

Perfect Shape

팀원: 이도영, 유재우 이도

목차

1. 문서 개요.....	2
1.1. 기획서 개요.....	3
1.2. 개발 환경.....	4
1.3. 개인별 역할 분담.....	5
1.4. 개발 일정.....	6
2. 게임 개요.....	8
2.1. Perfect Shape 개요.....	8
3. 게임 플레이.....	9
3.1. 게임 플레이 개요.....	9
4. Level Design.....	10
4.1. High - Level Design	10
4.2 Logine Flow Chart.....	11
4.3 Game Flow Chart.....	12
4.4 Code Flow Description.....	13
4.5 Low – Level Design.....	14
4.5.1 Protocol.....	14
4.5.2 함수.....	17
5. History.....	18

1. 문서 개요

1.1 기획서 개요

네트워크 게임 프로그래밍 프로젝트를 만들기 위한 기획서

이미 만들어진 게임을 멀티게임으로 바꾸는 것으로 서버 개발에 중점을 둔 기획서.

1.2 개발 환경

- OS : Windows 10
- IDE : Visual Studio 2019
- API : Win32 API / Windows Socket API
- 네트워크 IO모델 : 다중 스레드 모델
- 언어 : C / C++ (Open GL)

1.3 개인별 역할 분담

1. 이도영

- void send_login_packet() : 클라이언트가 접속하면 접속확인과 id 를 보내는 함수
- void send_add_packet() : 다른 클라이언트의 접속 전송 함수
- void send_start_packet() : 게임이 시작했다는 정보를 보내는 함수
- void send_remove_packet() : 죽은 플레이어 삭제 함수
- void Disconnect() : 플레이어 연결 종료 함수

2. 유재우

게임로직 수정

- void send_move_packet() : (클라이언트) 키 입력(W, A, S, D) 송신 함수
- void send_attack_packet() : 마우스 좌(attack) 클릭 송신 함수
- void send_move_packet() : (서버)클라이언트의 움직임 전송 함수
- void send_bullet_packet() : 총알의 위치 값 전송 함수
- int NetInit() : 서버 접속 함수
- do_recv() : (클라)서버가 전송한 패킷 수신 함수

3. 이도

- bool player_collide() : 플레이어 충돌 판단 함수
- bool enemy_collide() : 적 충돌 판단 함수
- Send_all(): (서버)데이터 송신 함수.
- void process_packet(); 패킷 재 조립 함수
- void GameStart() : 게임이 시작 및 정보 초기화 함수

1.4 개발 일정

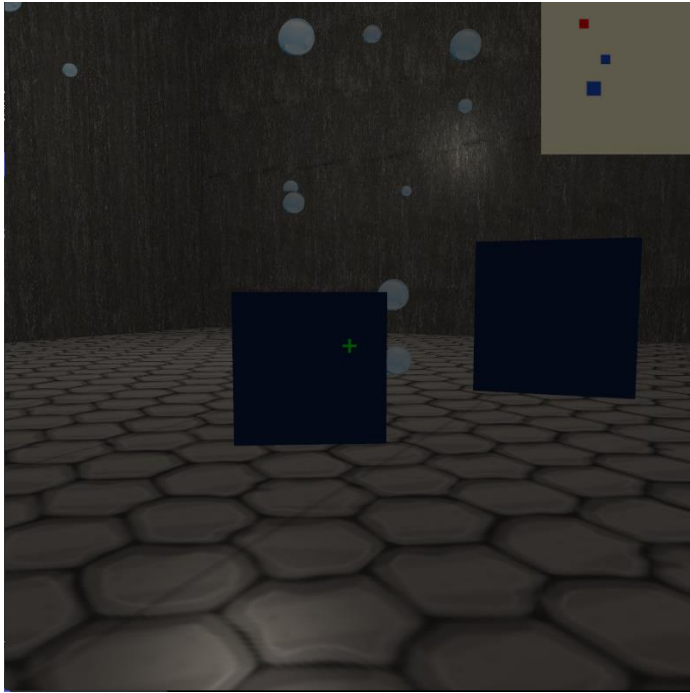
	유재우	이도	이도영
10/22(토)	전체 기획 회의 코드분석, 클라이언트 추가 사항, Git 생성 및 clone, 게임 플로우 작성		
10/26(수)	코드 리뷰 원래 게임 변경 Break Down -> Perfect Shape		
10/29(토)	코드 리뷰(서버와 클라에 들어갈 코드 나눔) protocol 생각		
10/30(일)	High Level Design		
11/1(화)	Low Level, Game Flow Chart, Code Flow Description, 문서 작성		
11/2(수)	High Level Design 추가일정 작성 및 문서 최종 점검		
11/3(목)	문서 제출		
11/4(금)			Send_login_packet()클라이언트가 접속하면 접속확인과 id를 보내는 함수
11/5(토)	NetInit()서버 접속 함수		
11/6(일)		Send_all()(서버) 데이터 수신 송신 함수	
11/7(월)			
11/8(화)	do_recv() : (클라)서버가 전송한 패킷 수신 함수		Send_add_packet()다른 클라이언트의 접속 전송 함수
11/9(수)	클라 서버 연결 확인		
11/10(목)			
11/11(금)	일정 점검 부족한 부분 수정		
11/12(토)	void send_move_packet() : (클라이언트) 키 입력(W, A, S, D) 송신 함수 void send_move_packet() : (서버)클라이언트의 움직임 전송 함수		
11/13(일)		void process_packet(); 패킷 재 조립 함수	
11/14(월)			
11/15(화)	게임로직수정		void send_remove_packet() 죽은 플레이어 삭제 함수
11/16(수)		void GameStart() : 게임이 시작 및 정보 초기화 함수	
11/17(목)			void send_start_packet() : 게임이 시작했다는 정보를 보내는 함수
11/18(금)	일정 점검 부족한 부분 수정		
11/19(토)	void send_attack_packet() : 마우스 좌(attack) 클릭 송신 함수	bool player_collide() : 플레이어 충돌 판단 함수	
11/20(일)			

11/21(월)			void Disconnect() : 플레이어 연결 종료 함수
11/22(화)	게임로직수정		
11/23(수)		bool player_collide() : 플레이어 충돌 판단 함수	
11/24(목)			
11/25(금)	일정 점검 부족한 부분 수정		
11/26(토)	void send_bullet_packet() : 총알의 위치 값 전송 함수	bool enemy_collide() : 적 충돌 판단 함수	
11/27(일)			
11/28(월)			
11/29(화)			
11/30(수)		bool enemy_collide() : 적 충돌 판단 함수	
12/1(목)			
12/2(금)	일정 점검 부족한 부분 수정		
12/3(토)	게임 검증		
12/4(일)	게임 검증		
12/5(월)	게임 검증		
12/6(화)	최종 점검		
12/7(수)			
12/8(목)			

2. 게임 개요

2.1 Perfect Shape 개요

플레이어 3 명에서 캐릭터를 조종하여 적을 쓰러트리며 다음 방으로 진행하며 지속적으로 살아남는 1 인칭 슈팅게임



<게임화면>: 검정색 정육면체가 적이다.

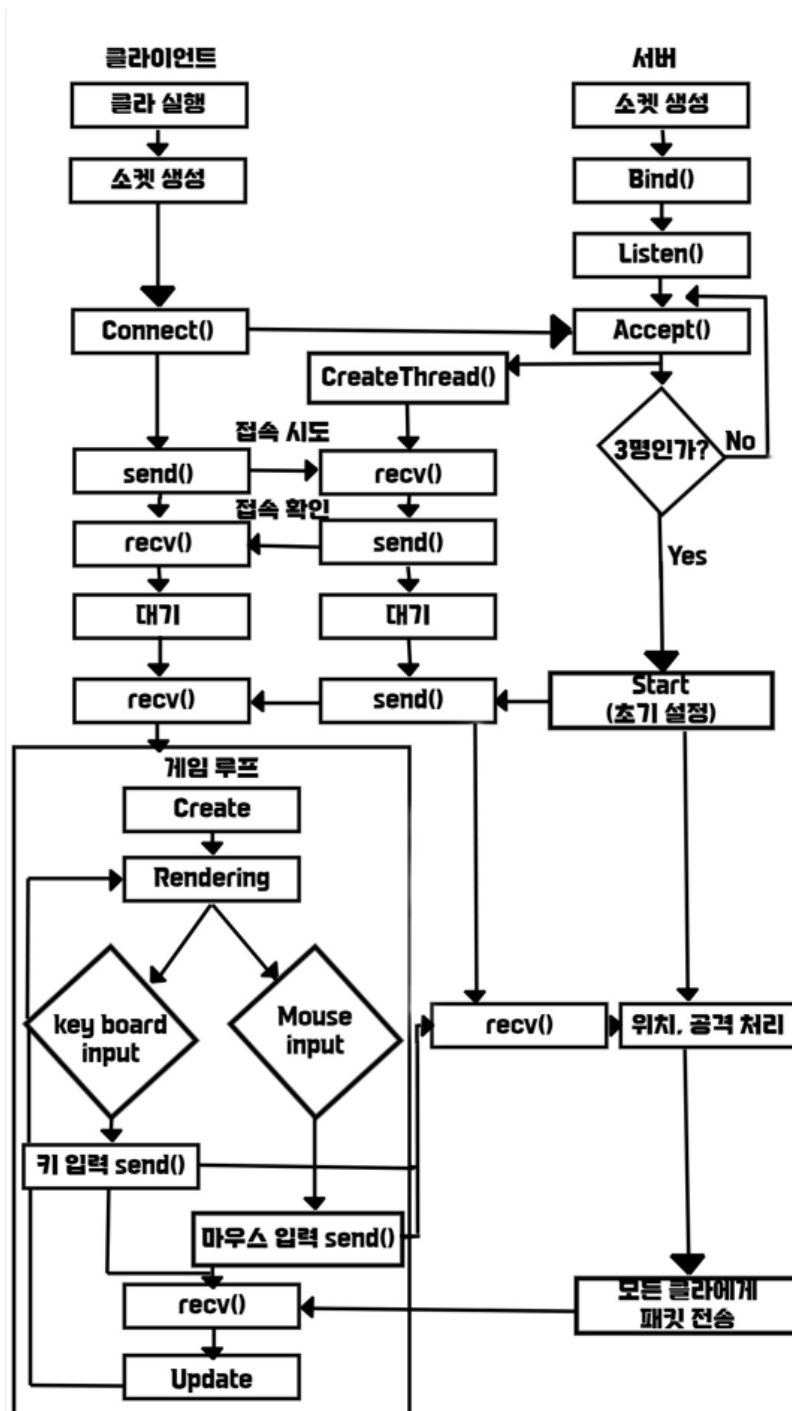
3. Game Play

3.1 게임플레이

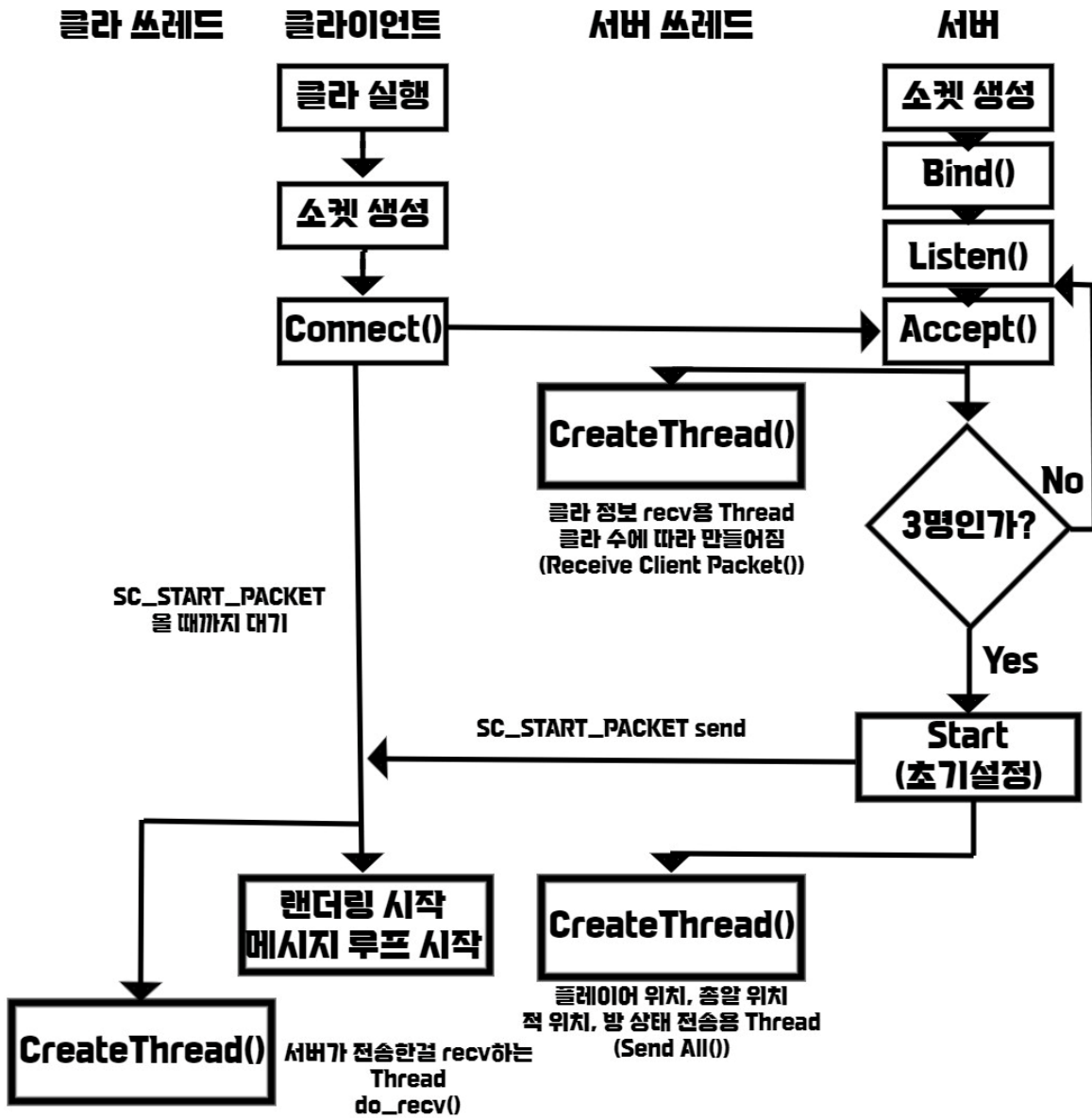
- WSAD(상하좌우) 키보드를 통해 입력을 받아 플레이어를 움직일 수 있음.
- 마우스 왼쪽 커서를 통해 적을 공격
- 적에게 충돌하거나 적의 총알에 충돌할 경우 데미지를 입음.
- 게임의 흐름
 1. 스테이지 시작
 2. 모든 적 처치
 3. 다음 방 이동 (다음 스테이지 시작)

4. Level Design

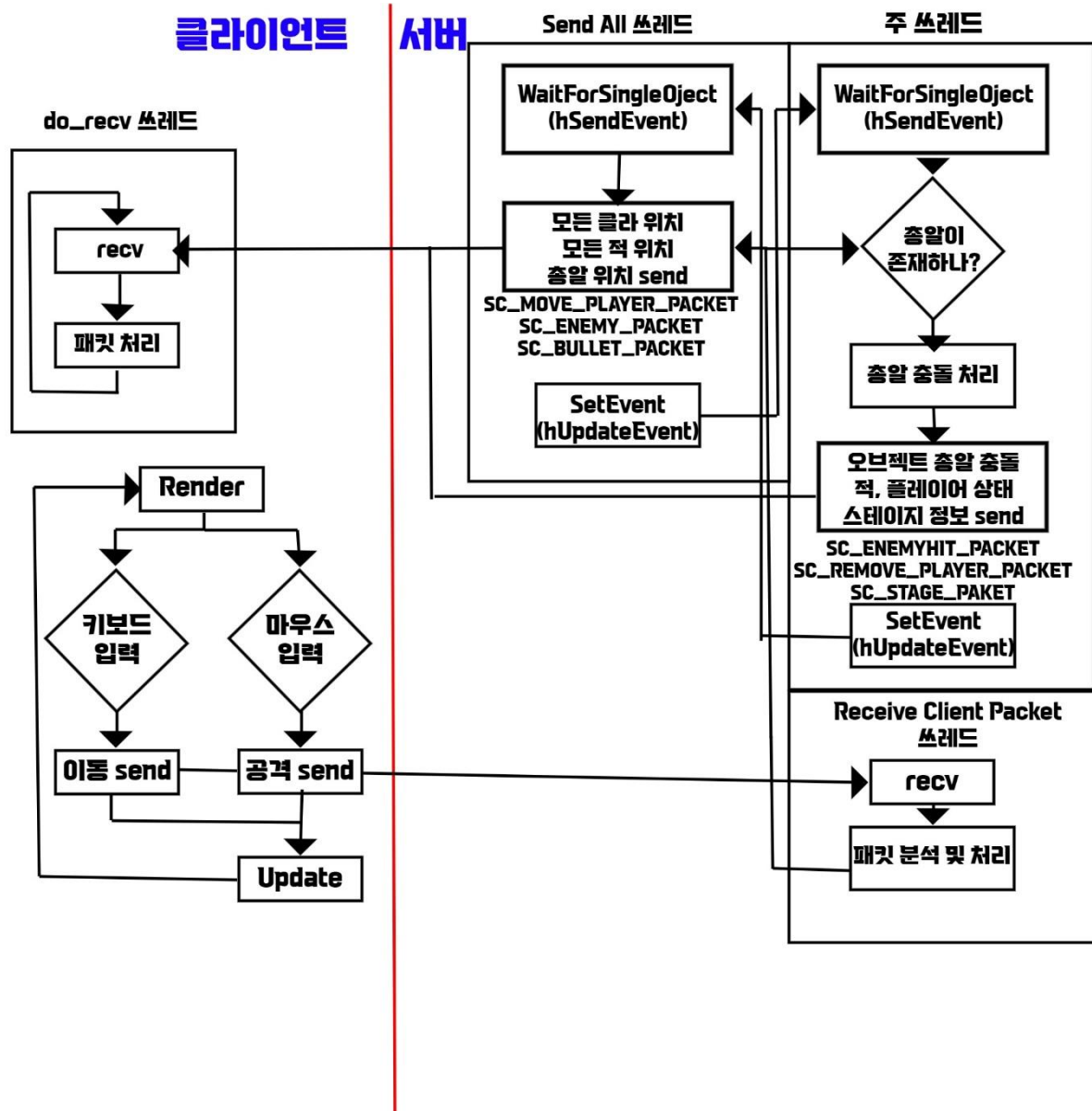
4.1 High – Level Design



4.2 Login Flow Chart



4.3 Game Flow Chart



4.4 Code Flow Description

- 접속
 - 클라이언트 실행 시 로그인 GUI 생성 서버로 접속 요청을 보냄
 - 서버는 접속요청을 받고 accept가 되면 해당 클라이언트를 위한 쓰레드를 만듦.
 - 그 후 해당 클라이언트에게 id를 부여하고 id와 함께 접속이 되었다는 의미의 패킷을 보내줌.
 - 이미 접속한 다른 클라이언트가 있다면 새로 접속한 클라이언트에게 이미 접속해 있는 클라이언트의 ID정보를 담은 패킷을 보내고, 이미 접속해 있는 클라이언트에게는 새로 접속한 클라이언트의 ID정보를 담은 패킷을 보내줌.
 - 서버는 3명의 클라이언트가 접속하면 게임시작을 위한 정보 초기화 (플레이어의 위치)를 해주고 게임이 시작되었다는 패킷을 클라이언트에게 보내줌.
 - 게임을 시작하면 send를 해주는 쓰레드를 만들고 메인 쓰레드는 플레이어와 적의 위치, 공격, 충돌 처리를 해줌.
- 게임 시작
 - 게임 시작 시 서버에서 게임이 시작되었다는 패킷을 전송하고 클라이언트에서 수신 받으면 게임 GUI 렌더링을 해줌
 - 움직임이나 공격을 위한 입력이 있다면 서버에 해당 키를 send()해주고, 서버는 알맞은 처리를 해줌
 - 서버에서 대부분 계산처리 작업을 진행하기에 충돌처리를 서버에서 진행할 예정
성능적인 부분은 3인 게임에서 큰 차이는 없을 거라 생각하였음.
 - 동기화방식은 이벤트 방식을 사용할 예정
크리티컬 섹션에 비해서 성능적인 이점을 취할 수 있기에 채택함.

4.5 Low – Level Design

4.5.1 Protocol

- 클라이언트에서 플레이어 로그인 시 보내는 패킷

```
struct CS_LOGIN_PACKET {  
    unsigned char size;  
    char    type;  
    char    name[NAME_SIZE];  
};
```

- 클라이언트에서 플레이어 이동시 보내는 패킷

```
struct CS_MOVE_PACKET {  
    unsigned char size;  
    char    type;  
    char    direction; // 0 : UP, 1 : DOWN, 2 : LEFT, 3 : RIGHT  
};
```

- 클라이언트에서 플레이어 마우스 클릭 시 보내는 패킷

```
struct CS_MOUSECLICK_PACKET {  
    unsigned char size;  
    char type;  
    short id; // 클라이언트 아이디  
    float dx, dy, dz; // 시선 벡터  
};
```

- 처음 접속했을 때 접속한 클라이언트 아이디 패킷

```
struct SC_LOGIN_INFO_PACKET {  
    unsigned char size;  
    char    type;  
    short    id;  
};
```

- 접속한 클라이언트의 정보 패킷

```
struct SC_ADD_PLAYER_PACKET {  
    unsigned char size;  
    char    type;  
    short    id;  
};
```

- 게임시작시 알림 패킷

```
struct SC_START_PACKET {
    char size;
    char type;
};
```

- 플레이어 사망할 시 없애는 패킷

```
struct SC_REMOVE_PLAYER_PACKET {
    unsigned char size;
    char type;
    short id;
};
```

- 플레이어 이동시 위치 패킷

```
struct SC_MOVE_PLAYER_PACKET {
    unsigned char size;
    char type;
    short id;
    float x, y, z;
};
```

- 적 패킷

```
struct SC_ENEMY_PACKET {
    unsigned char size;
    char type;
    char state;
    short id;
    float x, y, z;
};
```

-적 피격 패킷

```
struct SC_ENEMYHIT_PACKET {
    unsigned char size;
    char type;
    short hp;
    short id;
};
```

- 총알 패킷

```
struct SC_BULLET_PACKET {
    unsigned char size;
    char type;
    short bullet_id;
    float x,y,z;
};
```

- 총알 피격 패킷

```
struct SC_BULLETHIT_PACKET {
    unsigned char size;
    char type;
    short bullet_id;
};
```

- 스테이지 패킷

```
struct SC_STAGE_PACKET {
    unsigned char size;
    char type;
    short state;
};
```

상수

- 클라이언트와 서버 송수신간 보내는 패킷 타입

```
// Packet ID
constexpr char CS_LOGIN = 0;
constexpr char CS_MOVE = 1;

constexpr char SC_LOGIN_INFO = 3;
constexpr char SC_ADD_PLAYER = 4;
constexpr char SC_START = 5;
constexpr char SC_REMOVE_PLAYER = 6;
constexpr char SC_MOVE_PLAYER = 7;
constexpr char SC_ENEMY = 8;
constexpr char SC_BULLET = 9;
```

송수신 시 보내는 패킷 타입을 정의하여 서버에서 어떤 패킷을 받았는지 알 수 있게 한다.

기타

```
constexpr int PORT_NUM = 9000;
constexpr int BUF_SIZE = 256;
constexpr int NAME_SIZE = 20;
```

- 포트번호, 사이즈, 정의

4.5.2 함수

클라이언트

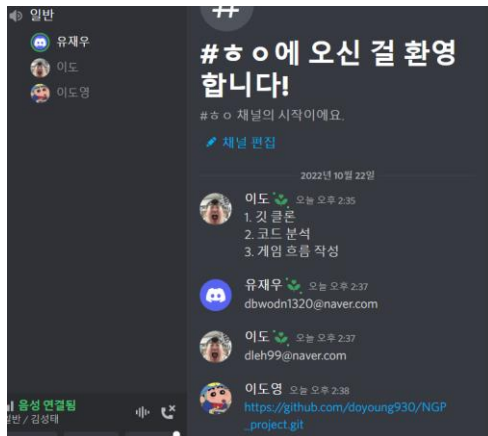
- void send_move_packet() : 키 입력(W, A, S, D) 송신 함수
- void send_attack_packet() : 마우스 좌(attack) 클릭 송신 함수
- int NetInit() : 서버 접속 함수
- do_rcv() : 서버가 전송한 패킷을 받고 패킷을 처리하는 쓰레드 함수
- bool player_collide() : 플레이어 충돌 판단 함수
- bool enemy_collide() : 적 충돌 판단 함수

서버

- 클라이언트 전송 패킷 수신 함수
- void process_packet();
- void GameStart() : 게임이 시작 및 정보 초기화 함수
- void send_login_packet() : 클라이언트가 접속하면 접속확인과 id를 보내는 함수
- void send_add_packet() : 다른 클라이언트의 접속 전송 함수
- void send_start_packet() : 게임이 시작했다는 정보를 보내는 함수
- void send_remove_packet() : 죽은 플레이어 삭제 함수
- void Disconnect() : 플레이어 연결 종료 함수
- void send_move_packet() : 클라이언트의 움직임 전송 함수
- void send_bullet_packet() : 총알의 위치 값 전송 함수
- void Send_all() : 서버데이터 송신 함수

6. HISTORY

2022 10/22 14:30



<Discord를 통한 비 대면 회의>

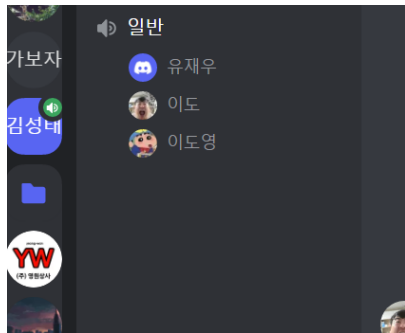
- 코드분석
- 클라이언트 추가 사항
- Git 생성 및 clone
- 게임 플로우 작성

2022 10/26

넷겜플 수업시간 코드 분석

➔ 게임 변경 Break Down -> Perfect Shape

1029회의 Discord를 통한 온라인 회의



1. 주석

2. 서버 클라 코드 나누기

3. protocol 생각하기

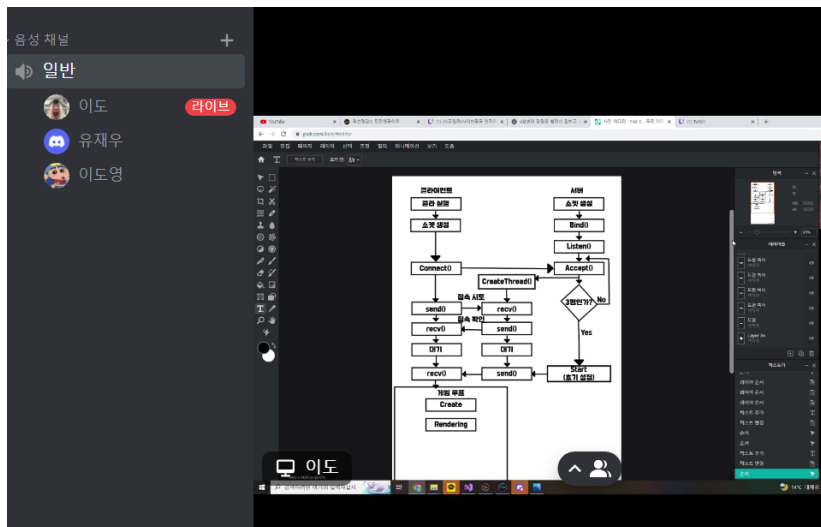
맵 -> 전부?

- Server :: Collide
- Server :: State -> 열리는지 닫히는지
- Server :: Open_door
- Server :: pos
- Enemy -> ADD -> id 값만 받아오면 된다. -> 각 클라 뿌려준다.
 - Enemy protocol 만들어야 함.
 - ◆ 이동 -> 플레이어 따라간다.
 - ◆ Type -> attack
 - ◆ State -> 체력
 - ◆ EnemyBulletPos
 - ◆ EnemyBulletNum
 - ◆
- Player
 - Protocol

- ◆ Key_input
- ◆ PlayerBulletPos
- ◆ PlayerBulletNum -> 배열로 함. 20발
- ◆

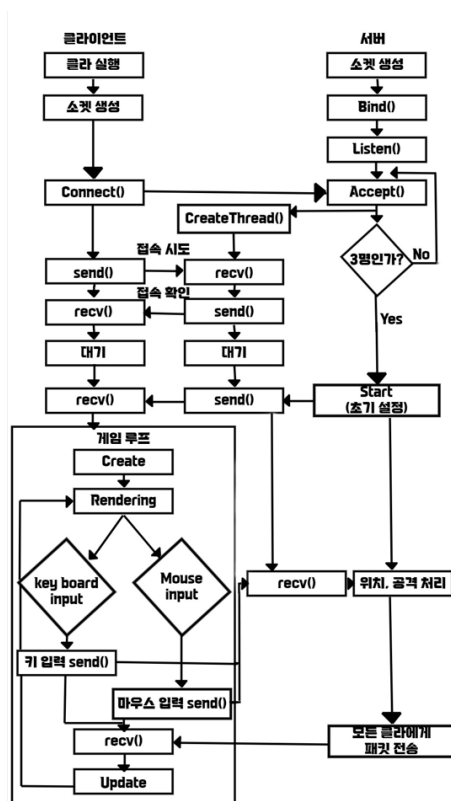
행렬 -> protocol

10/30 일 discord를 이용한 온라인 회의

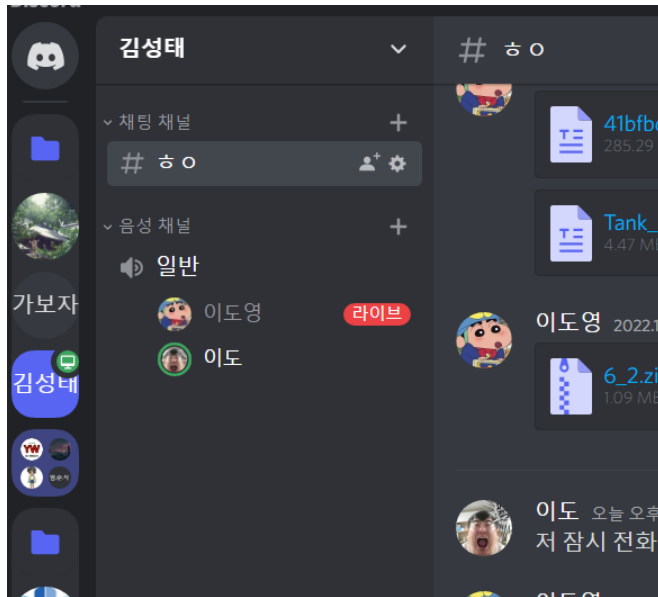


회의 내용

1. High Level Design



11/01 월 discord를 이용한 온라인 회의



1. Low level

2. Level Design에 따른 코드 흐름도 작성

역할 분담

(서버) 플레이 관련 전송

(서버) 쓰레드 생성 -

(서버) 스레드간 동기화(Event)

(서버) 플레이어 총알 충돌처리

(서버) 클라이언트 연결 -

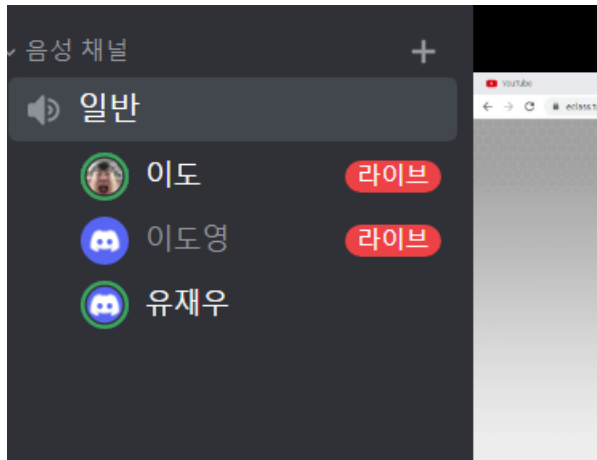
(서버) 게임시작과 접속 관련 전송

(클라이언트) 메인 함수 세팅

(클라이언트) 총알 구현

(클라이언트) 네트워크 연결 및 전송 함수

11/02 화 discord를 이용한 온라인 회의



1. Game Flow Chart

2. Login Flow Chart

3. 개발일정