1. 게임에서 설계한 부분

소켓 프로그래밍으로 숫자 야구 게임을 구현해보았다. 게임은 서버의 initGame 함수와 check 함수를 통해 이루어진다. initGame은 게임을 초기화하는 부분이다. 0부터 9까지 수 중에 세 개의 숫자를 랜덤으로 선택하여 snum이라는 배열에 저장한다. check 함수는 클라이언트로부터 받아온 세 숫자를 서버의 숫자와 비교하여 strike와 ball을 계산하는 역할을 한다. 두 배열의 원소를 각각 비교하여 숫자와 위치(인덱스)가 모두 같다면 strike, 숫자는 같으나 위치가 다르다면 ball로 간주한다. Strike가 3이라면 클라이언트가 숫자를 모두 맞춰 한 판의 게임이 끝나는 경우이다. 이때 현재 점수와 클라이언트의 최고 점수를 비교하여 현재 점수가 더 높다면 최고 점수를 갱신한다.

점수의 계산 방법은 클라이언트가 정답을 맞추기까지 시도한 횟수를 이용한다. 게임 한 판에서 획득할 수 있는 최고 점수는 950점으로 설정하였다. 즉 클라이언트가 1번 만에 3 strike로 모든 숫자를 맞췄다면 950점을 받게 된다. 횟수가 늘어날수록 50점씩 줄어드는 방식으로 점수를 계산하였다. 예를 들어 3번 만에 맞췄다면 850점, 6번 만에 맞췄다면 700점이다. 한 판의 게임이 종료될 때 해당 판의 점수와 클라이언트의 최고 점수를 출력해 준다.

서버와 클라이언트 간에 데이터를 주고받을 때는 배열을 이용한다. 클라이언트는 서버에 크기가 4인 배열 cnum을 넘겨준다. 이는 클라이언트가 추측한 숫자 3개와 게임 한 판이 끝났을 때 이어서 게임을 새로 할 것인지를 결정하는 숫자(1또는 2)를 저장하고 있다. 서버는 클라이언트에 크기가 5인 배열을 넘겨준다. 여기에는 클라이언트의 숫자와 서버의 숫자를 비교한 결과인 strike, ball의 개수, 현재 점수, 최고 점수, 시도 횟수(클라이언트가 게임을 몇 판 하고 있는지)를 저장하고 있다.

게임 한 판이 끝나면 클라이언트는 게임을 이어서 새로 할지 종료할지를 정할 수 있다. Y를 입력하면 cnum[3]의 값을 1로 하여 cnum 배열을 서버로 넘겨준다. 서버는 cnum[3]의 값이 1인 것을 확인하면 initGame 함수를 실행하여 새로운 게임을 시작한다. 이때 cnum[3]의 값은 0으로 다시 초기화된다. 클라이언트가 n를 입력한다면 cnum[3]의 값을 2로 하고 cnum 배열을 서버에 보내준다. 그리고 반복문을 빠져나와 클라이언트 프로그램은 종료된다. 서버는 클라이언트로부터 받은 cnum[3]이 2인 것을 확인하면 현재 클라이언트와의 접속을 종료한다.

2. 코드

- 서버 코드

```
#pragma comment(lib, "ws2_32.lib")
#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>
#include <time.h>
#define SERV_PORT 5000
#define MAXPENDING 5
```

```
int snum[3];
int currentscore = 0; // 현재 판에서의 점수
int maxscore = 0; // 클라이언트의 최대 점수
int trynum = 0; // 클라이언트의 게임 시도 횟수
int curtrynum = 0; // 현재 판에서 맞추기를 시도한 횟수

void initGame(int *trynum, int* snum) { // 게임 초기화
	*trynum += 1; // 총 시도 횟수 1회 증가
	currentscore = 0; // 현재 판의 점수 초기화
	curtrynum = 0; // 현재 판 안에서 시도 횟수 초기화
	srand((unsigned)time(NULL));
	for (int i = 0; i < 3; i++) { // 서버의 숫자 세 개 랜덤 생성
		snum[i] = rand() % 9;
		for (int j = 0; j < i; j++) {
			if (snum[j] == snum[i])
				i--;
		}
	}
	printf(" 서버의 숫자: %d %d %d₩n₩n", snum[0], snum[1], snum[2]);
}

void check(int *cnum, int *ball, int *strike) { // 결과 확인
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
			if ((snum[i] == cnum[j]) && (i == j)) // 값과 위치 모두 같을 때
				*strike += 1; // strike 값 증가
			if ((snum[i] == cnum[j]) && (i != j)) // 값은 같으나 위치가 다를 때
				*ball += 1; // ball 값 증가
		}
	}

	currentscore = 1000 - 50 * curtrynum; // 시도한 횟수가 늘어날수록 점수가 줄어들도록 함

	if (*strike == 3) {
		// 한 판의 게임이 끝났을 때 최고 점수와 현재 점수 비교
		if (maxscore < currentscore)
			maxscore = currentscore; // 현재 점수가 더 크다면 최고 점수 갱신
	}
}

int main(void) {
```

```
int servSock;
int clntSock;
struct sockaddr_in echoServAddr;
struct sockaddr_in echoClntAddr;
int clntLen;
char Buffer[256];
int maxLen = sizeof(Buffer);
int recvMsgSize;
WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        fprintf(stderr, "WSAStartup() failed₩n");
        exit(1);
}

/* Construct local address structure */
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = AF_INET;
echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);
echoServAddr.sin_port = htons(SERV_PORT);

/* Create socket for incoming connection */
if ((servSock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("Error: socket() failed₩n");
        exit(1);
}

/* Bind to the local address */
if (bind(servSock, (struct sockaddr *) &echoServAddr,
        sizeof(echoServAddr)) < 0) {
        printf("Error: bind() failed₩n");
        exit(1);
}

/* Mark the socket so it will listen for incoming connections */
if (listen(servSock, MAXPENDING) < 0) {
        printf("Error: listen() failed₩n");
        exit(1);
}

for (;;) { /* Run forever */
```

```
		/* Set the size of the in-out parameter */
		clntLen = sizeof(echoClntAddr);

		/* Wait for a client to connect */
		if ((clntSock = accept(servSock, (struct sockaddr *) &echoClntAddr, &clntLen)) < 0) {
			printf("Error: accept() failed₩n");
			exit(1);
		}

		initGame(&trynum, snum); // 게임 초기화

		while (1) {
			int cnum[4]; // 클라이언트로부터 받아온 숫자 3개 + 새로운 게임 시작 여부
			int result[5]; // 클라이언트에게 전달할 결과
			int ball = 0; // ball 개수
			int strike = 0; // strike 개수

			if ((recvMsgSize = recv(clntSock, (char *)cnum, sizeof(cnum), 0)) > 0) {
				for (int i = 0; i < 3; i++)
					cnum[i] = ntohl(cnum[i]); // 네트워크 바이트 숫자를 호스트 바이트로 바꿔줌

				if (cnum[3] == 1) { // 새로운 게임 시작
					printf("₩n₩n");
					initGame(&trynum, snum);
					continue;
				}
				if (cnum[3] == 2) { // 게임 종료 & 현재 클라이언트와 접속 종료
					printf("₩n₩n 현재 클라이언트와의 접속 종료₩n");
					break;
				}

				curtrynum++; // 현재 판에서의 맞추기 시도 횟수 증가

				printf(" 클라이언트의 입력 숫자 : %d  %d  %d₩n", cnum[0], cnum[1], cnum[2]);
				check(cnum, &ball, &strike);
				printf(" %d strike₩t%d ball₩n", strike, ball);

				// 클라이언트에게 전송하기 위해 데이터를 네트워크 바이트로
```

변환

```
                                        result[0] = htonl(strike);
                                        result[1] = htonl(ball);
                                        result[2] = htonl(currentscore);
                                        result[3] = htonl(maxscore);
                                        result[4] = htonl(trynum);
                                        send(clntSock, (char *)result, sizeof(result), 0); /* Send Result */
                                }
                                else
                                        printf("ERROR: Receiving error₩n");

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

- 클라이언트 코드

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define _CRT_SECURE_NO_WARNINGS
#pragma comment(lib, "ws2_32.lib")
#include <stdio.h>
#include <winsock2.h>
#include <stdlib.h>
#define SERV_PORT 5000

int main(int argc, char *argv[]) {
        int sock;
        struct sockaddr_in echoServAddr;
        char servIP[15];
        int bytesRcvd;
        WSADATA wsaData;

        if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
                fprintf(stderr, "WSAStartup() failed₩n");
                exit(1);
        }
        printf("₩n 서버의 IP 주소를 입력하세요 : ");
        scanf("%s", servIP);
```

```
/* Construct the server address structure */
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = AF_INET;
echoServAddr.sin_addr.s_addr = inet_addr(servIP);
echoServAddr.sin_port = htons(SERV_PORT);

/* Create a reliable, stream socket using TCP */
if ((sock = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        printf("ERROR: socket() failed₩n");
        exit(1);
}

/* Establish the connection to the echo server */
if (connect(sock, (struct sockaddr *)&echoServAddr, sizeof(echoServAddr)) < 0) {
        printf("ERROR: connect() failed₩n");
        exit(1);
}

printf("₩n₩t***숫자 야구 게임***₩n");

for (;;) { // 사용자가 종료를 원할 때까지 반복
        int num[4]; // 서버로 전달할 숫자 및 새로운 게임 시작 여부
        int result[5]; // 서버로부터 받아온 결과
        char restart; // 새로운 게임 싶가 여부
        num[3] = 0;
        printf("₩n 숫자 세 개를 입력하세요 : ");
        scanf("%d %d %d", &num[0], &num[1], &num[2]);

        // 서버에게 전송하기 위해 데이터를 네트워크 바이트로 변환
        for (int i = 0; i < 3; i++)
                num[i] = htonl(num[i]);

        /* Send the string, including the null terminator, to the server */
        send(sock, (char *)num, sizeof(num), 0);
        if ((bytesRcvd = recv(sock, (char *)result, sizeof(result), 0)) > 0) {
                // 서버로부터 받아온 네트워크 바이트 데이터를 호스트 바이트로 바꿔줌
                int strike = ntohl(result[0]);
                int ball = ntohl(result[1]);
                int currentscore = ntohl(result[2]);
                int maxscore = ntohl(result[3]);
```

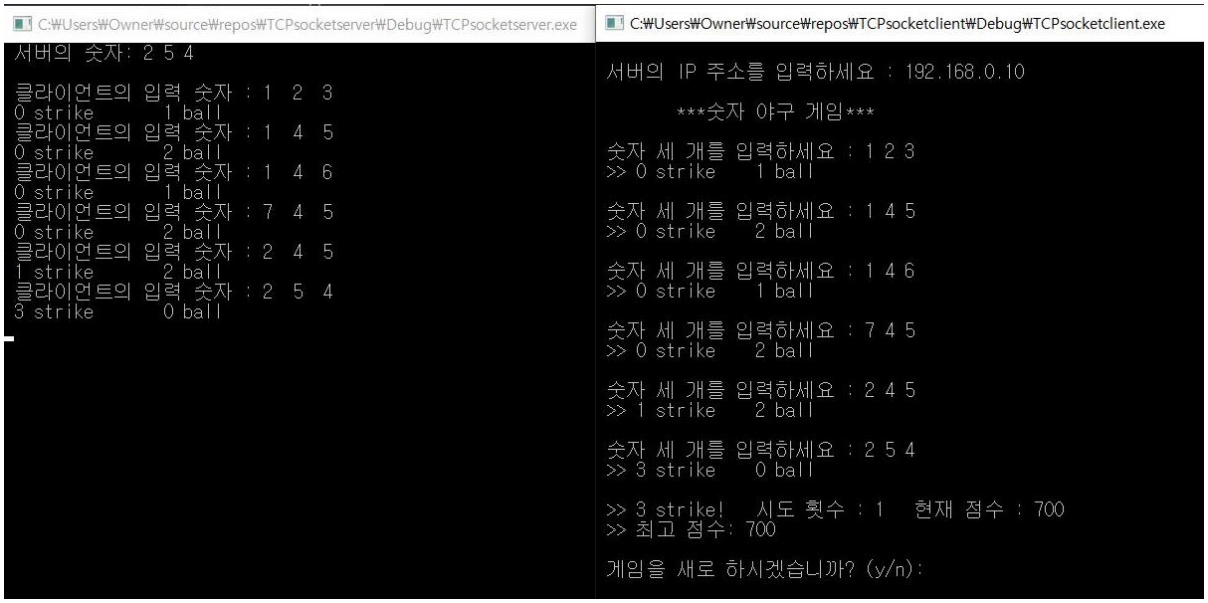
```
                        int trynum = ntohl(result[4]);

                        // 결과 출력
                        if (strike == 0 && ball == 0) printf("out₩n");
                        else printf(" >> %d strike₩t%d ball₩n", strike, ball);

;                       if (strike == 3) { // 사용자가 숫자를 모두 맞춰 게임 한 판이 끝났을 때
                                printf("₩n >> 3 strike!₩t시도 횟수 : %d₩t현재 점수 : %d₩n",
trynum, currentscore);
                                printf(" >> 최고 점수: %d₩n₩n", maxscore);
                                printf(" 게임을 새로 하시겠습니까? (y/n): ");
                                scanf(" %c", &restart);
                                if (restart == 'y') { // 새로운 게임 시작
                                        num[3] = 1;
                                        send(sock, (char *)num, sizeof(num), 0); // 서버에 새로운
게임 시작을 보냄
                                }
                                if (restart == 'n') { // 게임 종료
                                        num[3] = 2;
                                        send(sock, (char *)num, sizeof(num), 0); // 서버에 게임
종료를 보냄
                                        printf("게임을 종료합니다.₩n₩n");
                                        break;
                                }
                        }
                }
                else
                        printf("ERROR: Receiving echoed string error₩n");
        }

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

3. 게임 실행 화면

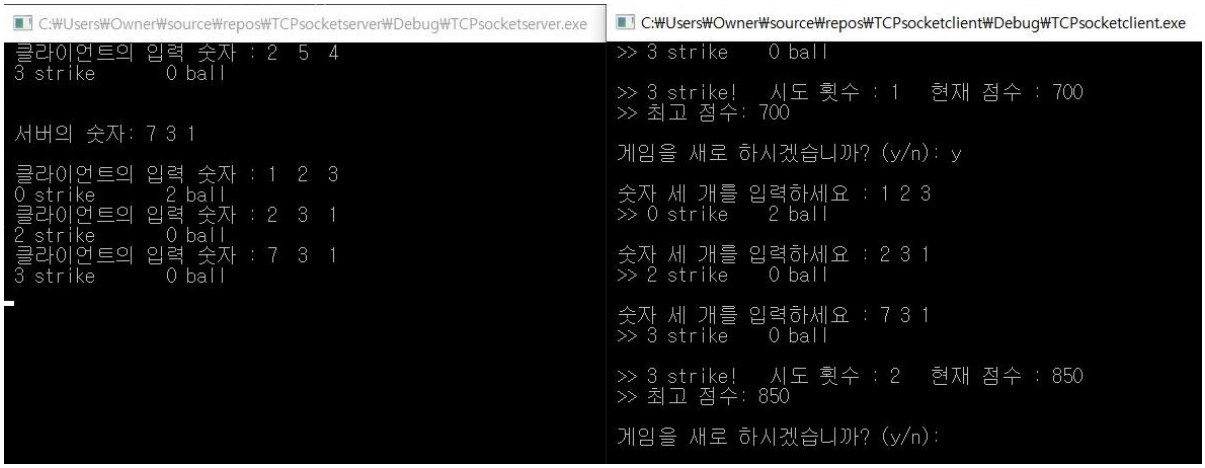
- 첫 실행 화면

- IP 주소 입력 후 게임 시작 화면

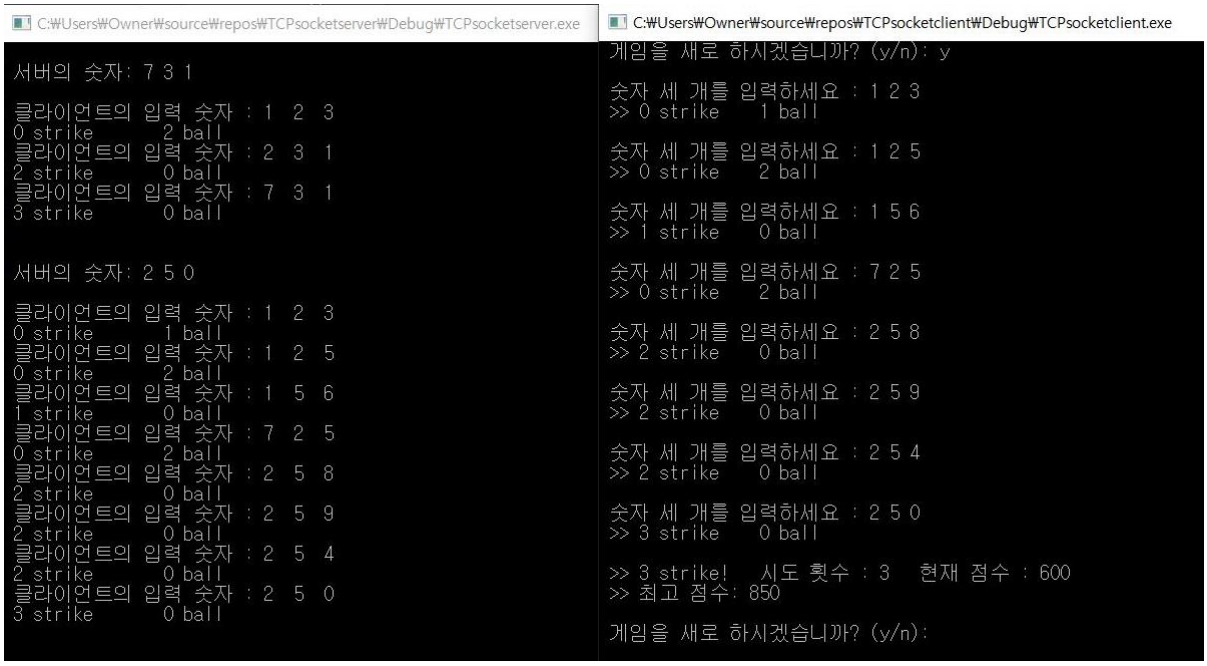
- 첫 번째 게임 수행



- y를 눌러 새로운(두번째) 게임 수행



- 세번째 게임 수행



- 게임 종료