**실행방법**

1. 서버 실행
2. 클라이언트 실행 -> ip주소, id 입력 -> ok버튼

**조작법**

**이동:** 화살표키

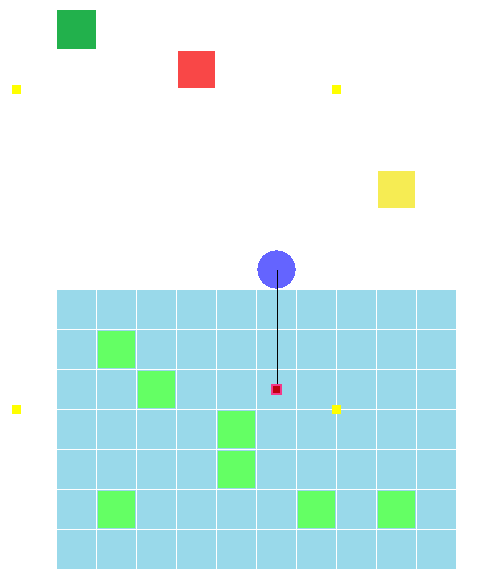
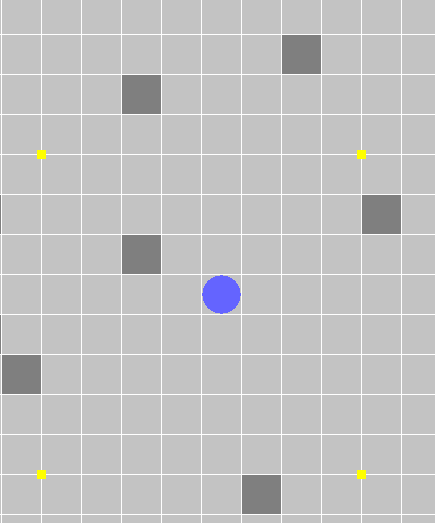
**공격:**

일반 공격: wasd

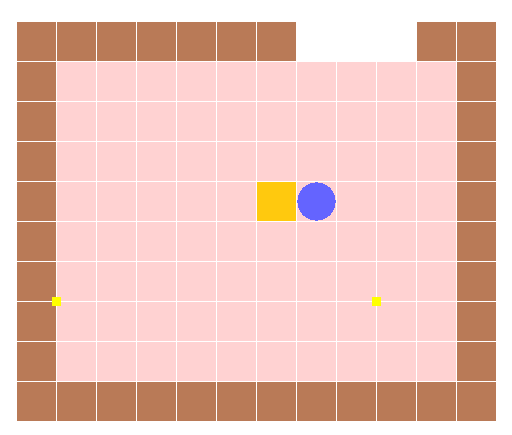
좁은 광역 공격: e

넓은 광역 공격: q

**낚시:** 물가에서 space **채굴:** 광산에서 space

**상점:** 마을 내의 오렌지색 부분에서 space



아이템 **물약 사용:** ‘1’ 아이템 **방어구 사용**: ‘2’

**프로그램 구조**

**Data Race**

비교적 세밀한 동기화를 위해 sector 별로 mutex가 존재하고 이동, 공격할 때 필요한 sector들의 lock을 얻은 후 필요한 동작을 하도록 구현했다. 이렇게 함으로써 각 player들의 view list에 허수아비나 유령이 존재하지 않게 했다. Lock을 걸 때 최대 4개 정도 sector에 lock을 걸며 sector id 숫자가 낮은 순서대로 건다.

DB event 큐나, Timer event 큐, Memory Pool 스택은 성긴 동기화를 이용해 구현했다.

**Memory 관리**

자주 new, delete 되는 Send Packet이나 Overlapped 확장 구조체 들은 메모리 풀을 이용하여 new를 최소한으로 하도록 관리하였다. 각 메모리 풀은 프로그램이 종료될 때 할당 받은 메모리 들을 해제한다.

**Protocol**

알고리즘 강의자료에 정의된 프로토콜을 확장해 사용했다.

**NPC 이동, 공격**

플레이어가 잠들어 있는 NPC를 발견하면 해당 NPC의 active를 true로 만들고 Timer event 큐에 해당 NPC의 이동 event를 등록한다. Timer thread에서 해당 event를 PQCS로 worker thread로 보내면 NPC는 이동 후 플레이어가 근처에 있으면 다시 이동 event를 등록한다. 플레이어가 안보이기 시작하면 자연스러운 이동을 위해 배정된 임의의 숫자만큼 더 이동한 후 Timer event 큐에 이동 event를 등록하지 않는다.

상태가 Aggressive인 NPC는 이동하기전에 플레이어를 공격할 수 있는지 확인 후 공격할 수 있으면 공격하고, 플레이어가 보이는 거리면 플레이어를 쫓는다. 둘다 false이면 상태를 idle로 바꾼다.

초반 길 찾기는 A\*를 이용해 구현하였으나 장애물이 복잡하지 않은 것에 비해 overhead가 커서 가면서 길 찾기(장애물 만나면 벽 잡고 가기)로 구현했다.

이동이나 공격 시 주변의 볼 수 있는 플레이어들에게 이동이나 공격 상황을 알려 준다.