P o r t f o l i o

김기윤

서 버 프 로 그 래 머

Phone. 010 - 5100 - 2974

E-mail. kimkiyoun33@gmail.com

Education. 한국공학대학교(구 한국산업기술대학교) 게임공학과 졸업예정(2023.02)

Birth. 1997. 09.10

Address. 경기도 의정부시

Skills

언어

- C
- C++
- python
- Lua Script
- java script
- react

클라이언트

- DirectX 12
- OpenGL
- Unity
- Unreal4

서버

- IOCP
- 멀티쓰레딩
- Ms SQL
- PostgreSQL
- Overlapped I/O
- Select Thread

Etc.

- github
- Google cloud platform

CONTENTS

SSU(졸업작품)

게임 장르	MMORPG
작업 기간	2022.01~2022.07
작업 인원	3명 - 클라이언트 1명 - 서버 2명
구현내용 (역할)	서버 프로그래머 - 몬스터 : 길찾기(A* 알고리즘), 로밍 - 파티 시스템 - Raid 보스 - 섹터 분할(최적화 및 동접 향상)
깃허브	https://github.com/hjs0913/SSU

파티시스템

01

●파티 객체의 상태

DUN_ST_FREE : 할당되지 않은 상태

DUN_ST_ROBBY : 로비 상태

DUN_ST_START : 레이드를 하고 있는 상태

●파티창 생성과 정보 제공

- 클라이언트에서 P키를 누르면 파티창 생성


```
dun->state_lock.lock();
if (dun->get_dun_st() == DUN_ST_ROBBY) {
    // 던전의 정보들을 보내준다
    dun->state_lock.unlock();
    send_party_room_packet(pl, dun->get_party_name(), dun->get_dungeon_id());
    continue;
}
```

- 방의 목록에서 방을 클릭하면 해당 방의 파티의 정보 제공
  (단 파티에 참가하고 있으면 안됨)


```
int r_id = reinterpret_cast<cs_packet_party_room_info_request*>(p)->room_id;
Gaia* dun = m_ObjectManger->get_dungeon(r_id);
send_party_room_info_packet(pl, dun->get_party_palyer(),
    dun->player_cnt, dun->get_dungeon_id());
```

파티시스템

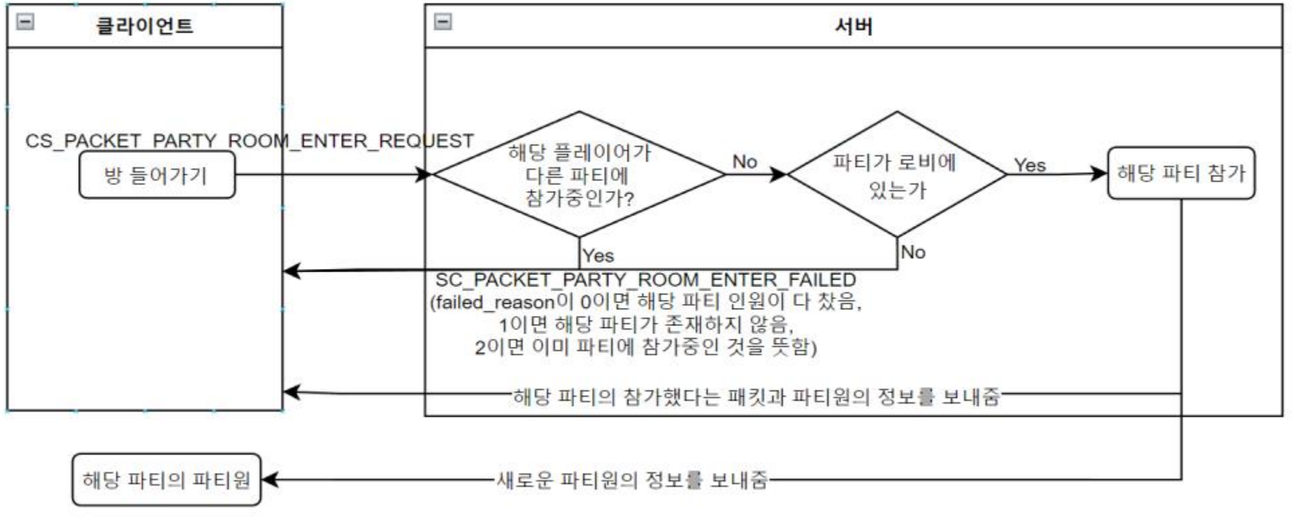
● UI버튼과 상호작용

- 방만들기

파티 객체 중 상태가 DUN_ST_FREE인 객체를 사용

```
dun->state_lock.lock();
if (dun->get_dun_st() == DUN_ST_FREE) {
    dun->set_dun_st(DUN_ST_ROBBY);
    dun->state_lock.unlock();

    // 이 방에 이 플레이어를 집어 넣는다
    dun->set_party_name(pl->get_name());
    dun->join_player(pl);
```

- 방들어가기



- AI넣기

인원의 여유가 있는지 검사 후 AI추가

Lv은 AI를 추가한 플레이어의 Lv에 맞춤

파트너는 플레이어의 객체랑 같이 관리

(Index의 Id로 범위가 나누어져 있다)

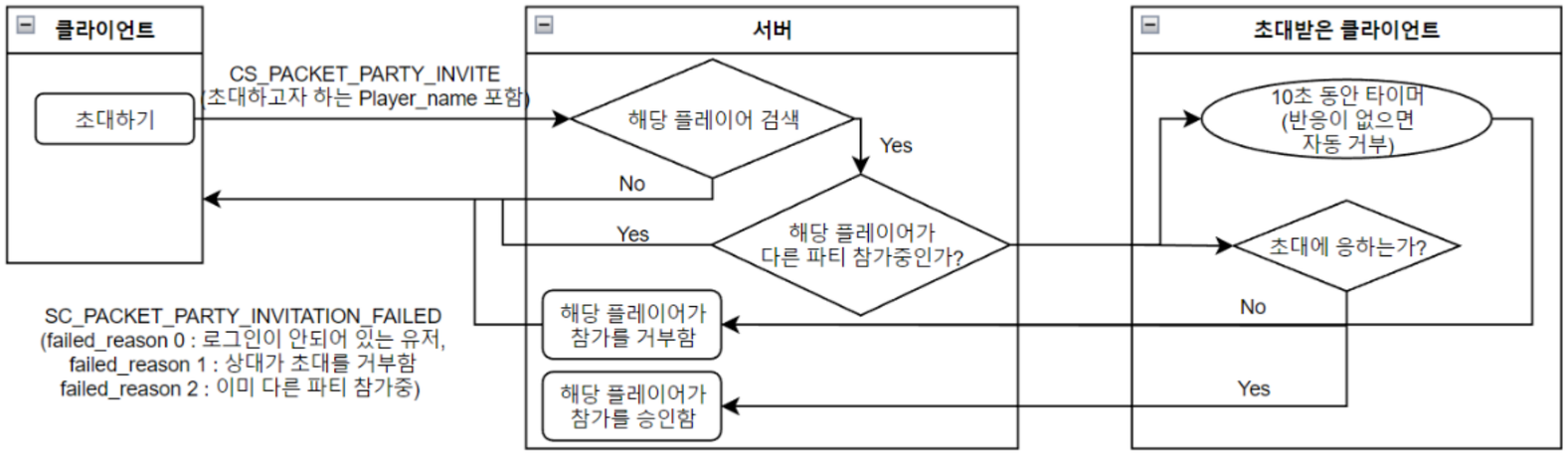
```
int new_id = m_ObjectManger->get_new_ai_id();
if (-1 == new_id) {
    cout << "Maxmum user overflow.Accept aborted.\n";
}
players[new_id]->state_lock.lock();
players[new_id]->set_state(ST_ACCEPT);
players[new_id]->state_lock.unlock();

Partner* partner = reinterpret_cast<Partner*>(players[new_id]);
partner->set_tribe(PARTNER);
```

이후 플레이어랑 똑같이 처리

파티시스템

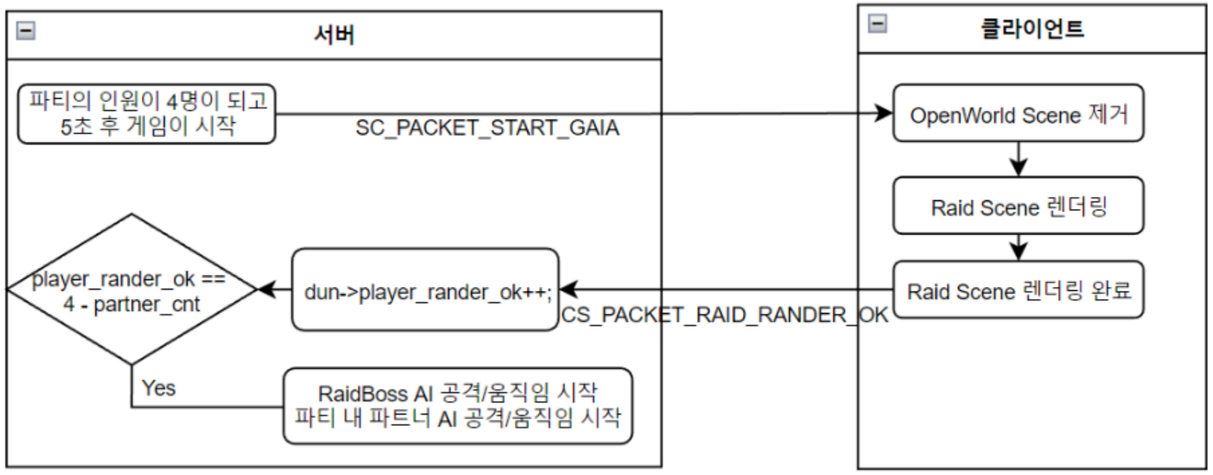
01

● UI버튼과 상호작용(초대하기)



- 초대에 응하면 '방들어가기'랑 똑같은 방식으로 처리

● 게임시작



- 맨 처음 인원이 4명이 되면 클라이언트가 OpenWorld Scene에서 Raid Scene으로 씬을 전환하기 전에 AI들이 먼저 움직이는 문제점을 발견

해결방안 : 파티원 클라이언트가 렌더링을 완료하면
CS_PACKET_RAID_RANDER_OK패킷을 보내주고
서버에서는 player_rander_ok++를 해준다
'player_rander_ok == 4 - 파트너의 수'이면 AI를 움직이기 시작한다

Raid 보스 AI

01

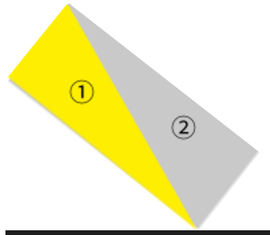
● Lua스크립트

- Lua스크립트를 통해 보스의 스탯, 정보 쉽게 조정

```
my_id = 99999;
my_element = 0;
my_lv = 100;
my_name = "가이아";
--my_hp = 5100000;
my_hp = 1000000;
my_physical_attack = 350;
my_magical_attck = 400;
my_physical_defence = 500;
my_magical_defence = 350;
my_basic_attack_factor = 50;
my_defence_factor = 0.0018;
my_x = 300;
my_y = 0;
my_z = 300;


function set_uid(id)
    my_id = id;
    return my_element, my_lv, my_name, my_hp, my_physical_attack, my_magical_attck,
        my_physical_defence, my_magical_defence, my_basic_attack_factor, my_defence_factor;
end
```

● 패턴 처리

- 3개의 패턴 존재(랜덤으로 3개중 1개가 실행된다)
- 움직이는 패턴경우 타이머에 넣어 100ms마다 움직이도록 설정
- 피격판정



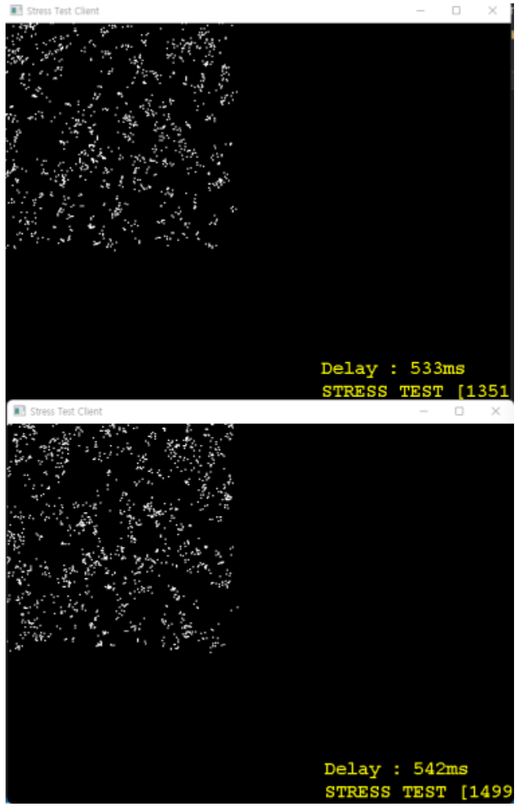
1. 직사각형을 2개의 삼각형으로 나눈다
2. 외적(check_inside)을 통해
점이 삼각형 내에 있는지(isInsideTriangle) 알아보고
삼각형 내에 있다면 피격 판정을 시킨다

```
bool Gaia::check_inside(pos a, pos b, pos c, pos n) {
    pos A, B, C;
    A.first = b.first - a.first;
    A.second = b.second - a.second;
    B.first = c.first - a.first;
    B.second = c.second - a.second;
    C.first = n.first - a.first;
    C.second = n.second - a.second;

    if ((A.first * B.second - A.second * B.first) * (A.first * C.second - A.second * C.first) < 0)
        return false;
    return true;
}

bool Gaia::isInsideTriangle(pos a, pos b, pos c, pos n)
{
    if (!check_inside(a, b, c, n)) return false;
    if (!check_inside(b, c, a, n)) return false;
    if (!check_inside(c, a, b, n)) return false;
    return true;

}
```

섹터분할(최적화 및 동접향상)

하드웨어 성능
- CPU : AMD RYZEN 7 5800H(8core)
- RAM : 16GB

●섹터 분할 전(동접 2500)

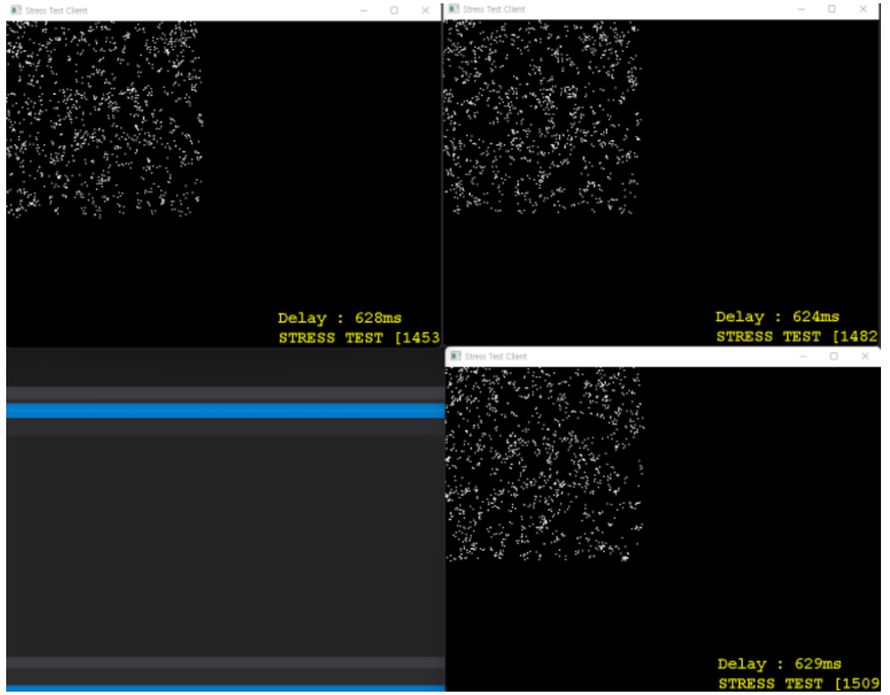


Put(Login)/Move시 시야처리

-> 모든 플레이어를 상대로 시야처리

((접속해 있는 플레이어 + Npc의 갯수) 번 탐색)

●섹터 분할 후(동접 4200)



Put(Login)/Move시 시야처리

-> 전체 지역을 64등분 하여 **자신이 해당하는 섹터와**

그 주위의 8개의 섹터에 있는 플레이어에 한해 시야처리

(플레이어가 골고루 퍼져 있다는 가정하에 ((9/64)*섹터분할 전 탐색 수) 만큼 탐색)

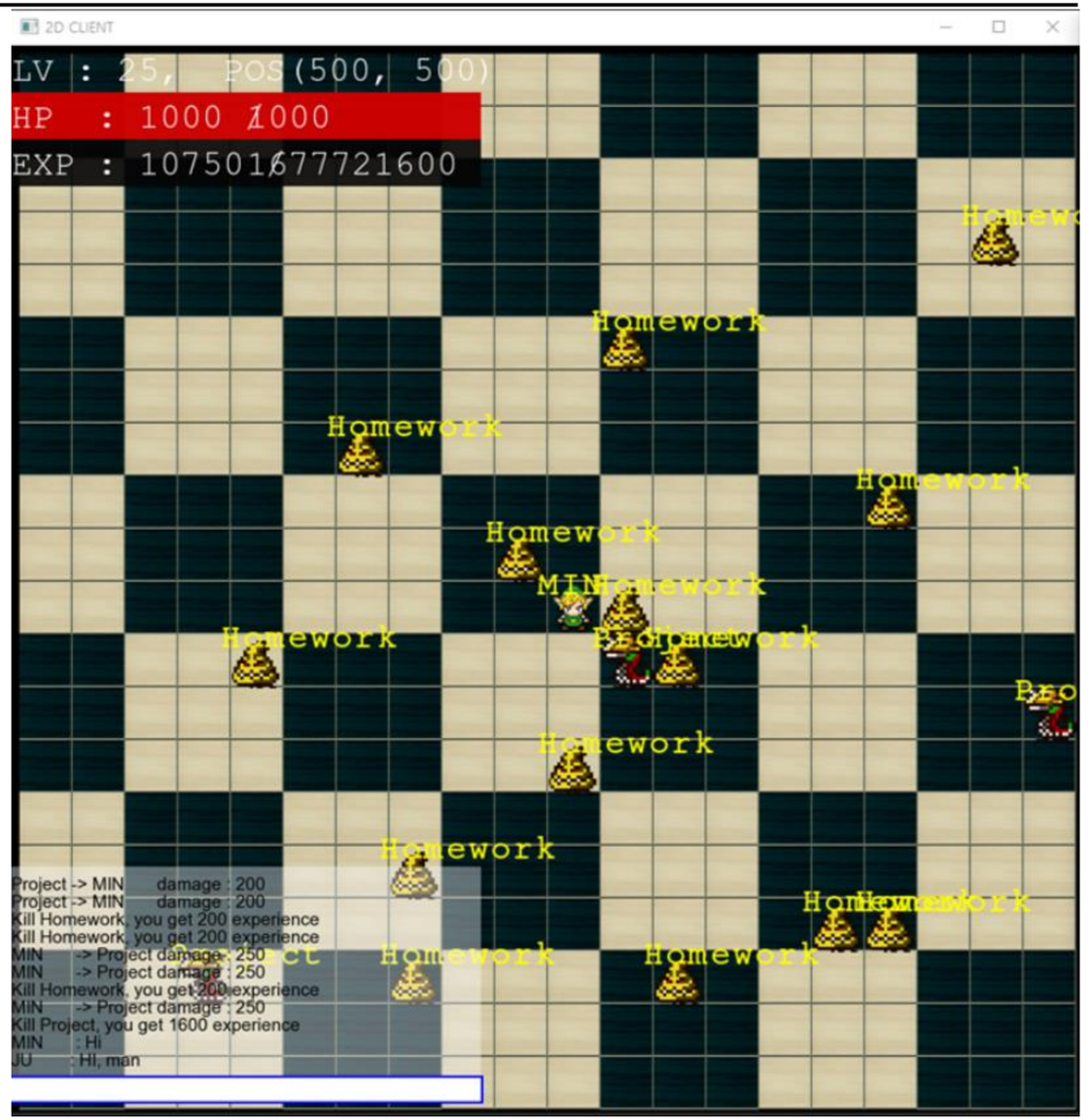
추가적으로 Move와 Look의 send를 합쳐보내줌으로써 Network I/O를 줄여

SystemCall를 덜 호출하도록 변경

게임 서버 프로그래밍

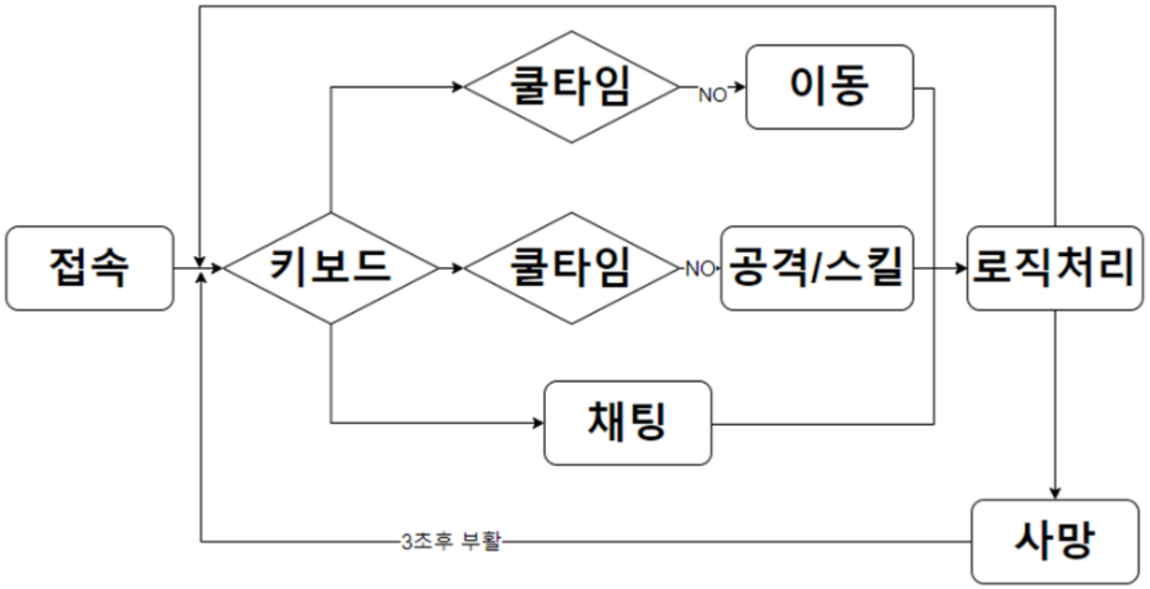
02

게임 장르	2D MMORPG
작업 기간	2021.11~2021.12
작업 인원	1명(개인 프로젝트)
구현내용	기술 - 멀티쓰레드 IOCP이용 - DB(My SQL, ODBC 사용) - Lua Script 이용 컨텐츠 - 고정형 몬스터(때려야지 따라오는 몬스터) - Agro 몬스터(범위에 들어오면 따라오는 몬스터) - 기본 공격과 스킬 2개 - 채팅 기능
깃허브	https://github.com/Asias-ara/2021-2GameServer-TermProject

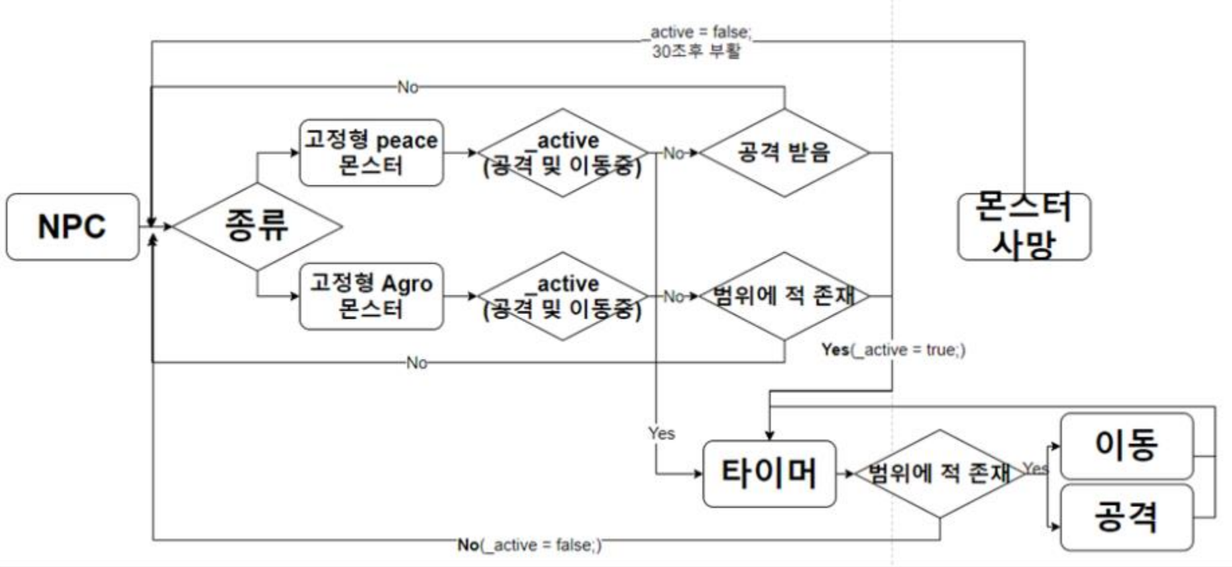


Game Flow Chart

Game Flow Chart (플레이어)



Game Flow Chart (Npc)



● Stored Procedure 사용

- 플레이어 검색

```
sprintf_s(temp, sizeof(temp), "EXEC search_player %s", login_id);
wchar_t* exec;
int strSize = MultiByteToWideChar(CP_ACP, 0, temp, -1, NULL, NULL);
exec = new WCHAR[strSize];
MultiByteToWideChar(CP_ACP, 0, temp, sizeof(temp) + 1, exec, strSize);

retcode = SQLExecDirect(hstmt, (SQLWCHAR*)exec, SQL_NTS);

if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
    retcode = SQLBindCol(hstmt, 1, SQL_C_LONG, &c_id, 100, &cbP_id);
    retcode = SQLBindCol(hstmt, 2, SQL_C_WCHAR, c_name, MAX_NAME_SIZE, &cbP_name);
    retcode = SQLBindCol(hstmt, 3, SQL_C_SHORT, &c_x, 100, &cbP_x);
    retcode = SQLBindCol(hstmt, 4, SQL_C_SHORT, &c_y, 100, &cbP_y);
    retcode = SQLBindCol(hstmt, 5, SQL_C_LONG, &c_hp, 100, &cbP_hp);
    retcode = SQLBindCol(hstmt, 6, SQL_C_SHORT, &c_lv, 100, &cbP_lv);
    retcode = SQLBindCol(hstmt, 7, SQL_C_LONG, &c_exp, 100, &cbP_exp);
    retcode = SQLBindCol(hstmt, 8, SQL_C_LONG, &c_maxhp, 100, &cbP_maxhp);
```

Stored Procedure 내용 캡쳐와 설명

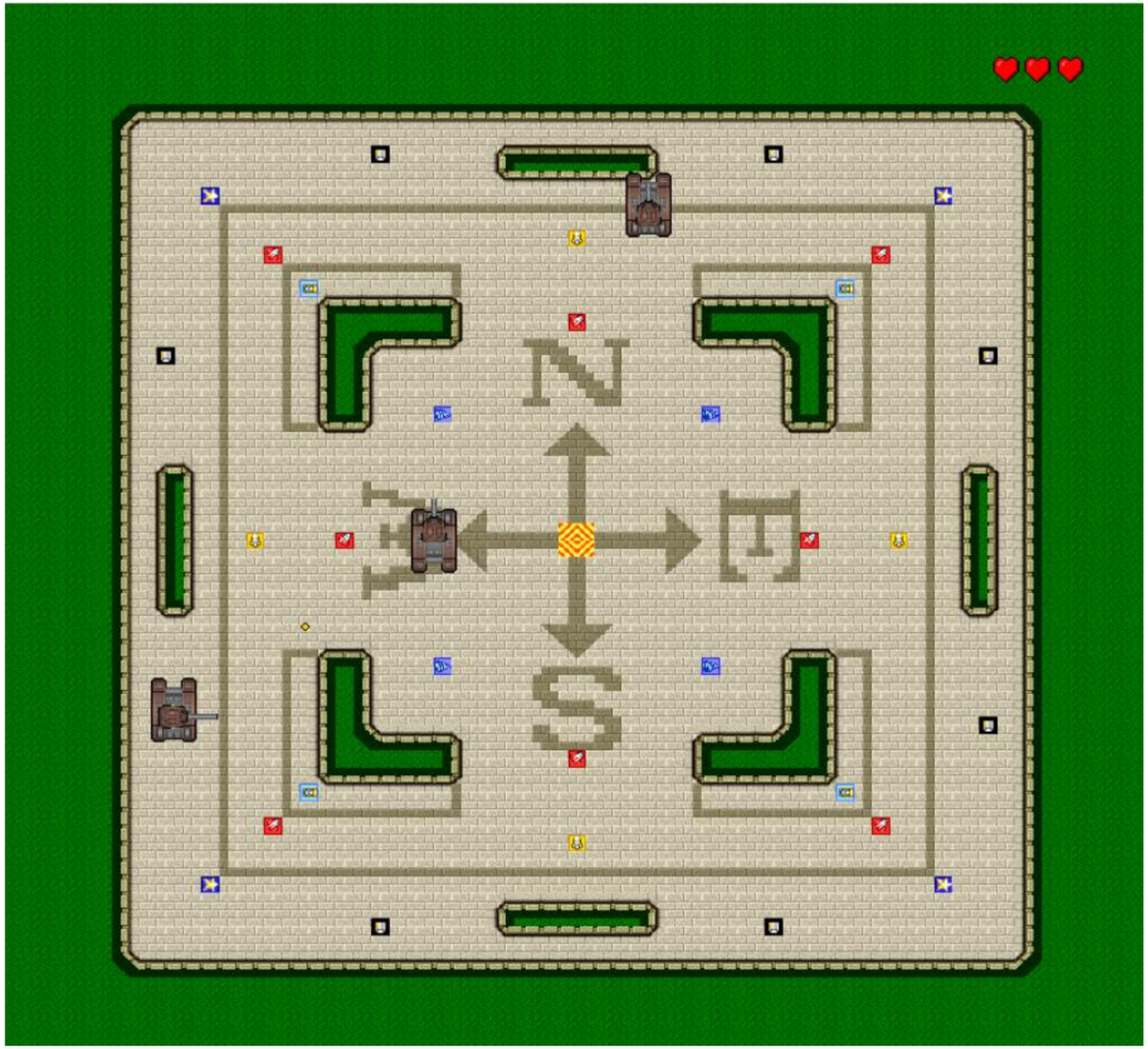
- 플레이어 위치 저장

```
void Save_position(Player* pl)
{
    SQLRETURN retcode;

    char temp[100];
    sprintf_s(temp, sizeof(temp), "EXEC save_player_info %d, %d, %d, %d, %d, %d, %d",
        pl->get_login_id(), pl->get_x(), pl->get_y(), pl->get_hp(),
        pl->get_lv(), pl->get_exp(), pl->get_maxhp());
    wchar_t* exec;
    int strSize = MultiByteToWideChar(CP_ACP, 0, temp, -1, NULL, NULL);
    exec = new WCHAR[strSize];
    MultiByteToWideChar(CP_ACP, 0, temp, sizeof(temp) + 1, exec, strSize);

    retcode = SQLExecDirect(hstmt, (SQLWCHAR*)exec, SQL_NTS);
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
        SQLLEN* pcrow = new SQLLEN;
        retcode = SQLRowCount(hstmt, pcrow);
        if (retcode == SQL_ERROR || retcode == SQL_SUCCESS_WITH_INFO) {
            HandleDiagnosticRecord(hstmt, SQL_HANDLE_STMT, retcode);
        }
    }
```

Stored Procedure 내용 캡쳐와 설명

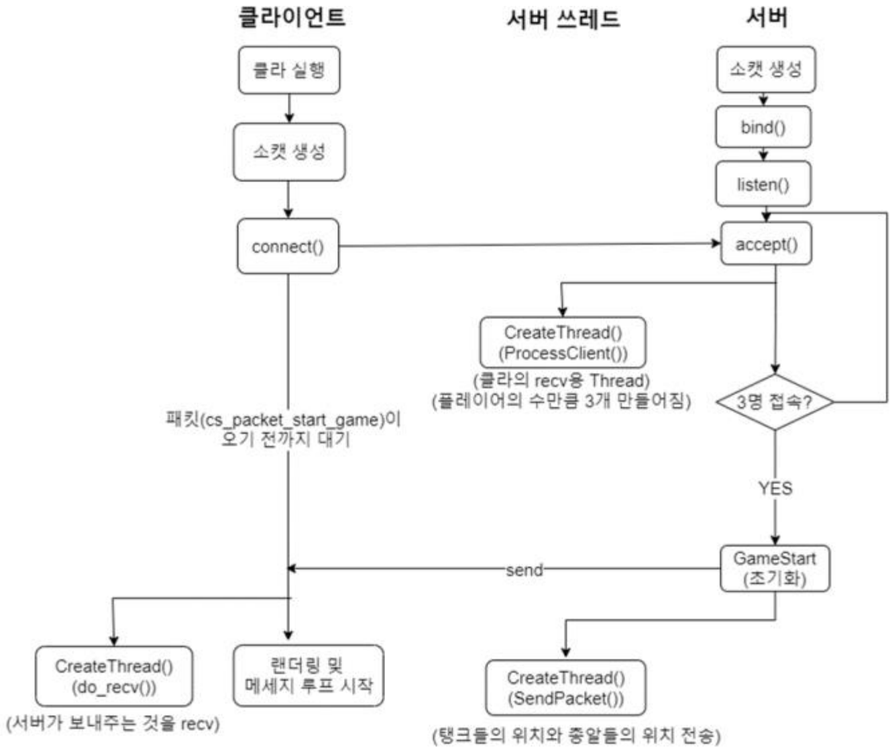
네트워크 게임 프로그래밍

게임 장르	3인 2D 어드벤처 게임
작업 기간	2021.11~2021.12
작업 인원	3명 - 클라이언트 1명 - 서버 2명
구현내용 (역할)	서버 프로그래머 - 게임 로그인 처리 - 서버 스레드간 동기화 - 총알 충돌처리
깃허브	https://github.com/Asias-ara/2021-2GameServer-TermProject



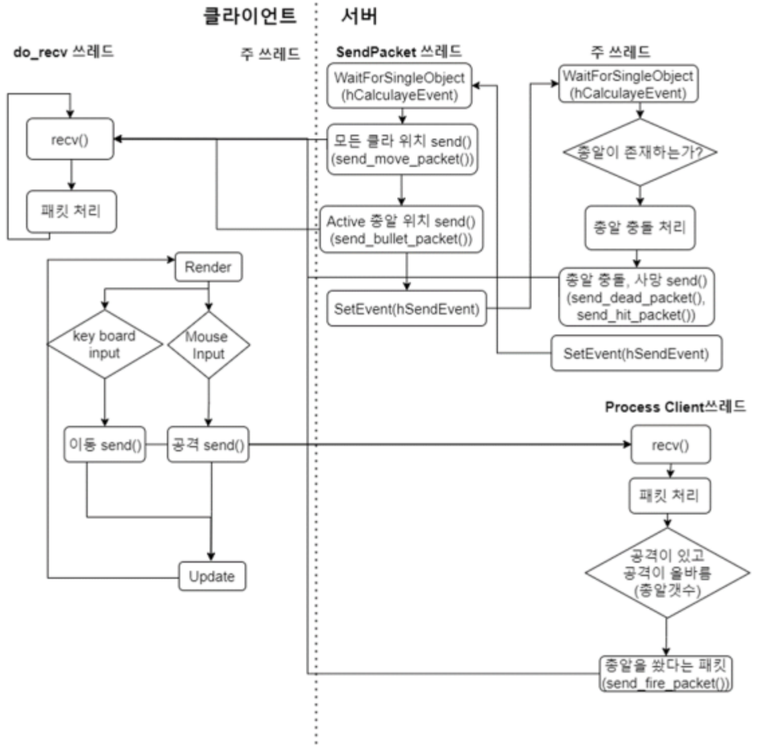
Game Flow Chart

Login Flow Chart



- 클라이언트가 접속시 각각에 대응하는 Thread(ProcessClient())를 만들며 이 쓰레드는 클라이언트가 보내는 패킷을 받는 역할을 한다
- 세명이 접속시 패킷을 보내는 Thread(SendPacket())를 하나 더 만들어 준다
- 마지막에는 총 5개의 쓰레드(주 쓰레드 1개, ProcessClient 쓰레드 3개, SendPacket 쓰레드 1개)가 생긴다

Game Flow Chart



- 팀원과 논의를 통해 가장 중요하게 생각한 것은 **충돌처리를 하는 주 쓰레드**이며 send는 주 쓰레드 한 후 처리하는게 옳다고 생각하여 SendPacket쓰레드와 주쓰레드를 **Event방식으로 공유자원에 대한 동기화**를 하였다.

Project Video Link

클라이언트 프로젝트 요약 영상(30초)

https://www.youtube.com/watch?v=TY211GPtiLI

클라이언트 프로젝트 풀 영상(1분 40초)

https://www.youtube.com/watch?v=fxH1n6ZDOX8