**네트워크 게임 프로그래밍**

**Term Project**

- 추진 계획서 -

게임공학과 6조

2018180011 김우찬

2018180019 박소영

2018182002 고은비

목차

[1. 애플리케이션 기획 3](#_Toc86697686)

[1-1. 게임 소개 3](#_Toc86697687)

[1-2. 게임 진행방식 4](#_Toc86697688)

[1-3. 게임 프레임워크(Flow Chart) 5](#_Toc86697689)

[1-4. 게임 클래스 다이어그램 6](#_Toc86697690)

[2. 개발 환경 7](#_Toc86697691)

[3. High-level 디자인 8](#_Toc86697692)

[3-1. 서버-클라이언트 관계 8](#_Toc86697693)

[3-2. 서버 다이어그램 9](#_Toc86697694)

[3-3. 클라이언트 다이어그램 10](#_Toc86697695)

[3-4. 서버 플로우차트 11](#_Toc86697696)

[3-5. 클라이언트 플로우차트 13](#_Toc86697697)

[4. Low-level 디자인 14](#_Toc86697698)

[4-1. 패킷 유형 14](#_Toc86697699)

[4-2. 서버 전송방식 설명 15](#_Toc86697700)

[4-3. 서버 주요기능 설명 16](#_Toc86697701)

[4-4. 클라이언트 주요기능 설명 18](#_Toc86697702)

[4-5. 월간 계획표 & 개인별 계획 20](#_Toc86697703)

**1. 애플리케이션 기획**

**1-1. 게임 소개**



1. 게임 이름 : Crazy Arcade (모작)
2. 장르 : 액션, 전략 비디오 게임
3. 등장 캐릭터 : 배찌, 다오, 우니, 디즈니

다오는 기준 캐릭터이다.

배찌는 기준(다오)보다 속도가 1 더 빠르다.

우니는 기준(다오)보다 물줄기 파워가 1 더 세다.

디즈니는 기준(다오)보다 설치 가능 물풍선 수가 1개 더 많다.

1. 승리 조건 : 스테이지 안의 모든 몬스터(귤, 악어) 및 보스(물개)와 충돌하지 않고, 물줄기를 피하며 모든 적을 무찌르면 승리하게 된다.

**1-2. 게임 진행방식**

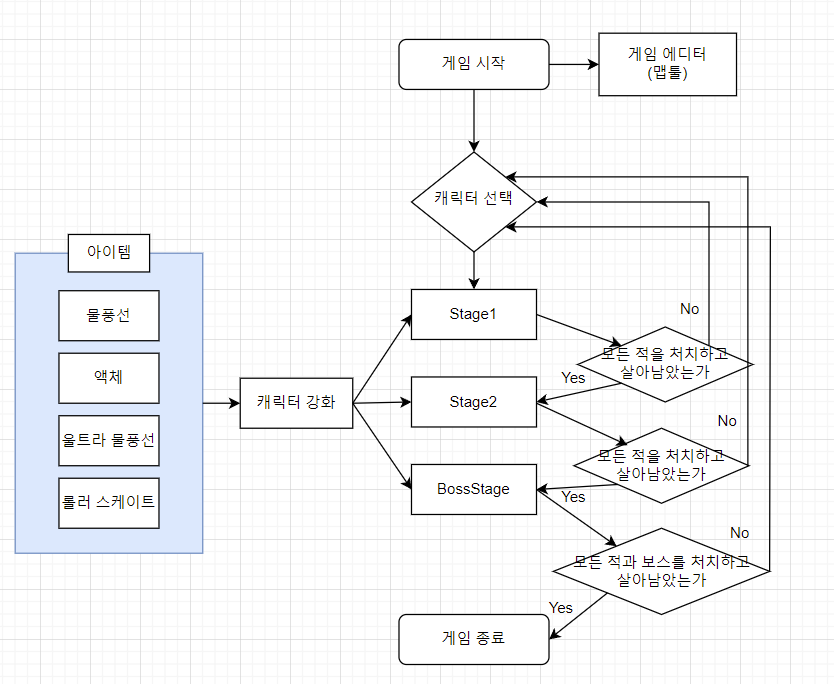
**- 본 게임은 1인용 게임으로 제작되었으나 텀프로젝트의 취지에 맞게 3인용 게임으로 재구성한다**

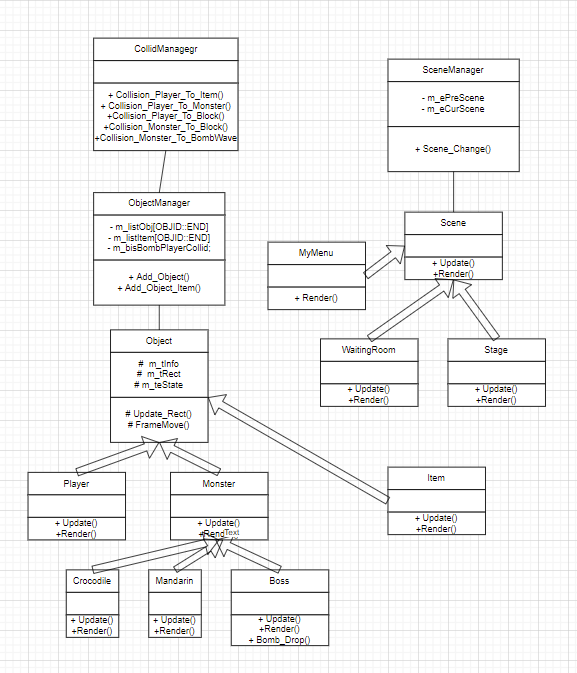
1. 처음 메뉴 화면에서 GameStart버튼을 눌러 대기실 씬으로 이동한다.
2. 대기실 씬에서 캐릭터를 선택하고 Stage씬으로 이동한다.
3. 플레이어 두명이 각자 스폰 포인트에서 근처 블록을 물풍선으로 깨며 아이템을 획득하여 캐릭터를 강화한다.
4. 본인이 설치한 물풍선에 닿거나, 몬스터와 충돌하거나, 보스가 던지는 물풍선에 맞게 되면 승리하지 못한다.
5. 보스의 공격 패턴인 물풍선 던지기와 구르기를 피하며 스테이지 안의 모든 몬스터 및 보스를 처치한다.
6. 만약 3인의 플레이어가 모두 충돌하게 되면 5초동안 동일한 버프 효과(이동속도 증가, 물줄기 최대 증가)를 얻는다.
7. 스테이지 안의 모든 몬스터를 처치하고 보스를 Bubble상태로 만들어 플레이어와 충돌하면 승리하게 된다.

**(예시)**



**1-3. 게임 프레임워크 (Flow Chart)**



**1-4. 게임 클래스 다이어그램**

**2. 개발 환경**

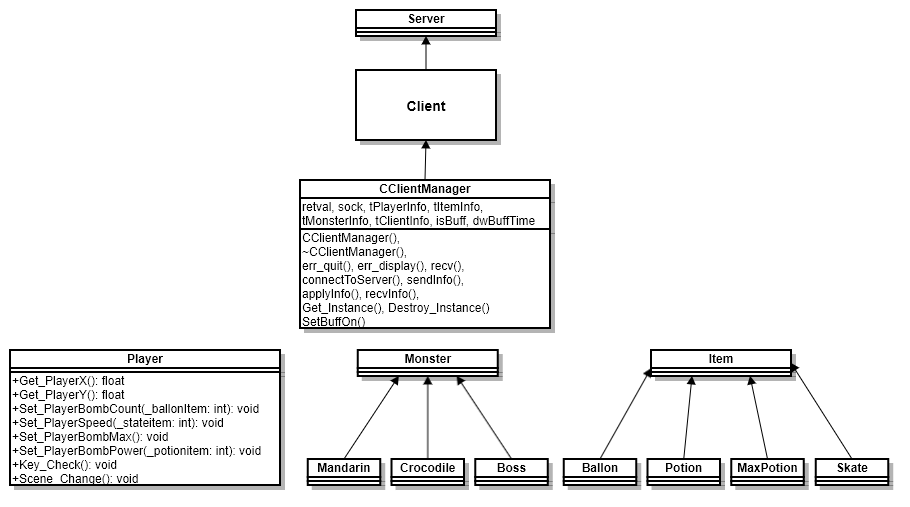
(1) Visual Studio 2019 (2) GitHub



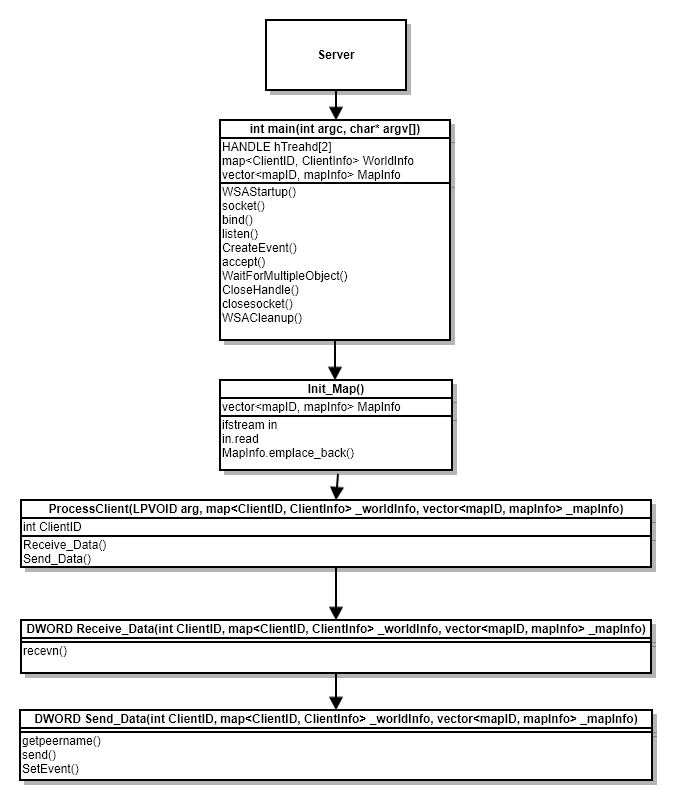
(3) Source Tree

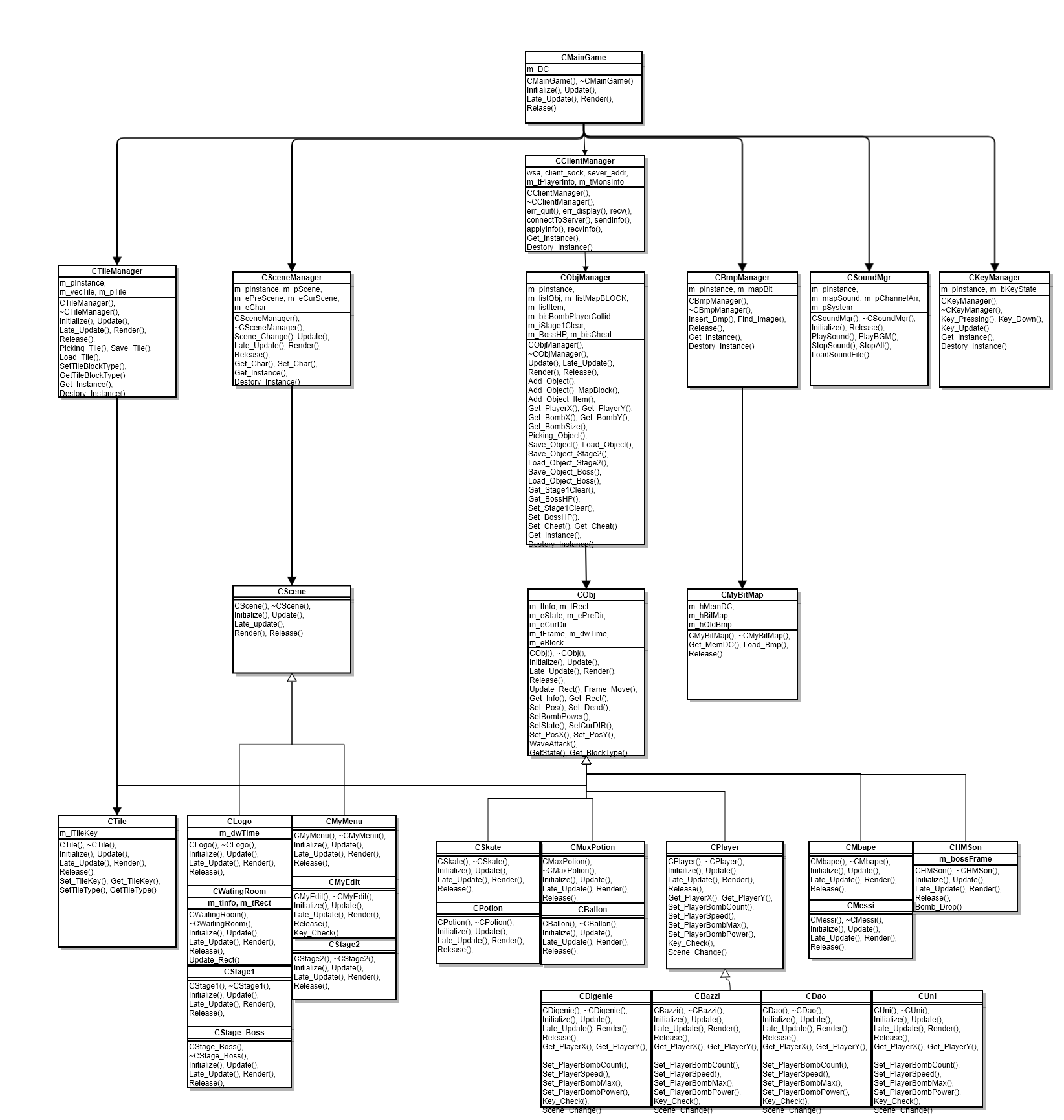


**3. High-level 디자인**

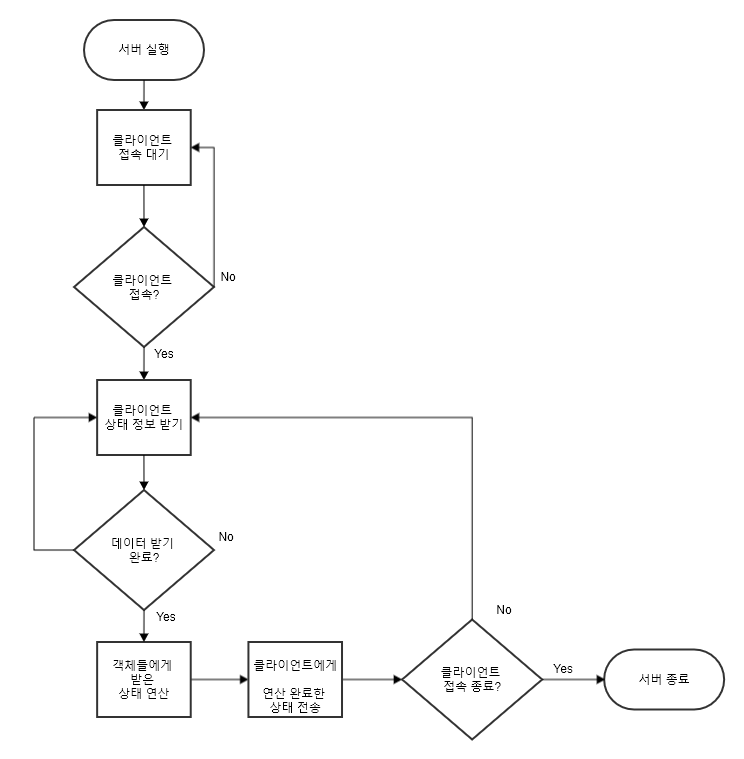
** 3-1. 서버-클라이언트 관계**

**3-2. 서버 다이어그램**



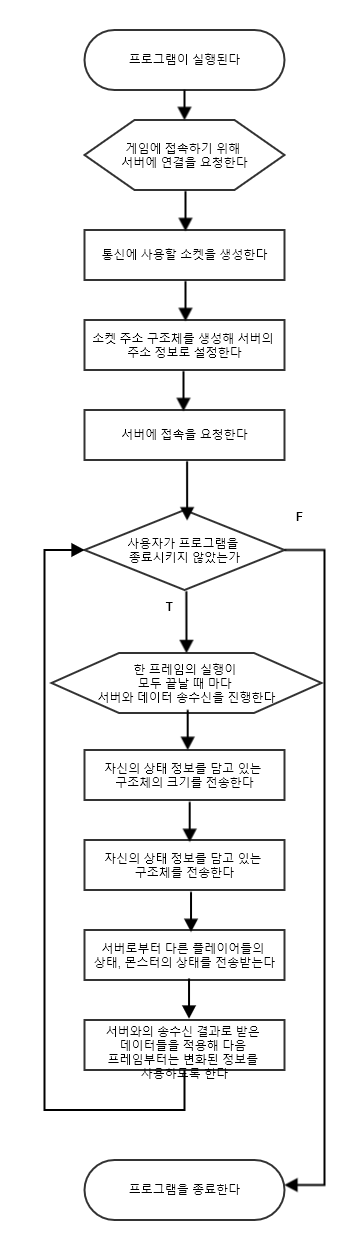
**3-3. 클라이언트 다이어그램**

**3-4. 서버 플로우차트**

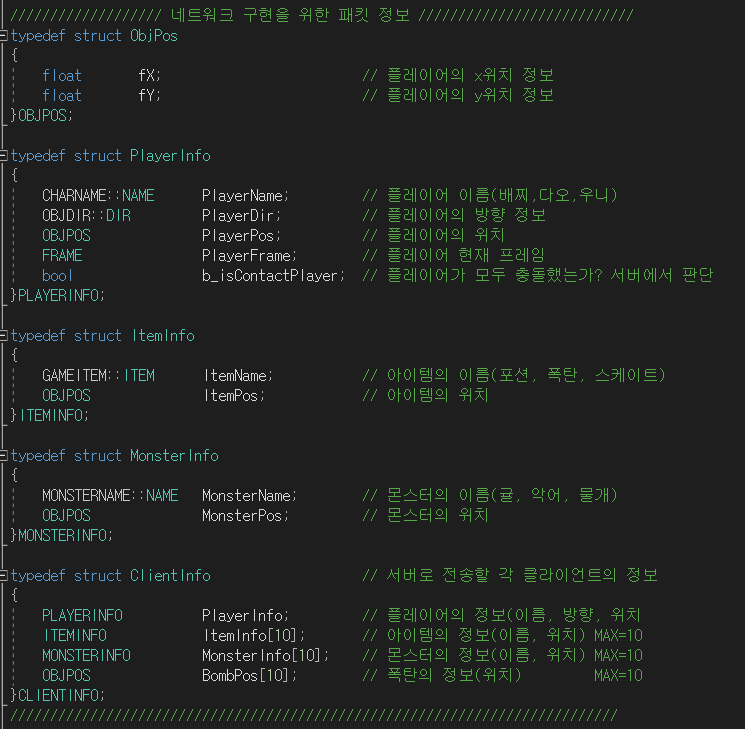
 (1) 간단한 서버 플로우차트

(2) 상세한 서버 플로우차트



**3-5. 클라이언트 플로우차트**

**4-1. 패킷 유형**



**서버로 보낼 패킷**

**4-2. 서버 전송방식 설명**

서버 전송 방식 : TCP/IP 고정 길이 전송방식

고정 데이터: 구조체(ClientInfo), 데이터 컨테이너(WorldInfo)

**\* 서버 전송 과정**

1. ClientID와 ClientInfo를 담을 컨테이너 map<ClientID, ClientInfo> 을 선언한다.
2. 클라이언트와 연결될때마다 각 클라이언트와 통신할 수 있는 스레드를 만든다
3. 스레드 동기화를 위한 컨테이너 mapIsReceive컨테이너에 ClientID를 key로 가진 bool변수를 false로 초기화 한 다음 삽입해준다.
4. 클라이언트와 연결되면 각각의 클라이언트마다 ClientID를 설정하여 클라이언트로 전송한다
5. 각각의 클라이언트에서 캐릭터 종류만 정하여 서버로 ClientInfo를 보내면 각 클라이언트로 모든 플레이어의 캐릭터 정보를 보내어 대기실 Scene을 완성한다
6. 모든 클라이언트들의 연결이 완료되면 아래와 같은 그림처럼 대기실 Scene이 그려진다(ClientID에 따라서 캐릭터의 위치 정해줌)
7. 인게임으로 들어가기 전 ClientID에 따라 초기 위치를 다르게 잡아서 클라이언트로 전송한다
8. mapIsReceive컨테이너에 ClientID의 value값을 확인한 후 false이면 (아직 데이터를 전송받지 않았으면) 각 클라이언트에서 보낸 구조체를 전송받는다
9. 각 클라이언트에서 보낸 ClientInfo 데이터 중 플레이어 정보(이름, 위치, 방향, 프레임, 물풍선 위치)를 ClientID로 구분한 맵 컨테이너에 추가한다
10. 클라이언트로부터 데이터 수신이 끝나면 mapIsReceive컨테이너에 ClientID에 맞는 value를 true로 바꿔준다.
11. 서버에서 받은 데이터를 바탕으로 각 클라이언트로 보낼 월드 패킷을 업데이트한다
12. mapIsReceive컨테이너 안의 모든 value값이 true이면 SetEvent()인자값으로

SendEvent를 넣어서 신호 상태로 만든다.

1. Send\_Data()안에서 Send()를 호출한 후 리턴값이 SOCK\_ERROR가 아니라면

mapIsReceive의 현재 ClientID의 value값을 false로 바꿔준다.

(업데이트된 월드 패킷을 각 클라이언트로 전송한다.)

1. mapIsReceive의 모든 value값이 false라면 SetEvent()인자값으로 ReceiveEvent를 넣어서 신호 상태로 만든다.
2. 8 ~ 15번을 반복하는 중 각 플레이어들의 위치가 모두 겹친다면 (충돌) ClientInfo 안의 PlayerInfo 안의 b\_isContactPlayer를 true로 바꿔주어 충돌했음을 알린다.

**4-3. 서버 주요기능 설명**

int main(int argc, char \*argv[]): 이벤트 2개를 생성해서 스레드 순서를 정한다.

DOWRD WINAPI ProcessClient(LPVOID arg): 클라이언트가 접속할 때마다 클라이언트와 통신할 스레드를 만든다.

DWORD Receive\_Data(LPVOID arg): 데이터 수신을 위한 함수

DWORD Send\_Data(LPVOID arg): 데이터 발신을 위한 함수

**\* 전역 변수**

1. map<ClientID, ClientInfo> WorldInfo: 클라이언트 ID를 Key값으로 받는 ClientInfo를 저장할 컨테이너 변수
2. #define SERVERPORT 9000: 서버 포트번호
3. HANDLE hReceiveEvent: 각 클라이언트와의 수신 결과를 알려주기 위한 핸들값
4. HANDLE hSendEvent: 각 클라이언트와의 송신 결과를 알려주기 위한 핸들값
5. map<ClientID, bool> mapIsReceive: 클라이언트에서 데이터를 전송 받았는지 판단하기 위한 맵
6. vector<USHORT> vecIsFirstConnect: 접속한 클라이언트의 포트번호를 저장하여 후에 클라이언트가 처음 접속했는지 판단하는 컨테이너
7. map<ClientID, bool> mapIsCollision: 버프를 위한 충돌 체크 확인 맵

**\* 전역 함수**

1. int main(int argc, char \*argv[])

윈속 초기화 후 클라이언트와 통신하기 위한 소켓을 생성한다. 서버의 로컬 IP 주소와 포트번호를 정해준다. 서버의 TCP 프로토콜 상태를 LISTENING으로 설정해서 클라이언트의 접속을 기다리게 한다.

클라이언트의 접속을 받는다. 접속한 클라이언트마다 스레드를 생성해준다.

vecIsFirstConnect컨테이너를 탐색하여 처음 접속했는지 여부를 확인한 뒤 처음 접속하였으면 접속한 클라이언트에게 ClientID를 지정하여 send()해준다.

1. DWORD WINAPI ProcessClient(LPVOID arg, map<ClientID, ClientInfo> \_worldInfo)

연결된 클라이언트와 Receive\_Data() 함수를 호출하여 데이터를 받고 Send\_Data() 함수를 호출하여 데이터를 연결된 클라이언트에게 보낸다.

1. DWORD Receive\_Data(LPVOID arg, map<ClientID, ClientInfo> \_worldInfo)

연결된 클라이언트로부터 각 플레이어의 ClientInfo를 받는다.

\_worldInfo의 ClientID 키값에 ClientInfo를 저장한다.

1. DWORD Send\_Data(LPVOID arg, map<ClientID, ClientInfo> \_worldInfo))

getpeername()으로 현재 통신하고 있는 클라이언트의 정보를 알아낸다..

현재 통신하고 있는 클라이언트에 \_WorldInfo를 보낸다.

1. Void CheckBuff()

모든 클라이언트에게 정보를 받은 후, 모든 플레이어가 충돌했으면 PlayerInfo 안의 b\_IsContactPlayer를 true로 설정해준다.

**4-4. 클라이언트 주요기능 설명**

클라이언트 내 서버와의 통신을 위해 새로 만들 클래스 : CClientManager

**\* 멤버 변수**

1. int retval : TCP 클라이언트 함수의 리턴값을 확인하기 위한 변수
2. SOCKET sock : 서버와 통신할 때 사용할 소켓
3. PLAYERINFO tPlayerInfo: 플레이어 정보를 담을 구조체 변수
4. ITEMINFO tItemInfo: 아이템 정보를 담을 구조체 변수
5. MONSTERINFO tMonsterInfo: 몬스터 정보를 담을 구조체 변수
6. CLIENTINFO tClientInfo: 클라이언트 정보를 담을 구조체 변수
7. bool isBuff: 버프 효과 판단을 위한 변수
8. DWORD dwBuffTime: 버프 지속시간을 나타내는 변수

**\* 전역 변수**

1. extern int ClientID

**\* 멤버 함수**

1. int connectToServer() : 윈속을 초기한 후, 서버와 통신을 하기 위한 소켓을 생성하고 서버에게 접속을 요청하는 함수

리턴값 : retval - connect()의 리턴값

1. void recvClientID() : 서버가 지정한 현재 클라이언트 아이디를 전역 변수 ClientID에 저장
2. int sendInfo() : 서버에게 자신이 현재 직접 플레이하고 있는 플레이어 정보와 현재 맵의 아이템 정보와 현재 맵의 몬스터 정보를 담고 있는 ClientInfo 구조체를 전송한다.

리턴값 : retval – send()의 리턴값

1. int recvInfo() : 서버로부터 모든 클라이언트의 플레이어 정보, 아이템 정보, 몬스터 정보를 담고 있는 WorldInfo 맵 컨테이너를 받는다.

리턴값 : retval – send()의 리턴값

1. void applyInfo() : 처음 호출될 땐 recvInfo()에서 서버로부터 받은 맵 컨테이너에 저장된 Key값의 수 만큼 플레이어를 만들고, 각 플레이어 위치를 Key값(ClientID)에 맞게 저장하고, 아이템과 몬스터 정보를 받아 인게임에 적용한다.
2. ~CClientManager() : 소켓을 닫고 윈속을 종료한다.
3. void Set\_BuffOn(): isBuff가 true라면 일정 시간동안(5초) (dwBuffTime) 스피드와 물줄기가 최대치가 된다.

**\* 멤버 함수 호출 순서**

1. CMainGame::CMainGame() 마지막 부분에서 connectToServer() 호출

if (isInit == true)

CClientManager::Get\_Instance()->sendInfo() 호출

CClientManager::Get\_Instance()->recvInfo() 호출

CClientManager::Get\_Instance()->applyInfo() 호출

isInit = false

1. CMainGame::Update() 마지막 부분에서

CClientManager::Get\_Instance()->Set\_BuffOn() 호출

1. CMainGame::Render() 마지막 부분에서

CClientManager::Get\_Instance()->sendInfo() 호출

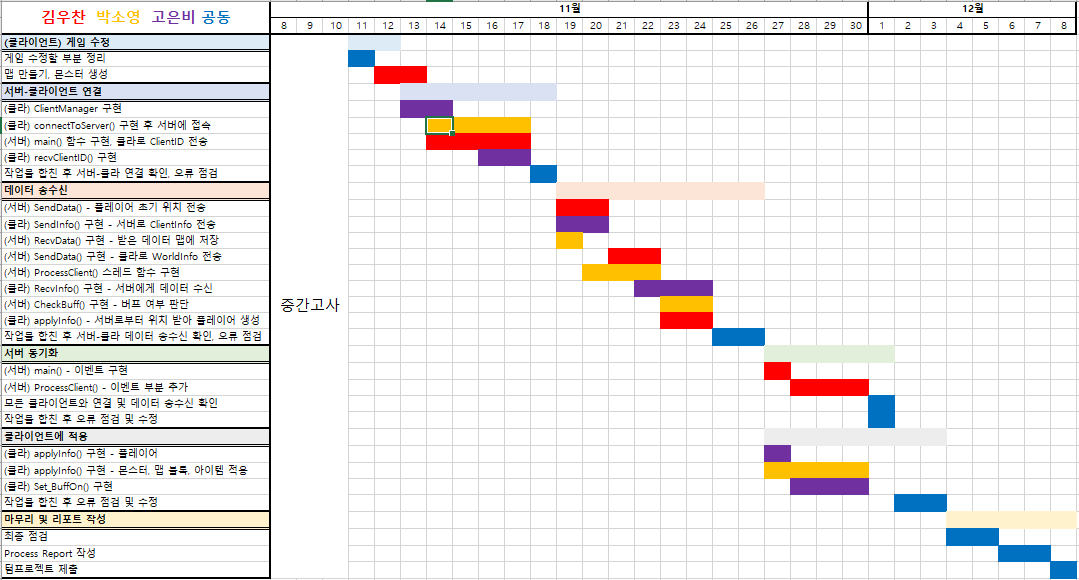
CClientManager::Get\_Instance()->recvInfo() 호출

CClientManager::Get\_Instance()->applyInfo() 호출

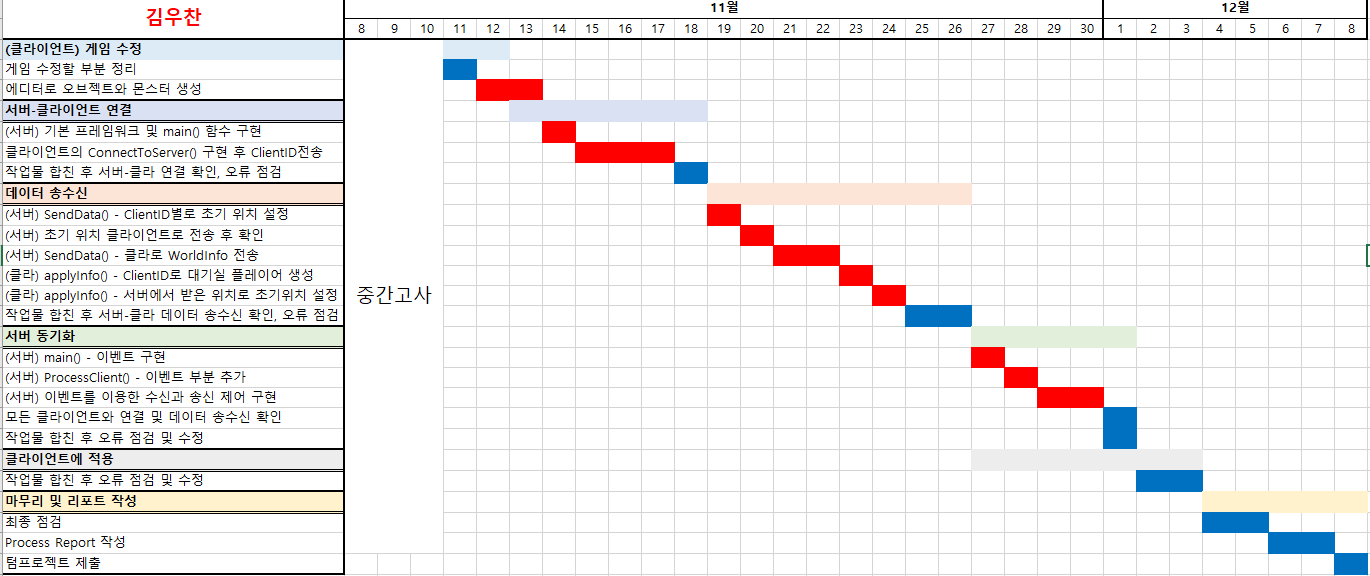
1. CMainGame::~ CMainGame() 마지막 부분에서

CClientManager::~ CClientManager()

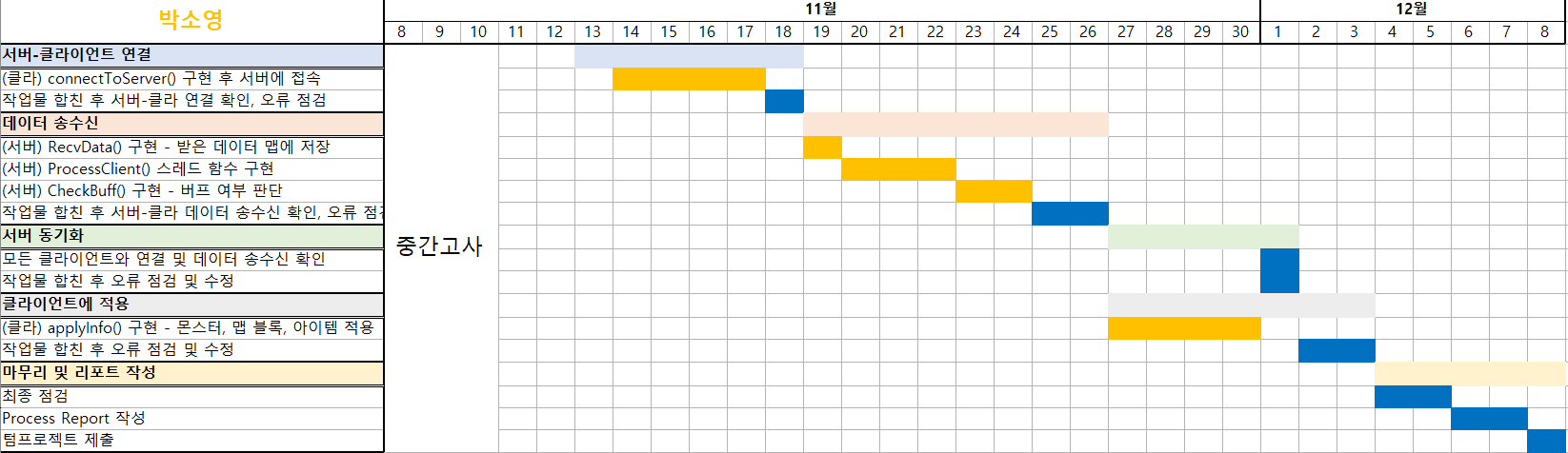
**4-6. 월별 계획 및 역할 분담**



**\* 개인별 계획 및 역할 분담 – 김우찬**



**\* 개인별 계획 및 역할 분담 – 박소영**



**\* 개인별 계획 및 역할 분담 – 고은비**

