**네트워크 게임 프로그래밍**

**추진 계획서**

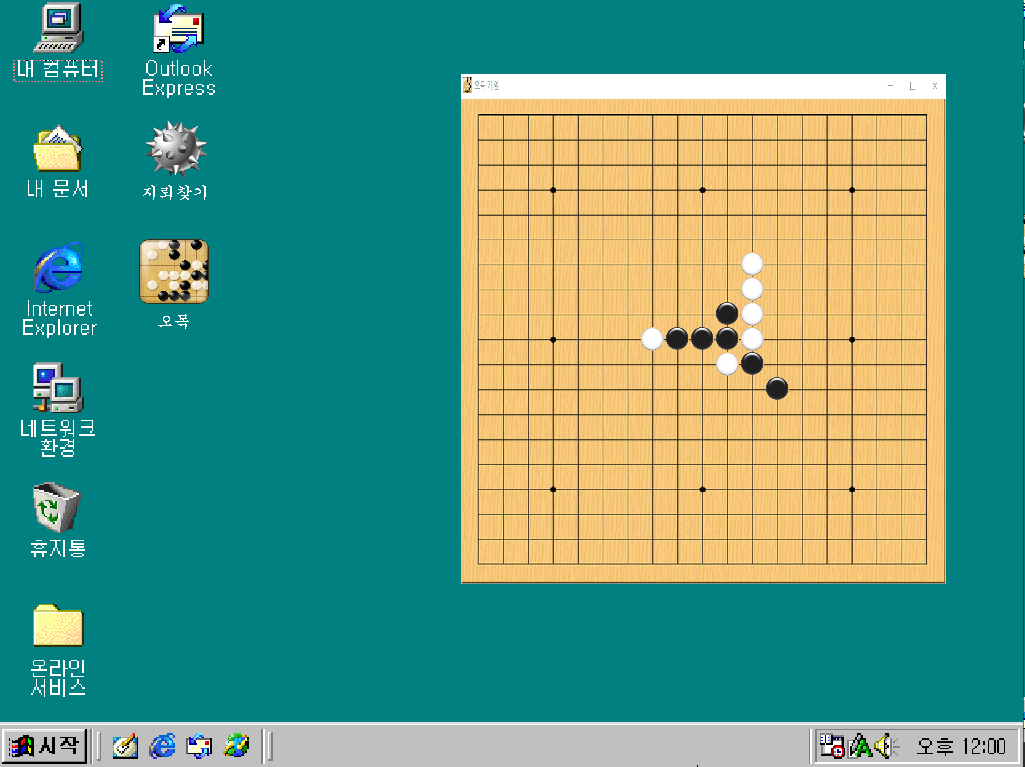
**<2019년도 네트워크게임프로그래밍 텀프로젝트>**

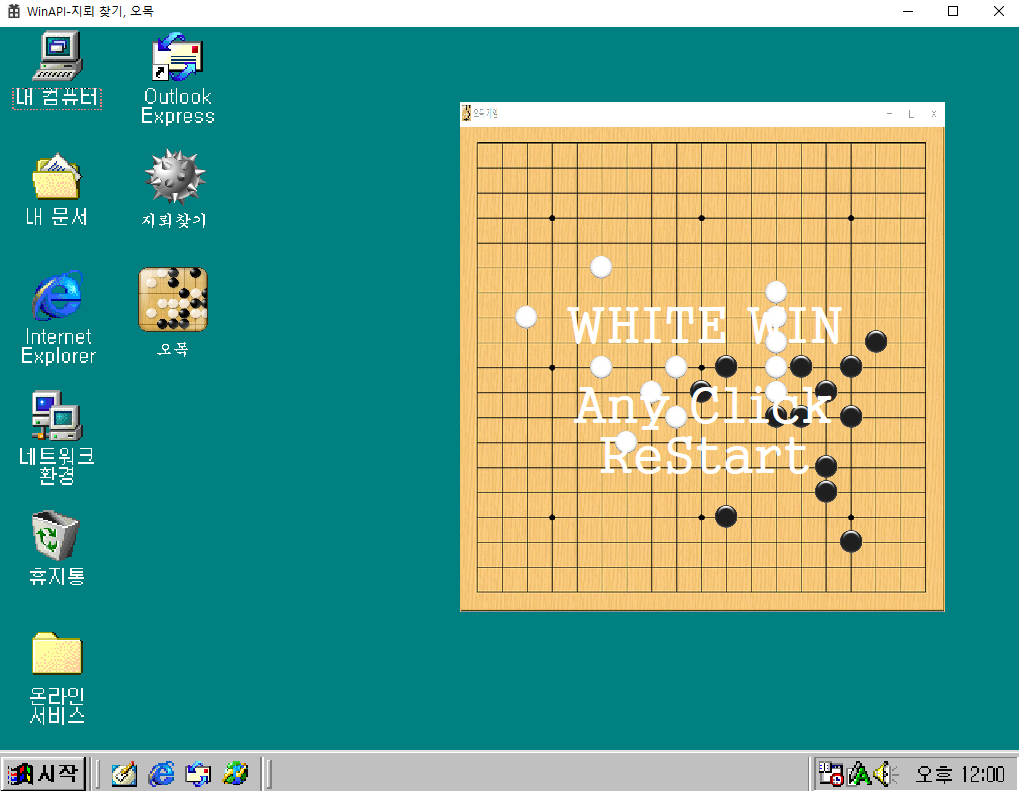
**차지환[2016184041]**

**목차**

* **게임 소개**
* **개발 환경**
* **개발 내용**
* **플로우 차트**
* **프로토콜 소개**
* **게임 시나리오.**

1. **게임 소개**

****

 **2인 플레이 오목 게임.**

**2. 개발환경**

**A. 플랫폼**

PC, Windows 10 이상 버전

**B. IDE, API**

마이크로소프트 비주얼 스튜디오 2019 사용.

윈도우 API

**3. 개발내용**

**1.주요 기능 목록**

**-Socket 모델**

* + - * **서버 : select 모델.**
      * **클라이언트 : non-blocking 소켓 모델**
  + **멀티 스레드를 이용한 유저 쫓아 내기와 시간 업데이트**
  + **고정 길이 + 가변길이 방식.**
  + **각 종 소켓 옵션 적용.**
  + **사용 프로토콜 : TCP**

**2.적용 내용 표.**

|  |  |  |  |
| --- | --- | --- | --- |
| **사용 기능** | **적용 내용** | **적용 파일** | **비고** |
| **TCP** | **서버 모든 데이터 송수신** | **OmokServer.cpp** |  |
| **Select** | **서버 모든 데이터 송수신** | **OmokServer.cpp**  **( 126 ~ 281 라인)** |  |
| **소켓 옵션** | **No delay 옵션**  **Linger 옵션** | **OmokServer.cpp**  **(103 ~ 124 라인)** |  |
| **Non-blocking socket** | **클라이언트 모든 데이터 송수신** | **OmokClient.cpp**  **(357 ~ 361 라인)** | **클라이언트에서 사용** |
| **멀티 스레드** | **타이머 스레드**  **/킥 스레드(유저 내 쫓아내는 전용 스레드)** | **OmokServer.cpp**  **(452 ~ 479 라인)**  **/**  **OmokServer.cpp**   1. **514 라인)** |  |

**3.상세 설명..**

**기존 수업과 PPT에 에서 사용한 코드를 최대한 활용하는 방안으로 클래스로 래핑 하고 부족한 부분의 경우 추가로 기능을 덧붙여서 사용했다.**

**소켓 모델.**

**서버**

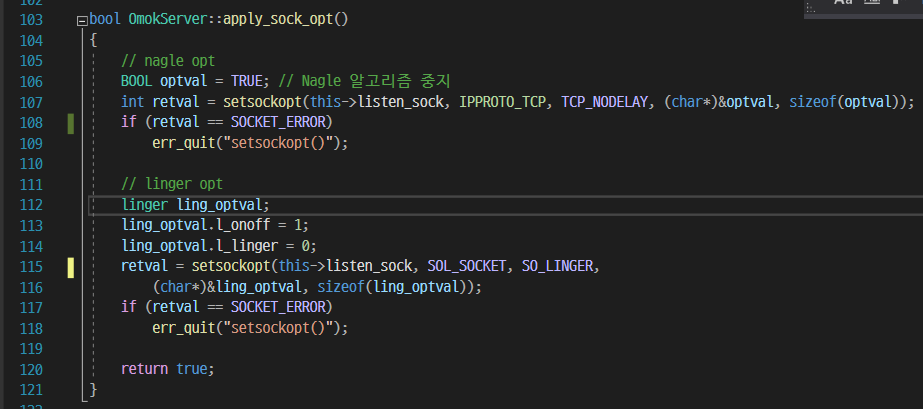
1. **select모델 사용 ( OmokServer.cpp**

**멀티 스레드 없이 소규모로 IO를 진행하기 위해서 select모델을 사용하였다.**

**클라이언트**

1. **클라이언트의 경우 단일 세션을 유지하기만 하면 되므로 non-blokcing 모델을 사용하였다.**

**소켓 옵션. ( OmokServer.cpp 103 ~ 121 라인 )**

****

1. **Linger 옵션**

**서버에셔 연결을 끊은 후 세션을 빨리 정리할 수 있도록 Linger옵션을 적용 했다.**

1. **No Delay 옵션**

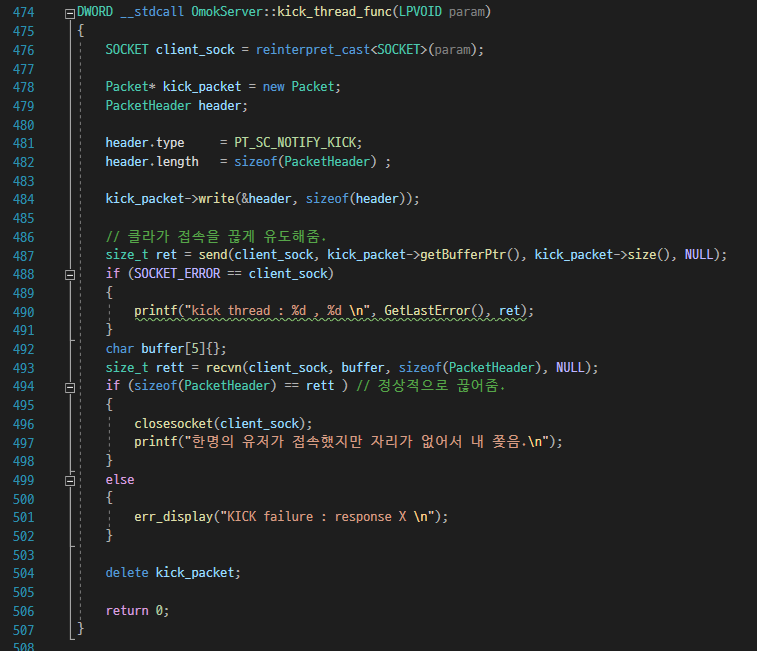
**네이글 알고리즘 옵션을 끄면 데이터를 모아서 보내지 않아 트래픽은 늘어나지만 빠른 반응 속도를 위해서 Nagle옵션을 껐다.**

**스레드 역할**

**스레드는 메인 스레드 외에 두 종류를 만들어서 사용하고 있다.**

* + - **Kick Thread**

**스레드 함수 : ( OmokServer.cpp 의 kick\_thread\_func() )**

****

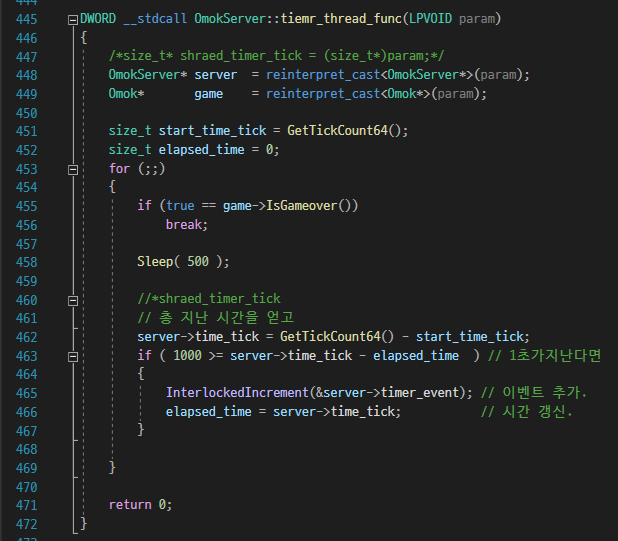
**서버의 메인 스레드에서 이미 두 명이 들어와서 게임이 시작한 상태라면 추가적으로 접속한 인원을 받지 않고 Kick Thread를 만들고 유저를 안전하게 접속종료 시키도록 유도함.**

* + - * **접속 종료 동작**

**유저에게 종료용 패킷을 보내고 그 종료 패킷을 클라이언트가 잘 받았는지에 대한 응답 패킷을 받고 종료 시킨다.**

* + - **Timer Thread**

**스레드 함수 : ( OmokServer.cpp 의 timer\_thread\_func() )**

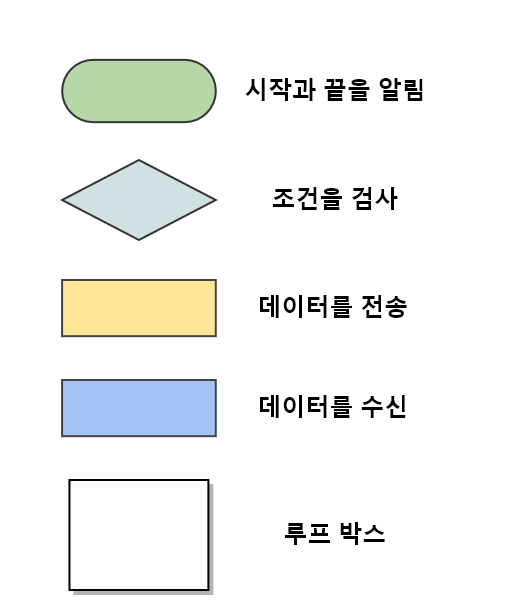
****

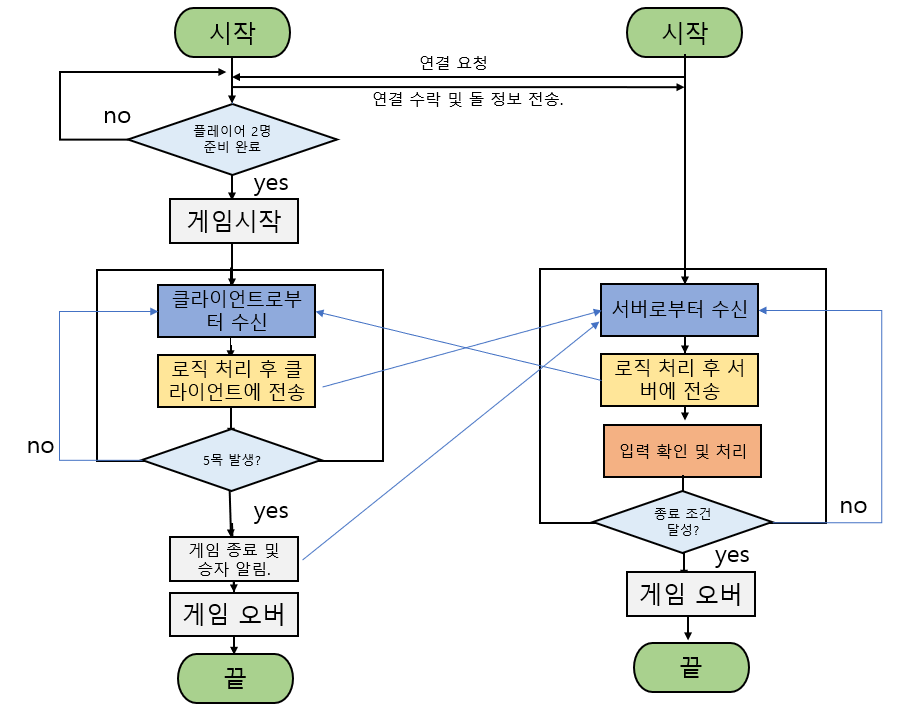
**주기적으로 시간 갱신을 해주는 업데이트를 따로 스레드를 만들어 작업 배분을 시키고 있다.**

**게임이 종료되면 알아서 스레드를 종료 시키고 있다.**

**4. 플로우 차트**

#플로우 차트 설명



****

#서버 흐름

1. 서버를 실행시키고 게임을 시작하기 전 클라이언트의 연결 요청을 기다린다.

2. 연결이 완료되면 먼저 온 순서대로 클라이언트에게 돌(흑 돌, 백 돌 순)을 부여한다.

3. 두 명이 들어오면 게임을 시작한다.

4. 만약 이 이후에 인원이 접속을 한다면 접속한 인원을 킥 스레드로 보내 인원이 꽉 차서 더 이상 받지 못한다는 메시지를 교환하고 접속을 종료 시킨다.

5. 클라이언트한테 돌을 두었다는 메시지를 받고 해당 클라이언트의 차례에 돌을 둔 것인지와 또 돌을 둘 수 있는 위치인지에 따라 처리할지 무시할지 판단한다.

6. 해당 클라이언트의 차례여서 무시하지 않고 적용을 했다면 오목이 있는지 판단한다.

7. 오목이 낫다면 게임 중인 플레이어들에게 게임 결과 패킷을 전송하고 게임을 끝낸다.

8. 없다면 루프를 돌면서 계속 게임을 진행하고 있다면 결과를 모든 클라이언트에게 전송을 하고 게임을 종료한다.

#클라이언트 흐름

1. 게임을 키고 서버 아이피를 입력 후 서버에 연결 요청을 시도한다.

2. 접속이 됐다면 서버로부터 받는 메시지로 게임이 가능한지 판단 함

3. 게임이 가능하다면 받은 메시지로부터 돌 종류를 파악 후, 플레이어 2명이면 게임을 시작.

4. 게임이 불가능 하다면 인원이 다 찾다는 메시지를 받고 답장을 하여 종료 메시지를 받았다고 알린 후 서버에서 처리 된 후 종료 된다.

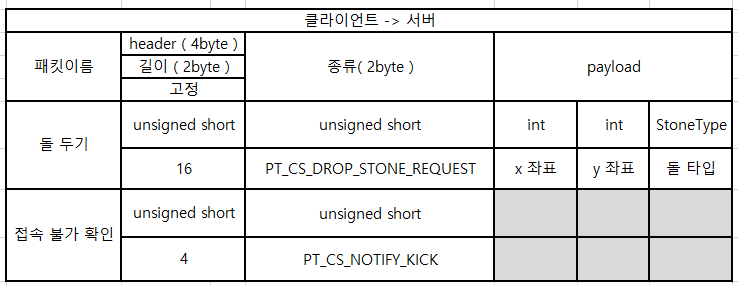
5. 자신의 차례면 돌을 두어서 게임을 진행한다.

6. 클라이언트가 서버로부터 게임결과 데이터를 전송 받기 전이라면 위 작업을 계속해서 진행한다(loop).

7. 오목이 나서 게임이 끝나는 상황이면 서버가 게임 결과 패킷을 보내줘 처리하고 승자를 화면에 띄운다.

**5.프로토콜 소개**

****

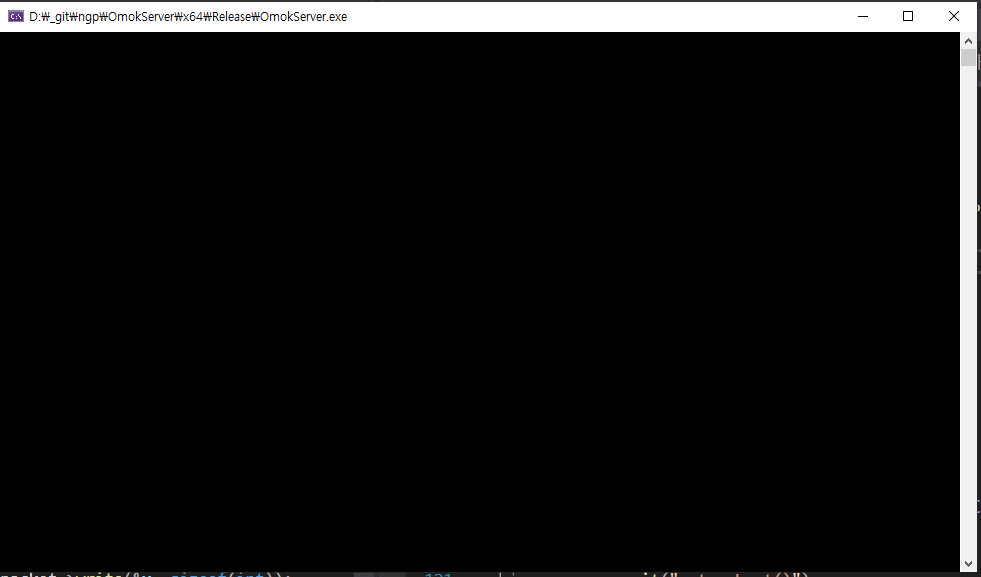
****

**\*StoneType이 코드상에선 StoneStatus로 대체되었습니다.**

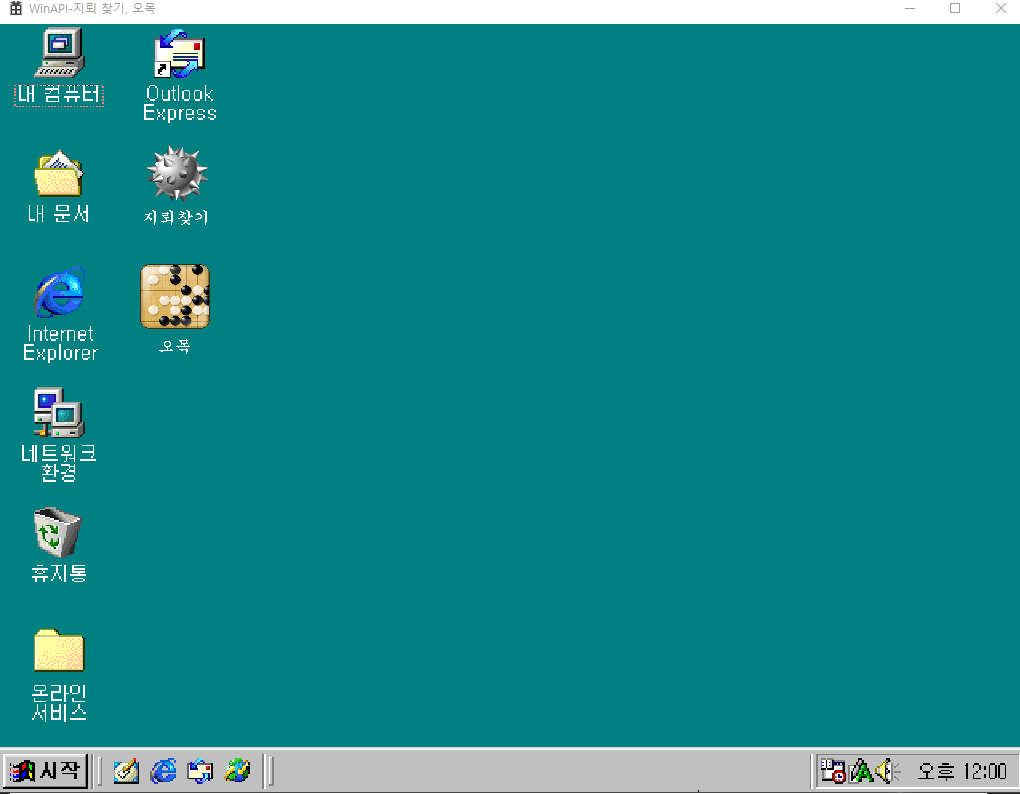
**6.게임 시나리오**

**.**

**서버를 킨다.**

****

**두 명이 게임을 킨다.**

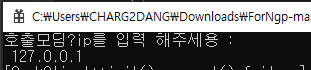
****

**두 명이 게임을 킨다.**

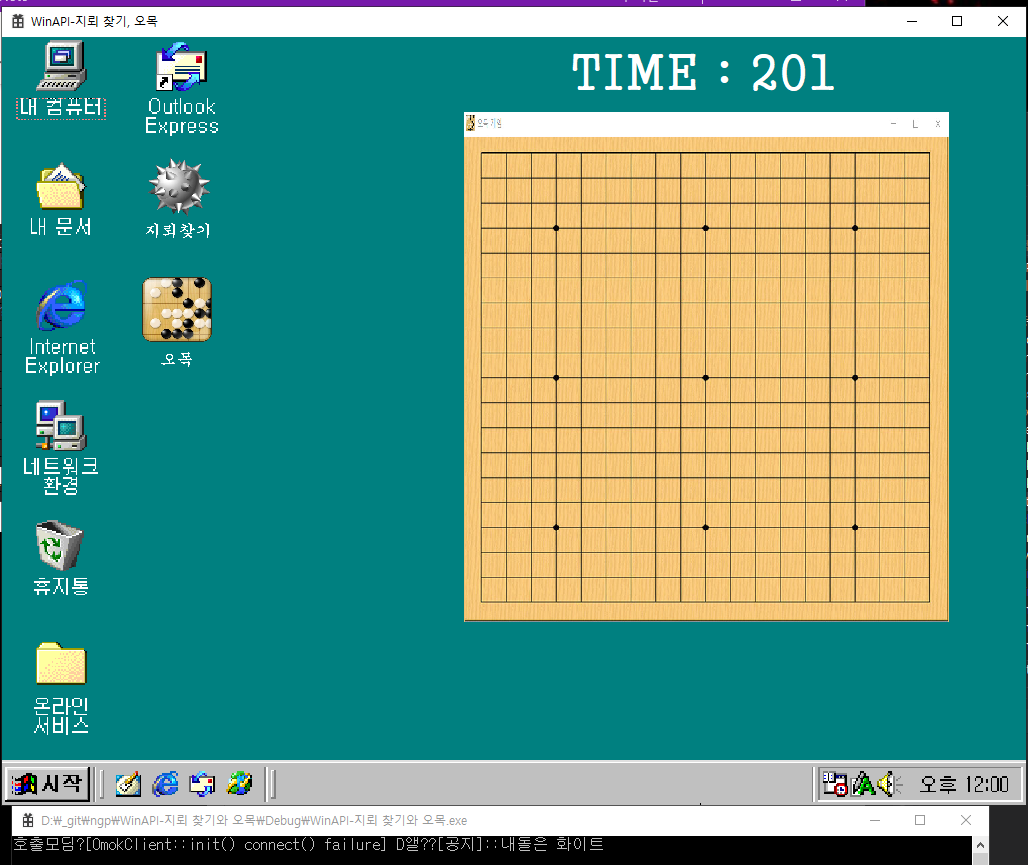
**실제 네트워크 접속은 오목 버튼을 누러야 한다.**

****

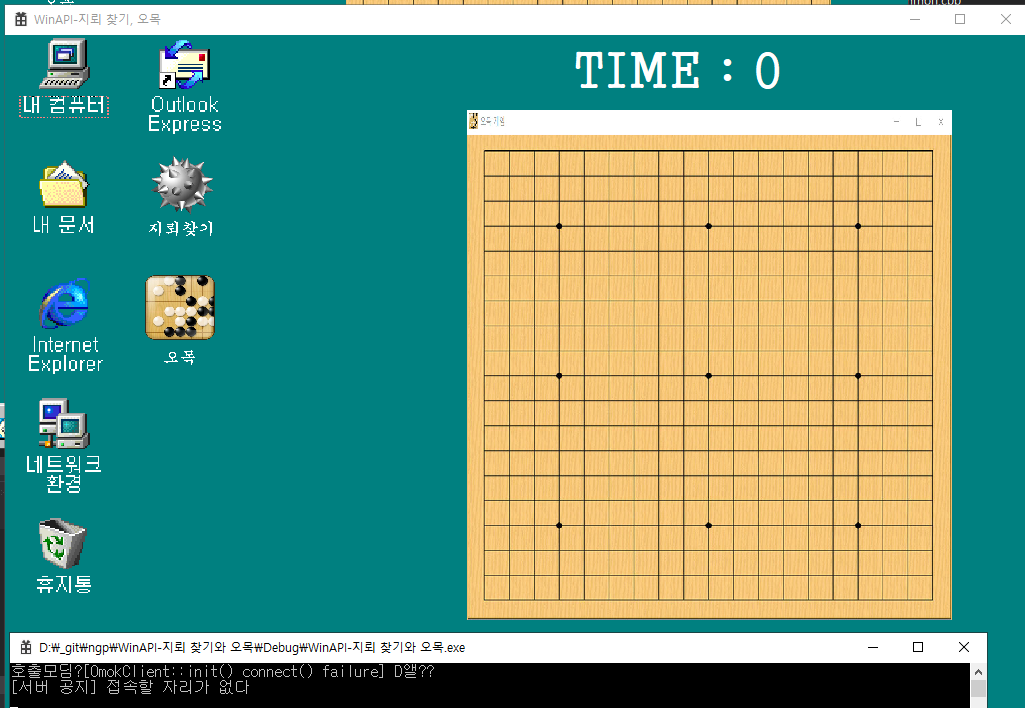
**클릭시 오목판이 열린다.**

****

**그리고 오목 클라이언트 뒤에 있는 게임과 같이 켜진 콘솔창에 아이피를 입력한다.**

****

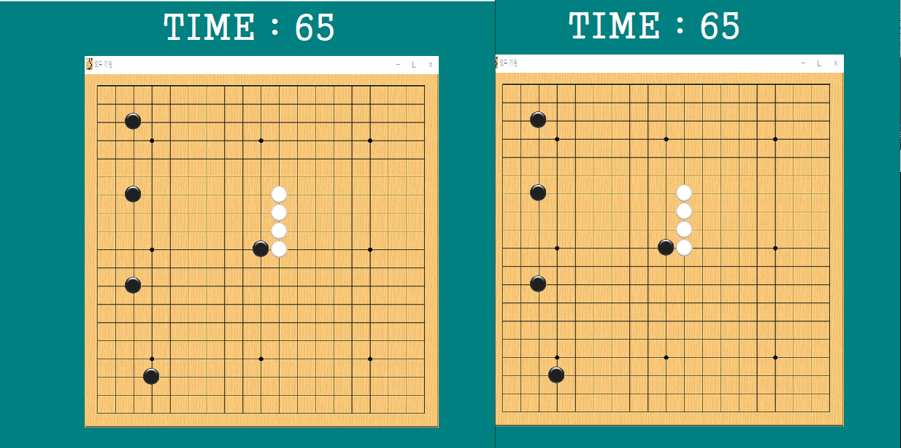
**입력후 서버에 제대로 접속했다면 서버 시간과 동기화 되고 콘솔창을 보면 자신이 백 돌인지 흑 돌인지 알려준다.**

****

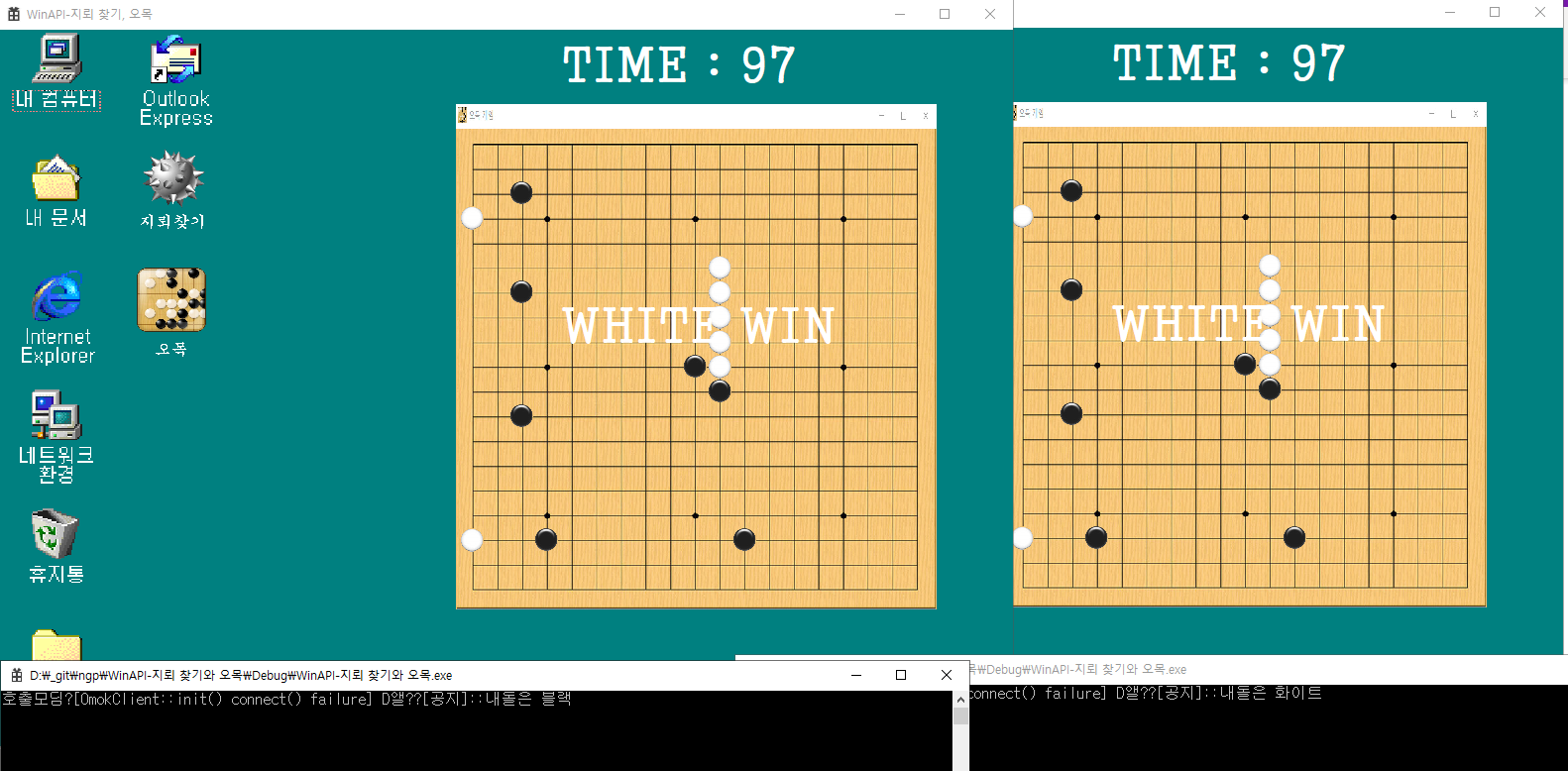
**만약 기존에 두 명 이상 들어온 상황이라 서버에 자리가 없다면**

**콘솔창에 자리가 없어서 서버에 못 들어 간다는 알림이 뜨고 서버에서 접속을 끊어준다.**

**흑 돌부터 돌을 둘 수 있고 자기차례가 아닌데 돌을 놀려 해도 무시 된다.**

****

**각자 차례에 돌을 두면 된다.**

****

**이렇게 해서 진행하다가 서버에서 결과가 낫다고 판단 되면 승자를 판정해줘서 결과를 받아 화면에 뿌려준다.**