**고급 프로그래밍 Term Project**

**설계과제 보고서 (캐치마인드)**

2팀

2010152003 김민수

2012154021 문동선

목차

1. 개요
2. 설계
3. 역할분담
4. 접속 방법 (X server 연동)
5. 프로그램 실행 방법
6. 프로그램 실행 화면
7. 성능 비교 및 분석
8. Raw code
9. **개요**

유명 온라인 게임 캐치마인드를 모티브로 한 프로그램이다.

캐치마인드는 팀원이 그린 그림을 보고 단어를 추측하는 게임이다. 게임에 참여한 사람들 중 한 명이 문제의 출제자가 된다. 무작위 키워드가 제시되고 출제자가 제시어를 캔버스에 그림으로 그려서 묘사하면, 나머지 인원들은 그림이 의미하는 바를 맞춰야 한다.

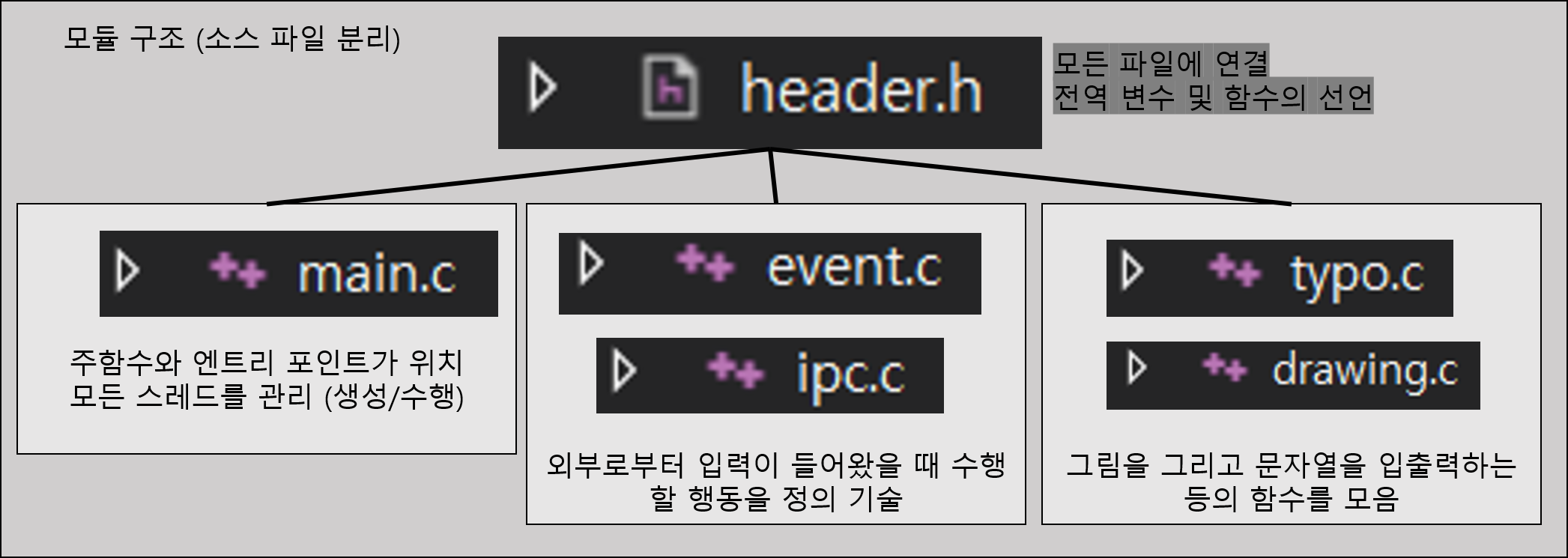
문제를 맞히는 측은 출제자가 그림을 그리는 과정을 실시간으로 지켜볼 수 있어야 하며 정답을 입력하면 프로그램은 종료한다.

1. **설계**

프로그램은 기본적으로 문제를 맞히는 reader와 제시어를 입력하고 그림을 그리는 writer의 두 부분으로 나뉜다. 두 프로그램은 유닉스 서버에서 실행하며 IPC 기법을 이용하여 실시간으로 상호 통신한다.

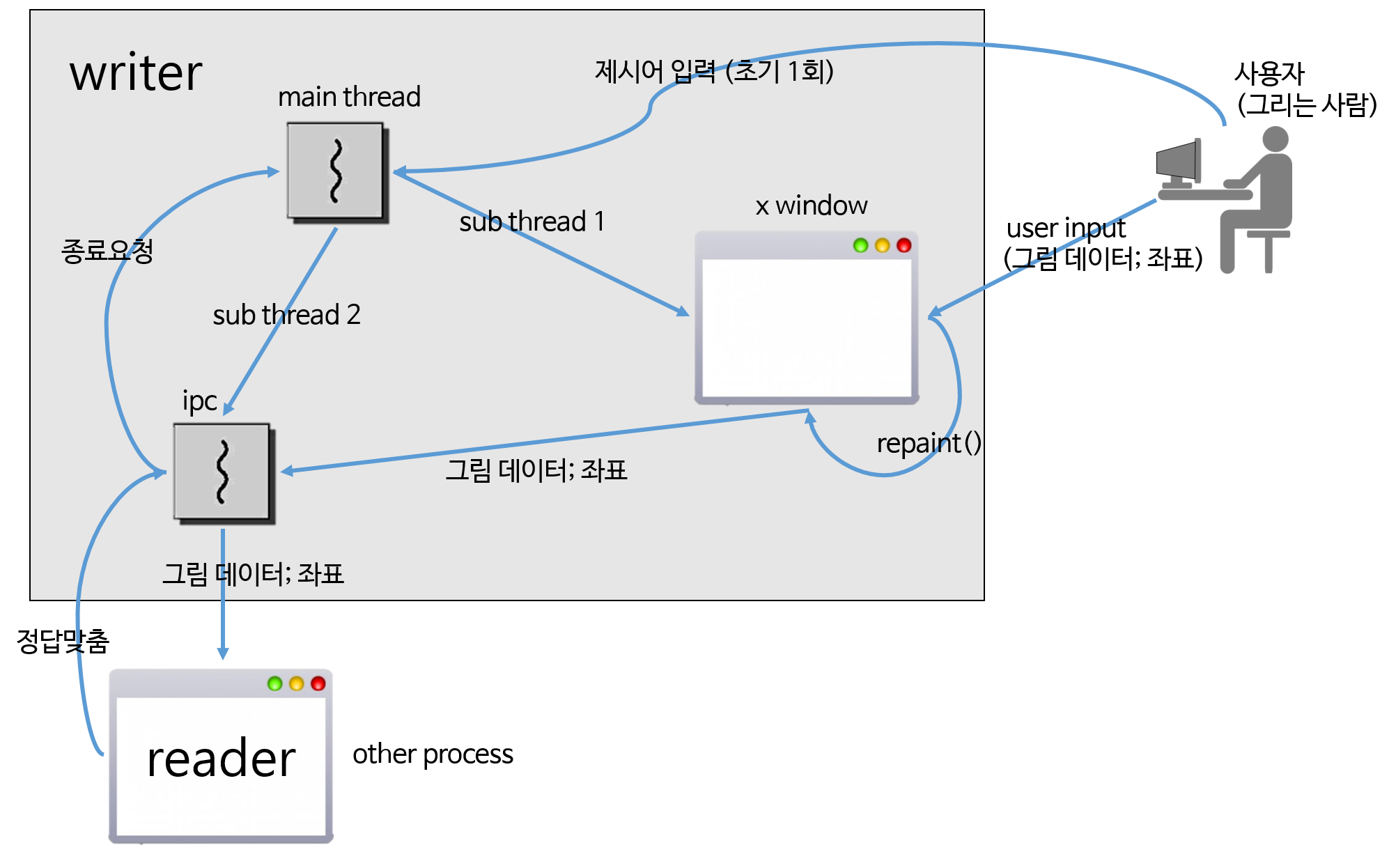
각 프로그램은 3개의 스레드를 가지고 있다. 메인 스레드는 모든 서브 스레드를 생성 및 관리하며 x window 창을 띄우고 유저의 TUI상 입력을 받는다. 메인 스레드 이외에 x window에 대한 입력을 받고 각 이벤트에 대한 처리를 수행하는 스레드와 다른 프로세스로부터의 ipc 메시지를 읽어 들이기 위한 스레드가 있다.

1. **모듈 구성 (파일)**

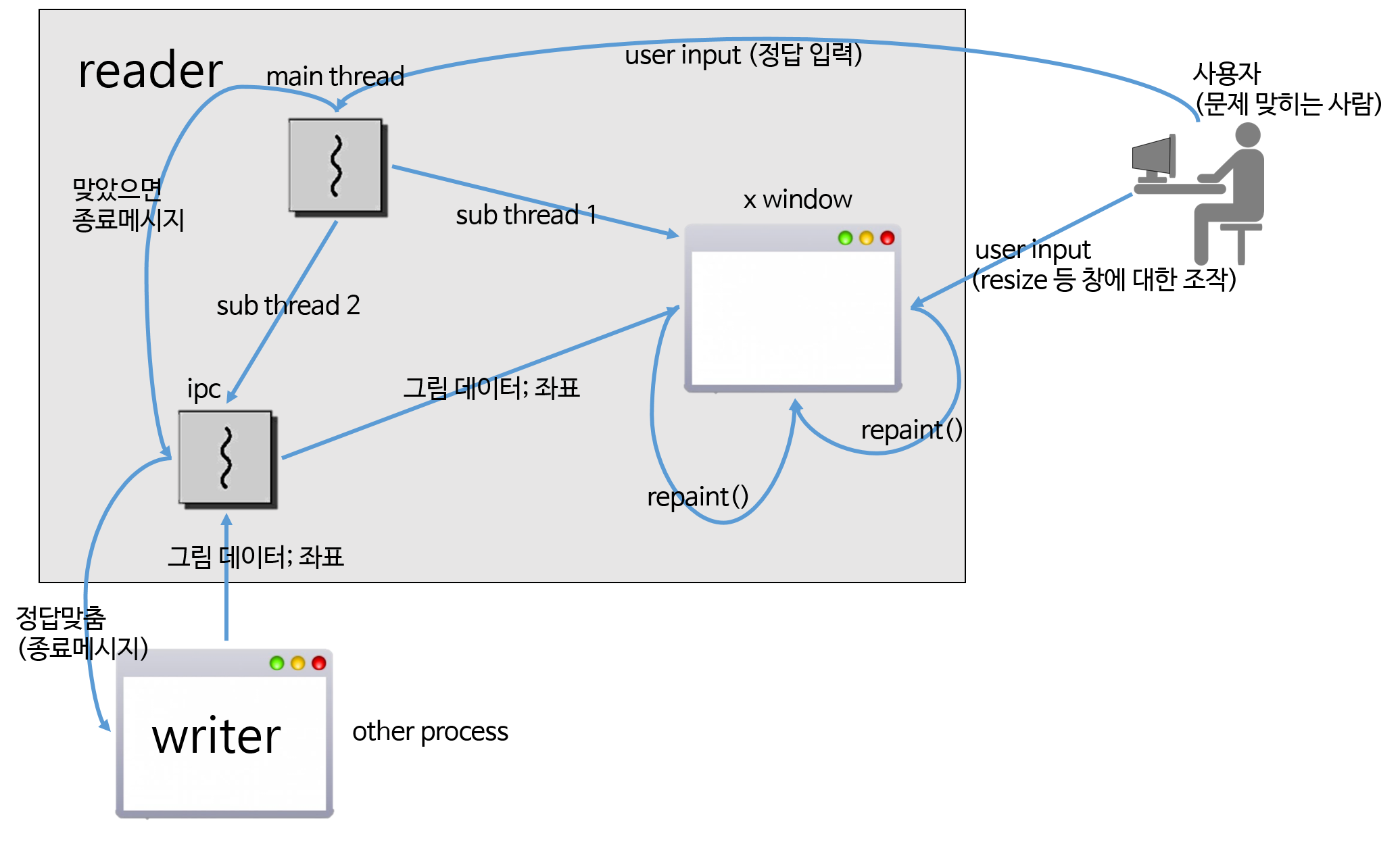


* + header.h 파일: 다른 모든 파일에 연결된다. 필요한 라이브러리를 모두 include하고 전역변수 및 함수의 원형들을 전부 선언한다.
  + main.c 파일: 메인 스레드를 실행하고 나머지 모든 스레드를 관리하며 프로그램 초기화와 사용자의 키입력을 처리한다.
  + event.c 파일: 서브 스레드 1의 행동이 정의되어 있다. x window 시스템으로 생성된 창에 대한 유저의 조작이 있을 때 이벤트를 감지하고 해당하는 이벤트 핸들러를 통해 이벤트에 대한 처리를 수행한다.
  + ipc.c 파일: 프로그램이 종료할 때까지 반복하면서 ipc를 통해 수신된 메시지를 받아 읽고 메시지에 따라 필요한 처리를 호출하는 서브 스레드 2의 처리루틴을 정의한다.
  + typo.c 파일: 사용자가 콘솔에 입력하는 것에 관련한 함수들만이 정의되어 있다. 필요할 때 다른 파일에서 이 함수의 파일을 호출하게 된다.
  + drawing.c 파일: 윈도우 창을 초기화하고 다시 그리는 것에 관련한 함수들만이 정의되어 있다.

1. **writer 프로그램 모델**



1. **reader 프로그램 모델**



1. **역할분담**

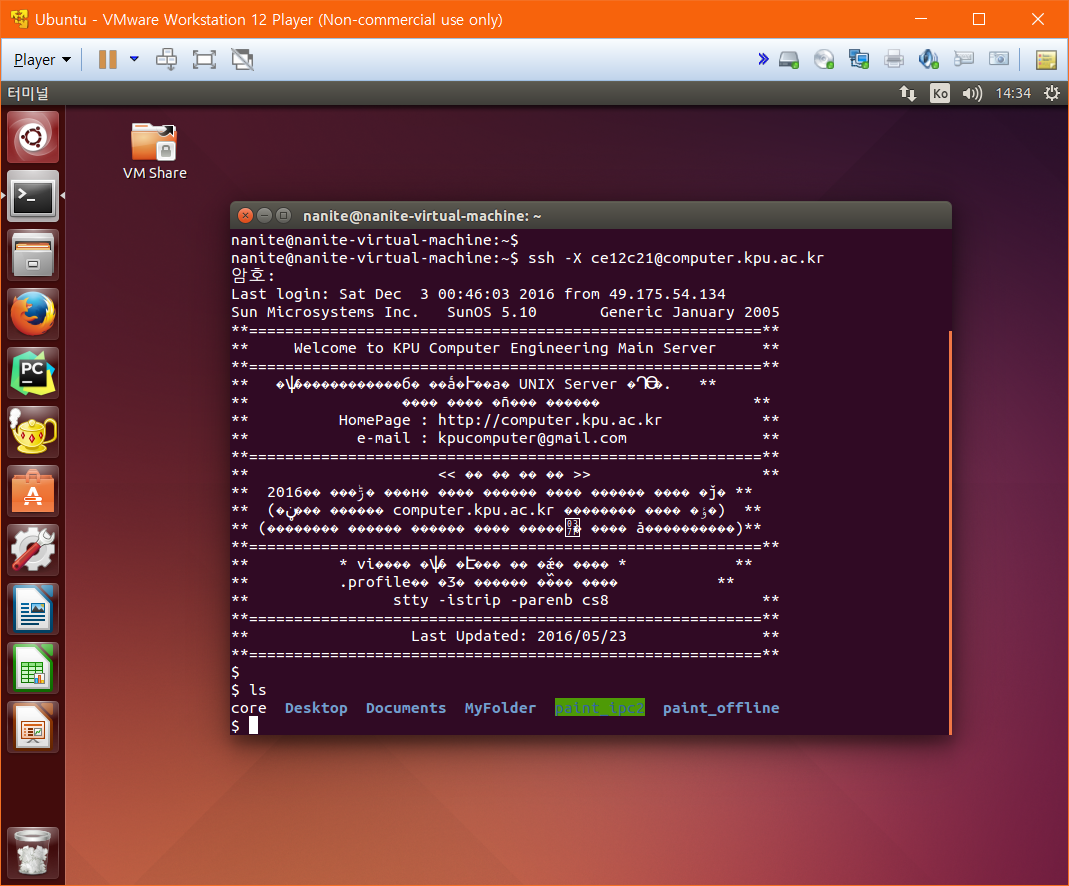
문동선 : 모듈화 및 메이크 파일 작성, 버전 관리, Advanced IPC 기법 (Message passing) 을 이용한 통신 구현

김민수 : 그리기 구현, 메이크 파일 작성, 파이프 (Named FIFO) 통신 구현

1. **접속 방법 (X server 연동)**
2. **리눅스 등 유닉스 계열 데스크탑 실행 환경**

리눅스는 X window 시스템을 기본 지원하기 때문에 터미널을 열고 ssh 명령어로 유닉스 서버에 접속하기만 하면 된다. 명령어는 다음과 같다.

ssh -X (계정명)@computer.kpu.ac.kr



1. **윈도우즈 환경**

윈도우즈 환경에서는 리눅스에 원격으로 접속할 때 주로 PuTTY를 사용한다. PuTTY는 기본적으로 CUI만을 지원하지만 Xming과 연동하여 X windows 환경을 이용할 수 있다.

Xming을 다운로드 받고 실행하는 과정은 다음과 같다.

Xming 은 여기서 다운로드 받을 수 있다. - http://sourceforge.net/projects/xming/

6.9버전을 내려 받을 수 있다.

**1. Xming을 다운로드 받고 설치한다.**

설치는 매우 간단하게 Next 버튼 몇 번 클릭하면 끝난다.

Xming을 설치하면 Xming 실행 관련 프로그램이 두 개가 생긴다.

하나는 Xming 또 다른 하나는 XLaunch이다.

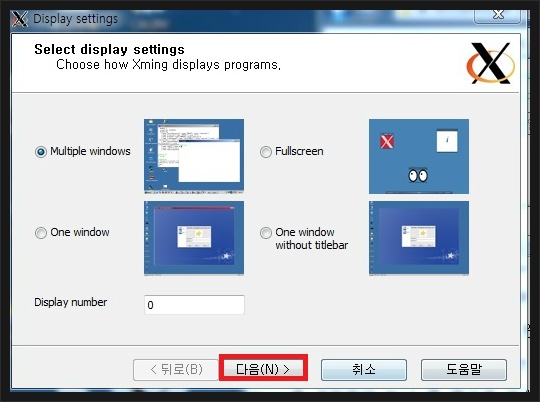
Xming은 실제 Xming을 실행시키는 프로그램이고, XLaunch는 화면 설정 등을 할 수 있다.

Xming을 실행시키고 XLaunch를 실행시키면 아래와 같은 Display settings 창이 뜬다.

**2. Display settings에서 디스플레이 설정을 할 수 있다.**

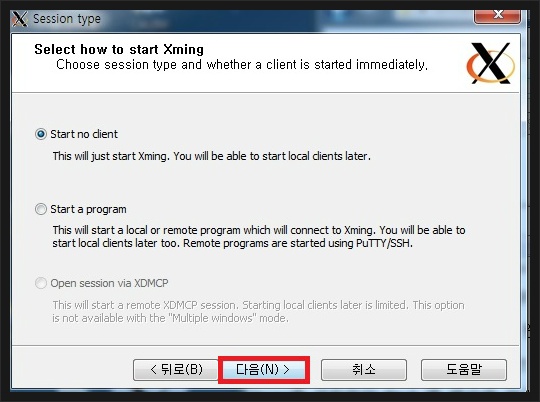
리눅스 서버에 원격 연결한 PuTTY에서 원하는 프로그램을 X-Windows 환경으로 띄우기에는

Multiple windows가 적당한 것 같다.



**3. 바로 창을 실행할지 선택하는 화면**

Start no client를 클릭하고 나중에 푸티와 연동해서 사용하면 된다.

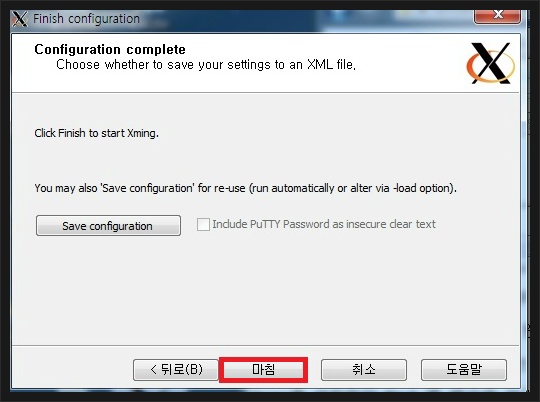


**4. Putty와 연동해서 프로그램을 그래피컬하게 띄우려는게 목적이기 때문에**

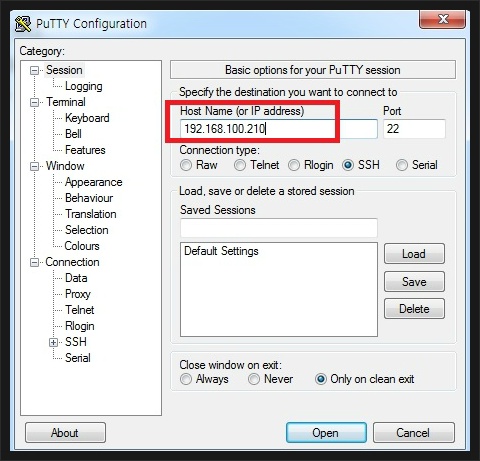
**그냥 다음을 클릭한다.**



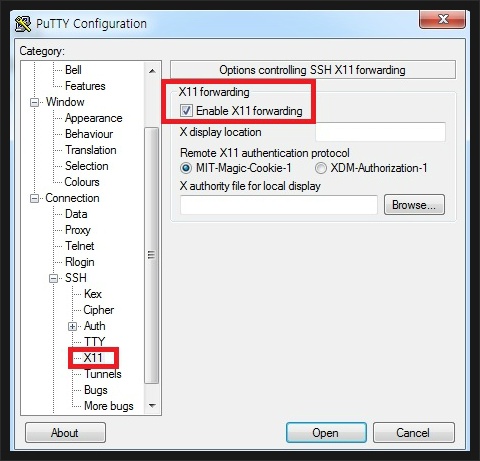
**5. 설정이 다 되었다. 마침을 클릭한다.**



**6. PuTTY를 실행하여 리눅스 서버에 원격 접속할 아이피를 입력한다.**

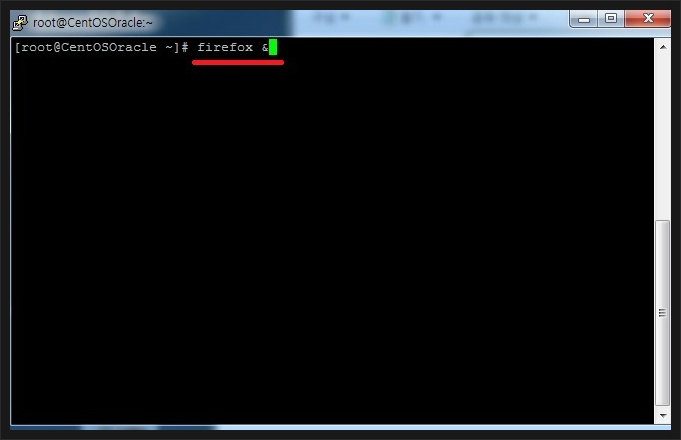


**7. Connection - SSH - X11을 클릭한다.**

**X11 forwarding에서 Enable X11 forwarding 버튼을 클릭한다.**

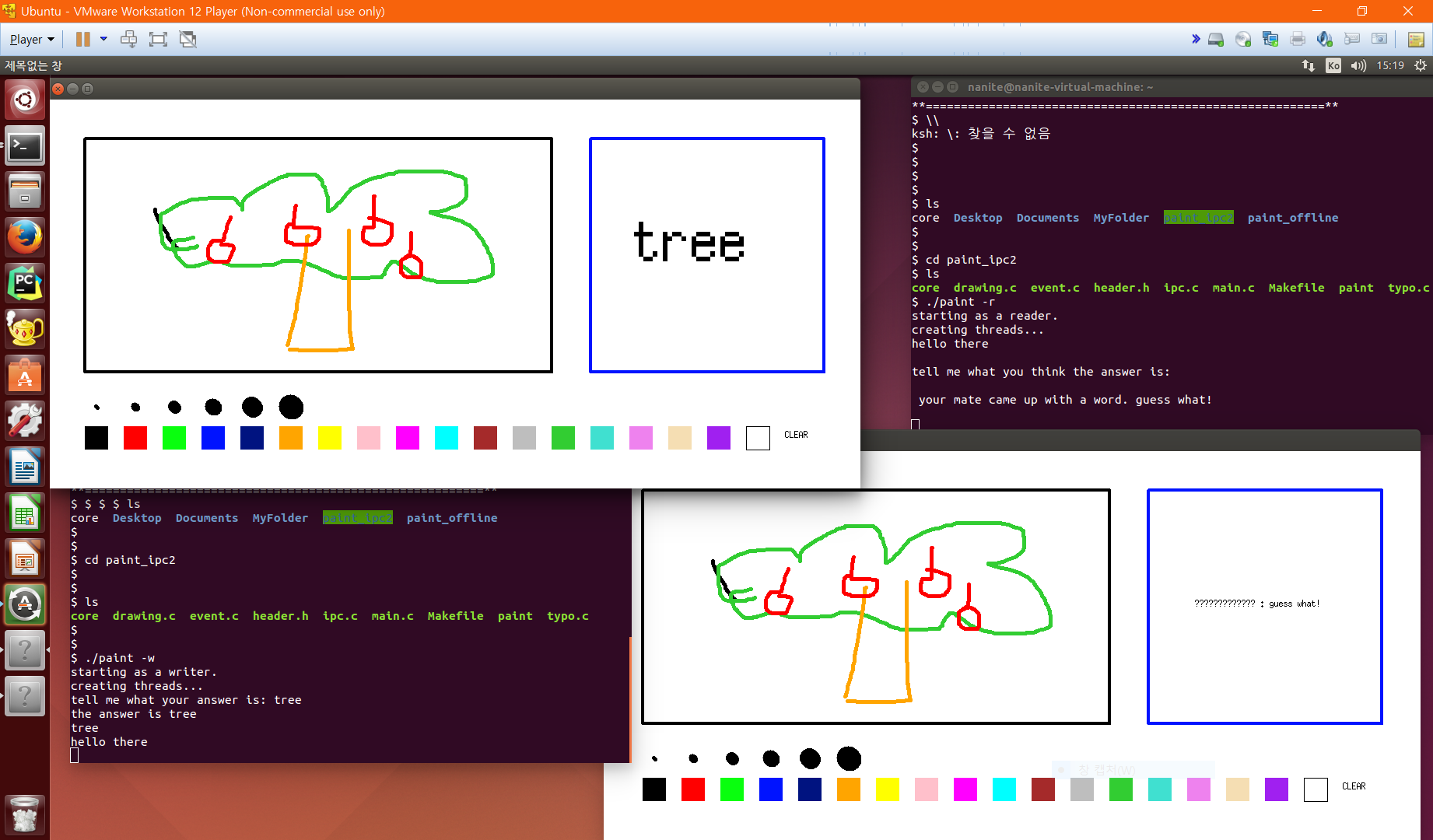
**8. 파이어폭스를 X Windows 환경으로 실행시켜보자.**

(호스트 컴퓨터에 Xming 프로그램이 실행되어 있는 상태여야 한다.)



**9. 파이어폭스가 X windows 환경으로 실행되었다.**(출처 : <http://klero.tistory.com/entry/Xming%EC%9D%84-PuTTY%EC%99%80-%EC%97%B0%EA%B2%B0%ED%95%B4%EC%84%9C-%EA%B0%84%EB%8B%A8%ED%95%98%EA%B2%8C-%EC%82%AC%EC%9A%A9%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95> )

1. **프로그램 실행 방법**



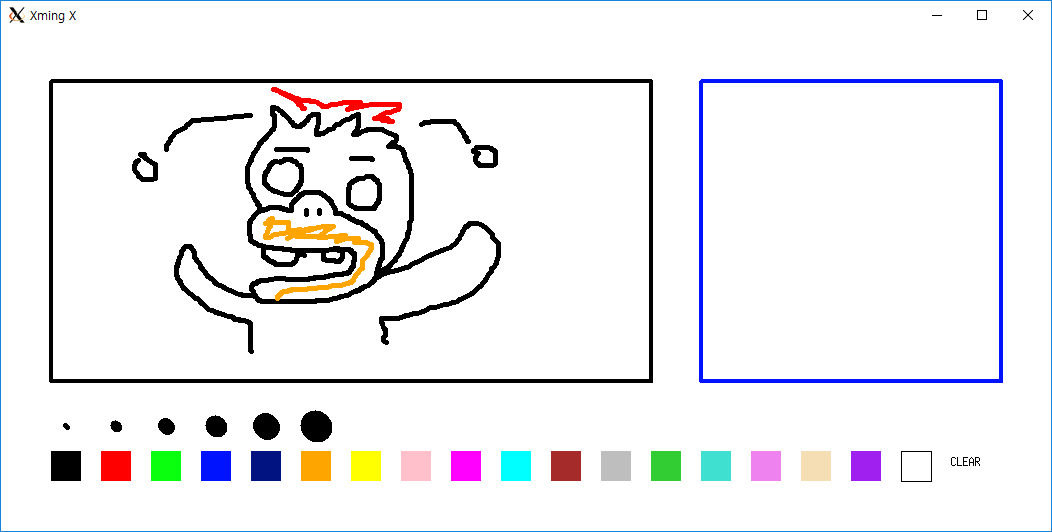
1. **Advanced IPC 기법(message passing)을 이용한 버전**
2. 유닉스 상에서 (computer.kpu.ac.kr) 프로그램(Advanced IPC 기법(message passing)을 이용한 버전)의 소스가 위치한 경로는 /u4/cestu/12/ce12c21/paint\_ipc2

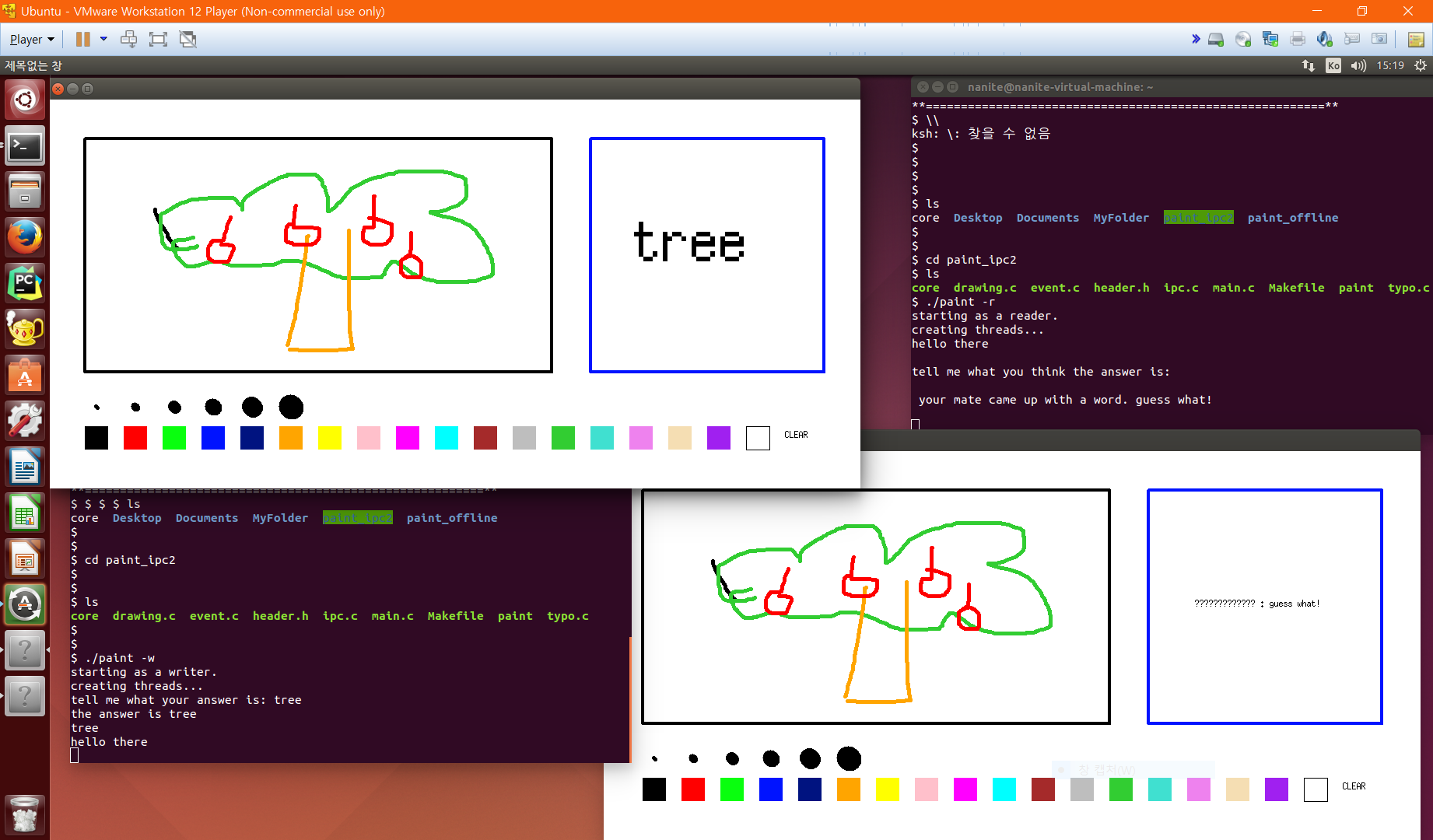
이다. 해당 디렉토리로 이동한 뒤 make 명령어를 수행하여 프로그램을 빌드한다.

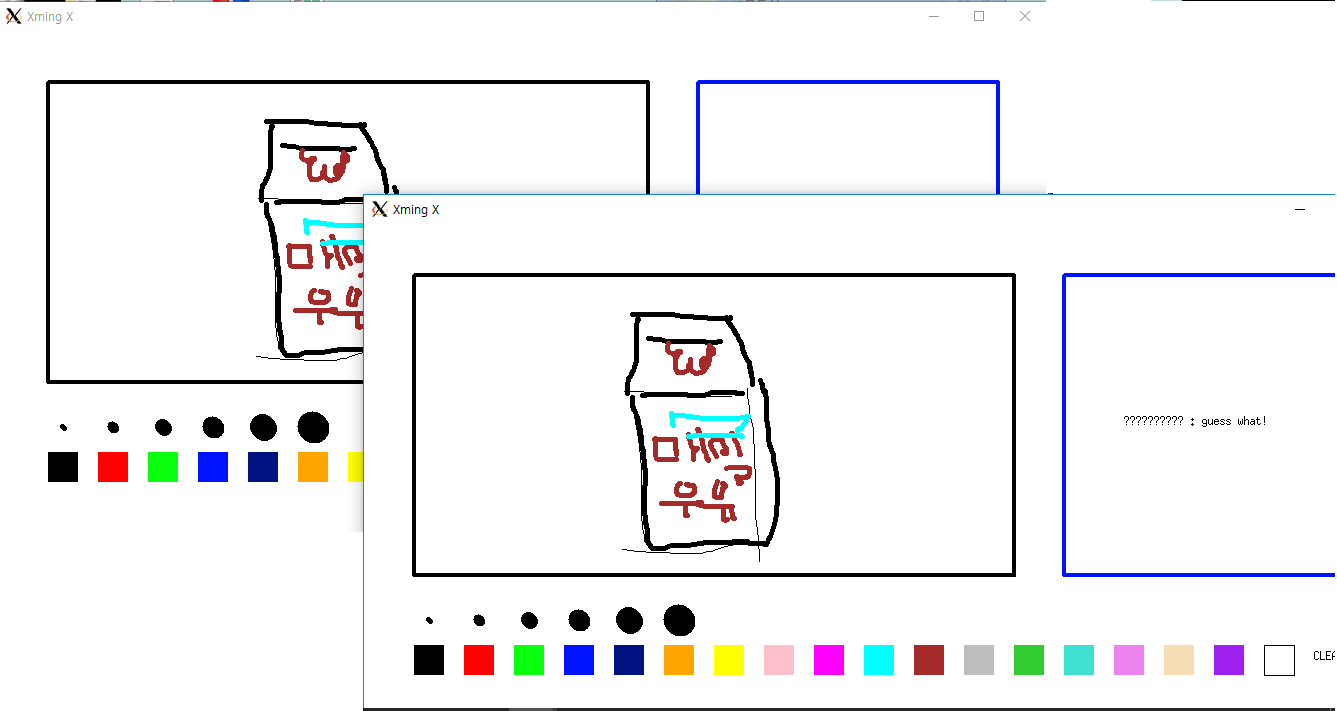
1. 최종 생성된 프로그램의 이름은 paint이다. 그리는 측은 -w, 문제를 맞추는 측은 -r 옵션을 주어 실행한다.
2. Advanced IPC 기법(message passing)을 이용한 버전에서는 ./paint -w를 먼저 실행하고 그 다음 ./paint -r를 실행해야 한다.
3. 그림을 그리는 측(writer)에서 제시어를 입력하면, 정답을 맞히는 측(reader) 프로세스에서 문제가 제출되었음을 알리는 문구를 확인할 수 있다.
4. writer가 그림을 그리면 reader는 그 과정을 실시간으로 확인한다. reader는 콘솔에 단어를 입력하고 확인 받을 수 있다. 단어가 정답과 일치하면 writer와 reader의 두 프로세스 모두 즉시 종료한다.
5. **파이프 통신(Named FIFO)을 이용하는 버전**
6. 유닉스 상에서 프로그램(파이프 통신(Named FIFO)을 이용하는 버전)의 소스가 위치한 경로는 /u4/cestu/10/ce10b003/Painter

이다. 해당 디렉토리로 이동한 뒤 make 명령어를 수행하여 프로그램을 빌드한다.

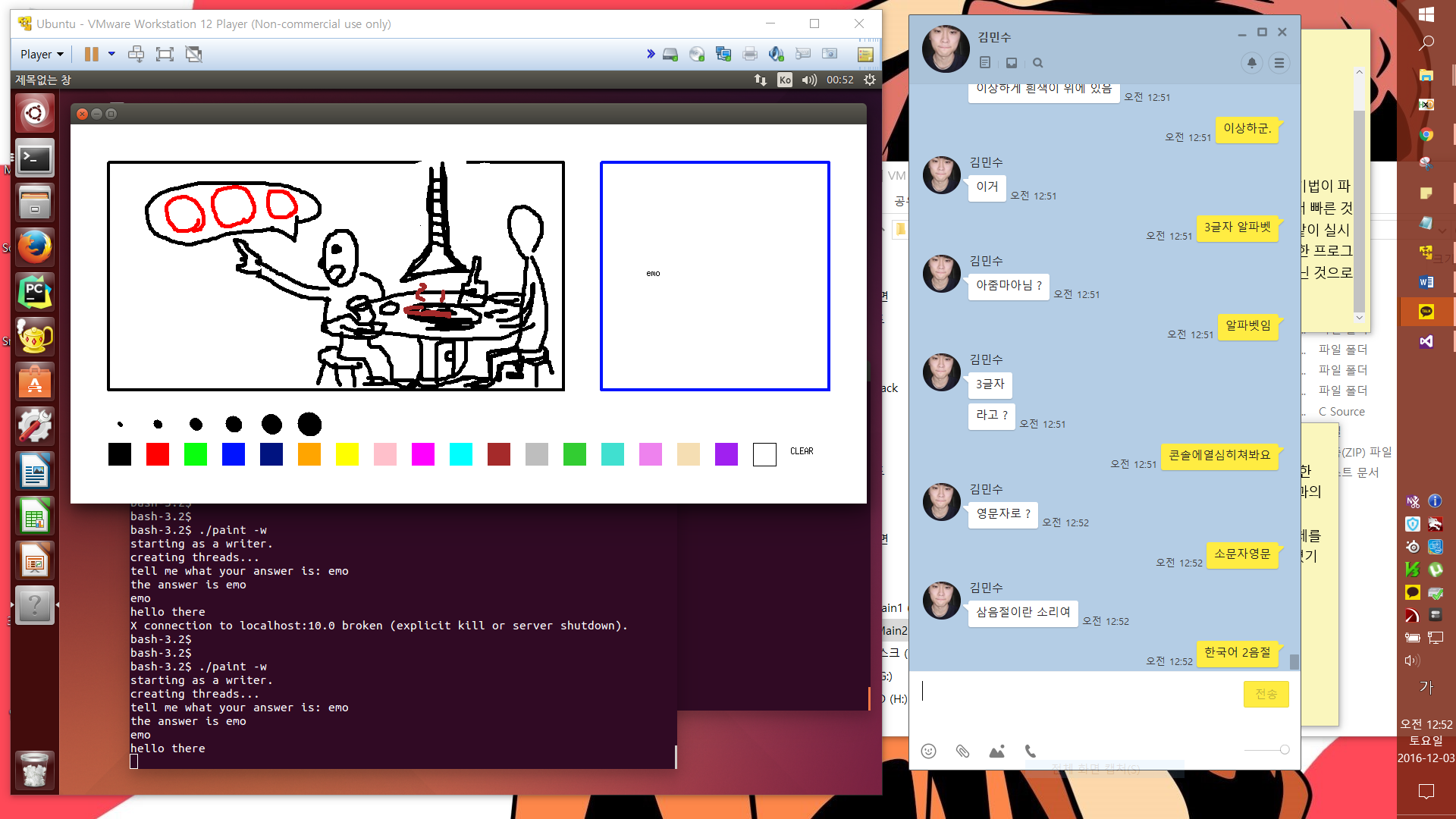
1. 최종 생성된 프로그램의 이름은 paint이다. 그리는 측은 -w, 문제를 맞추는 측은 -r 옵션을 주어 실행한다.
2. 파이프 통신(Named FIFO)을 이용한 버전에서는 ./paint -r를 먼저 실행하고 그 다음 ./paint -w를 실행해야 한다.
3. writer가 그림을 그리면 reader는 그 과정을 실시간으로 확인할 수 있다. (FIFO를 이용한 버전에서 문제 맞추기는 구현되지 않았다.)
4. 프로그램 실행 화면

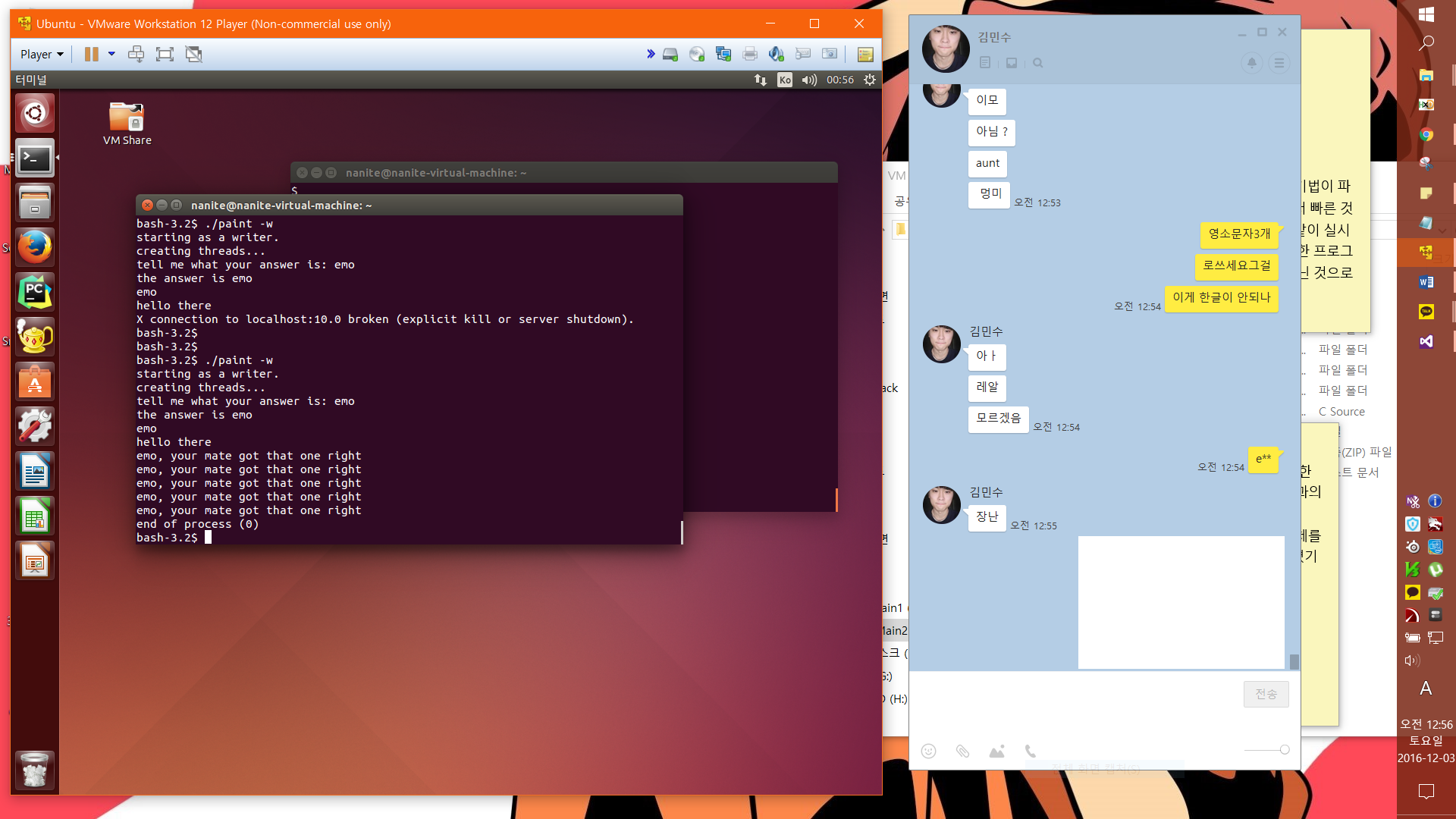












1. **성능비교 및 분석**

두 버전은 그림이 전송되어 표현되기까지의 응답속도에 차이가 있다. 그림 그리는 측이 직선을 그렸을 때 그 데이터를 받는 쪽에서 그림의 표현을 창에 반영하는 데까지 대략 0.1초 정도 지연시간이 있다. 파이프 방식의 경우 그보다 몇 배 이상 늦다. 또 파이프로 통신을 구현한 버전에서는 그림 그리는 측이 일시적으로 많은 양을 그릴 때 미처 따라 그리지 못하고 오류를 일으킨다.

결론적으로 메시지 패싱을 이용한 Advanced IPC 기법이 파이프를 통한 프로세스간 통신에 비해서 빠른 것을 알 수 있다. 하지만 본 과제에서와 같이 실시간으로 좌표를 주고 받는 수준의 간단한 프로그램에서는 전체적으로 크게 체감할 만한 차이는 아닌 것으로 보인다.

1. **Raw code**

(소스 코드는 별도 출력함)