스크립트 연동

MM4220 게임서버 프로그래밍

정내훈

실습

- 지난번 실습
  - IOCP를 사용한 다중 접속 서버
  - 시야구현
  - NPC구현
    - Heart Beat
    - Timer Thread
    - PQCS연동
    - 시야처리
    - Sleep & WakeUp

차 례

- NPC 스크립트 연동
- 성능측정

NPC SCRIPT 연동

- NPC의 동작을 스크립트 언어로 정의
  - 설정 : 몬스터의 기본값 설정
    - HP, Level, 기본 반응, 기본 대사
  - 반응 : 어떠한 이벤트가 일어 났을 때 그것에 대한 반응
    - 플레이어 출현, 공격 받음
  - 자체 동작
    - 주위 이동, Idle message, 토끼 잡는 늑대

NPC SCRIPT 구현

- NPC의 동작을 스크립트 언어로 정의
  - 기본적으로 FSM이다.
    - event driven구현
    - Event의 출처
      - 시작, 종료, packet receive
      - timer, 다른 객체로 부터의 호출
  - VM(Virtual Machine에 상태 저장)
    - 스크립트로 관리되는 객체 정보는 VM에 저장
      - 예) LUA의 lua_State
    - 스크립트언어 인터프리터는 VM의 내용을 업데이트

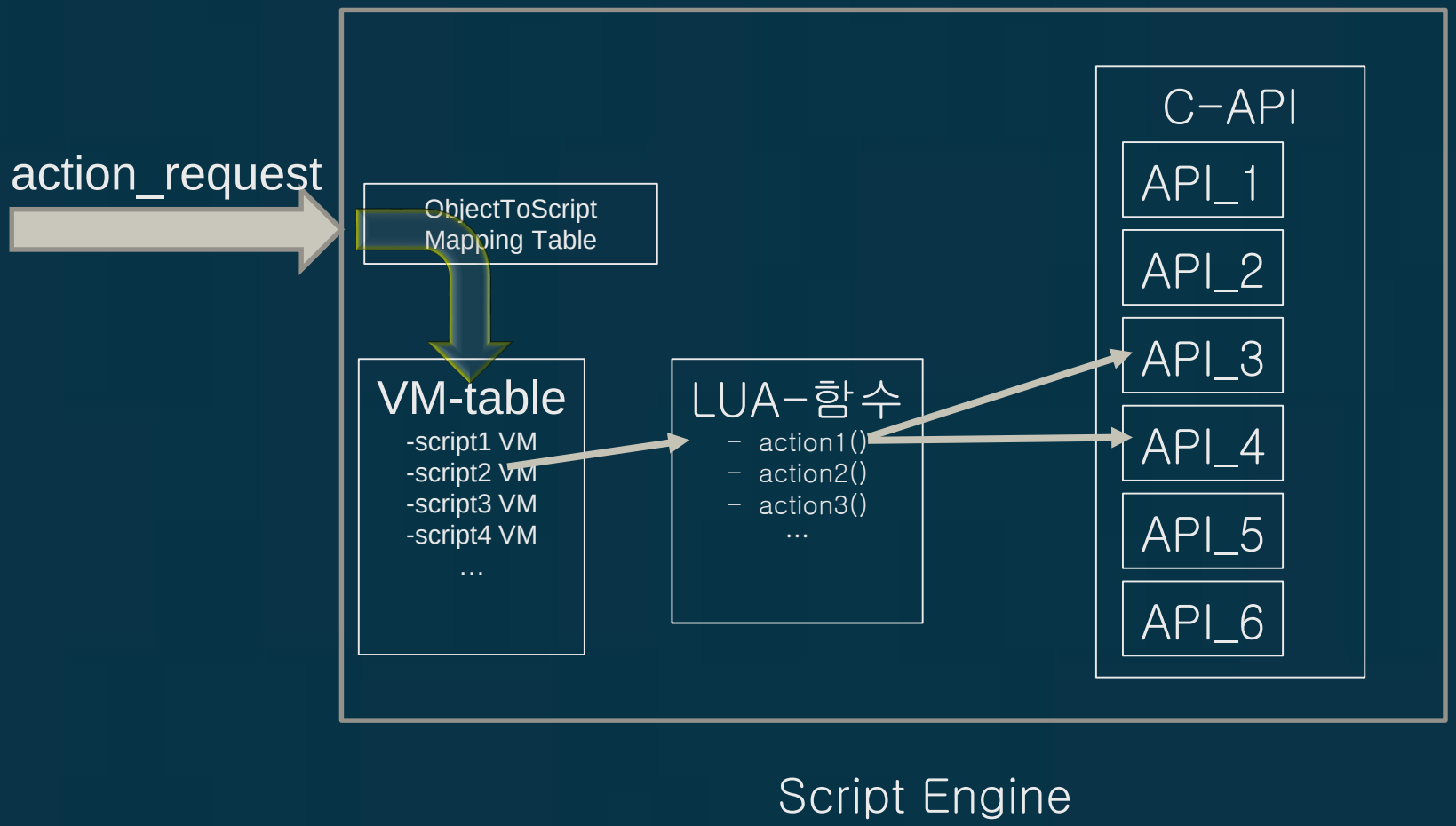
NPC SCRIPT 구현

- 성능 문제
  - 모든 NPC를 script로 돌리는 것은 거대한 삽질
    - 간단한 AI는 하드코딩으로.
  - 멀티 쓰레드 최대한 활용
    - 1 Thread & 1 VM : 하나의 VM이 모든 스크립트 객체를 콘트롤, 성능 문제, bottle neck
    - N Thread & 1 VM : 멀티 쓰레드에서 동시 호출이 가능한 스크립트 언어 필요, Worker Thread들과의 스케쥴링 충돌
    - N Thread & N VM : 하나의 VM이 여러 개의 스크립트객체를 실행, Load Balancing 문제, Worker Thread들과의 충돌
    - N Thread & MM VM : 객체 하나당 하나의 VM, VM은 worker thread에서 실행, 대부분의 VM이 대기상태, 메모리 낭비

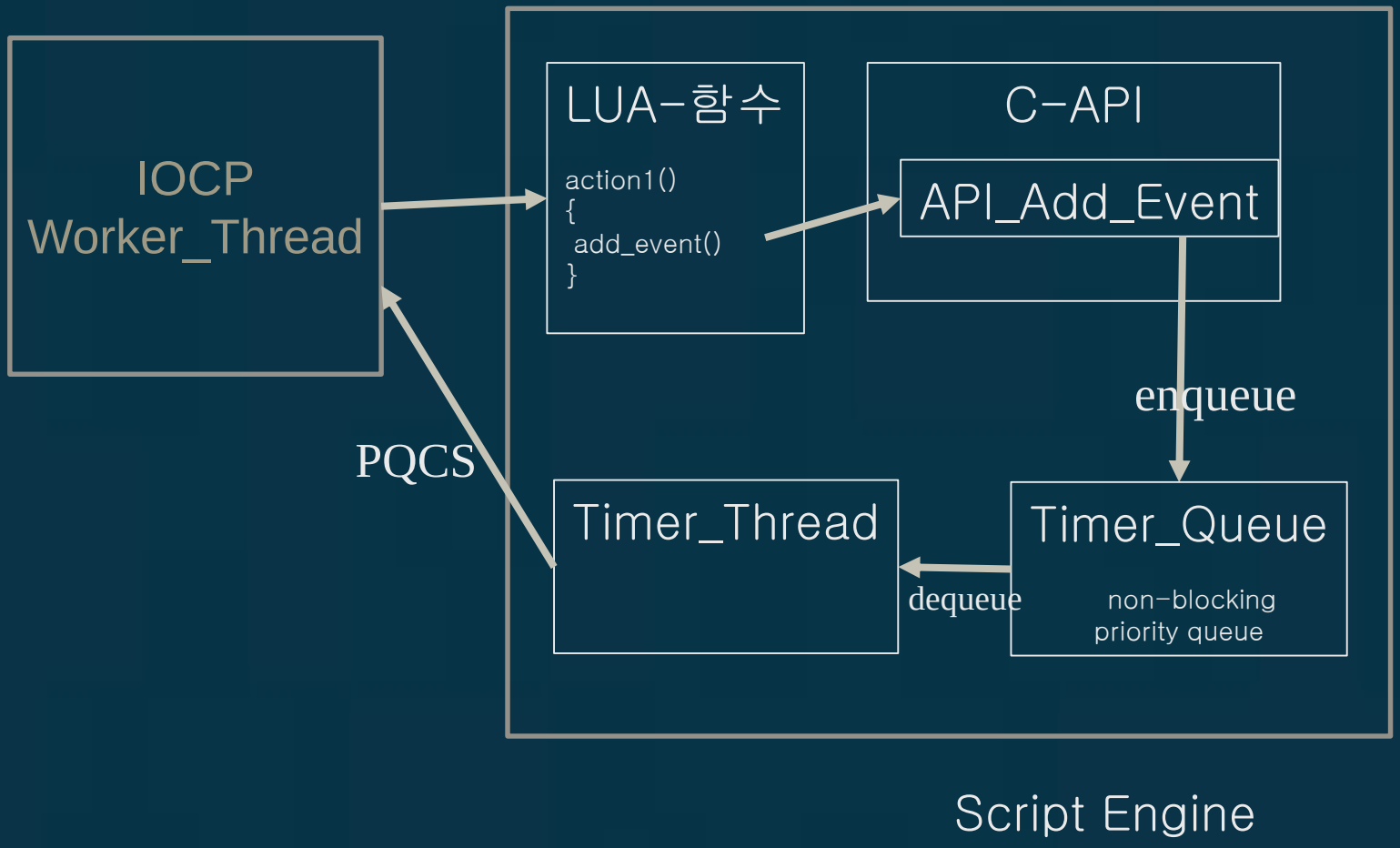
NPC SCRIPT 구현

- Massively Multi Virtual Machine
  - LUA니까 가능
  - Event 별로 해당 객체의 적절한 LUA함수 호출
  - Script에서 사용할 API를 게임서버에서 제공해 주어야 함.
    - Script언어가 실행하기 곤란한 기능들
    - 예) 이동, 메시지, 다른 객체 이벤트 호출, 주변 Object 검색, 클라이언트에 명령 전달

NPC SCRIPT 구현



NPC SCRIPT 구현



NPC SCRIPT 구현

- 구현
  - NPC 정보 구조체에 LUA_STATE를 하나 할당.

```
struct CLIENT_INFO {
   int   id;
   bool in_use;
   SOCKET my_socket;
   int   x;
   int   y;
   EXOVER        recv_over_ex;
   std::set<int>  view_list;
   CRITICAL_SECTION  vl_lock;
   lua_State*L;
};
```

NPC SCRIPT 구현

- AI 구현
  - 플레이어가 바로 옆에 다가가면 “Hello” 메시지 출력.

NPC SCRIPT 구현

- 프로토콜 추가

```
#define MAX_STR_SIZE  100

#define SC_CHAT  4

struct sc_packet_chat{
   BYTE size;
   BYTE type;
   WORD id;
   WCHAR message[MAX_STR_SIZE];
};
```

- 클라이언트 수정
  - 채팅 메시지 표시

NPC SCRIPT 구현

- Chatting Packet 처리 함수 추가

```
void send_chat_packet(int to, int id, WCHAR *message)
{
   sc_packet_chat  pos_packet;

   pos_packet.id = id;
   pos_packet.size = sizeof(sc_packet_chat);
   pos_packet.type = SC_CHAT;
   wcsncpy(pos_packet.message, message, MAX_STR_SIZE);
   send_packet(to, reinterpret_cast<char*>(&pos_packet));
}
```

NPC SCRIPT 구현

- EVENT 정의
  - WorkerThread에서 처리해야 하는 이벤트들

```
#define IOTYPE_SEND        1
#define IOTYPE_RECV        2
#define IOTYPE_PLAYER_MOVE 3
#define IOTYPE_MOVE_SELF   4
```

NPC SCRIPT 구현

- 몬스터 LUA 프로그램

```
myid = 99999;

function set_uid(x)
   myid = x;
end

function event_player_move(player)
   player_x = API_get_x(player);
   player_y = API_get_y(player);
   my_x = API_get_x(myid);
   my_y = API_get_y(myid);
   if (player_x == my_x) then
      if (player_y == my_y) then
         API_SendMessage(myid, player, "HELLO");
      end
   end
end
```

NPC SCRIPT 구현

- 초기화

```
void Initialize_VM()
{
   CreateThread(NULL, 0, timer_thread, NULL, 0, NULL);

   for (int i=0;i < NUM_OF_NPC;++i)
      Create_Monster();

}
```

NPC SCRIPT 구현

- 초기화

```
void Create_Monster()
{
…
    lua_State *L = players[my_id].L = luaL_newstate();
    luaL_openlibs(L);

    int error = luaL_loadfile(L, "monster.lua") ||
    lua_pcall(L,0,0,0);

    lua_getglobal(L,"set_uid");
    lua_pushnumber(L, my_id);
    lua_pcall(L,1,1,0);
    lua_pop(L, 1);// eliminate set_uid from stack after call
…
```

NPC SCRIPT 구현

- 초기화

```
…

    lua_register(L, "API_SendMessage", API_SendMessage);
    lua_register(L, "API_get_x", API_get_x);
    lua_register(L, "API_get_y", API_get_y);
}
```

NPC SCRIPT 구현

- API

```
int API_get_y(lua_State *L)
{
   int user_id =
        (int)lua_tointeger(L, -1);
   lua_pop( L, 2 );
   int y = players[user_id].y;
   lua_pushnumber(L, y);
   return 1;
}
```

```
int API_SendMessage(lua_State *L)
{
   int my_id = (int)lua_tointeger(L, -3);
   int user_id = (int)lua_tointeger(L, -2);
   char *mess = (char *)lua_tostring( L, -1);
   size_t wlen, len = strlen( mess ) + 1;
   wchar_t wmess[MAX_STR_SIZE];

   lua_pop( L, 3 );
   len = (MAX_STR_SIZE-1<len) ?
               MAX_STR_SIZE-1 : len;

   mbstowcs_s( &wlen, wmess, len, mess, _TRUNCATE );
   wmess[MAX_STR_SIZE-1] = (wchar_t)0;

   send_chat_packet( user_id, my_id, wmess);
   return 0;
}
```

NPC SCRIPT 구현

- Worker Thread

```
} else if (overlapped->io_type == IOTYPE_SEND) {
          delete overlapped;
} else if (overlapped->io_type == IOTYPE_PLAYER_MOVE) {
          lua_getglobal( players[id].L, "event_player_move" );
          lua_pushnumber( players[id].L, overlapped->ioc_caller_id );
          lua_pcall( players[id].L, 1, 1,0 );

          lua_pop( players[id].L, 1 );
          delete overlapped;
} else if (overlapped->io_type == IOTYPE_MOVE_SELF) {
          // 랜덤 이동 호출
          // AddTimer
} else {
          printf("Error invalid overlapped\n");
          exit(-1);
}
```

NPC SCRIPT 구현

- Event 생성
  - Process Packet에서 이동 후 주위 NPC에게 IOTYPE_PLAYER_MOVE event를 PQCS로 보냄

숙제 (#8)

- 게임 서버/클라이언트 프로그램 작성
  - 내용
    - 숙제 (#5)의 프로그램의 확장
    - 프로그램 수정
      - NPC들은 멈춰있는 상태에서 시작
      - 플레이어가 다가가면 “Hello”란 메시지를 출력하고 랜덤한 방향으로 3칸 이동
        - 1초에 한 칸 씩 이동
        - 이동 종료 후 “BYE”란 메시지 출력
      - 위의 AI는 LUA로 구현
  - 목적
    - Timer와 LUA 스크립트 연동 학습
  - 제약
    - Windows에서 Visual Studio로 작성 할 것
  - 제출
    - 6월 5일 수요일 오후 1시까지, (수목) 6월 12일 오전 11시 까지
    - 재목에 “게임서버프로그래밍 화목 학번 이름 숙제 8번”
    - Zip으로 소스를 묶어서 e-mail로 제출
      - .sdf, .obj .log같은 필요없는 파일 제거