data.X;

msg.Y = (\*Pos)->data.Y;

MyUniCast(session, &msg);

}

//새로추가된 별을 Slist에 추가

Slist.push\_back(session);

}

void MyBroadCast(\_SESSION\* ExpSession, \_MSG\* msg)

{

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

{

if ((\*Pos) != ExpSession)

{

MyUniCast((\*Pos), msg);

}

}

}

void MyUniCast(\_SESSION\* DstSession, \_MSG\* msg)

{

int ret = send(DstSession->socket, (const char\*)msg, sizeof(\_MSG), NULL);

}

void Disconnect(\_SESSION\* session)

{

closesocket(session->socket);

\_MSG msg;

msg.Type = T\_DELETE;

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

{

if ((\*Pos)->socket == session->socket)

{

msg.ID = session->data.ID;

msg.X = session->data.X;

msg.Y = session->data.Y;

\_SESSION\* dSession = (\*Pos);

Pos = Slist.erase(Pos);

delete dSession;

dSession = nullptr;

break;

}

}

MyBroadCast(NULL, &msg);

}

void MoveStar(\_MSG\* msg)

{

//좌표 예외처리

if (msg->X > CONOSLE\_WIDTH || msg->Y > CONSOLE\_HEIGHT

|| msg->X < 0 || msg->Y < 0)

return;

\_SESSION\* ExpSession = NULL;

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

{

if ((\*Pos)->data.ID == msg->ID)

{

(\*Pos)->data.X = msg->X;

(\*Pos)->data.Y = msg->Y;

ExpSession = (\*Pos);

break;

}

}

MyBroadCast(ExpSession, msg);

}

void Render()

{

system("cls");

cs\_Initial();

cs\_MoveCursor(0, 0);

\_tprintf(\_T("Connect Client : %d"), Slist.size());

for (std::list<\_SESSION\*>::iterator Pos = Slist.begin(); Pos != Slist.end(); ++Pos)

{

cs\_MoveCursor((\*Pos)->data.X, (\*Pos)->data.Y);

\_tprintf(\_T("\*"));

}

}

void cs\_Initial(void)

{

CONSOLE\_CURSOR\_INFO stConsoleCursor;

stConsoleCursor.bVisible = FALSE;

stConsoleCursor.dwSize = 1; // 커서 크기

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorInfo(hConsole, &stConsoleCursor);

}

void cs\_MoveCursor(int iPosX, int iPosY)

{

COORD stCoord;

stCoord.X = iPosX;

stCoord.Y = iPosY;

SetConsoleCursorPosition(hConsole, stCoord);

}

========================================================

별이동 예제 CLIENT

========================================================

//

#pragma comment(lib, "ws2\_32.lib")

#include <winsock2.h>

#include <Ws2tcpip.h>

#include <tchar.h>

SOCKET Socket;

UINT g\_MyID;

UINT OldX, OldY;

INT MyConnect(SOCKADDR\* sockaddr\_in);

void KeyInput();

void Networkprocessing();

void cs\_Initial(void);

void cs\_MoveCursor(int iPosX, int iPosY);

struct \_MSG

{

UINT Type;

UINT ID;

UINT X, Y;

};

enum DIRECT

{

NON = 0, UP, DOWN, RIGHT, LEFT

};

class CStars

{

public:

struct Star

{

UINT Id;

UINT X, Y;

};

public:

explicit CStars() { cs\_Initial(); };

virtual ~CStars() {};

public:

void CreateStar(UINT Id, UINT X, UINT Y)

{

\_stars[\_Count] = new Star;

\_stars[\_Count]->Id = Id;

\_stars[\_Count]->X = X;

\_stars[\_Count]->Y = Y;

if (\_Count++ >= \_countof(\_stars))

return;

}

bool SearchID(UINT ID)

{

for (int i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

if (\_stars[i]->Id == ID)

{

OldX = \_stars[i]->X;

OldY = \_stars[i]->Y;

return TRUE;

}

}

}

return FALSE;

}

void DeleteStar(UINT Id)

{

for (int i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

if (\_stars[i]->Id == Id)

{

delete \_stars[i];

\_stars[i] = nullptr;

--\_Count;

}

}

}

}

void Move(UINT Id, DIRECT Dir)

{

int i;

for (i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

if (\_stars[i]->Id == Id)

break;

}

}

switch (Dir)

{

case UP: \_stars[i]->Y--;

break;

case DOWN: \_stars[i]->Y++;

break;

case RIGHT: \_stars[i]->X++;

break;

case LEFT: \_stars[i]->X--;

break;

}

}

void SetPosition(UINT Id, UINT X, UINT Y)

{

int i;

for (i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

if (\_stars[i]->Id == Id)

{

\_stars[i]->X = X;

\_stars[i]->Y = Y;

}

}

}

}

void GetPosition(UINT Id, UINT\* X, UINT\* Y)

{

for (int i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

if (\_stars[i]->Id == Id)

{

\*X = \_stars[i]->X;

\*Y = \_stars[i]->Y;

}

}

}

}

bool IsMove(UINT Id)

{

for (int i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

if (\_stars[i]->Id == Id)

{

if (\_stars[i]->X != OldX || \_stars[i]->Y != OldY)

return TRUE;

}

}

}

return FALSE;

}

void Render()

{

system("cls");

for (int i = 0; i < \_countof(\_stars); ++i)

{

if (\_stars[i] != nullptr)

{

cs\_MoveCursor(\_stars[i]->X, \_stars[i]->Y);

\_tprintf(\_T("\*"));

}

}

}

private:

Star\* \_stars[24] = { 0, };

UINT \_Count = 0;

};

CStars Stars;

int main()

{

WSADATA wsa;

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

return -1;

//소켓 생성

Socket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (Socket == INVALID\_SOCKET)

return -1;

//비동기 전환

ULONG on = 1;

if (ioctlsocket(Socket, FIONBIO, &on) == SOCKET\_ERROR)

return -1;

//SOCKADDR 세팅

SOCKADDR\_IN sockaddr\_in;

memset(&sockaddr\_in, 0, sizeof(sockaddr\_in));

sockaddr\_in.sin\_family = AF\_INET;

sockaddr\_in.sin\_port = htons(3000);

//IP세팅

TCHAR IP[48] = { 0, };

\_tprintf(\_T("IP :"));

\_tscanf\_s(\_T("%s"), IP, \_countof(IP));

if (InetPton(AF\_INET, IP, &sockaddr\_in.sin\_addr) != 1)

{

int err = WSAGetLastError();

return -1;

}

while (true)

{

//연결

if (MyConnect((SOCKADDR\*)&sockaddr\_in) != 0)

{

Networkprocessing();

if (Stars.SearchID(g\_MyID) == TRUE)

{

KeyInput();

if (Stars.IsMove(g\_MyID) == TRUE)

{

\_MSG msg;

Stars.GetPosition(g\_MyID, &msg.X, &msg.Y);

msg.Type = 3;

msg.ID = g\_MyID;

int ret = send(Socket, (const char\*)&msg, sizeof(msg), NULL);

}

}

}

//1234

Stars.Render();

}

closesocket(Socket);

WSACleanup();

}

INT MyConnect(SOCKADDR\* sockaddr\_in)

{

int time = GetTickCount64();

int ret = connect(Socket, sockaddr\_in, sizeof(\*sockaddr\_in));

int err = GetLastError();

if (ret == 0)

return 1;

else

{

// 일정시간이 지났다면 실패로 간주한다.

if (WSAGetLastError() == WSAEWOULDBLOCK)

{

if (GetTickCount64() - time >= 300)

return 0;

}

else if (WSAGetLastError() == 10056)

{

return 1;

}

}

}

void KeyInput()

{

if ((GetAsyncKeyState(VK\_RIGHT) & 0x8000) != false)

Stars.Move(g\_MyID, RIGHT);

if ((GetAsyncKeyState(VK\_LEFT) & 0x8000) != false)

Stars.Move(g\_MyID, LEFT);

if ((GetAsyncKeyState(VK\_UP) & 0x8000) != false)

Stars.Move(g\_MyID, UP);

if ((GetAsyncKeyState(VK\_DOWN) & 0x8000) != false)

Stars.Move(g\_MyID, DOWN);

}

void Networkprocessing()

{

FD\_SET rset;

FD\_ZERO(&rset);

FD\_SET(Socket, &rset);

timeval time;

time.tv\_sec = 0; //프레임에 맞게 들어가야한다.

time.tv\_usec = 1;

int ret = select(NULL, &rset, NULL, NULL, &time);

\_MSG MsgBuff = { 0, };

if (ret > 0)

{

while (sizeof(\_MSG) == recv(Socket, (char\*)&MsgBuff, sizeof(MsgBuff), NULL))

{

//받을게 없을때까지

switch (MsgBuff.Type)

{

case 0:

{

g\_MyID = MsgBuff.ID;

break;

}

case 1: Stars.CreateStar(MsgBuff.ID, MsgBuff.X, MsgBuff.Y);

break;

case 2: Stars.DeleteStar(MsgBuff.ID);

break;

case 3: Stars.SetPosition(MsgBuff.ID, MsgBuff.X, MsgBuff.Y);

break;

}

}

}

}

HANDLE hConsole;

void cs\_Initial(void)

{

CONSOLE\_CURSOR\_INFO stConsoleCursor;

stConsoleCursor.bVisible = FALSE;

stConsoleCursor.dwSize = 1;

hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorInfo(hConsole, &stConsoleCursor);

}

void cs\_MoveCursor(int iPosX, int iPosY)

{

COORD stCoord;

stCoord.X = iPosX;

stCoord.Y = iPosY;

SetConsoleCursorPosition(hConsole, stCoord);

}

**UDP방찾기**

========================================================

UDP 방번호 알아내기 예제 SERVER

========================================================

// SERVER\_TEST.cpp : 이 파일에는 'main' 함수가 포함됩니다. 거기서 프로그램 실행이 시작되고 종료됩니다.

//

#pragma comment(lib, "ws2\_32.lib")

#include <wchar.h>

#include <iostream>

#include <ws2tcpip.h>

#include <winsock2.h>

#include <tchar.h>

int main()

{

WSADATA wsa;

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

return -1;

SOCKET USocket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (USocket == SOCKET\_ERROR)

return -1;

SOCKADDR\_IN sockaddr\_in;

memset(&sockaddr\_in, 0, sizeof(sockaddr\_in));

sockaddr\_in.sin\_family = AF\_INET;

sockaddr\_in.sin\_port = htons(10010);

if (InetPton(AF\_INET, L"1.236.10.181", &sockaddr\_in.sin\_addr) != 1)

return -1;

bind(USocket, (SOCKADDR\*)&sockaddr\_in, sizeof(sockaddr\_in));

TCHAR protocol[10] = { '0xff', '0xee','0xdd', '0xaa', '0x00', '0x99', '0x77', '0x55' ,'0x33', '0x11' };

while (true)

{

FD\_SET rset;

FD\_ZERO(&rset);

FD\_SET(USocket, &rset);

int selret = select(NULL, &rset, NULL, NULL, NULL);

if (selret > 0)

{

SOCKADDR\_IN rsockaddr;

int addrSize = sizeof(rsockaddr);

TCHAR rbuff[10] = { 0, };

int rret = recvfrom(USocket, (char\*)rbuff, sizeof(rbuff), NULL, (SOCKADDR\*)&rsockaddr, &addrSize);

if (rret == 20)

{

if (\_tcscmp(rbuff, protocol) == 0)

{

TCHAR sbuff[10] = { \_T("good.") };

if (sizeof(sbuff) == sendto(USocket, (const char\*)sbuff, sizeof(sbuff), NULL, (SOCKADDR\*)&rsockaddr, sizeof(rsockaddr)))

{

\_tprintf(\_T("send success\n"));

}

}

}

}

}

closesocket(USocket);

WSACleanup();

}

========================================================

UDP 방번호 알아내기 예제 CLIENT

========================================================

// TEST.cpp : 이 파일에는 'main' 함수가 포함됩니다. 거기서 프로그램 실행이 시작되고 종료됩니다.

//

#pragma comment(lib, "ws2\_32.lib")

#include <iostream>

#include <winsock2.h>

#include <WS2tcpip.h>

#include <tchar.h>

int main()

{

WSADATA wsa;

if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)

return -1;

SOCKET UDP\_Socket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (UDP\_Socket == INVALID\_SOCKET)

return -1;

BOOL fBroadCaset = 1;

setsockopt(UDP\_Socket, SOL\_SOCKET, SO\_BROADCAST, (const char\*)&fBroadCaset, sizeof(fBroadCaset));

ULONG on = 1;

if (ioctlsocket(UDP\_Socket, FIONBIO, &on) == SOCKET\_ERROR)

return -1;

SOCKADDR\_IN addr\_in;

memset(&addr\_in, 0, sizeof(addr\_in));

addr\_in.sin\_family = AF\_INET;

if (InetPton(AF\_INET, L"255.255.255.255", &addr\_in.sin\_addr) != 1)

return -1;

TCHAR sbuff[] = { '0xff', '0xee','0xdd', '0xaa', '0x00', '0x99', '0x77', '0x55' ,'0x33', '0x11' };

SOCKADDR\_IN recv\_addr;

int addr\_len = sizeof(SOCKADDR\_IN);

int ret;

for (int i = 10001; i < 10099; ++i)

{

FD\_SET rset;

FD\_ZERO(&rset);

FD\_SET(UDP\_Socket, &rset);

addr\_in.sin\_port = htons(i); //10001 ~ 10099

ret = sendto(UDP\_Socket, (const char\*)sbuff, sizeof(sbuff), NULL, (SOCKADDR\*)&addr\_in, sizeof(addr\_in));

\_tprintf(\_T("%d : %d\n"), i, ret);

timeval time;

time.tv\_sec = 0;

time.tv\_usec = 200000;

ret = select(NULL, &rset, NULL, NULL, &time);

if (ret > 0)

{

if (FD\_ISSET(UDP\_Socket, &rset) != 0)

{

\_tprintf(\_T("%d"), i);

char rbuff[10] = { 0, };

recvfrom(UDP\_Socket, (char\*)rbuff, sizeof(rbuff), NULL, (SOCKADDR\*)&recv\_addr, &addr\_len);

// \_tprintf(\_T("%s"), recv\_addr, sizeof(recv\_addr));

}

}

}

}

--------------------------------------------------------------------------------------------------------------