네트워크 게임 프로그래밍

텀프로젝트 추진 계획서

1조: 윤우영, 김강휘, 김지호

목차

1. 애플리케이션 기획 p.3~4
2. High level design p.5~8
3. Low level design p.9~11
4. 개발 환경 p.12~13
5. Calendar p.14~6
6. 애플리케이션 기획

출처: 윤우영, SDL 라이브러리를 사용하여 제작

(C++로 2D게임을 제작한 팀원이 없어 이번학기에 자체 제작)

장르:

위에서 내려다보는 3인칭 슈팅 게임

Scene:

Main 화면:

Ip address, Port number, Nickname 입력하고 Connect 버튼 누름.

서버와 연결하면 매칭찾기 버튼이 생김

매칭 찾기를 누르고 3인이 모이면 카운트다운 후 In Game으로 전환

In game:

게임 맵과 캐릭터가 보이고 카운트다운이 시작된 후 게임이 시작

조종: w, a, s, d를 사용하여 움직이고 마우스를 사용하여 조준하고 총을 발사

무기: 권총(탄약 무한), 라이플(탄약 얻어야 함), 저격총(탄약 얻어야 함)

탄약에 가까이 가면 플레이어가 탄약(라이플(30발), 저격총(10발))을 얻음

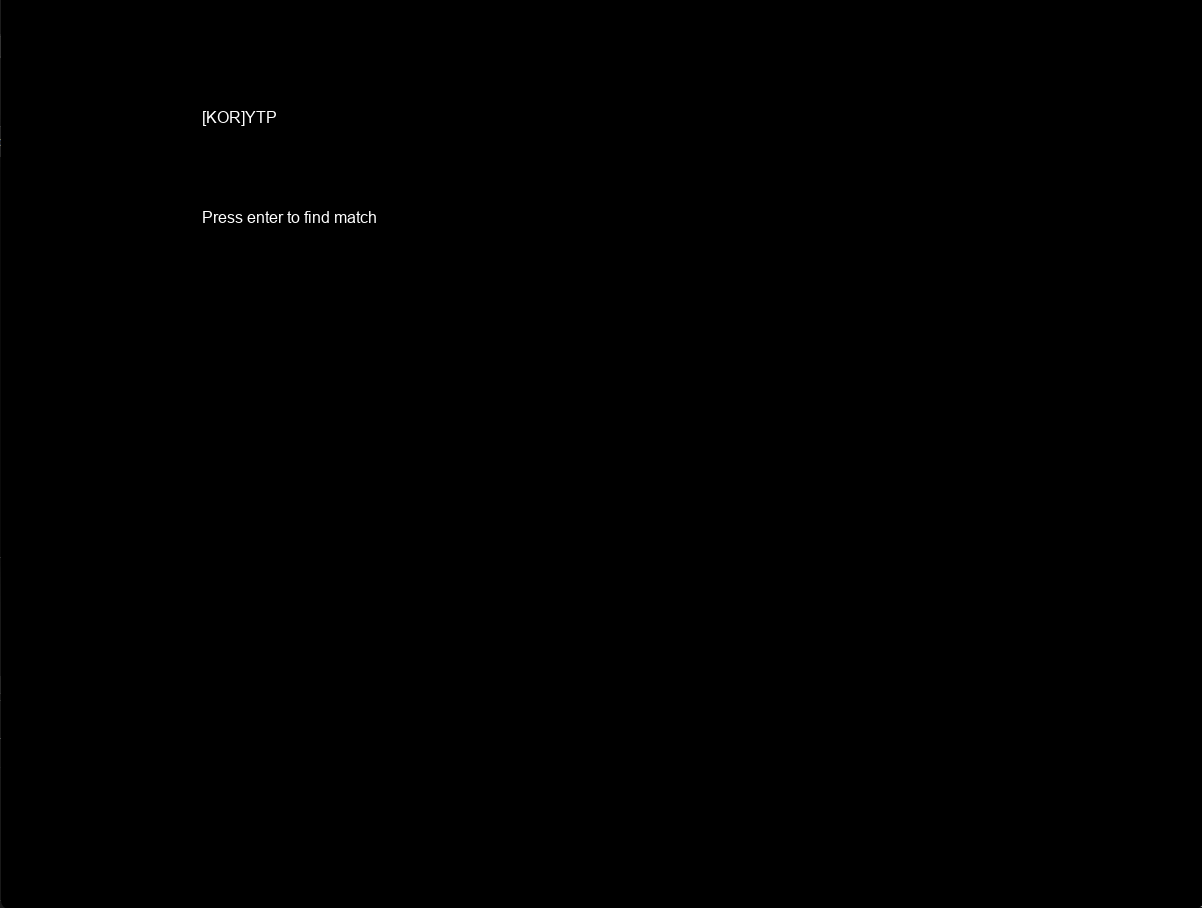
맵: 곳곳에 장애물들(벽, 드럼통, 박스, 바위)이 있고 탄약이 떨어져있음

캐릭터: 캐릭터는 원 모양을 하고 있고 플레이어마다 색깔이 다름

캐릭터 상단에 HP bar가 표시됨

HUD: 화면 하단의 무기슬롯에 3가지 무기가 있음

죽으면 Game over가 뜨고 아무 입력을 하면 메인화면으로 돌아감





1. High level design

Server Client

send\_name()

recv\_name()

send\_player\_input();

recv\_players\_state(player\_state p1, player\_state p2, player\_state p3);

recv\_player\_input();

send\_players\_state(player\_state p1, player\_state p2, player\_state p3);

send\_players\_information()

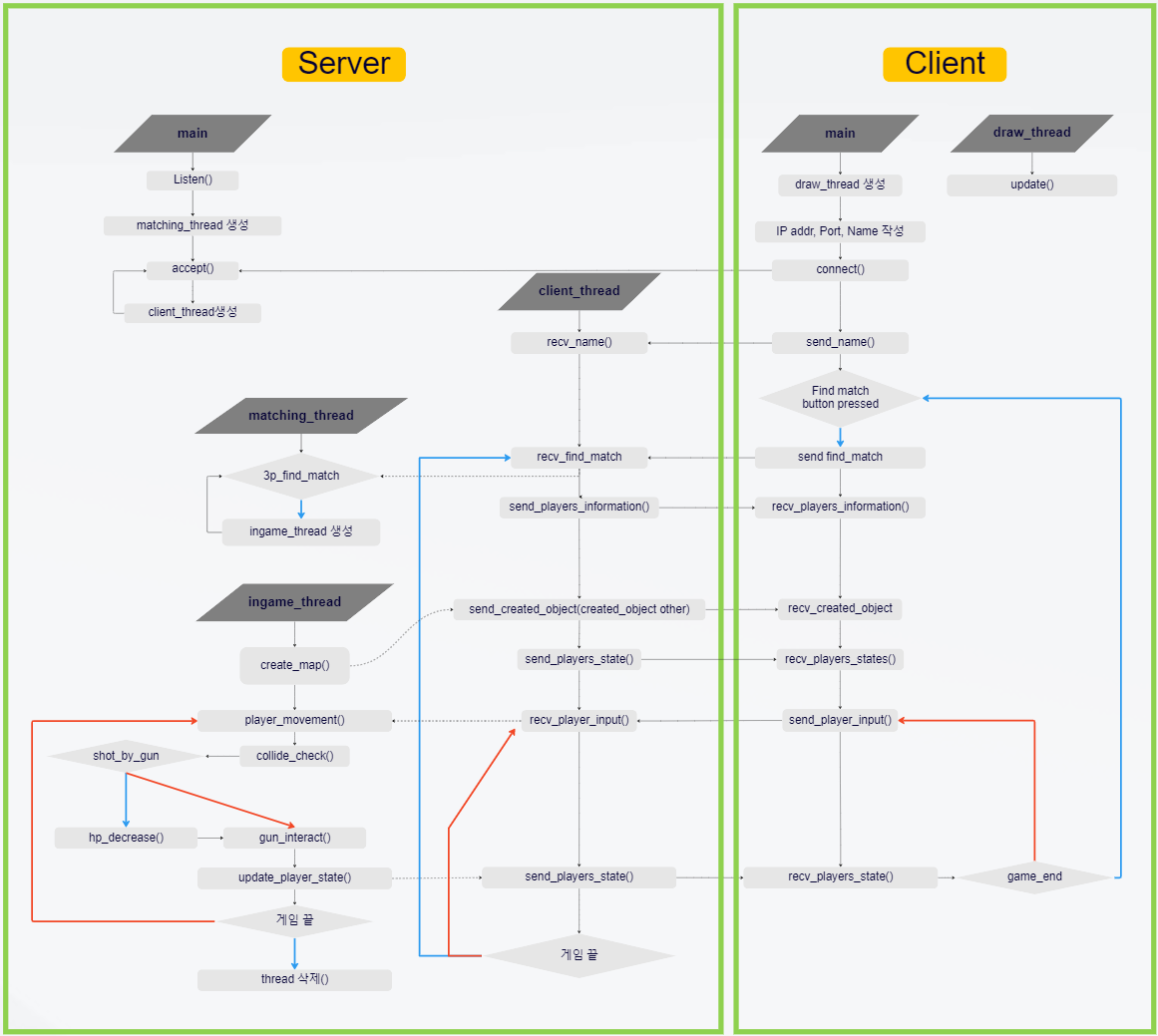
recv\_players\_information()

send\_find\_match(bool find\_match)

recv\_find\_match(bool find\_match)

recv\_created\_object(created\_object other)

send\_created\_object(created\_object other)



====SERVER====

main

1. Listen서버 생성
2. matching\_thread 생성
3. accept()
4. client\_thread생성
5. 3,4반복

client\_thread

1. recv\_name으로 name 받기
2. recv\_find\_match로 클라이언트가 match\_button을 눌렀는지 확인
3. 매칭시 send\_player\_information()으로 타 플레이어의 정보 송신
4. send\_create\_object로 모든 오브젝트 송신
5. send\_players\_state로 플레이어 상태 송신
6. recv\_player\_input로 플레이어 입력 감지
7. player\_input처리 후 send\_player\_state로 플레이어의 상태 배포
8. 게임이 끝날때까지 6,7 반복
9. 게임 끝날 시 2로 이동

matching\_thread

1. 3p\_find\_match에서 3명의 플레이어가 매칭 될때까지 대기
2. 매칭 될 시 ingame\_thread 생성후 1 이동

ingame\_thread

1. create\_map으로 맵 오브젝트 생성
2. player\_movement에 player 3명의 input값 대입 후 예상 경로를 변수에 저장
3. collide\_check에서 벽이랑 부딪치지 않는다면 2에서 계산한 변수값을 player\_state에 값을 변경해준다.
4. collide\_check에서 총알에 맞은 경우 he\_decrease에서 hp감소 후 6으로 이동
5. collide\_check에서 총알에 맞지 않은 경우 6으로 바로 이동
6. gun\_interact에서 플레이어 입력값과 연사 속도에 따라 계산 후 총알 생성
7. update\_player\_state에서 player 상태값 변경
8. 게임이 끝날때까지 2~7 반복
9. 게임이 끝날시 일정시간 후 스레드 삭제

====CLIENT====

1. draw\_thread 생성
2. IP addr, port, Name 작성
3. connect 시도
4. connect 성공시 send\_name으로 name 전송
5. Find match 버튼 활성화
6. Find match 버튼 클릭시 send find\_match로 버튼 눌렀다는 것을 서버에 전송
7. 이후 recv\_player\_information이 완료 될 때 까지 대기
8. recv\_create\_object로 모든 오브젝트 position와 type 변수에 저장
9. recv\_players\_states로 초기 위치 및 화면 전환
10. send\_player\_input으로 input 전송
11. recv\_player\_state로 player상태 값 수신
12. 게임이 끝날때까지 10,11반복
13. 게임이 끝날시 메인화면으로 이동후 find\_button 활성 상태로 이동

서버 동기화 작업: 게임에 참가하는 인원이 초과하지 않게 순서 부여

클라이언트가 아이템을 먹을 때

Winner 선정

Mutex를 사용하여 동기화를 구현할 것임

1. Low level design

\*표시는 기존에 없는 기능으로, 새로 제작해야 하는 것

|  |  |
| --- | --- |
| Server | |
| 함수 | 설명 |
| \*maching\_thread 생성, maching\_thread() 함수 | 3인이 매치를 잡기위한 처리 과정 쓰레드 |
| \*client\_thread 생성, client\_thread() 함수 | 클라이언트와 직접적으로 통신하는 쓰레드 |
| \*ingame\_thread 생성, ingame\_thread() 함수 | 인게임에서 처리할 로직을 담당하는 쓰레드 |
| \*void recv\_name() | 플레이어 이름 받기 |
| \*void send\_find\_match(bool findmatch) | 게임 찾기를 눌렀다는 정보를 서버에 송신 |
| \*void send\_players\_information()  struct player\_info  {  int player\_color[3];  char name[3][20];  } | 참가한 다른 플레이어들의 정보 전송 |
| \*void send\_created\_object(created\_object other)  struct create\_object  {  int object\_type;  TI object\_position;  } | 모든 오브젝트의 랜덤생성, 좌표 값과 오브젝트의 종류를 보내준다.  게임 매칭이 시작되면 모든 플레이어들에게 오브젝트 종류와 좌표 값을 송신. |
| \*void send\_player\_input();  struct player\_input{  bool w, a, s, d;  int gun\_type;  bool mouse\_clicked;  TF mouse\_state;  } | 클라이언트의 입력 값을 서버에 송신 |
| \*void send\_players\_state(player\_state p1, player\_state p2, player\_state p3);  struct player\_state{  int hp;  int gun\_type;  int bullet[3];  TI object\_position;  TF player\_position;  float player\_rotation;  bool gun\_fired;  int game\_state;  } | 플레이어의 좌표 값을 클라이언트에 송신, 플레이어가 아이템과 충돌 시 아이템 좌표 송신 |
| \*void create\_map() | 초기 맵 랜덤 생성하는 함수(바위, 벽, 아이템, 플레이어 색깔 등) |
| \*void update\_players\_state() | 클라이언트의 상태 값 배포하는 함수 |
| \*void player\_disconnect() | 플레이어가 프로그램 종료시 정보 삭제 및 쓰레드 종료 |
| \*void 3p\_find\_match | 3인이 매치를 잡게 하는 함수 |
| \*void player\_movement() | 플레이어 이동 작업 함수 |
| \*void shot\_by\_gun() | 총알 맞았는지 확인하는 함수 |
| \*void hp\_decrease() | hp를 감소시키는 함수 |
| \*void gun\_interact() | 총 쐈는지 확인하고, 연사 간격을 통해 총알의 좌표를 보내주는 함수 |
| \*void game\_end() | 게임이 끝났는지 확인하는 함수 |
| int timer(int start\_time, int delay); | 타이머 |
| double calc\_angle\_from\_points  (TF first\_point, TF second\_point); | 두 점 사이의 각도 계산 |
| void collide\_check(float a, float b, float c, float d, float e, float f, float g, float h); | 충돌체크 함수 |

|  |  |
| --- | --- |
| Client | |
| 함수 | 설명 |
| \*draw\_thread 생성, draw\_thread() 함수 | 화면에 그리는 기능을 하는 쓰레드 |
| \*void send\_name(char name[20]) | 플레이어 이름 보내기 |
| \*void recv\_find\_match(bool find\_match) | 클라이언트가 게임 찾기를 눌렀다는 정보를 수신 |
| \*void recv\_players\_information()  struct player\_info  {  int player\_color[3];  char name[3][20];  } | 타 플레이어들의 정보 전송 |
| \*void recv\_created\_object(created\_object other)  struct created\_object  {  int object\_type;  TI object\_position;  } | 모든 오브젝트의 랜덤생성, 좌표 값과 오브젝트의 종류를 수신  게임 매칭이 시작되면 모든 플레이어들에게 오브젝트 종류와 좌표 값을 수신. |
| \*void send\_player\_input();  struct input\_info {  bool w, a, s, d;  int gun\_type;  bool mouse\_clicked;  TF mouse\_state;  } | 클라이언트의 입력값을 송신 |
| \*void recv\_players\_state(player\_state p1, player\_state p2, player\_state p3);  struct player\_state {  int hp;  int gun\_type;  int bullet[3];  TI object\_position;  TF player\_position;  bool gun\_fired;  int game\_state;  } | 플레이어의 좌표 값을 서버로부터 수신, 서버로부터 사라진 아이템의 좌표를 수신. |
| \*void draw\_objects(); | 오브젝트 그리기 함수 |
| \*void delete\_object(); | map에서 오브젝트 삭제 함수 |
| void init\_variables(); | 변수 초기화 함수 |
| void init\_window(); | 윈도우 초기화 함수 |
| void clear\_renderer(); | 렌더 클리어 |
| void update\_renderer(); | 렌더 업데이트 |
| SDL\_Texture\* load\_image(const char\*); | 이미지 로드 |
| void load\_wavs() | 사운드 로드 |
| void load\_font() | 폰트 로드 |
| void key\_event\_ingame() | 인게임에서의 키 이벤트 |
| void mouse\_event\_menu() | 메뉴에서의 마우스 이벤트 |
| void mouse\_event\_ingame() | 인게임에서의 마우스 이벤트 |
| int timer() | 타이머 |
| void draw\_background() | 백그라운드 그리기 |
| void draw\_ground() | 땅 그리기 |
| void draw\_character(player\_state p1, player\_state p2, player\_state p3) | 캐릭터 그리기 |
| void draw\_healthbar() | 체력바 그리기 |
| void draw\_bullet() | 발사된 총알 그리기 |
| void draw\_flash() | 섬광 그리기 |
| void draw\_crosshair() | 크로스헤어 그리기 |
| void draw\_text(int x, int y, char[]) | 글씨 그리기 |
| void draw\_weapon\_list() | 무기 HUD그리기 |
| void draw\_menu(); | 메뉴 화면 그리기 |
| void draw\_in\_game() | 인게임 화면 그리기 |
| void update(); | 업데이트 |

1. 개발환경

운영체제: Windows

언어: C++

사용하는 통신 환경: TCP

1. Calendar

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 김강휘 | | | | | | |
| 11월 / 12월 | | | | | | |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  | 1 | 2 | 3 | 4  send\_find\_match()  recv\_find\_match() | 5 |
| 6  send\_players\_information()  recv\_players\_information() | 7 | 8 | 9 | 10 | 11 | 12 |
| 13  send\_player\_input()  recv\_player\_input() | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23  update\_players\_state() | 24 | 25 | 26 |
| 27  동기화 작업 | 28 | 29 | 30 | 1 | 2 | 3 |
| 4  플레이 하면서 오류 찾기 | 5 | 6 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 김지호 | | | | | | |
| 11월 / 12월 | | | | | | |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  | 1 | 2 | 3  send\_name  recv\_name | 4 | 5 |
| 6 | 7 | 8 | 9 | 10  create\_map() | 11 | 12 |
| 13 | 14 | 15 | 16 | 17  delete\_object() | 18 | 19 |
| 20 | 21 | 22 | 23 | 24  send\_players\_state();  recv\_players\_state(); | 25 | 26 |
| 27 | 28 | 29 | 30 | 1  player\_disconnect() | 2 | 3 |
| 4  플레이 하면서 오류 찾기 | 5 | 6 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 윤우영 | | | | | | |
| 11월 / 12월 | | | | | | |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  | 1 | 2 | 3 | 4 | 5 |
| 6  게임 로직 server로 이동 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13  send\_created\_object()  recv\_created\_object() | 14 | 15 | 16 | 17 | 18 | 19 |
| 20  collide\_check(); | 21 | 22 | 23 | 24 | 25 | 26 |
| 27  end\_game(); | 28 | 29 | 30 | 1 | 2 | 3 |
| 4  플레이 하면서 오류 찾기 | 5 | 6 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 윤우영 | | | | | | |
| 11월 / 12월 | | | | | | |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  |  | 9  서버 listen\_thread 만들고 client와 연결 | 10  send\_find\_match(), | 11  recv\_find\_match() | 12 |
| 13  주간 작업 오류 및 수정 사항 개선 | 14 | 15  send\_players\_information() | 16  recv\_players\_information() | 17  send\_created\_object() | 18  recv\_created\_object() | 19 |
| 20  주간 작업 오류 및 수정 사항 개선 | 21 | 22 | 23  player\_movement() | 24  gun\_interact() | 25  shot\_by\_gun() | 26 |
| 27  주간 작업 오류 및 수정 사항 개선 | 28 | 29  game\_end() | 30  게임끝날 때 thread 정리 | 1 | 2  player\_disconnect() | 3 |
| 4  플레이 하면서 오류 찾기 | 5  플레이 하면서 오류 찾기 | 6 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 김강휘 | | | | | | |
| 11월 / 12월 | | | | | | |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  |  | 9 | 10  클라이언트의 draw\_thread 생성 | 11) | 12 |
| 13  주간 작업 오류 및 수정 사항 개선 | 14  matching\_thread 생성 | 15 | 16  3p\_find\_match() | 17  send\_players\_state() | 18  recv\_players\_state() | 19  timer 함수 총 종류별 변경 |
| 20  주간 작업 오류 및 수정 사항 개선 | 21  클라이언트 내의 인게임 함수  서버로 옮기기 | 22 | 23 | 24  collide\_check() (사물과 플레이어) | 25 | 26 |
| 27  주간 작업 오류 및 수정 사항 개선 | 28  hp\_decrease() | 29 | 30 | 1.  플레이어가 아이템 먹을 때,  동기화- 필요없음 | 2 | 3 |
| 4  플레이 하면서 오류 찾기 | 5  플레이 하면서 오류 찾기 | 6 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 김지호 | | | | | | |
| 11월 / 12월 | | | | | | |
| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|  |  |  | 9 | 10  Client\_thread 생성 , | 11  send\_name() | 12 |
| 13  주간 작업 오류 및 수정 사항 개선 | 14  recv\_name() | 15  ingame\_thread 생성 | 16  유저정보 구조체 | 17  create\_map( ) - 랜덤 생성 | 18  create\_map( ) - object 겹치지 않게 보완 | 19 |
| 20  주간 작업 오류 및 수정 사항 개선 | 21 | 22  send\_player\_input(), | 23  recv\_player\_input() | 24 | 25  collide\_check() (플레이어와 총알) | 26 |
| 27  주간 작업 오류 및 수정 사항 개선 | 28 | 29  update\_players\_state() | 30 | 1. | 2  총을 맞았을 때 동기화 작업 | 3 |
| 4  플레이 하면서 오류 찾기 | 5  플레이 하면서 오류 찾기 | 6 |  |  |  |  |

**계획 수정사항**

|  |  |
| --- | --- |
| \*void send, recv\_players\_information()  struct player\_info  {  int player\_color[3];  char name[3][20];  } | typedef struct players\_info  {  char name[3][20]{ 0 };  }PI;  int player\_color[3]: 색 할당의 필요성이 없어서 삭제 |
| \*void send, recv\_created\_object(created\_object other)  struct created\_object  {  int object\_type;  TI object\_position;  } | typedef struct created\_object  {  int object\_type;  TI object\_position;  }CO; |
| \*void send, recv\_player\_input();  struct input\_info {  bool w, a, s, d;  int gun\_type;  bool mouse\_clicked;  TF mouse\_state;  } | typedef struct client\_input  {  bool w\_Pressed;  bool a\_Pressed;  bool s\_Pressed;  bool d\_Pressed;  int gun\_type;  bool clicked;  float mouse\_rotation;  float clicked\_mouse\_rotation;  TF unconditional\_fired\_pos\_input{};    }CI;  clicked\_mouse\_rotation: 발사했을 때의 각도를 저장  unconditional\_fired\_pos\_input: |
| \*void send, recv\_players\_state(player\_state p1, player\_state p2, player\_state p3);  struct player\_state {  int hp;  int gun\_type;  int bullet[3];  TI object\_position;  TF player\_position;  bool gun\_fired;  int game\_state;  } | typedef struct players\_state  {  int hp = 100;  int gun\_type{};  int bullet[3]{};  TF velo{};  TI object\_position{};  TF player\_position{};  float player\_rotation{};  bool gun\_fired = false;  int game\_state{};  bool collide = false;  }PS;  Velo: collide check에서 플레이어의 속도를 제어하기 위해  Collide: 총알이 충돌하여 사라지는 것을 클라이언트에 전송 |

typedef struct players\_profile {

int room\_num = 0;

PI player\_info;

PS player\_state;

CI input;

TF unconditinal\_fired\_pos;

TF bullet\_pos{};

}PP;

플레이어의 모든 정보를 map에서 관리하기 위해 만든 구조체

map<int, PP\*> player\_list;

서버에 연결된 모든 클라이언트의 port번호, 플레이어 정보로 이루어진 map

**SERVER**

**character\_movement(CI, TF&, TF&) 클라이언트의 입력을 받고 플레이어의 위치를 서버에서 계산**

**collide\_check(PP\*, CI\*) 플레이어의 위치를 이용해서 서버에서 충돌처리 계산**

**bullet\_movement(float, PP\*) 클라이언트에서 총을 발사한 신호를 보내면 서버에서 총알의 움직임을 계산**

**rand\_player\_pos(PP\*) 플레이어의 생성 위치를 랜덤으로 만듦**

**player\_list 동기화 (중간에 나가는 플레이어가 있으면 map에서 삭제)**

**CLIENT**

**draw\_thread 대신 통신 쓰레드인 server\_thread 생성**

**draw\_log() 왼쪽 상단에 플레이어들의 상태를 표시**

**drawEngGame() 게임이 끝나면 승패자 목록 띄움**