IOCP 실습- 2

MM4220 게임서버 프로그래밍

정내훈

차 례

- NPC 구현

NPC 구현

- 고려할 점
  - ID 생성 체계
    - 플레이어와 NPC의 구분
      - bool is_npc?
        - Object 객체 내부 접근 필요.
      - ID의 영역의 구별
        - 멤버 변수를 확인 하지 않아도 됨.
        - NPC_START 상수 필요

NPC 구현

- 1차 구현
  - 구현 내용
    - 1초에 1번 랜덤 무브
  - AI Thread를 사용한 구현
    - AI Thread에서 시간 검사 및 이동 판단
  - HEART BEAT을 사용한 구현
    - HEART BEAT자체가 기준 시간에 맞춰 호출됨.
  - NCP 시야 구현
    - 시야 리스트를 따로 저장할 필요는 없음.
      - 클라이언트가 연결되어 있지 않기 때문

NPC 구현

- 1차 구현 – 실습
  - AI Thread 구현

NPC 구현

- 1차 구현 – HEART BEAT 구현
  - AI Thread의 간략화
  - 시간 동기화 Logic의 간소화

NPC 구현

- 2차 구현 – 시야 구현
  - 부하 측정 및 비교.

NPC

- Timer 기반 동작
  - Timer : 서버 동작 기본 요소
    - Real time동작을 위해 필요
    - Timing에 맞춘 동작들 구현
      - 이동, 마법 시전, HP회복…
  - NPC 이동
    - Timer에 동기화 되어 있다.
      - 이동시간이 될 때 까지 기다렸다가 이동 (X)
      - Timer에 다음 이동시간을 등록하고 종료 (O)

NPC

- Timer 구현
  - Timer Thread

```
Event_queue timer_queue

TimerThread()
        do
                sleep(1)
                do
                        event k = timer_queue.top
                        if  k.starttime > current_time()
                                break
                         timer_queue.pop(&id, &k)
                        process_event_callback(id, k)
                while true;
        while true;
```

Timer

- NPC
  - IOCP를 통해서 NPC제어?
    - Thread programming!
    - PC에 관련된 Event도 IOCP를 통해서!!
  - Timer Thread에서 A*를 할 수는 없다.
    - Timer Thread과부하!

Timer

- 이벤트 큐
  - 저장 정보
    - 어떤 오브젝트가 언제 무엇을 누구에게 해야 하는가.

```
struct event_type {
    int obj_id;
    high_resolution_clock::time_point wakeup_time;
    int event_id;
    int target_id;
};
```

Timer

- 이벤트 큐
  - 시간 순서대로 정렬된 우선순위 큐가 필요하다.

```
class mycomparison
{
  bool reverse;
public:
  mycomparison() {}
  bool operator() (const event_type lhs, const event_type rhs) const
  {
    return (lhs.wakeup_time > rhs.wakeup_time);
  }
};

priority_queue<event_type,vector<event_type>,mycomparison> p_queue;
```

NPC

- NPC
  - Timer Queue와 Worker Thread 만으로는 부족
  - 대부분의 NPC가 timer queue로 동작한다면 timer thread의 과부하
  - 플레이어가 관찰할 수 있는 NPC만 움직여야 한다.
    - Is_alive 이외에 is_active 필요.

NPC

- NPC : 타이머를 사용한 이동

```
Event_queue timer_queue

NPC_Create()
        foreach NPC
                push (timer_queue, id, MOVE_EVENT, 1)

NPC_CALLBACK(id, event)
    if (event == MOVE_EVENT)
        id -> move_npc()
        push (timer_queue, id, MOVE_EVENT, 1)

MOVE_NPC()
        overlap_ex.command = MOVE
        PostQueuedCompletionStatus(port, 0, &NPC_INFO, overlapex)
```

NPC

- NPC : 타이머를 사용한 이동

```
Event_queue timer_queue

NPC_Create()

MOVE_PLAYER() {
        foreach_monster_in_range( push(timer_queue, monster_id, MOVE_EVENT,1);
        foreach_monster_get_away( try_erase_timer_queue(monster_id));
}

NPC_CALLBACK(id, event)
      if (event == MOVE_EVENT)
                id -> move_npc()
                push (timer_queue, id, MOVE_EVENT, 1)

MOVE_NPC()
                overlap_ex.command = MOVE
                PostQueuedCompletionStatus(port, 0, &NPC_INFO, overlapex)
```

NPC

- NPC : 실습
  - InActive상태 구현
  - 플레이어나 NPC 이동/생성 시 active 여부 검사