**네트워크 게임 프로그래밍**

**팀프로젝트**

윤승규

신동원

김석현

**목차**

[1. 게임 기획 3](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084126)

[1.1 게임 개요 3](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084127)

[1.2 조작 방법 3](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128)

[1.3 구현 내용](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128) 4

[2. High Level Design](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084131) 5

[2.1 서버](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084132) 5

[2.2 클라이언트](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084133) 5

[2.3 서버-클라이언트 관계도](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084128) 6

[3. Low Level Design 7](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084136)

[3.1 서버 7](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084137)

[3.2 클라이언트](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084139) 10

[4. 팀원 역할](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084143) 11

[5. 개발 일정 1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084145)2

[5.1 윤승규 일정 1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084146)2

[5.2 신동원 일정 1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084147)2

[5.3 김석현 일정 1](file:///C:\Users\tjrgu\Desktop\학교\넷겜플\새%20폴더\네트워크%20게임%20프로그래밍%20팀프.docx#_Toc55084148)2

1. **게임 기획**
   1. **게임 개요**



게임: 봄버맨 모작게임

플랫폼: PC

규칙: 2명의 플레이어가 일정 시간 후에 폭발하여 4방위로 일정 영역에 데미지를 입히는 폭탄을 설치해 상대를 공격, 최종 생존자 1명이 승리.

* 1. **조작 방법**

|  |  |
| --- | --- |
| 조작키 | 설명 |
| W, A, S, D | 플레이어의 이동 방향키 W-위/A-왼/S-뒤/D-오 |
| Space bar | 플레이어의 위치에 폭탄을 설치한다. |

* 1. **구현 내용**
* **클라이언트**
  + 타이틀 씬
    - 게임 시작 및 매칭
  + 게임 씬
    - 플레이어 - 이동, 폭탄 설치
    - 게임 정보 수신 및 display
    - 승패 여부 판단
* **서버**
  + 메인 스레드
    - 매칭 스레드 생성
    - 매칭 스레드에 클라이언트 연결
    - 업데이트를 담당할 게임 스레드 생성 및 환경 구성
  + 매칭 스레드
    - 매칭 환경 구성
    - 각 클라이언트를 담당할 메세지 스레드 생성 및 연결
  + 메세지 스레드
    - 클라이언트 입력 데이터 수신
    - 수신한 데이터를 게임 스레드에 업데이트
    - 업데이트된 게임 스레드의 정보를 클라이언트에 송신
  + 게임 스레드
    - 메세지 스레드를 통해 전달받은 데이터를 통해 연산
    - 연상 후 업데이트된 데이터를 메세지 스레드에 전달

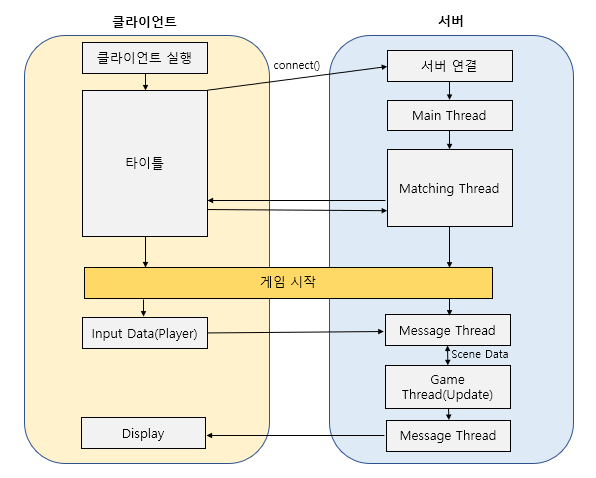
1. **High level design**
   1. **서버**

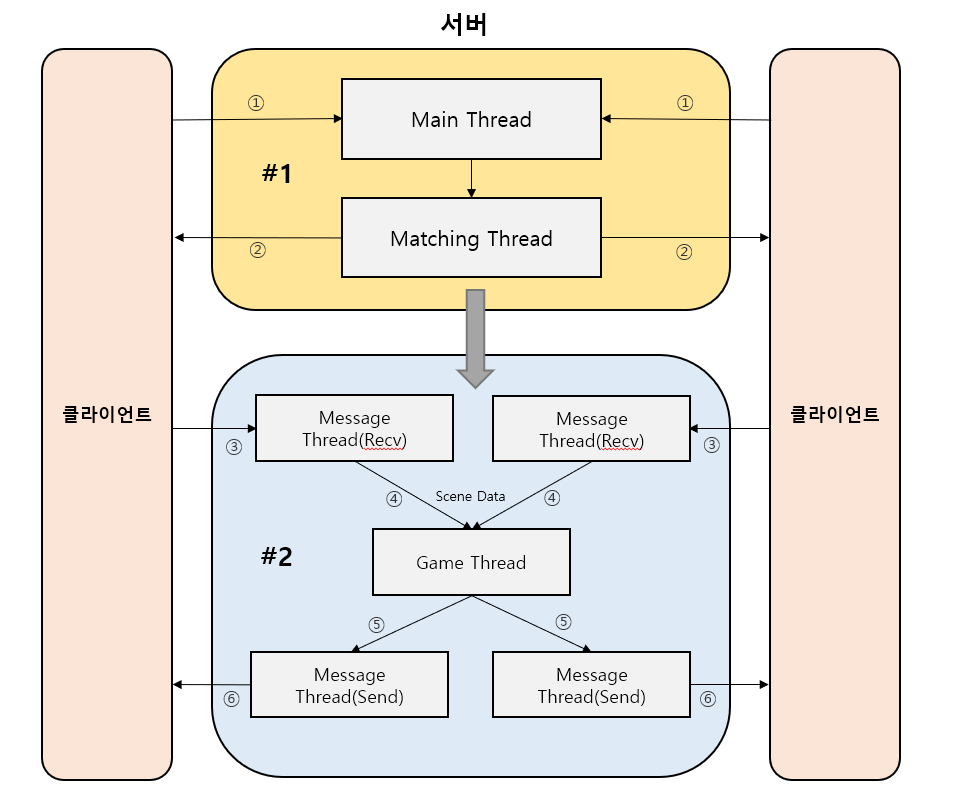
* TCP/IP 프로토콜 방식
* 모듈 별 역할
  + main : 매칭 스레드, 게임 스레드 생성 및 클라이언트 연결
  + matching : 매칭 환경 구성 및 각 클라이언트 담당 메세지 스레드 생성 및 연결
  + message : 클라이언트와 통신
  + game : 수신 데이터를 통해 연산 및 업데이트 데이터 전달

**2.2 클라이언트**

* 모듈 별 역할
  + title : 게임 시작 및 매칭
  + game : 게임 플레이 및 서버와의 통신

**2.3 서버-클라이언트 관계도**





1. **Low level design**
   1. **서버**

* 통신 데이터 구조

unsigned char ClientNum

vector<SOCKET> MatchingQueue

typedef struct ClientState

{

SOCKET ClientSocket

bool ReadyState = 0

}

class SceneData

{

Private:

PlayerStatus m\_playerStatus[MAX\_PLAYER];

MapData m\_mapData[MAP\_SIZE][MAP\_SIZE];

Public:

update();

keyInput(SOCKET socket, KeyInput key)

}

struct PlayerStatus

{

SOCKET playerSocket;

char speed;

char power;

bool isAlive;

};

* 함수

송신

Recv\_Con\_data() 클라이언트 로부터 데이터 소켓을 전달받는 함수

Recv\_input\_data() 클라이언트 로부터 입력 데이터를 받는 함수

Recv\_check\_state() 클라이언트로부터 실시간으로 게임 상태를 확인하도록 요청 받아 확인하는 함수

수신

Send\_Con\_data() 클라이언트 에 데이터 소켓을 전달하는 함수

Send\_draw\_data() 서버에서 Update 처리한 데이터를 클라이언트에 보내는 함수

send\_check() 2명의 player의 상태를 보내는 함수

처리

Change\_status() 게임의 상태를 play, wait, end 로 바꿔주는 함수

Check\_player() 2명의 player가 서버와 연결되어 준비상태인지 확인하는 함수

Update() 입력 받은 데이터를 확인해 최신화 하는 함수

Check\_state() 게임의 진행 상황을 서버가 확인하도록 요청하는 함수

Timeout() 일정시간 안에 입력 데이터의 전송이 되지 않고 지연 시 False로 처리하는 함수

* 변수

struct Point

{

short x;

short y;

};

struct PlayerStatus

{

KeyInput key;

char speed; // 플레이어 속도

char power; // 폭탄 길이

Point position; // 플레이어 위치

bool isAlive; // 생존 여부

};

struct MapData

{

Bool isRock

Bool isBubble;

};

* 1. **클라이언트**
* 통신 데이터 구조

struct KeyInput

{

bool key\_UP;

bool key\_Down;

bool key\_Left;

bool key\_Right;

bool key\_Space;

};

* 함수

송신

Recv\_Con\_data() 서버 로부터 데이터 소켓을 전달받는 함수

Recv\_draw\_data() 서버에서 받은 Update 데이터를 그리는 함수

Recv\_check() Player가 모두 준비되었는지 데이터를 받는 함수

수신

Send\_Con\_data() 서버에 데이터 소켓을 전달하는 함수

Send\_input\_data() 입력 받은 데이터를 서버에 전달하는 함수

처리

Change\_status() 게임의 상태를 play, wait, end 로 바꿔주는 함수

Update() 입력받은 데이터를 확인해 최신화 하는 함수

Check\_state() 게임의 진행 상황을 서버가 확인하도록 요청하는 함수

Timeout() 일정시간 안에 입력 데이터의 전송이 되지 않고 지연 시 False로 처리하는 함수

* 변수

enum Item

{

EMPTY, // empty

BALLON, // 폭탄 개수 증가

SHOES, // 플레이어 이동 속도 증가

POTION, // 폭탄 길이 증가

};

enum PlayerColor

{

RED,

GREEN,

BLUE,

PLAYEREMPTY,

};

1. **팀원 역할**

|  |  |
| --- | --- |
| 이름 | 역할 |
| 윤승규 | 서버 송/수신 함수 및 클라이언트 송신 함수 구현 |
| 신동원 | 서버 프레임 워크 및 송신 함수 구현 |
| 김석현 | 클라이언트 프레임워크 및 수신함수 구현, 문서 작성 |

1. **개발 일정**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **11월** | | | | | | | |
| **구분** | **일** | **월** | **화** | **수** | **목** | **금** | **토** |
| **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **신동원** | **계획서 작성** | | | | | | |
| **김석현** |
| **윤승규** |
|  | **13** | **14** | **15** | **16** | **17** | **18** | **19** |
| **신동원** | **정기 회의** | **계획서 작성** | | **Message Thread 구현** | | **서버 수신함수 구현** | |
| **김석현** | **클라이언트 수신함수 구현** | | | **보완** |
| **윤승규** | **세부 함수 구현(1) – update()** | | | **보완** |
|  | **20** | **21** | **22** | **23** | **24** | **25** | **26** |
| **신동원** | **정기 회의** | **서버 송신함수 구현, Main Thread 구현** | | | | | **1차 병합** |
| **김석현** | **서버 – 클라이언트**  **접속 테스트** | | **문제점 보완 및**  **클라이언트 송신함수 구현** | | |
| **윤승규** | **Matching Thread 구현** | | |
|  | **27** | **28** | **29** | **30** | **1** | **2** | **3** |
| **신동원** | **정기 회의** | **Game Thread 구현**  **문제점 보완 및 세부 함수 구현** | | | | 보완 | 2차 병합 |
| **김석현** |
| **윤승규** | **세부 함수 구현** | | | |
|  | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **신동원** | **정기 회의** | 최종 테스트 및 수정, 개선, 검수 | | | 발표 |  | |
| **김석현** |
| **윤승규** |