자료구조

기존의 확장 오버랩 구조체는 iocp\_buf가 쓰이지 않아도 생성이 되어서 개선하기 위해 타이머 오버랩 구조체와 db 오버랩 구조체를 작성하였다.

struct T\_OVER\_EX {

WSAOVERLAPPED wsa\_over;

char op\_mode;

};

struct DB\_OVER\_EX {

WSAOVERLAPPED wsa\_over;

char op\_mode;

int object\_id;

char name[10];

};

기존 클라이언트 객체를 npc와 클라이언트 전용으로 나눠서 데이터 낭비를 줄였다

class object\_info

class npc\_info : public object\_info

class client\_info : public object\_info

게임흐름

게임 접속 시 아이디를 입력 받는다. 입력 받은 아이디를 db에서 검색하여 존재하면 기존 데이터를 사용하고 존재하지 않으면 db에서 새로 데이터를 생성한다.

게임에 접속하면 오른쪽 위 화면에 자신의 상태를 알 수 있다. 그리고 enter 키를 누르면 채팅을 할 수 있으며 채팅로그는 10개까지 볼 수 있다. 채팅로그에는 자신과 다른 플레이어의 채팅과 자신이 공격하거나 스탯이 바뀌었을 경우의 문구가 기록된다.

화살표 키를 이용하여 맵을 돌아다닐 수 있다. 맵에는 세가지 종류의 몬스터가 있다. 고정되어서 공격 받으면 반격하는 타입과 로밍형에 공격 받으면 반격하는 타입 그리고 자신의 영역에 접근하면 쫓아와서 공격하는 타입이 있다. 이 중 로밍형과 어그로형 몬스터는 자신의 배치 좌표에서 일정 거리만큼 멀어지면 더 이상 플레이어에게 적대적이지 않게 된다. 몬스터들의 AI는 스크립트로 구현되어 있다.

플레이어는 A 키를 이용해 공격을 할 수 있다. 공격의 딜레이를 타이머 설정하여 1초에 한 번만 공격할 수 있다. 몬스터에게 공격 받으면 체력이 감소하고 몬스터를 잡게 될 경우 경험치가 오른다. 경험치가 일정수치 이상이면 레벨업을 하고 최대 체력과 경험치 요구량이 증가한다.

플레이어가 몬스터의 공격에 사망하게 되면 초기 위치로 전송되고 체력은 다 회복되고 경험치는 반이 감소한다. 체력은 5초마다 10프로씩 자동회복 된다. 클라이언트 윈도우를 종료하면 db에 자신의 위치와 레벨, 체력, 경험치 등을 저장한다.

알고리즘

섹터 – 자신과 가까운 오브젝트들을 빠르게 검색하기 위해서 만들었다.

struct SECTOR {

unordered\_set<int> sector;

shared\_mutex s\_lock;

};

SECTOR sectors[WORLD\_HEIGHT / SECTOR\_HEIGHT][WORLD\_WIDTH / SECTOR\_WIDTH];

섹터는 2차원배열로 지정되었고 각각의 섹터는 unordered\_set을 사용해 클라이언트 id를 기록하였다. 시야에 따라 검색해야 하는 섹터의 수가 1개에서 4개까지 증가한다. 이 섹터들에 들어있는 오브젝트에 대해서 시야에 들어와 있는 오브젝트를 검색하였다.

프리리스트

플레이어와 npc의 숫자가 많은 게임이고 그에 따라 전송에 필요한 오버랩 객체가 배 이상 증가하였다. 기존에는 확장 오버랩 구조체는 new로 할당 받고 사용을 다하면 delete를 하는 방식이었다. 이런 메모리 할당 및 해제의 오버헤드를 줄이고자 사용하였다.

이 외에도 성능 향상을 위해 멀티코어 프로그래밍에서 배운 TBB의 container들을 사용해보았는데 성능에 큰 차이가 느껴지지 않아서 의아함을 느꼈다. 멀티코어 프로그래밍에서 좋은 것들을 많이 배우긴 했는데 응용을 많이 못해서 아쉽다.

몬스터 스크립트에는 FSM 형식의 AI를 작성하였다.

성능테스트

