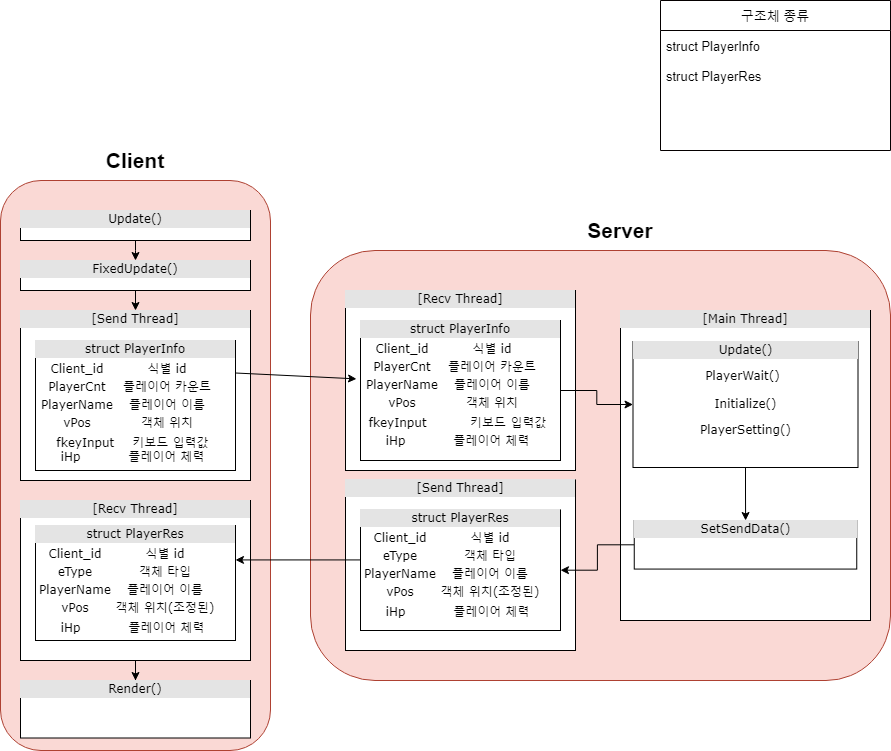
하이레벨



* FixedUpdate() 호출 후 PlayerInfo 구조체를 서버로 send한다. 넘겨주는 구조체에 있는 정보는 식별 id, 플레이어 이름, 플레이어 카운트, 객체 현재 위치, 키보드 입력 값이다.
* 서버는 이 정보를 받아서 Update 함수로 넘기고 객체의 타입을 확인하여 어떤 임무를 해야 하는 객체인지 확인하여 키보드 입력 값에 따라 이를 처리한다.
* 처리된 결과값을 Client에게 send하고 Client는 이를 Recv한다. 받은 정보를 토대로 객체의 위치를 재설정 후 렌더링을 한다.
* 4명의 플레이어가 로비에 연결이 되면 Initialize()에서 Client\_id와 PlayerName을 제외한 플레이어 정보 초기화.
* PlayerSetting()에서 팀을 나누고 Client\_id, PlayerName을 클라이언트로 전달

로우레벨

**Client send() -> Server recv()**

|  |  |  |
| --- | --- | --- |
| **PlayerInfo 구조체** | | |
| 인자 | 변수명 | 설명 |
| int | Client\_id | 식별 id |
| int | mPlayerCnt | 플레이어 수 |
| string | PlayerName | 플레이어 이름 |
| Vector<float2> | vPos | 플레이어 위치 |
| float | fkeyInput | 키보드 입력값 |
| int | iHp | 플레이어 체력 |

* 클라이언트에서 서버로 PlayerInfo 구조체를 전달한다. 구조체의 정보로 Update()에 있는 함수를 실행한다.

**Sever send() -> Client recv()**

|  |  |  |
| --- | --- | --- |
| **PlayerRes 구조체** | | |
| 인자 | 변수명 | 설명 |
| int | Client\_id | 식별 id |
| TEAM\_TYPE | TEAM1, TEAM2 | 플레이어 팀 구분 타입 |
| char | PlayerName | 플레이어 이름 |
| Vector<float2> | vPos | 플레이어 위치(조정된) |
| int | iHp | 플레이어 체력 |
| bool | bStartgame | 게임 시작 여부 |

* + - 1. Login
* 서버: PlayerData(): 접속한 클라이언트의 주소를 받아 식별id 생성.
  + - 1. Lobby
* ClientThread(): 서버가 클라이언트의 요청을 recv후 send
* CreateLobby(): 로비 생성함수
* PlayerWait(): 4명의 플레이어가 서버에 준비되었다고 전달 후 자동으로 게임 시작.
* Initalize(): 플레이어의 객체 타입, 팀, 좌표, 체력 등 플레이어에 대한 정보 초기화
* PlayerSetting(): 플레이어의 객체 타입, 팀 등 설정
* 멀티스레드 동기화
  + - 1. 각 클라이언트가 접속하는 순서대로 이벤트를 생성시 4개의 이벤트 생성
      2. 생성된 순서대로 스레드 함수가 실행되고, recv(), send()함수가 호출된다.
      3. 플레이어 팀에 따라 2개의 스레드는 레드 팀 스레드 함수가 실행되고, 다른 2개의 스레드는 블루 팀 스레드 함수가 실행된다.