2021 NGP

Term Project 추진계획서

9팀

2018182021안정인

2018182021 윤성주

2018184033 최경훈

1. 어플리케이션 기획

A. 참고 게임

어몽어스 인게임 화면

출처:https://www.bluestacks.com/ko/blog/game-guides/among-us/au-confirmation-guide-ko.html

'위자드 오브 레전드' 모작 인게임 화면

B. 게임 설명

어몽어스와 비슷한 게임으로 플레이어 4명이 참가해야 하고, 1명은 범인, 3명은 일반인으로 역할을 배정받아 게임을 진행한다. 게임 내 컨텐츠로는 임무 수행, 투표, 채팅 등이 있으며 이를 서버를 구현하여 모든 플레이어가 게임 내에서 상호작용할 수 있도록 한다.

C. 게임 진행 순서

I. 로그인

① 게임 시작화면
② 로그인(IP연결)을 하고 게임에 입장하면 플레이어 대기화면이 뜬다.
③ 클라이언트 4명이 입장할 때까지 기다린다.
④ 4명 모두 입장하면 게임이 시작된다.
⑤ 범인 1명, 일반인 3명 임무를 각 클라이언트에게 부여한다.
⑥ 인게임에 4명의 클라이언트가 배치된다.

II. 인게임 – 임무수행

① 일반인 3명은 임무 하는 곳에 가서 각자 주어진 임무를 한다. 이때 임무는 상대방에게 보여지는 것도 있고 보여 지지 않는 것도 있다.
② 임무 중 범인은 일반인을 1명씩 죽일 수 있다. 일반인은 죽으면 게임오버.(게임 더이상 참여 불가하나 연결은 된 상태)
③ 일반인이 시체를 목격하거나 범인으로 의심되는 행동을 보면 투표시작 버튼을 누르고 투표가 시작된다.
④ 투표가 끝나면 다시 ③단계와 같은 위치에 살아남은 플레이어가 배치되고 각자 임무를 수행하러 간다.
⑤ 임무 진척도가 100%을 달성하면 일반인이 이기게 되고, 범인이 누구였는지 공개된다.
⑥ 일정시간 후 게임이 종료되고 시작화면으로 돌아온다.

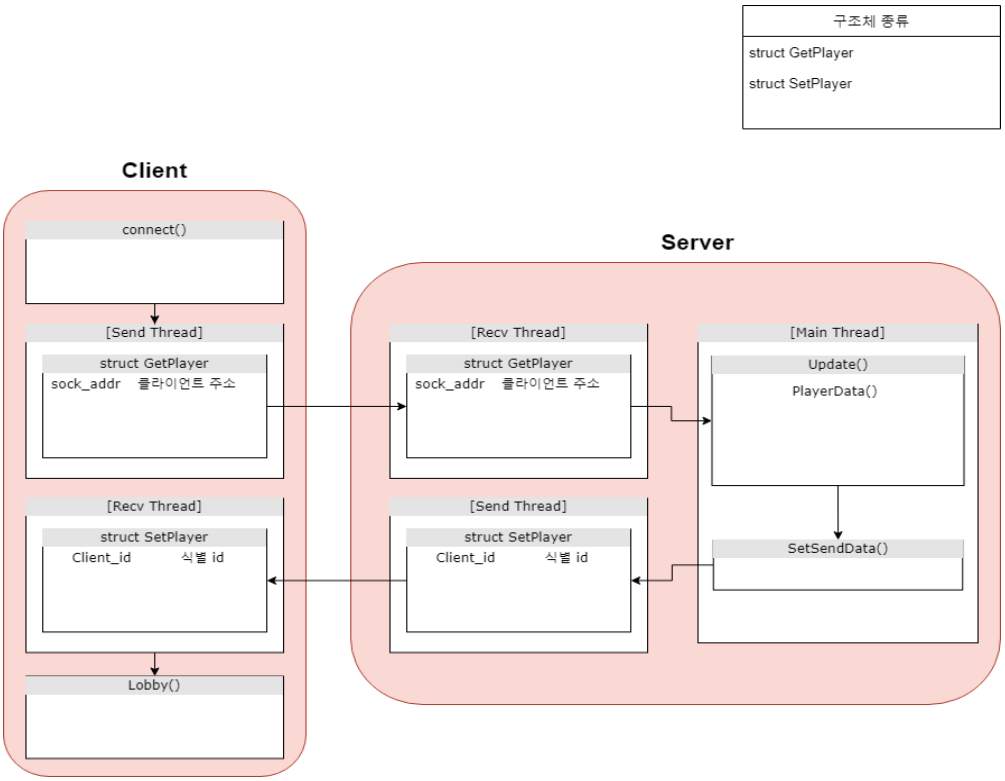
III. 인게임 – 투표&채팅

① 투표시작시 모든 플레이어가 채팅에 참여할 수 있다.
② 채팅 시간이 끝나거나 모든 플레이어가 투표 완료했을 시 채팅은 끝나고 투표 결과가 보여진다.
③ 투표 결과는 1.아무도 죽지 않음(과반수 안 될 때) 2. 죽음(과반수 넘을 때)
④ 투표가 끝나면 다시 ③단계와 같은 위치에 살아남은 플레이어가 배치되고 각자 임무를 수행하러 간다.

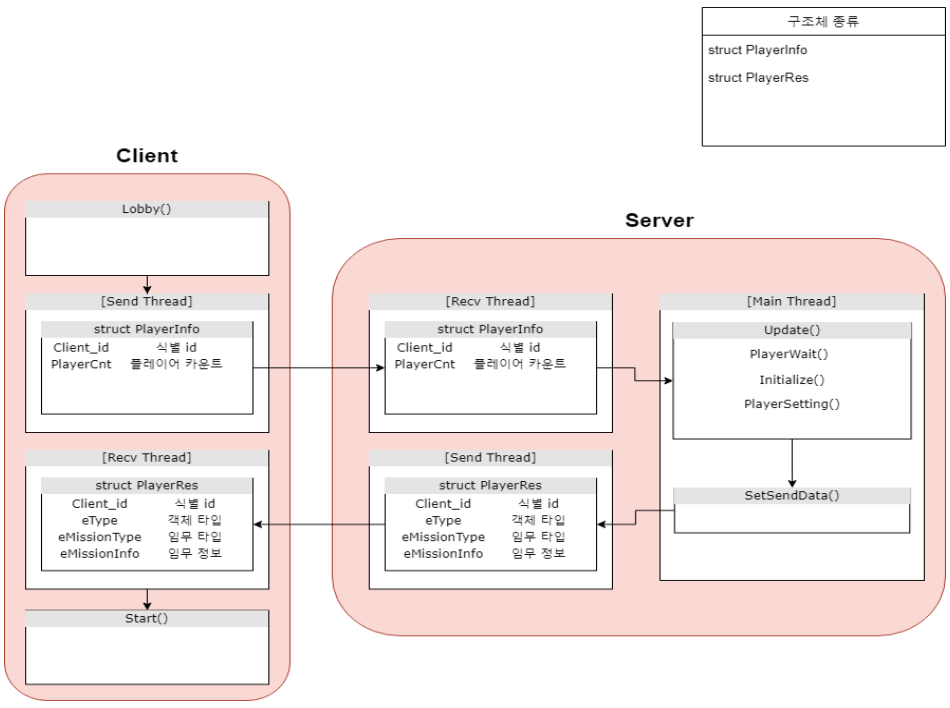
2. High-level design

A. 로그인

I. 함수 다이어그램



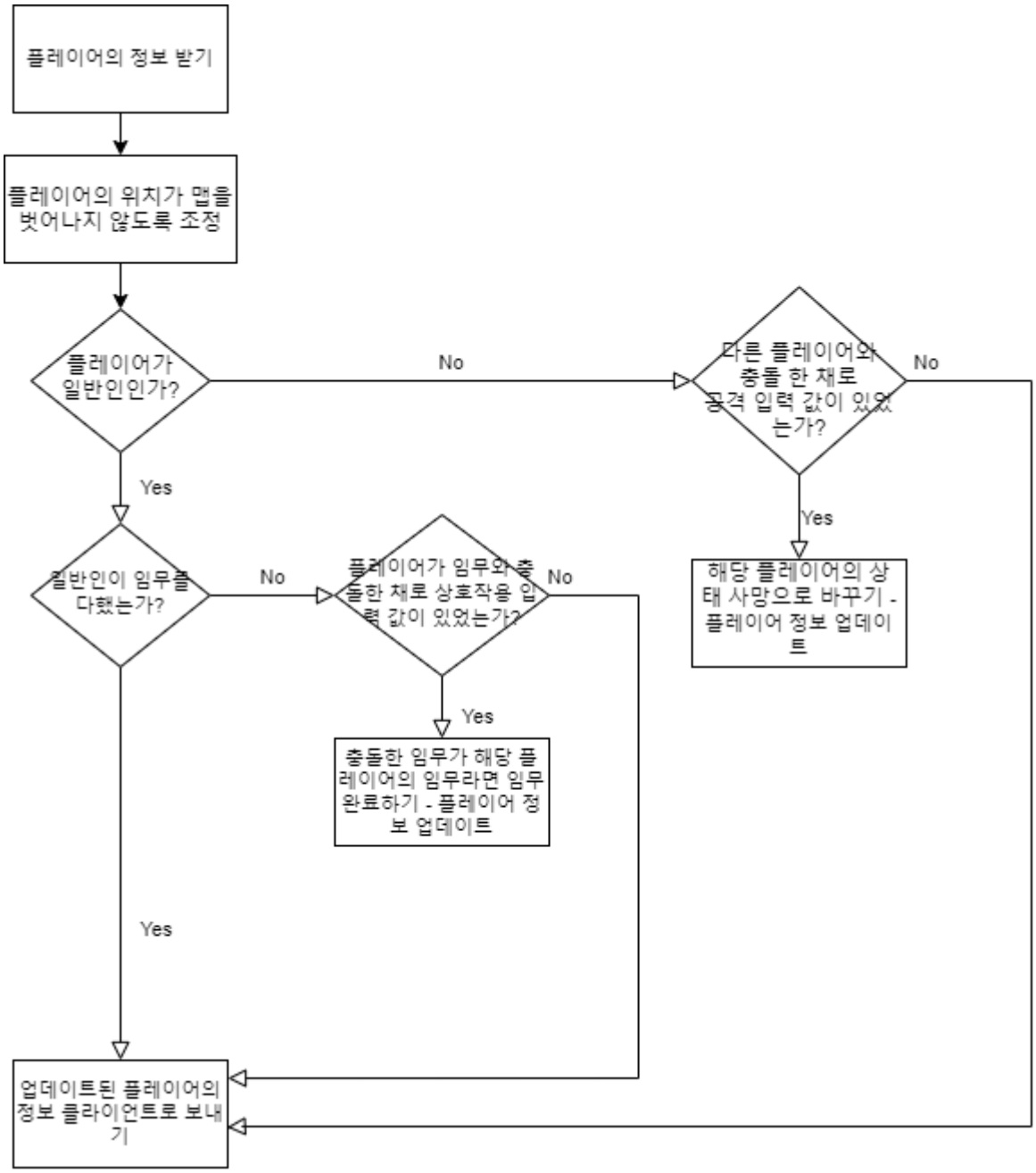
- 게임실행 후 클라이언트가 서버에 연결 요청 후 연결이 완료되면 서버에서 PlayerData()로 클라이언트 식별을 위한 Id 전송 후 로비로 입장



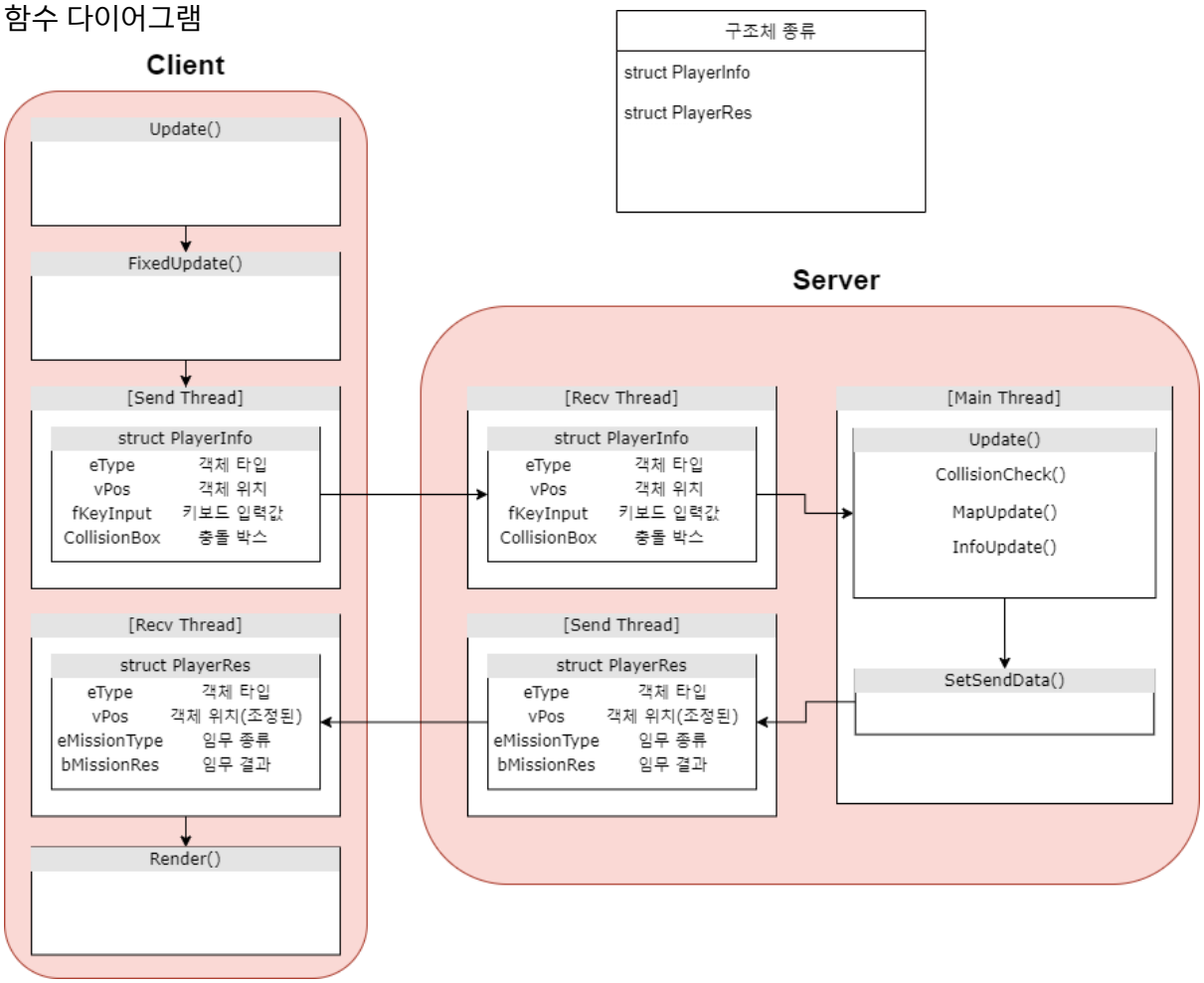
- 클라이언트에서 플레이어 id와 카운트를 받아 플레이어가 4명이 될 때까지 기다림.
- 4명의 플레이어가 로비에 연결이 되면 Initialize()에서 id를 제외한 플레이어 정보 초기화.
- PlayerSetting()에서 일반인과 범인을 나누고(eType) 임무 부여 eMissionType으로 일반임무와 범인 임무 구분 eMissionInfo 각각 다른 임무 정보 부여 후 클라이언트로 전달

B. 인게임 – 임무수행

I. 플로우차트



II. 함수 다이어그램



- Client의 FixedUpdtae() 호출 후에 PlayerInfo 구조체를 서버로 Send한다. 이때 넘겨주는 구조체에 들어있는 정보는 객체가 일반인1,2,3 또는 범인 중에 누구인지를 알려주는 객체 타입, 객체 현재 위치, 키보드 입력 값, 객체의 충돌박스이다.

- 서버는 이 정보를 받아서 Update 함수로 넘기고 객체의 타입을 확인하여 어떤 임무를 해야 하는 객체인지 확인하여 키보드 입력 값에 따라 이를 처리한다.

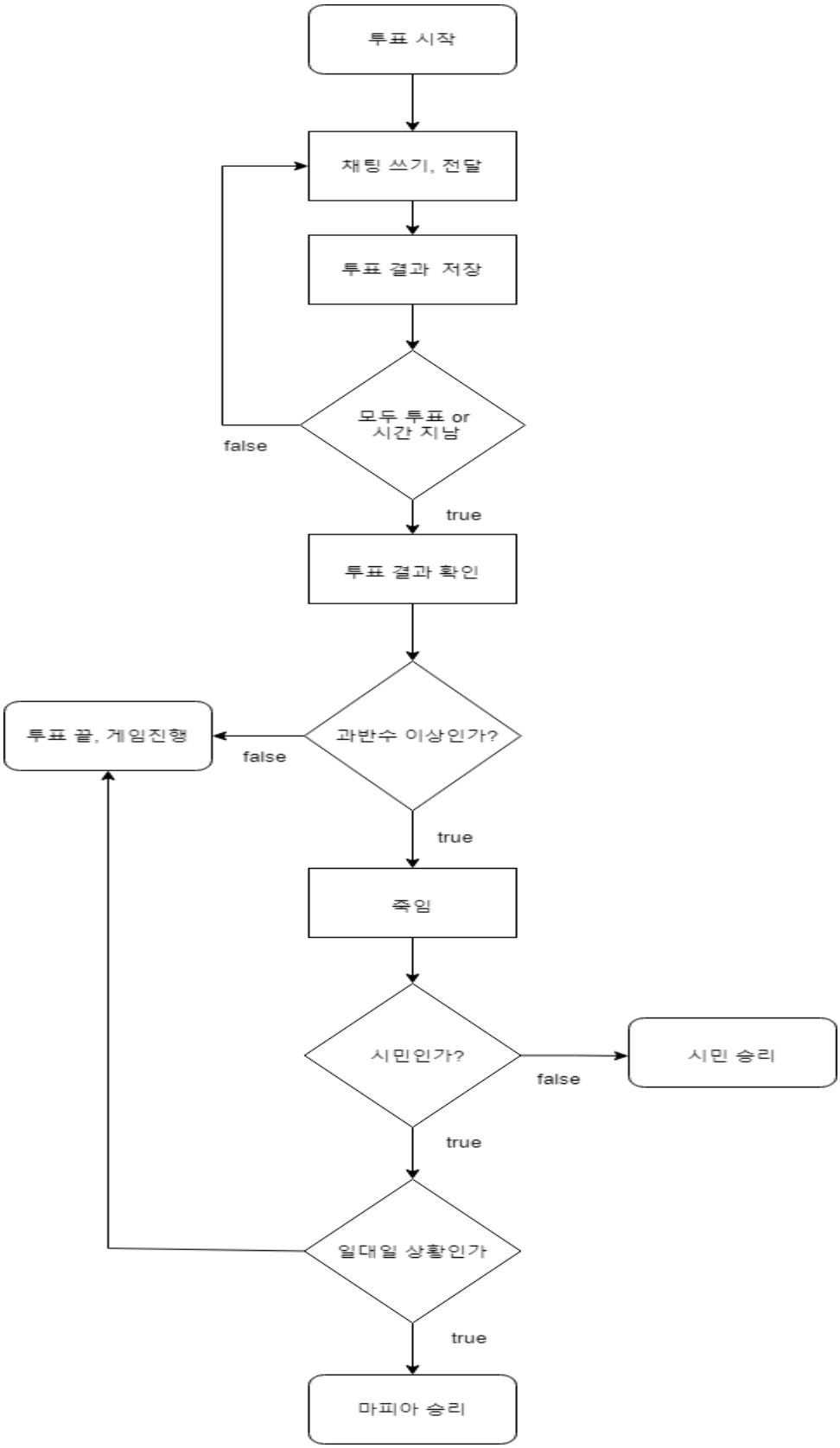
- 처리된 결과값을 Client에게 send하고 Client는 이를 Recv한다. 받은 정보를 토대로 객체의 위치를 재설정, 임무 상태 갱신을 하고 렌더링을 한다.

- 객체의 타입이 범인이라면 임무가 아닌 공격을 판정하고, 서버가 가지고 있는 일반인 플레이어의 위치와 연산하여 결과를 클라이언트에게 보내게 된다.

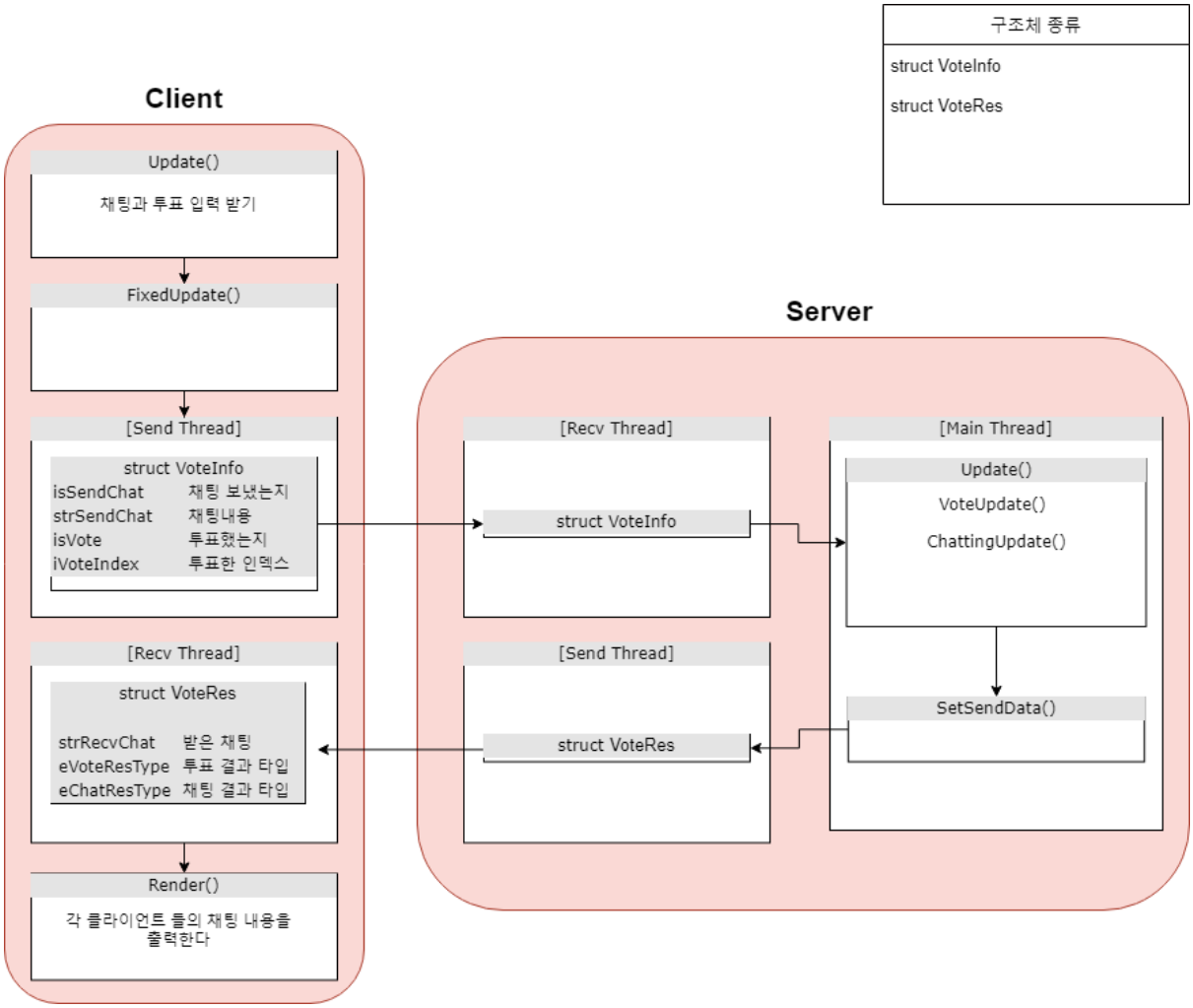
- 범인, 일반인이 다른 점이 임무를 하는지 피격을 하는지 이므로 두 종류의 플레이어 모두 같은 구조체로 데이터를 주고받는다.

C. 인게임 – 투표&채팅

I. 플로우차트



II. 함수 다이어그램

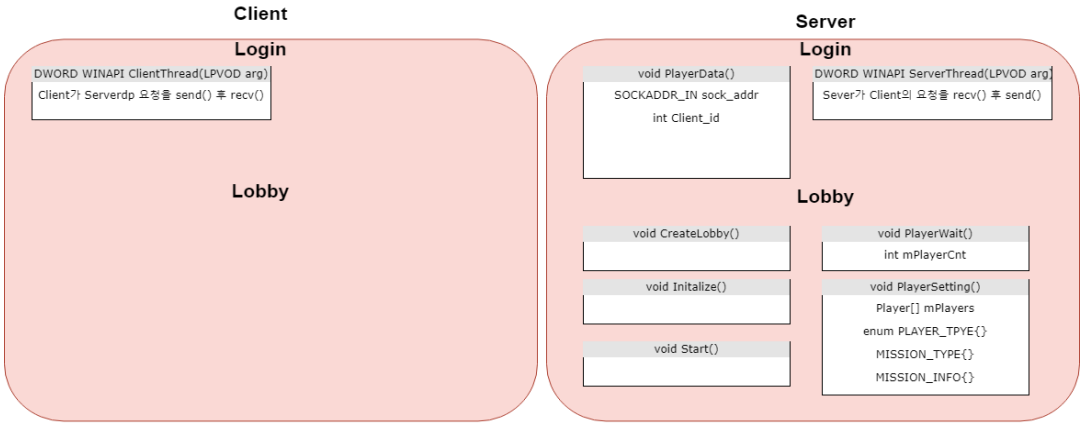


- 클라이언트는 채팅과 누구를 투표할지 입력을 받고, 그 정보를 담은 구조체인 VoteInfo를 서버에 넘긴다.
- 모든 클라가 서버에게 값을 전달했으면, 서버는 받은 구조체에서 채팅을 보냈는지, 투표를 했는지에 대한 정보를 확인하며,저장해야알 정보는 Vector에 담아 저장한다.
- 투표 결과와 채팅을 조합하여 각 클라에게 VoteRes구조체로 전달한다.
- 서버로 부터 받은 정보를 이용하여 채팅과 투표 현황을 화면에 출력한다

3. low-level design

A. 로그인

A. 객체



- 멀티스레드 동기화

  ✓ 임계영역을 사용하기전 InitializeCriticalSection()을 호출하여 초기화

  ✓ 공유 리소스 사용하기전에 EnterCritcalSection()을 호출하고 공유 리소스를 사용하는 스레드가 없다면 바로 리턴.

  ✓ 사용하는 스레드가 있다면 리턴하지 않고 대기상태.

  ✓ 공유 리소스 사용이 끝나면 LeaveCriticalSection() 호출

  ✓ 임계 영역 사용이 끝나면 DeleteCriticalSection() 호출

타입	변수	설명
SOCKET	socket	클라이언트 소켓
SOCKAADR_IN	sock_addr	클라이언트 주소
ID	Client_id	식별 id
Int	mPlayerCnt	플레이어 수
Player[]	mPlayers	플레이어들 정보 배열
	eType	플레이어 객체 타입 범인 or 일반인
	eMissionType	임무 타입
	eMissionInfo	임무 정보

B. 함수설명

I. Login

- 서버: PlayerData() : 접속한 클라이언트의 주소를 받아 고유 id 생성.
- 클라이언트: ClientThread(): 클라이언트 데이터 관리 쓰레드

II. Lobby

- CreateLobby(): 로비 생성함수
- PlayerWait() : 로비에 접속한 플레이어가 4명이되면 게임시작 버튼 활성화
- Initalize(): 플레이어의 객체 타입, 임무, 좌표 등 플레이어에 대한 정보 초기화
- PlayerSetting() : 플레이어의 객체 타입, 임무 등 설정
- Start(): 게임시작, 인게임 접속

B. 인게임 – 임무수행

I. 구조체 및 열거형

- Enum PlayerType

상수	설명
CITIZEN1	시민1
CITIZEN2	시민2
CITIZEN3	시민3
CRIMINAL	범인

- struct PlayerInfo

타입	변수	설명

enum PlayerType	eType	객체 타입
POINT	pos	객체 위치
float	fKeyInput	키보드 입력 값
RECT	collisionBox	충돌 박스

- enum MissionType (미션 개수만큼 상수 지정)

상수	설명
MISSION1	임무 1
MISSION2	임무2
MISSION3	임무 3
MISSION4	임무4
…	…

- struct PlayerRes

타입	변수	설명
enum PlayerType	eType	객체 타입
POINT	pos	조정된 객체 위치
enum MissionType	eMissionType	임무 종류
bool	bMissionRes	임무 결과

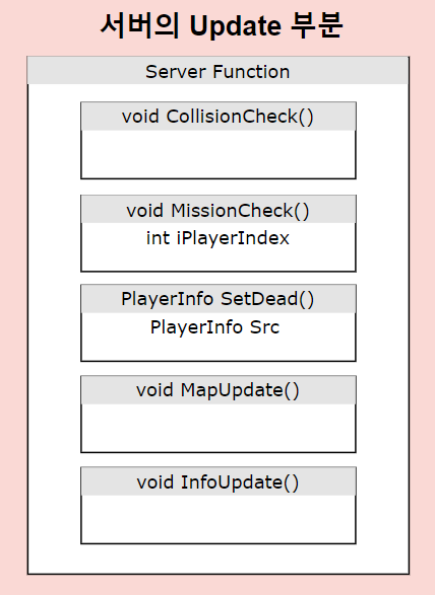
II. Client 함수

- SetPlayerInfo(): 서버로 보낼 플레이어의 정보

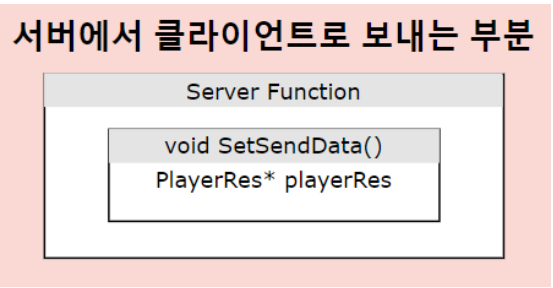
- UpdatePlayerInfo(): 서버에서 받아온 정보로 플레이어 정보 갱신. 객체 위치 조정, 어떤 임무를 수행했는지를 받아오면 해당 임무를 수행완료로 바꾸기.
- UpdateMapInfo(): 서버에서 받아온 정보로 맵 갱신. 임무가 완료되었으면 완료된 표시. 임무 진척도 증가 등 서버에서 변한 정보 확인하여 바꾸기.

III. 서버 함수

- CheckPlayerInfo(): 클라이언트에서 받아온 정보로 플레이어의 타입을 파악하여 어떤 함수를 호출할 지 정하기. Ex〉 플레이어1이면 무슨 임무를 수행해야 하는지 등

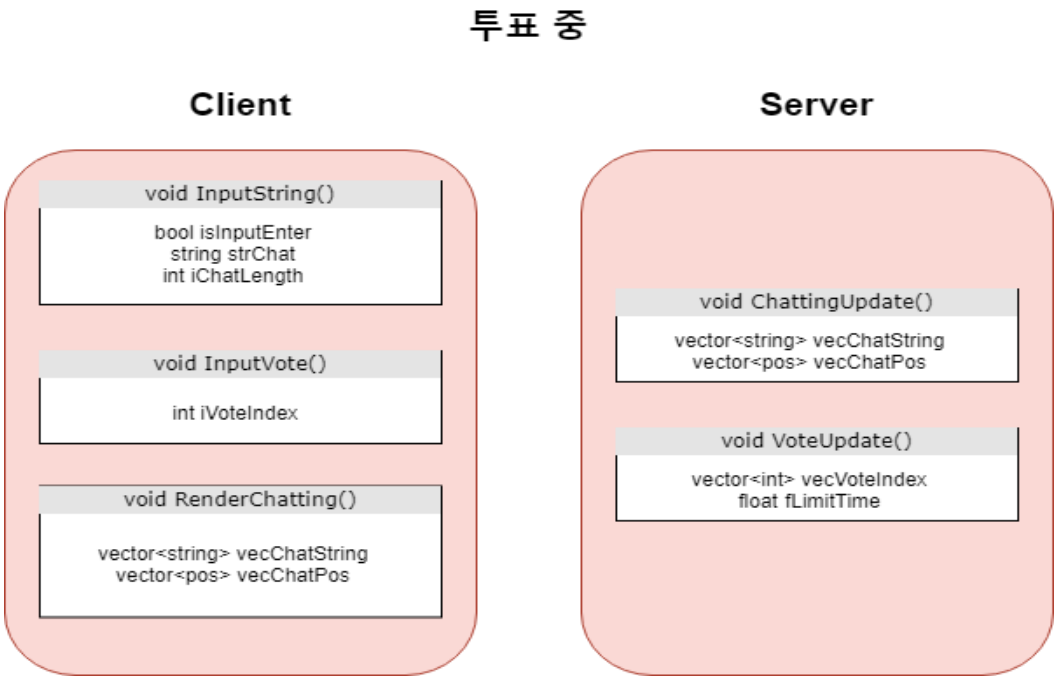


- CollisionCheck(): 충돌 체크하는 함수. 객체 타입에 따라 충돌체크 다르게 함. 플레이어가 범인일 경우 모든 플레이어와 범인, 플레이어가 일반인일 경우 임무 오브젝트와 충돌 체크한다.
- MissionCheck(): 충돌체크 이후 호출하는 함수로, 일반인일 경우 어떤 임무를 수행했는지 확인한다.
- SetDead(): 충돌체크 이후 호출하는 함수로, 범인일 경우 어떤 플레이어와 충돌했는지 확인하여 충돌한 플레이어 사망상태로 바꾼다.
- MapUpdate(): 충돌 처리된 데이터로 맵 갱신 임무, 플레이어 사망, 플레이어 이동 등
- InfoUpdate(): 임무진척도, 각 플레이어의 남은 임무 등을 갱신한다.



- SetSendData(): 갱신된 정보를 보낼 데이터의 구조체에 맞게 채워 넣는다. 이 데이터는 클라이언트가 받아서 클라이언트에서 데이터를 갱신하는데 사용한다.

C. 인게임 – 투표



A. Client

- InputString()
  ① 엔터키를 입력 받았는지 확인
  ② 입력이 있으면 현재 입력되는 문자열을 저장
  ③ 다시 엔터키 입력이 있으면 문자열을 서버로 전송

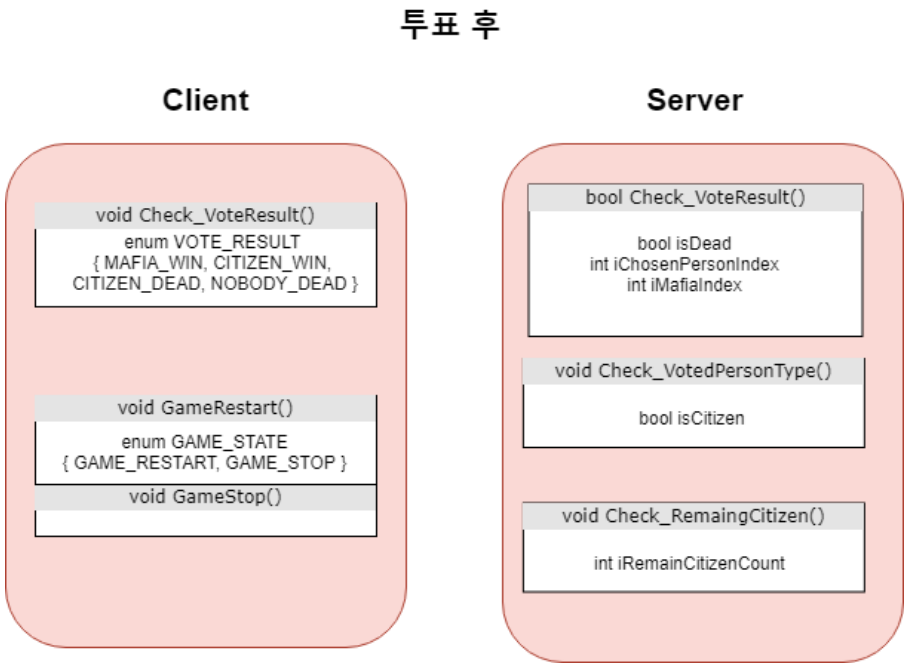
- InputVote()
  ① 숫자키1~4까지 입력을 확인
  ② 입력이 있다면 입력된 번호를 인덱스로 저장 후, 서버에 전송

- RenderChatting()
  ① 서버로부터 받아온 문자열들을 좌표값(뷰포트 좌표)을 이용하여 TextOut()함수로 렌더

B. Server

- ChattingUpdate()
  ① 모든 클라로 부터 받은 문자열 저장
  ② 모든 클라로 부터 받은 문자열을 출력할 위치 저장

- VoteUpdate()
  ① 모든 클라로 부터 받은 투표 결과를 저장
  ② 투표 제한시간을 확인



〈Client〉

- Check_VoteResult()
  ① 서버에서 확인한 투표 결과를 enum값으로 전달받음.
  ② 마피아가 이겼을 경우 혹은 시민이 이겼을 경우 게임 종료 상태 확인
  ③ 시민이 죽었을 경우, 혹은 아무도 죽지 않았을 경우 게임은 계속 진행

- GameStop(), GameRestart()
  ① 투표 결과로 인한 게임 상태를 바꿔 줌

- Check_VoteResult()
  ① 투표결과가 과반수 이상인지 확인
  ② 투표로 선택된 사람(인덱스)을 저장
  ③ 만약 마피아가 지목받았으면 시민승리

- Check_VotedPersonType()
  ① 투표로 선택된 사람이 시민인지, 마피아 인지 확인

- Check_RemaingCitizen()
  ① 현재 남은 시민 인원 수 확인
  ② 만약 마피아와 시민이 일대일 상황이면 마피아 승리

- 멀티스레드 동기화

① 각 클라이언트가 접속하는 순서대로 이벤트를 생성하면 총 4개의 이벤트가 있을것이고, 그 순서대로 이벤트를 이용하여 스레드를 돌리고, send, recv가 될것

② 3개의 스레드는 시민 전용 함수가 실행되고, 1개의 스레드는 마피아 전용 함수가 실행된다.

4. 팀원 별 역할분담

5. 개발환경

A. Git Hub, VS2019, WIN API

6. 개발일정

A. 클라이언트 개발(플레이어 기본 움직임 애니메이션)

B. 접속한 클라이언트들을 화면에 출력

분량

1. 어플리케이션 기획: 3p
2. 하이레벨: 5p
3. 로우레벨: 8p (각자 2~3p씩)
4. 팀원별 역할분담: 1p
5. 개발환경: 1p
6. 개발 일정: 2p