Project Progress Report

2022180031 이우진, 2022184012 김유빈, 2022182041 하승민

1. 애플리케이션 기획

[Trip of the ball]

하승민, 김유빈, 윈도우프로그래밍

- **게임 설명**

  기존 게임 바운스볼에 블록을 추가/삭제하고 새로운 모드를 추가하여 구상한 게임입니다.

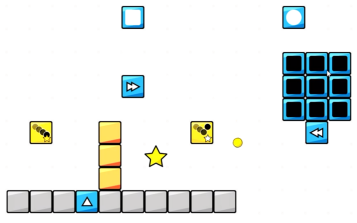
- **게임 구성**
  - **스테이지 모드**

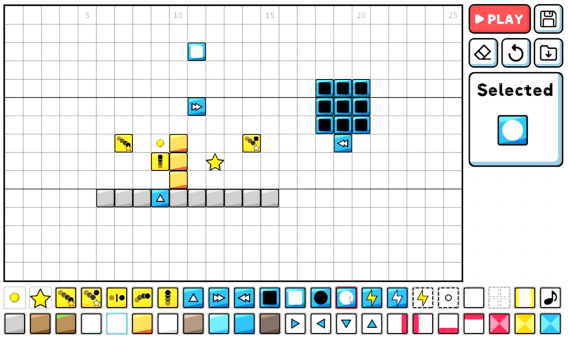
    두 플레이어가 스테이지 내의 별을 모두 얻는 것이 목표이다. 별을 모두 얻으면 스테이지 선택 화면으로 넘어간다. 장애물에 닿거나 아래로 떨어지면 죽고 두 플레이어 모두 죽으면 해당 스테이지를 처음부터 다시 해야 한다.

  - **커스텀 모드**

    ~~플레이어들이 직접 블록을 선택, 설치하여 같이 맵을 제작하고 플레이 할 수 있는 모드이다. 만든 맵을 서버에 올리거나 서버에서 사람들이 올린 맵을 다운 받아서 플레이 해볼 수 있다.~~

    각 플레이어가 직접 블록을 선택, 설치하여 맵을 제작하고, 완성한 맵을 서버로 보낸다. 모든 플레이어들은 서버로부터 플레이어들이 만든 커스텀 맵 목록을 받고 선택하여 멀티 플레이가 가능하다.

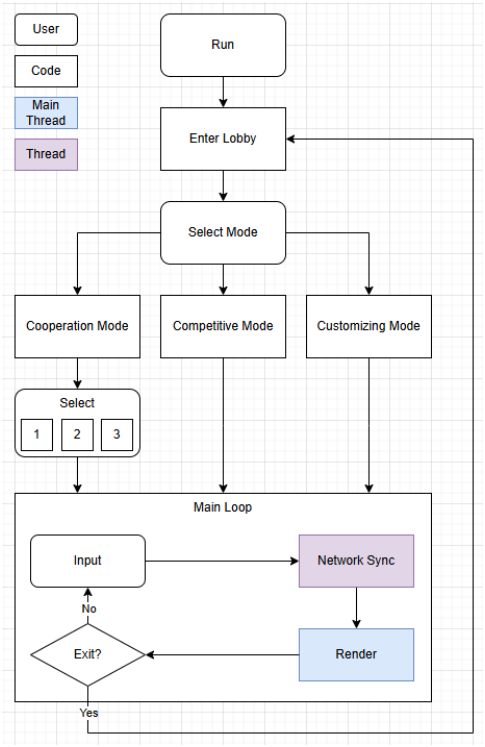




2. 하이 레벨 디자인

- **클라이언트**

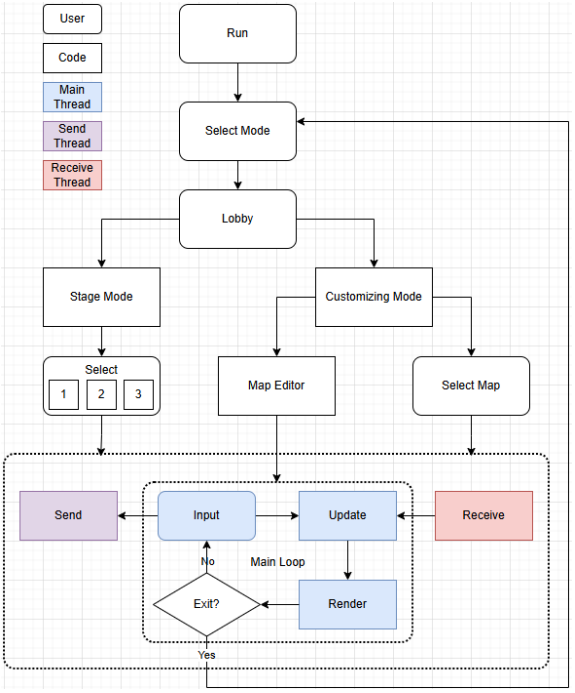
~~-기존~~



~~**클라이언트 흐름**~~

1. ~~클라이언트 프로그램 실행~~
2. ~~ClientManager 객체의 이니셜라이징, 소켓 생성 후 서버에 연결 요청~~
3. ~~서버와 연결이 되면 로그인 정보를 서버로 전달~~
4. ~~서버로부터 유저 id를 수신받음~~
5. ~~로그인 후에 로비 씬으로 입장 후 준비완료 버튼을 눌러 서버에게 준비완료 정보를 전달함~~
6. ~~모든 클라이언트가 준비완료 상태가 되면 게임모드 선택창으로 입장함~~

-수정



클라이언트 흐름

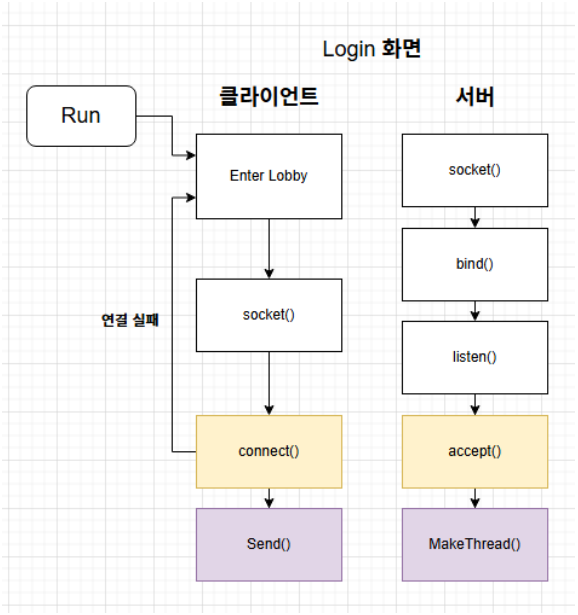
1. 클라이언트 프로그램 실행
2. ClientManager 객체의 이니셜라이징, 소켓 생성 후 서버에 연결 요청
3. 서버와 연결이 되면 로그인 정보를 서버로 전달
4. 서버로부터 유저 id를 수신받음
5. 로그인 후에 메인 메뉴에서 게임모드 선택
6. 첫 번째 클라이언트는 선택한 맵에서 다른 클라이언트가 접속할 때까지 로비에서 대기
7. 다른 클라이언트도 접속 시 스테이지 시작

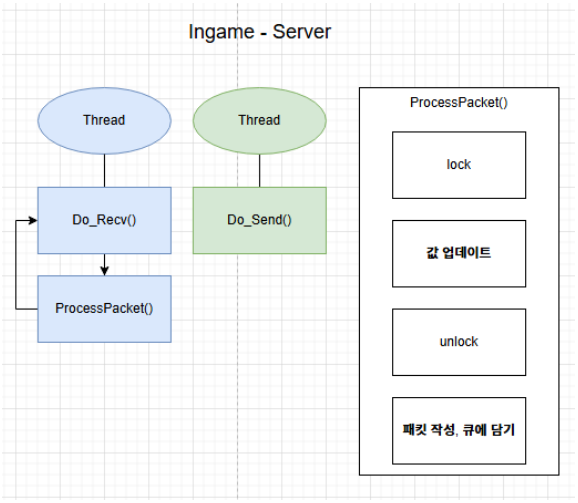
7. ~~첫 번째 클라이언트가 게임 모드 선택 후 게임 루프에 돌입함~~
8. ~~메인 쓰레드에서 키 입력을 받음~~
9. ~~ClientMain 쓰레드에서는 키 입력에 따라 서버로부터 송수신을 받음~~

8. 메인 쓰레드에서 키 입력을 받음
9. send 쓰레드에서는 키 입력에 따라 서버로송신함
10. Receive쓰레드에서는 서버로부터 데이터를 수신받아 데이터를 변경함

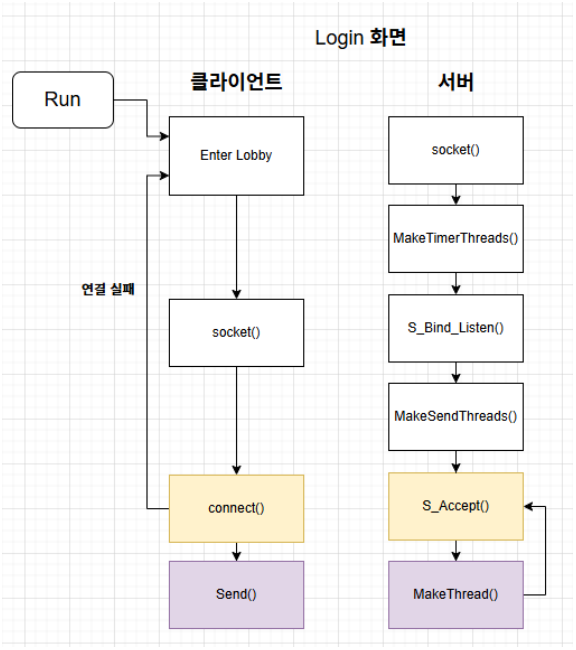
2.2 서버

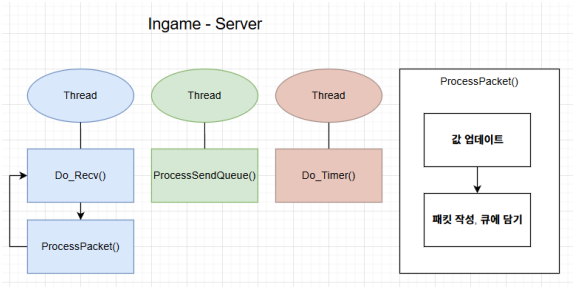
수정 전





수정 후





Thread 접속중인 클라이언트마다 각 1개, 데이터를 받은 후 값을 변경하고 패킷을 작성해 큐에 담는다.

Thread 프레임마다 send()를 진행한다.

Thread 타이머 역할, 게임 중이라면 프레임 단위로 frame packet을 보낸다.

MainThread 새 클라이언트를 받는다.

동기화 → concurrent_queue를 사용해 보낼 패킷을 담아두고 프레임마다 send()

- **서버 흐름**

1. socket(), bind(), listen();
2. accept() 대기
3. 클라이언트가 접속하면 담당 thread 1개 부여, id 회신
4. 게임에 필요한 정보 송수신

3. 로우 레벨 디자인

1. 공통

1.1 enum 클래스

1.1.1 Direction

공과 블록이 어느 방향으로 충돌했는지를 표시한다.

dirRight	블록의 오른쪽에서 충돌
dirLeft	블록의 왼쪽에서 충돌
dirDown	블록의 아래쪽에서 충돌
dirUp	블록의 위쪽에서 충돌

1.1.2 BlockType

블록의 종류를 표시한다.

JumpBk	점프 블록
RStraightBk	오른쪽 직진 이동 블록
LStraightBk	왼쪽 직진 이동 블록
RectBHBk	사각형 블랙홀 블록
CircleBHBk	원 블랙홀 블록

RectWHBk	사각형 화이트홀 블록
CircleWHBk	원 화이트홀 블록
ClimbBK	끈적이 블록
MusicBk	주크박스 블록
Star	별
BasicBk	지형 블록
SwitchBk	전기 스위치 블록
ElectricBk	전기 블록

1.1.3 Game

게임의 상태를 표시한다.

Start	시작 화면
StageSelect	스테이지 선택창
StagePlay	스테이지 모드 플레이 상태
StageStop	스테이지 모드 일시정지 상태
StageClear	스테이지 모드 클리어 상태
StageWaiting	스테이지 모드 대기 상태
StageDeath	스테이지 모드에서 죽은 상태
CustomMode	커스텀 모드 상태
CustomPlay	커스텀 모드 플레이 상태
CustomSelect	커스텀 모드 선택창 1
CustomSelect2	커스텀 모드 선택창 2
CustomDeath	커스텀 모드에서 죽은 상태

1.1.4 Key

키 입력을 관리한다.

```cpp
enum class KEY_TYPE
{
    LEFT = VK_LEFT,
    RIGHT = VK_RIGHT,
    ESCAPE = VK_ESCAPE,
    SPACE = VK_SPACE,
    L     = 'L',

    LBUTTON = VK_LBUTTON,
    RBUTTON = VK_RBUTTON
};

enum KEY_STATE : char
{
    NONE,
    PRESS,
    DOWN,
    UP,
    END
};

enum
{
    KEY_TYPE_COUNT = static_cast<UINT>(UINT8_MAX + 1)
};
```

1.2 protocol.h

1.2.1 전역 상수

```cpp
constexpr int NAME_SIZE = 20;
constexpr int M_WIDTH   = 25;
constexpr int M_HEIGHT  = 15;    // map size, block의 개수
constexpr int B_WIDTH   = 20;
constexpr int B_HEIGHT  = 15;    // block size, block의 개수

static const int MAX_USER = 2; /*std::thread::hardware_concu

constexpr short g           = 10;   // 중력가속도
constexpr float t           = 0.3;  // 속도 조절 변수
constexpr short side        = 60;   // 블럭 한 변의 길이
constexpr float rd          = 12.5; // 공 반지름
```

1.2.2 패킷 아이디

```
// Client -> Server Packet ID --------------------

constexpr char CS_LOGIN                     = 0;
constexpr char CS_KEY_PRESS                 = 1;
constexpr char CS_MOUSE_POS                 = 2;
constexpr char CS_SAVE_CUSTOM_MAP           = 3;
constexpr char CS_LOAD_CUSTOM_MAP_LIST      = 4;
constexpr char CS_SELECT_LOAD_CUSTOM_MAP    = 5;

// Server -> Client Packet ID --------------------

constexpr char SC_LOGIN_INFO        = 0;
constexpr char SC_FRAME             = 1;
constexpr char SC_DEATH             = 2;
constexpr char SC_EDIT_MAP          = 3;
constexpr char SC_LOAD_MAP          = 4;
constexpr char SC_LOGOUT            = 5;
constexpr char SC_GAME_STATE        = 6;
constexpr char SC_SOUND_STATE       = 7;
constexpr char SC_CUSTOM_MAP_LIST   = 8;
```

1.2.3 패킷 부모 구조체

```
struct PACKET {
    unsigned short  size;
    char            packetID;
    char            sessionID;

    PACKET() = default;
    PACKET(unsigned short s, char id, char sID)
        : size(s), packetID(id), sessionID(sID) {}
    ~PACKET() = default;
};
```

1.2.4 클라이언트 → 서버 패킷

```
// Client -> Server Packet ----------------------

struct CS_LOGIN_PACKET : PACKET {   // 로그인 패킷
    char name[NAME_SIZE]{ 0 };
    CS_LOGIN_PACKET(char sID)
        : PACKET(sizeof(CS_LOGIN_PACKET), CS_LOGIN, sID) {}
};

struct CS_KEY_PACKET : PACKET {     // 키 입력 패킷
    KEY_TYPE keyType;
    KEY_STATE keyState;
    CS_KEY_PACKET(char sID)
        : PACKET(sizeof(CS_KEY_PACKET), CS_KEY_PRESS, sID) {}
};

struct CS_MOUSE_PACKET : PACKET {   // 마우스 입력 패킷
    KEY_TYPE keyType;
    POINT mousePos;
    CS_MOUSE_PACKET(char sID)
        : PACKET(sizeof(CS_MOUSE_PACKET), CS_MOUSE_POS, sID) {}
};

struct CS_SAVE_CUSTOM_MAP_PACKET : PACKET { // 맵 저장 패킷
    char            map[M_WIDTH * M_HEIGHT];
    unsigned short  x, y, isSwitchOff;
    char mapName[NAME_SIZE]{ 0 };

    CS_SAVE_CUSTOM_MAP_PACKET(char sID)
        : PACKET(sizeof(CS_SAVE_CUSTOM_MAP_PACKET), CS_SAVE_CUSTOM_MAP, sID) {}
};

struct CS_LOAD_CUSTOM_MAP_LIST_PACKET : PACKET {    // 커스텀 맵 리스트 로드 패킷
    char mapName[NAME_SIZE]{ 0 };

    CS_LOAD_CUSTOM_MAP_LIST_PACKET(char sID)
        : PACKET(sizeof(CS_LOAD_CUSTOM_MAP_LIST_PACKET), CS_LOAD_CUSTOM_MAP_LIST, sID) {}
};

struct CS_SELECT_LOAD_CUSTOM_MAP_PACKET : PACKET {  // 커스텀 맵 로드 패킷
    char mapName[NAME_SIZE]{ 0 };

    CS_SELECT_LOAD_CUSTOM_MAP_PACKET(char sID)
        : PACKET(sizeof(CS_SELECT_LOAD_CUSTOM_MAP_PACKET), CS_SELECT_LOAD_CUSTOM_MAP, sID) {}
};
```

1.2.5 서버 → 클라이언트 패킷

```cpp
// Server -> Client Packet ----------------------

struct SC_LOGIN_INFO_PACKET : PACKET {
    unsigned short  c_id;
    char            name[NAME_SIZE];
    SC_LOGIN_INFO_PACKET(char sID)
        : PACKET(sizeof(SC_LOGIN_INFO_PACKET), SC_LOGIN_INFO, sID) {}
};

struct SC_LOGOUT_PACKET : PACKET {
    unsigned short  c_id;
    SC_LOGOUT_PACKET(char sID)
        : PACKET(sizeof(SC_LOGOUT_PACKET), SC_LOGOUT, sID) {}
};

struct SC_FRAME_PACKET : PACKET {
    unsigned short  c1_id;
    unsigned short  x1, y1;
    unsigned short  c2_id;
    unsigned short  x2, y2;
    SC_FRAME_PACKET(char sID)
        : PACKET(sizeof(SC_FRAME_PACKET), SC_FRAME, sID) {}
};

struct SC_DEATH_PACKET : PACKET {
    unsigned short  c1_id;

    SC_DEATH_PACKET(char sID)
        : PACKET(sizeof(SC_DEATH_PACKET), SC_DEATH, sID) {}
};

struct SC_EDIT_MAP_PACKET : PACKET {
    char            block[sizeof(Block)];
    SC_EDIT_MAP_PACKET(char sID)
        : PACKET(sizeof(SC_EDIT_MAP_PACKET), SC_EDIT_MAP, sID) {}
};

struct SC_LOAD_MAP_PACKET : PACKET {
    char            map[M_WIDTH * M_HEIGHT];

    SC_LOAD_MAP_PACKET(char sID)
        : PACKET(sizeof(SC_LOAD_MAP_PACKET), SC_LOAD_MAP, sID) {}
};

struct SC_GAME_STATE_PACKET : PACKET {
    int             gameState;
    SC_GAME_STATE_PACKET(char sID)
        : PACKET(sizeof(SC_GAME_STATE_PACKET), SC_GAME_STATE, sID) {}
};

struct SC_SOUND_STATE_PACKET : PACKET {
    int             soundState;
    SC_SOUND_STATE_PACKET(char sID)
        : PACKET(sizeof(SC_SOUND_STATE_PACKET), SC_SOUND_STATE, sID) {}
};

struct SC_CUSTOM_MAP_LIST_PACKET : PACKET {
    char mapList[NAME_SIZE * 10]{ 0 };

    SC_CUSTOM_MAP_LIST_PACKET(char sID)
        : PACKET(sizeof(SC_CUSTOM_MAP_LIST_PACKET), SC_CUSTOM_MAP_LIST, sID) {}
};
```

2. 서버 low-level 디자인

2.1 구조체

2.1.1 Ball

공과 관련된 정보를 담는다.

float x, y	위치 정보
float remx, remy	이전 위치 정보
float vx, vy	속력
float ax	가속도

2.1.2 Block

블록과 관련된 정보를 담는다.

int x, y	위치 정보 (열, 행)
int type, subtype	블록의 타입

2.1.3 CrashedBk

충돌 처리를 위한 구조체이다.

int dir	충돌 방향
int i, j	블록의 위치
float x, y	공의 위치
int quality	충돌한 블록의 충돌 처리 우선순위

2.1.4 floatRECT

게임 월드에서 실수형 사각형 범위를 표현한다. 자세한 충돌 처리를 위한 구조체이다.

float left, top, right, bottom	좌우상하 좌표

2.2 Session 클래스

한 클라이언트를 관리한다.

2.2.1 멤버 변수

Ball ball, last_send_ball	공
bool isLeftPressed, isRightPressed	좌우 눌린 여부

int GamePlay	게임 상태. 이넘 사용
vector <Block> block[15]	플레이 하고 있는 맵
vector <CrashedBk> crash	충돌 처리할 블록들
floatRECT ballrc	공의 실수형 RECT
Block list[24]	블록들의 리스트
array<array<char, 25>, 15> Map	송수신 위한 맵
int starcnt	별 개수
bool isSwitchOff	전기 스위치가 켜졌냐 꺼졌냐
int Scheck	출력할 사운드
int stage	플레이하고 있는 스테이지

int id	클라이언트 아이디
char name[NAME_SIZE]	클라이언트 이름
SOCKET sock	소켓
ServerManager* serverManager	서버 매니저 포인터
int recv_remain	패킷 재조립 위한 변수
char save_buf[BUFSIZE * 2]	패킷 재조립 위한 변수

2.2.2 메서드

void Initialize();	ServerManager 의 멤버변수 초기설정
void GameInitialize();	게임 초기화
void CrashExamin();	블록과의 충돌 검사
void Crash(int dir, int i, int y);	충돌한 블록들 처리
int BlockQuality(const Block* block);	입력된 블록의 충돌검사 우선순위 반환
int MyIntersectRect(const floatRECT* ballrc, const floatRECT* blockrc);	RECT와 RECT가 교차하는지
int isCrashed(const floatRECT* ballrc, const floatRECT* blockrc);	어느 방향에서 충돌했는지 반환
Block* Search(const int type);	순간이동할 화이트홀 검색
bool CrashBottom();	창 아래 충돌검사
void CrashBasicRight(const Block* block);	블록 오른쪽에서 충돌했을 때 기본 처리
void CrashBasicLeft(const Block* block);	블록 왼쪽에서 충돌했을 때 기본 처리

void CrashBasicBottom(const Block* block);	블록 아래에서 충돌했을 때 기본 처리
void CrashBasicTop(const Block* block);	블록 위에서 충돌했을 때 기본 처리
void MoveBall();	공 이동
void ClearVector();	맵 벡터 Clear
void MakeVector();	맵 행렬형식에서 벡터 형식으로 변환

DWORD WINAPI Do_Recv(LPVOID lpParam);	클라이언트에서 보내는 정보를 recv
void AddPacketToQueue (shared_ptr<PACKET> packet);	보낼 패킷을 큐에 담기
void Send_login_info_packet(Session* client);	해당 패킷에 값을 넣고 보내는 함수
void Send_load_map_packet(Session* client);	``
void Send_logout_packet(Session* client);	``
void Send_game_state_packet(Session* client);	``
void Send_sound_state_packet(Session* client);	``
void Send_load_custom_map_list_packet(string mapList);	``
bool CustomMapSave(char*);	커스텀 맵 저장

2.3 ServerManager 클래스

서버 전체 관리

2.3.1 멤버 변수

SOCKET s_sock, c_sock	소켓 저장
concurrency::concurrent_priority_queue <shared_ptr<PACKET>> sendPacketQ	서버에서 클라이언트에게 전송할 패킷들을 담아두는 큐
CRITICAL_SECTION cs	임계 구역(Critical Section)을 정의
int retval	받은 바이트
struct sockaddr_in clientaddr	클라이언트 주소
int addrlen	주소 길이
array<Session, MAX_USER> clients	각 클라이언트별로 관리하는 Session 배열
array<POINT, MAX_USER> ballStartPos	공들 시작 위치
bool isSwitchOff	스위치 상태 저장
array<array<char, 25>, 15> Map	맵 저장해놓을 변수

bool isWaiting[4]	대기중인 클라이언트 대열

2.3.2 메서드

void S_Bind_Listen();	bind(), listen()
void S_Accept()	accept()
void MakeThreads();	클라이언트가 접속하면 해당 클라에서 보내는 데이터를 recv 하기 위한 스레드 호출
void MakeSendThreads();	send() 스레드 호출
void MakeTimerThreads();	타이머 스레드 호출
void Do_timer();	프레임 당 1번 전송하기 위해 timer를 돌리고, 일정 시간이 되면 send 호출
void Disconnect(int c_id);	연결 끊기
void ProcessPacket(int c_id, char* packet);	recv한 정보들을 받아 재조립 후 분류, 필요한 값 업데이트 후 send
void Do_Send(const std::shared_ptr<PACKET>& packet);	클라이언트에 패킷을 전송
void ProcessSendQueue();	큐에 담긴 패킷을 하나씩 send()
void Send_frame_packet();	해당 패킷에 값을 넣고 보내는 함수
void Send_death_packet(int deathID);	``
void Send_edit_map_packet(Block* block, int i, int y);	``
void EnterTheStage(Session& client, int stageNum);	스테이지 대기, 입장, 초기화 관리
void MapLoad(int mapNumber);	스테이지 맵 로드
void MapLoad(int c_id, char* mapName);	커스텀 맵 로드
void MapListLoad(int c_id);	커스텀 맵 이 목록 로드

3. 클라이언트 low-level 디자인

3.1 전역

3.1.1 constexpr 상수

ClientManager game	클라이언트 매니저 클래스

queue<pair<KEY_TYPE, KEY_STATE>> keyEventQueue	키 입력 이벤트 큐
queue<KEY_TYPE> mouseEventQueue	마우스 입력 이벤트 큐

3.1.2 함수

LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)	윈도우 프로시저
DWORD WINAPI gameSend(LPVOID arg);	send() 스레드
DWORD WINAPI gameReceive(LPVOID arg);	recv() 스레드
LRESULT CALLBACK DialogProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);	대화상자 프로시저

void LoadResources();	리소스 로드 함수
void Update();	업데이트 함수
void Render();	렌더 함수
void ProcessInput();	입력 처리 함수
void SendKeyPackets();	키 패킷 전송 함수

3.2 객체 클래스

3.2.1 Ball

공과 관련된 정보를 담는다.

u_short playerID	공을 조종하고 있는 플레이어 아이디
float x, y	위치 정보
bool isDead	공 상태

3.2.2 Particles

파티클 애니메이션과 관련된 정보를 담는다.

int x, y	위치 정보 (열, 행)
int type, subtype	애니메이션의 타입
int ani	타이머 변수

3.3 ClientManager 클래스

게임에 필요한 객체들을 제어한다.

3.3.1 멤버 변수

RECT window	창 크기
Ball ball, otherPlayer	공
POINT ballStartPos	공 시작 위치
bool isSwitchOff	전기 스위치가 켜졌냐 꺼졌냐
vector <Particles> animation	파티클
Block list[24]	블록들의 리스트
int Map[15][25]	커스텀 모드에서 편집할 때 출력될 맵
int GamePlay	게임 상태. 이넘 사용
int starcnt	별 개수
int Scheck	출력할 사운드

int retval	받은 바이트
int recv_remain	패킷 재조립 위한 변수
char save_buf[BUFSIZE * 2]	패킷 재조립 위한 변수
SOCKET clientSocket	소켓
CRITICAL_SECTION packetQueueCS	임계영역
HANDLE hThreadForSend	send() 스레드 핸들
HANDLE hThreadForReceive	recv() 스레드 핸들
bool isConnected	연결 확인 변수
array<string, NAME_SIZE> customList	커스텀 맵 이름 리스트

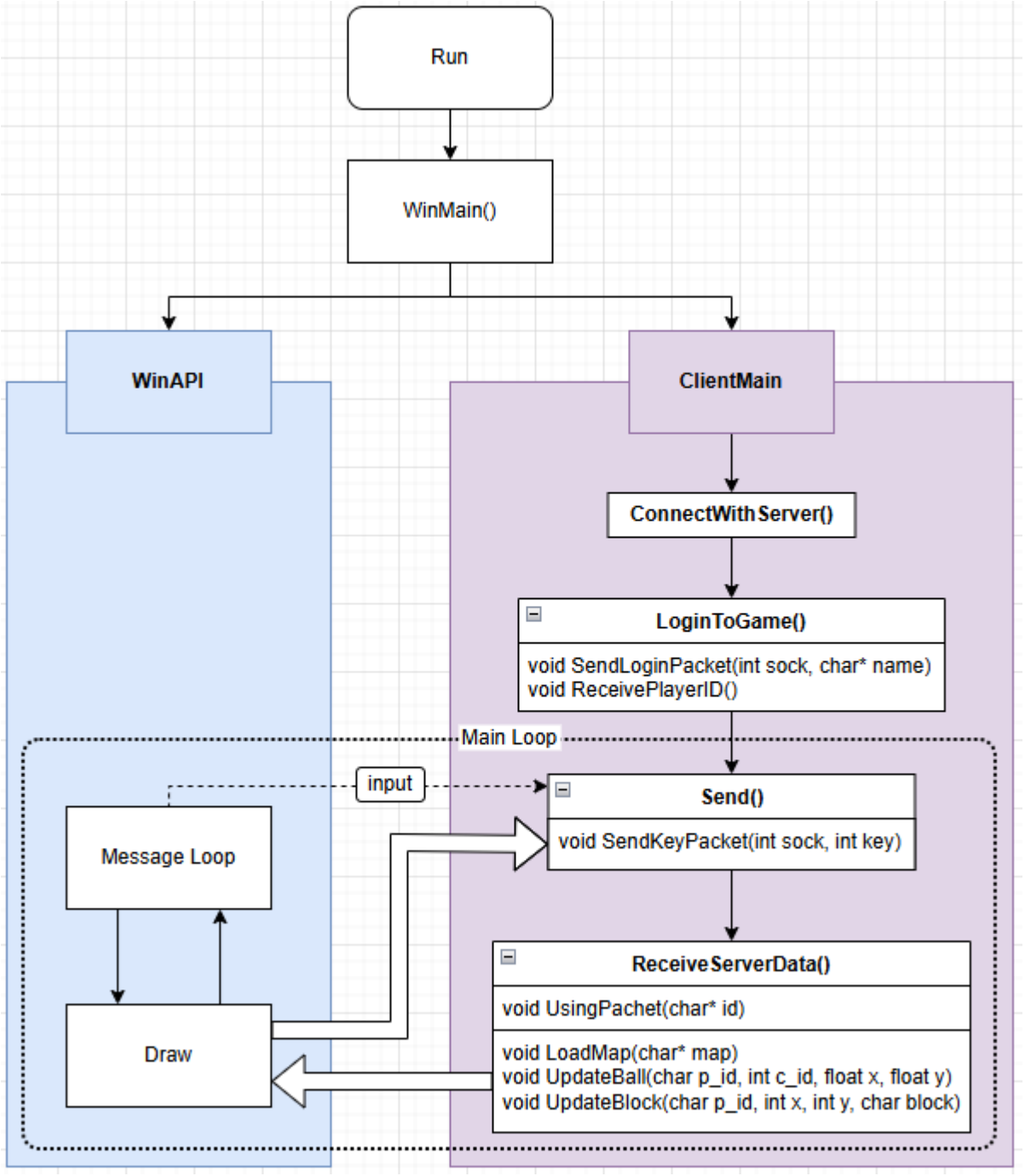
3.3.2 메서드

void Initialize();	ClientManager 의 멤버변수 초기설정
void Destroy();	Client, 스레드 종료 함수

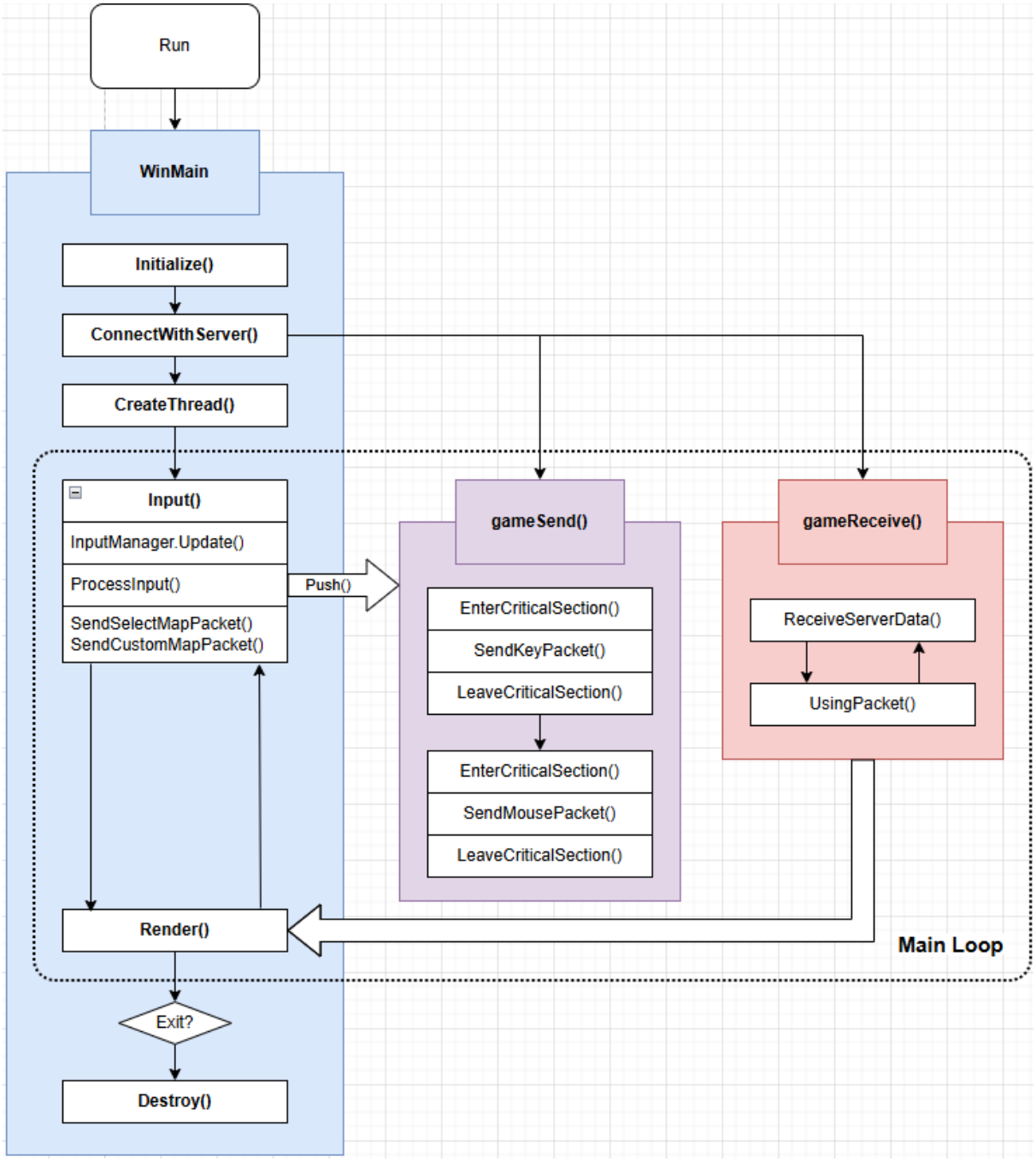
void ConnectWithServer();	윈속 초기화 후에 서버와의 통신을 위한 소켓 생성 후 연결 요청
void LoginToGame()	SendPlayerInfo로 서버에게 정보 전달 후 ReceivePlayerID 함수로 id를 받음
void SendLoginPacket(int sock, char* name)	해당 패킷에 값을 넣고 보내는 함수
void SendKeyPacket(int sock, int key)	``

bool SendMousePacket(KEY_TYPE key, POINT mousePos);	``
bool SendCustomMapPacket(POINT startPos, char* name);	``
bool SendSelectMapPacket(int idx);	``
void ReceivePlayerID()	서버로부터 클라이언트 전용 id를 전송받음
void ReceiveServerData();	소켓을 통해 서버로부터 데이터들을 받음
void UsingPacket(char* buffer);	받은 데이터들을 패킷ID에 따라 처리

- 수정 전


Run
WinMain()
WinAPI
ClientMain
ConnectWithServer()
LoginToGame()
void SendLoginPacket(int sock, char* name)
void ReceivePlayerID()
Main Loop
input
Send()
void SendKeyPacket(int sock, int key)
Message Loop
ReceiveServerData()
void UsingPachet(char* id)
Draw
void LoadMap(char* map)
void UpdateBall(char p_id, int c_id, float x, float y)
void UpdateBlock(char p_id, int x, int y, char block)

- 수정 후



4. 역할분담

- **김유빈** : 서버 기본 프레임워크, 스테이지 모드
- **이우진** : 서바이벌 모드 관련 클라 패킷 송수신 담당
- **하승민** : 커스터마이징 모드 클라이언트 수정, 패킷 송수신

5. 개발환경

Visual Studio 2022, Github, Notion, diagrams



6. 최종 개발 일정

- 김유빈

일	월	화	수	목	금	토
11/3	11/4	11/5	11/6	11/7	11/8	11/9
~~ServerManager 클래스 작성(선언)~~	~~Session 클래스 작성(선언)~~	~~ServerManager 멤버함수 구현 void S_Bind_Listen(); void S_Accept() void Disconnect(~);~~ main 코드 작성		~~ServerManager 멤버함수 구현 void MakeThreads();~~ Session 클래스 작성(선언) 필요 구조체 작성		~~클라이언트와 연결 확인~~
11/10	11/11	11/12	11/13	11/14	11/15	11/16
~~ServerManager 멤버함수 구현 void ProcessPacket(~);~~	~~Session 멤버함수 구현 void Do_Recv();~~	~~Session 멤버함수 구현 void Do_Recv();~~ ServerManager 클래스 작성(선언)	~~Session 멤버함수 구현 void Do_Send();~~ ServerManager 멤버함수 구현 void MakeThreads();	ServerManager 멤버함수 구현 void Do_Send(~) void AddPacketToQueue(~);	클라이언트와 송수신 점검	클라이언트와 연결 확인
11/17	11/18	11/19	11/20	11/21	11/22	11/23
	~~로그인, 동시 접속 확인~~ Send Thread 관련 함수 선언 및 정의	Session 멤버함수 구현 게임 필요 연산 함수들	Session 멤버함수 구현 Send_load_map_packet(~);	Session 멤버함수 구현 Send_frame_packet(~);		Session 멤버함수 구현 Send_edit_map_packet(~);
11/24	11/25	11/26	11/27	11/28	11/29	11/30
서로 공 잘 움직이는지 확인 ServerManager 멤버함수 구현 void Do_timer();	서로 공 잘 움직이는지 확인		맵 로드		~~ProcessPacket SC_DEATH_PACKET 전송 부분 구현~~	
12/1	12/2	12/3	12/4	12/5	12/6	12/7
ProcessPacket SC_DEATH_PACKET 전송 부분 구현 서버 주소 인수로 받도록 수정			커스터마이징 모드 맵 저장 구현	커스터마이징 모드 맵 불러오기 구현		오류 점검
12/8	12/9	12/10	12/11	12/12		
오류 점검		Process Report 작성	Process Report 작성	최종 발표		

- 이우진

일	월	화	수	목	금	토
11/3	11/4	11/5	11/6	11/7	11/8	11/9
클라이언트 로직 수정				ClientManager 멤버함수 ConnectWithServer 함수 구현	ClientManager 멤버함수 ConnectWithServer 함수 구현	서버와 연결 확인
11/10	11/11	11/12	11/13	11/14	11/15	11/16
동시 접속 확인		~~ClientManager~~ ~~멤버함수~~ ~~SendLoginPacket~~ ~~함수 구현~~ ClientMain 쓰레드 생성		~~ClientManager~~ ~~멤버함수~~ ~~ReceivePlayerID~~ ~~함수 구현~~ ClientManager 멤버함수 SendKeyPacket 함수 구현		
11/17	11/18	11/19	11/20	11/21	11/22	11/23
~~Session 멤버함수 구현~~ ~~void Send_login_info_packet(~);~~ render 잘 되는지 확인			ClientMain 쓰레드와 메인 쓰레드 동기화 확인			동기화 테스트
11/24	11/25	11/26	11/27	11/28	11/29	11/30
~~survival 모드 패킷 정의~~ 서버와 패킷 송수신 동기화테스트	서로 공 잘 움직이는지 확인		스테이지 모드에서 패킷 송수신 확인		ClientManager 멤버함수 SendLoginPacket 함수 구현	
12/1	12/2	12/3	12/4	12/5	12/6	12/7
ClientManager 멤버함수 SendLoginPacket 함수 구현		ClientManager 멤버함수 ReceivePlayerID 함수 구현		Session 멤버함수 구현 void Send_login_info_packet(~);	서버와 송수신 점검	
12/8	12/9	12/10	12/11	12/12		
오류 점검		Process Report 작성	Process Report 작성	최종 발표		

- **하승민**

일	월	화	수	목	금	토
11/3	11/4	11/5	11/6	11/7	11/8	11/9
클라이언트 로직 수정	클라이언트 로직 수정	~~클라이언트 로직 수정~~ 커스터마이징 리소스 제작, 수정	~~서바이벌 모드~~ ~~리소스 제작, 수정~~ 시작화면 리소스 제작, 수정		~~커스터마이징~~ ~~리소스 제작, 수정~~ 로비화면 리소스 제작, 수정	~~연결 확인~~
11/10	11/11	11/12	11/13	11/14	11/15	11/16
	스테이지 2인 게임에 맞게 맵 수정	~~로비 리소스 제작~~	~~로비 싼 구현~~		~~클라이언트, 서버~~ ~~송수신 점검~~	서버와 연결 확인
11/17	11/18	11/19	11/20	11/21	11/22	11/23
~~동시접속 확인~~		ReceiveServerData() 구현	UsingPacket() 틀 구현	~~LoadMap() 구현~~ UsingPacket() 틀 구현	~~LoadMap() 구현~~	~~UpdateBall()~~ 구현
11/24	11/25	11/26	11/27	11/28	11/29	11/30
GameManager() 수정 서버와 패킷 송수신 동기화테스트	~~UpdateBall() 구현~~ Do_Timer() 스레드 생성 서로 공 잘 움직이는지 확인	UpdateBall() 구현	MapLoad() 구현 공 움직이는 로직 완성	~~커스터마이징 맵~~ ~~서버 멤로드 구현~~ 스테이지 모드 구현	~~서버에서 저장된 맵~~ ~~클라에서 목록 띄우기~~	동기화 테스트
12/1	12/2	12/3	12/4	12/5	12/6	12/7
			~~UpdateBlock()~~ 구현 맵 동기화 구현	~~UpdateBlock()~~ 구현	~~MakeReadyVector()~~ ~~수정~~ ~~커스터마이징 맵~~ ~~서버 멤로드 구현~~	동기화 테스트
12/8	12/9	12/10	12/11	12/12		
커스터마이징 모드 완성	오류 점검	Process Report 작성	Process Report 작성	최종 발표		