네트워크 게임 프로그래밍 추진서

2016182024 윤선규

2016182026 이동수

2017180015 서동현

**애플리케이션 기획**

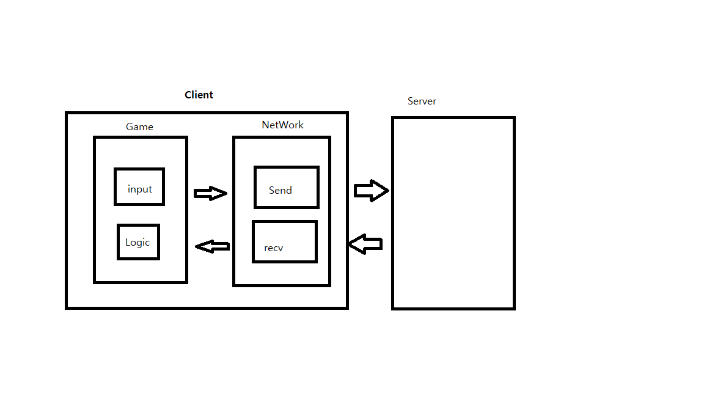
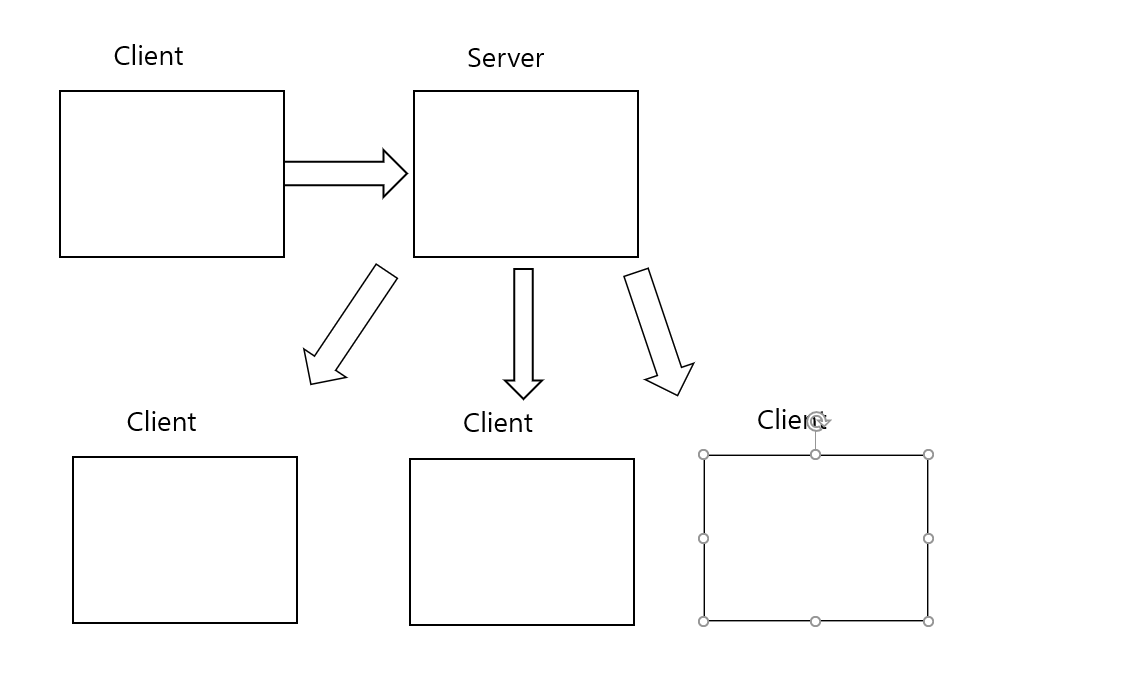
상대방과 채팅으로 대화를 나누며 상대방과 충돌을 해서 시비를 걸거나.

타격하여 장난을 칠 수 있는 애플리케이션입니다.

잔디, 꽃, 식물, 다채로운이(가) 표시된 사진

자동 생성된 설명

**High Level**



클라이언트의 로직은 서버에 의해서만 변경 가능합니다.

**Low Level .**

**구현될 함수들의 수도코드**

void CALLBACK send\_callback(DWORD errflag, DWORD num\_bytes, LPWSAOVERLAPPED send\_over, DWORD flag);

// overlapped io 사용, send 콜백함수. 별일 하지 않는다. send\_over delete 해준다.

void CALLBACK recv\_callback(DWORD errflag, DWORD num\_bytes, LPWSAOVERLAPPED recv\_over, DWORD flag)

{

process\_packet();

}

// overlapped io 사용, recv 콜백함수.

// process\_packet 함수를 호출해서 패킷을 처리한다.

// 콜백함수에 쓸 OVERLAPPED 구조체의 변형.

struct EXP\_OVER

{

WSAOVERLAPPED \_wsa\_over{};

Int \_s\_id;

WSABUF \_wsa\_buf;

char \_send\_msg[BUFSIZE];

};

// 서버가 사용하는 클래스. 각각의 SESSION은 클라이언트하나를 의미한다.

class SESSION

{

public:

void do\_recv()

{

WSARecv(\_socket, &\_recv\_wasbuf, &\_recv\_over, recv\_callback);

}

void do\_send(int sender\_id, int num\_bytes, char\* msg)

{

EXP\_OVER\* ex\_over = new EXP\_OVER{};

WSASend(\_socket, &ex\_over->\_wsa\_buf, &ex\_over->\_wsa\_over, send\_callback);

}

private:

int \_id;

SOCKET \_socket;

WSABUF \_recv\_wasbuf;

char \_recv\_buf[MAX\_PACKET\_SIZE]{};

WSABUF \_send\_wasbuf;

char \_send\_buf[MAX\_PACKET\_SIZE]{};

WSAOVERLAPPED \_recv\_over{};

};

// recv 한 패킷을 처리하는 함수.

// 서버는 패킷 처리 후 모든 session 에 적절한 패킷을 send 한다.

void process\_packet(int client\_id, unsigned char\* packet\_start)

{

unsigned char packet\_type = packet\_start[1];

SESSION& client = session[client\_id];

switch (packet\_type)

{

case CS\_PACKET\_LOGIN:

{

// to do

}

break;

case CS\_PACKET\_MOVE:

{

// to do

}

break;

}

}

**네트워크 기능**

로그인 / 로그아웃 / 채팅 / 타격 / 이동 / 모델변경

**패킷**

template<class T>

struct packet\_base

{

int8 size = sizeof(T);

PAKCET\_TYPE packet\_type;

};

#define PACKET(name) struct name : packet\_base<name>

//=============== LOG\_IN =================

// => 이 이름으로 로그인 할래

PACKET(cs\_try\_login)

{

char name[MAX\_NAME\_SIZE];

};

// => ok 너의 id는 이거야

PACKET(sc\_ok\_login)

{

int8 login\_id;

};

// 맵에 누구누구가 어디어디에 있는지 알려주어야 함

// PACKET(sc\_new\_charator) \* numofchractor; to new player

PACKET(sc\_new\_charator)

{

int8 login\_id;

char name[MAX\_NAME\_SIZE];

float x;

float y;

};

PACKET(cs\_make\_map\_done)

{

};

// 다른 클라이언트들 한테 새 캐릭터 로그인알림

// PACKET(sc\_new\_charator); to old players

//============= LOG\_OUT ===========

// => logout (esc 누름)

PACKET(sc\_logout)

{

int8 logout\_id;

};

// => recv 의 리턴값이 0 인 경우 또는 PACKET(sc\_logout) 패킷이 왔을 경우.

PACKET(cs\_logout)

{

int8 logout\_id;

};

// => 맵에서 플레이어 제거.

//=============== MOVE\_INPUT =================

// 이동연잔자들, 비트연산으로 press,unpress 설정

enum class MOVE\_DIR : int8

{

FORWARD = 1 << 0,

BACK = 1 << 1,

LEFT = 1 << 2,

RIGHT = 1 << 3

};

// => input 이 변화했

PACKET(cs\_moved)

{

MOVE\_DIR arrow\_key;

};

// => 다른플레이어들한테 전송

PACKET(sc\_moved)

{

int8 mover\_id;

MOVE\_DIR arrow\_key;

};

//=============== CHATTING =================

// => 채팅 enter

PACKET(cs\_chat)

{

char chat[30];

char padding = '\0';

};

// => 다른 플레이어들한테 전송

PACKET(sc\_chat)

{

int8 chatter\_id;

char chat[30];

char padding = '\0';

};

//============= 타격 ===========

// => 플레이어의 타격시도

PACKET(sc\_try\_attack)

{

int8 defender\_id;

};

// => 서버에서 위치와 접속여부를 토대로 검증

PACKET(cs\_attacked)

{

int8 attacker\_id;

int8 defender\_id;

};

// => attacket 의 공격 이펙트 발동

// => defender 의 피격 이펙트 발동

//==================== MODEL\_CHANGE ==========================

// => 변경할래

PACKET(cs\_model\_change)

{

int8 model\_id;

};

// => ok, 모든 플레이어들한테 전송

PACKET(sc\_model\_change)

{

int8 id;

int8 model\_id;

};

// => 적용.

**일정 및 역할 분담**

윤선규 :: Overlaped\_IO 프레임워크 제작, 클라이언트 구현 // logout , 타격 패킷로직 구현

서동현 :: 로그인, 모델변경 패킷로직 구현

텍스트이(가) 표시된 사진

자동 생성된 설명이동수 :: 이동, 채팅 패킷로직 구현