1 - 표지

2 – 목차

3 ~

애플리케이션 기획

컴퓨터 그래픽스 최종 프로젝트 / 2020182031 이서연, 2020182037 정롭비 제작

기존 게임 간단한 설명

네트워크 결합 후 바뀐 점

같은 Scene에서 게임하고

3명이 협동하여 목표 지점까지 도달하면 클리어

ㄴ 자세한 게임 플레이 설명 (누가 블록을 설치하면 다른 누가 밟고 올라가거나 등등)

( 누가 목표 지점까지 가면 다같이 클리어함 )

원래 보이지 않던 플레이어 -> 아군 플레이어가 보임

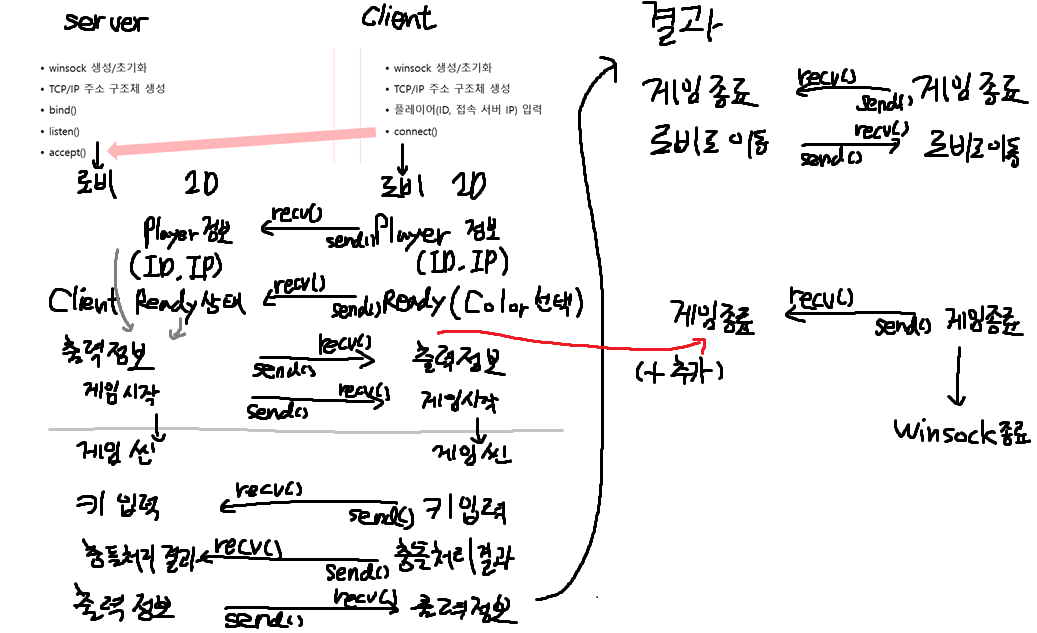
(+ 채팅 서버)

콘솔 등의 창에 IP쳐서 3인이 되면 게임 시작?

매치 메이킹 X

High-Level 디자인 – 클래스 단위 , 기능 단위

ex) 게임 흐름 구상도 (서버 그림, 클라 그림 두고 어떻게 굴러가는가)



1. 서버, 클라이언트 소켓 생성/연결

서버

서버가 실행되면 윈속 초기화와 소켓 하나를 생성하고 그 소켓을 bind()로 서버의 대기 소켓으로 만든다. 그 후 클라이언트의 연결이 이루어지면 클라이언트마다 전용 소켓과 스레드를 할당한다.

클라이언트

클라이언트에서 서버의 IP, 본인의 ID를 입력하고 올바른 입력과 서버와 연결 할 수 있는 상태일 때 서버와 연결한다.

1. 게임 로비

서버

각 클라이언트에게 저장되어 있는 모든 클라이언트 정보(ID, IP, Color, Ready 상태)를 보내고 클라이언트는 이 정보로 화면을 출력한다.

모든 클라이언트가 Ready 상태인 것을 확인하면, 서버는 게임 Scene 상태로 넘어가고 이 상태를 클라이언트들에게 넘겨준다.

클라이언트

클릭 메시지(Ready 버튼)로 클라이언트 정보(ID, IP, Color, Ready 상태)를 서버로 보내 서버는 접속된 모든 클라이언트의 정보(ID, IP, Color, Ready 상태)를 저장한다.

1. 게임 Scene

서버

클라이언트로부터 키 입력을 받으면 플레이어의 위치를 계산한다.

클라이언트로부터 받은 각 플레이어들위 위치정보, 상호작용 블록 및 아이템의 충돌처리 결과, 설치된 블록의 좌표를 클라이언트들에게 넘겨준다.

클라이언트

1. 게임 결과 및 종료

충돌처리 **Client** vs Server 장단점

Low\_Level 디자인 – 클래스 안의 함수, 세부적인 정보

ex) 관련되어 보이면 모든 함수 다 넣어

1. 데이터 전송구조

클라이언트

struct Vector3

enum MSG

{

MSG\_PLAYER\_INFO\_LOBBY – LOBBY에서의 플레이어 정보 메시지

MSG\_PLAYER\_INFO\_SCENE – SCENE에서의 플레이어 정보 메시지

MSG\_CHAT – 채팅 메시지

MSG\_ADD\_BLOCK – 블록을 설치하는 메시지

MSG\_COLLIDE – 충돌 메시지

MSG\_LEAVE – 플레이어가 종료한 것을 알려주는 메시지

MSG\_GAMECLEAR – 게임을 클리어한 것을 알려주는 메시지

}

Class Object {

int id – 오브젝트마다 고유 ID

}

Class Player {

char ID[10], char color, char chatBuf[30] – 플레이어 ID, 플레이어 색상, 채팅 버퍼

}

송신

void sendPlayerInfo() – 플레이어 정보를 보내준다.

* Lobby 상태일 때 - ID, color 순으로 데이터를 서버로 보내준다.
* Scene 상태일 때 – Position 데이터를 서버로 보내준다.

void sendChat() – chatBuf를 서버로 보내준다.

void sendCollide(int id) – 해당 id와 충돌되었다는 정보를 보내준다.

void sendAddBlock(float x, float z) – 블록을 설치할 위치를 서버에 보내준다.

수신

void recvPlayerInfo() – 다른 플레이어의 정보를 받아온다.

* Lobby 상태일 때 - ID, color 순으로 데이터를 받아온다.
* Scene 상태일 때 – Position 데이터를 받아온다.

void recvChat() – 채팅 메시지를 받아온다.

void recvCollide() – 충돌된 정보를 받아온다.

void recvAddBlock() – 블록 위치를 받고 떨어지는 블록을 설치한다.

void recvGameClear() – 게임을 클리어 했다는 메시지를 받는다.

void recvPlayerLeave(char\* id) – 해당 id를 가진 플레이어가 게임을 종료했다는 메시지를 받는다.

서버

송신

void sendPlayerInfo() – 모든 클라이언트에게 모든 플레이어 정보를 보내준다.

* Lobby 상태일 때 - ID, color 순으로 데이터를 서버로 보내준다.
* Scene 상태일 때 – Position 데이터를 서버로 보내준다.

void sendChat() – 채팅 메시지를 모든 클라이언트에게 보내준다.

void sendCollide(int id) – 입력된 id를 가진 오브젝트와 충돌된 것을 보내준다.

void sendAddBlock(float x, float z) – 모든 클라이언트에게 설치할 블록 위치를 보내준다.

void sendGameClear() – 게임을 클리어 했다는 메시지를 보내준다.

void sendPlayerLeave(char\* id) – 해당 id를 가진 플레이어가 게임을 종료했다는 메시지를 보내준다.

수신

void recvPlayerInfo() – 스레드마다 플레이어 정보를 받아온다.

* Lobby 상태일 때 - ID, color 순으로 데이터를 받아온다.
* Scene 상태일 때 – Position 데이터를 받아온다.

void recvChat() – 스레드마다 보낸 채팅 메시지를 받는다.

void recvCollide() – 충돌된 정보를 받는다.

void recvAddBlock() – 설치할 블록 위치를 받는다.

1. .
2. .

팀원 별 역할 분담

개발환경

운영체제 Windows 10

프로토콜 TCP/IP

IDE Visual Studio 2022

언어 C/C++ ws2\_32

라이브러리 OpenGL

VCS – DVCS , Git 형상 관리

개발일정(일별/개인별 계획 수립, 달력 형태로 작성)