IOCP 실습

MM4220 게임서버 프로그래밍

정내훈

실습

- IOCP를 사용한 서버 제작
- 제출한 숙제 같이 해보기

실습

- 클라이언트 : 게임 공학과 홈페이지에서 다운로드
  - AsyncSelect 네트워크 모델 사용
- 서버 : 게임 공학과 홈페이지에서 다운로드
  - 클라이언트 테스트용
  - 새로 작성할 것임

실습

- 클라이언트
  - DirectX사용
    - 2D Game Programming과목의 GPDUMB엔진 사용
    - Direct3D9을 사용
  - AsyncSelect사용
    - Recv만 AsyncSelect사용
    - Send는 평범한 blocking Send

실습

- 프로토콜

```
#define CS_UP        1
#define CS_DOWN      2
#define CS_LEFT      3
#define CS_RIGHT     4

#define SC_POS           1
#define SC_PUT_PLAYER    2
#define SC_REMOVE_PLAYER 3
```

실습

- 프로토콜 (클라이언트 -> 서버)

```
struct cs_packet_up {
BYTE size;
BYTE type;
};

struct cs_packet_down {
BYTE size;
BYTE type;
};

struct cs_packet_left {
BYTE size;
BYTE type;
};

struct cs_packet_right {
BYTE size;
BYTE type;
};
```

실습

- 프로토콜 (서버 -> 클라이언트)

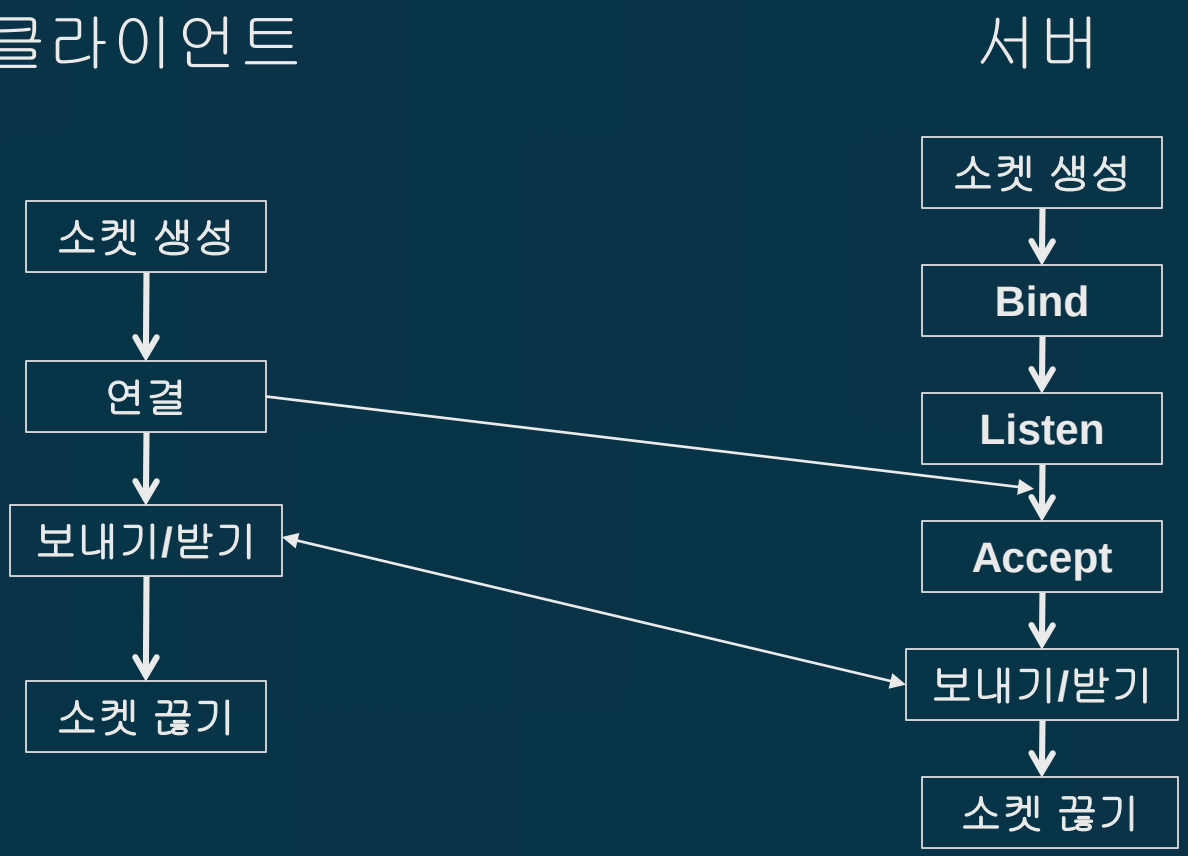
```
struct sc_packet_pos {
        BYTE size;
        BYTE type;
        BYTE id;
        BYTE x;
        BYTE y;
};

struct sc_packet_put_player {
        BYTE size;
        BYTE type;
        BYTE id;
        BYTE x;
        BYTE y;
};
struct sc_packet_remove_player {
        BYTE size;
        BYTE type;
        BYTE id;
};
```

실습

- 프로토콜
  - Object ID
    - 0 – 9 : Player
    - 재사용 필요
  - 맵의 크기
    - 8 X 8
  - 시작 위치
    - (4, 4)

기본 프로그래밍


클라이언트
서버
소켓 생성
연결
보내기/받기
소켓 끊기
소켓 생성
Bind
Listen
Accept
보내기/받기
소켓 끊기

IOCP

- IOCP 서버 작동 순서
  1. 초기화
     - IOCP 핸들 생성
  2. Worker thread 생성
  3. Accept thread 생성

IOCP

- Accept thread
  - 새로 접속해 오는 클라이언트를 IOCP로 넘기는 역할
  - 무한 루프를 돌면서
    - Accept() 호출
    - 새 클라이언트가 접속했으면 **클라이언트 정보 구조체**를 만든다.
    - IOCP에 소켓을 등록한다.
      - Send/recv가 IOCP를 통해 수행됨
    - WSARecv()를 호출한다.
      - Overlapped I/O recv 상태를 항상 유지해야 한다.

IOCP

- 클라이언트 정보 구조체
  - IOCP에서 오고 가는 것은 completion_key와 overlapped I/O pointer 뿐
  - GetQueuedCompletionStatus를 받았을때 어느 클라이언트인지 가장 쉽게 아는 법?
    - Completion_key를 구조체 배열의 index로 한다.
  - Overlapped I/O pointer를 확장한다 던데?
    - 하나의 클라이언트당 recv용으로 overlapped I/O 구조체가 1개 필요.

IOCP

- Overlapped I/O pointer를 확장
  - Overlapped I/O 구조체 자체는 쓸만한 정보가 없다.
  - 따라서 정보들을 더 추가할 필요가 있다.
    - 뒤에 추가하면 IOCP는 있는지 없는지도 모르고 에러도 나지 않는다. (pointer만 왔다 갔다 하므로)
  - 꼭 필요한 정보
    - 지금 이 I/O가 send인지 recv인지????
  - 추가로 필요한 정보
    - 실제 데이터가 저장되는 버퍼 (모아서 관리하면 편리, new/delete를 따로 하지 않아도 됨)

IOCP

- WorkerThread
  - 무한루프
    - GetQueuedCompletionStatus를 부른다.
    - 에러처리/접속종료처리를 한다.
    - Send/Recv처리를 한다.
      - **확장** Overlapped I/O 구조체를 유용하게 사용한다.
      - Recv
        - 패킷이 다 왔나 검사 후 다 왔으면 패킷 처리
        - 여러 개의 패킷이 한번에 왔을 때 처리
        - 계속 Recv호출
      - Send
        - 구조체 및 버퍼 Free

IOCP 서버 구현

- 먼저 할 일
  - 다중 접속 관리
    - 클라이언트 접속 시 마다 ID 부여
  - 패킷 포맷 및 프로토콜 정의
    - 기본 패킷 포맷
      - 길이 (Byte) + 타입 (Byte) + Data (….)
    - Client -> Server
      - 이동 패킷
    - Server -> Client
      - 위치 지정, 플레이어 추가

IOCP 서버 구현

- 먼저 할 일
  - 패킷 처리 루틴 작성

```
void ProcessPacket(int id, unsigned char buf[])
```

  - 이동 방향에 따라 좌표 수정 후 Broadcast

IOCP 서버 구현

- Recv의 구현

Start :
    모든 데이터를 처리했으면 Goto 종료
    남는 데이터로 패킷을 완성할 수 있느가?
        예 : 패킷 버퍼 완성 , 패킷 처리 함수 호출
        아니오 : 남는 데이터 모두 패킷 버퍼로 전송
    goto Start
종료:
    Recv를 호출

IOCP 서버 구현

- 중간 정리
  - 지금 까지 한 일
    - 기본 틀 작성
      - IOCP, worker_thread, accept_thread
  - 앞으로 남은 일
    - send, recv 완성
      - recv시 패킷 조립
      - WSABUF, 확장 Overlapped 구조체 완성

IOCP 서버 구현

- Recv의 구현
  - overapped 구조체의 확장

```
struct EXOVER {
        WSAOVERLAPPED   overlapped;
        WSABUF          wsabuf;
        bool            is_send;
        CHAR            io_buf[MAX_BUFF_SIZE];
        unsigned char   packet_buf[MAX_PACKET_SIZE];
        int             curr_packet_size;
        int             stored_packet_size;
};
```

IOCP 서버 구현

- Recv의 구현

```
int rest = io_size;
CHAR *buf = overlapped->io_buf;
int packet_size = overlapped->curr_packet_size;
int old_received = overlapped->stored_packet_size;
while(0 != rest) {
          if (0 == packet_size) packet_size = buf[0];
          int required = packet_size - old_received;
          if (rest >= required) {  // 패킷 제작 가능
                    memcpy(overlapped->packet_buf + old_received, buf, required);
                    ProcessPacket(id, overlapped->packet_buf);
                    packet_size = old_received = 0;
                    buf += required;
                    rest -= required;
          } else { // 패킷을 완성하기에는 데이타가 부족
                    memcpy(overlapped->packet_buf + old_received, buf, rest);
                    old_received += rest;
                    rest = 0;
          }
} // 패킷 조립 while 루프
overlapped->curr_packet_size = packet_size;
overlapped->stored_packet_size = old_received;
DWORD recv_flag = 0;
WSARecv(players[id].my_socket, &players[id].recv_over_ex.wsabuf, 1, NULL, &recv_flag,
                    &players[id].recv_over_ex.overlapped, NULL);
```

3번째 시간

- 지금까지
  - 서버 코드 완성
  - 컴파일에러
    - 타입 문제
  - 클라이언트 먹통
    - 어떠한 에러가 발생했는지 알아 보아야 한다.
    - send, recv에 error 체킹을 넣어보자.
  - 여러가지 실습 오류
    - Windows socket API 호출 시 에러 검사 필요

3번째 시간

- 네트워크 관련 에러 검출

```
void error_display(char *msg, int err_no )
{
	WCHAR *lpMsgBuf;
	FormatMessage(
		FORMAT_MESSAGE_ALLOCATE_BUFFER |
		FORMAT_MESSAGE_FROM_SYSTEM,
		NULL, err_no,
		MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
		(LPTSTR)&lpMsgBuf, 0, NULL );
	std::cout << msg
	std::wcout << L"에러 << lpMsgBuf << std::endl;
	LocalFree(lpMsgBuf);
}
```

3번째 시간

- 네트워크 관련 에러 검출

```
int ret = WSASend(g_clients[cl].client_socket,
        &over->wsabuf, 1, NULL, 0,
        &over->over, NULL);
if (0 != ret) {
   int err_no = WSAGetLastError();
   if (WSA_IO_PENDING != err_no)
   error_display("Error in SendPacket:", err_no);
}
```

3번째 시간

- 한글이 나오지 않는데???
  - 다음 추가

```
_wsetlocale(LC_ALL, L"korean");
```

```
std::wcout.imbue(std::locale("korean");
```

OLD_ERROR

- listen socket도 OVERLAPPED 소켓으로.
- Bind()대신 ::bind()

3번째 시간

- 문제해결
  - Accept할 때 INVALID_HANDLE이 나옴
    - 주소 구조체의 크기를 제대로 넣지 않음
  - 체스말이 이동하지 않음
    - GQCS에서 패킷을 받지 못함
    - **WSARecv시 초기화 한 flag을 넣어줘야 함**
    - WSASend시 WSABUF.len을 제대로 세트해야 한다.
    - 접속종료 처리시 continue;
    - Accept에서 WSARecv시 BUFCOUNT
    - MAX, MIN 오류
    - WSA_IO_PENDING 처리하지 않음
    - WSABUF의 크기는 1 이다.

3번째 시간

- Error의 원인
  - WSASend/WSARecv
    - 버퍼 카운트는 1로 한다.!!!

3번째 시간

- 다중 접속의 구현
  - 내가 접속했을 때 상대방이 보인다.
  - 내가 접속했을 때 상대방에게 내가 보인다.
  - 새로 접속하는 상대방이 보인다.
  - 나의 움직임이 상대방에게 보인다.
  - 접속을 끊은 상대방이 사라진다.
  - 내가 접속을 끊으면 상대방에게서 사라진다.

3번째 시간

- 다중 접속의 구현
  - 1. ACCEPT에서
    - 새 플레이어의 위치를 다른 모든 플레이어에게 전송
    - 다른 플레이어의 위치를 새 플레이어에게 전송
  - 2. Process_Packet에서
    - 이동할 때 마다 새 좌표를 모든 플레이어에게 전송
  - 3. Worker_Thread에서
    - 접속 종료 시 다른 모든 플레이어에게 알림

정리

- IOCP MMOG 서버의 뼈대 완성
- 완벽하지 않음
  - 멀티 쓰레드 버그
  - 최적화 필요
  - id 재사용 금지
  - 등등등

정리

- 이후의 내용
  - 멀티쓰레드 한번 더
  - 컨텐츠 구현
    - Quest, Item, Skill, 전투....
      - DB에 연동되는 것 말고 별다른 것 없으므로 PASS
    - 시야
      - 성능, 맵핵 방지
    - 충돌 처리
      - 해킹 방지
    - AI
      - 몇 십만 마리의 몬스터..., 스크립트 연동