2 0 2 2 – T e a m P r o j e c t

Network

Game

Programing

2 0 2 2 – T e a m P r o j e c t

추 진 계 획 서

게임공학과

2 0 1 8 1 8 0 0 1 4 도 인 혁

2 0 1 8 1 8 2 0 3 8 조 성 택

2 0 2 2 – T e a m P r o j e c t

Contents

001 애플리케이션 기획

002 H I g h – l e v e l 디자인

003 L o w – l e v e l 디자인

004 개발 환경

005 팀원 역할

006 개발 일정

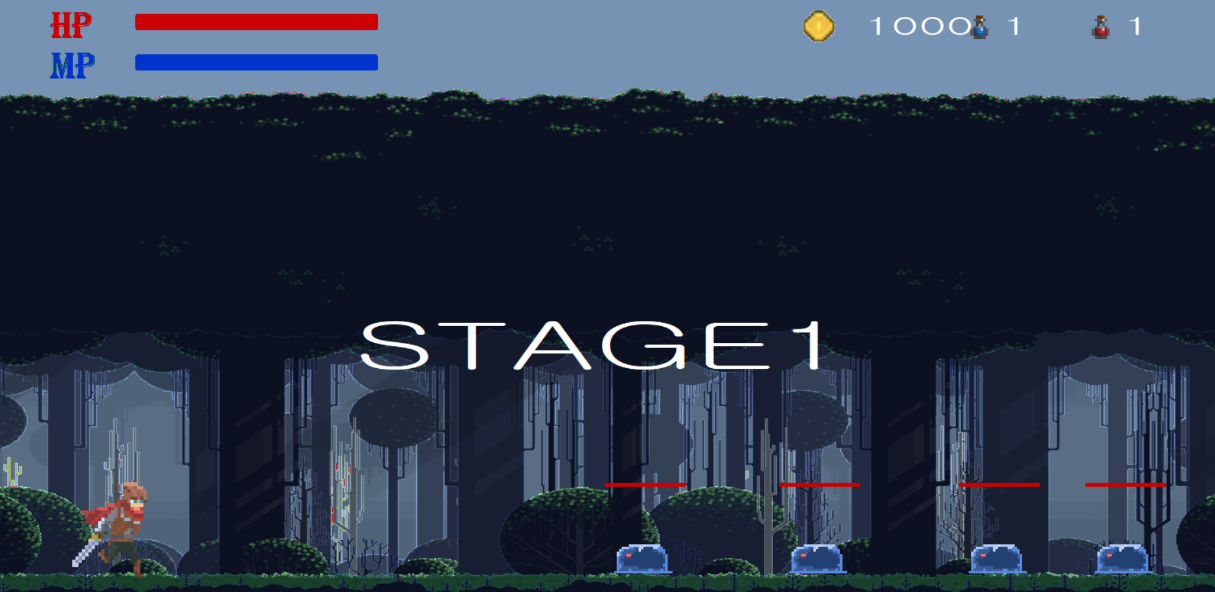
* **애플리케이션 기획**

추진할 게임은 윈도우프로그래밍 텀프로젝트로 현재 팀원인 도인혁, 조성택이 작업한 게임입니다. 기존 게임은 러닝 로그라이크 형식의 게임으로 서버를 추가하여 **1 vs 1** **PVP**(**Player Versus(VS) Player**) 형식의 게임 방식을 도입할 계획입니다.

1. 기존 게임 설명



<마을 씬>

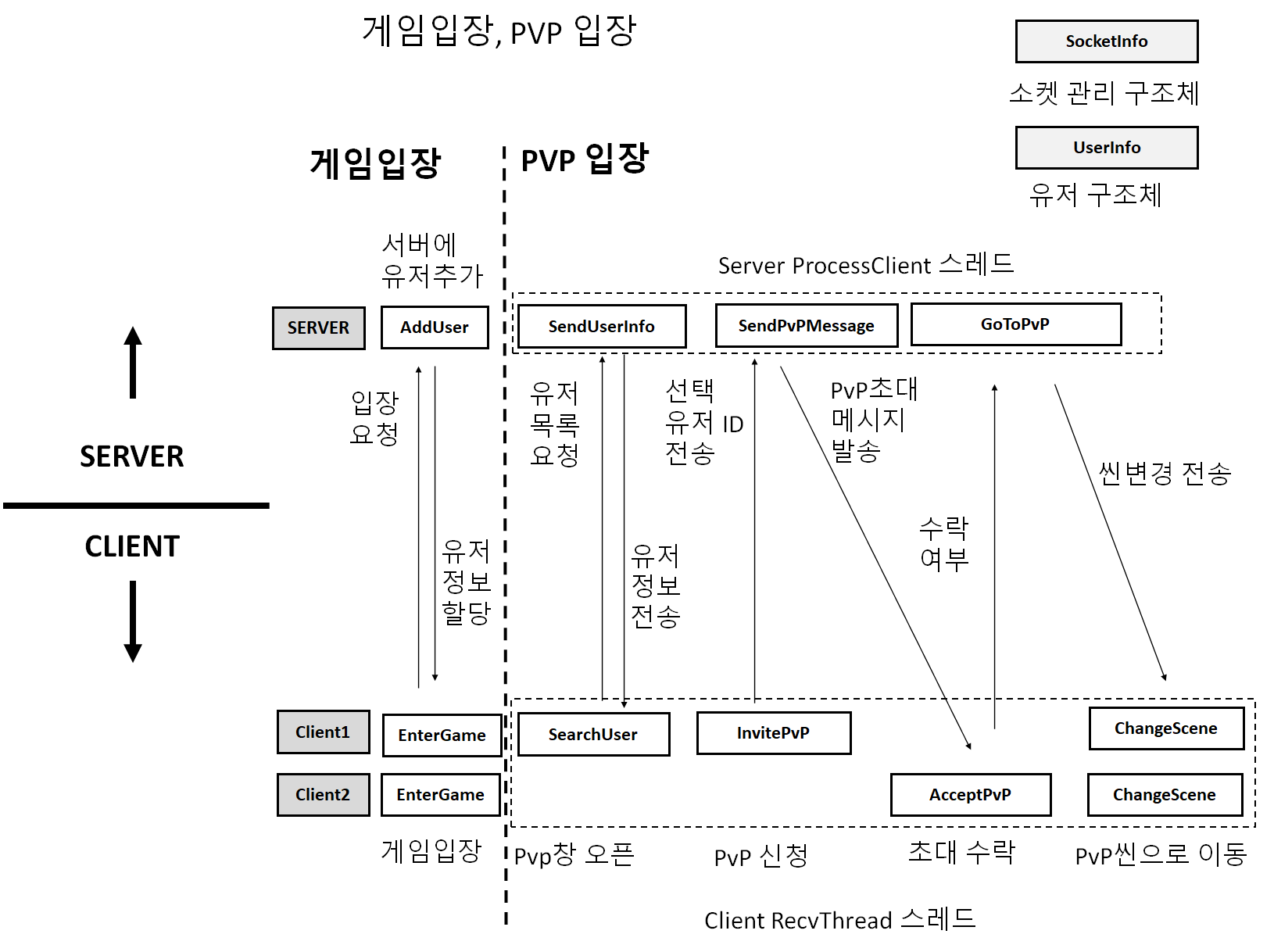


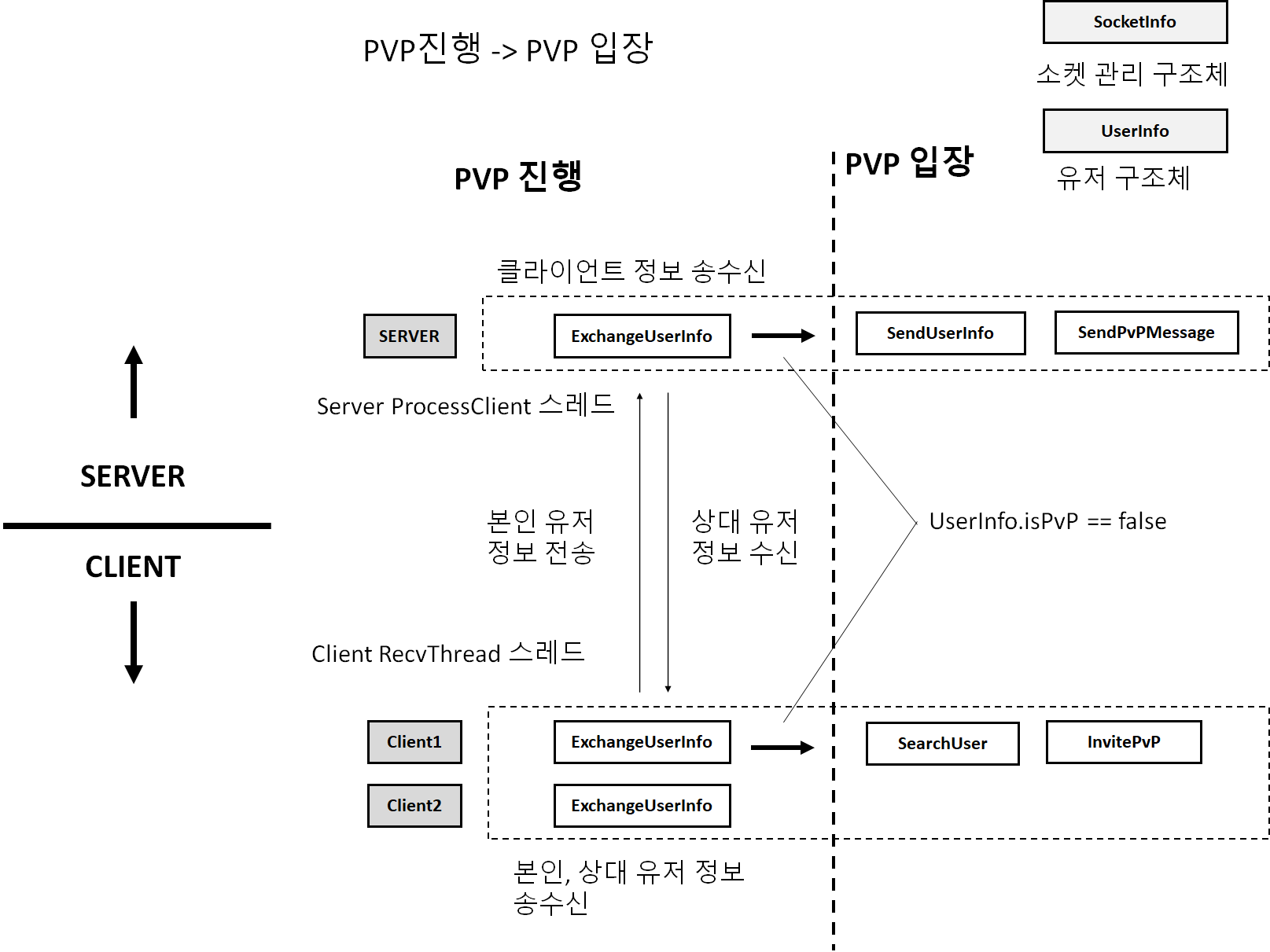
<던전 씬>

러닝 로크라이크 게임으로 마을과 던전을 반복하여 진행되는 게임입니다. 던전에서 획득한 골드로 포션과 아이템을 구매하여 장착합니다.

1. PVP 방식(+추가)
2. PVP를 신청할 플레이어가 PVP서버에서 싸우고 싶은 플레이어를 골라 신청메시지를 보냅니다.
3. PVP메시지를 받은 플레이어가 메시지창을 수락하면 두 플레이어가 콜로세움으로 입장합니다.
4. 체력과 마나가 모두 찬상태에서 PVP가 진행되며 기존에 골드로 구매했던 아이템을 착용합니다.

* **H I g h - l e v e l 디자인**



****

* **L o w – l e v e l 디자인**

**struct UserInfo**

Int Frame, Int ID, int maxhp, int maxmp, int power, int nowhp, int nowmp, int x, int y, int state, bool isPvP, int ClientTime, int ServerTime, int OtherRTT

플레이어 정보를 가지는 구조체, 서버와 클라이언트가 통신할 유저 정보

**struct SocketInfo**

SOCKET client\_sock, int ID

소켓을 관리하는 구조체로 접속 한 플레이어의 아이디와 일치합니다.

**Server**

Accept를 최대 스레드 범위 안에서 계속해 시도한다.

**List<userinfo> UserList**

현재접속중인 유저의 정보를 저장, 관리하는 전역 리스트

**List<SocketInfo> socketList**

현재 접속 중인 유저의 정보에 맞춰 소켓을 관리하는 전역 리스트

**Int Init\_Server()**

윈속 초기화, 소켓 생성 및 Bind Listen Accept를 시도

**int IDCount**

현재 생성된 스레드 id넘버와 유저 id할당에 쓰이는 전역변수

**bool isUsing**

다른 클라이언트가 Userlist 에 접근하고 있는지 확인하는 bool 변수

**bool Adduser()**

유저의 입장 요청을 받아 각각의 유저에 스레드 1개씩 할당하고 성공여부를 반환한다.

**DWORD WINAPI ProcessClient(LPVOID arg)**;

현재 접속 중인 유저에 할당된 스레드, 생성시에 전역변수 IDCount 값을 가져와 스레드ID와 유저 ID를 할당한다. 초기화 한 UserInfo 값을 클라이언트에 할당한다. 할당 후 UserList에 현재 UserInfo 값을 추가한다.

스레드 생성 시에 받은 client\_sock와 유저 ID를 전역변수인 socketList에 추가한다.

PVP입장 전 클라이언트에게 고정(enum 데이터 식별 값) + 가변(실제 데이터) 데이터 형식으로 데이터를 받는다. 고정 데이터를 통해 다음 실행될 함수 (SendUserInfo, SendPvPMessage, GotoPVP)를 식별하고 가변 데이터 값을 넘겨준다.

지역변수 enemyID를 만들어 PvP진행 시 상대 id를 기억하게 한다.

UserInfo.isPvP를 통해 현재 PvP가 진행중임을 판단하고 이 경우 ExchangeUserInfo 로직만 실행한다.

PVP 중인 경우 1초에 10번, pvp가 진행중이지 않은 경우 1초에 1번 time elapsed를 사용해 통신한다.

**bool SendUserInfo(SOCKET client\_sock)**

클라이언트에게 유저 정보 요청을 받은 경우 실행서버에 저장된 UserList 값을 클라이언트에 전송하고 성공여부를 반환한다.

**bool SendPvPMessage(SOCKET client\_sock, int enemyID)**

클라이언트에게 PvP를 신청할 유저에 대한 UserInfo.id(enemyID) 를 받은 후 UserList와 socketList 에서 해당 id를 찾아 고정(enum 데이터 식별 값) + 가변(PvP를 신청한 유저 정보)를 전송한다. 성공여부를 반환한다.

**bool GotoPVP(SOCKET client\_sock, int myID, bool isYes, int enemyID)**

PvP를 승낙한 클라이언트로 부터 승낙 여부와 상대 ID를 넘겨 받은 후 현재 id(myID)의 UserList에 있는 userInfo.isPvP값을 true로 변경한다. client\_sock에 고정(enum 데이터 식별 값) + 가변(bool 승낙) 데이터 전송

enemyID를 기반으로 해당하는 socket을 가져와 고정(enum 데이터 식별 값) + 가변(bool 승낙) 데이터 전송, enemyID의 UserList에 있는 userinfo.isPvP 값 true로 변경

**bool ExchangeUserInfo (SOCKET client\_sock, UserInfo myInfo, enemyID)**

본인의 UserInfo 값을 받아와 UserList에서 해당 id의 UserInfo 값을 초기화 시킨다. client\_sock에게 ProcessClient 스레드의 지역변수인 enemyID(상대 id)를 받아와 UserList에서 해당 id의 UserInfo 값을 전송한다. 전달받은 UserInfo.isPvP값이 false인 경우 pvp가 종료된 것으로 판단한다.

**Client**

**enum Datatype {}**

서버가 데이터를 식별하여 함수를 조정하도록 데이터 타입형태를 enum으로 초기화한다.

**bool EnterGame()**

서버 주소와 포트번호를 통해 서버에 입장요청(connect)을 하여 초기화된 유저 정보를 할당 받는 함수이다. 그리고 UserInfo와 데이터를 받는 스레드를 생성한다.

**DWORD WINAPI RecvThread(LPVOID arg)**

클라이언트 소켓을 초기화 하고 서버에 넘겨준다. 고정(enum 데이터 식별 값) + 가변(실제 데이터) 데이터형식으로 서버에 정보를 요청하고 받습니다.

클라이언트의 요청 및 서버가 전송하는 데이터에 따라 **SearchUser, InvitePVP, AcceptPVP, ChangeScene**를 수행한다.

지역변수 enemyID를 만들어 PvP진행 시 상대 id를 기억하게 한다.

UserInfo값을 받아 해당되는 ID의 UserInfo값을 초기화합니다. 상대ID(enemyID)를 받아 해당되는 정보를 다시 전송합니다.

PVP가 종료되었을 때 UserInfo.isPVP 값을 false로 바꾸어 서버에 전송하여 PVP 종료여부를 알린다.

**bool SearchUser(SOCKET client\_sock)**

서버에 입장 되어 있는 유저목록을 요청하여 유저목록을 받습니다. 유저목록이 있는지 없는지에 대한 여부를 반환한다.

**bool InvitePVP(SOCKET client\_sock,UserInfo enemyInfo)**

클라이언트에서 선택한 유저의 UserInfo.id를 서버에 전송합니다. 상대의 승낙 여부를 반환한다.

**bool AcceptPVP(SOCKET client\_sock)**

SendPvPMessage()를 받은 클라이언트에서 승낙 여부와 상대의 ID를 서버의 넘겨 성공여부를 반환한다.

**bool ChangeScene(SOCKET client\_sock)**

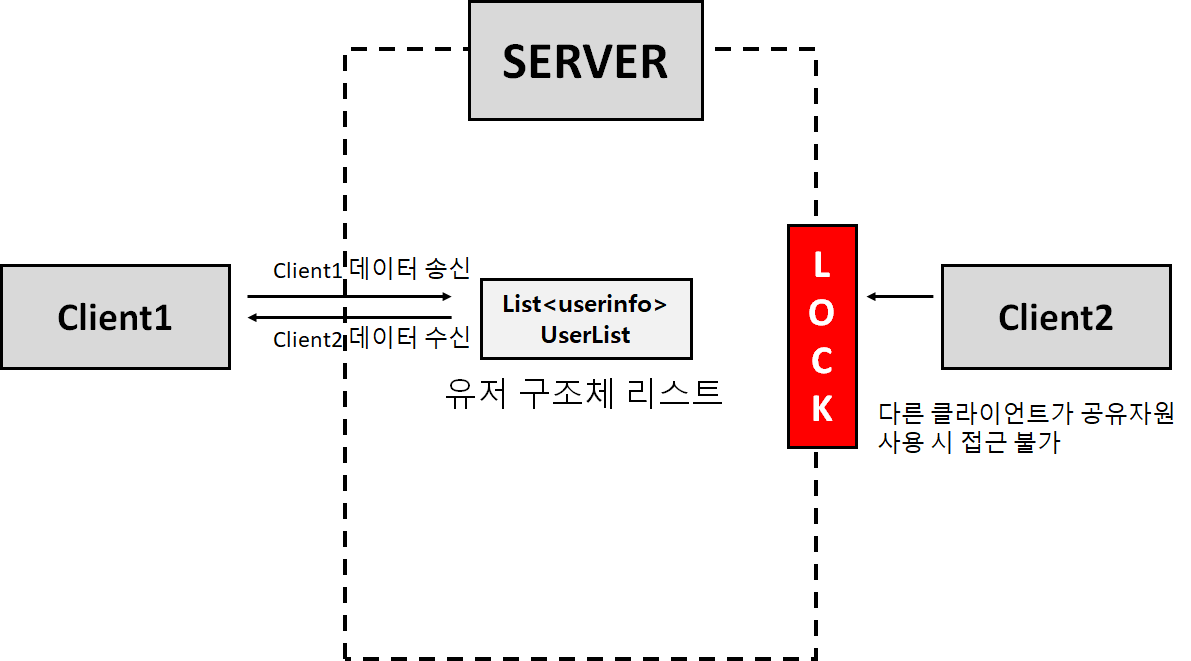
고정(enum 데이터 식별 값) + 가변(bool 승낙) 데이터 값을 받아 승낙의 여부에 따라 클라이언트의 씬을 변경한다.

**bool ExchangeUserInfo (SOCKET client\_sock, UserInfo Info, enemyID)**

상대의 enemyID 값을 받아와 UserList에서 해당 id의 UserInfo 값을 초기화 하고, 나의 UserInfo를 서버에 전송하여 초기화시킨다. PVP가 종료되었을때 UserInfo.isPVP 값을 false로 바뀐다.

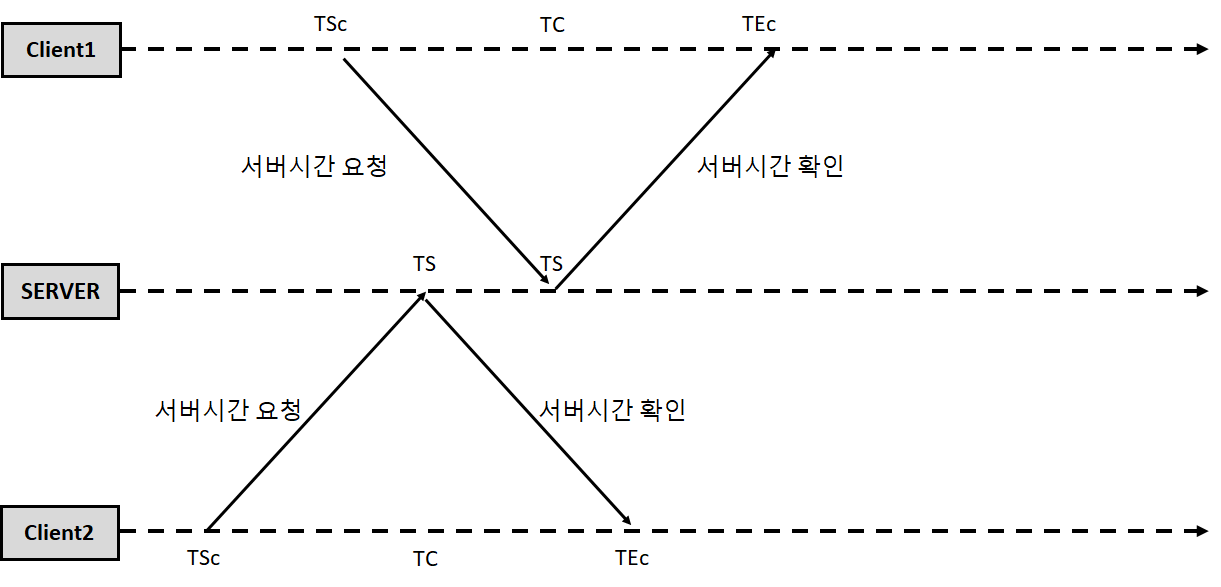
UserInfo.isPvP 가 false가 되면 서버에서는 PvP가 종료된 것으로 인식한다.

**스레드 동기화**



Server에서 관리하는 접속중인 유저 구조체 리스트 UserList에 대해 다른 클라이언트가 접속 시 접근을 방지합니다.

실시간 전투를 통해 상호작용과정에서 Delay를 보간 해주기 위해 서버와 여러 클라이언트의 진행하는 시간정보를 맞추어 동일하게 인지하게끔 하는 작업이 필요하고 클라이언트들의 위치를 일정 프레임 마다 임계영역을 통해 UserInfo를 전송합니다.

****

|  |  |
| --- | --- |
| **TSc** | Client가 Server에 시간을 요청한 시간 |
| **TS** | Server가 Client에게 요청받은 시간 |
| **TC** | TS일 때 Client의 현재 시간 |
| **TEc** | Client가 Server시간을 확인한 시간 |

위 그림과 같이 TSc에서 서버에게 시간을 요청하고 Ts를 두 클라이언트와 공유합니다.

Tc와 Ts를 비교하여 서버의 시간을 항상 체크합니다. 문제는 네트워크의 전송지연으로 인해 Tc의 정보를 정확히 알기 힘듭니다. 그래서 이를 고려해 클라이언트에서 네트워크 지연 시간과 위치값을 예측하는 방식을 수행합니다.

* **개발환경**

통합개발환경(IDE): Visual Studio 2022

협업, 버전관리 툴: Github

라이브러리: 윈도우API

* **팀원 역할 & 개발 일정**

도인혁: 기존 게임 수정 & 추가 구현, PVP 신청과 PVP 입장 구현

조성택: 기존 게임 수정 & 추가 구현, PVP 구현

**11월**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 구분 | 일 | 월 | 화 | 수 | 목 | 금 | 토 |
| 날짜 |  |  | 1 | 2 | 3 | 4 | 5 |
| **도인혁** |  | | | 추진계획서제출 | 기존 게임 수정 계획 | 추진계획서 수정 |  |
| **조성택** | 추진계획서 제출 | 기존 게임 수정 계획 | 추진계획서 수정 |  |
| 날짜 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **도인혁** | 추진계획서 수정 | 추진계획서 수정 | 추진계획서제출 | 플레이어 State 수정(2단 점프) | 기타, 클래스, 변수 수정 및 최적화 | PVP 신청, 메시지 창 등 UI 구현 |  |
| **조성택** | 추진계획서 수정 | 추진계획서 수정 | 추진계획서제출 | 마을 인벤토리렌더 수정 | 기타, 클래스, 변수 수정 및 최적화 | PVP 씬 추가 및 렌더 추가 |  |
| 날짜 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| **도인혁** | 서버 생성, 함수 선언, 기본 틀 추가 | 서버-addUser, 클라이언트-EnterGame 추가, 로컬 확인 | 클라-SearchUser  서버-SendUserInfo추가 | 클라-버튼에 연동 유저검색 후 11일 구현 내용 로컬 확인 | 클라-13일 받아온 데이터 출력 후, 선택 구현 | 클라-invitePvP추가 후 서버에서 확인  서버 – 클라이언트에서 보낸 상대id CMD 출력 | 서버-SendPvP Message를추가하여 다른 클라이언트 유저에게 데이터 전송 |
| **조성택** | PvPEnemy 클래스 추가 | 클라이언트 스레드 추가 | 스레드에서PvP씬 적 데이터 적용 | 스레드에서PvP씬 적 데이터 적용(2) | 서버, 클라 더미 PVP적 데이터 로컬 전송테스트 | Time elapsed를 사용해 통신을 확인 | 각 클라이언트 기본이동, 점프 로컬 확인 |
| 날짜 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| **도인혁** | 클라-AcceptPVP를 추가하고 버튼에 연동 및 함 수간 데이터 전송확인 | 서버-GotoPVP를 추가 승낙여부를 전송하고 로컬확인 | 클라-ChangeScene추가 씬변경 확인 | 주간테스트 | 서버 동기화 작업 | 서버 동기화 작업 |  |
| **조성택** | 유저간 스킬에의한 공격, 공격당함 로컬 확인 | PVP씬 승리, 패배구현 | PVPend 씬 추가 및 렌더 | 주간테스트 | 서버 동기화 작업 | 서버 동기화 작업 |  |
| 날짜 | 27 | 28 | 29 | 30 |  |  |  |
| **도인혁** | 디버그 및 추가 구현 | 디버그 및 추가 구현 | 구현완료 | 주간테스트 |  | | |
| **조성택** | 디버그 및 추가 구현 | 디버그 및 추가 구현 | 구현완료 | 주간테스트 |

**12월**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **구분** | **일** | **월** | **화** | **수** | **목** | **금** | **토** |
| 날짜 | 11/27 | 11/28 | 11/29 | 11/30 | 1 | 2 | 3 |
| **도인혁** |  | | | | 발표 자료 준비 | 발표 자료 준비 |  |
| **조성택** | 발표 자료 준비 | 발표 자료 준비 |  |
| 날짜 | 4 | 5 | 6 |  | | | |
| **도인혁** | 최종테스트 | 최종테스트 | 최종발표 |
| **조성택** | 최종테스트 | 최종테스트 | 최종발표 |