네트워크 게임 프로그래밍

추진 계획서

Term Project

CUPHEAD

2020184008 김민수

2020184021 우상훈

2021182047 박신혜

목차

<CUPHEAD>

어플리케이션 기획

1. 게임 소개

우상훈 학생이 2학년 1학기 윈도우 프로그래밍 과목에서 제작한 **CupHead**를 기획한다.

CupHead는 기존 존재하는 Cuphead의 모작으로서 횡스크롤 액션 어드벤처 게임이다.

플레이 방식은 플레이어가 보스의 공격을 피하면서 총을 쏴 보스를 처리하는 게임이다.

월드맵에서는 횡스크롤 기반으로 카메라가 이동할 예정이며 보스맵에서는 3인칭 측면뷰로 고정된다.

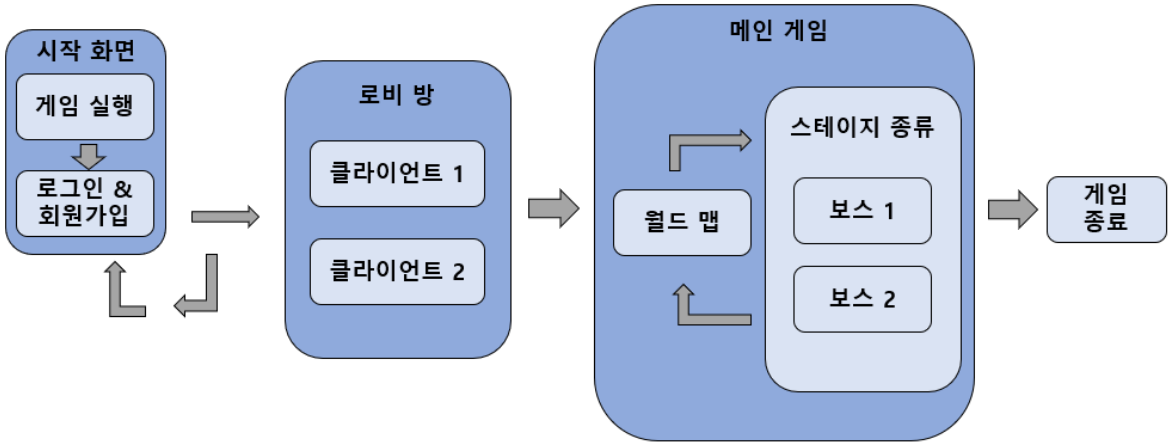
네트워크 게임으로서 **2인 멀티플레이 게임**으로 만들 예정이다.

구현되어 있는 내용은 보스 1, 타이틀 화면이 구현되어 있다.

서버 기능을 추가하기 위해 추가 구현할 클라이언트 내용은 보스2, 월드맵, 로그인, 회원가입이 있다.

cuphead	
장르	횡스크롤 액션 어드벤처 게임
최대 인원	2명
조작	키보드
시점	3인칭, 측면뷰
개발 언어	C++, winapi
플레이 화면 자료	
	

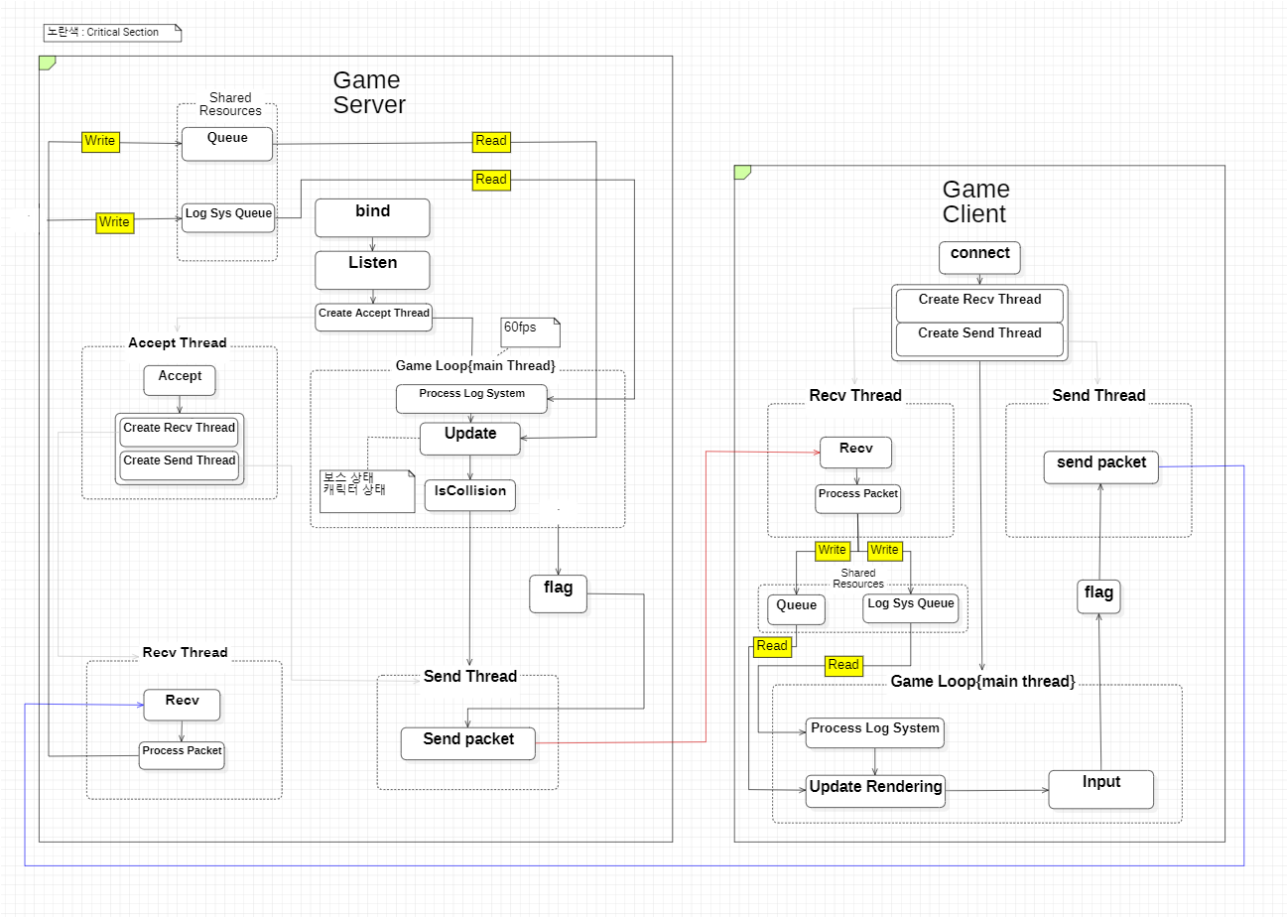
2. 게임 플레이



게임을 실행하면, 로그인과 회원가입을 할 수 있는 시작 화면이 나온다.

- 로그인을 했을 때 기존에 등록된 아이디가 아니면 로비로 넘어가지 못한다.
- 회원가입을 진행한 후 로그인을 해야 로비로 넘어갈 수 있다.
- 로비에서 클라이언트 2명이 모두 접속했을 때 게임 버튼이 활성화가 된다.
- 이전에 게임을 했던 기록이 있다면 선택하여 저장된 내용부터 진행이 되도록 한다.
- 이전에 게임을 했던 기록이 없다면 처음부터 시작하게 된다.
- 메인게임에 들어가게 된다면 월드 맵으로 시작하게 된다.
- 월드 맵에서 기존에 클리어 한 스테이지를 확인할 수 있으며 선택할 수 있다.
- 스테이지를 들어가게 된다면 보스가 진행되게 된다.
- 보스에게 기본공격, 특수스킬을 사용해 타수 형태로 데미지를 주게 된다.
- 보스를 처리하게 되면 월드맵으로 돌아가게 되며 다른 스테이지를 선택할 수 있다.
- 모든 스테이지를 클리어하게 되면 게임이 종료되게 된다.
- 중간에 클라이언트 1명이 튕긴 후 다시 들어오게 되면 저장된 정보 기준으로 계속 게임을 진행할 수 있다.

4. High-Level



우선, 서버가 각 클라이언트마다 recv 스레드를 갖지만, send 스레드는 하나만 갖는 구조로 결정했다. 이 구조는 서버 측의 스레드 수가 적어 자원 소모를 줄일 수 있다는 점과 send 스레드 안에서 각 클라이언트에게 데이터를 송신할 것이기 때문에, 서버 측 운영체제의 송신 버퍼 접근에 대한 동기화 작업을 수행하지 않아도 된다는 점을 장점으로 갖고 있다.

다음으로는 Game Loop(Main Thread)와 Accept Thread, Recv Thread, Send Thread 사이의 순서 제어의 필요성에 대한 생각이다. 첫 번째, Accept Thread는 단지 클라이언트의 접속 요청을 수락하고 해당 클라이언트의 정보를 저장하는 역할이기 때문에 독립적으로 수행돼도 괜찮을 것 같다고 생각한다. 두 번째, Recv Thread이다. Recv Thread는

계속해서 받은 데이터를 처리하고 공유 자원인 Queue 혹은 LogSysQueue에 넣어줘야 하기 때문에 순서에 제한받지 않아야 한다고 생각한다. 세 번째, Send Thread는 순서 제어가 필요하다고 생각한다. Game Loop에서 recv로 받은 데이터를 기반으로 게임 상태 갱신 및 충돌 처리 계산을 마치고 나서 전달할 데이터가 있을 경우 send를 해야 한다. 하지만 Send Thread가 순서 제어 없이 독립적으로 실행되고 있다면, 보낼 데이터가 없을 때에도 send를 하게 되므로, 클라이언트 측에 올바르지 않은 데이터가 전달될 가능성이 있다고 판단했다.

Send Thread의 순서 제어는 전역적으로 클라이언트들에게 보낼 데이터와 충돌 처리 결과 값을 각각 담고 있는 Interface Class와 보낼 데이터가 있는지 판단하기 위한 atomic<bool> 타입의 flag를 사용할 것이다. Game Loop의 update에서는 보낼 데이터를 Interface Class에 담고, isCollision에서는 충돌 처리 결과 값을 담는다. Game Loop가 한 번 끝나고 나서 데이터가 담겨 있으면 flag를 true로 만든다. Send Thread는 flag 값을 계속해서 지켜보다가 true가 되면 데이터를 보낸다. 데이터를 다 보내면 flag의 값을 다시 false로 바꾼다. flag 값을 변경하는 부분은 원자적으로 동작해야 하기 때문에, atomic 타입을 사용한 것이다. 하지만 이 부분에 대해서는 교수님의 의견도 들어보는 것이 좋을 것 같다고 생각한다. 클라이언트의 동작 방식은 서버와 마찬가지로 동일하다.

5. Low-Level

- 전송 패킷 타입

```
enum class LogSystem : unsigned char {
    SignUp,
    LogIn,
    LogOut,

    CompletedSignUp,
    CompletedLogIn,
    CompletedLogOut,

    FailedSignUp,
    FailedLogIn,
    FailedLogOut

    SaveLogfile
};
```

```
enum class ClientPacketType : unsigned char {
    Input
};

enum class ServerPacketType : unsigned char {
    PlayerStateUpdate,
    EnemyStateUpdate,
    CollisionEvent,
    DeathEvent
};
```

- 전송 패킷

클라 -> 서버

```
struct SignUpOrLogInPacket {
    LogSystem type;
    std::string username;
    std::string password;
};
```

```
struct LogOutPacket {
    LogSystem type;
    std::string username;
};

struct InputPacket{
    LogSystem type;
    unsigned char dir;
    unsigned char jump;
    unsigned char attack;
}
```

서버 -> 클라

```
struct ClientStatePacket {
    ServerPacketType type;
    short x, y;
    unsigned char hp;
};
```

```
struct EnemyStatePacket {
    ServerPacketType type;
    short x, y;
    unsigned char entityId;
    unsigned char hp;
};

struct CollisionEventPacket {
    ServerPacketType type;
    unsigned char entityId;
};

struct DeathEventPacket {
    ServerPacketType type;
    unsigned char entityId;
};

struct LoadSaveFile{
    ServerPacketType type;
    unsigned char stage;
};
```

2024.11.22 / 패킷 재설계

```
enum class PacketType {
	NONE,

	LOGIN,
	LOGIN_RESULT,
};

enum class Direction {
	NONE,
	E, W, S, N,
	NE, NW, SE, SW,
};

struct LoginPacket {   // client -> server
	char id[ 16 ];
	char pw[ 16 ];
};

struct LoginResultPacket {     // server -> client
	bool result;};
```

```
};

struct Packet {
	PacketType type;
	union /*PacketData*/ {
		LoginPacket lg;
		LoginResultPacket lr;
	};
};
```

2024.11.29 / 패킷 재설계

```
extern bool gCupheadLogin;
extern bool gMugmanLogin;

extern bool gImCuphead;

enum class PacketType {
	NONE,

	LOGIN,
	LOGIN_RESULT,
```

```
	LEAVE,


	MOVE,
	INPUT,
	TRY_GAME_START,
	LOGOUT,
	CHANGE_SCENE,
};


enum class Direction {
	NONE,


	E, W, S, N,
	NE, NW, SE, SW,
};


struct LoginPacket {	// client -> server
	char id[ 16 ];
	char pw[ 16 ];
};
```

```
struct LoginResultPacket {     // server -> client
        bool result;
        bool cupheadLogin;
        bool mugmanLogin;

        enum class Type {
                None,
                Cuphead,
                Mugman
        };
        Type who;
};

struct RegisterPacket {         // server -> client
        std::uint16_t id;

        /*
        특정 타입
        해당 타입을 초기화하는데 필요한 모든 인자
        */
        GROUP_TYPE groupType;
```

```
        Vec2 pos;
};

struct MovePacket {   // server -> client
        std::uint16_t id;
        Direction dir;
        Vec2 pos;
};

struct InputPacket {    // client -> server
        std::uint16_t id;
        bool left, right, up, down;
        Direction dir;
};

struct DestroyPacket {// server -> client
        std::uint16_t id;
};

// struct TryGameStartPacket {};
```

```
struct ChangeScenePacket {
	SCENE_TYPE scene;
};

struct LeavePacket {
	bool imCuphead;
};

struct Packet {
	PacketType type;
	union /*PacketData*/ {
		LoginPacket lg;
		LoginResultPacket lr;
		RegisterPacket rg;
		MovePacket mv;
		InputPacket in;
		DestroyPacket ds;
		LeavePacket lv;
		ChangeScenePacket cs;
	};
};
```

2024.12.06 / 패킷 재설계

```
extern bool gCupheadLogin;

extern bool gMugmanLogin;


extern bool gImCuphead;


enum class PacketType {
	NONE,


	LOGIN,
	LOGIN_RESULT,
	LEAVE,


	REPLICATION,
	REGISTER,
	DESTROY,
	MOVE,
	INPUT,
	TRY_GAME_START,
	LOGOUT,
	CHANGE_SCENE,
```

```
	CHANGE_SCENE_ACK,
	ANIMATION_RPC,
};

enum class Direction {
	NONE,

	E, W, S, N,
	NE, NW, SE, SW,
};

struct LoginPacket {	// client -> server
	char id[ 16 ];
	char pw[ 16 ];
};

struct LoginResultPacket {	// server -> client
	bool result;
	bool cupheadLogin;
	bool mugmanLogin;
```

```
	enum class Type {
		None,
		Cuphead,
		Mugman
	};
	Type who;
};

struct RegisterPacket {		// server -> client
	std::uint16_t id;

	/*
	특정 타입
	해당 타입을 초기화하는데 필요한 모든 인자
	*/
	GROUP_TYPE groupType;
	Vec2 pos;
};

struct MovePacket {	// server -> client
	std::uint16_t id;
```

```
        Direction dir;
        Vec2 pos;
};

struct InputPacket {    // client -> server
        std::uint16_t id;
        bool left, right, up, down;
        Direction dir;
};

// Animation Remote Procedure Call
struct AnimationRPC {           // server -> client
        enum class Type {
                IdleDown,
                IdleLeft,
                IdleLeftDown,
                IdleLeftUp,
                IdleRight,
                IdleRightDown,
                IdleRightUp,
                IdleUp,
```

```
                WalkDown,
                WalkLeft,
                WalkLeftDown,
                WalkLeftUp,
                WalkRight,
                WalkRightDown,
                WalkRightUp,
                WalkUp,
        };

        std::uint16_t id;
        Type anim;
};

// struct TryGameStartPacket {};

struct ChangeScenePacket {
        SCENE_TYPE scene;
};

struct ChangeSceneAckPacket {
```

```
	bool imCuphead;
};

struct LeavePacket {
	bool imCuphead;
};

struct LogoutPacket {
	bool imCuphead;
};

struct Packet {
	PacketType type;

	union /*PacketData*/ {
		LoginPacket lg;
		LoginResultPacket lr;
		RegisterPacket rg;
		MovePacket mv;
		InputPacket in;
		LeavePacket lv;
```

```
            LogoutPacket lo;
            ChangeScenePacket cs;
            ChangeSceneAckPacket ca;
            AnimationRPC ar;
        };
};
```

2024.12.12 / 패킷 마지막 설계

```
extern bool gCupheadLogin;
extern bool gMugmanLogin;

extern bool gImCuphead;

enum class PacketType {
        NONE,

        LOGIN,
        LOGIN_RESULT,
```

```
	LEAVE,


	REPLICATION,
	REGISTER,
	DESTROY,
	MOVE,
	INPUT,
	TRY_GAME_START,
	LOGOUT,
	CHANGE_SCENE,
	CHANGE_SCENE_ACK,
	ANIMATION_RPC,
};


enum class Direction {
	NONE,


	E, W, S, N,
	NE, NW, SE, SW,
};
```

```
struct LoginPacket {    // client -> server
        char id[ 16 ];
        char pw[ 16 ];
};

struct LoginResultPacket {      // server -> client
        bool result;
        bool cupheadLogin;
        bool mugmanLogin;

        enum class Type {
                None,
                Cuphead,
                Mugman
        };
        Type who;
};

struct RegisterPacket {         // server -> client
        std::uint16_t id;
```

```
	/*
	특정 타입
	해당 타입을 초기화하는데 필요한 모든 인자
	*/
	GROUP_TYPE groupType;
	Vec2 pos;
};

struct MovePacket {   // server -> client
	std::uint16_t id;
	Direction dir;
	Vec2 pos;
};

struct InputPacket {   // client -> server
	std::uint16_t id;
	bool left, right, up, down;
	Direction dir;
};

// Animation Remote Procedure Call
```

```
struct AnimationRPC {           // server -> client
        enum class Type {
                IdleDown,
                IdleLeft,
                IdleLeftDown,
                IdleLeftUp,
                IdleRight,
                IdleRightDown,
                IdleRightUp,
                IdleUp,
                WalkDown,
                WalkLeft,
                WalkLeftDown,
                WalkLeftUp,
                WalkRight,
                WalkRightDown,
                WalkRightUp,
                WalkUp,
        };

        std::uint16_t id;
```

```
	Type anim;
};

struct DestroyPacket {// server -> client
	std::uint16_t id;
};

// struct TryGameStartPacket {};

struct ChangeScenePacket {
	SCENE_TYPE scene;
};

struct ChangeSceneAckPacket {
	bool imCuphead;
};

struct LeavePacket {
	bool imCuphead;
};
```

```
struct LogoutPacket {
	bool imCuphead;
};

struct ReplicationPacket {	// server -> client
	std::uint16_t id;

	/*
	특정 타입
	해당 타입을 초기화하는데 필요한 모든 인자
	*/
	GROUP_TYPE groupType;
	Vec2 pos;
};

struct Packet {
	PacketType type;

	union /*PacketData*/ {
		LoginPacket lg;
		LoginResultPacket lr;
```

```
        RegisterPacket rg;
        MovePacket mv;
        InputPacket in;
        DestroyPacket ds;
        LeavePacket lv;
        LogoutPacket lo;
        ChangeScenePacket cs;
        ChangeSceneAckPacket ca;
        ReplicationPacket rp;
        AnimationRPC ar;
    };
};
```

5-1. 서버

- 함수

void severSend(void* buf, int bufSize) -> Send Thread

- 서버가 GameLoop에서 관리하는 모든 업데이트 데이터를 클라이언트에게 보내준다.

void serverRecv(LPVOID sockInfo) -> Recv Thread

- 클라이언트에서 전달한 input값들을 받는다.

void acceptClient(SOCKET sock, sockaddr* addr, int* addrLen) -> Accept Thread

- 클라이언트의 접속 요청을 받는다.

void processPacket(void* buf)

- serverRecv()로 받은 데이터들을 해석해서 순차적으로 Queue에 저장한다.
  만약 해석된 데이터 중 log와 관련된 데이터는 Log Sys Queue에 저장한다.

void processLogSystem()

- Log Sys Queue에서 가져온 정보를 이용해 회원가입, 로그인 성공, 실패 여부를 반환한다.

void update()

- Queue에서 받아온 패킷을 토대로 게임 상태를 업데이트한다.

bool isCollision(int id1, int id2)

- 업데이트된 상태를 가지고 충돌 여부를 확인한다.

5-2. 클라이언트

- 함수

void clientSend(void* buf, int bufSize) -> Send Thread

- 클라이언트는 회원가입 요청, 로그인 요청, 입력된 키값을 보낸다.

void clientRecv(LPVOID sockInfo) -> Recv Thread

- 서버에서 계산된 게임 상태에 대한 데이터를 받는다.

void processPacket(void* buf)

- clientRecv()로 받은 데이터들을 해석해서 순차적으로 Queue에 저장한다.

  만약 해석된 데이터 중 log와 관련된 데이터는 Log Sys Queue에 저장한다.

void processLogSystem()

- Log Sys Queue에서 회원가입 요청, 로그인 요청 한것에 대해서 성공 실패 여부를 받는다.

void update() / void render(HDC hdc)

- Queue에서 꺼낸 패킷들을 가지고 게임 상태를 렌더링한다.

void input(int keyType)

- 키 입력을 처리한다.

6. 팀원별 역할 분담

Server – acceptClient **우상훈**

Server – severSend **박신혜**

Server – serverRecv **박신혜**

Server – processPacket **박신혜, 우상훈**

Server – processLogSystem **박신혜**

Server – update **우상훈**

Server – isCollision **우상훈**

Client – clientSend **김민수**

Client – clientRecv **김민수**

Client – processPacket **우상훈, 김민수**

Client – processLogSystem **김민수**

Client – update **우상훈, 박신혜**

Client – render **우상훈, 김민수**

Client – input **박신혜**

- 밀린 개발 일정을 급하게 채우는 바람에 함수와 역할 분담 부분을 제때 갱신하지 못하였습니다. 죄송합니다.

7. 개발 일정

<table>
<tr><th></th><th colspan="2">11월7일 8일</th><th>9일</th><th>10일</th><th>11일</th><th>12일</th><th>13일</th></tr>
<tr><td>우상훈</td><td colspan="3" rowspan="4">클라이언트 수정</td><td colspan="2">Accept 스레드</td><td rowspan="4">시험 공부</td><td>Server Update</td></tr>
<tr><td>김민수</td><td colspan="2">Client Send 스레드</td><td>Client Recv 스레드</td></tr>
<tr><td>박신혜</td><td colspan="2">Server Send 스레드</td><td>Client Recv 스레드</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td></tr>
</table>

<table>
<tr><td></td><td>14일</td><td>15일</td><td>16일</td><td>17일</td><td>18일</td><td>19일</td><td>20일</td></tr>
<tr><td>우상훈</td><td>Server Update</td><td colspan="2">Server Process Packet
Client Process Packet</td><td rowspan="2">패킷
재설계</td><td rowspan="4">디버깅
및
수정사항
체크</td><td colspan="2" rowspan="2">클라이언트 개발
Client-render
(로그 화면)</td></tr>
<tr><td>김민수</td><td>Client Recv 스레드</td><td colspan="2">Client Process Packet</td></tr>
<tr><td>박신혜</td><td>Client Recv 스레드</td><td colspan="2">Server Process Packet</td><td>Input Client</td><td colspan="2">processLogSystem Server</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>21일</td><td>22일</td><td>23일</td><td>24일</td><td>25일</td><td>26일</td><td>27일</td></tr>
<tr><td>우상훈</td><td colspan="2">Client-update</td><td rowspan="3">패킷
재설계</td><td rowspan="3">디버깅
및
수정사항
체크</td><td>isCollision</td><td>Client_ update</td><td rowspan="3">클라 개발
(보스 1)</td></tr>
<tr><td>김민수</td><td colspan="2">ProcessLogSystem Client</td><td colspan="2">Client_render</td></tr>
<tr><td>박신혜</td><td colspan="2">Client-update</td><td colspan="2">Client_render</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>28일</td><td>29일</td><td>30일</td><td>12월1일</td><td>2일</td><td>3일</td><td>4일</td></tr>
<tr><td>우상훈</td><td rowspan="3">클라 개발
(보스 1)</td><td rowspan="3">패킷
재설계</td><td rowspan="3">디버깅
및
수정사항
체크</td><td colspan="2">Server_update</td><td rowspan="3">디버깅
및
수정사항
체크</td><td rowspan="3">최종 발표 준비</td></tr>
<tr><td>김민수</td><td colspan="2">재연결 클라</td></tr>
<tr><td>박신혜</td><td colspan="2">재연결 서버</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5일</td><td>6일</td><td>7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td></tr>
<tr><td>우상훈</td><td rowspan="3">최종 발표 준비</td><td rowspan="3">최종 발표</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>김민수</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>박신혜</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>12일</td><td colspan="6" rowspan="3"></td></tr>
<tr><td>우상훈</td><td></td></tr>
<tr><td>김민수</td><td></td></tr>
</table>

박신혜		
리뷰		

2024.11.15 / 구현 실패

	11월7일	**8일**	**9일**	**10일**	**11일**	**12일**	**13일**
우상훈	클라이언트 수정			Accept 스레드		시험 공부	Server Update
김민수				Client Send 스레드			Client Recv 스레드
박신혜				Server Send 스레드			Client Recv 스레드
리뷰	지속적인 수정이 필요하다.			기간 안에 수행하지 못함			역할이 잘못 입력됨
	14일	**15일**	**16일**	**17일**	**18일**	**19일**	**20일**
우상훈	Server Update	AcceptThread ServerUpdate Server Process Packet Client Process Packet		패킷 재설계	**디버깅 및 수정사항 체크**	클라이언트 개발 Client-render (로그 화면)	
김민수	Client Recv 스레드	ClientSend Thread Client Process Packet					
박신혜	Client Recv 스레드	serverRecv Thread Server Process Packet		Input Client		processLogSystem Server	
리뷰	기간 안에 수행하지 못함	기간 안에 수행하지 못함					

<table>
<tr><td></td><td>21일</td><td>22일</td><td>23일</td><td>24일</td><td>25일</td><td>26일</td><td>27일</td></tr>
<tr><td>우상훈</td><td colspan="2">Client-update</td><td rowspan="3">패킷
재설계</td><td rowspan="5">디버깅
및
수정사항
체크</td><td>isCollision</td><td>Client_
update</td><td rowspan="3">클라
개발
(보스 1)</td></tr>
<tr><td>김민수</td><td colspan="2">ProcessLogSystem
Client</td><td colspan="2">Client_render</td></tr>
<tr><td>박신혜</td><td colspan="2">Client-update</td><td colspan="2">Client_render</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>28일</td><td>29일</td><td>30일</td><td>12월1일</td><td>2일</td><td>3일</td><td>4일</td></tr>
<tr><td>우상훈</td><td rowspan="3">클라
개발
(보스 1)</td><td rowspan="3">패킷
재설계</td><td rowspan="3">디버깅
및
수정사항
체크</td><td colspan="2">Server_update</td><td rowspan="3">디버깅
및
수정사항
체크</td><td rowspan="3">최종
발표
준비</td></tr>
<tr><td>김민수</td><td colspan="2">재연결 클라</td></tr>
<tr><td>박신혜</td><td colspan="2">재연결 서버</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5일</td><td>6일</td><td>7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td></tr>
<tr><td>우상훈</td><td rowspan="3">최종
발표
준비</td><td rowspan="3">최종
발표</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>김민수</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>박신혜</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>12일</td><td colspan="6" rowspan="5"></td></tr>
<tr><td>우상훈</td><td></td></tr>
<tr><td>김민수</td><td></td></tr>
<tr><td>박신혜</td><td></td></tr>
<tr><td>리뷰</td><td></td></tr>
</table>

2024.11.22 / 파란색: 지난 주 일정, 보라색: 현재 일정

<table>
<tr><td></td><td>11월7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td><td>12일</td><td>13일</td></tr>
<tr><td>우상훈</td><td colspan="3" rowspan="3">클라이언트 수정</td><td colspan="2">Accept 스레드</td><td rowspan="3">시험 공부</td><td>Server Update</td></tr>
<tr><td>김민수</td><td colspan="2">Client Send 스레드</td><td>Client Recv 스레드</td></tr>
<tr><td>박신혜</td><td colspan="2">Server Send 스레드</td><td>Client Recv 스레드</td></tr>
<tr><td>리뷰</td><td colspan="3">지속적인 수정이 필요하다.</td><td colspan="2">기간 안에 수행하지 못함
-> 구현 완료</td><td></td><td>기간 안에 수행하지 못함
-> 구현 완료</td></tr>
<tr><td></td><td>14일</td><td>15일</td><td>16일</td><td>17일</td><td>18일</td><td>19일</td><td>20일</td></tr>
<tr><td>우상훈</td><td>Server Update</td><td colspan="2">AcceptThread
ServerUpdate
Server Process Packet
Client Process Packet</td><td rowspan="3">패킷
재설계

Input Client</td><td rowspan="3">디버깅 및 수정사항 체크</td><td colspan="2" rowspan="3">클라이언트 개발
Client-render
(로그 화면)

processLogSystem
Server</td></tr>
<tr><td>김민수</td><td>Client Recv 스레드</td><td colspan="2">Client Recv 스레드
Client Process Packet</td></tr>
<tr><td>박신혜</td><td>Client Recv 스레드</td><td colspan="2">Server Recv 스레드
Server Process Packet</td></tr>
<tr><td>리뷰</td><td>기간 안에 수행하지 못함
-> 구현 완료</td><td colspan="2">기간 안에 수행하지 못함
-> 구현 완료</td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td>Client의 leave 패킷을 서버가 받지 못함.
-> 주고받는 데이터의 자료형이 달랐던 문제가 있었음.</td><td colspan="2">과제에서 사용했던 DialogBox로 로그인을 구현함.</td></tr>
</table>

<table>
<tr><td></td><td>21일</td><td>22일</td><td>23일</td><td>24일</td><td>25일</td><td>26일</td><td>27일</td></tr>
<tr><td>우상훈</td><td colspan="2" rowspan="2">Client-update</td><td rowspan="3">패킷
재설계</td><td rowspan="4">디버깅
및
수정사항
체크</td><td>isCollision</td><td>Client_
update</td><td rowspan="3">클라
개발
(보스 1)</td></tr>
<tr><td>김민수</td><td colspan="2" rowspan="2">ProcessLogSystem
Client</td><td colspan="2" rowspan="2">Client_render</td></tr>
<tr><td>박신혜</td><td colspan="2">Client-update</td></tr>
<tr><td>리뷰</td><td colspan="2"></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>28일</td><td>29일</td><td>30일</td><td>12월1일</td><td>2일</td><td>3일</td><td>4일</td></tr>
<tr><td>우상훈</td><td rowspan="3">클라
개발
(보스 1)</td><td rowspan="3">패킷
재설계</td><td rowspan="3">디버깅
및
수정사항
체크</td><td colspan="2">Server_update</td><td rowspan="3">디버깅
및
수정사항
체크</td><td rowspan="3">최종
발표
준비</td></tr>
<tr><td>김민수</td><td colspan="2">재연결 클라</td></tr>
<tr><td>박신혜</td><td colspan="2">재연결 서버</td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5일</td><td>6일</td><td>7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td></tr>
<tr><td>우상훈</td><td rowspan="3">최종
발표
준비</td><td rowspan="3">최종
발표</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>김민수</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>박신혜</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>12일</td><td colspan="6" rowspan="5"></td></tr>
<tr><td>우상훈</td><td></td></tr>
<tr><td>김민수</td><td></td></tr>
<tr><td>박신혜</td><td></td></tr>
<tr><td>리뷰</td><td></td></tr>
</table>

2024.11.29

	11월7일	8일	9일	10일	11일	12일	13일
우상훈	클라이언트 수정			Accept 스레드		시험 공부	Server Update
김민수				Client Send 스레드			Client Recv 스레드
박신혜				Server Send 스레드			Client Recv 스레드
리뷰	지속적인 수정이 필요하다.			기간 안에 수행하지 못함 -> 구현 완료			기간 안에 수행하지 못함 -> 구현 완료

	14일	15일	16일	17일	18일	19일	20일
우상훈	Server Update	AcceptThread ServerUpdate Server Process Packet Client Process Packet		패킷 재설계	**디버깅 및 수정사항 체크**	클라이언트 개발 Client-render (로그 화면)	
김민수	Client Recv 스레드	Client Recv 스레드 Client Process Packet					
박신혜	Client Recv 스레드	Server Recv 스레드 Server Process Packet		Input Client		processLogSystem Server	
리뷰	기간 안에 수행하지 못함 -> 구현 완료	기간 안에 수행하지 못함 -> 구현 완료		패킷의 지속적인 수정 및 추가가 필요함.	Client의 leave 패킷을 서버가 받지 못함. -> 주고받는 데이터의 자료형이 달랐던 문제가 있었음.	과제에서 사용했던 DialogBox로 로그인을 구현함.	

<table>
<tr><td></td><td>21일</td><td>22일</td><td>23일</td><td>24일</td><td>25일</td><td>26일</td><td>27일</td></tr>
<tr><td>우상훈</td><td colspan="2" rowspan="2">Client-update</td><td rowspan="3">패킷
재설계</td><td rowspan="3">디버깅
및
수정사항
체크</td><td><s>isCollision</s></td><td>Client_
update</td><td rowspan="3">클라
개발
(보스 1)</td></tr>
<tr><td>김민수</td><td colspan="2">ProcessLogSystem
Client</td><td colspan="2" rowspan="2">Client_render</td></tr>
<tr><td>박신혜</td><td colspan="2">Client-update</td></tr>
<tr><td>리뷰</td><td colspan="2"></td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td colspan="3">클라이언트에 있는 object들의 id에 대응되는 object를 서버에도 등록만 하게 되면 input 등 client의 입력 처리가 가능해짐. 게임에는 여러 scene들이 존재하기 때문에 하나의 충돌 처리 함수로 처리하기 힘들 것 같다고 판단했다. 충돌 처리를 각 scene에게 맡기는 방식으로 구현하기로 정했다.</td></tr>
<tr><td></td><td>28일</td><td>29일</td><td>30일</td><td>12월1일</td><td>2일</td><td>3일</td><td>4일</td></tr>
<tr><td>우상훈</td><td rowspan="3">클라
개발
(보스 1)</td><td rowspan="3">패킷
재설계</td><td rowspan="3">디버깅
및
수정사항
체크</td><td colspan="2">Server_update</td><td rowspan="3">디버깅
및
수정사항
체크</td><td rowspan="3">최종
발표
준비</td></tr>
<tr><td>김민수</td><td colspan="2">재연결 클라</td></tr>
<tr><td>박신혜</td><td colspan="2">재연결 서버</td></tr>
<tr><td>리뷰</td><td>클라이언트 보스를 구현하지 못함.</td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td></td><td colspan="2"></td><td></td><td></td></tr>
<tr><td></td><td>5일</td><td>6일</td><td>7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td></tr>
<tr><td>우상훈</td><td rowspan="3">최종
발표
준비</td><td rowspan="3">최종
발표</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>김민수</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>박신혜</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>리뷰</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>12일</td><td colspan="6" rowspan="3"></td></tr>
<tr><td>우상훈</td><td></td></tr>
<tr><td>김민수</td><td></td></tr>
</table>

박신혜		
리뷰		

2024.12.06

	11월7일	**8일**	**9일**	**10일**	**11일**	**12일**	**13일**
우상훈	클라이언트 수정			Accept 스레드		시험 공부	Server Update
김민수				Client Send 스레드			Client Recv 스레드
박신혜				Server Send 스레드			Client Recv 스레드
리뷰	지속적인 수정이 필요하다.			기간 안에 수행하지 못함 -> 구현 완료			기간 안에 수행하지 못함 -> 구현 완료

<table>
<tr><td></td><td>14일</td><td colspan="2">15일</td><td>16일</td><td>17일</td><td>18일</td><td>19일</td><td>20일</td></tr>
<tr><td>우상훈</td><td>Server Update</td><td colspan="2">AcceptThread
ServerUpdate
Server Process Packet
Client Process Packet</td><td rowspan="2">패킷
재설계</td><td rowspan="3">디버깅 및 수정사항 체크</td><td colspan="2" rowspan="2">클라이언트 개발
Client-render
(로그 화면)</td></tr>
<tr><td>김민수</td><td>Client Recv 스레드</td><td colspan="2">Client Recv 스레드
Client Process Packet</td></tr>
<tr><td>박신혜</td><td>Client Recv 스레드</td><td colspan="2">Server Recv 스레드
Server Process Packet</td><td>Input Client</td><td colspan="2">processLogSystem Server</td></tr>
<tr><td>리뷰</td><td>기간 안에 수행하지 못함
-> 구현 완료</td><td colspan="2">기간 안에 수행하지 못함
-> 구현 완료</td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td>Client의 leave 패킷을 서버가 받지 못함.
-> 주고받는 데이터의 자료형이 달랐던 문제가 있었음.</td><td colspan="2">과제에서 사용했던 DialogBox로 로그인을 구현함.</td></tr>
<tr><td></td><td>21일</td><td>22일</td><td>23일</td><td>24일</td><td>25일</td><td>26일</td><td>27일</td></tr>
<tr><td>우상훈</td><td colspan="2" rowspan="2">Client-update
ProcessLogSystem Client</td><td rowspan="3">패킷 재설계</td><td rowspan="3">디버깅 및 수정사항 체크</td><td><s>isCollision</s></td><td rowspan="2">Client_ update</td><td rowspan="3">클라 개발 (보스 1)</td></tr>
<tr><td>김민수</td><td colspan="2">Client_render</td></tr>
<tr><td>박신혜</td><td colspan="2">Client-update</td><td colspan="2">Client_render</td></tr>
<tr><td>리뷰</td><td colspan="2"></td><td>패킷의 지속적인 수</td><td></td><td colspan="3">클라이언트에 있는 object들의 id에 대응되는 object를 서버에도 등록만 하게 되면 input 등 client의 입력 처리가 가</td></tr>
</table>

			정 및 추가가 필요 함.		능해짐. 게임에는 여러 scene들이 존재하기 때문에 하나의 충돌 처리 함수로 처리하기 힘들 것 같다고 판단했다. 충돌 처리를 각 scene에게 맡기는 방식으로 구현하기로 정했다.		
	28일	**29일**	**30일**	**12월1일**	**2일**	**3일**	**4일**
우상훈	클라 개발 (보스 1)	패킷 재설계	**디버깅 및 수정사항 체크**	Server_update 재연결 클라 재연결 서버		**디버깅 및 수정사항 체크**	최종 발표 준비
김민수							
박신혜							
리뷰	클라이언트 보스를 구현하지 못함.	패킷의 지속적인 수정 및 추가가 필요함.		다른 구현의 완성도가 떨어져 우선 로그인부터 월드맵까지 다중 접속과 재접속의 완성을 목표로 구현을 사작함.		재접속을 구현 부분에서 많은 오류가 발생함.	
	5일	**6일**	**7일**	**8일**	**9일**	**10일**	**11일**
우상훈	최종 발표 준비	**최종 발표**					
김민수							
박신혜							
리뷰							
	12일						
우상훈							
김민수							
박신혜							
리뷰							

2024.12.12

<table>
<tr><td></td><td>11월7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td><td>12일</td><td>13일</td></tr>
<tr><td>우상훈</td><td colspan="3" rowspan="3">클라이언트 수정</td><td colspan="2">Accept 스레드</td><td rowspan="3">시험
공부</td><td>Server Update</td></tr>
<tr><td>김민수</td><td colspan="2">Client Send 스레드</td><td>Client Recv 스레드</td></tr>
<tr><td>박신혜</td><td colspan="2">Server Send 스레드</td><td>Client Recv 스레드</td></tr>
<tr><td>리뷰</td><td colspan="3">지속적인 수정이 필요하다.</td><td colspan="2">기간 안에 수행하지 못함
-> 구현 완료</td><td></td><td>기간 안에 수행하지 못함
-> 구현 완료</td></tr>
<tr><td></td><td>14일</td><td>15일</td><td>16일</td><td>17일</td><td>18일</td><td>19일</td><td>20일</td></tr>
<tr><td>우상훈</td><td>Server Update</td><td colspan="2">AcceptThread
ServerUpdate
Server Process Packet
Client Process Packet</td><td>패킷
재설계</td><td rowspan="3">디버깅 및 수정사항 체크</td><td colspan="2" rowspan="3">클라이언트 개발
Client-render
(로그 화면)

processLogSystem
Server</td></tr>
<tr><td>김민수</td><td>Client Recv 스레드</td><td colspan="2">Client Recv 스레드
Client Process Packet</td><td></td></tr>
<tr><td>박신혜</td><td>Client Recv 스레드</td><td colspan="2">Server Recv 스레드
Server Process Packet</td><td>Input Client</td></tr>
<tr><td>리뷰</td><td>기간 안에 수행하지 못함
-> 구현 완료</td><td colspan="2">기간 안에 수행하지 못함
-> 구현 완료</td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td>Client의 leave 패킷을 서버가 받지 못함.
-> 주고받는 데이터의 자료형이 달랐던 문제가 있었음.</td><td colspan="2">과제에서 사용했던 DialogBox로 로그인을 구현함.</td></tr>
</table>

<table>
<tr><td></td><td>21일</td><td>22일</td><td>23일</td><td>24일</td><td>25일</td><td>26일</td><td>27일</td></tr>
<tr><td>우상훈</td><td colspan="2" rowspan="2">Client-update</td><td rowspan="3">패킷 재설계</td><td rowspan="3">디버깅 및 수정사항 체크</td><td><s>isCollision</s></td><td>Client_ update</td><td rowspan="3">클라 개발 (보스 1)</td></tr>
<tr><td>김민수</td><td colspan="3" rowspan="2">Client_render</td></tr>
<tr><td>박신혜</td><td colspan="2">ProcessLogSystem Client
Client-update</td></tr>
<tr><td>리뷰</td><td colspan="2"></td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td colspan="4">클라이언트에 있는 object들의 id에 대응되는 object를 서버에도 등록만 하게 되면 input 등 client의 입력 처리가 가능해짐. 게임에는 여러 scene들이 존재하기 때문에 하나의 충돌 처리 함수로 처리하기 힘들 것 같다고 판단했다. 충돌 처리를 각 scene에게 맡기는 방식으로 구현하기로 정했다.</td></tr>
<tr><td></td><td>28일</td><td>29일</td><td>30일</td><td>12월1일</td><td>2일</td><td>3일</td><td>4일</td></tr>
<tr><td>우상훈</td><td rowspan="3">클라 개발 (보스 1)</td><td rowspan="3">패킷 재설계</td><td rowspan="3">디버깅 및 수정사항 체크</td><td colspan="2" rowspan="3">Server_update
재연결 클라
재연결 서버</td><td rowspan="3">디버깅 및 수정사항 체크</td><td rowspan="3">최종 발표 준비</td></tr>
<tr><td>김민수</td></tr>
<tr><td>박신혜</td></tr>
<tr><td>리뷰</td><td>클라이언트 보스를 구현하지 못함.</td><td>패킷의 지속적인 수정 및 추가가 필요함.</td><td colspan="2">다른 구현의 완성도가 떨어져 우선 로그인부터 월드맵까지 다중 접속과 재접속의 완성을 목표로 구현을 사작함.</td><td>재접속을 구현 부분에서 많은 오류가 발생함.</td><td></td></tr>
<tr><td></td><td>5일</td><td>6일</td><td>7일</td><td>8일</td><td>9일</td><td>10일</td><td>11일</td></tr>
<tr><td>우상훈</td><td rowspan="3">최종 발표 준비</td><td rowspan="3">최종 발표</td><td rowspan="3">서버, 클라 재접속</td><td rowspan="3">시험 공부, 재접속 오류 수정</td><td rowspan="3">서버, 클라 재접속</td><td rowspan="3">시험 공부, 재접속 오류 수정</td><td rowspan="3">서버, 클라 재접속</td></tr>
<tr><td>김민수</td></tr>
<tr><td>박신혜</td></tr>
</table>

리뷰			공유 자원 동기화 설계 문제, 패킷 설계 오류, 오브젝트 삭제 문제, send 실패 문제 등을 겪으며 다중 접속과 재접속을 구현하게 되었음.
	12일		
우상훈	서버, 클라 재접속		
김민수			
박신혜			
리뷰			

주요기능

회원가입 기능	X
로그인 기능	O
게임 진행도 저장 기능	X
모든 플레이어는 로비에서 대기한 뒤, 다같이 게임을 시작	O
월드맵	O
보스 스테이지	X
재접속 기능	O