Server를 어떻게 쓸까?

서버를 이용해서 통신할 변수들!

GameManager

OBJ\_MAX -> 최대 객체 개수 : X

objPool[] -> 게임에서 관리하는 객체 풀 : X

HeapCheckFlow-> 사용하지 않는 변수 : Delete

Page

First -> 초기화 확인 변수 : CheckInitialize로 이름변경 | X

Wait -> 삭제

Flow -> 삭제

TitleSprite\* -> 타이틀에서 사용할 배경화면 정보 : X

GameButton 3개 -> 타이틀 씬 조작버튼 : X

iConImage -> 아이콘 이미지 : X

iconNum -> 아이콘 인덱스 : X

ismenu -> 시작 씬과 선택 씬 확인 : O

nextChar -> 메뉴 선택 문자 : X

musicDataSet -> 노래 관련 데이터 : O

tempcount -> 현재 선택된 노래 인덱스 : X

PlayStation

isPlaying -> 재생여부 : X

autoPlaying -> 자동재생여부 : X

Time -> 현재 시간 : X

Score -> 점수 : O

Combo -> 콤보 수 : X

MaxCombo -> 최대 콤보 수 : X

noteArr -> 노트 배열 : X

ShowArr -> 쇼 배열 : 삭제

Background -> 배경 스프라이트 : X

EditStation

PlayStation -> 테스트화면 객체 : X

Tempo -> 템포 : X

Time -> 시간 : X

checkInitialize -> 초기화 확인 : X

bPlay -> 일시정지 확인 : X

MaxShowTime -> 화면에 보여지는 최대 줄 개수 : X

SaveLevelBtn\_FileName -> 저장할 곡 이름 : O

서버를 구상해보자

Struct musicData

{

Char\* musicname

Char\* notename

}

Struct LoginInfo

{

Char\* id

Char\* password

}

Struct UserInfo

{

Char\* id

Unsigned int maxscore

}

Struct LobbyInfo

{

unsigned int LobbyID

Char\* id[4]

Bool isReady[4]

unsigned int musicindex

}

Struct PlayStationInfo

{

Char\* id[4]

Unsigned int score[4]

}

Class Server

{

Variable

Vector<MusicData> musicDataSet : 노래 정보(노래 이름, 노트 이름) 저장

Vector<LoginInfo> loginInfoSet : 서버에 저장한 유저 로그인 정보(ID, Password)

Vector<UserInfo> userInfoSet : 게임에 접속한 유저의 데이터(ID, 최대 점수)

Vector<LobbyInfo> lobbySet : 로비정보(로비id, 유저 정보, 진행여부)

Function

Void CheckLogin() : 클라에서 ID/Password를 수신받아서 로그인여부를 송신 O

Void UploadMusic() : 클라에서 업로드한 노래정보를 musicDataSet에 저장 O

Void DownloadMusic() : 서버에 있는 노래 데이터 및 musicDataSet를 클라에 송신 O

Void RecvUserScore() : 점수를 수신받아서 서버의 해당 유저의 점수에 저장 O

Void SendAllScore() : ID와 점수를 클라에 송신 O

Void SendPlayersScore() : 로비에 있는 플레이어들의 점수를 수신 O

Void SendGameResult() : 클라에 게임 결과를 송신 O

Void EnterLobby() : 로비 입장 O

Void LeaveLobby() : 로비 퇴장 O

Void EnterPlayStation() : 클라에서 게임이 시작되게함

Void LeavePlayStation() : 클라에서 게임이 끝났음을 알림

Void RecvReady()

Void RecvCancleReady()

Void SendAllReady() : 로비에서 스타트할 때 전부 레디상태인지 확인

}

우리는 서버로 무엇을 할까?

1. 에딧모드에서 만든 노래를 업로드 O
2. 서버에 있는 노래 정보를 클라에 다운로드 O
3. 클라에서 획득한 점수를 서버에 저장 O
4. 서버에 저장된 점수를 클라에서 확인
5. 아이디/비밀번호를 확인해서 로그인
6. 멀티플레이 기능
7. 로비 추가
8. 게임 플레이시 멀티중인 유저의 점수 표시 O
9. 종료 후 전체 결과 확인 O
10. 승패를 기록해서 서버에 저장 O
11. 로비로 복귀 O

Lobby class

Variable

LobbyInfo lobbydata

Function

RecvLobbyInfo() : 유저가 들어왔을때 로비의 유저 닉네임 목록 갱신

SendPlayerReady() : 클라에서 서버로 준비상태 송신

SendGameStart() : 호스트가 게임시작을 서버에 송신 - Page에 같은 함수명이 있는데 이 클래스에서 이 함수명을 사용하는 것이 더 맞을듯함

Update() : 로비 갱신

클라 클래스에서 network::send 호출

Network 클래스에서 해당 함수 실행

* 관련 패킷 구조체 생성 (구조체변수: 패킷 타입, 크기, 관련 데이터)
* 구조체에 데이터 입력
* Send 실행
* 끝!

Network class -> 클라의 통신 담당 클래스

Variable

SOCKET socket

Static Network\* pInst;

Function

Static Network\* GetInst()

Bool Init()

Bool Connect()

Void SendCheckLogin(char\* id, char\* password) : 전달받은 id,ps를 서버에 송신

Void ProcessCheckLogin()

Void SendDownloadMusic() : 서버의 musicdata 다운로드 요청

Void ProcessDownloadMusic() : 서버의 musicdata 다운로드 수신

Void SendRequestAllScore() : 서버의 상위 점수 유저id 요청

Void ProcessRequestAllScore() : 상위 점수 유저id 수신

Void SendEnterLobby() : 서버에 로비 입장 여부를 송신

void ProcessEnterLobby()

Void SendLeaveLobby() : 서버에 로비 퇴장 송신

Void ProcessLeaveLobby()

Void RecvLobbyInfo():

Void SendGameResult(char lobbyNum, unsigned int ) : 서버에 게임결과 전송

Void SendEnterPlayStation() : 서버에 PlayerStation 입장 여부(게임 시작 여부) 송신

Void ProcessEnterPlayStation()

Void SendLeavePlayStation() : 서버에 PlayerStation 퇴장 여부 송신

Void ProcessLeavePlayStation()

Void SendPlayersScore()

Void ProcessPlayersScore()

Void SendReady()

Void SendCancleReady()

Void RecvAllReady()

Void SendUploadMusic(char\* music, char\* note) : 서버에 음악 업로드