

# LEAGUE OF RUNNERS



## **TERM PROJECT PROGRESS REPORT** **PROJECT "LEAGUE OF RUNNERS"**

김나단 2013180003  
강태규 2013182002  
원성연 2013182027

# PROJECT “LEAGUE OF RUNNERS”

## 목차

### 애플리케이션 기획

개요	2
게임 기본 규칙 및 승리 조건, 조작	3
캐릭터	4
UI	5

### High-Level 디자인

게임 씬 구조	6
서버 - 클라이언트 구조	7
서버 구조	8
클라이언트 - 베이스 구조	10
클라이언트 - 호스트 구조	12
클라이언트 - 게스트 구조	13
전송 지연 시간 미 고려, Guest 주도 계산 방식	14
평균 지연 시간을 활용한 Guest 주도 계산 방식	15
전송 지연 시간 미 고려, Host 주도 계산 방식	16
멀티 쓰레드 구조	17

### Low-Level 디자인

Communication protocol	18
Server	20
Client-Base	20
Client-Host	21
Client-Guest	22

팀원 별 역할 분담	23
------------	----

개발 환경	24
-------	----

개발 일정	25
-------	----

## 애플리케이션 기획



### “신개념 온라인 횡스크롤 Running Game”

League Of Runners	
장르	러닝 액션 게임
시점	2D
플레이 방식	횡 스크롤
게임 구성	유한, 레이싱형
조작	키보드(Game), 마우스(UI)
맵	2 개
캐릭터	4 개 (러너형 2 개, 어태커형 1 개, 서포터형 1 개)
아이템	4 개(어태커형 2 개, 서포터형 2 개)

## ■ 게임 기본 규칙 및 승리 조건 수정사항은 빨간색으로 표시하였습니다.

- 게임은 2 VS 2, 팀전으로 진행됩니다.
- ~~맵의 도착점에 도달한 순서에 따라, 순위를 부여합니다.~~
- ~~무조건 1등이 속한 팀이 승리합니다.~~
- Plat 에 도달할 때 마다 콤보가 쌓입니다.
- 콤보가 50~~20~~ 의 배수일 때마다 속도가 증가합니다.
- 일반적으로 다음 중 하나의 조건을 만족할 때, 이상 상태가 됩니다.
  1. 공격형 아이템(~~번개~~)에 피격 시, (~~슬리핑은 콤보 미 초기화~~)
  2. Plat 에 도달하지 못하고, 캐릭터가 화면 아래로 떨어졌을 때
  3. ~~장애물에 부딪혔을 때 (충돌은 아이템 박스와의 충돌로 대신)~~
- 이상 상태가 된다는 것은, 아래와 같은 영향을 받습니다.
  1. 콤보가 0 으로 초기화됩니다.
  2. 속도가 캐릭터 0, 또는 기본 속도로 변경됩니다.
  3. 이상 상태에 따른 일정 시간 조작 불가 등의 영향을 받습니다.
- 그러나 서포터형 아이템 중 각 조건에 대응하는 아이템(~~방패~~)이 사용 중이면 위 조건을 만족시키더라도 , 초기화되지 않습니다.
- ~~또, 이상 상태일 경우, 서포터형 아이템(천사)를 사용하면, 이상상태에서 바로 회복하며, 일정 시간 속도가 증가합니다.~~
- 아이템은 아이템 상자와 충돌 시, 획득할 수 있으며, 해당 아이템 상자는 사라지지 않음을 원칙으로 합니다.

## ■ 조작

SPACE

: 점프

Q

W

: 아이템 1, 아이템 2 사용

1

2





3

4

: 감정표현 1, 2, 3, 4

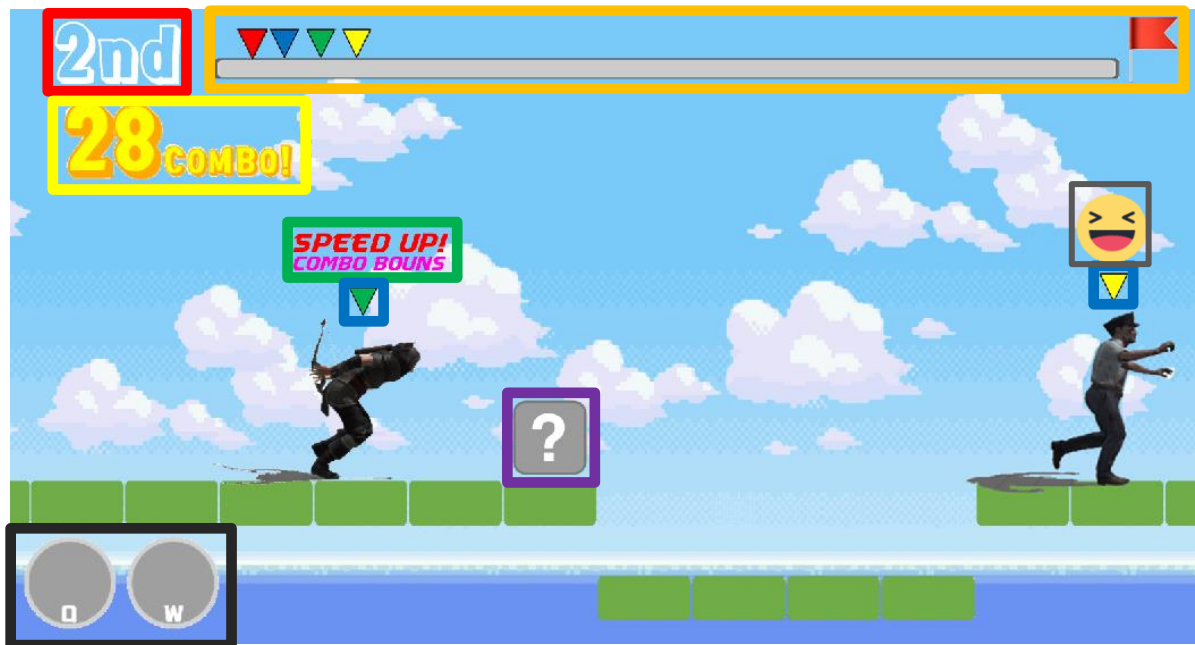
## ■ 캐릭터

- 캐릭터는 그 특징과 역할에 따라 러너형(Runner), 어태커형(Attacker), 서포터형(Supporter)으로 구분됩니다.

Archer	Zombie
	
<b>Runner</b>	<b>Runner</b>
속도 : ★★★★★ 트리플 점프가 가능합니다.	속도 : ★★★★★ 일정확률로 장애물을 무시합니다.
Knight	witch
	
<b>Attacker</b>	<b>Supporter</b>
속도 : ★★★ 공격형 아이템 적중 시, 적 상태 이상 시간, 효과 증가	속도 : ★★★ 서포터형 아이템 사용 시, 아군 이로운 상태 시간, 효과 증가



## ■ UI

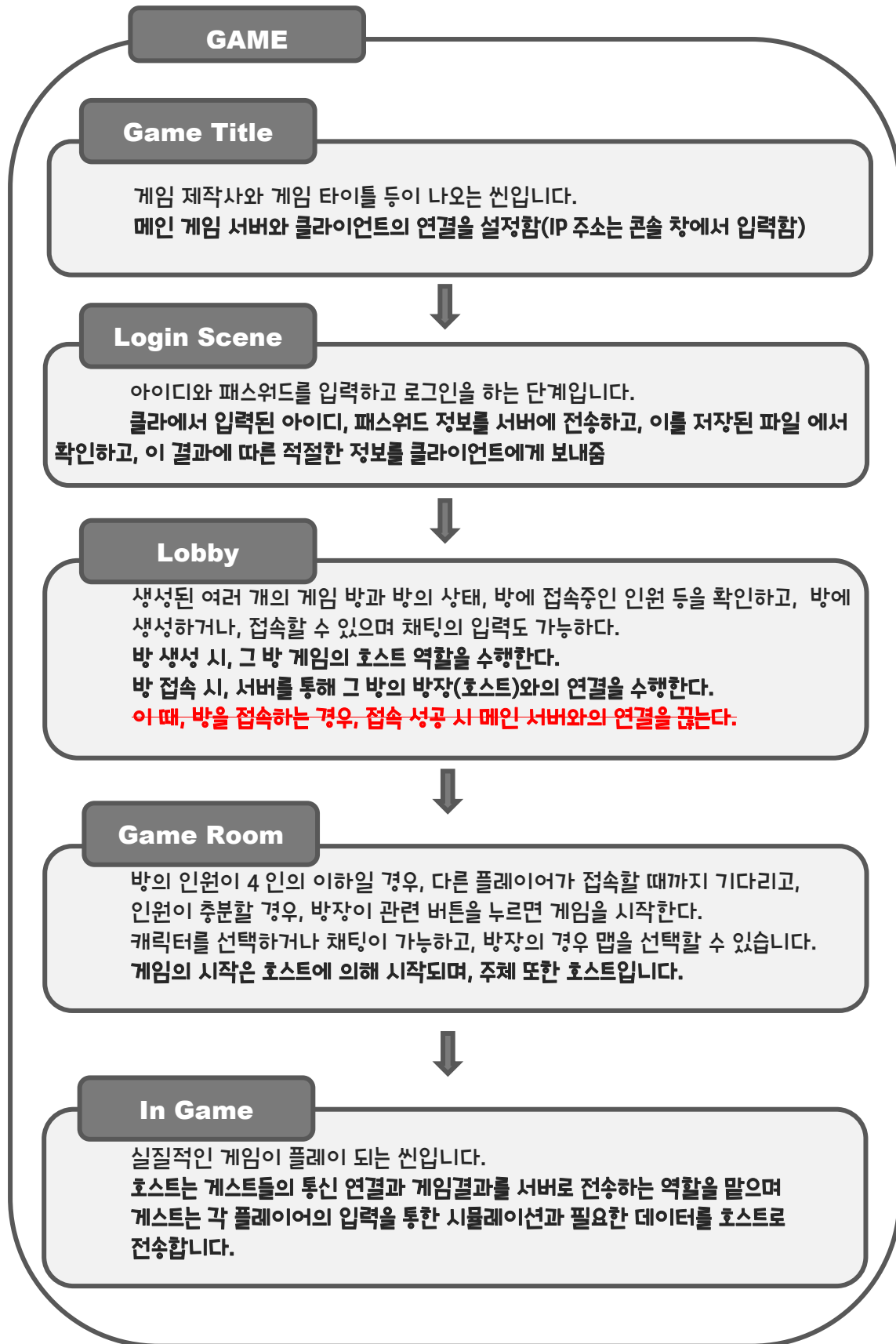


- 순위 UI : 현재 플레이어의 순위를 표시합니다.
- Bar UI : 게임 진행도와 상대 플레이어의 위치를 표시합니다.
- Combo UI : 현재 플레이어의 Combo 를 표시합니다.
- 상태 UI : 캐릭터의 상태를 표시합니다.
- 식별 UI : Bar UI 와 동일한 색으로 표시해주어 각 플레이어를 식별할 수 있도록 합니다.
- 아이템박스 : 아이템 박스와 충돌 시, 아이템을 획득할 수 있습니다.
- 이모티콘 UI : 다양한 감정 표현이 가능하며, 이를 다른 플레이어에게 보여줍니다.
- 아이템 UI : 현재 플레이어가 소유하고 있는 아이템을 보여줍니다.

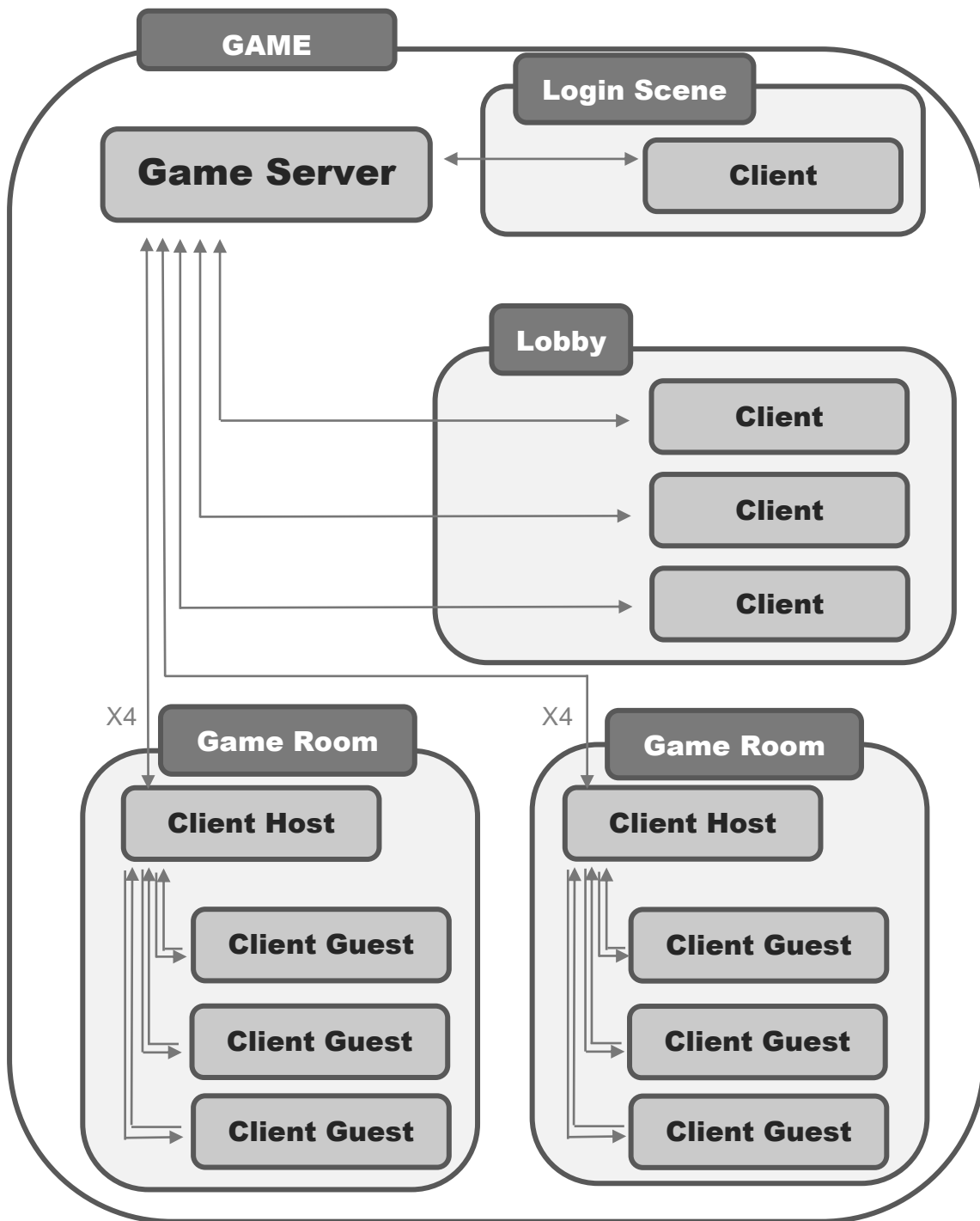


# High-level 디자인

## ■ 게임 씬 구조



## ■ 서버 - 클라이언트 구조



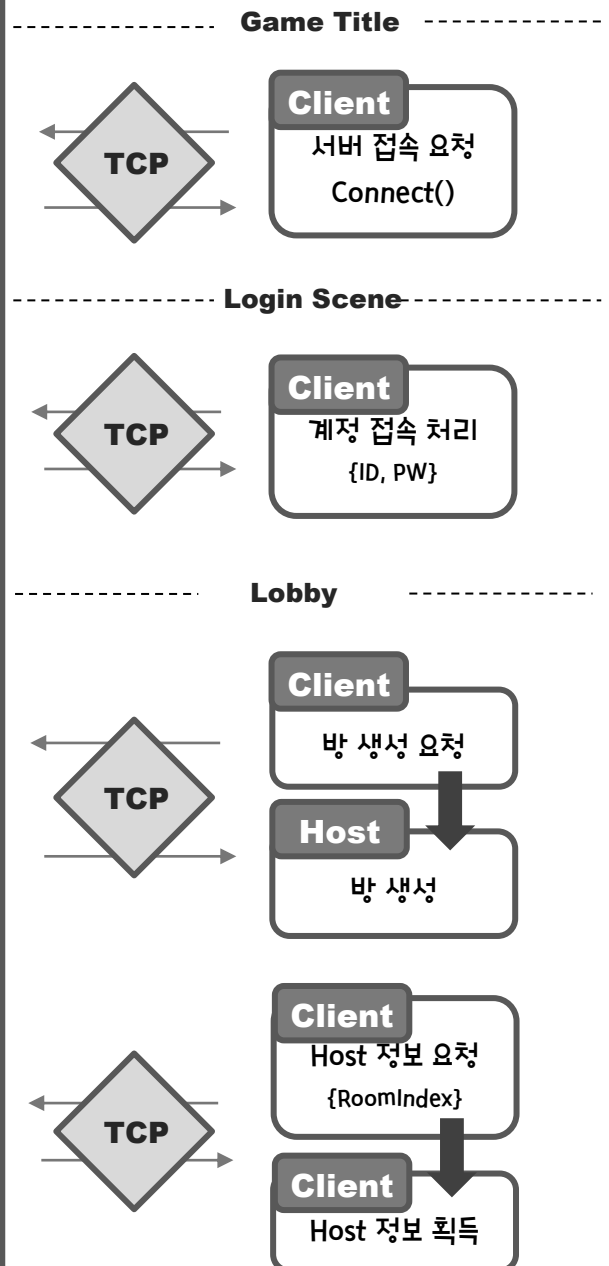
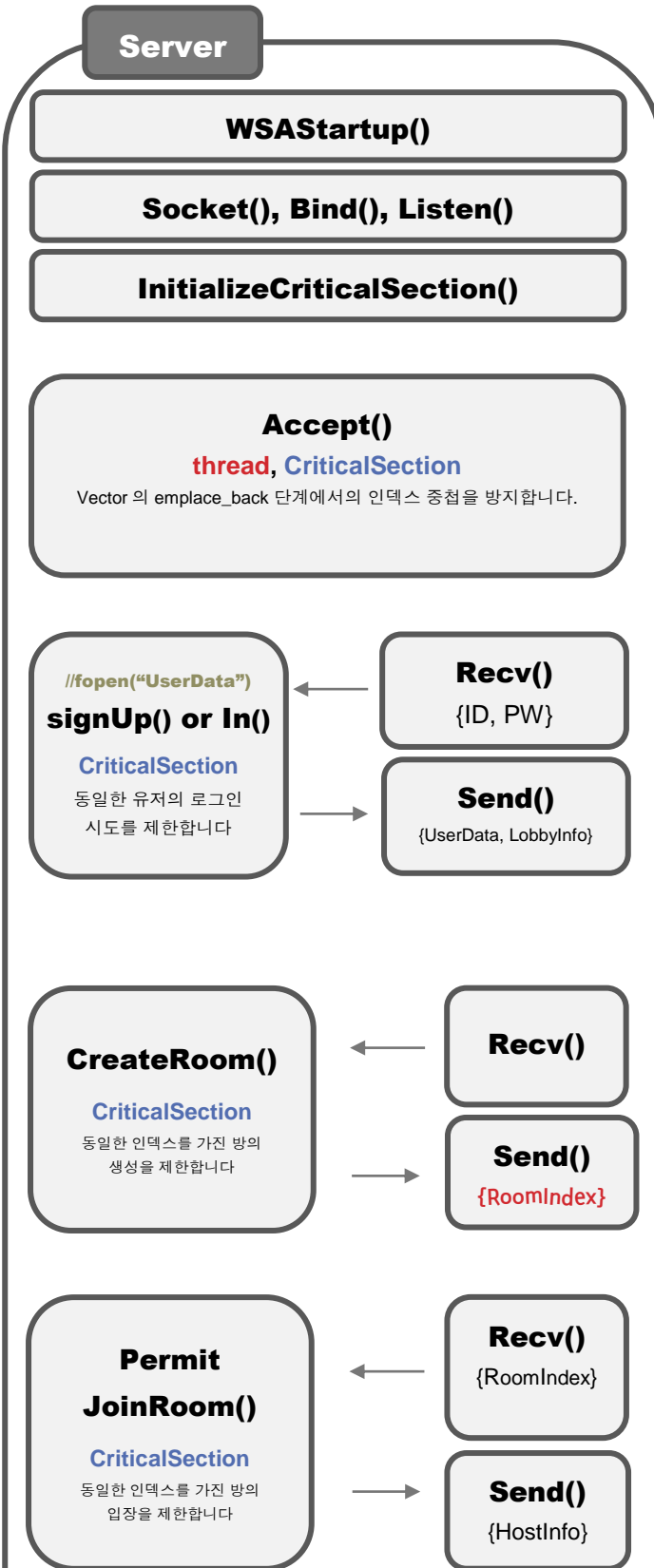
### 교착상태를 해결하기 위한 방안

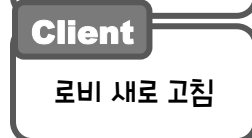
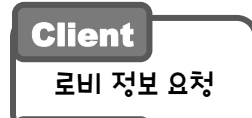
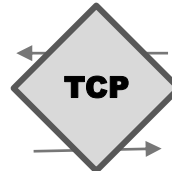
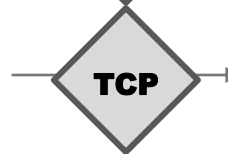
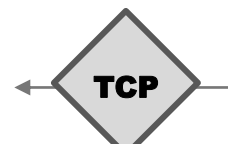
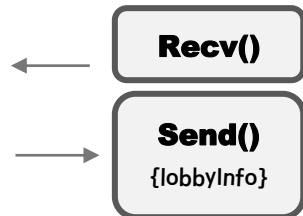
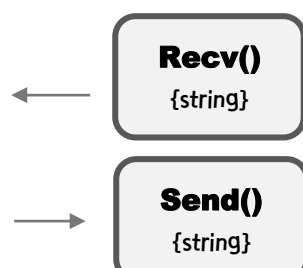
1. LoginScene, Lobby에서는 성능을 크게 요구하지 않기 때문에, 소켓에 적절한 타임아웃 옵션을 적용합니다. (SOCKET, SO\_RCVTIMEO, 200 (0.2 초))
1. 적절한 네트워크 설계를 통해 교착 상태를 극복하였습니다.
2. 높은 성능, 빠른 작동이 요구되는 부분인 inGame, GameRoom에서는 타임아웃이 부적절하기 때문에, TCP 소켓을 두 개를 생성하여 Recv 전용 소켓과 Send 전용 소켓을 통해 교착상태를 해결하고자 했습니다.



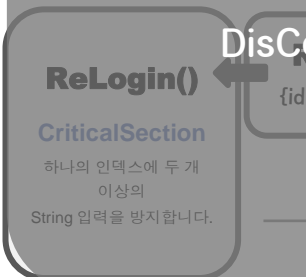
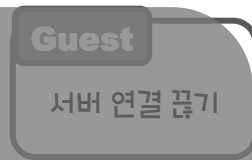
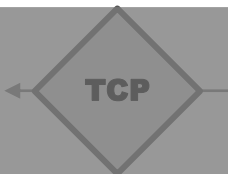
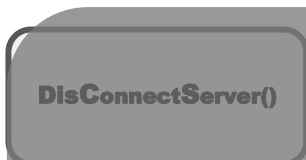
## ■ 서버 구조

- 메인 게임 서버는 그래픽이 없으며, 통신을 위해서 사용됩니다.

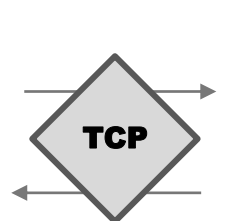
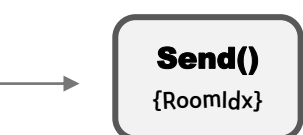
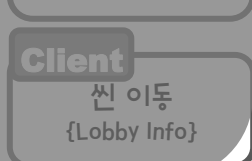
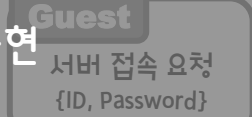
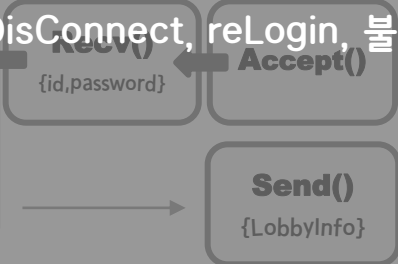




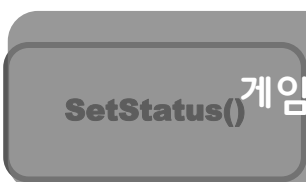
### Game Room



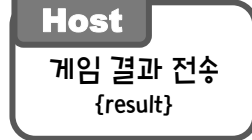
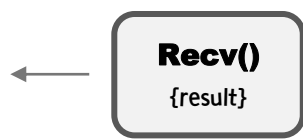
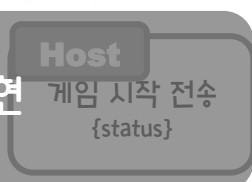
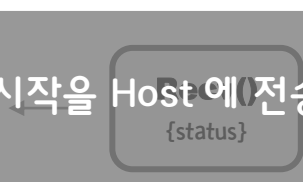
DisConnect, reLogin, 불필요에 따른 미 구현



### In Game



게임 시작을 Host에 전송하지 않아 미 구현



## ■ 클라이언트 Base 구조

### Client

WSAStartup()

Socket()

InitializeCriticalSection()

Connect()

SignIn()  
SignOut()

Send()  
{ID, PW}

SetUserData()  
ChangeScene()

Recv()  
{userData, lobbyInfo}

DemandChat()

Send()  
{string}

UpdateChat()

Lobby Info 에서 종합적으로 이 부분 처리

Recv()  
{ string }

Demand  
RefreshLobby()

Send()

Permit  
RefreshLobby()

Recv()

### Game Title

TCP

### Server

Accept()

### Login Scene

TCP

### Server

로그인 정보 확인  
{id, pw}

### Server

유저, 로비정보  
{userData, lobbyInfo}

### Lobby

TCP

### Server

채팅 내용을 처리  
{string}

TCP

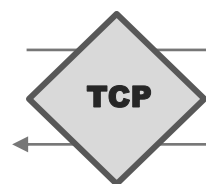
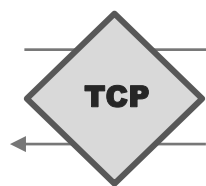
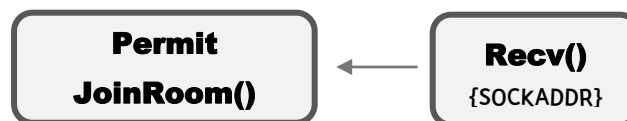
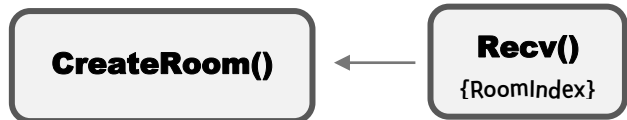
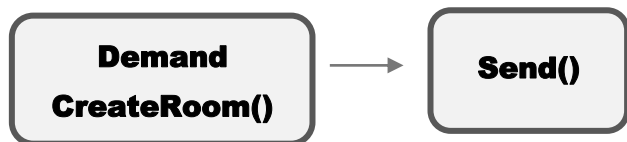
### Server

채팅 데이터 전송  
{strings}

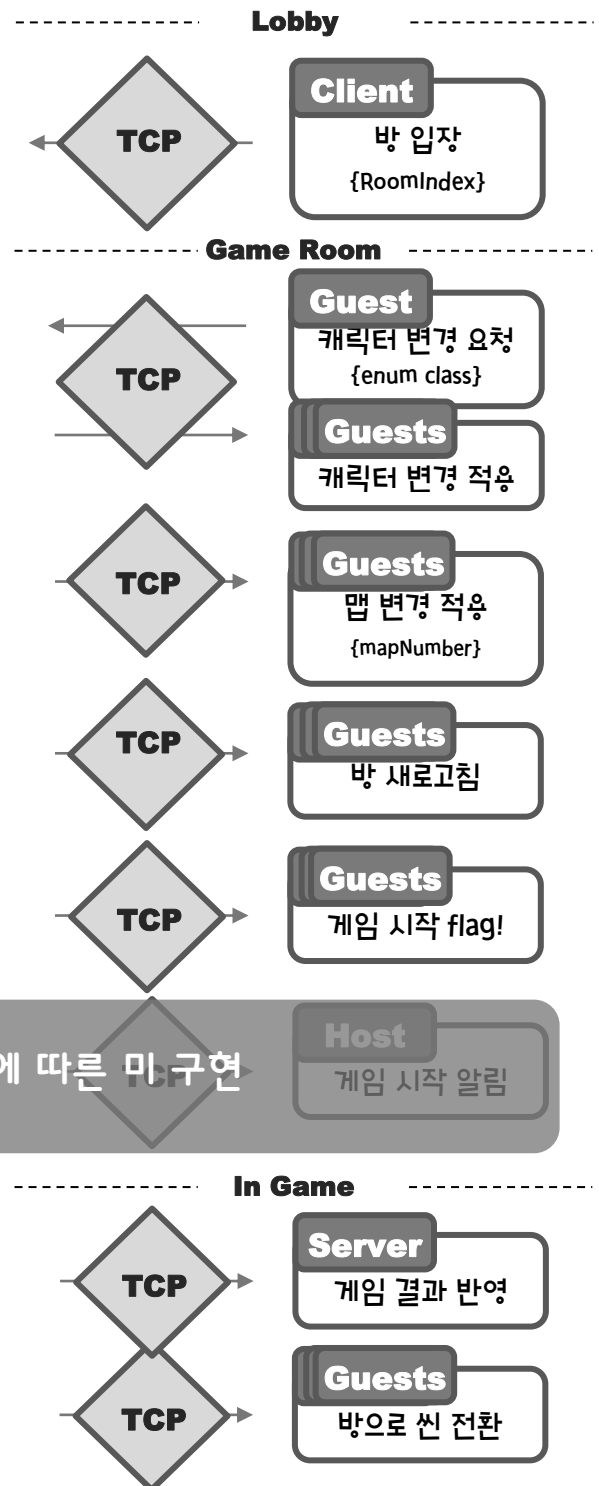
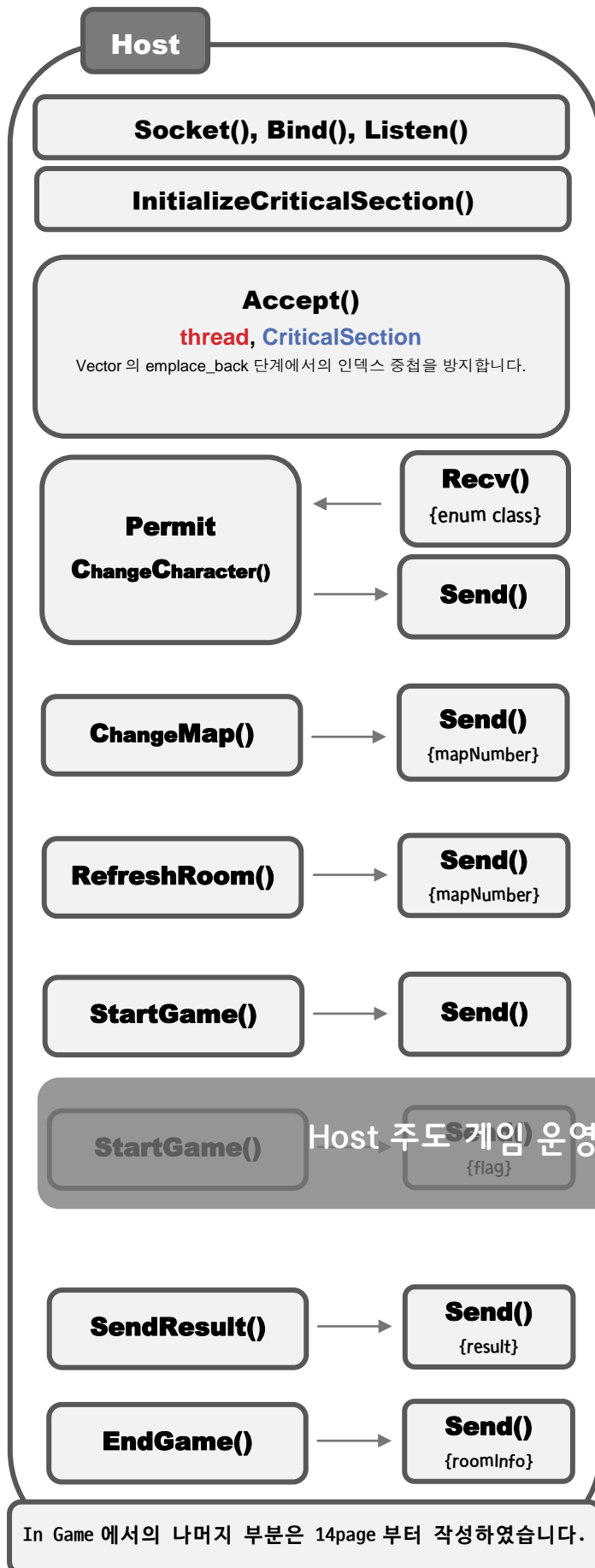
TCP

### Server

로비 정보 전송  
{채팅, 방}

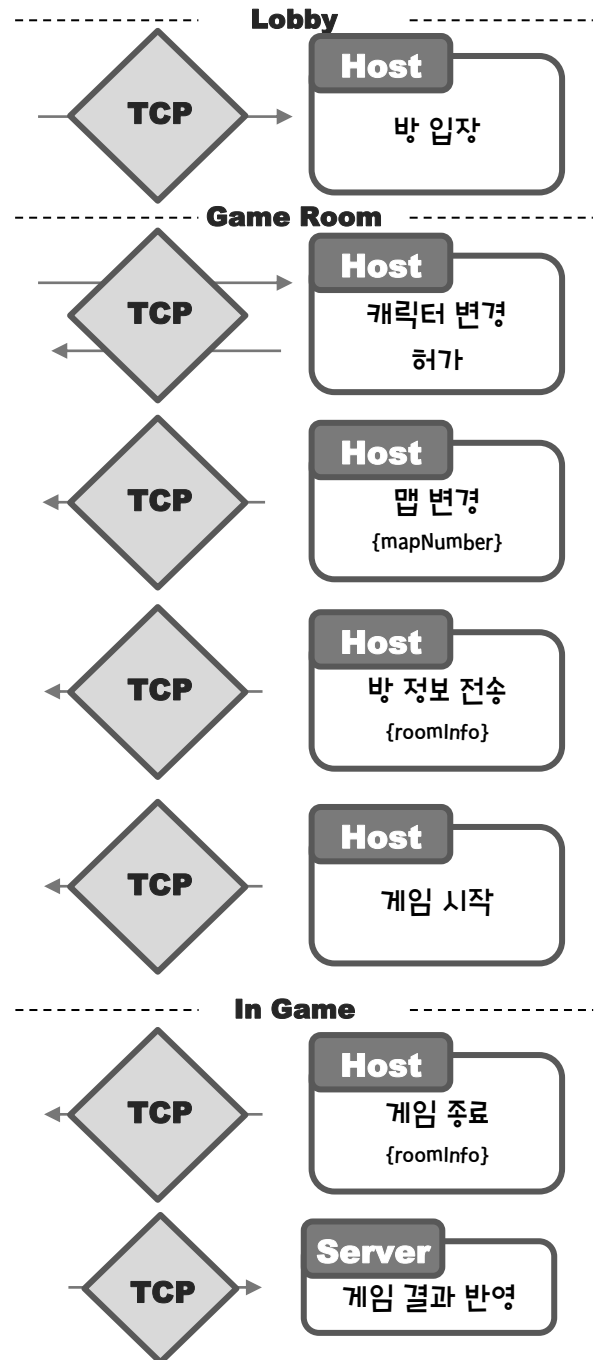
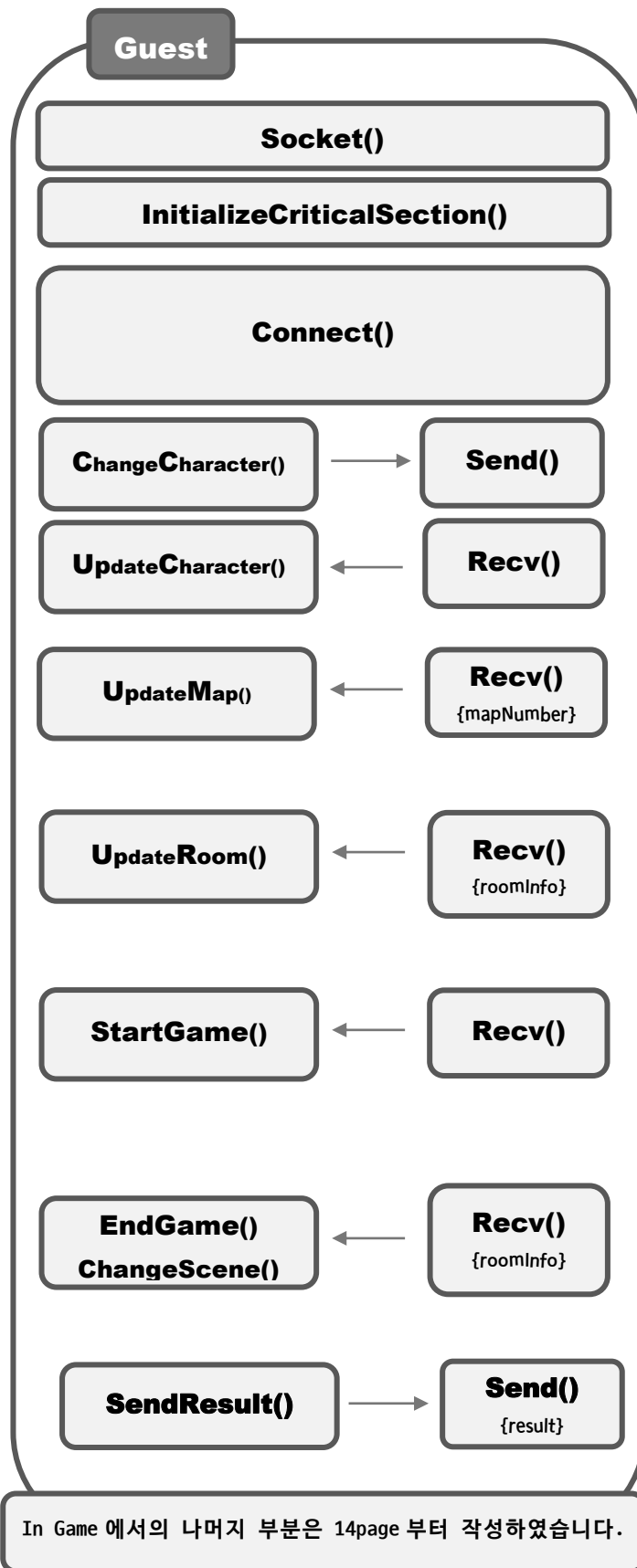


## ■ 클라이언트 Host 구조





## ■ 클라이언트 Guest 구조



## ■ In Game 네트워크 구현의 두 가지 방안

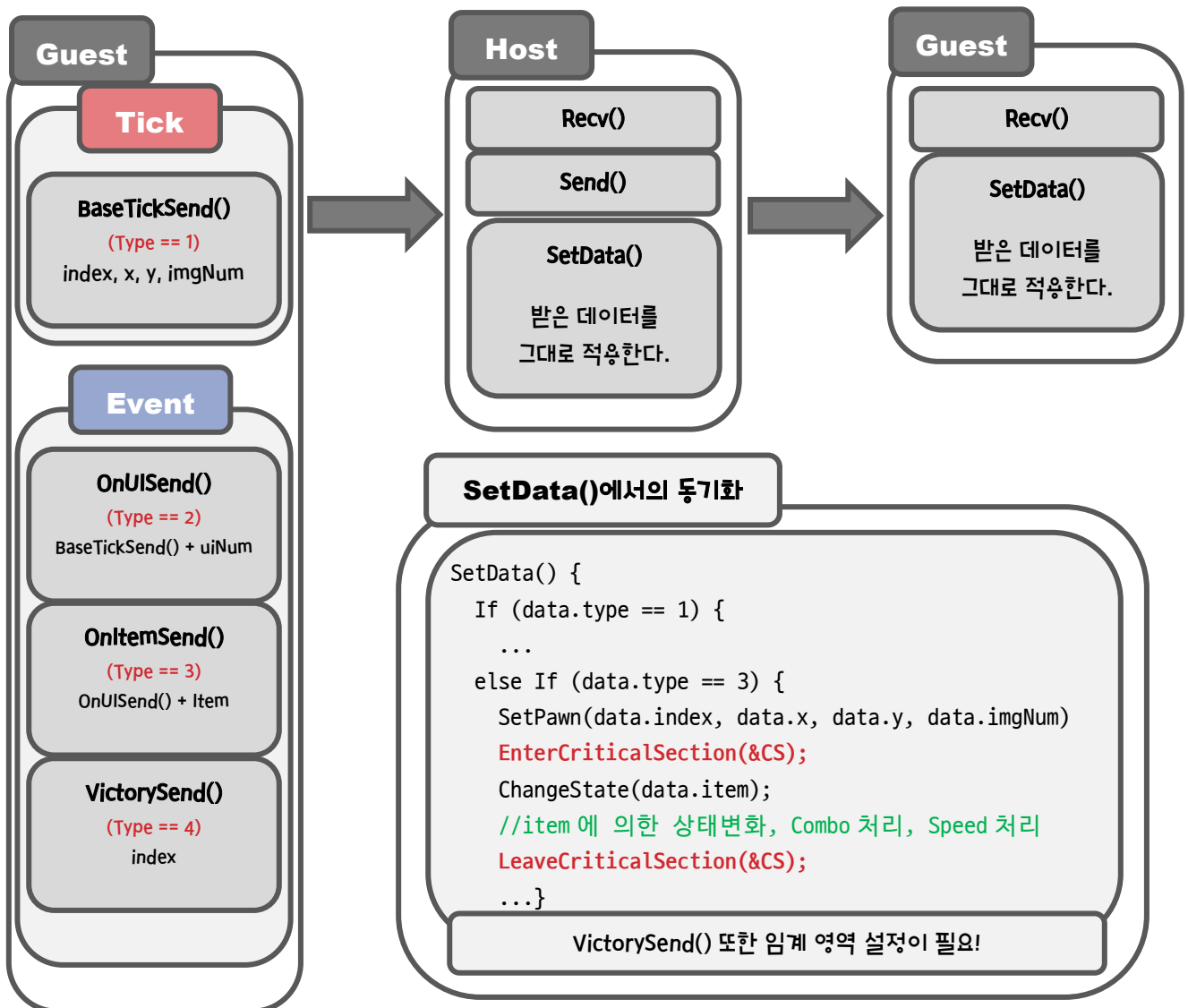
—In-GameScene 에서 가장 중요한 이슈인 시간 동기화를 구현하기 위해 두 가지 방법을 최종적으로 고려했으며, 1안 구현 후 문제가 있다고 파악될 경우, 2안의 개발에 바로 착수, 구현하여 1안과 2안을 비교하여 결정합니다.

— 추진 계획서 평가 과정에서, 교수님의 피드백에 따라 게스트에서는 입력과 드로우만 처리하고 실질적인 게임 업데이트는 호스트에서 처리하도록 구현, 즉 2안으로 개발하였습니다. 1안은 peer to peer 입니다.

### 방법 1.1) 전송 지연 시간 미 고려, Guest 주도 계산 방식

— 데이터 전송 지연 시간을 고려하지 않기 때문에, Guest 에서 자신의 캐릭터 좌표와 상태 등을 계산하고, 이 결과를 Host 에 전달하며, Host 가 이 결과를 어떠한 처리 없이 다른 게스트에 전송하여 적용.

가정 : Host 와 Guest 사이의 전송 시간 지연의 차이는 무시 가능한 수준일 것이다.



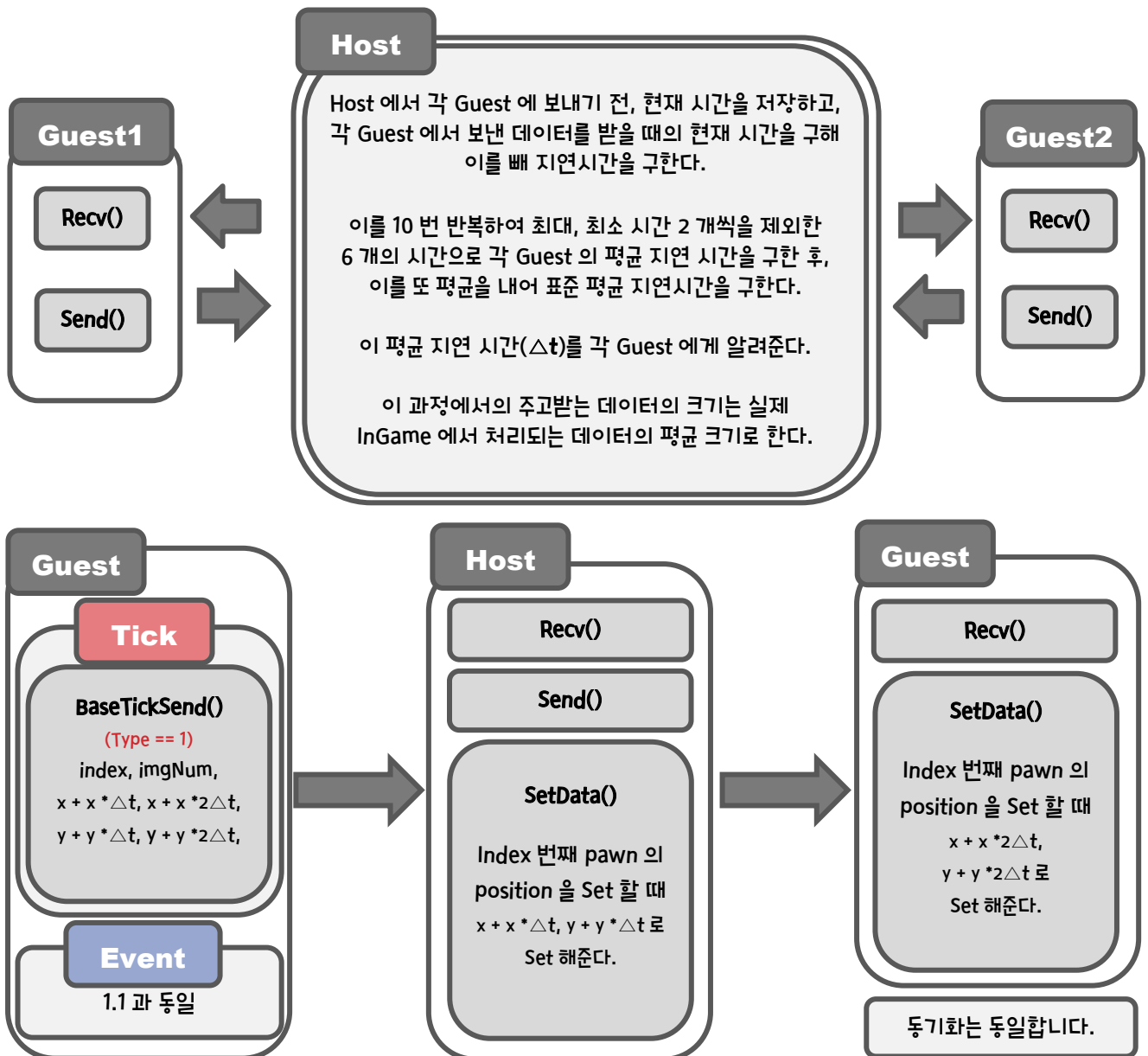
평가 사항 (단점) : Host 와 Guest 사이의 전송 시간 지연의 차이가 응답이 가능하며 올바른가?

## 방법 1.2 ) 전송 지연 시간 고려, 평균 지연 시간을 활용한 Guest 주도 계산 방식

- 최초 게임 시작 시, Host 와 Guest 간의 평균지연시간을 Host 에서 계산하여, 이를 각 Guest 에 전달한 후, 자신의 캐릭터 좌표와 상태 등을 계산하고, 이 결과에 평균 지연 전송 시간을 적용한 값을 Host 에 전달하며, Host 가 이 결과를 다른 게스트에 전송하여 적용합니다.

가정 : 전송 지연 시간에 의한 차이는 존재하나 어느 정도 일정하게 유지되며, 이 평균값을 통하여 처리한 결과에서의 차이는 무시 가능한 수준일 것이다.

### 게임 시작 시, 평균 지연 시간 구하는 방안 고려

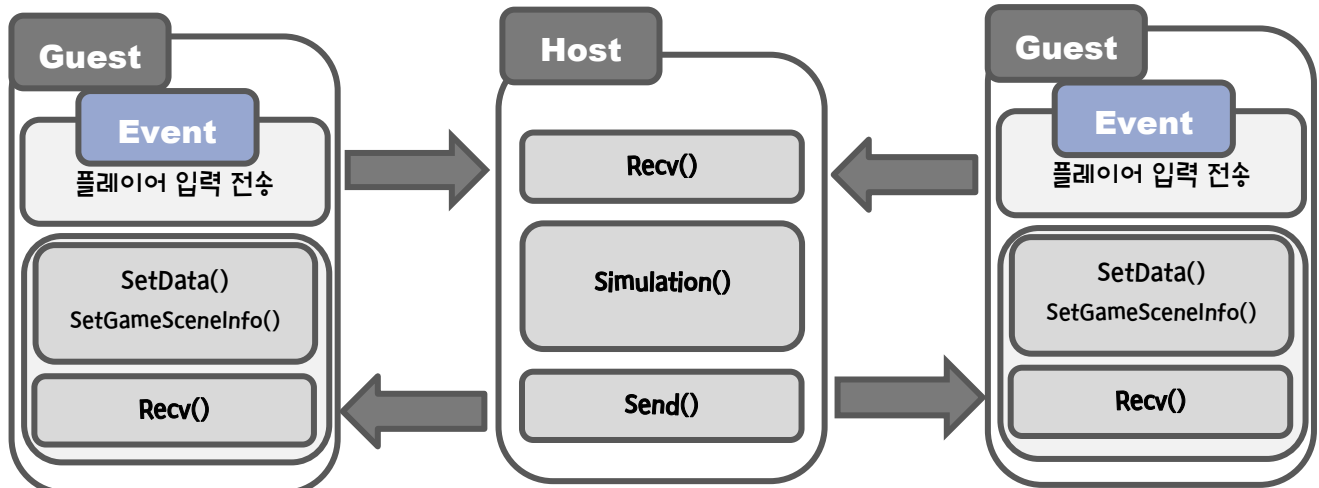


평가 사항 (단점) : 평균 지연 시간을 통한 동기화가 응답이 가능한 수준인가?

## 방법 2) 전송 지연 시간 미 고려, Host 주도 계산 방식

- 1.1 과 1.2 안에서 평가 사항을 통과하지 못할 시, **2 안으로 먼저 개발합니다.** Host 에서 모든 것을 시뮬레이션하고, 이를 각 Guest 에게 현재 장면에 대한 정보를 보내주는 것으로 구현합니다. Guest 는 1 안과 달리, 플레이어의 입력 정보를 전송합니다.

**가정 :** 전송 지연 시간에 의한 차이는 실질적으로 존재하나, 그 값이 작아 무시하며, Host 가 비교적 동등하게 Client 에게 정보를 전달한다.



### Simulation()에서의 동기화

#### Event 를 통한 Send() 와 Recv() 동기화

RecvData 에서 WaitForMultipleObjects 로 모든 sendEvents 가 Set 되기를 기다린다.

Recv 한 데이터를 버퍼에 저장한다. readEvents 를 모두 Set 한다.

SendData 에서 WaitForSingleObject 로 readEvents[자기인덱스] 이벤트가 Set 되기를 기다린다.

버퍼의 내용을 send 한다. sendEvents[자기 인덱스]를 Set 한다.

#### Critical\_Section 을 통한 Status 동기화

공격용 아이템에 의하여 공격을 당할 시에,

- 1) 콤보를 0 으로 바꾸고, 2) 속도를 0 으로 변경, 3) 상태 "기절"로 변경,
- 4) 기절 후 일정시간을 측정하여 이 후, 속도를 기본 속도로 변경해주는 타이머 작동.

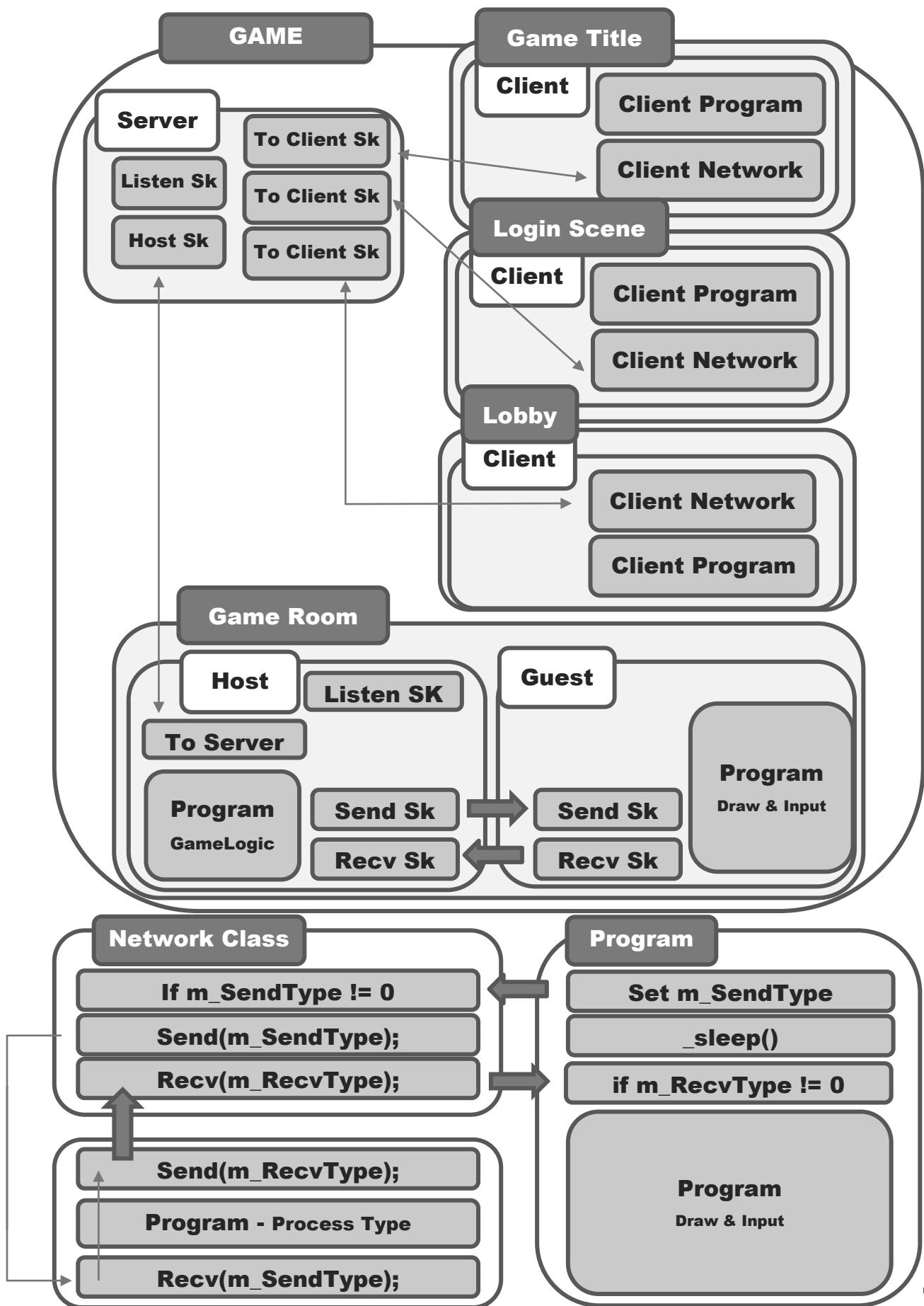
이 때, 임계 영역 설정이 안되어 있는 상황에서 서포터용 아이템에 의해 공유 자원인 콤보, 속도, 상태에 비정상적인 결과가 나올 수 있기 때문에 임계 영역을 사용합니다.

#### Queue 을 통한 동기화

1) 보내야 하는 데이터가 있는 경우, Host 는 어떤 데이터를 누가 누구에게 보내는지 알아야 하므로 보낼 데이터 종류, 누가 보냈는지를 저장하는 Queue 를 구성하고, Queue 는 인원 수 만큼 생성하여 Queue 의 인덱스는 누구에게, 큐에 저장할 데이터로 보낼 데이터 종류, 누가 보냈는지에 대한 구조체를 사용한다.

2) Guest 의 경우 자신이 호스트에게 보내는 것이 결정되어 있으므로 어떤 종류의 데이터를 보낼지를 Queue 에 저장하여 사용한다. 실제 Send 에서는 While 문 내부에서 큐가 비었는지를 검사하며 큐에 데이터가 있는 경우 보내야 할 데이터가 있는 것으로 판단하여 그 보내야 할 데이터의 종류에 따라 작업을 진행한다.

## ■ 멀티 쓰레드 구조





# Low-level 디자인

## 0. Communication protocol

- 기존의 Protocol 를 사용하지 않고, 다른 방식으로 적용했습니다.  
기존 방식은 유연하지 않아 확장, 변경에 있어 문제가 있을 것으로 판단하여,  
썬마다 여유있게 할당하여, 유연성있도록 변경, 새로 규정하여 적용했습니다.

```
enum Protocol{
    //LoginScene
    DEMAND_LOGIN          = 100
    , FAIL_LOGIN          = 101
    , PERMIT_LOGIN         = 102

    //LobbyScene
    , DEMAND_CREATEROOM   = 200
    , FAIL_CREATEROOM     = 201
    , PERMIT_CREATEROOM   = 202

    , DEMAND_JOINROOM     = 203
    , FAIL_JOINROOM       = 204
    , PERMIT_JOINROOM     = 205

    , DEMAND_LOBBYINFO    = 206
    , PERMIT_LOBBYINFO    = 207

    , DEMAND_CHAT         = 208
    , PERMIT_CHAT         = 209

    , REQUEST_ROOMINFO    = 210    //Server -> Host
    , SUPPLY_ROOMINFO     = 211    //Host  -> Server

    , UPDATE_LOBBY        = 212

    //RoomScene
    , SUPPLY_GAMESTART    = 395    //HOST  -> Server

    , DEMAND_EXITROOM     = 396    //Host  -> Server
    , PERMIT_EXITROOM     = 397    //Server -> Host

    , DEMAND_RELOGIN      = 398    //GUEST -> Server
    , PERMIT_RELOGIN      = 399    //

    //INGAMESCENE
    , SUPPLY_GAMEEND      = 499    //HOST  -> Server
};
```

# 1. Server

**DWORD WINAPI ProcessClient(LPVOID arg) // Server -> Client-Base**

서버-클라이언트에서 서버가 클라이언트의 요청을 recv()하고 그에 대한 처리를 send 하는 함수

**void CreateRoom(LPSOCKADDR addr, struct& RoomData)**

[Critical Section] 방을 만드는 함수(Client 를 Host 로 바꿈), addr 로 신청한 클라의 ip 를 받는다.

**bool PermitJoinRoom(SOCKADDR \*addr, int roomIndex)**

[Critical Section] 게스트가 방에 접속을 원할 시, 이를 처리해주는 함수

~~**int GetRoomInfo(int roomIndex)**~~

~~서버가 호스트에 roomIndex 에 해당하는 방 정보 요청 => 서버에서 방에 대한 정보를 관리~~

~~**void SetRoomStatus(int status) // status == 5 -> start;**~~

~~방의 상태를 Server 에서 설정한다. => 방의 상태 자체는 호스트에서 컨트롤~~

**void InitCS(void)**

임계영역들을 초기화

**bool ReadUserData(const struct& UserData, const struct& users)**

받은 아이디와 비밀번호로 검색해서 맞는지 확인해서 맞으면 users 에 저장하고 true 반환

**void WriteUserData(const struct& data)**

유저 데이터를 쓰기 위한 함수

**void RefleshLobby(SOCKADDR \*addr)**

방 목록을 새로고침 요청한 클라이언트에 보내준다.

**void UpdateChat(char\* )**

클라이언트에서 보낸 채팅 내용을 다른 클라이언트 들에게 보낸다.

## 2. Client-Base

**DWORD WINAPI AccessServer(LPVOID arg) Client->Server**

서버-클라이언트에서 클라이언트가 서버에 요청을 Send 한 뒤 그에 대한 결과를 Recv 하는 함수

**void SignUp(char \*id, int \*pw)**

아이디와 비밀번호를 이용하여 서버에 회원 가입을 요청한다.

**void SignIn(char \*id, int \*pw)**

아이디와 비밀번호를 이용하여 서버에 로그인을 시도한다.

**void SignOut(char \*id)**

서버에 로그아웃을 알리고 서버와 연결을 종료한다.

**void SetUserData(struct& UserData, struct& lobbyInfo)**

받아온 유저데이터(승, 패 등)와 로비정보(방 개수와 상태, 채팅버퍼)를 적용한다.

**void DemandCreateRoom(void)**

방 만들기를 서버에 요청한다. 성공 하면 Room 씬으로 이동하며 Host 역할을 수행한다.

**void SelectRoom(int roomIndex)**

[Critical Section] 해당 번호의 방에 접속 요청한다.(Host 가 아닌 Server 가 대상)

**SOCKADDR& PermitJoinRoom(int roomIndex)**

SelectRoom 이 성공한 경우 해당 방에 접속한다.

**void InputChat(char\*)**

[Critical Section] 플레이어가 입력한 채팅 내용을 서버에 전송한다.

**void DemandRefreshLobby(void)**

서버에 로비 정보(방 개수, 정보 등)을 요청한다.

**void DemandConnectServer(void)**

[Critical Section] 서버의 IP 정보를 입력 받고 그 IP 에 접속 요청한다.

**void DisconnectServer(void)**

. 서버와의 연결을 종료한다

### 3. Client-Host

**DWORD WINAPI RecvData(LPVOID arg)**

recv 전용 함수, recv 처리를 담당한다.

**DWORD WINAPI SendData(LPVOID arg)**

send 전용함수 send 필요시 플래그 설정하고 데이터를 전송한다.

**void StartGame(void)**

각 게스트들에게 게임이 시작되었다고 알린다.

**void SendResult(result)**

서버에 게임의 결과(승, 패)를 전송한다. 0,1 은 승리, 2,3 인덱스 인원은 패배처리

**void EndGame(void)**

게임이 종료된 것을 각 게스트에게 알린다. 이때 승,패 정보도 보낸다.

**void ChangeMap(void)**

호스트에서 맵을 변경한 경우 그 정보를 각 게스트에 알린다.

**void PermitChangeCharacter(enum class character)**

캐릭터 변경을 허락하고 해당 변경사항을 다른 게스트들에게 알린다.

**void RefreshRoom(roomInfo)**

서버(호스트)가 게스트에게 최신화된 방의 정보를 보내준다.

## 4. Client-Guest

**void DemandChanageCharacter(void)**

캐릭터 변경을 요청한다.

**void UpdateCharacter(void)**

캐릭터를 업데이트 한다.(선택한 것을 허용한 경우 바꾸고 아니면 그냥 실패하고 기존 캐릭터를 이용하도록 한다.)

**void UpdateMap(int mapNumer)**

해당 mapNumber 으로 현재 활성화된 맵을 변경한다.

**void UpdateRoom(roomInfo\* info)**

방 정보를 호스트에서 보낸 정보로 업데이트 한다.

**void StartGame(void)**

호스트에서 StartGame 으로 신호를 보낸 경우 게스트에서 게임 시작 씬으로 이동한다.

**void EndGame(roomInfo\* info)**

호스트에서 방 정보를 얻어 다시 방 씬으로 이동한다.

# 팀원 별 역할분담

## ■ 책임 파트

Server	원성연
Client-Base	김나단, 강태규
Host	김나단
Guest	강태규

## ■ 역할

Server	원성연
Client-Base (Title, Login)	김나단
Client-Base (Lobby)	강태규
Client-Host	김나단
Client-Guest	강태규
App 개발	원성연
테스트용 프로그램 개발	김나단, 강태규, 원성연
문서 작성/편집	원성연
Project Manager	김나단
Version Control Manager	강태규
Resource 제작	원성연



# 개발 환경

## ■ 하드웨어

개발 시연	팀원들의 개인 노트북 3 개 개발용 노트북 3 개, 김병진, 이상기, 석진호의 노트북 3 개, 총 6 개
----------	---

## ■ 소프트웨어

운영 체제	Microsoft Windows 10
IDE	Visual Studio 2017 Community
리소스	Photoshop CS6, Fuse CC, Mixamo
VCS	Git (Github)
PMS	Trello
문서작성	Word 2016
커뮤니케이션	KakaoTalk

## ■ 시간 & 장소

정기 모임 시간	매주 화요일 19:30 ~ 21:30
정기 모임 장소	E-room
일일 단위 보고	다음날 12:00 시까지

## ■ 애플리케이션

프로그래밍 언어	C++
타겟 플랫폼	Windows 10 (WinAPI)

	강태규	김나단	원성연
13	Client-Guest 개발 요구 사항 확인 애플리케이션 코드 리뷰	Client-Host 개발 요구 사항 확인 애플리케이션 코드 리뷰	애플리케이션 이미지 리소스 제작 (Title, Login, Lobby, GameRoom Scene)
14			
15			
16	Test Host(Recv) 개발	Test Guest(Recv) 개발	애플리케이션 개발
17	Client 통신 구조체 정의 및 Client-Guest 함수 구현	Client 통신 구조체 관련 Client-Host 함수 구현	
18			
19			
20	Guest 전송 테스트 및 디버깅 원격 테스트 확인 및 디버깅	Host 전송 테스트 및 디버깅 원격 테스트 확인 및 디버깅	Test Sever 개발 및 Title/Login 통신 부분 개발
21			
22			
23	테스트 코드 적용	테스트 코드 적용	Lobby 통신 부분 개발, Game Room, Lobby Scene 에 대한 Client-part 호환성 테스트
24	1.1, 1.2 안 구현 및 테스트, 2 안 구현 및 테스트	1.1, 1.2 안 구현 및 테스트, 2 안 구현 및 테스트	
25			Client-part 구현 보조
26			
27	리팩토링 및 최적화작업 (1 안 부적절 시, 2 안 구현)	리팩토링 및 최적화작업 (1 안 부적절 시, 2 안 구현)	Title, Login Scene 통신 부분 개발
28	2 인 인게임 테스트 (else, 2 안 평가 및 최종결정 모두 부적절 시 새 방안 탐색 및 구현)	2 인 인게임 테스트 (else, 2 안 평가 및 최종결정 모두 부적절 시 새 방안 탐색 및 구현)	Server 리팩토링 작업 (else, 2 안 평가 및 최종결정 모두 부적절 시 새방안 탐색)

	강태규	김나단	원성연
29			
30	GameRoom Guest part 구현 및 연동 확인 (else, 통신 관련 방안 및 구현, 결정 종료)	GameRoom Host part 구현 및 연동 확인 (else, 통신 관련 방안 및 구현, 결정 종료)	GameRoom Server part Client 파트 보조 및 연동 (else, 통신 관련 방안 및 구현, 결정 종료)
1			
2			
3			
4	Client-Base(Lobby) 구현	Client-Base(Title, Login) 구현	Client-Base 와 Server 의 연동 테스트 및 디버그
5			
6			
7	Client-Base(Lobby) Client-Guest Code 리팩토링	Client-Base(Title, Login) Client-Host Code 리팩토링	Testcase 작성 및 Server Code 리팩토링
8			
9	Testcase 를 통한 QA 진행 (원격 통신 Test 진행) 및 반영	Testcase 를 통한 QA 진행 (원격 통신 Test 진행) 및 반영	Testcase 를 통한 QA 진행 (원격 통신 Test 진행) 및 반영
10			
11	게임성 테스트 및 반영	게임성 테스트 및 반영	UX Test 및 UI 추가 작업, 최종 문서 작업
12			
13	최종 테스트		
14	제출		