포트폴리오

게임 서버 프로그래머 지원

이종혁

010-8995-9851

gurpaper@naver.com

https://github.com/2JongHyeok

목차

Academy RPG

목적 : 소규모 MMORPG 제작

개발 기간 : 2024년 6월 10일 ~ 2024년 6월 30일

사용 도구 : C++(서버, 클라이언트)

개발 인원 : 1명

담당 업무 : MultiThread와 IOCP를 사용한 소규모 MMORPG 서버 제작.

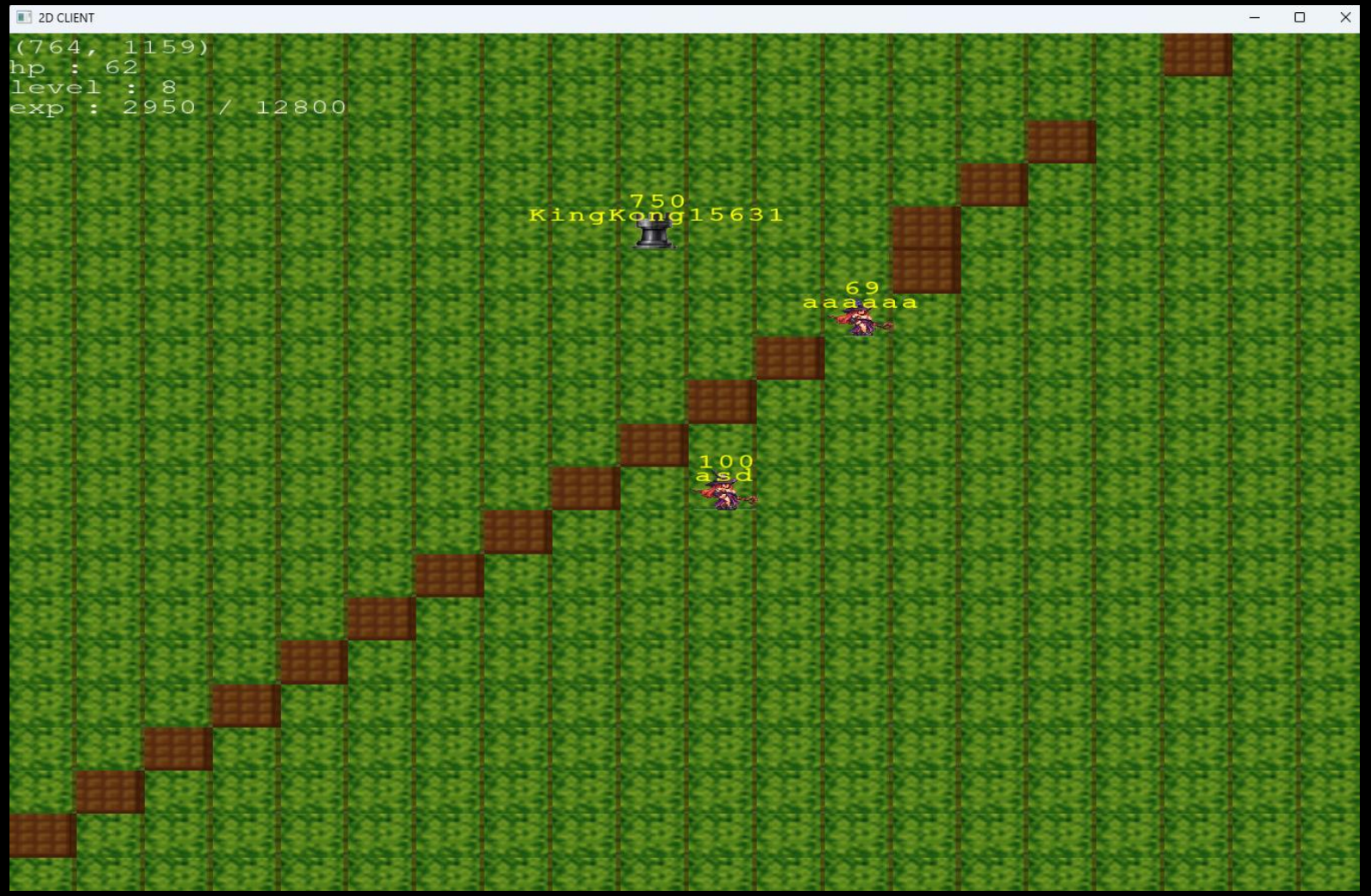
SFML을 사용한 간단한 클라이언트 제작.

게임 설명 : 넓은 맵을 돌아다니며 몬스터를 사냥하고 레벨 업을 하며 더 강한 몬스터들을 사냥하는 게임입니다.

Github :
https://github.com/2JongHyeok/GameServerTermProject/tree/main

Academy RPG

인 게임 화면



Academy RPG

몬스터 20만 마리 배치

몬스터 5종류 구현

보스 몬스터 구현

몬스터별 공격범위를 다르게 구현

몬스터 레벨에 따른 공격력, 체력 구현

고정몬스터 + 로밍몬스터 구현

몬스터 사망시 30초 후 부활하도록 구현

플레이어 3종류 구현

플레이어 3종류
Warrior : 시선방향 1칸 공격
Mage : 시선방향 3칸 공격
Prist : 주위 5칸 공격 + 주위 5칸 플레이어 힐

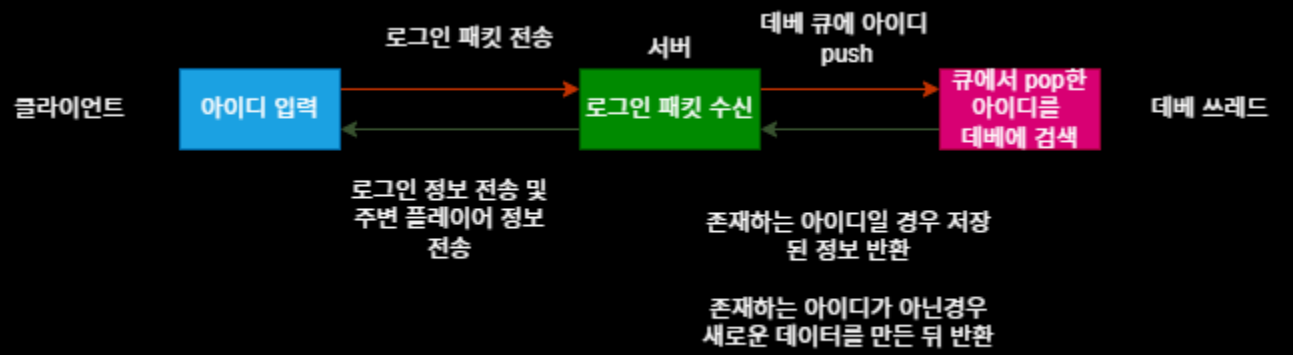
경험치 시스템, 레벨업, 성장 공격력, 방어력 구현

Mssql을 사용한 데이터 저장

Academy RPG

게임 로그인

https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
line 245 ~ 317 (서버 패킷 처리), line 1103 ~ 1139(데베 검색)



데베에 저장되는 정보 : id, name, level, exp, position

```sql
ALTER PROCEDURE [dbo].[SearchClient]
    @id INT
AS
BEGIN
    SET NOCOUNT ON;

    -- 주어진 ID로 플레이어 검색
    IF EXISTS (SELECT 1 FROM dbo.Players WHERE id = @id)
    BEGIN
        -- 플레이어가 존재하면 정보 반환
        SELECT name, levels, exp, x, y
        FROM dbo.Players
        WHERE id = @id;
    END
    ELSE
    BEGIN
        -- 플레이어가 존재하지 않으면 새로운 정보 추가
        DECLARE @newX INT, @newY INT;

        -- x, y 좌표를 2 ~ 1999 사이의 랜덤 값으로 설정
        SET @newX = FLOOR(RAND() * (1998) + 2);
        SET @newY = FLOOR(RAND() * (1998) + 2);

        -- 새 플레이어 정보 삽입
        INSERT INTO dbo.Players (id, name, levels, exp, x, y)
        VALUES (@id, CAST(@id AS NCHAR(10)), 1, 0, @newX, @newY);

        -- 새로 삽입된 정보 반환
        SELECT name, levels, exp, x, y
        FROM dbo.Players
        WHERE id = @id;
    END
END
```

SearchClient
Stored Procedure 쿼리문

Academy RPG

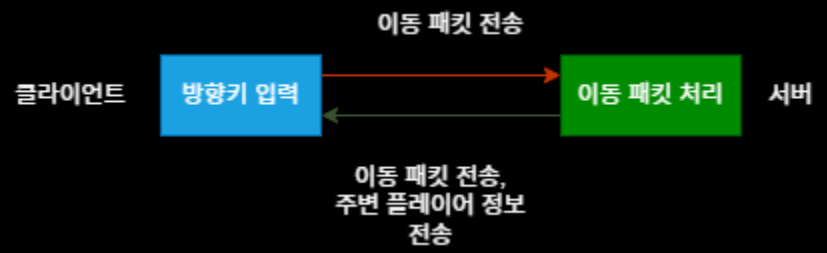
주변 플레이어 탐색 로직

https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/Grid.cpp
(섹터 탐색)
https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
Line 197 ~ 201 (can_see함수)

플레이어가 이동을 하면 위치에 맞는 섹터를 확인 후 섹터에 변경사항이 있으면 shared_lock을 사용해 섹터를 이동, 검색도 shared_lock을 사용해 주변 섹터 검색 .
주변 섹터에 있는 플레이어들 상대로 can_see 함수를 통해 시야 안에있는 플레이어 탐색

플레이어 이동

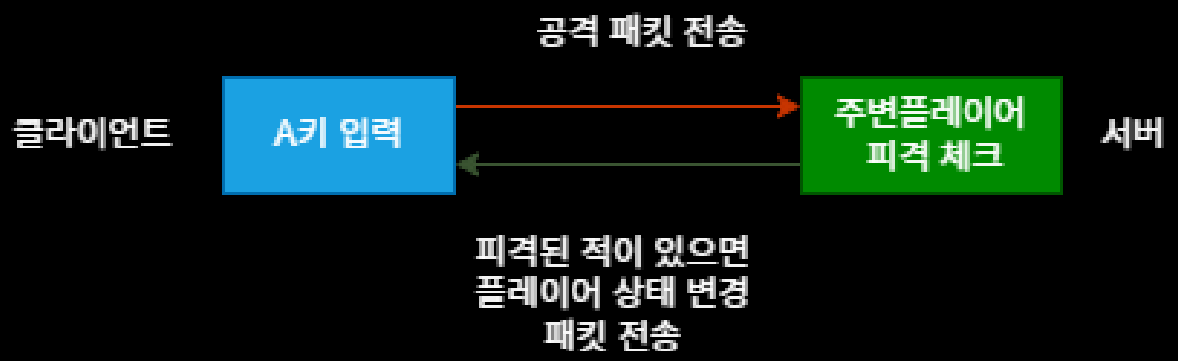
https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
line 318 ~ 401



Academy RPG

플레이어 공격 로직

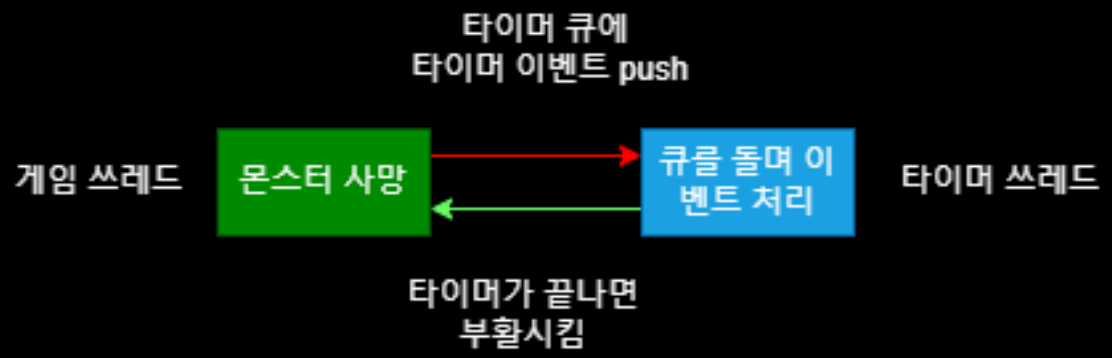
https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
line 402 ~ 572



공격 패킷을 받으면 주변 플레이어에 대하여
나의 직업에 맞는 피격 체크 후
플레이어 상태 변경 패킷 전송
만약 죽인 몬스터가 있으면 경험치 획득 및 레벨 업

몬스터 부활 로직

https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
line 672 ~696 (타이머), line 1023 ~ 1044 (부활)

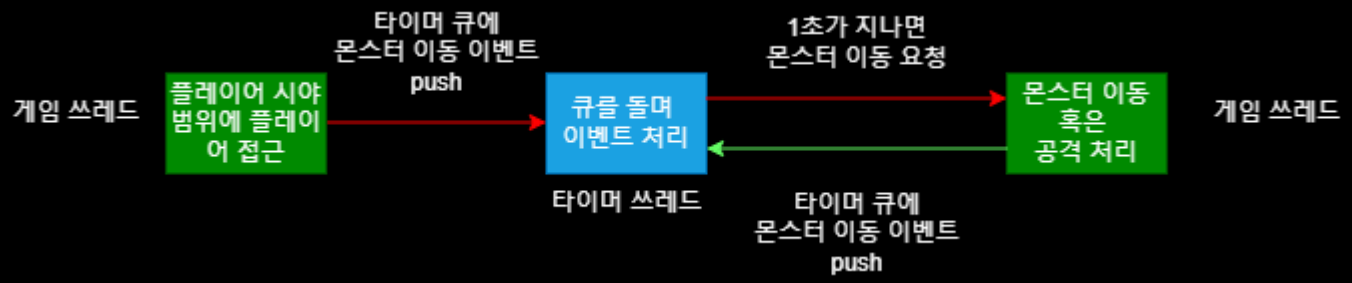


타이머 쓰레드는 1초마다 큐에 있는 모든 이벤트를 처리
30초가 지나면 죽었던 자리에서 몬스터 부활
몬스터는 부활 시 레벨, 체력, 데미지 증가.

Academy RPG

몬스터 이동, 공격 로직

https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
line 672 ~696 (타이머), line 1023 ~ 1044 (부활)



플레이어의 시야 안에 몬스터가 들어오면
몬스터는 이동을 함.
몬스터 시야범위 안에 플레이어가 들어오면
몬스터는 플레이어를 공격함.

Academy RPG

로그아웃시 정보 저장

플레이어가 게임을 종료하거나
비정상적인 접속종료 감지시 데베에 데이터 저장

https://github.com/2JongHyeok/GameServerTermProject/blob/main/Server/server.cpp
Line 1141 ~ 1157 (Stored Procedure 호출)
Line 923 ~ 941 (클라이언트 정상, 비정상 종료 부분)

```
ALTER PROCEDURE [dbo].[SaveData]
    @id int,
    @level int,
    @exp int,
    @x float,
    @y float
AS
BEGIN
    SET NOCOUNT ON;
    -- 일치하는 레코드가 있으면 값 업데이트
    UPDATE dbo.Players
    SET levels = @level,
        exp = @exp,
        x = @x,
        y = @y
    WHERE id = @id;
END;
```

SaveData
Stored Procedure 쿼리문

The Toys

목적 : 언리얼을 사용한 비대칭 PvP 게임 제작

개발 기간 : 2023년 12월 ~ 2024년 7월

사용 도구 : C++(서버), Unreal 5(클라이언트)

개발 인원 : 3명 (서버 1명, 클라이언트 1명, 모델러/기획자 1명)

담당 업무 : IOCP 로비 서버 구현,

Boost ASIO 게임 서버 구현,

로비 서버에서 매칭 시스템 구현,

작업 비중 : 서버(100%), 클라이언트 (5%, 서버와 연동 부분)

Github : https://github.com/NewbieProgrammerCrew/graduation-project

The Toys

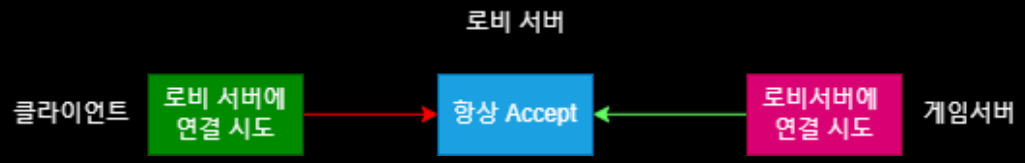
인게임 화면

The Toys

IOCP 로비 서버

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Lobby%20Server/Server.cpp
Line 217 ~252

서버 연결
로비서버 파트
로직



GameServer와 Clients에게
각각 다른 포트로 접속 하게 한 뒤.
각각 다른 Key값을 주어.
이 후에는 key값으로 구분.

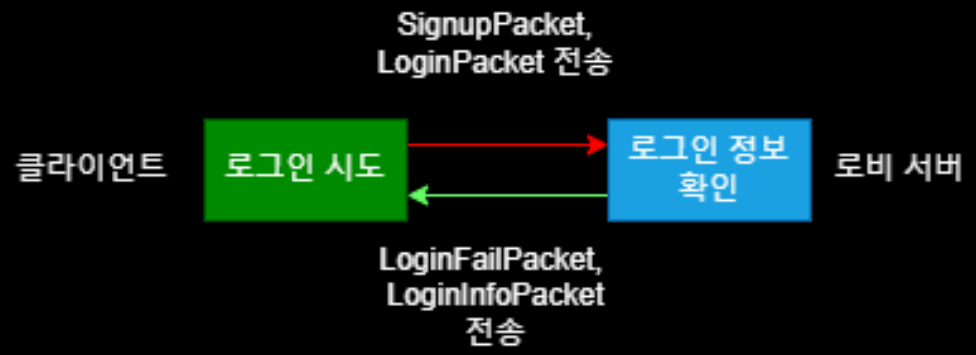
```
case OP_ACCEPT: {
    if (static_cast<int>(key) == 9000) {
        lsession.socket_ = l_c_socket;
        lsession.id_ = 100'000;
        lsession.prev_remain_ = 0;
        CreateIoCompletionPort(reinterpret_cast<HANDLE>(l_c_socket), h_iocp, 100'000, 0);
        lsession.do_recv();
        l_c_socket = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, WSA_FLAG_OVERLAPPED);
        ZeroMemory(&l_a_over.over, sizeof(l_a_over.over));
        int addr_size = sizeof(SOCKADDR_IN);
        AcceptEx(l_s_socket, l_c_socket, l_a_over.send_buf, 0, addr_size + 16, addr_size + 16, 0, &l_a_over.over);
    }
    else {
        int client_id = get_new_client_id();
        if (client_id != -1) {
            {
                lock_guard<mutex> ll(clients[client_id].s_lock_);
                clients[client_id].state_ = ST_ALLOC;
            }
            clients[client_id].id_ = client_id;
            clients[client_id].prev_remain_ = 0;
            clients[client_id].socket_ = g_c_socket;
            CreateIoCompletionPort(reinterpret_cast<HANDLE>(g_c_socket),
                h_iocp, client_id, 0);
            clients[client_id].do_recv();
            g_c_socket = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, WSA_FLAG_OVERLAPPED);
        }
        else {
            cout << "Max user exceeded.₩n";
        }
        ZeroMemory(&g_a_over.over, sizeof(g_a_over.over));
        int addr_size = sizeof(SOCKADDR_IN);
        AcceptEx(g_s_socket, g_c_socket, g_a_over.send_buf, 0, addr_size + 16, addr_size + 16, 0, &g_a_over.over);
    }
    break;
}
```

The Toys

Game Client

게임 로그인 로직

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World/Private/NetworkingThread.cpp
Line 49 ~107 (Run 함수)
https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/UnrealProject/SampleServer_Unreal%205.3/Source/NPC_World/Private/Manager/MyGameInstance.cpp
Line 429 ~ 430 (SignupPacket, LoginPacket)



Client가 실행될때
FRunnable 을 상속받아 FsocketThread를 생성
Run() 함수가 불리고
SignupPacket과 LoginPacket을 보내
로그인 시도및 완료.

게임 매칭 로직

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Lobby%20Server/Server.cpp
Line 100~134

술래 1명, 도망자 4명이 모이면
플레이어들을 큐에서 빼 와서
게임을 시작할 준비를 함.

ChaserQueue와 RunnerQueue는
concurrent_queue 로 구현

게임을 시작 가능한 최소 인원이 모이면
Try_pop으로 queue에서 pop시도.
다른 쓰레드가 이미 pop을 해서
게임을 시작할 인원이 부족하면
지금까지 pop한 플레이어들을
다시 queue에 push 한 후.
재시도

```
bool allPlayersReady = false;
if (ChaserQueue.unsafe_size() >= MAX_CHASER_NUM) {
    if (RunnerQueue.unsafe_size() >= MAX_RUNNER_NUM) {
        allPlayersReady = true;
    }
}

if (allPlayersReady) {
    int chaser;
    if (!ChaserQueue.try_pop(chaser))
        break;
    if (clients[chaser].state_ == ST_FREE)
        break;
    int runners[MAX_RUNNER_NUM];
    for (int i = 0; i < MAX_RUNNER_NUM; ++i) {
        runners[i] = -1;
        if (!RunnerQueue.try_pop(runners[i])) {
            for (int rn : runners) {
                if (rn == -1) {
                    break;
                }
                RunnerQueue.push(rn);
            }
            ChaserQueue.push(chaser);
            return;
        }
        if (clients[runners[i]].state_ == ST_FREE) {
            i--;
            continue;
        }
    }
```

게임 서버로 클라이언트 재연결 로직

매칭이 잡히면
관리하는 방이 가장 적은
게임서버의 쓰레드에게
방 생성 요청을 보낸 후
각각의 클라이언트에게
접속해야 할 게임서버의
주소 및 포트번호를 전송

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Lobby%20Server/Server.cpp
Line 136 ~ 157

```
while (true) {
    int thread_index = 0;
    int min_thread_contention = 0x7FFFFFFF;
    for (int i = 0; i < GameServerThreadContention.size(); ++i) {
        if (GameServerThreadContention[i] < min_thread_contention) {
            min_thread_contention = GameServerThreadContention[i];
            thread_index = i;
        }
    }
    if (thread_index >= 0 && thread_index < GameServerThreadContention.size()) {
        if (GameServerThreadContention[thread_index].compare_exchange_weak(min_thread_contention, min_thread_contention + 1)) {
            int room_num = GameServerThreadRoomCount[thread_index]++;
            lsession.SendCreateRoomPacket(chaser, runners);
            if (chaser >= 0 && chaser <MAX_USER) {
                clients[chaser].SendGameStartPacket(GameServerPortNums[thread_index]);
                for (int rn : runners) {
                    clients[rn].SendGameStartPacket(GameServerPortNums[thread_index]);
                }
                break;
            }
        }
    }
}
```

서버 연결
게임서버 파트 로직

게임서버가 로비 서버에 연결(Connect) 시도.
로비서버는 항상 Accept 만 하도록 함.

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/server.cpp
Line 58 ~65

```
class cClient : public std::enable_shared_from_this<cClient> {
public:
    cClient(boost::asio::io_context& io_context, const std::string& host, const std::string& port)
        : io_context_(io_context), socket_(io_context) {
        tcp::resolver resolver(io_context_);
        auto endpoints = resolver.resolve(host, port);
        do_connect(endpoints);
    }

    void do_write(unsigned char* packet, std::size_t length)
    {
        socket_.async_write_some(boost::asio::buffer(packet, length), [this, packet, length](boost::system::error_code ec, std::size_t bytes_transferred) {
            if (!ec)
            {
                if (length != bytes_transferred) {
                    cout << "Incomplete Send occured on Session To Lobby Server. This Session should be closed.₩n";
                }
                delete packet;
            }
            });
    }
    void send_packet(void* packet)
    {
        int packet_size = reinterpret_cast<unsigned char*>(packet)[0];
        unsigned char* buff = new unsigned char[packet_size];
        memcpy(buff, packet, packet_size);
        do_write(buff, packet_size);
    }
private:
    void do_connect(const tcp::resolver::results_type& endpoints) {
        boost::asio::async_connect(socket_, endpoints,
            [this](boost::system::error_code ec, tcp::endpoint) {
                if (!ec) {
                    GAME_SERVER_OPENED_PACKET p;
                    p.size = sizeof(GAME_SERVER_OPENED_PACKET);
                    p.type = GAME_SERVER_OPENED;
                    memcpy(p.address, "127.0.0.1", sizeof("127.0.0.1"));
                    p.portNum = 9001;
                    send_packet(&p);
                    do_read();
                }
            });
    }
```

The Toys

게임 시작 로직

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/server.cpp
Line 169 ~ 196 (class cServer),
https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/Session.cpp
Line 625 ~ 636

매칭이 잡힌
클라이언트들이 로비서버에서 준 주소로
게임서버에 접속을 하면
게임서버에서는 해당 Room 플레이어들이
모두 접속할때까지 대기하였다가.
모든 플레이어들이 접속을 하면
게임을 시작 함.
게임 서버는 멀티쓰레드를 사용하지만
각 쓰레드가 각각 다른 방을 관리. 따라서
컨테이너들의 lock/unlock 불필요.

```
class cServer
{
private:
    tcp::acceptor   acceptor;
    void do_Accept()
    {
        acceptor.async_accept([this](boost::system::error_code ec, tcp::socket socket) {
            if (!ec)
            {
                int p_id = Get_New_ClientID();
                clients[p_id] = std::make_shared<cSession>(std::move(socket), p_id);
                clients[p_id]->ingame_ = true;
                clients[p_id]->Start();
                do_Accept();
            }
            else {
                cout << "ERROR : " << ec.what() << endl;
            }
            });
    }

public:
    cServer(boost::asio::io_context& io_service, int port, int thread_num) : acceptor(io_service, tcp::endpoint(tcp::v4(), port))
    {
        do_Accept();
    }
};
```

```
int my_count = WaitingQueue[p->GroupNum]++;
if (my_count == MAX_ROOM_PLAYER-1)
    isGroupReady = true;
WaitingMap[p->GroupNum][my_count] = c_id;

if (isGroupReady) {
    WaitingQueue.erase(p->GroupNum);
    IngameMapData igmd;
    int player_count = 1;
    for (int id : WaitingMap[p->GroupNum]) {
        if (clients[id]->charactor_num_ >= 6) {
            igmd.player_ids_[0] = id;
        }
        else {
            igmd.player_ids_[player_count++] = id;
        }
    }
```

The Toys

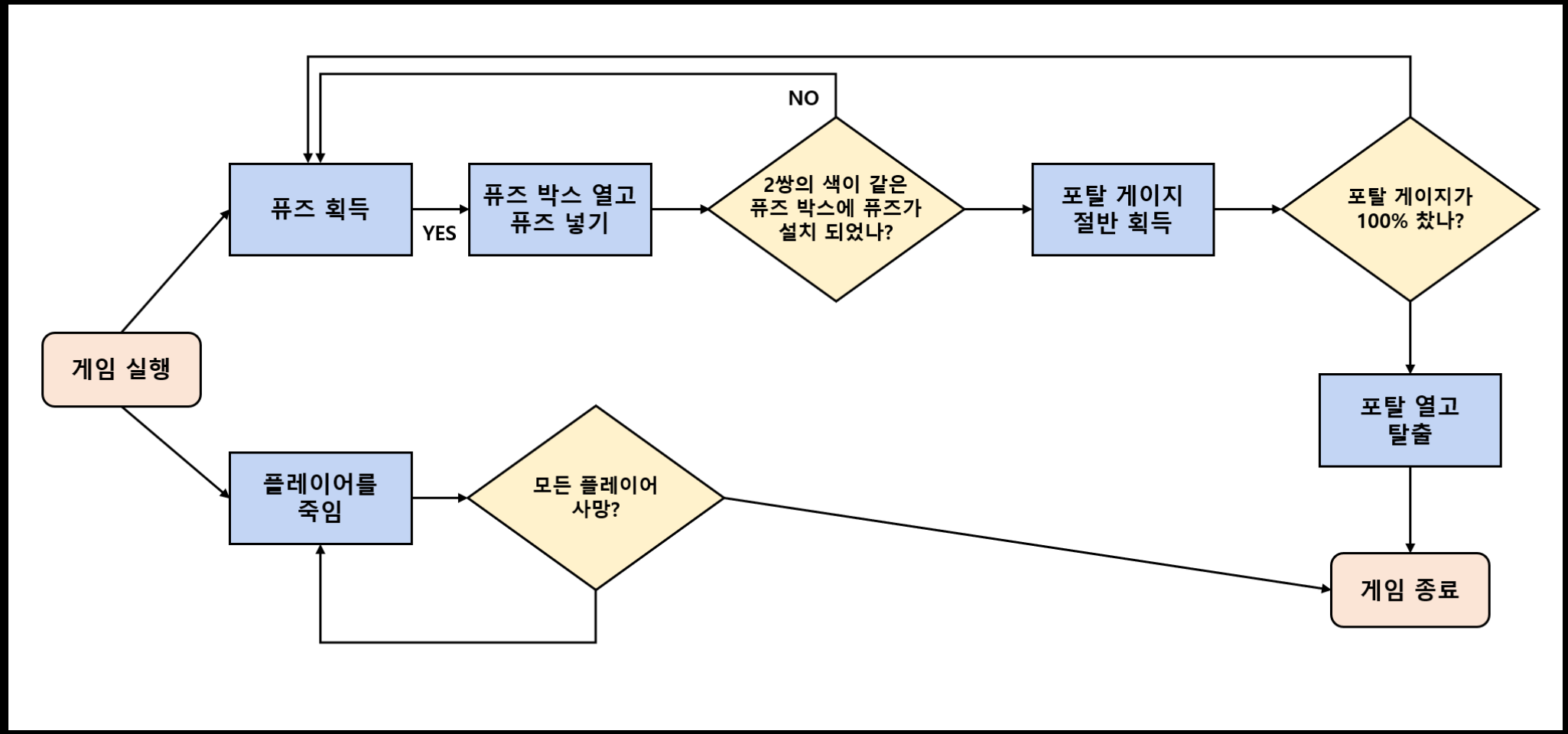
맵 오브젝트 로드 로직

클라이언트에서 object들의 정보를 .json 파일로 넘겨주면
그 파일들을 읽어 Sector에 배치
이후 캐릭터들의 이동시 충돌처리 및 상호작용에 사용

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/server.cpp
Line 272 ~ 414

게임 흐름도

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/Session.cpp
Line 760 ~ 1221 (여러가지 게임에 필요한 패킷들)



The Toys

게임 타이머 로직

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/Session.cpp
Line 415 ~ 586 (타이머 로직)

https://github.com/NewbieProgrammerCrew/graduation-project/blob/main/Server/Single%20Thread%20ASIO/source/server.cpp
Line 421 ~ 422, 442 ~ 454 (Timer 생성 및 등록)

시간 측정이 필요한 인게임 요소 발생시
타이머 큐에 넣은 후
DoTimer 함수에서 시간 처리
Ex) 아이템 박스 오픈, 퓨즈 박스 오픈,
술래 부활, 캐릭터 공격, 스킬(무적)

The Toys

소감

팀 프로젝트에서 팀장을 맡고
처음으로 팀을 이끌며 제작한 게임입니다.

모델러의 갑작스러운 포지션 변환, 기획자와의 마찰등 여러 요소에 의해
게임 제작시 소통이 중요하다는 것을 절실하게 느꼈던 프로젝트였습니다.

많은 중압감과 부족한 일정에 개인적으로 아쉬움이 많이 남은
프로젝트였습니다.
하지만 제가 지금까지 배워온 내용을 일부 잘 녹여낸 게임이라고 생각합니다.
감사합니다.

추가 사항

- OPIC IM3 보유