네트워크 게임 프로그래밍
Term Project 추진계획서 V2

7조

2019184030 조성원

2022180005 김상혁

2020180040 진현서

목 차

1. 애플리케이션 기획

•게임 소개

•- 게임 이름 : Unity Of Mind

•- 게임 장르 : 아케이드 게임 , 협동 게임

•- 게임 플레이 방법 : 앞으로 다가오는 블럭을 피해서 끝까지 살아남는 협동 게임

•- 플레이어 수 : 3명

조성원 팀원이 컴퓨터 그래픽스 과목에서 과제로 제출
한 프로젝트를 기반으로 제작하였습니다.

플레이어들은 앞으로 다가오는 블럭을 피해야 함. 블럭
에 맞으면 체력이 감소 (1/3 감소) . 모든 플레이어가 체력
을 공유함, 체력이 0이 되면 게임 오버, 모든 스테이지
를 통과하면 게임 클리어

1. 애플리케이션 기획

조작법

- Left 키 , Right 키 : 각 방향에 맞는 이동
- z 키 : 플레이어 색을 빨간색으로 변경
- x 키 : 플레이어 색을 초록색으로 변경
- c 키 : 플레이어 색을 파란색으로 변경
- v 키 : 플레이어 크기 축소 / 확대
- 1 / 3 키 : 카메라 모드 변환

스테이지 별 컨셉

1 스테이지 : 블럭이 없는 공간으로 이동해 블럭을 통과

2 스테이지 : 블럭의 색과 캐릭터의 색을 일치시켜 블럭을 통과

3 스테이지 : 캐릭터의 크기를 축소시켜 빈 공간으로 블럭을 통과

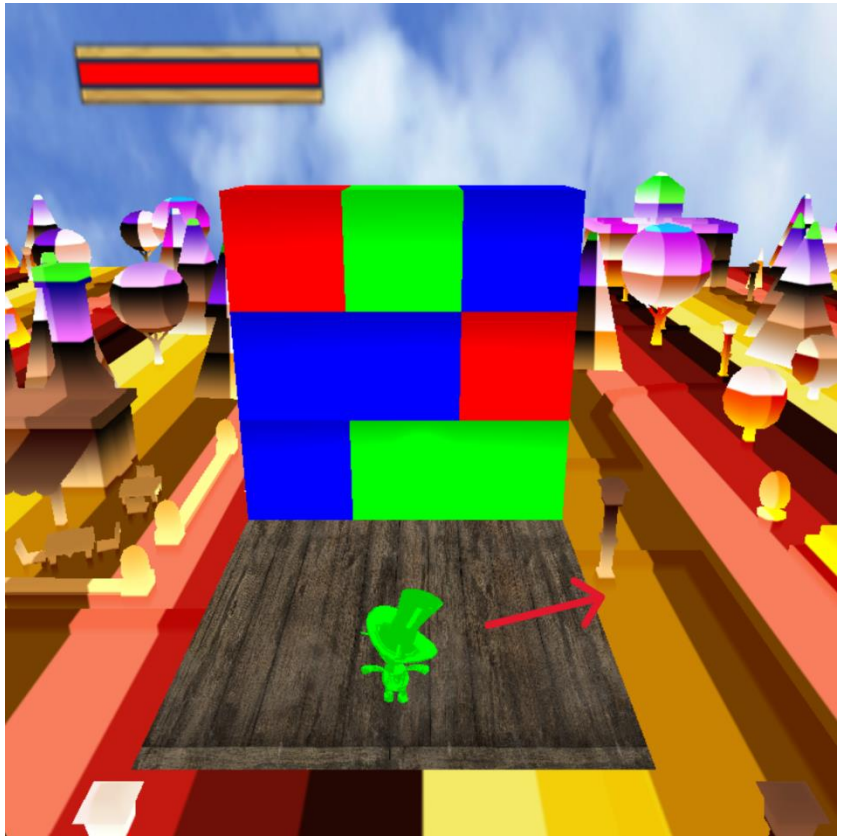
로비화면

1. 애플리케이션 기획

1스테이지

- 블럭이 스테이지 마다 10번 씩 앞으로 앞으로 다가옴
- 블럭이 다가오면 플레이어는 빈공간으로 이동해서 피해야 함
- 블럭에 맞으면 HP 감소

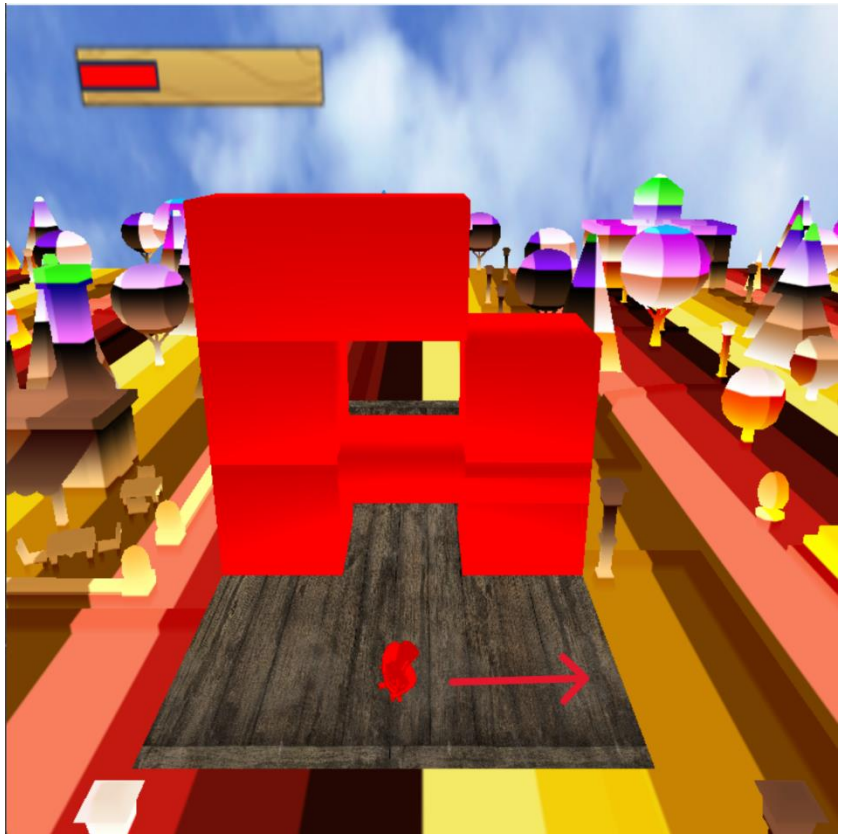
1. 애플리케이션 기획



2스테이지

- 플레이어는 블럭에 보이는 색과 똑같은 색으로 캐릭터를 바꾸어서 블럭을 통과해야 함
- 다른 색으로 통과 시 HP 감소

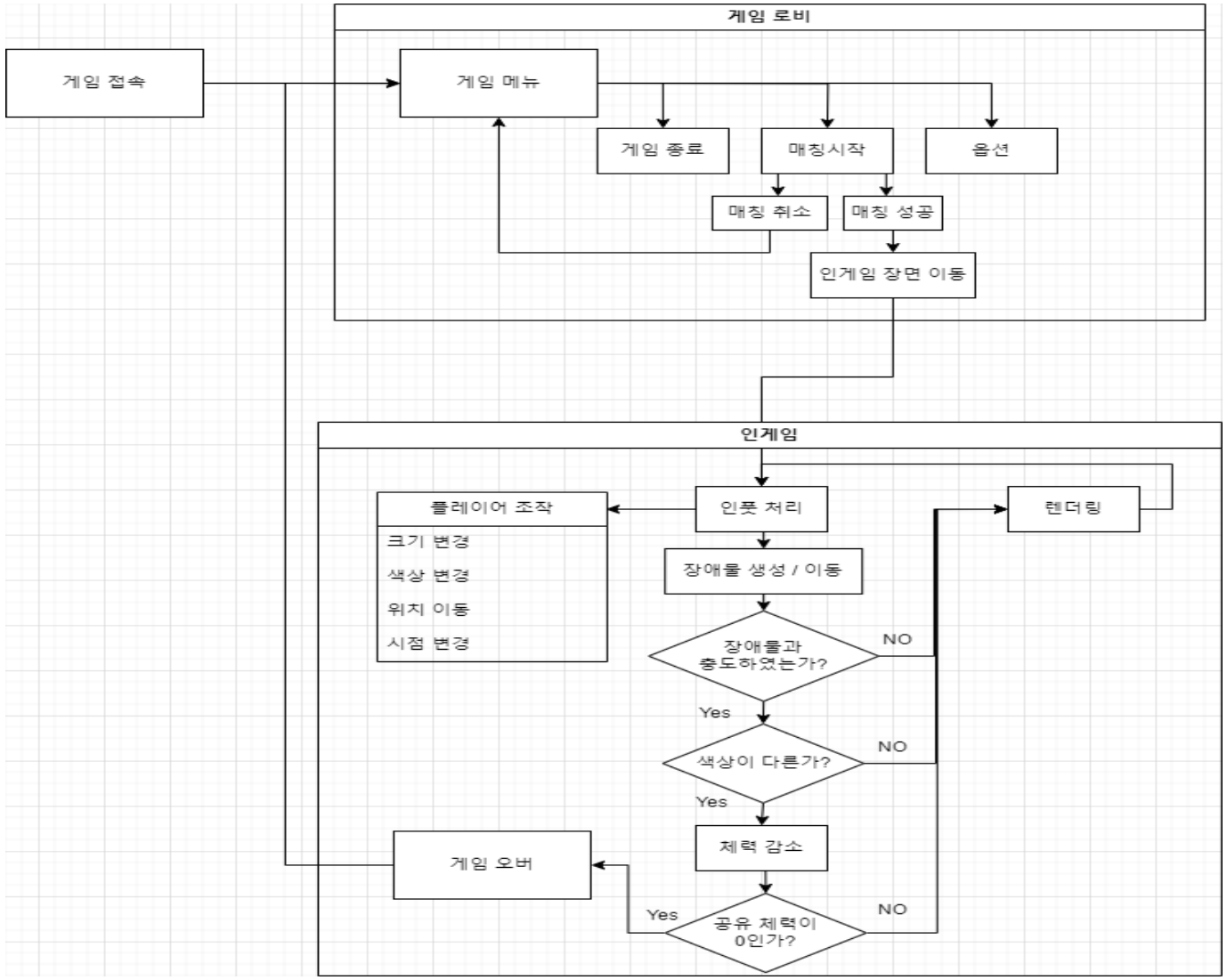
1. 애플리케이션 기획



- 플레이어는 캐릭터의 크기를 줄여서 블럭을 통과해야 함

3스테이지

게임 플로우 및 개발 환경

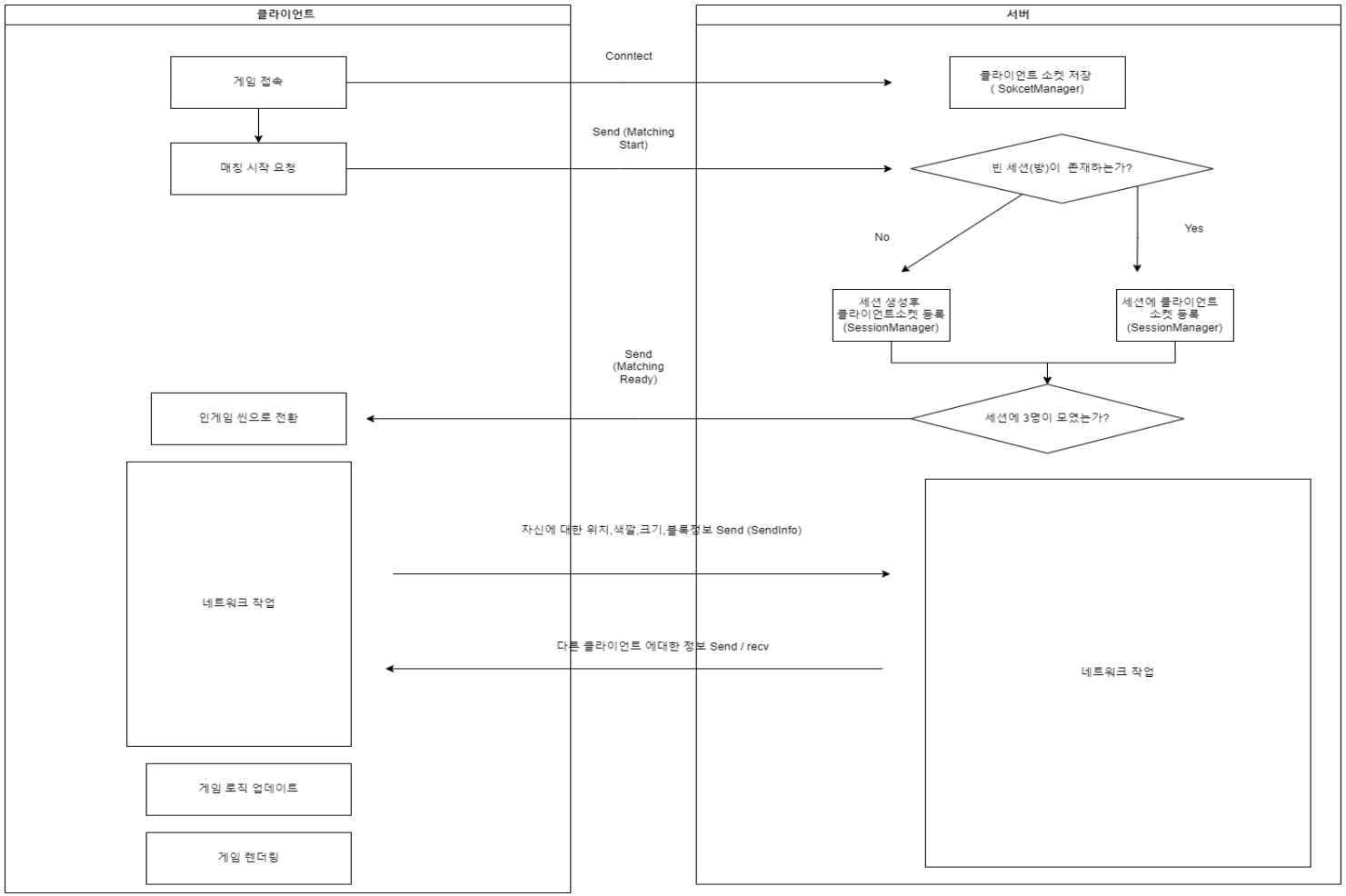


개발 환경

- IDE : Visual Studio
- API : OpenGL
- 사용 언어 : C , C++

2. High Level Design

Server / Client 데이터 송수신 플로우



2. High Level Design

Packet 처리 프로토콜 => Send 두번

(MYCMD 전송 + 실질적인 데이터) – 서버 ,클라이언트 모두 적용

```
//어떠한 데이터를 보낼것인지.
enum CMDCODE
{
    Connect =0, //채널에 입장하자 ClientId 부여받음
    ClientInfoData , // 클라이언트 위치,색깔,크기 정보
    BlockData, //블록에 대한 정보
    ChattingData, // 채팅에 대한정보
    MatcingStartReady, //클라이언트에서 Matching 준비완료가 됐다는 신호 / 서버에서 3명이 되서 Matching 이 준비가 완료됐다는 신호
    MatchingCancle // 클라이언트에서 Matching 을 취소하겠다는 신호를 서버에게 보냄.

    //필요한것을 추가할예정
};

struct MYCMD
{
    int Code=0;         //명령코드
    int Size=0;         //질실적인 내용부의 데이터 크기
    int ClientID = 0;
};
```

2. High Level Design

- Server Architecture

메인쓰레드 : ListenSocket 생성 => Accpet 담당

쓰레드1 :Recv/Send 담당 – 클라이언트1 - PakcetDecode() 함수 후 개별적인 Send함수 호출

쓰레드2 :Recv/Send 담당 – 클라이언트2 - PakcetDecode() 함수 후 개별적인 Send함수 호출

쓰레드3 :Recv/Send 담당 – 클라이언트3 - PakcetDecode() 함수 후 개별적인 Send함수 호출

쓰레드갯수는 클라이언트 개수에 비례

- Client Architecture

메인쓰레드 : 클라이언트 로직(렌더링+업데이트)

쓰레드1 :Send 담당 – Send() 함수

쓰레드2 :Recv 담당 – PakcetDecode() 함수

쓰레드2에서 Recv 를 받은후 메인쓰레드 가 실행되게 할수있도록 (동기화작업)
Event객체를 사용할 예정

3. Low Level Design

-Server

```
class ServerManager
{
public:

    void    PrintClinetInfo(SOCKET socket, string Message);
    void    PushClient(SOCKET socket);
    void    DeleteClient(SOCKET socket);
    int     GetClinetCount( );
public:
    //실질적인 함수구현부
    void PacketDecode(SOCKET socket); //Packet의 정보를 확인하여 실질적인 함수부를 실행해줌.
    void ConnectClient(SOCKET socket); //요청을 받아 ID 부여해줌
    void MatchingAccept(SOCKET socket); // Matching 요청 받아서 처리해줌.
    void MathcingOff(SOCKET socket); //MachingCancle 요청 받아서 처리해줌
    void SendMessageToAllclinet(SOCKET socket,int size); //채팅
    /* 아래는 구현해야함*/
    void BlockCollision( ); //피검사를 해서 GameOver 시키거나 hp를 깍음.
    void MakeBlockSend(SOCKET socket); //블록생성해서 보내줌
    void PlayerInfo(SOCKET socket); //플레이어의 정보를 받아서 브로드캐스팅방식으로 쏴줌.
    void Restart(SOCKET socket);  // 재시작
    void ReturnMenu(SOCKET socket); //연결을 끊고 다시 방매칭 할수있게해줌
private:
    mutex _mutex;
    concurrent_queue<SOCKET> _listClient;
    atomic<int> _readyCount = 0;
    atomic<int> IDGenator =1;
    atomic<int> _hp = 3;
};
```

클라이언트를 관리하기위한 클래스

3. Low Level Design

-Server

```
void ServerManager::PacketDecode(SOCKET socket)
{
    MYCMD cmd;

    while (::recv(socket, (char*)&cmd, sizeof(MYCMD), MSG_WAITALL) > 0)
    {
        switch (cmd.Code) // 명령부의 해석에 따라
        {
        case Connect:
            ConnectClient(socket);
            break;
        case MatcingStartReady:
            MatchingAccept(socket);
            break;
        case MatchingCancle:
            MathcingOff(socket);
            break;
        case ClientInfoData:
            break;
        case ChattingData:
            SendMessageToAllclinet(socket,cmd.Size);
            break;
        default:
            ErrorHandler("알수없는 명령어 수신했습니다.");
            break;
        }

    }
}
```

클라이언트의 요청을 recv 받고 send 해주는 코어 함수

3. Low Level Design

-Client

```
class Client
{
public:
    Client() = default;
    ~Client();
    void Init();
    void Send(); //매프레임 메인 쓰레드 에서 필요한 데이터를 서버에게 전송
public:
    void PacketDecode(); //서버로부터 패킷을 받아서 실질적인 해석하여 실질적인 함수부를 구현.
    void SendConnectServer(); //서버에게 접속할때 ID 부여 받을때 메세지보냄
    void RecvMessageFromServer(int size); //채팅
    void SendMessageToAllclinet(); //채팅
    void SendMatchingStart(); //서버에게 Maching 요청
    void SendMatchignCancle();//서버에게 Maching 취소요청
    //플레이어 위치정보
    void PlayerInfo();
    void FindClientPlayer(int clientID);
    void UpdateViewerScale(const glm::vec3& newSacle);
    //아래는 구현해야함
    void BlockCollision();
    void CreateClinetPlayer(int clinetID);
    void RemoveClinetPlayer(int clinetID);
    void DisConnetPlayer(); //연결 종료신호
    void UpdateViewerPos(float newPosX);
    void UpdateViewerColor(vec3 Color);
public:
    int _clientID;
private:
    SOCKET _connectedSocket;
};
```

3. Low Level Design

-Client

```
void Client::PacketDecode( )
{
    MYCMD cmd;

    while (::recv(_connectedSocket, (char*)&cmd, sizeof(MYCMD), MSG_WAITALL) > 0)
    {
        switch (cmd.Code) // 명령부의 해석에 따라
        {
        case Connect:
            _clientID = cmd.ClientID;
            break;
        case MatcingStartReady: //3명이 모임
            std::cout << "매칭 준비가 완료되었습니다!" << "₩n";
            screen.status = 4;
            break;
        case ChattingData:
            RecvMessageFromServer(cmd.Size);
            break;
            break;
        default:
            ErrorHandler("알수없는 명령어 수신했습니다.");
            break;
        }

    }

    puts("수신 스레드가 끝났습니다.");

}
```

서버의 요청을 recv 받아서 로직을 업데이트해주는 코어함수
동기화가 필요한곳에는 Event객체를 사용할 예정

4. 팀원별 역할 분담

조성원

매칭 화면 제작

다른 클라이언트의 동작을 볼 수 있는 뷰어 객체를 만드는 작업

전송할 정보들 구조체로 묶는 작업

다른 클라이언트 유닛 객체 렌더링 분리 작업

클라이언트 간의 위치동기화 작업

작업 Event 객체를 활용한 게임로직 동기화작업

김상혁

클라이언트 소켓을 세션 단위 그룹화 작업

로비 에 대한 로직구현 (start,cancel,end,restart,채팅)

클라이언트에서 텍스트를 생성하여 렌더링 해 주는 코드추가

프레임워크 공용 클래스 설계 및 작업

패킷 프로토콜 설계 및 패킷 코드 작성

HP 조건 검사

진현서

블록생성 작업

마스터 클라이언트 에 대한 로직 분리

블록 충돌 에 대한 처리

HP 체력바 UI 작업

작업 Event 객체를 활용한 게임로직 동기화작업

일정 : 2022180005 김상혁

일	월	화	수	목	금	토
10/27	10/28	10/29	10/30	10/31	11/1	11/2
			공통프레임워크 제작 -PacketProtocol 작업 (MYCMD 구조 설계)	공통프레임워크 제작 ServerManager 작업 -Listenr ,PacketDecode	공통프레임워크 제작 ClientManager 작업 -ConnetcionSocket, PacketDecode 작업	공통프레임워크 제작 -ThreadManager 작업 -MainThread -Thread 단위작업설계
11/3	11/4	11/5	11/5	11/7	11/8	11/9
공통프레임워크 제작 ServerManager 작업 -LobbySession 작업 (SendMessageToAllclient) (ConnectionClient)	공통프레임워크 제작 ClientManager 작업 -LobbySession 작업 (SendMessageToAllclient) (SendConnectServer)	LobbySession 작업 (SendMathcingStart, MathcingAccept,	11/5 작업이어서 공통프레임워크 제작 로비테스트 검증 ->Acccept 검증 완료		LobbySession 작업 SendMatchingCancle, MathcingOff)	각각의 작업 브랜치를 Merge해서 종합빌드 테스트 및 개발 일정 피드백
11/10	11/11	11/12	11/13	11/14	11/15	11/16
				LobbySession 작업 SendMatchingCancle, MathcingOff) ->Cancle 검증 테스트		각각의 작업 브랜치를 Merge해서 종합빌드 테스트 및 개발 일정 피드백
11/17	11/18	11/19	11/20	11/21	11/22	11/23
				서버측에서 블록 충돌 패킷받아서 처리 (HP감소)	HP 감소를 체크하여 GAMEOVER 패킷보냄	각각의 작업 브랜치를 Merge해서 종합빌드 테스트 및 개발 일정 피드백
11/24	11/25	11/26	11/27	11/28	11/29	11/30
LobbySession -GameOver 패킷 에 대해 어떻게 반응할것인지 에대한 처리			LobbySession -Restart() 에 대한처리	LobbySession ReturnMenu() 에 대한처리		각각의 작업 브랜치를 Merge해서 종합빌드 테스트 및 개발 일정 피드백

일정 : 2020180040 진현서

일	월	화	수	목	금	토
					11/1	11/2
11/3	11/4	11/5	11/6	11/7 접속한 클라이언트에 대한 정보를 기반으로 마스터 클라이언트를 선정하는 작업	11/8 마스터 클라이언트 여부를 토대로 클라이언트내 동작 권한 분리작업	11/9 각각의 작업 브랜치를 Marge해서 종합 빌드 테스트 및 개발 일정 피드백
11/10	11/11	11/12	11/13	11/14 일반 클라이언트에서 마스터 클라이언트가 보내준 정보로 동기화 하는 함수 작업	11/15 PacketDecode() 함수에 HP 감소 로직 추가	11/16 각각의 작업 브랜치를 Marge해서 종합 빌드 테스트 및 개발 일정 피드백
11/17 마스터 클라이언트에 Block 생성을 위임하는 작업	11/18 현재 방에 블록을 생성하는 MakeBlock() 함수 해당 정보를 Send로 송신	11/19 블록과 충돌을 검사하여 해당 사실을 서버에게 알리 BlockCollision() 작업	11/20	11/21 서버측에서 충돌을 감지해서 체력을 감소시키는 BlockCollision() 작업	11/22 HP 체력 감소에 대한 이벤트를 서버로부터 수신하는 작업.	11/23 각각의 작업 브랜치를 Marge해서 종합 빌드 테스트 및 개발 일정 피드백
11/24 HP 체력의 감소 정보를 클라이언트 UI에 반영하는 작업	11/25 마스터 클라이언트가 방에서 접속 종료 시 마스터 클라이언트의 위임 작업	11/26 마스터를 위임 받은 클라이언트가 다시 마스터로 동작할 수 있도록 초기화 해주는 작업.	11/27	11/28 각각의 작업 브랜치를 Marge해서 종합 빌드 테스트 및 개발 일정 피드백	11/29	11/30
12/1	12/2	12/3	12/4	12/5	12/6	

일정 : 2019184030 조성원

일	월	화	수	목	금	토
					11/1	11/2
					매칭 UI 제작 매칭 , 매칭취소 클라이언트 작업	
11/3	11/4	11/5	11/6	11/7	11/8	11/9
				플레이어 위치 , 색상 , 크기 정보 구조체 패킷화 작업	뷰어 클라이언트 객체 변수 및 함수 정의 ViewPlayer.h , cpp	각각의 작업 브랜치를 Merge해서 종합 빌드 테스트 및 개발 일정 피드백
11/10	11/11	11/12	11/13	11/14	11/15	11/16
				FindClientPlayer() CreateClientPlayer() RemoveClientPlayer()	ConnectClient(int clientID) DisconnectClient(int clientID)	각각의 작업 브랜치를 Merge해서 종합 빌드 테스트 및 개발 일정 피드백
11/17	11/18	11/19	11/20	11/21	11/22	11/23
플레이어 패킷 서버로 Send() , Recv() 작업		서버에서 받은 패킷을 토대로 다른 클라이언트 위치 동기화 updateViewerPos		서버에서 받은 패킷을 토대로 다른 클라이언트 색상 동기화 updateViewerColor	서버에서 받은 패킷을 토대로 다른 클라이언트 크기 동기화 updateViewerScale	각각의 작업 브랜치를 Merge해서 종합 빌드 테스트 및 개발 일정 피드백
11/24	11/25	11/26	11/27	11/28	11/29	11/30
개발 일정 피드백을 토대로 위치 동기화 문제점 수정				각각의 작업 브랜치를 Merge해서 종합 빌드 테스트 및 개발 일정 피드백		
12/1	12/2	12/3	12/4	12/5	12/6	