네트워크 게임 프로그래밍
Term 프로젝트 추진계획서

학번	이름
2020184043	박자은(팀장)
2021146031	이정범
2020182024	우현정

[목차]

1. 애플리케이션 소개

1.1 게임 소개

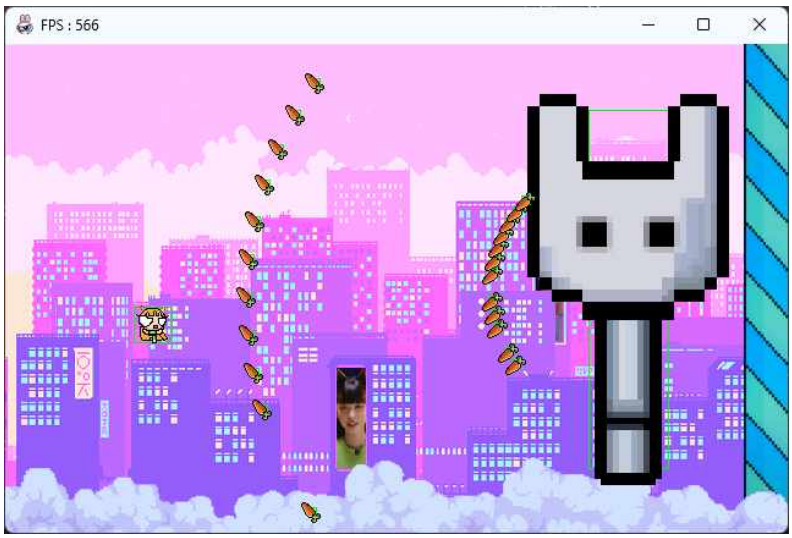
-게임 이름 : 파워퍼프걸

-게임 장르 : 3인칭 슈팅게임

-게임 컨셉 : 3명의 플레이어가 몬스터를 처치하는 게임

-게임 작업자 : 이정범(윈도우게임프로그래밍)

-인게임 사진



1.2 게임 플레이 방법

-조작법

이동 : 방향키

공격 : SPACE

-게임 흐름

(1) 타이틀

게임 타이틀 화면이 나타나고, 이후 로비에 접속한다.

(2) 로비

모두 접속이 되었을 시 각 플레이어의 캐릭터 선택 창이 뜬다. 모든 플레이어의 캐릭터 선택이 완료되면 게임이 시작된다.

(3) 게임 플레이

- 플레이어는 자유롭게 맵을 돌아다닐 수 있다.
- 플레이어는 상하좌우 이동, 스페이스바로 공격을 할 수 있다.
- 몬스터 처치 시, UI에 처치한 몬스터의 수를 확인할 수 있다.
- 획득한 아이템 개수를 UI에서 확인할 수 있다.
- 플레이어들은 3개의 목숨을 다 잃을 경우 사망한다.
- 사망한 플레이어는 제자리에서 부활이 가능하다.

(4) 게임 종료

- 로비버튼을 누르면 로비화면으로 돌아가고 재시작 할 수 있다,
- 나가기 버튼을 누르면 게임이 종료된다. 게임종료 시 서버 접속이 종료된다.

1.3 개발 목표

-최대 3인의 플레이어가 서버에 접속하여 각 플레이어가 캐릭터 선택이 완료되면 게임이 시작되도록 구현한다.

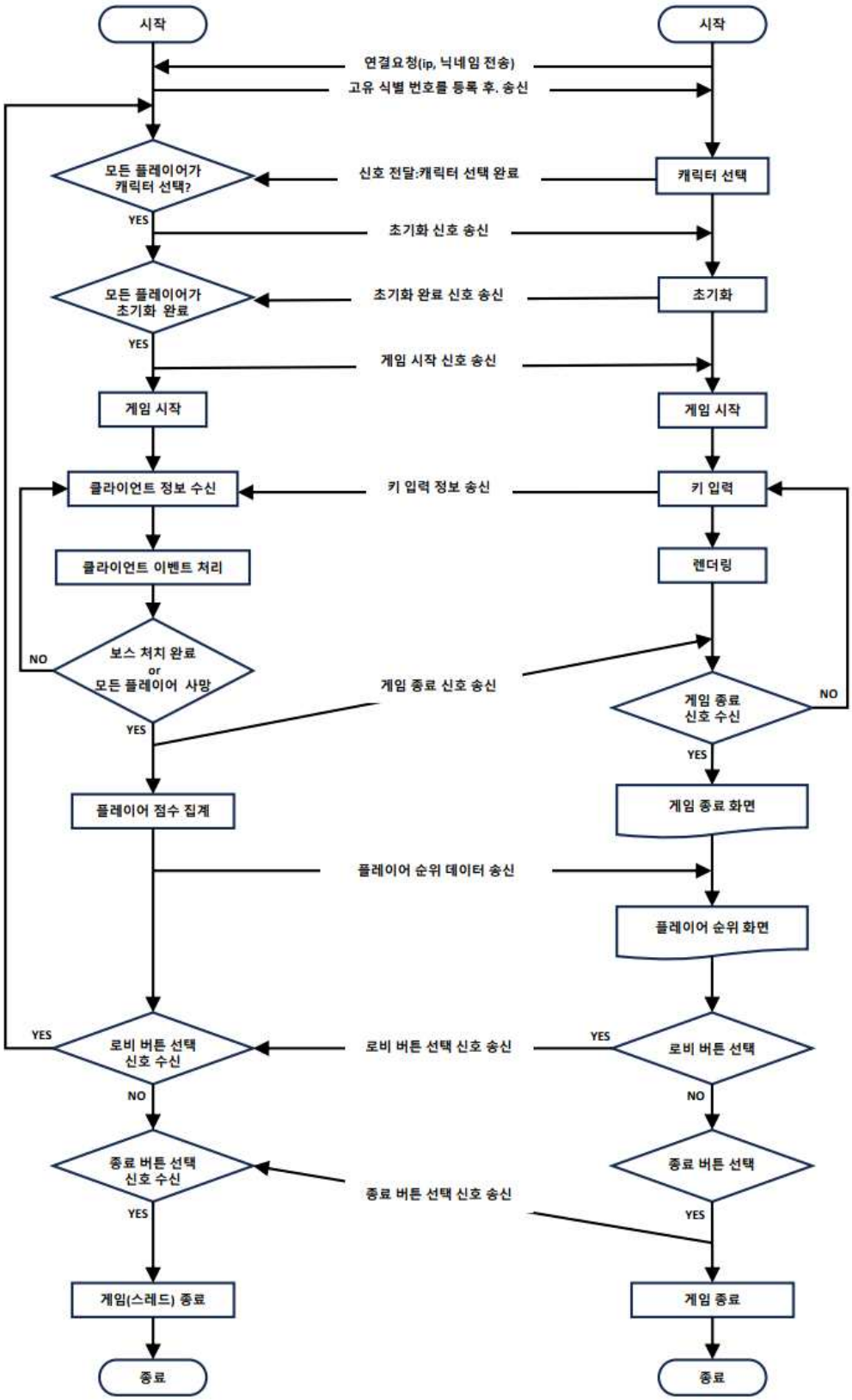
-모든 플레이어 사망 시 또는 보스 처치 시, 게임 종료 화면으로 넘어간다.

-게임 종료 화면에서 로비로 되돌아가는 것과 게임 종료를 선택할 수 있다.

-게임 종료 시, 각 플레이어의 점수(아이템, 보스 처치 점수 합산)를 표시하고 가장 높은 점수 순으로 화면에 표시된다.

2. High-Level Design

2.1 플로우차트



2.2 서버 구현 내용

(1) 서버 실행

- 서버를 시작하면, Winsock을 초기화하고 대기 소켓을 생성한다.
- bind 함수를 사용하여 서버의 지역 IP 주소와 고유 식별 번호를 설정한다.
- listen 함수를 호출하여 클라이언트의 연결을 기다리는 상태로 대기 소켓을 만든다.
- 클라이언트가 connect 함수를 사용하여 서버에 연결하면, 서버는 클라이언트의 연결을 accept하여 새로운 클라이언트를 위한 소켓을 생성한다.
- 이후, 서버는 클라이언트에게 로비 화면으로 전환하라는 메시지 정보를 송신한다.

(2) 클라이언트 데이터 통신

- 서버에 접속한 클라이언트마다 스레드를 할당한다. 이후 통신 소켓을 생성하여 스레드를 통해 클라이언트 ip주소와 닉네임을 수신한다.
- 게임 플레이 중, 스레드를 통해 클라이언트별로 캐릭터의 캐릭터 위치 정보, 캐릭터 상태 정보, 오브젝트(아이템, 총) 정보, 처치한 몬스터의 정보를 변경 사항에 대해 전송받아 처리하여 다시 각 클라이언트에 전송한다.
- 보스 전투 플레이 시, 각 클라이언트로부터 보스에 입힌 데미지 정보를 전송받아 처리하고, 보스의 체력이 0이 되면 모든 클라이언트에게 게임 종료 패킷을 전송한다.
- 게임 종료 패킷을 보냈다면, 각 클라이언트로부터 점수를 전송받아 총 점수를 계산하고 순위를 정리하여 모든 클라이언트에 전송한다.
- 게임 종료 시, 클라이언트에게 로비 패킷을 받으면 해당 클라이언트에게 로비 화면으로 이동하는 패킷을 전송한다. 클라이언트에게 게임 종료 패킷을 받으면 해당 클라이언트의 스레드를 종료하고 클라이언트 소켓을 닫아 클라이언트와의 연결을 종료한다.

2.3 클라이언트 구현 내용

(1) 타이틀

- 서버의 ip주소와 닉네임을 입력한다.
- Winsock 초기화 후, connect 함수를 이용해 서버에 연결을 요청한다.
- 로비에 접속한다.

(2) 로비

- 서버에 접속한 플레이어의 닉네임이 화면의 우측에 출력된다.
- 좌측의 플레이어의 캐릭터 선택 창에서 캐릭터를 선택하고 모든 플레이어의 캐릭터 선택이 완료되면 게임 시작 버튼이 활성화된다.
- 1P가 게임 시작 버튼을 누르면 게임이 시작된다.

(3) 게임 플레이

- 서버로 자신의 캐릭터 위치 정보, 캐릭터 상태 정보, 오브젝트(아이템, 총) 정보, 처치한 몬스터의 정보를 실시간으로 전송한다.
- 서버에게 다른 클라이언트들의 캐릭터 위치 정보, 캐릭터 상태 정보, 오브젝트(아이템, 총) 정보, 처치한 몬스터의 정보를 실시간으로 전송받아 자신의 화면에 적용한다.

- 보스 처치 시, 게임 종료 화면으로 이동한다.

(4) 게임 종료

- 서버에서 모든 플레이어의 점수를 받아 가장 높은 점수 순으로 화면에 출력한다.
- 로비버튼을 누르면 로비 화면으로 이동한다.
- 게임종료버튼을 누르면 게임이 종료되고 서버 접속이 종료된다.

3. Low-Level Design

3.1 데이터를 전송할 때 사용할 패킷

- 서버 패킷 타입

번호	패킷 타입	설명
1	SC_MAKE_ID	ID 할당 관련 패킷 타입
2	SELECT_CHARACTER	캐릭터 선택 관련 패킷 타입
3	SC_INIT	초기화 신호 관련 패킷 타입
4	SC_GAME_START	게임 시작 신호 관련 패킷 타입
5	SC_GAME_OVER	게임 오버 신호 관련 패킷 타입
6	SC_GAME_CLEAR	게임 클리어 신호 관련 패킷 타입
7	SC_PLAYER	플레이어 정보 관련 패킷 타입
8	SC_MONSTER	몬스터 정보 관련 패킷 타입
9	SC_BOSS	보스 정보 관련 패킷 타입
10	SC_BULLET	투사체 정보 관련 패킷 타입
11	SC_ITEM	아이템 정보 관련 패킷 타입
12	SC_RANK	게임 클리어 후, 플레이어들의 순위 정보 관련 패킷 타입

- 서버 패킷 설명

번호	패킷	설명	패킷 타입

1	typedef struct SELECT_CHARACTER_PACKET{
	char type; // 패킷 타입
	int id; // id
	CHARACTER_TYPE character; // 캐릭터
	}
		캐릭터 선택
		관련 패킷
			SELECT_CHARACTER

2	typedef struct SC_MAKE_ID_PACKET {
	char type; // 패킷 타입
	int id; // 할당할 id(1p, 2p, 3p)
	}
		ID 할당 관련 패킷	SC_MAKE_ID

3	typedef struct SC_INIT_PACKET { char type; // 패킷 타입	초기화 신호 수신 관련 패킷	SC_INIT

	}		

4	typedef struct SC_GAME_START_PACKET {
	char type; // 패킷 타입
	}
		게임 시작 신호
		수신 관련 패킷
			SC_GAME_START

5	typedef struct SC_GAME_OVER_PACKET {
	char type; // 패킷 타입
	}
		게임 오버 신호
		수신 관련 패킷
			SC_GAME_OVER

6	typedef struct SC_GAME_CLEAR_PACKET {
	char type; // 패킷 타입
	}
		게임 클리어 신호
		관련 패킷
			SC_GAME_CLEAR

7	typedef struct SC_PLAYER_PACKET{
	char type; // 패킷 타입
	CPlayer player; // 플레이어 정보
	}
		플레이어 정보
		수신 관련 패킷
			SC_PLAYER

8	typedef struct SC_MONSTER_PACKET{
	char type; // 패킷 타입
	CMonster monster; // 몬스터 정보
	}
		몬스터 정보 수신
		관련 패킷
			SC_MONSTER

9	typedef struct SC_BOSS_PACKET{
	char type; // 패킷 타입
	CBoss boss; // 보스 정보
	}
		보스 정보 수신
		관련 패킷
			SC_BOSS

10	typedef struct SC_BULLET_PACKET{
	char type; // 패킷 타입
	CBullet bullet; // 투사체 정보
	}
		투사체 정보 수신
		관련 패킷
			SC_BULLET

11	typedef struct SC_ITEM_PACKET{
	char type; // 패킷 타입
	Citem item; // 투사체 정보
	}
		아이템 정보 수신
		관련 패킷
			SC_ITEM

12	typedef struct SC_RANK_PACKET{
	char type; // 패킷 타입
	int id[MAX_PLAYER]; // 플레이어 id
	int score[MAX_PLAYER]; // 플레이어 score
	char name[BUFSIZE][MAX_PLAYER];
	// 플레이어 별명
	}
		게임 클리어 후,
		플레이어들의 순위
		정보 수신 관련
		패킷
			SC_RANK

- 클라이언트 패킷 타입

번호	패킷 타입	설명
1	SELECT_CHARACTER	캐릭터 선택 관련 패킷 타입
2	CS_INIT_FINISH	초기화 완료 관련 패킷 타입
3	CS_KEYBOARD_INPUT	키 입력 관련 패킷 타입
4	CS_SELECT_LOBBY	로비 버튼 선택 관련 패킷 타입
5	CS_SELECT_EXIT	종료 버튼 선택 관련 패킷 타입

- 클라이언트 패킷 설명

번호	패킷	설명	패킷 타입

1	typedef struct SELECT_CHARACTER_PACKET{
	char type; // 패킷 타입
	int id; // id
	CHARACTER_TYPE character; // 캐릭터
	}
		캐릭터 선택 수신 관련 패킷	SELECT_CHARACTER

2	typedef struct CS_INIT_FINISH_PACKET {
	char type; // 패킷 타입
	int id; // id
	}
		초기화 완료 수신 관련 패킷	CS_INIT_FINISH

3	typedef struct CS_KEYBOARD_INPUT_PACKET {
	char type; // 패킷 타입
	KEY key; // 눌린 키
	KEY_STATE key_state; // 눌린 키의 상태
	// 1. 키가 막 눌렸을 때 (TAP)
	// 2. 키가 눌려있을 때 (HOLD)
	// 3. 키를 놓았을 때 (AWAY)
	}
		키 입력 수신 관련 패킷	CS_KEYBOARD_INPUT

4	typedef struct CS_SELECT_LOBBY_PACKET {
	char type; // 패킷 타입
	int id; // id
	}
		로비 버튼 선택 수신 관련 패킷	CS_SELECT_LOBBY

5	typedef struct CS_SELECT_EXIT_PACKET {
	char type; // 패킷 타입
	int id; // id
	}
		종료 버튼 선택 수신 관련 패킷	CS_SELECT_EXIT

3.2 서버

번호	함수명	설명
1	void sendPlayerId(SOCKET sock, int id);	플레이어 ID를 할당하고, 클라이언트에게 아이디를 송신하는 함수
2	void sendSelectCharacter(SOCKET sock, int id, CHARACTER_TYPE character);	클라이언트로 캐릭터 선택 정보를 송신하는 함수
3	void recvSelectCharacter(SOCKET sock);	클라이언트로부터 캐릭터 선택 정보를 수신하는 함수
4	void sendInitSignal(SOCKET sock);	클라이언트에게 초기화 신호를 송신하는 함수
5	int recvInitFinishSignal(SOCKET sock);	클라이언트로부터 초기화 완료 신호를 수신하는 함수 (플레이어 id값을 반환)
6	void sendGameStateSignal(SOCKET sock, GAME_STATE state);	클라이언트에게 게임 상태 신호를 송신하는 함수
7	void sendObjectInfo(SOCKET sock, CObject object);	클라이언트에게 오브젝트(플레이어, 몬스터, 보스, 투사체, 아이템) 정보를 송신하는 함수
8	void sendRankInfo(SOCKET sock);	클라이언트에게 순위 정보를 송신하는 함수
9	KeyInfo recvKeyBoardInput(SOCKET sock);	클라이언트로부터 키입력을 수신하는 함수 (키 정보를 반환)
10	int recvLobbySignal(SOCKET sock);	클라이언트로부터 로비 신호를 수신하는 함수 (플레이어 id값을 반환)
11	int recvExitSignal(SOCKET sock);	클라이언트로부터 종료 신호를 수신하는 함수 (플레이어 id값을 반환)
12	void progress(KeyInfo keyInfo);	클라이언트에게 받은 키입력을 처리하는 함수 (충돌 및 플레이어 이동)
13	bool IsCollision(CObject* _pLHS, CObject* _pRHS);	오브젝트간의 충돌을 검사하는 함수 (충돌 여부 반환)
14	void Management_Monster_Info(CMonster*)	몬스터 관리 함수

15	void Player_Rank(CPlayer*)	플레이어 랭킹 계산 함수

3.3 클라이언트

번호	함수명	설명
1	int recvPlayerId(SOCKET sock)	서버로부터 플레이어 ID를 수신하는 함수
2	void setId(int id)	플레이어 ID를 설정하는 함수
3	void sendSelectCharacter(SOCKET sock, int id, CHARACTER_TYPE character);	서버에게 캐릭터 선택 정보를 송신하는 함수
4	void recvSelectCharacter(SOCKET sock);	서버로부터 캐릭터 선택 정보를 수신하는 함수
5	void recvInitSignal(SOCKET sock);	서버로부터 초기화 신호를 수신하는 함수
6	void sendInitFinishSignal(SOCKET sock);	서버에게 초기화 완료 신호를 송신하는 함수
7	void recvGameStateSignal(SOCKET sock);	서버로부터 게임 상태 신호를 수신하는 함수
8	void recvObjectInfo(SOCKET sock);	서버로부터 플레이어 정보를 수신하는 함수
9	void recvRankInfo(SOCKET sock);	서버로부터 순위 정보를 수신하는 함수
10	void sendKeyBoardInput(SOCKET sock);	서버에게 키보드 입력을 송신하는 함수
11	void sendLobbySignal(SOCKET sock);	서버에게 로비 신호를 송신하는 함수
12	void sendExitSignal(SOCKET sock);	서버에게 종료 신호를 송신하는 함수

3.4 스레드 동기화 함수

번호	함수명	설명
1	DWORD WINAPI ClientThread(SOCKET client_sock);	각 클라이언트와의 통신을 담당할 스레드 함수. 접속한 클라이언트마다 생성되고, 클라이언트의 키입력을 수신하고, 서버의 메인 스레드의 progress 함수에서 계산한 값을 클라이언트에게 송신한다.

3.4 동기화 기법

여러 스레드가 동시에 같은 데이터에 접근하려고 할 때, 경쟁 상태가 발생할 수 있다. 예를 들어, 여러 플레이어가 동시에 같은 아이템을 줍거나, 몬스터나 보스에게 데미지를 주려고 할 때, 그 데이터(아이템의 상태나 몬스터, 보스의 체력)가 예상치 못하게 변경될 수 있다.
이런 상황을 해결하기 위해 std::mutex라는 도구를 사용하여 경쟁 상태를 방지한다.

4. 팀원 역할 분담

구현 내용	이름		
	박자은	이정범	우현정
[공통]패킷 구현	○	○	○
[클라]리소스 제작	○	○	○
[서버]패킷과 클라이언트에 맞게 서버를 구현	○		
[서버]클라이언트-서버 연결 sendPlayerId, recvPlayerId, setId 함수 구현			○
[서버]패킷 전달 확인			○
[서버]클라이언트에서 서버로 ip와 닉네임으로 접속 구현(cmd)	○		
[서버]클래스 구현(CCollisionMgr, CEventMgr, CKeyMgr, CTimer)(CBoss, CBullet, CMonster, CPlayer, CObject)(CCollider)		○	
[클라]클래스 보완(CCollisionMgr, CEventMgr, CKeyMgr, CTimer)(CBoss, CBullet, CMonster, CPlayer, CObject)		○	
[클라]게임 수정(캐릭터 부활 처리, ui, 아이템)		○	
[공통]클라이언트에서 서버로 전달 구현, 확인	○	○	○
[공통]서버에서 클라이언트로 전달 구현, 확인	○	○	○
[클라]로그인 화면 구현(시작화면)		○	○
[서버]id랑 닉네임으로 서버 접속 구현	○		
[공통]클라이언트에서 서버로 전달 구현, 확인	○		
[공통]서버에서 클라이언트로 전달 구현, 확인	○		
[클라]로비 화면 구현(캐릭터 선택, 모든 플레이어 닉네임 출력, 모든 플레이어 캐릭터 확인)		○	○
[서버]캐릭터 선택 확인, 게임 시작 확인 sendSelectCharacter, recvSelectCharacter, sendInitSignal, recvInitFinishSignal, sendGameStateSignal, recvGameStateSignal 함수 구현		○	
[공통]클라이언트에서 서버로 전달 구현, 확인	○	○	○
[공통]서버에서 클라이언트로 전달 구현, 확인	○	○	○
[공통]게임 진행 정보(먹은 아이템, 충돌처리, 플레이어 위치 등) 전달 구현, 확인 recvObjectInfo, sendObjectInfo, sendKeyBoardInput, recvKeyBoardInput, progress, IsCollision, Management_Monster_Info 함수 구현	○	○	
[클라]게임 오버 구현 및 리소스 제작	○	○	○
[클라]게임 클리어 구현 및 리소스 제작	○	○	○
[클라]플레이어 점수 구현 및 리소스 제작	○	○	○
[클라]로비버튼과 종료버튼 구현 및 리소스 제작	○	○	○
[서버]플레이어 점수 합산 처리, 로비신호와 종료신호 처리 Player_Rank, sendRankInfo, recvRankInfo, sendLobbySignal, recvLobbySignal, sendExitSignal, recvExitSignal 함수 구현			○

[공통]클라이언트에서 잘 동작하는지 확인(점검)	○	○	○
[서버]멀티쓰레드로 구현 ClientThread 함수 구현	○		
[공통]클라이언트에서 잘 동작하는지 확인(점검)	○	○	○
[공통]오류 확인 및 수정	○	○	○

5. 개발 환경

OS : Windows 11

IDE : Visual Studio 2022

통신 프로토콜 : TCP/IP

API : Win32 API / Windows Socket API

네트워크 IO모델 : 다중 스레드 모델

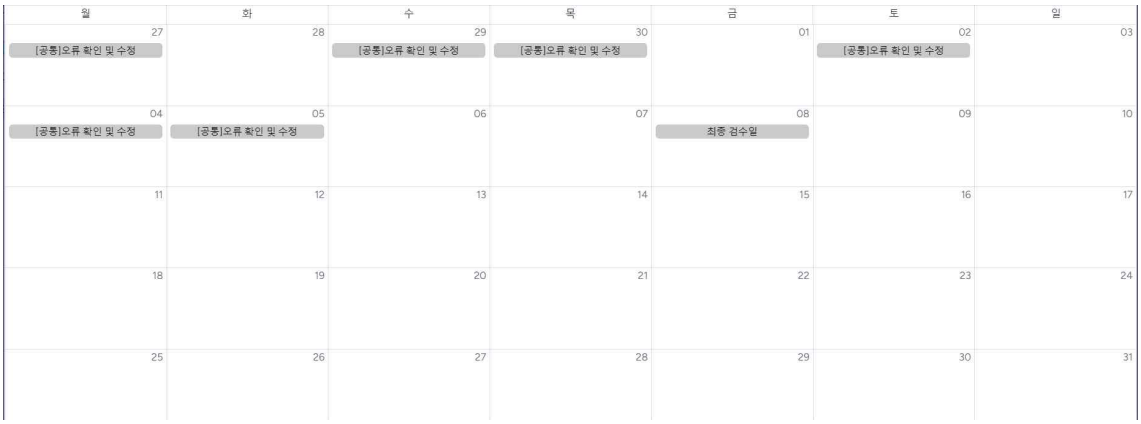
언어 : C / C++

관리 : 깃허브(https://github.com/JeongBeomLee/NetworkPrograming_TermProject)

6. 개발 일정

[박자은]

월	화	수	목	금	토	일
30	31 [공통]패킷 구현	01	02	03 [서버]클라이언트에서 서버...	04 [공통]클라이언트에서 서버... [공통]서버에서 클라이언트...	05 [공통]오류 확인 및 수정
06 [서버]id랑 닉네임으로 서버... [공통]클라이언트에서 서버... [공통]서버에서 클라이언트...	07	08 [서버]패킷과 클라이언트에...	09	10 [공통]클라이언트에서 서버... [공통]서버에서 클라이언트...	11	12
13 [공통]오류 확인 및 수정	14	15 [공통]게임 진행 정보(먹은 ...	16 [클라]게임 오버 구현 및 리...	17 [클라]게임 클리어 구현 및 ...	18 [클라]플레이어 점수 구현 ...	19 [클라]리소스 제작 [클라]로비버튼과 종료버튼...
20 [공통]오류 확인 및 수정	21 [공통]클라이언트에서 잘 ...	22	23	24	25	26 [서버]멀티쓰레드로 구현 [공통]클라이언트에서 잘 ...
27 [공통]오류 확인 및 수정	28	29 [공통]오류 확인 및 수정	30 [공통]오류 확인 및 수정	01	02 [공통]오류 확인 및 수정	03



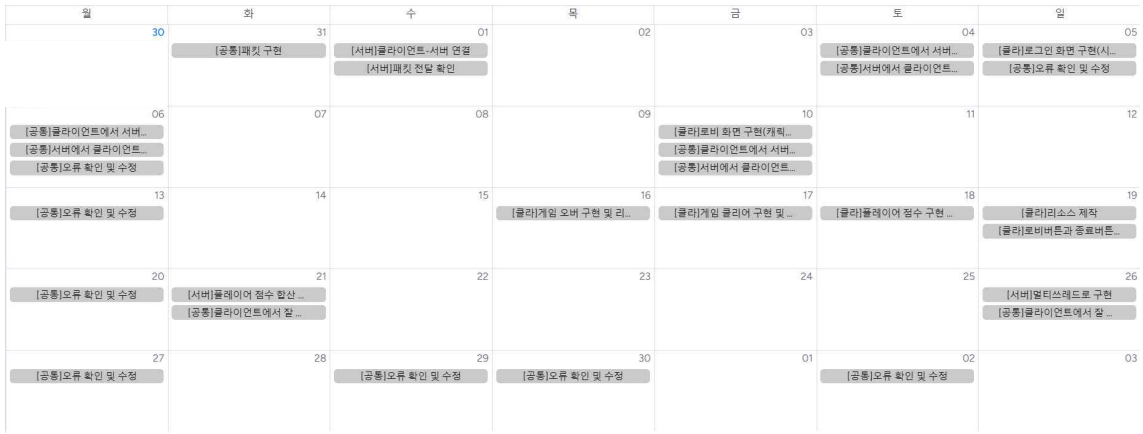
월	화	수	목	금	토	일
27 [공통]오류 확인 및 수정	28	29 [공통]오류 확인 및 수정	30 [공통]오류 확인 및 수정	01	02 [공통]오류 확인 및 수정	03
04 [공통]오류 확인 및 수정	05 [공통]오류 확인 및 수정	06	07	08 최종 검수일	09	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

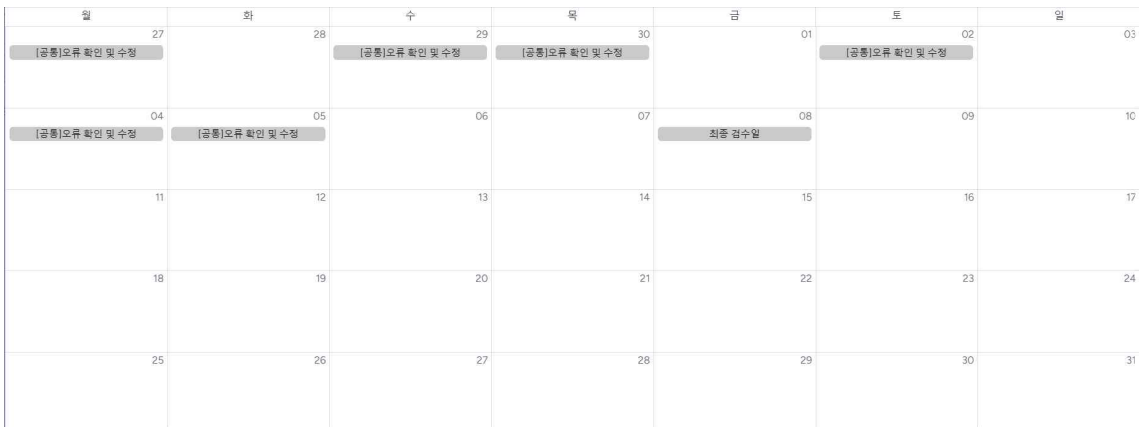
[이정범]

월	화	수	목	금	토	일
30	31 [공통]패킷 구현	01 [서버]클라이언트-서버 연결 [서버]패킷 전달 확인	02	03 [서버]클라이언트에서 서버... [서버]클래스 구현(CCollisi... [클라]클래스 보완(CCollisi...	04 [공통]클라이언트에서 서버... [공통]서버에서 클라이언트...	05 [클라]게임 수정(캐릭터 부... [클라]로그인 화면 구현(시... [공통]오류 확인 및 수정
06 [서버]id랑 닉네임으로 서버... [공통]클라이언트에서 서버... [공통]서버에서 클라이언트...	07	08	09	10 [클라]로비 화면 구현(캐릭... [서버]캐릭터 선택 확인, 게... [공통]클라이언트에서 서버...	11	12
13 [공통]오류 확인 및 수정	14	15 [공통]게임 진행 정보(먹은 ...	16 [클라]게임 오버 구현 및 리...	17 [클라]게임 클리어 구현 및 ...	18 [클라]플레이어 점수 구현 ...	19 [클라]리소스 제작 [클라]로비버튼과 종료버튼...
20 [공통]오류 확인 및 수정	21 [서버]플레이어 점수 합산 ... [공통]클라이언트에서 잘 ...	22	23	24	25	26 [공통]클라이언트에서 잘 ...
27 [공통]오류 확인 및 수정	28	29 [공통]오류 확인 및 수정	30 [공통]오류 확인 및 수정	01	02 [공통]오류 확인 및 수정	03


월 화 수 목 금 토 일
27 [공통]오류 확인 및 수정
28
29 [공통]오류 확인 및 수정
30 [공통]오류 확인 및 수정
01
02 [공통]오류 확인 및 수정
03
04 [공통]오류 확인 및 수정
05 [공통]오류 확인 및 수정
06
07
08 최종 검수일
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

[우현정]


월 화 수 목 금 토 일
30
31 [공통]패킷 구현
01 [서버]클라이언트-서버 연결
[서버]패킷 전달 확인
02
03
04 [공통]클라이언트에서 서버...
[공통]서버에서 클라이언트...
05 [클라]로그인 화면 구현(시...
[공통]오류 확인 및 수정
06 [공통]클라이언트에서 서버...
[공통]서버에서 클라이언트...
[공통]오류 확인 및 수정
07
08
09
10 [클라]로비 화면 구현(캐릭...
[공통]클라이언트에서 서버...
[공통]서버에서 클라이언트...
11
12
13 [공통]오류 확인 및 수정
14
15
16 [클라]게임 오버 구현 및 리...
17 [클라]게임 클리어 구현 및 ...
18 [클라]플레이어 점수 구현 ...
19 [클라]리소스 제작
[클라]로비버튼과 종료버튼...
20 [공통]오류 확인 및 수정
21 [서버]플레이어 점수 합산 ...
[공통]클라이언트에서 잘 ...
22
23
24
25
26 [서버]멀티쓰레드로 구현
[공통]클라이언트에서 잘 ...
27 [공통]오류 확인 및 수정
28
29 [공통]오류 확인 및 수정
30 [공통]오류 확인 및 수정
01
02 [공통]오류 확인 및 수정
03


월 화 수 목 금 토 일
27 [공통]오류 확인 및 수정
28
29 [공통]오류 확인 및 수정
30 [공통]오류 확인 및 수정
01
02 [공통]오류 확인 및 수정
03
04 [공통]오류 확인 및 수정
05 [공통]오류 확인 및 수정
06
07
08 최종 검수일
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31