2022 네트워크 게임 프로그래밍

Project Progress Report

이름: 강현석

학번: 2018180002

이름: 김성태

학번: 2018182007

이름: 김호준

학번: 2018182011

2022 November 7

**목 차**

**1. 애플리케이션 기획** ··························································· **2P**

1) 소개 ············································································································ 2P

2) 게임의 구성 ···························································································· 2P

3) 개발 환경 ································································································ 3P

**2. High Level Design** ··························································· **5P**

1) 전체 흐름도 ···························································································· 5P

2) 로그인 흐름도 ······················································································· 7P

3) 인 게임 흐름도 ··················································································· 10P

4) Server/Client Thread ········································································· 14P

**3. Low Level Design** ··························································· **16P**

1) 전역 상수 ······························································································ 16P

2) Client UML ···························································································· 21P

3) Server UML ··························································································· 26P

4) 패킷 ········································································································· 29P

**4. 팀원 별 역할분담** ··························································· **32P**

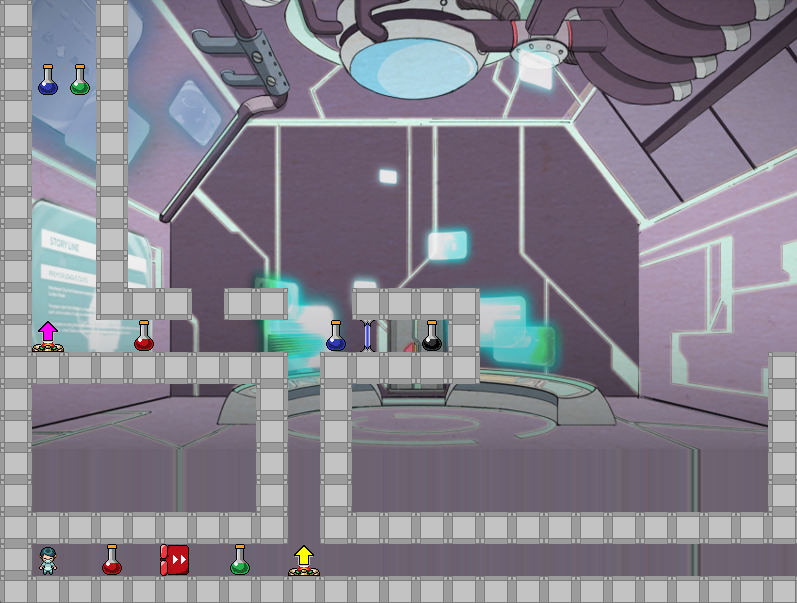
**5. 개발 일정** ··········································································· **33P**

**1. 애플리케이션 기획**

**1) 소개**

* 게임 제목 – Colorpher
* 게임 장르 – 2D 횡스크롤 멀티플레이 퍼즐 플랫포머
* 게임 방식 – 맵에 존재하는 장애물을 피해 포션을 전부 획득하여 5개의 라운드를 가장 먼저 클리어하는 플레이어가 우승하는 게임

**2) 게임의 구성**



[그림 1] 실제 구현되어 있는 게임의 실행 화면

1. 오브젝트  
    포션 – 플레이어의 색상을 포션과 동일하게 바꿔준다. 이미 플레이어의 색상이 3원색 중 하나라면, 포션의 색상과 섞이게 된다. 또한, 이미 두가지의 색상이 섞인 상태라면, 포션과 동일한 색으로 변한다. 다음은 빨간 포션을 획득하였을 때 변화이다.

EX) 플레이어의 색상 없음 -> 빨간색으로 변함

EX2) 플레이어의 색상 초록 -> 노란색으로 변함

EX3) 플레이어의 색상 노랑 -> 빨간색으로 변함

 점프대 – 플레이어와 점프대의 색상이 동일할 때 플레이어는 높게 점프할 수 있다.

 레이저 – 플레이어가 특정한 색일 때 작동하여 플레이어가 지나갈 수 없게 한다. 만약 억지로 지나가려 할 경우, 플레이어는 사망하게 된다.

 관문 – 플레이어와 관문의 색상이 동일할 때 지나갈 수 있다.

 스포이트 – 현재 플레이어의 색상을 저장한다. 이후 원할 때 저장해둔 색상으로 플레이어를 변화시킬 수 있다.

 권총 – 일정 시간마다 랜덤하게 생성된다. 획득하면 총알을 한 발 발사할 수 있으며, 이 총알에 맞은 플레이어는 라운드 리셋이 된다.

총알 – 권총에 의해 발사된다. 플레이어가 발사한 방향으로 일정 속도로 이동한다. 벽을 뚫고 지나간다.

1. 조작법

방향키 – 이동

Space Bar – 점프

R – 게임 준비/게임 재시작

F – 스포이트 아이템 사용

A – 권총 발사

ESC – 게임 종료

Enter – 로그인

**3) 개발 환경**

1. 사용한 프로젝트

“Colorpher”는 팀장 강현석의 2D 게임 프로그래밍 최종 프로젝트로 만들어진 게임이다. 이 게임을 팀원 김성태가 SFML API를 사용하여 Visual C++환경으로 옮겨 놓았다.

1. 언어

C++을 기반으로 개발한다.

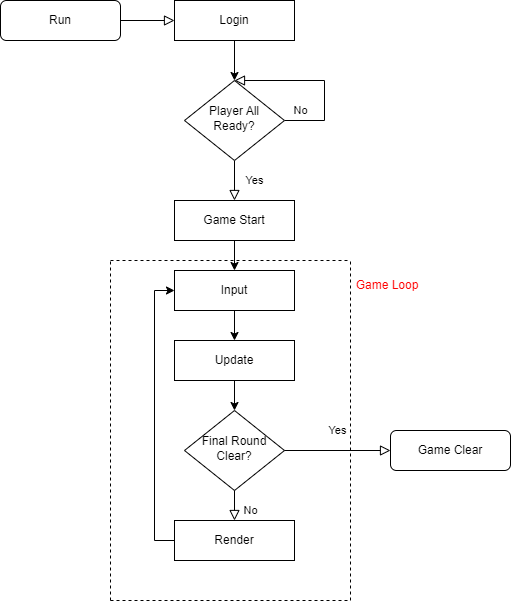
1. 개발 툴

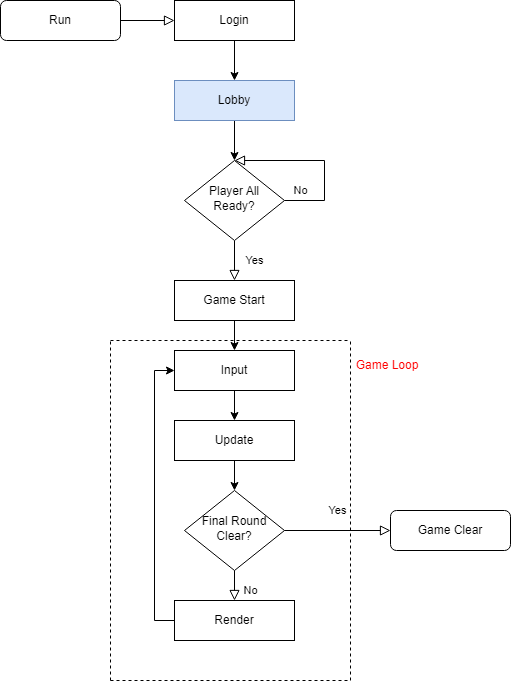
Visual Studio 2022를 사용하여 개발한다.

**2. High Level Design**

**1) 전체 흐름도**

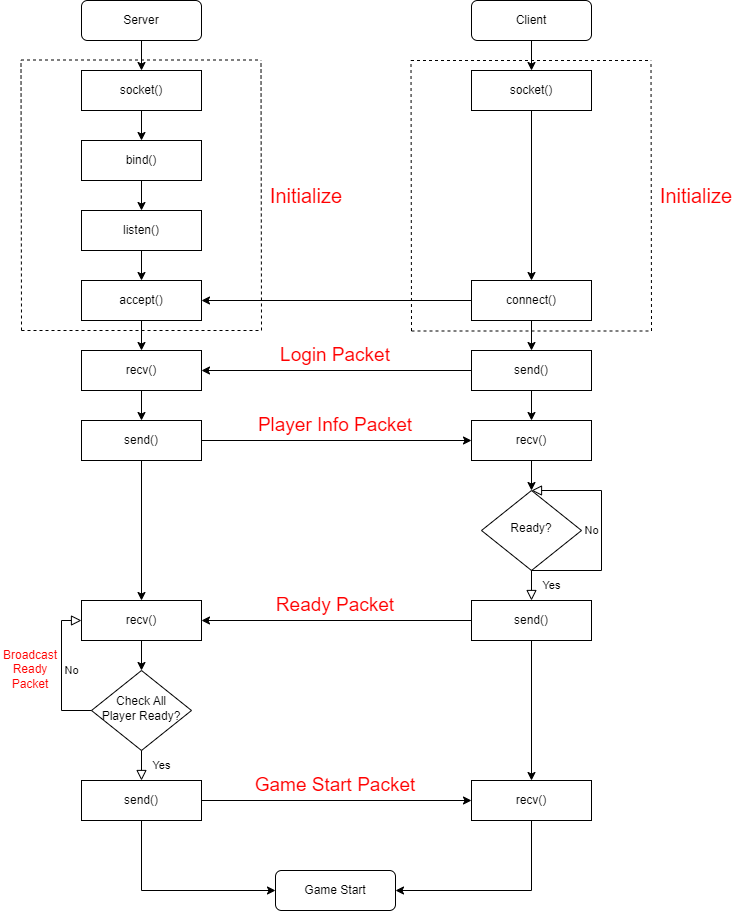
**1-1) 수정 전**



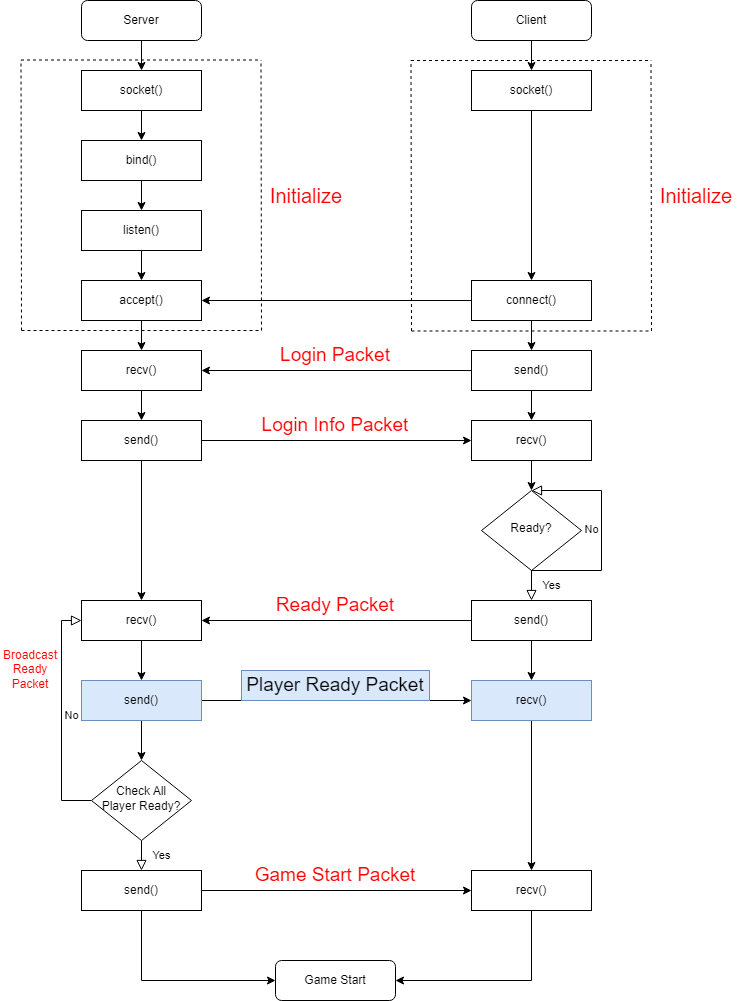
**1-2) 수정 후  
**

**2) 로그인 흐름도**

**2-1) 수정 전**

****

**2-2) 수정 후**

****

**Server**

1. 서버를 실행 후, socket을 생성하고 bind를 하고 listen상태에서 클라이언트의 connect 요청이 오면 accept를 해준다.

2. 클라이언트가 로그인을 하면 로그인 패킷을 받는다.

3. 다른 클라이언트들에게 로그인 정보 패킷을 보내준다.

4. 클라이언트가 준비하면 레디 패킷을 받는다.

5. 모든 클라이언트가 준비가 되지 않으면 모든 클라이언트에게 레디 패킷을 보낸다. 준비가 되면 모두에게 게임 스타트 패킷을 보낸다.

**Client**

1. 클라이언트 실행 후, socket을 생성하고 클라이언트가 connect 요청한다.

2. 클라이언트 로그인 정보를 입력하고 로그인 패킷을 보낸다.

3. 서버가 보내주는 로그인 정보 패킷 받는다.

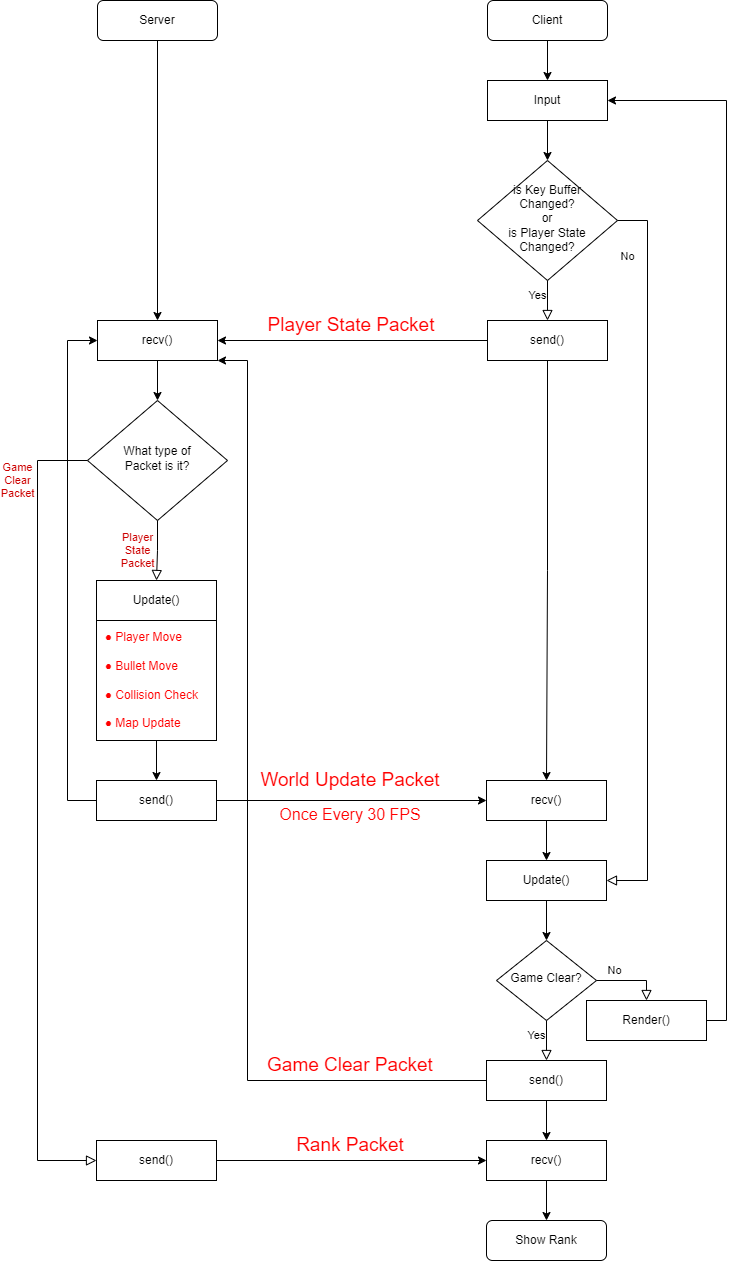
4. 준비를 실행하면 레디 패킷을 보낸다.

5. 다른 클라이언트들의 레디 패킷을 받는다. 모든 클라이언트가 준비가 되면 게임 스타트 패킷을 받는다.

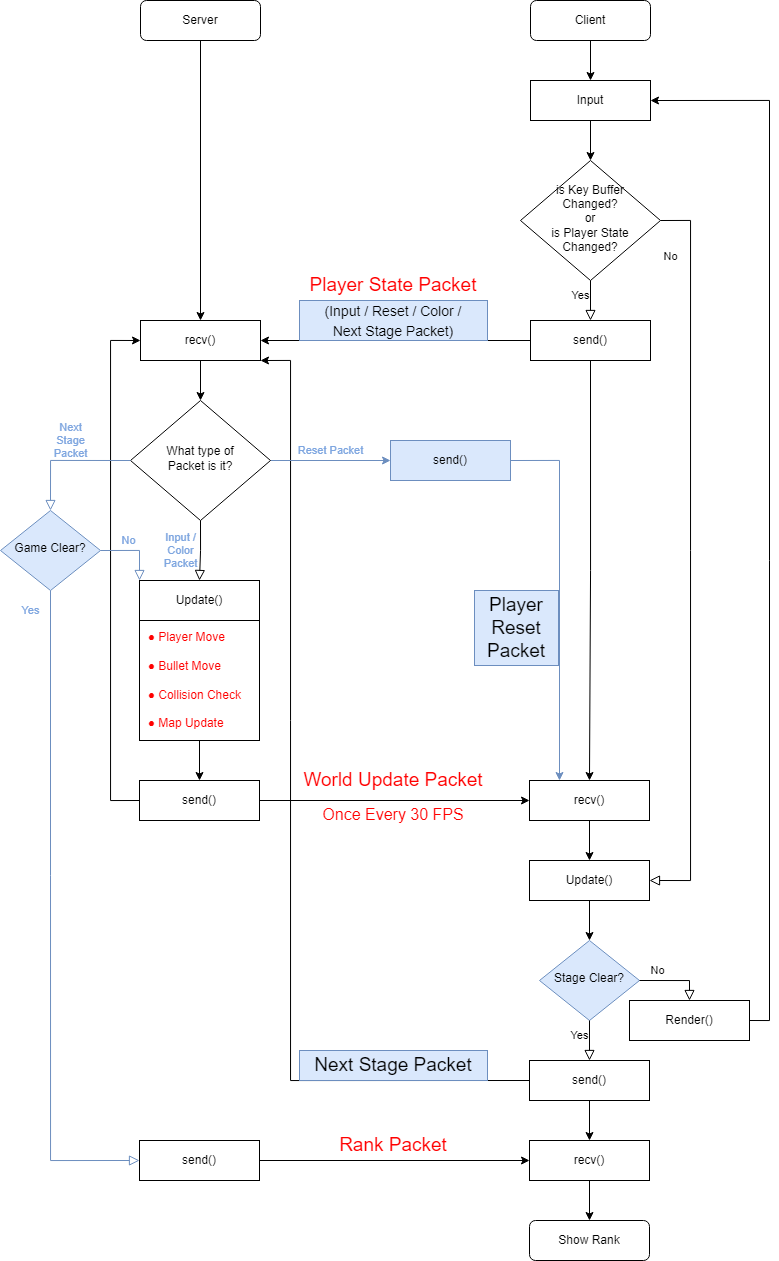
6. 게임을 시작한다.

**3) 인 게임 흐름도**

**3-1) 수정 전**

****

**3-2) 수정 후**

****

**Server**

1. 클라이언트가 보낸 패킷을 받는다.

~~2. 보낸 패킷이 Player State Packet이면 3번을 실행한다. Game Clear Packet이면 5번을 실행한다.~~

2. 받아온 패킷의 종류에 따라 다르게 작동한다.

2-1. Input Packet / Color Packet 🡪 Server에서 Update를 진행한다.

2-2. Reset Packet 🡪 서버의 해당 플레이어 정보를 Reset하고, 각 클라이언트로 Player Reset Packet을 보낸다.

2-3. Next Stage Packet 🡪 서버의 해당 플레이어를 다음 스테이지로 이동시킨다.

3. 2-1이 실행됐다면, 플레이어의 움직임, 총알의 움직임, 충돌체크, 맵 정보를 업데이트한다.

4. 모든 클라이언트에게 World Updated Packet을 보낸다. 다시 1번을 실행한다.

~~5. 1등 클라이언트의 랭킹 패킷을 보낸다~~

5. 2-3이 실행되고, Next Stage Packet을 전송한 클라이언트가 Stage 5에 있었다면, 각 클라이언트에 Rank Packet을 보낸다.

**Client**

~~1. 클라이언트에서 키를 입력하면 그 키와 입력된 키 버퍼가 다른 지와 플레이어의 상태가 바뀌었는지를 판단해 다르거나 바뀌면 Player State Packet을 보내고 아니면 3번을 실행한다.~~

1. 클라이언트에서 키를 입력하면 Input Packet, 포션을 획득하거나 스포이드 등을 사용하여 플레이어의 색이 변하면 Color Packet, 터렛이나 총알의 공격으로 사망하거나 R버튼을 눌러 리셋을 하였을 경우 Reset Packet, 스테이지를 클리어하였으면 Next Stage Packet을 보낸다.

2. 서버가 보내는 World Updated Packet 받는다.

3. Update를 한다.

~~4. 게임이 클리어가 안되었으면 렌더를 하고 다시 1번을 실행한다. 되었으면 Game Clear Packet을 보내준다.~~

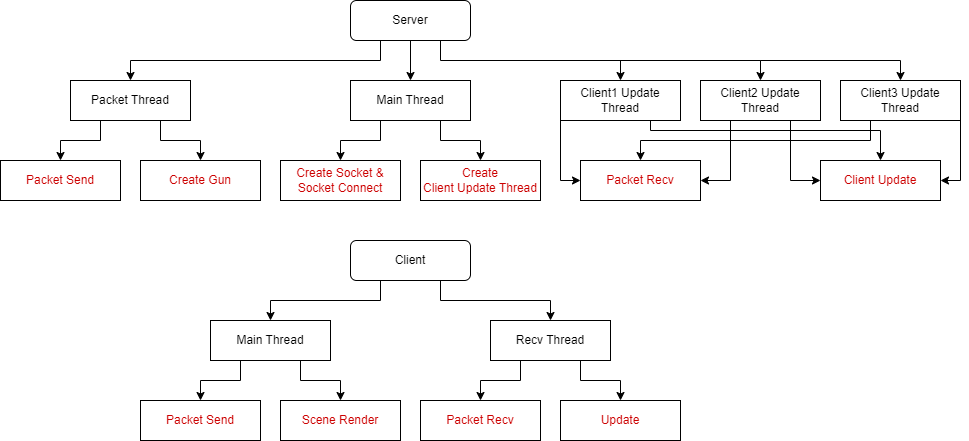
4. 이후 Render를 실행하고, 서버에서 Rank Packet이 오기 전까지 위의 과정을 반복한다.

5. 서버가 보내는 Rank Packet을 받는다.

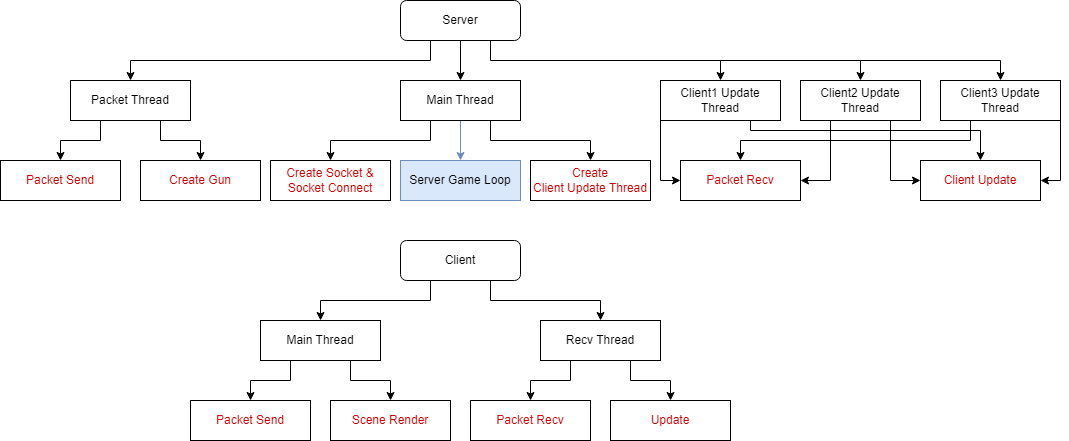
6. 랭킹을 보여준다.

**4) Server/Client Thread**

**4-1) 수정 전**



**4-2) 수정 후**



**Server**

1.Main Thread

소켓 생성과 클라이언트가 접속하면 소켓 연결을 하고 Client Update Thread를 생성한다. 동시에 각 클라이언트에 대한 업데이트를 진행한다. (좌표 갱신, 충돌 체크 등)

2.Packet Thread

총1개를 맵에 랜덤으로 생성해준다.

모든 클라이언트에게 동일한 패킷을 보낸다.

3. Client Update Thread

클라이언트가 보내는 패킷을 받고 그 정보를 업데이트 해준다.

**Client**

1.Main Thread

서버에게 패킷을 보내주고 가지고 있는 정보로 렌더한다.

2.Recv Thread

서버가 보내는 패킷을 받고 그 정보를 업데이트 해준다.

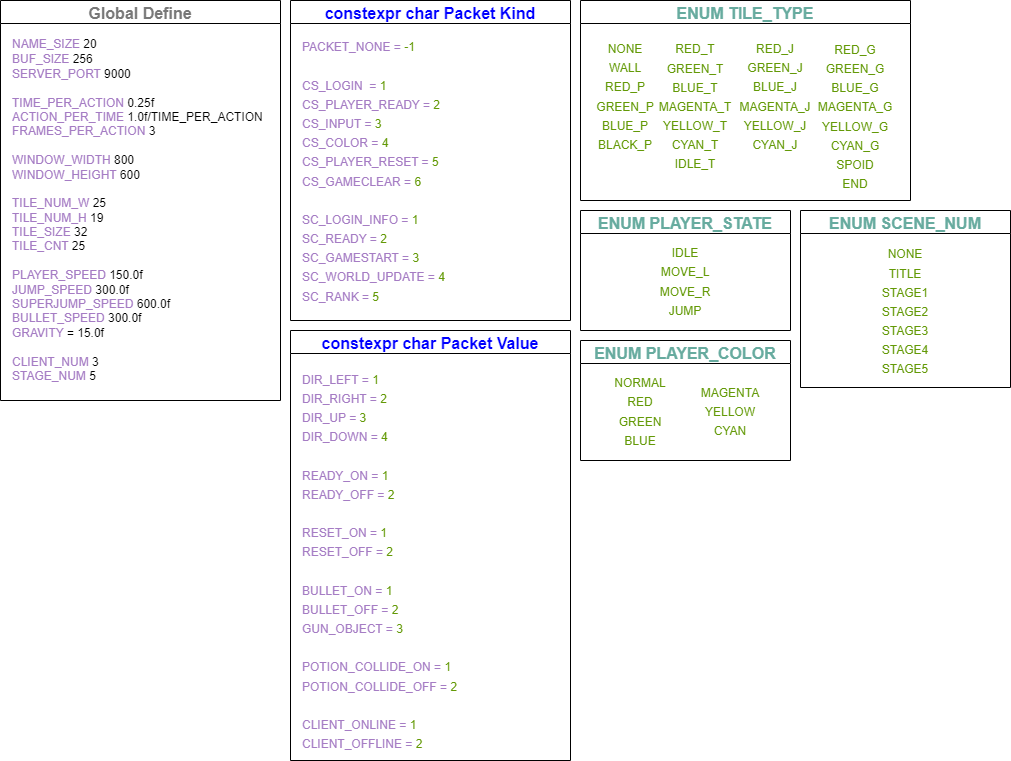
**Critical Section**

1. 서버 및 클라이언트에서 공유 변수를 사용함에 있어서 데이터 레이스가 발생하지 않게 하기 위해서 임계 영역을 사용한다.
2. 프로그램을 설계함에 있어서 스레드의 순서가 크게 상관이 없으므로 임계 영역을 선택했다.

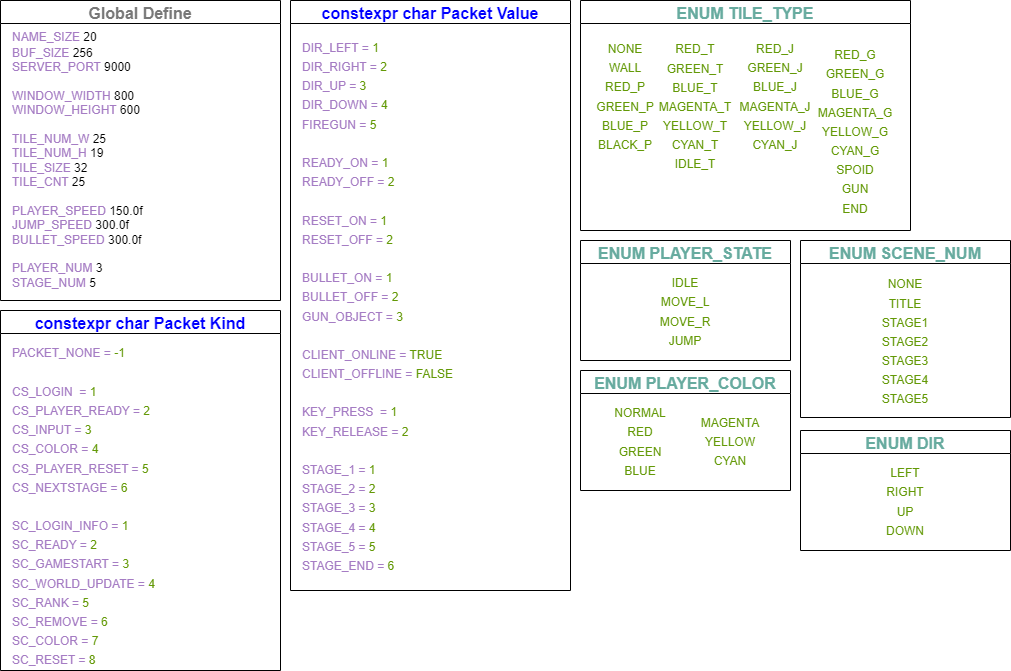
**3. Low Level Design**

**1) 전역 상수**

**1-1) 수정 전**

****

**1-2) 수정 후**

****

1. Global Define

* NAME\_SIZE : 로그인 시 이름의 길이
* BUF\_SIZE : 패킷을 받을 배열의 크기
* SERVER\_PORT : 서버의 포트 번호
* ~~TIME\_PER\_ACTION : 하나의 액션당 시간~~
* ~~ACTION\_PER\_TIME : 1초당 액션의 수~~
* ~~FRAMES\_PER\_ACTION : 프레임당 액션의 수~~
* WINDOW\_WIDTH : 윈도우 창의 가로 길이
* WINDOW\_HEIGHT : 윈도우 창의 세로 길이
* TILE\_NUM\_W : 가로 타일의 개수
* TILE\_NUM\_H : 세로 타일의 개수
* TILE\_SIZE : 타일의 가로세로 크기
* TILE\_CNT : 타일 종류의 개수
* PLAYER\_SPEED : 플레이어의 속도
* JUMP\_SPEED : 점프 속도
* ~~SUPERJUMP\_SPEED : 점프대에서 점프했을 때의 속도~~
* BULLET\_SPEED : 총알의 속도
* ~~GRAVITY : 중력~~
* ~~CLIENT\_NUM : 게임을 플레이하는 클라이언트의 수~~
* PLAYER\_NUM : 게임을 플레이하는 플레이어의 수
* STAGE\_NUM : 스테이지의 수

1. constexpr char

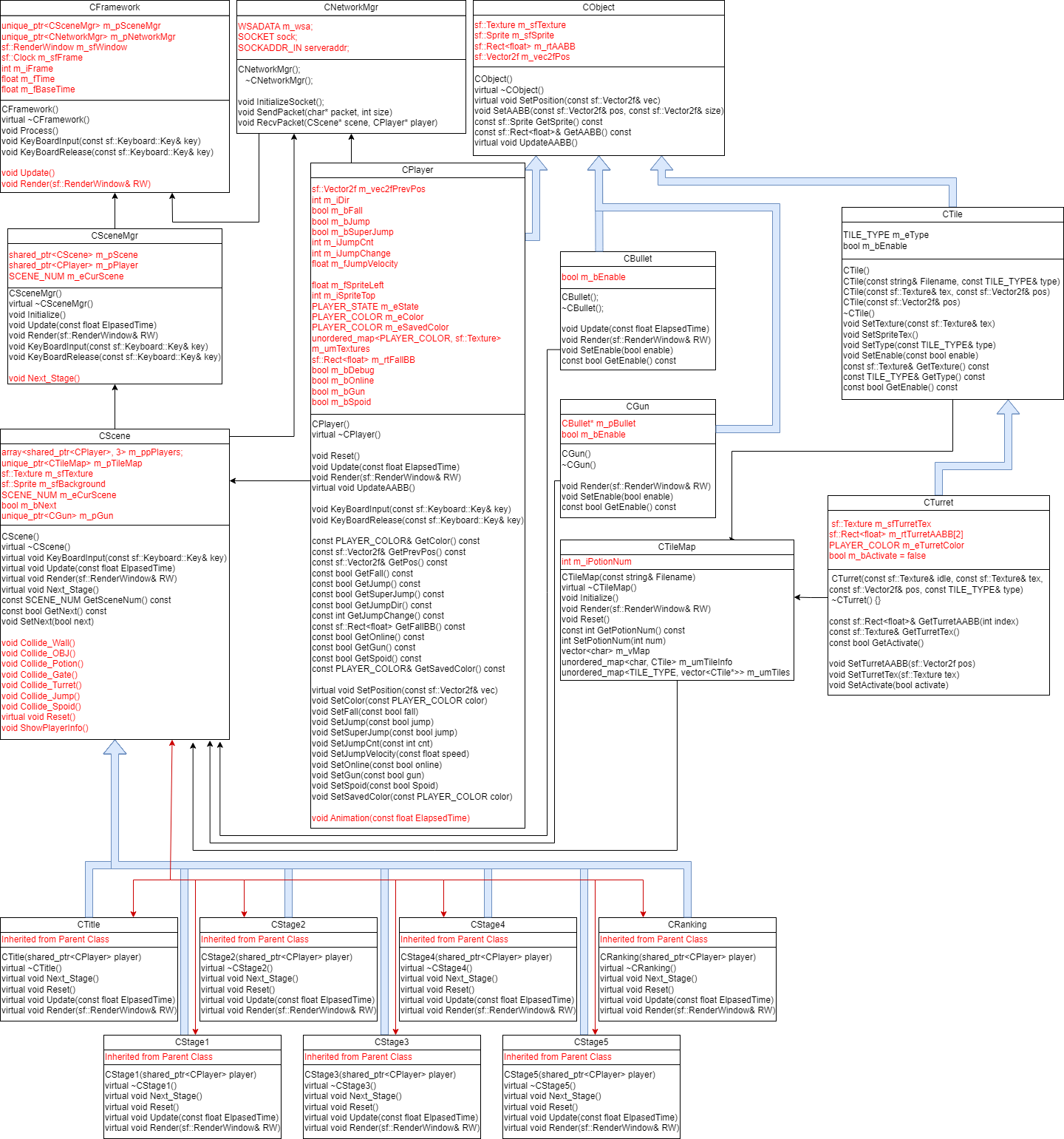
* PACKET KIND : 패킷의 종류
  + PACKET\_NONE : 패킷의 종류가 지정되지 않음
  + CS\_LOGIN, CS\_PLAYER\_READY … : Client To Server 패킷
  + SC\_LOGIN\_INFO, SC\_READY … : Server To Client 패킷
* PACKET VALUE : 패킷에서 전달하는 값의 종류
  + DIR\_LEFT, DIR\_RIGHT … : 객체 방향
  + FIREGUN : 총 발사하는 키
  + READY\_ON, READY\_OFF : 로그인 이후 게임을 준비 여부
  + RESET\_ON, RESET\_OFF : 캐릭터가 리셋 여부
  + BULLET\_ON, BULLET\_OFF … : 총알이 그려진 상태인지, 총이 그려진 상태인지
  + ~~POTION\_COLLIDE\_ON, POTION\_COLLIDE\_OFF : 포션과 충돌 여부~~
  + CLIENT\_ONLINE, CLIENT\_OFFLINE : 클라이언트의 접속 여부
  + KEY\_PRESS, KEY\_RELEASE : 키를 눌렀는지, 뗐는지
  + STAGE\_1, STAGE\_2 … : 스테이지 종류

1. ENUM CLASS

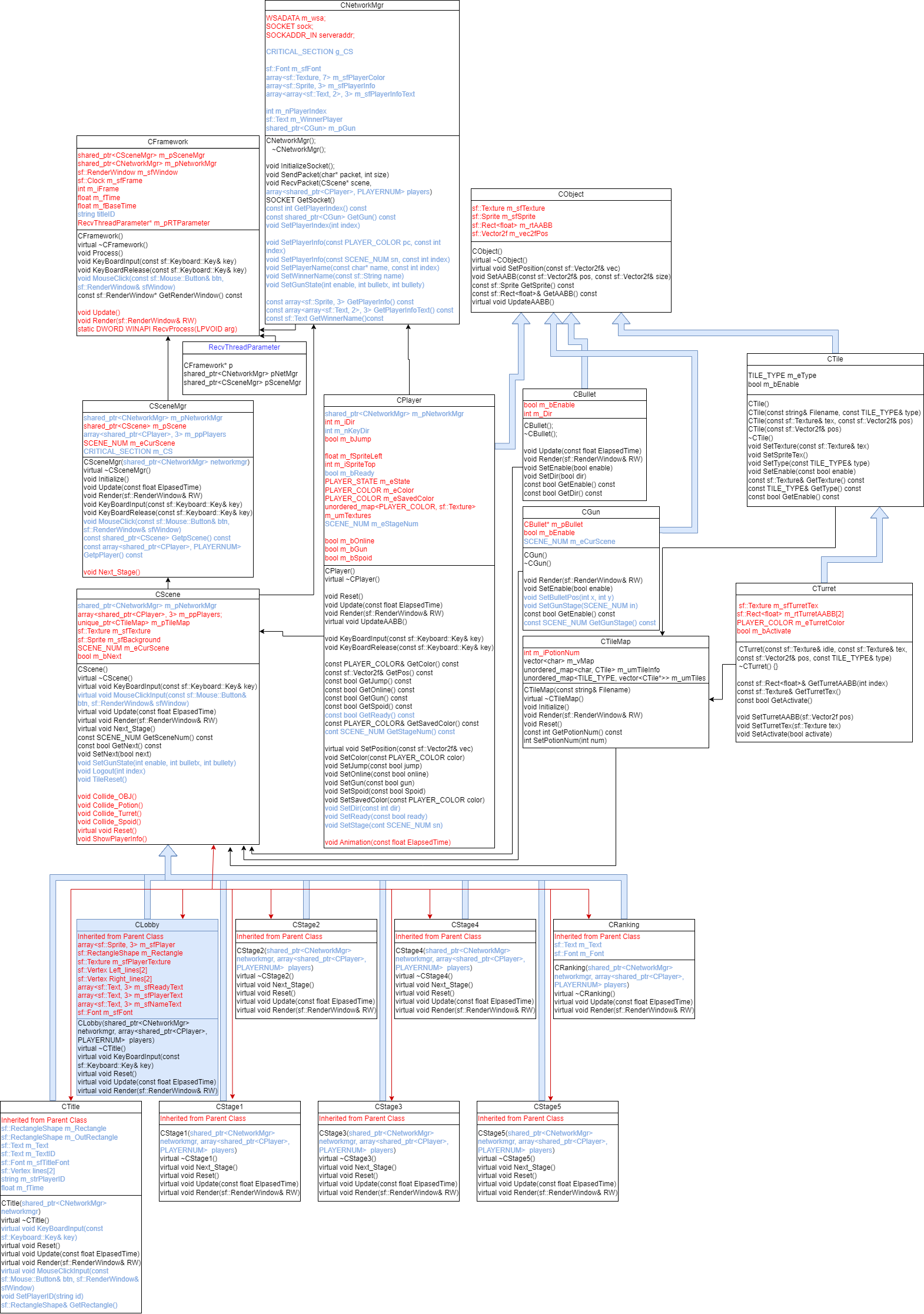
* ENUM TILE\_TYPE : 타일의 종류
  + NONE : 비어 있음
  + WALL : 벽 및 바닥, 천장 타일
  + RED\_P, GREEN\_P … : 색상 포션 아이템
  + RED\_T, GREEN\_T … : 플레이어와 색상이 동일할 때 작동하는 레이저
  + RED\_J, GREEN\_J … : 플레이어와 색상이 동일할 때 작동하는 점프대
  + RED\_G, GREEN\_G … : 플레이어와 색상이 동일할 때 지나갈 수 있는 관문
  + SPOID : 스포이트 아이템
  + GUN : 총 아이템
  + END : ENUM을 탐색할 때 ENUM의 끝을 알 수 있는 번호
* ENUM PLAYER\_STATE : 플레이어의 상태
  + IDLE : 그냥 서있는 상태
  + MOVE\_L : 왼쪽으로 이동하는 상태
  + MOVE\_R : 오른쪽으로 이동하는 상태
  + JUMP : 점프하는 상태
* ENUM PLAYER\_COLOR : 플레이어의 현재 색깔
  + NORMAL, RED, GREEN…
* ENUM SCENE\_NUM : 현재 그리고 있는 씬
  + NONE : 아무것도 그리지 않음
  + TITLE : 타이틀 화면
  + STAGE1, STAGE2 … : N번째 스테이지

**2) Client UML**

**2-1) 수정 전**

****

**2-2) 수정 후**

****

Client

1. CFramework

- void Process() : Game Loop

- void Update() : Frame Rate계산

2. CSceneMgr

- void Initailize() : Scene과 Player를 생성 및 초기화

- void Next\_Stage() : 다음 스테이지로 넘어가는 것을 체크

3. CNetworkMgr

- void InitalizeSocket() : 소켓 생성 및 연결

- void SendPacket(char\* packet, int size) : 서버에서 패킷 전송

- void RecvPacket(CScene\* scene, CPlayer\* player) : 서버로부터 패킷을 받고 클라이언트 정보 업데이트

4. CScene

- virtual void Update(const float ElapsedTime) : 렌더링에 필요한 정보 업데이트

- virtual void Next\_Stage() : 플레이어가 스테이지를 클리어 했는지 체크

~~- void Collide\_Wall() : 플레이어와 벽 충돌처리~~

- void Collide\_OBJ() : 플레이어와 오브젝트 충돌처리

- void Collide\_Potion() : 플레이어와 물약 충돌처리

~~- void Collide\_Gate() : 플레이어와 문 충돌처리~~

- void Collide\_Turret() : 플레이어와 터렛 충돌처리

~~- void Collide\_Jump() : 플레이어와 점프대 충돌처리~~

- virtual void Reset() : 스테이지 리셋

- void ShowPlayerInfo() : 게임 안에서 각각의 플레이어들에게 모든 플레이어의 스테이지 정보, 색, 총을 들고 있는지 여부를 화면 우측 상단에 그려준다.

- void Logout(int index) : 클라이언트 하나가 로그아웃 했을 경우 그 해당되는 클라이언트의 정보를 업데이트 해준다.

- void TileReset() : 타일 맵을 처음 상태로 초기화한다.

5. CObject

- void UpdateAABB() : 충돌박스 업데이트

6. CPlayer

- void Reset() : 플레이어 리셋

- void Animation() : 플레이어 렌더링 정보 업데이트

7. CTile

- 생성자들 : 텍스쳐와 스프라이트, 좌표 초기화

8. CTileMap

- void Initilze() : 타일맵을 생성하기 위해 타일 정보들을 매핑

- void Reset() : 타일맵, 매핑된 타일 정보 초기화  
9. CTurret

- SetTurretAABB(sf::Vector2f pos) : 플레이어가 가까이 오면 터렛이 작동하도록 충돌박스 추가 설정

공통함수

- void Update(const float Elapsed) : 객체 상태 업데이트

- void Render(sf::RenderWindow& RW) : 객체를 윈도우에 출력

- void KeyBoardInput(const sf::Keyboard::Key& key) : 클라이언트에서 눌린 키입력 처리(Key Down)

- void KeyBoardRelease(const sf::Keyboard::Key& key) : 클라이언트에서 해제된 키입력 처리(Key Up)

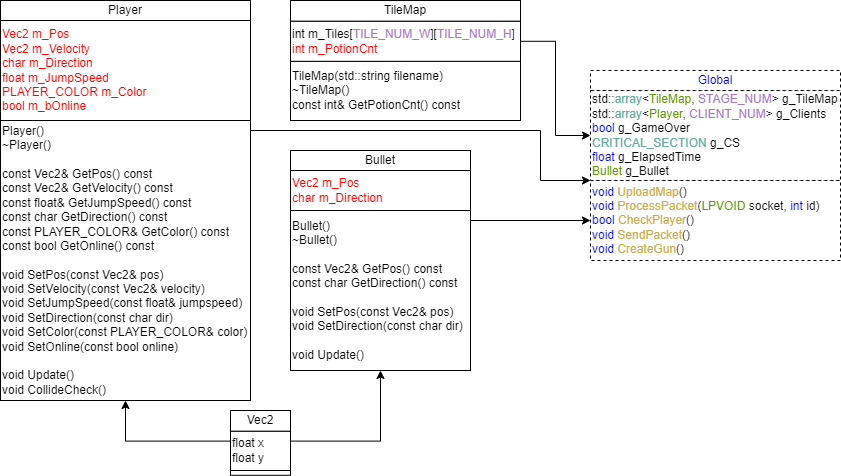
- const <Type> Get<Name>() const : <Name>에 해당하는 변수를 리턴

- void Set<Name>(<Type>) : <Name>에 해당하는 변수 값 설정

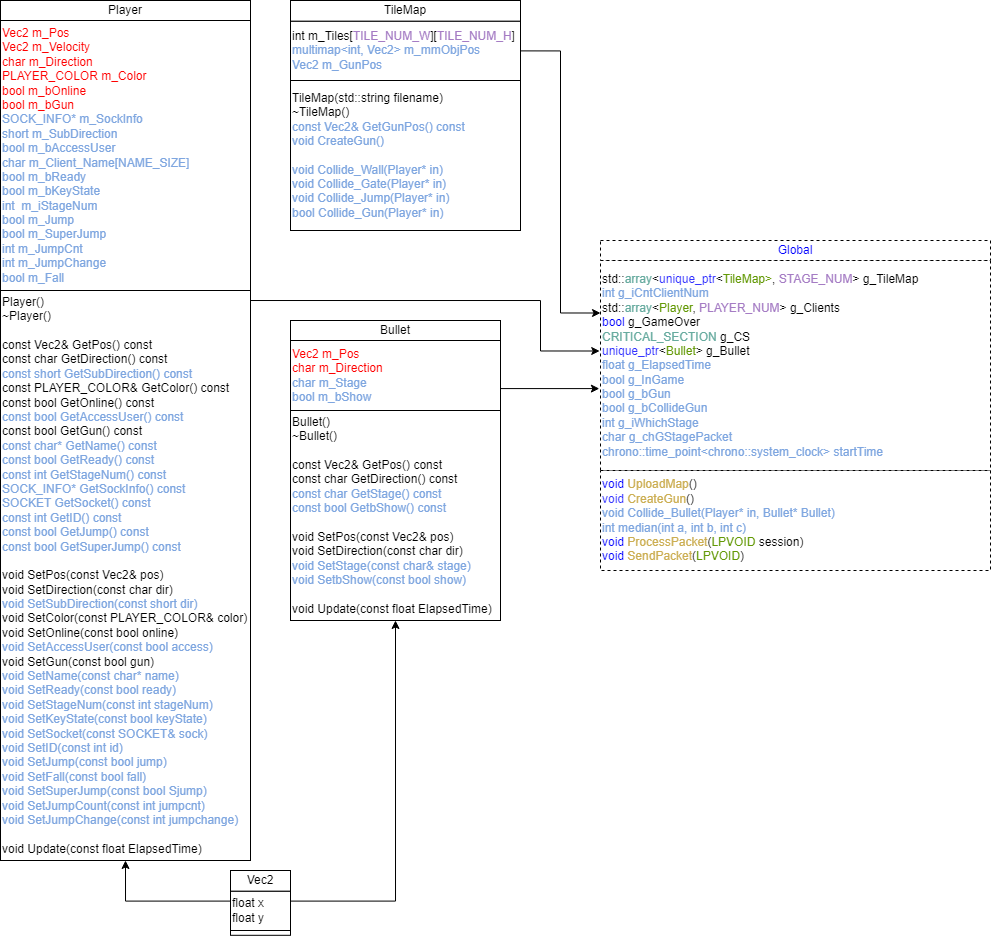
- void MouseClick(const sf::Mouse::Button& btn, sf::RenderWindow& sfWindow) : 마우스 입력 처리

**3) Server UML**

**3-1) 수정 전**

****

**3-2) 수정 후**

****

Server

1. Player

~~- void CollideCheck() : 플레이어와 객체 충돌체크~~

2. Global

- void UploadMap() : File I/O를 이용하여 타일맵 데이터 업로드

- void ProcessPacket(LPVOID socket, int id) : 클라이언트와 통신하기 위한 스레드에 할당되는 함수로서 패킷을 받아 처리한다(클라이언트 정보 업데이트, 벽과 플레이어 충돌처리)

- void SendPacket() : 패킷을 보내기 위한 스레드에 할당되는 함수이며 연결된 클라이언트들에게 패킷을 전송하며, 일정 시간마다 총을 생성한다

~~- void CheckPlayer() : 플레이어가 다음 스테이지로 넘어갈 수 있는지 체크하여 boolean타입을 반환~~

- void CreateGun() : ~~랜덤한 위치에 총을 생성한다~~ 각 클라이언트의 현재 스테이지 정보를 받아와 평균을 내서 중간에 위치하는 스테이지에 총을 생성하게 CreateGun을 호출한다.

- Collide\_Bullet(Player\* in, Bullet\* Bullet) : 총알과의 충돌을 검사한다.

- int median(int a, int b, int c) : 세 값의 평균을 구한다.

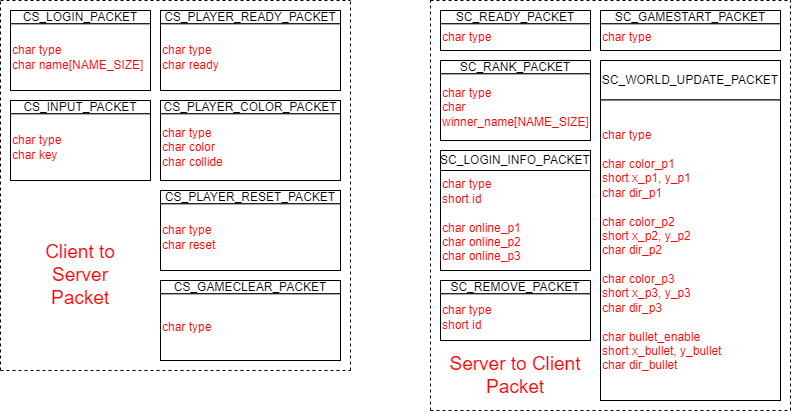
3. TileMap

- void CreateGun() : 해당 타일 맵에서 랜덤한 위치에 총을 생성한다.

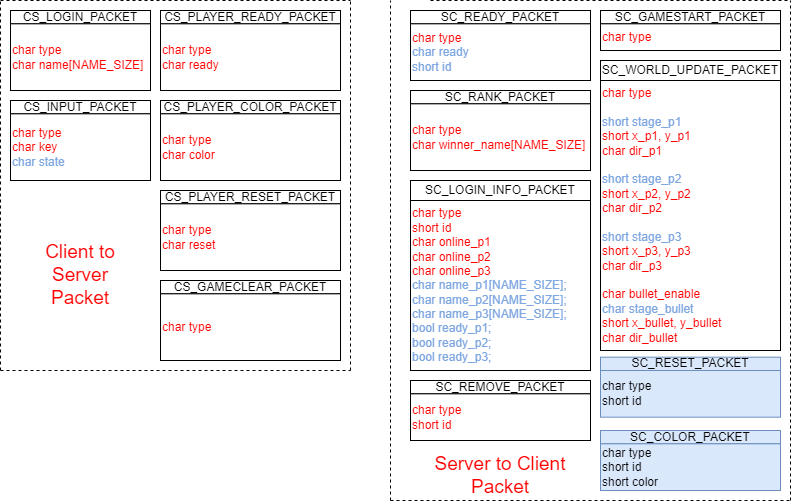
- void Collide\_Wall(Player\* in), Collide\_Gate… : 해당 객체와의 충돌을 검사한다.

**4) 패킷**

**4-1) 수정 전**

****

**4-2) 수정 후**

****

1. 공통

* type : 어떤 패킷인지 구분

1. Client To Server

* CS\_LOGIN\_PACKET : 클라이언트가 로그인할 때 설정한 게임 내의 이름을 서버에게 보내주는 패킷
* CS\_PLAYER\_READY\_PACKET : 클라이언트가 로비에서 준비를 누르면 서버에게 준비상태를 보내주는 패킷
* CS\_INPUT\_PACKET : 클라이언트가 플레이어의 키버퍼나 상태가 바뀔 때 서버에게 바뀐 키버퍼의 키를 보내주는 패킷
* CS\_PLAYER\_COLOR\_PACKER : 클라이언트가 색깔이 바뀌었을 때 서버에게 바뀐 색깔을 보내주는 패킷
* CS\_PLAYER\_RESET\_PACKER : 클라이언트가 죽을 때나 리셋 버튼을 누를 때 서버에게 리셋을 하라고 보내주는 패킷
* CS\_GAMECLEAR\_PACKET : 클라이언트가 게임 스테이지를 클리어하면 서버에게 클리어 정보를 보내주는 패킷

1. Server To Client

* SC\_LOGIN\_INFO\_PACKET : 클라이언트가 로그인 했을 때 모든 클라이언트에게 접속한 클라이언트 id 전송하기 위한 패킷
* SC\_READY\_PACKET : 특정 클라이언트가 준비완료 하였다는 것을 다른 클라이언트에게 알려주는 패킷
* SC\_GAMESTART\_PACKET : 모든 클라이언트가 준비완료 했을 때 게임시작을 알리는 패킷
* SC\_WORLD\_UPDATE\_PACKET : 클라이언트들의 정보를 업데이트 하기 위한 패킷
* SC\_RANK\_PACKET : 게임종료 후 1등을 알려주는 패킷
* SC\_REMOVE\_PACKET : 특정 클라이언트가 없어졌다는 것을 다른 클라이언트에게 알리는 패킷
* SC\_RESET\_PACKET : 특정 클라이언트가 서버에서 리셋 되었을 때 그 정보를 보내주기 위함
* SC\_COLOR\_PACKET : 색상이 변했을 때 각 클라이언트에게 그 정보를 뿌려주는 패킷

1. **역할분담**

|  |  |  |  |
| --- | --- | --- | --- |
| **구분** | **구현 내용** | **구현 함수** | **담당자** |
| **서버** | **Player 클래스** | **class Player** | **강현석** |
| **TileMap, Bullet클래스** | **class TileMap, Bullet** | **김성태** |
| **File I/O** | **UploadMap()** | **김성태** |
| **Thread 구분** | **Main, Packet, Client Thread** | **김성태** |
| **클라이언트에게**  **패킷 전송** | **SendPacket()** | **김성태** |
| **클라이언트에게서**  **패킷 받기** | **ProcessPacket(LPVOID, int)** | **김호준** |
| **패킷을 구분하여 처리(충돌처리, 플레이어**  **정보 업데이트)** | **ProcessPacket(LPVOID, int)** | **김호준** |
| **일정시간마다 총 생성** | **CreateGun()** | **김호준** |
| **플레이어 스테이지**  **클리어 확인** | **CheckPlayer()** | **김호준** |
| **클라이언트** | **서버로부터 패킷 받기** | **RecvPacket(CScene\*, CPlayer\*)** | **강현석** |
| **전달받은 패킷 처리** | **RecvPacket(CScene\*, CPlayer\*)** | **강현석** |
| **CBullet, CGun 클래스** | **class CBullet, class CGun** | **강현석** |
| **충돌처리 수정** | **CollideOBJ()** | **강현석** |
| **게임 종료 시**  **랭킹 출력 화면** | **class RankingScene** | **강현석** |
| **추가된 클라이언트**  **프로그램에 반영** |  | **김성태** |
| **CNetworkMgr 클래스** | **class CNetworkMgr** | **김성태** |
| **서버에게 패킷 전송** | **SendPacket(char\*, int)** | **김성태** |
| **각 플레이어에 대한**  **인게임 정보** | **ShowPlayerInfo** | **김성태** |
| **클라이언트**  **프레임워크 수정** | **스레드 추가하도록**  **구조변경** | **김호준** |
| **로그인 화면 및**  **기능 구현** | **class LoginScene** | **김호준** |

1. **개발 일정**

|  |  |
| --- | --- |
| **완료한 일** | **추가된 일정** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **11월 1주차** | | | | | | |
| **요일** | **일요일** | **월요일** | **화요일** | **수요일** | **목요일** | **금요일** | **토요일** |
| **날짜** |  |  | **1일** | **2일** | **3일** | **4일** | **5일** |
| **강현석** |  |  |  |  | **계획서**  **검수** | **Player Class**  **구현** |  |
| **김성태** |  |  |  |  | **계획서**  **검수** | **Tilemap, Bullet Class**  **구현** |  |
| **김호준** |  |  |  |  | **계획서**  **검수** | **클라이언트 프레임워크**  **수정** |  |
|  | | | | | | **완료** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **11월 2주차** | | | | | | |
| **요일** | **일요일** | **월요일** | **화요일** | **수요일** | **목요일** | **금요일** | **토요일** |
| **날짜** | **6일** | **7일** | **8일** | **9일** | **10일** | **11일** | **12일** |
| **강현석** | **CBullet, CGun Class**  **구현** |  | **스레드 함수 수정**  **각 구현 내용 테스트**  **및 피드백 후 수정** | |  | **서버로부터 패킷 받기**  **구현** | |
| **김성태** | **NetworkMgr Class 구현, Thread 구분** |  |  | **Thread 구분, 서버에게**  **패킷 전송**  **구현** | **서버에게**  **패킷 전송**  **구현** |
| **김호준** | **클라이언트 프레임워크**  **수정** |  | **클라이언트에게 패킷 받기 구현** | |  |
|  | **완료** |  | **완료** | | **패킷 구분하여 처리 완료되었으나 로그인 신 구현 필요 (11월 5주차)** | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **11월 3주차** | | | | | | |
| **요일** | **일요일** | **월요일** | **화요일** | **수요일** | **목요일** | **금요일** | **토요일** |
| **날짜** | **13일** | **14일** | **15일** | **16일** | **17일** | **18일** | **19일** |
| **강현석** |  | **패킷 구분하여 처리** | |  |  | **업데이트 관련 버그 해결**  **및 플레이어 관련 버그**  **해결** | |
| **김성태** |  | **클라이언트에게**  **패킷 전송**  **구현** | |  |  |
| **김호준** |  | **패킷 구분하여 처리** | |  |  |
|  |  | **패킷 처리 🡪 세명의 플레이어가 연동되나 서버에서 업데이트가 되지 않아 좌표 갱신이 안됨. (수정 필요)** | |  | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **11월 4주차** | | | | | | |
| **요일** | **일요일** | **월요일** | **화요일** | **수요일** | **목요일** | **금요일** | **토요일** |
| **날짜** | **20일** | **21일** | **22일** | **23일** | **24일** | **25일** | **26일** |
| **강현석** |  | **CollideOBJ**  **구현** | |  | **딜레이 버그 해결,**  **프레임 드랍 버그 해결,**  **프로토콜 전달 값 이상**  **해결, Gun 관련 문제 해결** | |  |
| **김성태** |  | **UploadMap 구현,**  **다중 클라이언트 적용** | **다중**  **클라이언트**  **적용** |  |  |
| **김호준** |  | **CreateGun**  **구현** | |  |  |
|  | | **구현 완료 🡪 딜레이, 프레임 드랍, 값 이상 등의 여러 버그 발생** | |  | **각 클라이언트가 특정 상황에 종료되는 버그 발생** | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **11월 5주차** | | | | **12월 1주차** | | |
| **요일** | **일요일** | **월요일** | **화요일** | **수요일** | **목요일** | **금요일** | **토요일** |
| **날짜** | **27일** | **28일** | **29일** | **30일** | **1일** | **2일** | **3일** |
| **강현석** |  | **~~Ranking Scene~~**  **~~구현~~ 🡪 버그로 인해 목/금 일정으로 변경 🡪 대신 서버에서의 충돌 체크 완성** | | **스테이지 넘어갈 시 발생하는 버그 및 총 버그, 스레드 버그 수정** | **충돌 버그 및 ShowPlayerInfo 버그 수정**  **치트 추가**  **로비 씬 추가**  **랭킹 씬 추가** | | **랭킹 씬 버그, 로그인 씬 버그 수정, 로그아웃 추가, 충돌체크 수정** |
| **김성태** |  | **ShowPlayerInfo**  **구현** | |
| **김호준** |  | **Login Scene**  **구현** | |
|  | | **스테이지 넘어갈 시 터지는 버그 발생** | |  | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **12월 2주차** | | | | | | |
| **요일** | **일요일** | **월요일** | **화요일** | **수요일** | **목요일** | **금요일** | **토요일** |
| **날짜** | **4일** | **5일** | **6일** | **7일** | **8일** | **9일** | **10일** |
| **강현석** |  | **최종 검수 및 문서 작업 완료** | | **최종**  **프로젝트**  **제출** |  |  |  |
| **김성태** |  |  |  |  |
| **김호준** |  |  |  |  |