네트워크 게임 프로그래밍 Term Project

2019180016 백종화

2019182007 김문수

2019182009 김석희

**애플리케이션 기획 (무슨 교과목에서 누가 작업한 게임인지 명시할 것)**

김석희, 백종화 학우가 윈도우 프로그래밍에서 C++로 만든 프로젝트를 이용할 예정입니다.

**게임 설명**

**A. 게임 소개 및 특징**

-> 적을 막는 디펜스 게임과 카드 장르를 합친 게임입니다. 또한 카드 게임 장르에서 덱 빌딩이라는 다양성으로 매 게임당 새로운 나만의 조합으로 즐길 수 있습니다.

-> 캐릭터가 적을 공격할 수 있는 방식이 카드 밖에 없기에 적절한 카드 분배와 마나 관리 같은 전략적 판단을 요구하기도 합니다. 또한 덱 빌딩 시 강력한 카드는 마나 소모가 크게 설정되어 있어서 밸런스 있는 덱 빌딩을 중요로 합니다.

**B. 게임 규모**

-> 크게 총 3웨이브로 이루어져 있습니다. 웨이브에 따라 추가되는 몬스터가 있으며 최종적으로 보스를 잡으면 끝납니다. 그 전에 자신의 타워가 피해를 입어 파괴되면 패배로 끝납니다. 몬스터는 지상 몬스터, 공중 몬스터가 있고, 보스까지 총 3종류가 존재합니다.

-> 게임의 메인이라고 할 수 있는 카드는 많지 않지만 카드 하나하나에 트라이포드 시스템이 있어 카드를 강화하거나 다양하게 바꿀 수 있습니다. 3단계로 이루어져 1단계 3가지, 2단계 3가지, 3단계 2가지 중 하나씩 골라 총 18개의 경우의 수가 한 카드에서 나옵니다.

골드와 상점 시스템이 존재하여 자신의 카드를 바꿀 수 있습니다. 몬스터를 잡으면 골드와 경험치가 증가합니다. 경험치를 얻어 레벨업하면 상점이 열려 모아둔 골드로 새롭거나 기존에 있던 카드를 덱에 추가할 수 있습니다. 그 뿐만 아니라 기존에 있던 카드 다음 단계의 트라이포드를 구매하여 강화할 수 있습니다.

**추가한 내용**

기존에는 솔로 플레이 게임으로 단순히 디펜스 및 덱 빌딩 게임이었지만 위 게임을 협동 게임으로 네트워크 게임을 설계하면 충분한 네트워크 설계를 경험할 수 없다 라는 교수님의 조언에 따라 디펜스 협동게임이 아닌, 디펜스 경쟁게임으로 바꾸었으며 이에 따라 카드 종류에 몬스터 카드를 추가하여 사용시 상대편 성으로 몬스터를 소환하는 효과를 지닌 카드를 만들었습니다. 또한 플레이어는 추가되는 뷰포트를 통해 상대진영의 상태를 확인할 수 있으며 이에 따라 몬스터 카드를 사용하는 타이밍을 적절히 판단할 수 있게 할 계획입니다.

**실행방법**

-> 키보드의 w, a, d키로 움직입니다. 마우스 드래그 앤 드랍으로 카드를 사용할 수 있으며, 일정 게이지를 다 채워야 쓸 수 있는 각성기는 E키로 쓸 수 있습니다.

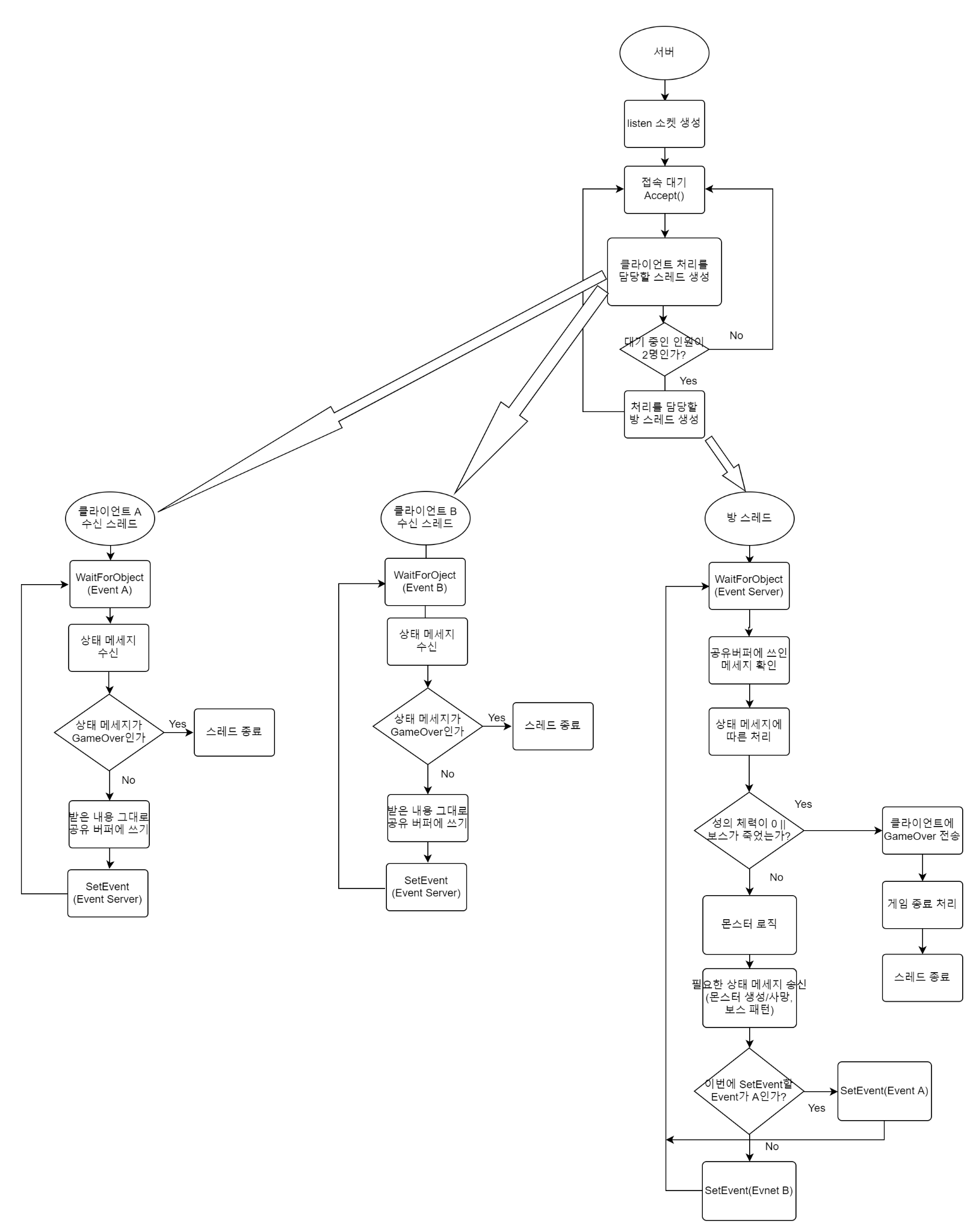
**인게임 스크린샷**



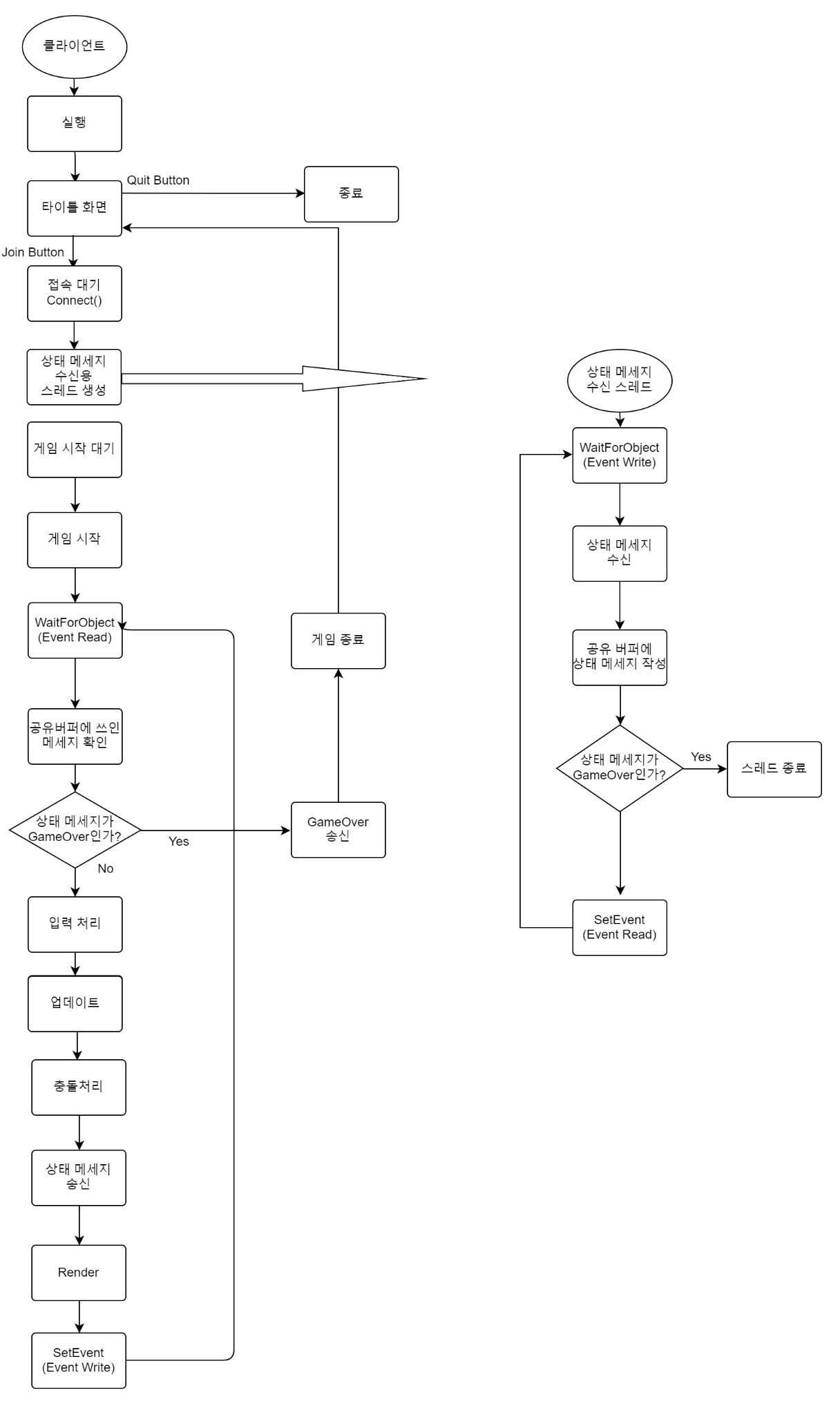
**위 사진의 상단 부분을 확장하여 상대편의 진영 상황을 확인하는 뷰포트를 만들 기획입니다.**

**High-level 디자인**

**서버**



**클라이언트**

****

**스레드 동기화 과정**

**서버**

1. 1번 스레드에서 상태 메시지를 수신 후 공유 버퍼에 쓴다.
2. 3번 스레드에서 클라이언트 A에게서 온 상태 메시지를 처리하고 게임 로직을 수행한다
3. 2번 스레드에서 상태 메시지를 수신 후 공유 버퍼에 쓴다.
4. 3번 스레드에서 클라이언트 B에게서 온 상태 메시지를 처리하고 게임 로직을 수행한다
5. 1~4의 내용을 반복하며 4번 단계에서 게임오버를 판단해 승패를 결정한다.
6. 위 순서는 이벤트를 통해 제어한다.

**클라이언트**

1. 수신 스레드에서 서버에게서 온 상태 메시지를 공유버퍼에 쓴다.
2. 메인 스레드에서 상태 메시지를 처리하고 게임로직을 수행한다.
3. 1~2번을 반복하며 1번 단계에서 게임오버 메시지를 수신하면 메시지를 공유버퍼에 쓰고 수신 스레드를 종료하며, 이후 메인스레드에서 게임오버 메시지를 처리한다.
4. 1~2번 과정의 순서는 이벤트를 통해 제어한다.

**Low-level 디자인**

변수 작성 법: (멤버 변수: m\_, 전역 변수: g\_)(자료형타입)(변수명) ex:m\_iNum

변수명은 파스칼 표기법으로 작성 ex: PersonState

함수명은 동사

상태 메시지 방식 정의:

사용할 열거자:

enum StateMsgSender {Server, Client1, Client2};

enum StateMsgType { Monster\_Spawn, Monster\_Hp, Player\_Move, Player\_Use\_Card, Castle\_Hp, … };

enum MonsterType {M\_Wolf, M\_Bat, M\_Boss}

enum CardType { … }

enum PlayerMove{ … }

사용할 변수:

BYTE StateMsg; // 상태 메시지를 정의하는 변수, 하위 6비트는 보내는 내용 상위2비트는 보내는 사람을 비트 연산으로 담을 예정 00|000000

StateMsg을 먼저 전송하고 정의에따라 아래의 구조체 중 하나를 전송한다.

struct StateMsgArgu

{

}

struct MonsterSpawnStateMsg : StateMsgArgu

{

MonsterType MonsterId;

int MonsterSerialId;

}

struct MonsterHpStateMsg : StateMsgArgu

{

MonsterType MonsterId;

int MonsterSerialId;

int Hp;

}

struct PlayerMoveStateMsg : StateMsgArgu

{

int PlayerId;

PlayerMove pMove;

}

struct MonsterUseCardStateMsg : StateMsgArgu

{

int PlayerId;

CardType card;

}

struct CastleHpStateMsg : StateMsgArgu

{

int Hp;

}

서버에 대기 중인 클라이언트 정보, 두 개라면 방을 생성 후 비운다.

(서버)HANDLE ReadyClient[2];

사용할 함수:

**Client**

인자 2개를 순서대로 전송

-void SendStateMsg(BYTE, StateMsgArgu\*);

상태 메시지를 수신하여 공유 버퍼로 쓰기

-void RecvStateMsg();

업데이트

공유버퍼에 있는 상태 메시지를 처리

-void StateMsgUpdate();

-void PlayerUpdate();

-void MonsterUpdate();

충돌 확인

void CheckCollision();

업데이트 후 그리기

-void Render();

입력 처리

-void InputCheck();

대기 방 입장 및 시작

void EnterTheRoom();

void StartGame();

BYTE StateMsgDefine을 만들기 위한 함수

-void MakeStateMsgDefine(StateMsgSender sender, StateMsgType type);

**Server**

인자 2개를 어느 클라이언트에게 보낼 것인가, 안 적으면 둘 다

-void Send(BYTE StateMsgDefine, StateMsgArgu\* msg, int ClientId = 0);

-void Recv();

플레이어가 다 들어올때까지 기다리기

void WaitPlayer();

몬스터 로직 관련 함수, 여기서 몬스터의 생성 상태 메시지를 전송할 계획

-void Update();

서버가 accpet후 플레이어 구분을 위한 ID설정

-void SetClientId();

게임이 끝나는 조건인지 체크

-bool IsGameOver();

BYTE StateMsgDefine을 만들기 위한 함수

-void MakeStateMsgDefine(StateMsgSender sender, StateMsgType type);

**팀원 별 역할분담 (구성원 모두가 네트워크 기능 구현에 참여해야 함)**

백종화: 서버 방 스레드 설계, 클라이언트 몬스터 및 카드 관련 설계

김석희: 클라이언트 메인스레드 설계, 클라이언트 상태 메시지 수신 스레드 설계

김문수: 서버 메인 스레드 설계, 서버 상태 메시지 수신 스레드 설계

**개발환경**

언어: C++

개발도구: Visual Studio 2022

운영체제: Windows

**개발일정 (일별/개인별 계획 수립, 달력 형태로 작성)**

**백종화: 서버 방 스레드 설계, 클라이언트 몬스터 및 카드 관련 설계**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 10/31  클라이언트 Card class 설계 | 11/1  클라이언트 Monster class 설계 | 2  클라이언트 Wolf class 설계 | 3  클라이언트 Boss class 설계 (렌더 및 기본적인 생성자) | 4 |
| 5  클라이언트 Bat class 설계 | 6  인공지능 시험 준비 | 7  게임수학 시험 준비 | 8  클라이언트 Boss class 설계 (공격 패턴) | 9 | 10  클라이언트 Boss class 설계 (피격 및 부위파괴, 무력) | 11 |
| 12  1주차 피드백 및 부족한 진도 보강 | 13  게임소프트공학 시험 준비 | 14 | 15  서버 방 class 설계  (대기 클라가 2명이었을때 스레드 생성) | 16  서버 방 class 설계  (스레드 생성 후 클라에 번호 부여) | 17  (지스타로 휴식) | 18  (지스타로 휴식) |
| 19  (지스타로 휴식) | 20 | 21  서버 방 class 설계  (상태 메시지 송수신 처리) | 22  서버 방 class 설계  (몬스터 로직 추가 생성과 그에 따른 상태 메시지) | 23 | 24  서버 방 게임 초기화 상태 메시지 관련 로직 구현 | 25  서버 방 게임 시작 상태 메시지 관련 로직 구현 |
| 26  2, 3주차 피드백 및 부족한 진도 보강 | 27 | 28  서버 방 게임 종료 상태 메시지 관련 로직 구현 | 29 | 30  서버 방 내에 있는 몬스터 식별번호 구현 | 12/1 | 2  식별번호로 메세지 통신 원활한지 확인 |
| 3  4주차 피드백 및 부족한 진도 보강 | 4  코드 최적화 및 리팩토링 | 5  코드 최적화 및 리팩토링 | 6  디버깅 및 버그수정 | 7  디버깅 및 버그수정 |  |  |

**김석희: 클라이언트 메인스레드 설계, 클라이언트 상태 메시지 수신 스레드 설계**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 10/31 | 11/1 | 2 | 3  부분통과로 인한 제안서 재설계 | 4 |
| 5 | 6  Player class 구현  좌우 이동 등 Input 명령에 대한 구현 | 7  Player class 구현  스킬 사용관련 애니메이션 구현 | 8  Player class 구현  1p, 2p를 구분하기 위한 것들 위주로 구현 | 9  Player class 구현  Player Update 함수 구현 | 10 | 11  Player 관련 상태 메시지 구현 |
| 12  1주차 피드백 | 13  충돌 체크관련 함수 구현 | 14  충돌 체크 상태 메시지 구현 | 15  카드와 스킬사용 연결 | 16 | 17 | 18 |
| 19 | 20  상태 메시지 수신 스레드 구현 | 21  이벤트 객체를 활용한 순서 제어 구현 | 22 | 23 | 24  상태 메시지 관련 코드 정리 | 25 |
| 26  2, 3주차 피드백 | 27  Player와 보스기믹간 상호작용 구현 | 28 | 29  Player의 각성기 관련 구현 | 30 | 12/1  게임 재시작 관련 구현  오브젝트들의 삭제 관련 | 2 |
| 3  4주차 피드백 | 4  게임 재시작 관련 구현 완성 | 5 | 6  디버깅 | 7  디버깅 |  |  |

**김문수: 서버 메인 스레드 설계, 서버 상태 메시지 수신 스레드 설계**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 10/31 | 11/1 | 2 | 3 | 4 |
| 5 | 6 | 7  listen 소켓  생성 | 8  서버 패킷 수신 구현  (accept) | 9  클라이언트 처리 담당 스레드 생성 | 10  waitPlayer() 구현 | 11  서버 패킷 송신 구현  Ready to Play 1p,2p |
| 12  1주차 피드백 | 13  시험준비 | 14  시험준비 | 15  시험준비 | 16 | 17  코드 정리 및 점검 | 18 |
| 19  2주차  피드백 | 20  방 스레드 생성(로직 처리) | 21  상태 메시지  수신 구현  StateMsg | 22  상태 메시지  수신 구현  StateMsg | 23 | 24  StateMsg 공유 버퍼에 write | 25  StateMsg 공유 버퍼에 write |
| 26  3주차 피드백 | 27  코드 정리 및 점검 | 28  WaitForObject, SetEvent  구현 | 29  WaitForObject, SetEvent  구현 | 30 | 12/1  IsGameOver() 구현 | 2  클라 로직 스레드 종료 구현 |
| 3  4주차  피드백 | 4 | 5  코드 정리 및 점검 | 6  디버깅 및 버그 수정 | 7  디버깅 및 버그 수정,  최종 점검 |  |  |