

네트워크 프로그래밍 구현

포트폴리오

악성코드분석 및 모의해킹 전문가 양성과정

1조 김 다 승

차 례

| | | | |
|-----|-----------------------|-------|----|
| 1. | 프로그램 개요 | ----- | 3 |
| 1) | 프로그램 소개 | ----- | 3 |
| 2) | 프로그램 상세설명 | ----- | 3 |
| (1) | 게임 시나리오 | ----- | 3 |
| (2) | 기존의 프로그램의 기능 및 진행방식 | ----- | 3 |
| (3) | 추가 구현한 기능 및 진행 방식 | ----- | 4 |
| 2. | 개요를 바탕으로 프로그램 수정 | ----- | 6 |
| 가. | 포함 되어야 할 기술 | ----- | 6 |
| 1) | 파일 입출력 | ----- | 6 |
| 2) | TCP 소켓 프로그래밍 | ----- | 7 |
| 3) | Unicast | ----- | 9 |
| 4) | Thread | ----- | 10 |
| 3. | 프로그램 시현 | ----- | 13 |
| 4. | 소스 코드 최적화 및 결과 보고서 작성 | ----- | 15 |

1. 프로그램 개요

1) 프로그램 소개

- ▷ 제목 : DODGE! DODGE! (닷지! 닷지!)
- ▷ 장르 : 아케이드
- ▷ 계기 : GomPlayer의 닷지를 모티브로 제작
- ▷ 게임방법 : 게임시작하면 플레이어는 키보드로 맵 벽 안에서 출현하는 예측하지 못하는 공(총알)들의 움직임을 피하도록 구현되었습니다.

2) 프로그램 상세설명

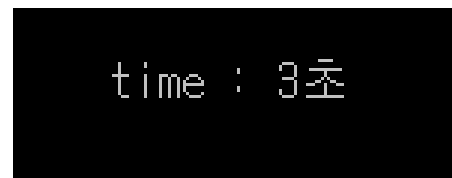
(1) 게임 시나리오

: 실제 GomPlayer의 닷지라는 게임을 모티브로 하여
 각각 유저(Player)가 아래, 위, 양 옆에서 나오는 총알을 피하는 게임으로 정하고
 C언어를 이용해서 구현하였습니다.

(2) 기존의 프로그램의 기능 및 진행방식



[그림 1. 기존 프로그램의 시작 화면]



[그림 2. 게임 시작 시 흘러가는 시간]

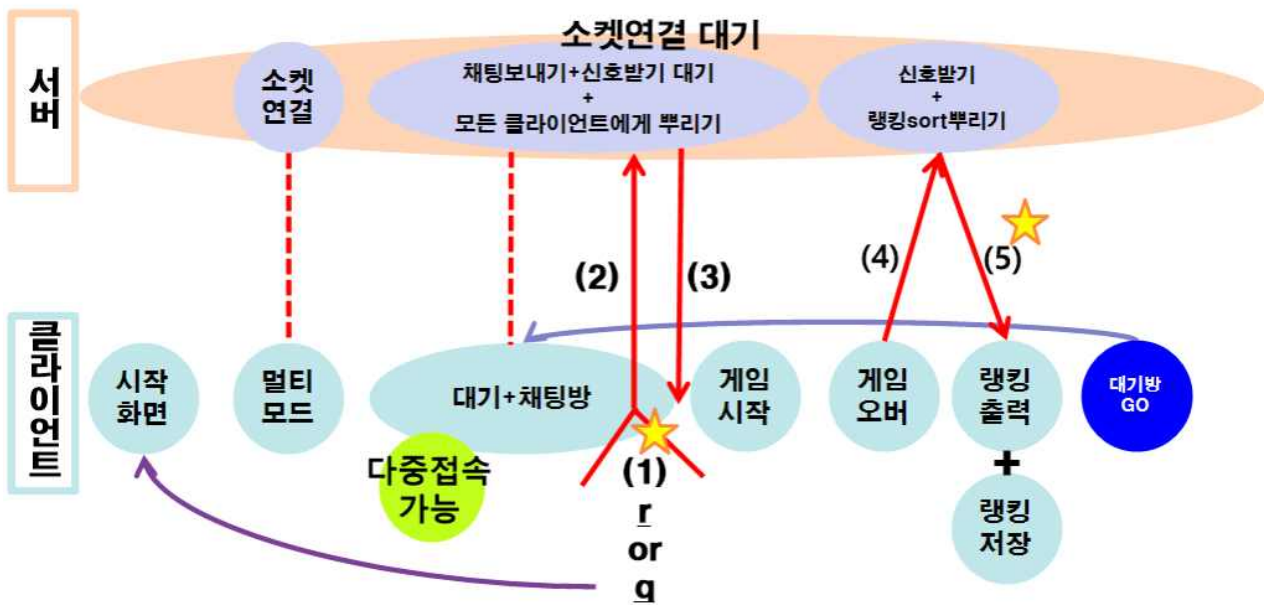
: 메인에서 메뉴를 호출하여 메뉴에 “1. 게임 시작, 2. 게임 방법, 3. 랭킹, 4. 끝내기”에서 선택하여 숫자를 입력하고 Enter키를 눌러서 선택지를 실행합니다.
 기존의 프로그램은 1인용 게임으로 서버와 클라이언트의 개념이 없었습니다.
 게임 시작 시, 대기 시간 없이 바로 시작되며 맵 벽안에서 상하좌우에서 나오는 공들을 플레이어가 키보드로 누를 때마다 출력하도록 하여 움직이는 것처럼 보이게 했습니다.
 플레이어의 점수를 측정하는 기준을 시간으로 잡고 맵 벽 오른쪽에 시간을 출력했습니다. 시간은 게임을 진행하는 도중에 계속 흘러가야 하기 때문에 Thread를 이용하여 구현했습니다.
 게임 오버가 되면 시간을 반환하여 게임오버 화면을 출력한 다음, 플레이어의 이름을 입력하여 이름, 시간 그리고 점수는 시간*10로 저장하여 랭킹정보에 저장합니다.
 그런 다음, 점수를 기준으로 랭킹을 정렬하여 파일입출력으로 rank.txt 파일로 저장하게 됩니다. 이렇게 생성된 랭킹정보는 메인 메뉴에서 “3. 랭킹”을 선택해야만 확인할 수 있습니다.
 그 외, 부가적으로 도움말과 종료를 진행할 수 있도록 구현했습니다.

최후의 클라이언트가 게임오버가 되면 서버는 클라이언트들에게 모든 클라이언트들이 게임오버가 되었다고 알리는 신호를 보내고 받은 클라이언트들은 전체 멀티랭킹정보를 받을 준비를 하게 됩니다.

서버는 클라이언트들에게 전체 멀티랭킹정보를 보내고 모든 클라이언트들은 멀티랭킹 정보를 출력하고 저장합니다.

이로써 멀티게임 1Round는 끝나고 다시 채팅방으로 이동합니다.

게임을 다시 시작할 수 있으며, 멀티모드를 종료하려면 채팅방에서 'q'를 입력하여 전송하는 방법 밖에 없습니다.



[흐름도]

2. 개요를 바탕으로 프로그램 수정

가. 포함 되어야 할 기술

1) 파일 입출력

(1) server - Rankstore()

```
void Rankstore() {
    FILE* savefile;
    savefile = fopen("rank.txt", "wt");

    for(int i = 0; i < cnt; i++){
        fprintf(savefile, "%s %d %d\n", use[i].nickname, use[i].sc, use[i].rank);
    }
    fclose(savefile);
}
```

[그림 5. Rankstore() 함수]

모든 클라이언트들에게 게임오버 신호를 받은 후, 서버에서 측정한 시간과 점수로 파일에 저장한다.

(2) client - RecvNSaveRank(SOCKET sock), MultiRankDraw()

```
void RecvNSaveRank(SOCKET sock)
{
    char buf[1024] = { 0, };
    FILE* SaveRank;
    int check;

    recv(sock, buf, sizeof(buf), 0);
    SaveRank = fopen("MultiRank.txt", "wt");
    fwrite(buf, strlen(buf), 1, SaveRank);
    fclose(SaveRank);

    return;
}
```

서버에서 받은 랭킹정보를 파일입출력으로 저장한다.

[그림 6. RecvNSaveRank() 함수]

```
void MultiRankDraw() { // 저장된 멀티모드 랭킹정보를 읽고 출력하기
    int i, j, key;
    FILE* LoadRank;
    struct info temp;

    LoadRank = fopen("MultiRank.txt", "rt");
    for (i = 0; i < 5; i++) {
        fscanf(LoadRank, "%s %d %d\n", &Multiuser[i].name, &Multiuser[i].score, &Multiuser[i].rank);
    }
    fclose(LoadRank);

    for (i = 0; i < 4; i++) {
        for (j = i + 1; j < 5; j++) {
            if (Multiuser[i].rank != 0 && Multiuser[j].rank != 0) {
                if (Multiuser[i].rank > Multiuser[j].rank) {
                    temp = Multiuser[i];
                    Multiuser[i] = Multiuser[j];
                    Multiuser[j] = temp;
                }
            }
        }
    }

    system("cls");
    for (i = 0; i < 5; i++) {
        gotoxy(32, (i + 4) * 3 + 1);
        printf("[%d위]", Multiuser[i].rank);
        gotoxy(32 + 6, (i + 4) * 3 + 1);
        printf("닉네임: %s", Multiuser[i].name);
        gotoxy(32 + 16, (i + 4) * 3 + 1);
        printf("점수: %d점", Multiuser[i].score);
        Sleep(200);
    }

    gotoxy(37, 30); SKY_BLUE; printf("PRESS SPACE BAR or ENTER!"); ORIGINAL;
    while (1) {
        if (_kbhit()) {
            key = _getch();
            if (key == SPACE || key == ENTER) { // 스페이스바나 엔터키를 누르면 함수 종료
                break;
            }
        }
    }
}
```

저장된 멀티랭킹 정보를 fscanf을 통해 읽습니다.

저는 test를 같은 호스트로 클라이언트 여러 대를 실행시키니, 랭킹정보가 역순으로 들어오는 경우가 있어서 다시 랭킹정렬을 합니다.

그런 다음, 랭킹정보를 출력합니다.

사용자가 다 확인한 다음, 스페이스바나 엔터키를 누르면 이 함수가 끝나도록 했습니다.

[그림 7. MultiRankDraw() 함수]

2) TCP 소켓 프로그래밍

(1) server

```
#define _CRT_SECURE_NO_WARNINGS
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#include<stdio.h>
// #include<WinSock2.h>
#include<stdlib.h>
// #include<WS2tcpip.h>
#include<string.h>
#include<Windows.h>
#include<process.h>
```

[그림 8. 헤더]

※ TCP를 선택한 이유

: UDP보다 속도는 느리지만 신뢰성을 보장하기 때문에 채팅과 같은 응답이 확실해야 하는 기능에 사용해야 하며 마지막에 랭킹정보와 같은 크기가 큰 자료를 보내는 데 적합할 것 같아서 선택하게 되었습니다.

1. WSASStartup : Winsock 초기화

- Winsock 동적 연결 라이브러리를 초기화하고 Winsock 구현이 애플리케이션 요구사항을 충족하는 확인

2. SOCKET : 소켓 생성

- socket(AF_INET, SOCK_STREAM, 0);
- AF_INET : IPv4(주소체계)
- SOCK_STREAM : TCP/IP 사용
- 0 : 프로토콜

3. bind : 주소저장

- 서버의 소켓을 SOCKADDR형의 주소에 묶는다

4. listen : 연결 대기

- 5 : 접속 클라이언트 수를 5명으로 제한

5. accept : 클라이언트의 요청 수리

- 리스닝 소켓(srvSock)을 이용하여 클라이언트의 주소 정보를 받고 반환값을 다른 소켓과 구분하기 위해 클라이언트 소켓으로 받는다.

6. closesocket : 소켓 해제

7. WSACleanup : Winsock 해제

```
int main() {
    WSADATA wsaData;
    SOCKET srvSock, cliSock;
    SOCKADDR_IN srvAddr, cliAddr;
    int cliSz;

    hmut = CreateMutex(NULL, FALSE, NULL);

    //1. WSASStartup
    if (WSASStartup(MAKEWORD(2, 2), &wsaData) != 0)
        printf("WSASStartup Error\n");

    //2. SOCKET
    srvSock = socket(AF_INET, SOCK_STREAM, 0);

    //3. bind
    memset(&srvAddr, 0, sizeof(srvAddr));
    srvAddr.sin_family = AF_INET;
    srvAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    srvAddr.sin_port = htons(12345);

    if (bind(srvSock, (SOCKADDR*)&srvAddr, sizeof(srvAddr)) == SOCKET_ERROR)
        printf("bind Error\n");

    //4. listen : 정원 5명
    if (listen(srvSock, 5) == SOCKET_ERROR)
        printf("listen Error\n");

    //5. accept
    while (1) {
        cliSz = sizeof(cliAddr);
        cliSock = accept(srvSock, (SOCKADDR*)&cliAddr, &cliSz);

        //클라이언트의 정보 저장하기
        WaitForSingleObject(hmut, INFINITE);
        use[cliSz].Conclit = cliSock; // 클라이언트 소켓 저장
        use[cliSz].state = FALSE; // 클라이언트의 초기상태 : FALSE
        reev[cliSz].Conclit = use[cliSz].nickname, sizeof(use[cliSz].nickname), 0); //닉네임 받기 & 저장하기
        cliSz++;
        gotoxy(2, 1); GOLD; printf("cliCnt : %d\n", cliCnt); ORIGINAL;
        ReleaseMutex(hmut);

        scTime = (HANDLE)_beginthreadex(NULL, 0, timer, 0, 0, NULL);
        gotoxy(20, 2); HIGH_GREEN; printf("Connect Client ip: %s\n", inet_ntoa(cliAddr.sin_addr)); ORIGINAL;
        gotoxy(20, 3); BLUE_GREEN; printf("Connect Port: %d\n", cliAddr.sin_port); ORIGINAL;
        for (int i = 0; i < cliCnt; i++) {
            if (use[i].Conclit == cliSock) {
                gotoxy(20, 4); GOLD; printf("%s님이 접속하였습니다.\n", use[i].nickname); ORIGINAL;
                break;
            }
        }

        //HandleClient(&cliSock);
        hThread = (HANDLE)_beginthreadex(NULL, 0, HandleClient, (void*)&cliSock, 0, NULL);
        //WaitForSingleObject(hThread, INFINITE);
    }

    closesocket(srvSock);
    WSACleanup();
    return 0;
}
```

[그림 9. server - tcp/ip 프로그래밍 코드]

(2) client - MultiSocket()

```
void MultiSocket()
{
    system("cls");
    SOCKET sock;
    WSADATA wsa;
    SOCKADDR_IN srv_addr = { 0, };
    char srvIp[30] = { 0, }, srvPort[10] = { 0, }, nick[NAME_SIZE] = { 0, };

    printf("Input server IP: ");
    scanf("%[^\n]s", srvIp);
    getchar();
    printf("Input server Port: ");
    scanf("%[^\n]s", srvPort);
    getchar();

    //1. WSAStartup
    if (WSAStartup(MAKEWORD(2, 2), &wsa) == -1) {
        printf("WSAStartup error\n");
        return;
    }

    //2. socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        printf("socket error\n");
        return;
    }

    //2. save server's address
    srv_addr.sin_family = AF_INET;
    srv_addr.sin_addr.s_addr = inet_addr(srvIp);
    srv_addr.sin_port = htons(atoi(srvPort));

    //3. connect
    if (connect(sock, (SOCKADDR*)&srv_addr, sizeof(srv_addr)) == -1) {
        //printf("It is incorrect to input data\n");
        printf("disconnected\n");
        system("pause");
        return;
    }
    printf("connected\n");

    hSema = CreateSemaphore(NULL, 1, 1, NULL);

    //4. 닉네임
    printf("Input nickname: ");
    scanf("%[^\n]s", nick);
    getchar();
    sprintf(name, "[%s]", nick); // 닉네임 저장하기
    send(sock, name, strlen(name), 0); // 서버에게 닉네임 보내기

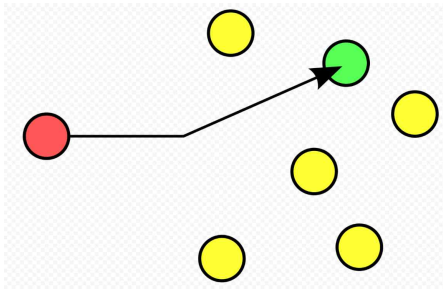
    while (1)
    {
        system("cls");
        gotoxy(22, 2); GOLD; printf("[ 대기방에 들어왔습니다. ]\n"); ORIGINAL;
        chatCheck = 0;
        sendThread = (HANDLE)_beginthreadex(NULL, 0, SendMsg, (void*)&sock, 0, NULL);
        recvThread = (HANDLE)_beginthreadex(NULL, 0, RecvMsg, (void*)&sock, 0, NULL);
        WaitForSingleObject(sendThread, INFINITE); // 이것을 쓰지 않으면 시작화면으로 돌아간다
        WaitForSingleObject(recvThread, INFINITE);
        if (chatCheck == 2) { // 시작신호를 받았다면
            mtThread1 = (HANDLE)_beginthreadex(NULL, 0, CallMulti, (void*)&sock, 0, NULL);
            WaitForSingleObject(mtThread1, INFINITE);
            continue;
        }
        else{
            break;
        }
    }

    system("cls");
    gotoxy(20, 1); GOLD; printf("시작화면으로 돌아옵니다.\n"); ORIGINAL;
    system("pause");
    CloseHandle(hSema);
    closesocket(sock);
    WSACleanup();
    return;
}
```

1. WSAStartup : Winsock 초기화
 - Winsock 동적 연결 라이브러리를 초기화하고 Winsock 구현이 애플리케이션 요구사항을 충족하는 확인
2. SOCKET : 소켓 생성
 - socket(AF_INET, SOCK_STREAM, 0);
 - AF_INET : IPv4(주소체계)
 - SOCK_STREAM : TCP 사용
 - 0 : 프로토콜
3. save server's address
 - 신호 요청을 받을 서버의 주소체계, IP주소, 포트번호를 지정한다.
4. connect : 서버에게 연결 요청을 한다.
5. closesocket : 소켓 해제
6. WSACleanup : Winsock 해제

[그림 10. client - MultiSocket()]

3) Unicast 통신 : 1:1 관계로 통신



[그림 11. Unicast]

```
//2. SOCKET
srvSock = socket(AF_INET, SOCK_STREAM, 0);

//3. bind
memset(&srvAddr, 0, sizeof(srvAddr));
srvAddr.sin_family = AF_INET;
srvAddr.sin_addr.s_addr = htonl(INADDR_ANY);
srvAddr.sin_port = htons(12345);

if (bind(srvSock, (SOCKADDR*)&srvAddr, sizeof(srvAddr)) == SOCKET_ERROR)
    printf("bind Error\n");
```

[그림 12. server - Unicast]

```
//2. socket
sock = socket(AF_INET, SOCK_STREAM, 0);
if (sock == -1) {
    printf("socket error\n");
    return;
}

//2. save server's address
srv_addr.sin_family = AF_INET;
srv_addr.sin_addr.s_addr = inet_addr(srvIp);
srv_addr.sin_port = htons(atoi(srvPort));

//3. connect
if (connect(sock, (SOCKADDR*)&srv_addr, sizeof(srv_addr)) == -1) {
    //printf("It is incorrect to input data\n");
    printf("disconnected\n");
    system("pause");
    return;
}
printf("connected\n");
```

[그림 13. client - Unicast]

TCP 연결 타입에 1:1로 통신하도록 구현했습니다.

유니캐스트를 이용한 이유는 단지 TCP를 사용하게 되니 멀티캐스트, 브로드캐스트를 제외한 유니캐스트로 선택이 제한되었습니다.

유니캐스트 통신이어서 UDP 연결 타입, 멀티캐스트 그룹 가입 및 탈퇴 관련 코드가 없습니다.

예)

```
socket(AF_INET, SOCK_DGRAM, 0);
setsockopt();
```

4) Thread (1) server

```
HANDLE handler, hmut, check, startsignal, hThread, recvHandler, scTime;
//handler : 소켓 생성스레드
//hmut : 뮤텍스
//check : 클라이언트 대기상태 확인
unsigned WINAPI HandleClient(void* arg);
```

[그림 14. server - HANDLE 전역변수로 선언]

```
unsigned WINAPI HandleClient(void* arg) {
    SOCKET cktSock = *((SOCKET*)arg);
    int strLen, i;
    char msg[BUF_SIZE] = { 0, }, clientout[BUF_SIZE] = { 0, }, gOverSig[5];

    while ((strLen = recv(cktSock, msg, sizeof(msg), 0)) != 0) {
        if (strcmp(msg, "q\n")) {
            send(cktSock, msg, strlen(msg)+1, 0); // 종료신호는 함수쓰지 않고 즉시 보내기
            break;
        }
        else if (strcmp(msg, "r\n")) {
            for (i = 0; i < cktCnt; i++) {
                if (use[i].Conclt == cktSock) {
                    printf("%s님의 준비신호를 받았습니.", use[i].nickname);
                    use[i].state = TRUE; // 클라이언트 상태변경 : false -> true
                    readyCheck(&cktSock);
                }
            }
        }
    }
}
```

```
else if (strcmp(msg, "end")) { // 게임오버 신호를 받았다면
    char buf[3] = { 0, };
    for (int i = 0; i < cktCnt; i++) {
        if (use[i].Conclt == cktSock) {
            use[i].se = score;
            use[i].rank = cktCnt - Rank;
            //out = (char)use[i].rank;
            printf("%s님의 점수는 %d이고 순위는 %d입니다. \n", use[i].nickname, use[i].sc, use[i].rank);
            //send(use[i].Conclt, use[i].rank, sizeof(use[i].rank), 0);
            use[i].state = FALSE;
        }
        if ((cktCnt - Rank) == 1) {
            printf("모두 끝났습니다\n");
            ready = FALSE;
            SendMsg(gOverSig, sizeof(gOverSig)); // 모든 클라이언트가 게임오버임을 알리기
            RankStore();
            endsend();
            reset();
        }
        //마지막 유저까지 반환이 진행이되면 ready 신호를 초기화 해준다.
        Rank++;
    }
    //클라이언트 종료 신호를 받고 각 클라이언트의 점수를 저장한다.
} else {
    SendMsg(msg, strlen(msg) + 1);
    memset(msg, 0, sizeof(msg));
}
```

```
//종료할 경우, 클라이언트 소켓들을 담은 배열 갱신하기!☆☆☆☆☆
//WaitForSingleObject(hmut, INFINITE);
if (strcmp(msg, "q\n")) {
    gotoxy(20, 5); GRAY; printf("Client left the chat\n"); ORIGINAL;
    for (i = 0; i < cktCnt; i++) {
        if (cktSock == use[i].Conclt) {
            sprintf(clientout, "%s 사용자 로그아웃\n", use[i].nickname);
            gotoxy(20, 7); SKY_BLUE; printf("%s", clientout); ORIGINAL;
            SendMsg(clientout, strlen(clientout) + 1);
            while (i++ < cktCnt - 1) {
                //초기화☆☆☆☆☆
                memset(&use[i].Conclt, 0, sizeof(use[i].Conclt));
                memset(&use[i].nickname, 0, sizeof(use[i].nickname));
                memset(&use[i].state, 0, sizeof(use[i].state));

                //대입하기
                use[i].Conclt = use[i + 1].Conclt;
                strcpy(use[i].nickname, use[i + 1].nickname);
                use[i].state = use[i + 1].state;
            }
            break;
        }
    }
    cktCnt--;
    if (cktCnt > 0) {
        gotoxy(18, 8); YELLOW; printf("●●●●● = 접속자 = ●●●●●\n"); ORIGINAL;
        for (i = 0; i < cktCnt; i++) {
            if (use[i].Conclt != 0)
                gotoxy(20, 9); GOLD; printf("%d. %s\n", i, use[i].nickname); ORIGINAL;
        }
    }
    else {
        gotoxy(18, 8); YELLOW; printf("●●●●● = 접속자 = ●●●●●\n"); ORIGINAL;
        gotoxy(20, 9); GOLD; printf(" none\n"); ORIGINAL;
    }
}
//ReleaseMutex(hmut); // 뮤텍스 중지
closesocket(cktSock);
return 0;
```

[그림 15. server - 전역변수로 선언]

```
//HandleClient(&cktSock);
hThread = (HANDLE)_beginthreadex(NULL, 0, HandleClient, (void*)&cktSock, 0, NULL);
//WaitForSingleObject(hThread, INFINITE);
```

[그림 16. server - 스레드함수 호출]

MultiSocket() 함수에서

accept() 함수로 클라이언트의 요청이 있을 때마다 매개변수를 클라이언트 소켓으로 전달하면서 HandleClient 스레드 함수를 호출하게 되고 진행된다.

recv(), send() 함수로 채팅을 구현하였습니다.

첫 번째, 클라이언트에게서 "q\n" 값 (종료신호)을 받는다면 다른 클라이언트들에게 해당 클라이언트의 로그아웃을 알리고 클라이언트 소켓들을 저장하는 배열의 정보를 갱신합니다.

두 번째, 클라이언트에게서 "r\n"값(준비신호)을 받는다면 해당 클라이언트의 상태정보를 false에서 true로 변경해주고 접속한 클라이언트들이 모두 준비신호를 받았는지 확인합니다.

아직 준비신호를 다 받지 않았다면 준비신호를 보낸 클라이언트는 대기합니다.

준비신호를 다 받았다면 서버는 ready 변수의 값을 false에서 true로 변경합니다.

ready 변수가 true인 경우, 접속한 클라이언트 전부에게 게임 시작신호를 보내어, 모든 클라이언트들이 게임을 시작할 수 있도록 하게 합니다.

(2) client

```
HANDLE sendThread, recvThread, mtThread1, hSema;
```

[그림 17. client - HANDLE 전역변수로 선언]

```
//4. 닉네임
printf("input nickname: ");
scanf("%s", nick);
getchar();
sprintf(name, "[%s]", nick); // 닉네임 저장하기
send(sock, name, strlen(name), 0); // 서버에게 닉네임 보내기

while (1)
{
    system("cls");
    gotoxy(22, 2); GOLD; printf("[ 대기방에 들어왔습니다. ]\n"); ORIGINAL;
    chatCheck = 0;
    sendThread = (HANDLE)_beginthreadex(NULL, 0, SendMsg, (void*)&sock, 0, NULL);
    recvThread = (HANDLE)_beginthreadex(NULL, 0, RecvMsg, (void*)&sock, 0, NULL);
    WaitForSingleObject(sendThread, INFINITE); // 이것을 쓰지 않으면 시작화면으로 돌아간다
    WaitForSingleObject(recvThread, INFINITE);
    if (chatCheck == 2) { // 시작신호를 받았다면

        mtThread1 = (HANDLE)_beginthreadex(NULL, 0, CallMulti, (void*)&sock, 0, NULL);
        WaitForSingleObject(mtThread1, INFINITE);
        continue;
    }
    else{
        break;
    }
}
```

[그림 18. client - 스레드 함수 호출]

```
unsigned WINAPI SendMsg(void* arg) //전송용 스레드함수
{
    SOCKET sock = *((SOCKET*)arg);
    char nameMsg[NAME_SIZE + BUF_SIZE] = { 0, };

    while (1)
    {
        fgets(msg, BUF_SIZE, stdin);
        if (!strcmp(msg, "q\n")) { // q or Q를 입력하면 종료
            send(sock, "q\n", 3, 0);
            break;
        }
        else if (!strcmp(msg, "Q\n")) {
            send(sock, "Q\n", 3, 0);
            break;
        }
        else if (!strcmp(msg, "r\n")) { // r 입력하면 대기
            send(sock, "r\n", 3, 0);
            system("cls");
            printf("게임 대기 중...★★★");
            break;
        }
        sprintf(nameMsg, "%s %s", name, msg);
        send(sock, nameMsg, strlen(nameMsg), 0); //nameMsg를 서버에게 전송한다
        memset(msg, 0, sizeof(msg));
        memset(nameMsg, 0, sizeof(nameMsg));
    }

    return 0;
}
```

[그림 19. client - SendMsg()]

클라이언트에서는 닉네임을 서버에게 보낸 후, SendMsg(), RecvMsg() 스레드 함수를 호출하여 클라이언트가 대기 및 채팅창에 들어온 것처럼 입력한 메시지를 서버에게 보내고 서버에서 보내는 메시지를 받습니다.

SendMsg() 스레드 함수에서

첫 번째, "q\n" 또는 "Q\n"를 입력한 경우 닉네임을 붙이지 않고 바로 보냅니다.

두 번째, "r\n"를 입력한 경우에도 닉네임을 붙이지 않고 바로 보냅니다.

세 번째, 일반 메시지를 입력한 경우 sprintf로 닉네임과 메시지를 붙이고 서버에게 보냅니다.

첫 번째와 두 번째의 경우, 닉네임을 배제하고 보내기 때문에 특별 신호를 인식하기 쉽도록 구현되었습니다.

세 번째의 경우, 서버에서 닉네임+메시지를 받고 echo로 모든 클라이언트에게 받은 메시지를 다시 보내어 어떤 클라이언트가 입력한 메시지인지 닉네임으로 확인할 수 있습니다.

```
//4. 닉네임
printf("input nickname: ");
scanf("%[^\n]s", nick);
getchar();
sprintf(name, "[%s]", nick); // 닉네임 저장하기
send(sock, name, strlen(name), 0); // 서버에게 닉네임 보내기

while (1)
{
    system("cls");
    gotoxy(22, 2); GOLD; printf("[ 대기방에 들어왔습니다. ]\n"); ORIGINAL;
    chatCheck = 0;
    sendThread = (HANDLE)_beginthreadex(NULL, 0, SendMsg, (void*)&sock, 0, NULL);
    recvThread = (HANDLE)_beginthreadex(NULL, 0, RecvMsg, (void*)&sock, 0, NULL);
    WaitForSingleObject(sendThread, INFINITE); // 이것을 쓰지 않으면 시작화면으로 돌아간다
    WaitForSingleObject(recvThread, INFINITE);
    if (chatCheck == 2) { // 시작신호를 받았다면

        mtThread1 = (HANDLE)_beginthreadex(NULL, 0, CallMulti, (void*)&sock, 0, NULL);
        WaitForSingleObject(mtThread1, INFINITE);
        continue;
    }
    else{
        break;
    }
}
```

[그림 18. client - 스레드 함수 호출]

```
unsigned WINAPI RecvMsg(void* arg) // 수신용 스레드함수
{
    SOCKET sock = *((SOCKET*)arg);
    char nameMsg[NAME_SIZE + BUF_SIZE] = { 0, };

    int strLen;

    while (1)
    {
        strLen = recv(sock, nameMsg, NAME_SIZE + BUF_SIZE - 1, 0); //서버로부터 메시지를 수신한다.
        if (strLen == -1)
            break;
        nameMsg[strLen] = 0; //문자열의 끝을 알리기 위해 설정
        if (strcmp(nameMsg, "q\n") || strcmp(nameMsg, "Q\n")) {
            printf("멀티모드를 종료합니다.\n");
            closesocket(sock);
            chatCheck = 4;
            break;
        }
        if (strcmp(nameMsg, "start")) { // 시작신호를 받았다면
            printf("시작신호를 받았습니다.\n");
            chatCheck = 2;
            break;
        }
        if (strcmp(nameMsg, name, strlen(name)))
            continue;
        fputs(nameMsg, stdout); //자신의 콘솔에 받은 메시지를 출력한다.
    }
    return 0;
}
```

[그림 20. client - RecvMsg()]

```
unsigned WINAPI CallMulti(void* arg)
{
    SOCKET sock = *((SOCKET*)arg);

    WaitForSingleObject(hSema, INFINITE);

    system("cls");
    for (int i = 0; i < 3; i++) {
        gotoxy(25, 15); YELLOW; printf("게임 시작까지 %d초", 3 - i); ORIGINAL;
        gotoxy(20, 13); YELLOW; printf("*****"); ORIGINAL;
        gotoxy(20, 14); YELLOW; printf("★"); ORIGINAL;
        gotoxy(20, 15); YELLOW; printf("★★★"); ORIGINAL;
        gotoxy(35, 15); YELLOW; printf("★★★★"); ORIGINAL;
        gotoxy(20, 16); YELLOW; printf("★"); ORIGINAL;
        gotoxy(20, 17); YELLOW; printf("*****"); ORIGINAL;
        Sleep(1000);
    }
    MultiGame(); // 멀티모드 게임함수, 플레이어의 플레이시간을 반환해준다
    SendEnd(sock); // 게임끝났다고 알리기 & 대기 중 각자의 순위 출력
    RecvSaveRank(sock); // 랭킹정보 받기
    MultiRankDraw(); // 멀티 랭킹정보 출력

    ReleaseSemaphore(hSema, 1, NULL);
    return 0;
}
```

[그림 21. client - CallMulti()]

RecvMsg() 스레드 함수에서

첫 번째, 반환값이 -1인 경우 while문 종료
두 번째, "q\n" 또는 "Q\n" 값을 받은 경우
해당 문구를 출력한 후 소켓 해제하고
chatCheck 값을 4로 대입하고 끝납니다.

세 번째, "start"값을 받은 경우 해당 문구를
출력한 후 chatCheck 값을 2로 대입하고
끝납니다.

세 번째, 일반 메시지를 받은 경우
nameMsg(닉네임)이 자신의 닉네임과 일치한
지 확인하고 동일하다면 출력하지 않도록
했습니다.

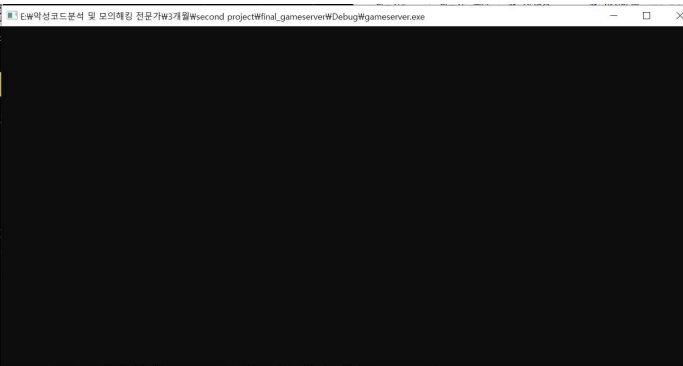
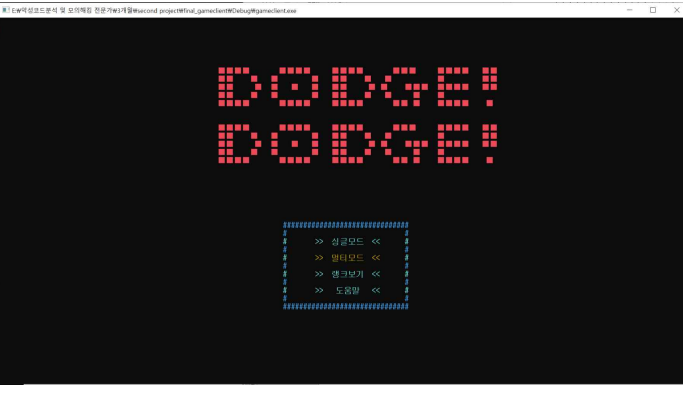
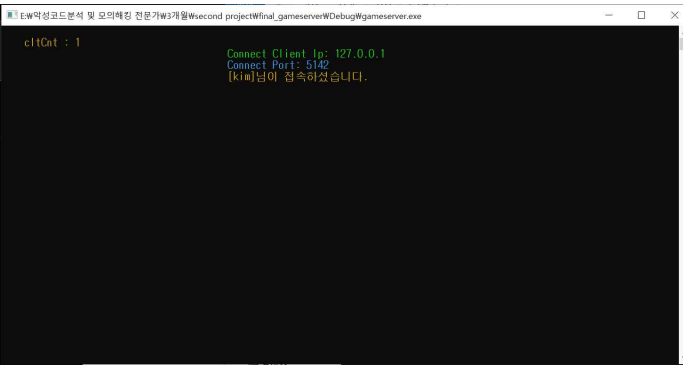
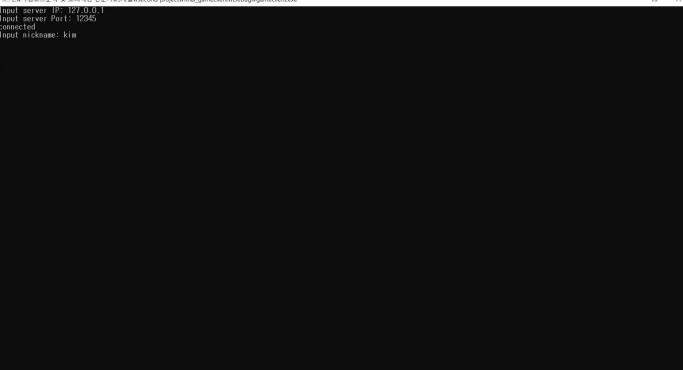
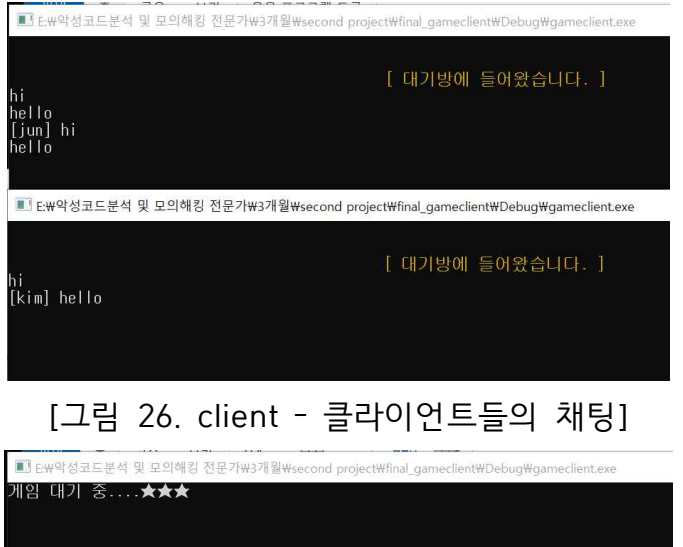
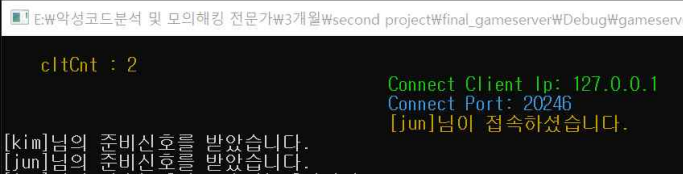

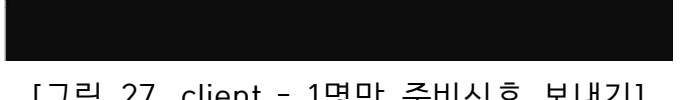
chatCheck값이

2인 경우, CallMulti() 스레드 함수를 호출하며
그 외의 chatCheck 값인 경우, 멀티모드가
끝나게 됩니다.

CallMulti() 스레드 함수에서

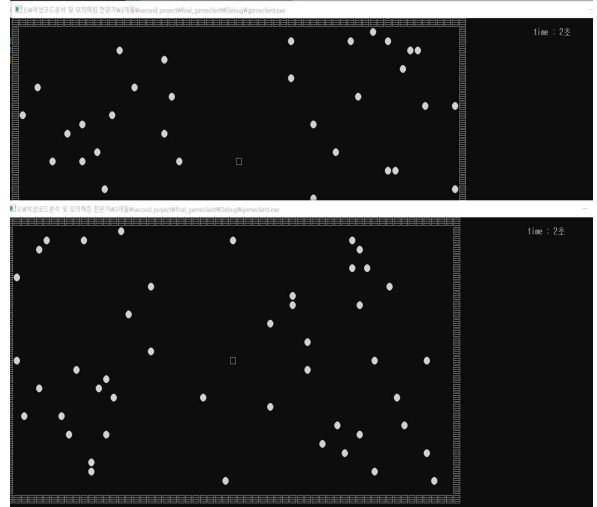
3초간 대기하는 화면을 보여주고
멀티모드 게임을 시작하게 했습니다.

3. 프로그램 시현

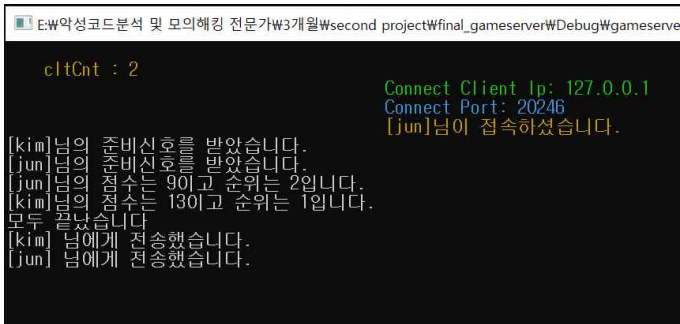
| server | client |
|--|--|
|  |  |
| <p>[그림 22. server - 소켓 연결 대기]</p> | <p>[그림 23. client - 시작화면]</p> |
|  |  |
| <p>[그림 25. server - 클라이언트1 접속]</p> | <p>[그림 24. client - 연결을 위한 서버정보 입력]</p> |
| |  |
|  |  |
| <p>[그림 28. server - 접속신호 모두 받음]</p> | <p>[그림 26. client - 클라이언트들의 채팅]</p> |
| |  |
| | <p>[그림 27. client - 1명만 준비신호 보내기]</p> |

server

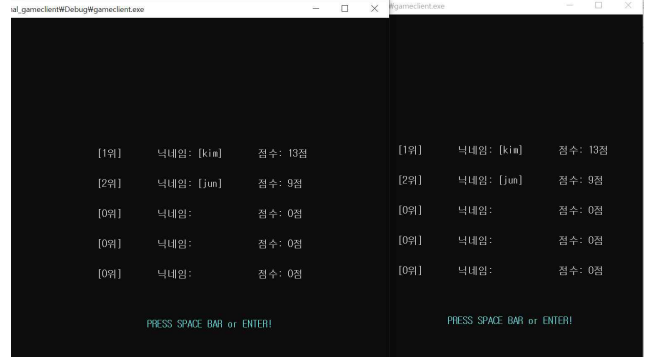
client



[그림 29. client - 클라이언트 동시게임시작]



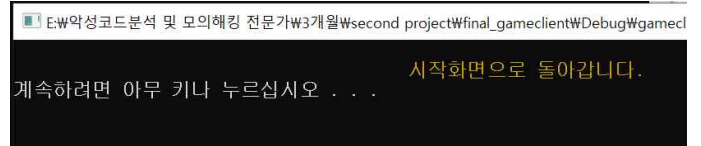
[그림 30. server - 클라이언트 모두 게임오버]



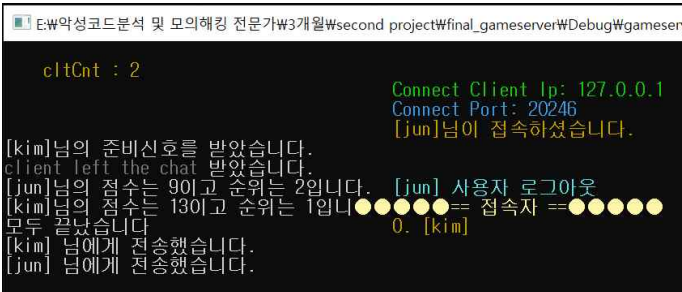
[그림 31. client - 멀티랭킹 출력]



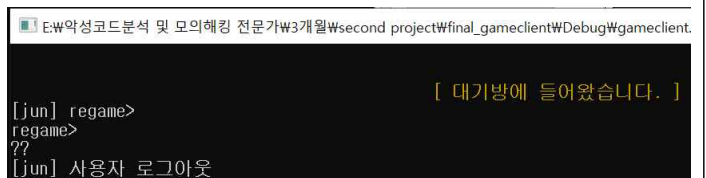
[그림 32. client - 다시 채팅방으로 돌아옴]



[그림 33. client - q를 입력하여 종료]



[그림 34. server - 클라이언트 1명 종료]



[그림 35. client - 1명 종료한 것을 확인]

4. 소스 코드 최적화 및 결과 보고서 작성

```
출력 보기 선택(S): 빌드
1>----- 빌드 시작: 프로젝트: gameserver, 구성: Debug Win32 -----
1>gameserver.vcxproj -> E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameserver\Debug\gameserver.exe
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
```

[그림 36. server]

```
468 {
469     Sleep(40);
470     DelEndBall();
471     srand((int)malloc(NULL));
472     CreatePooss();
473     CreatePoogg();
474     srand((int)malloc(NULL));
475     CreatePoog();
476     CreatePoos();
477     DeletePoo();
478     DeletePoo2();
479     MovePoo2();
480     PrintMap2();
481     MovePlayer();
482     MovePoo();
483     PrintMap();
484 }
```

```
출력 보기 선택(S): 빌드
1>----- 빌드 시작: 프로젝트: gameclient, 구성: Debug Win32 -----
1>game.c
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(436,34): warning C4047: '함수': 'size_t'의 간접 참조 수준이 'void *'과(와) 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(436,34): warning C4024: 'malloc': 형식 및 실제 매개 변수 1의 형식이 서로 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(439,34): warning C4047: '함수': 'size_t'의 간접 참조 수준이 'void *'과(와) 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(439,34): warning C4024: 'malloc': 형식 및 실제 매개 변수 1의 형식이 서로 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(471,25): warning C4047: '함수': 'size_t'의 간접 참조 수준이 'void *'과(와) 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(471,25): warning C4024: 'malloc': 형식 및 실제 매개 변수 1의 형식이 서로 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(474,25): warning C4047: '함수': 'size_t'의 간접 참조 수준이 'void *'과(와) 다릅니다.
1>E:\약성코드분석 및 모의해킹 전문가\3개월\second project\final_gameclient\gameclient\game.c(474,25): warning C4024: 'malloc': 형식 및 실제 매개 변수 1의 형식이 서로 다릅니다.
1>"gameclient.vcxproj" 프로젝트를 빌드했습니다.
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
```

[그림 37. client]

- 서버는 문제가 없었으나, 클라이언트에서 `srand((int)malloc(NULL));` 부분에서 warning이 뜨는 것을 확인할 수 있었습니다. 하지만 이 부분을 다른 코드로 대체하면 게임이 전보다 난도가 낮아져서 심심하여 그대로 두었습니다.
- 이 프로젝트를 마치며
: 3주간 프로젝트를 하면서 OpenGL도 해보고 게임 동시화면출력도 test를 해보았습니다. OpenGL인 경우 그래픽 출력 창과 텍스트 출력 창이 따로 생성되기 때문에 게임 출력하는데 어려움이 있을 것으로 생각하여 포기했습니다.
게임 동시화면출력은 서버에서 접속한 클라이언트들에게 움직이는 공의 무작위 값을 전달하여 클라이언트 화면에서 공의 출력까지 완료하였습니다. 하지만 당시에 전체 클라이언트의 게임 오버 후 전체 멀티랭킹을 출력하는 데 어려움이 생겨 그곳에 비중을 많이 두어 포기했습니다. 학원에서 시험을 하니 같은 호스트에서 여러 대의 클라이언트를 하는 것과 다른 호스트에 1대, 클라이언트 1명씩 시험하는 결과가 다르다는 것을 마지막 날에 알게 되어 안타까웠습니다.