|  |
| --- |
|  |
| 임베디드 프로젝트 보고서 |
|  |
| <픽셔너리> |

**컴퓨터공학과**

**20121154021 문동선, 2012152014 박경재**

**임베디드 시스템 프로젝트 보고서**

임베디드 프로젝트 보고서

1. 개요

픽셔너리를 모티브로 한 프로그램이다. 픽셔너리란 1985년에 출시된 보드 게임이다. 게임 방법은 2명부터 n명까지 진행한다. 술래가 한명 정해지면 인물, 사자성어 등 무작위 키워드가 제시된다. 술래가 제시된 단어를 그림으로 그린다. 그러면 술래를 제외한 모든 사람이 그림을 보고 단어를 추측하는 게임이다. 문제를 맞히는 측은 출제자가 그림을 그리는 과정을 실시간으로 지켜볼 수 있어야 하며 정답을 입력하면 프로그램은 종료한다. 한국에서는 캐치 마인드라는 온라인 게임으로 유명하다

2. 선정 이유

1. 복잡하지 않고 적절한 프로그램 규모다. 또 그림을 그려야 한다는 점에서 시각적인 표현을 요구하고, 그래픽적 구현요소로 인해서 새롭고 직관적이고 참신하다.
2. 대중적인 소재로 기존에 없던 것을 프로그램으로 구현하기 때문에 의미 있다.
3. 동시에 다수(두 명 이상)의 이용자가 다수의 프로세스로 정보를 주고 받는 게임의 특성상 네트워크간 통신 기법을 적용하기 좋은 주제라고 판단된다.

3. 요구사항

1. 문제를 맞추는 서버가 먼저 실행 된다.
2. 서버는 포트번호를 입력하고 실행 한다.
3. 서버가 켜지면 문제를 그리는 클라이언트를 실행 한다.
4. 클라이언트는 서버의 IP주소, 포트번호를 입력한다.
5. 서버, 클라이언트는 모두 실행 될 때 XWindow가 실행 된다.
6. 클라이언트와 서버가 연결 되면 클라이언트는 그림을 그릴 수 있다.
7. 그림을 그리면 그림의 좌표가 서버로 전송된다.
8. 서버는 좌표를 받고 클라이언트와 똑 같은 그림을 표시한다.

4. 팀원 업무 분담

|  |  |
| --- | --- |
| 문동선 | 모듈화 및 메이크 파일 작성, 버전 관리, 소켓 통신 구현 |
| 박경재 | 그리기 구현, 메이크 파일 작성, 소켓 통신 구현 |

5. 설계

프로그램은 기본적으로 문제를 맞히는 reader와 제시어를 입력하고 그림을 그리는 writer의 두 부분으로 나뉜다. 두 프로그램은 소켓 통신을 이용하여 데이터를 주고 받는다.

그림 맞추는

Server

(Fedora(Linux))

그림 그리는

Client

(Marsboad)

Socket

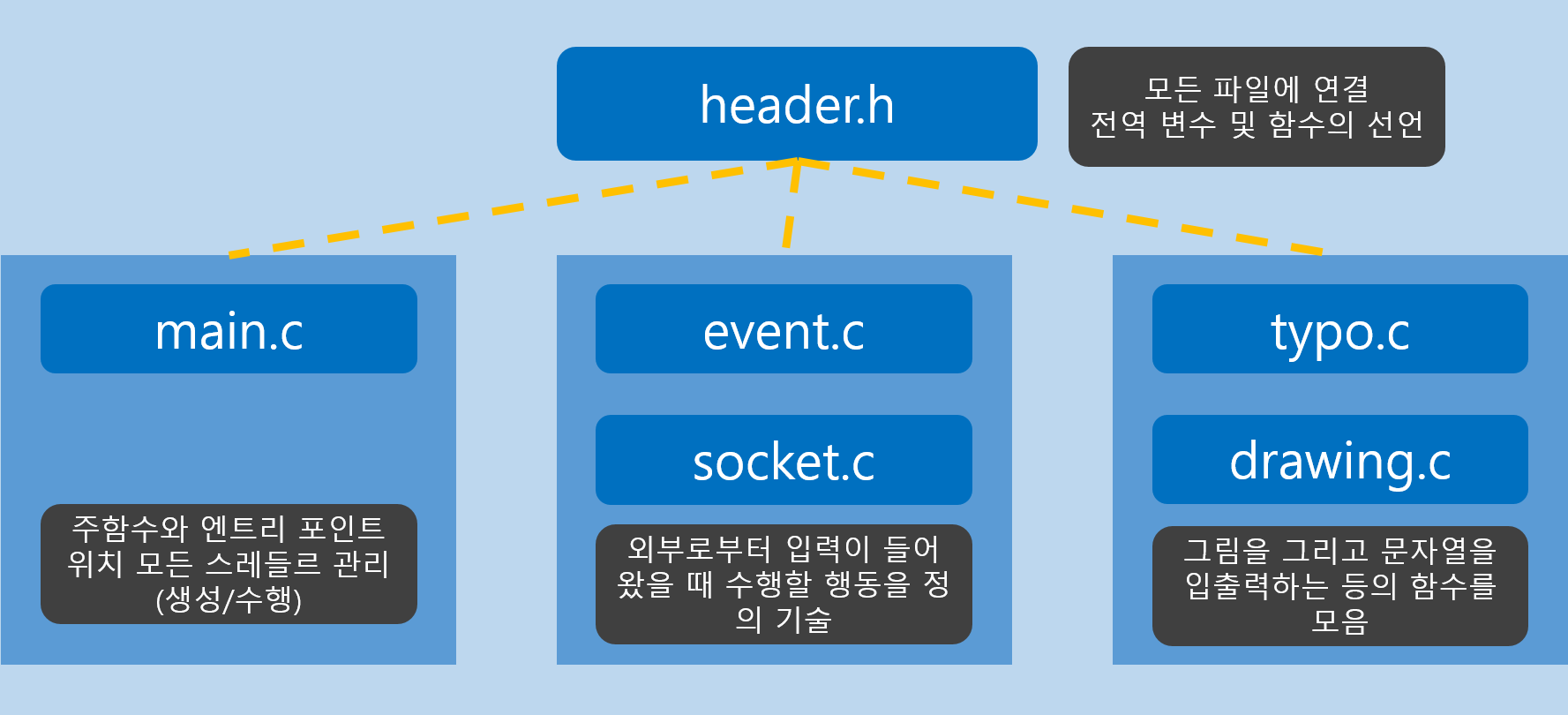
Socket

Network

각 프로그램은 3개의 스레드를 가지고 있다. 메인 스레드는 모든 서브 스레드를 생성 및 관리하며 x window 창을 띄운다. 메인 스레드 이외에 x window에 대한 입력을 받고 각 이벤트에 대한 처리를 수행하는 스레드와 다른 프로그램에서 소켓에서 넘어오는 데이터를 읽어 들이기 위한 스레드가 있다.

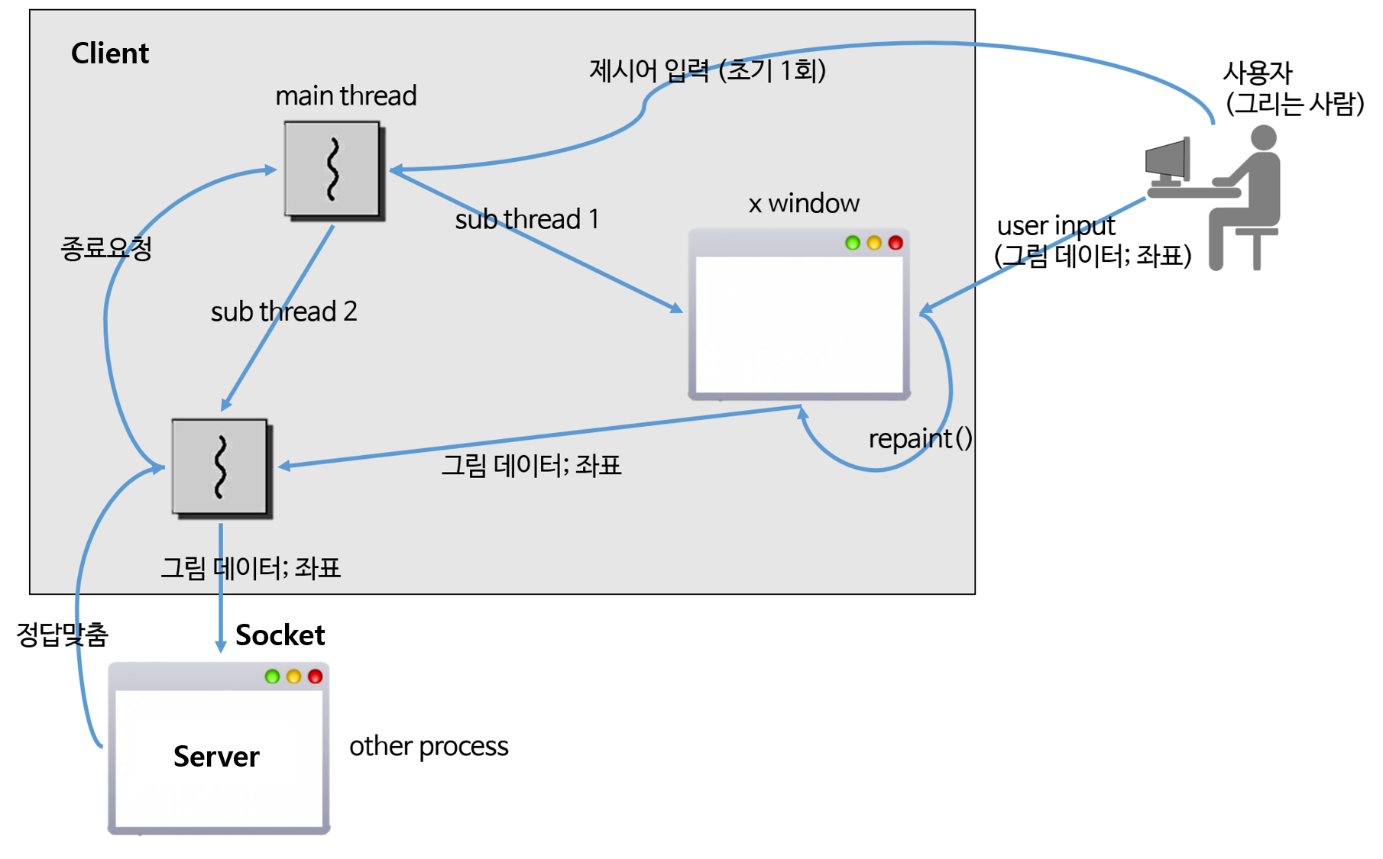
6. 모듈 구성

**1) 모듈 구성**

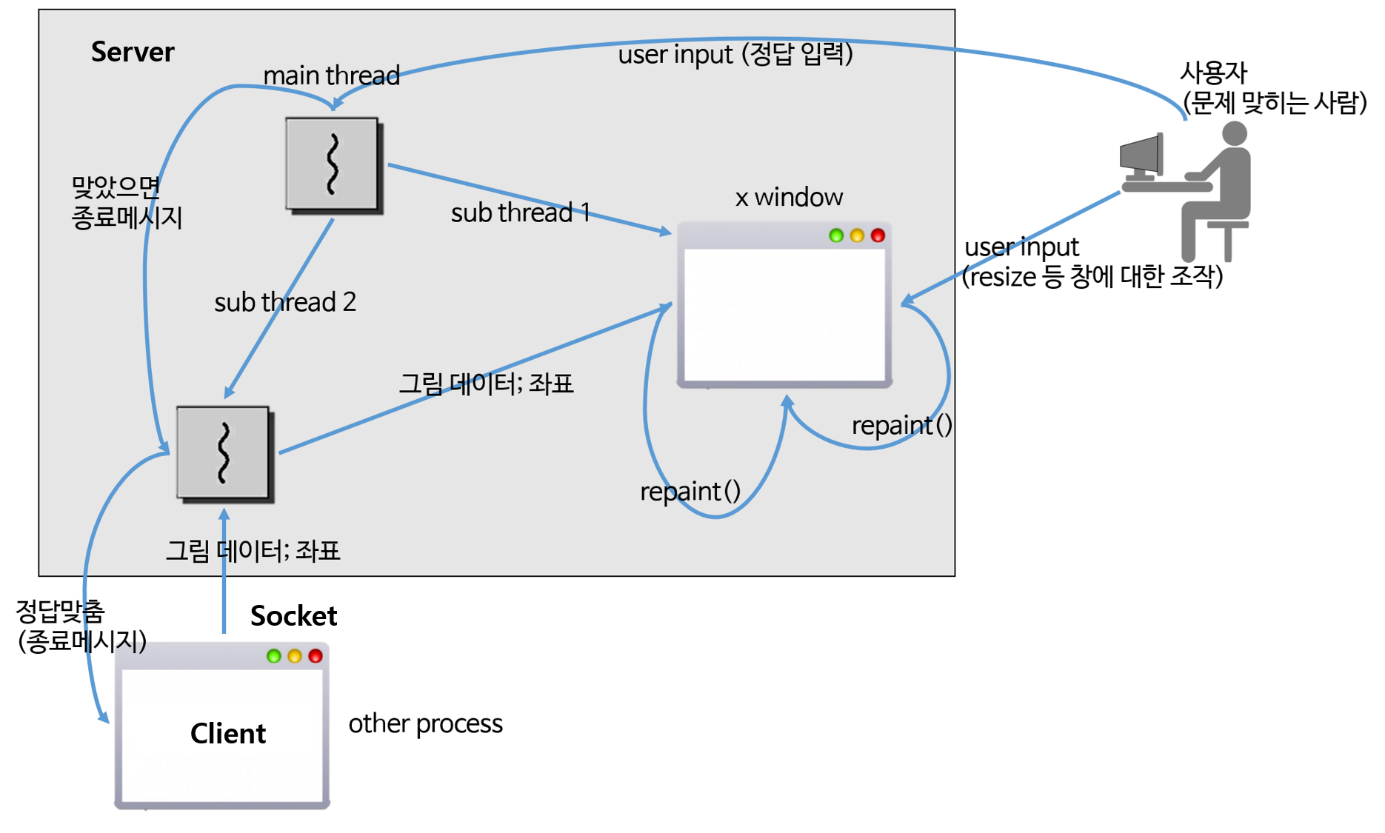


* + header.h 파일: 다른 모든 파일에 연결된다. 필요한 라이브러리를 모두 include하고 전역변수 및 함수의 원형들을 전부 선언한다.
  + main.c 파일: 메인 스레드를 실행하고 나머지 모든 스레드를 관리하며 프로그램 초기화와 사용자의 키입력을 처리한다.
  + event.c 파일: 서브 스레드 1의 행동이 정의되어 있다. x window 시스템으로 생성된 창에 대한 유저의 조작이 있을 때 이벤트를 감지하고 해당하는 이벤트 핸들러를 통해 이벤트에 대한 처리를 수행한다.
  + socket.c 파일: 프로그램이 종료할 때까지 반복하면서 소켓을 통해 수신된 데이터를 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.

**2) Client 프로그램 모델**

****

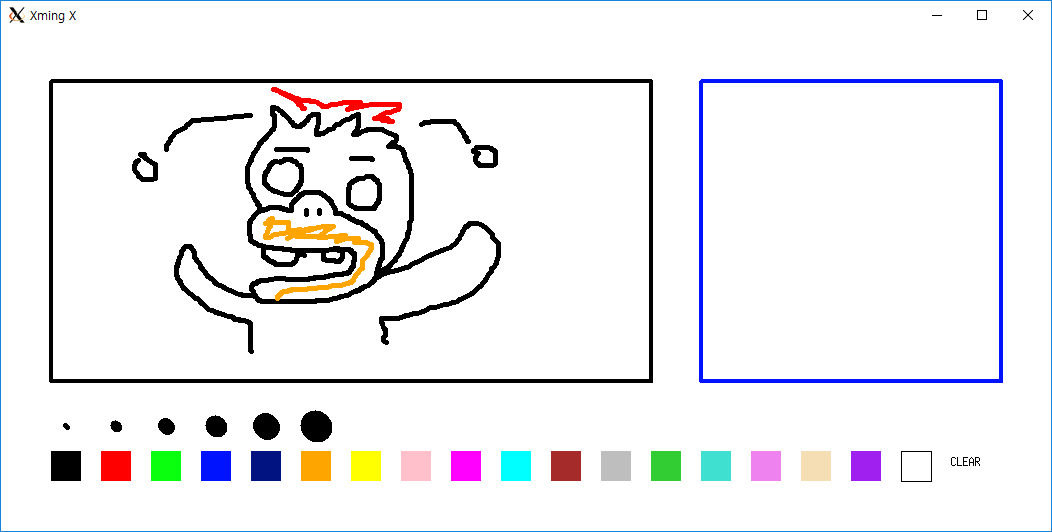
1. **Server 프로그램 모델**

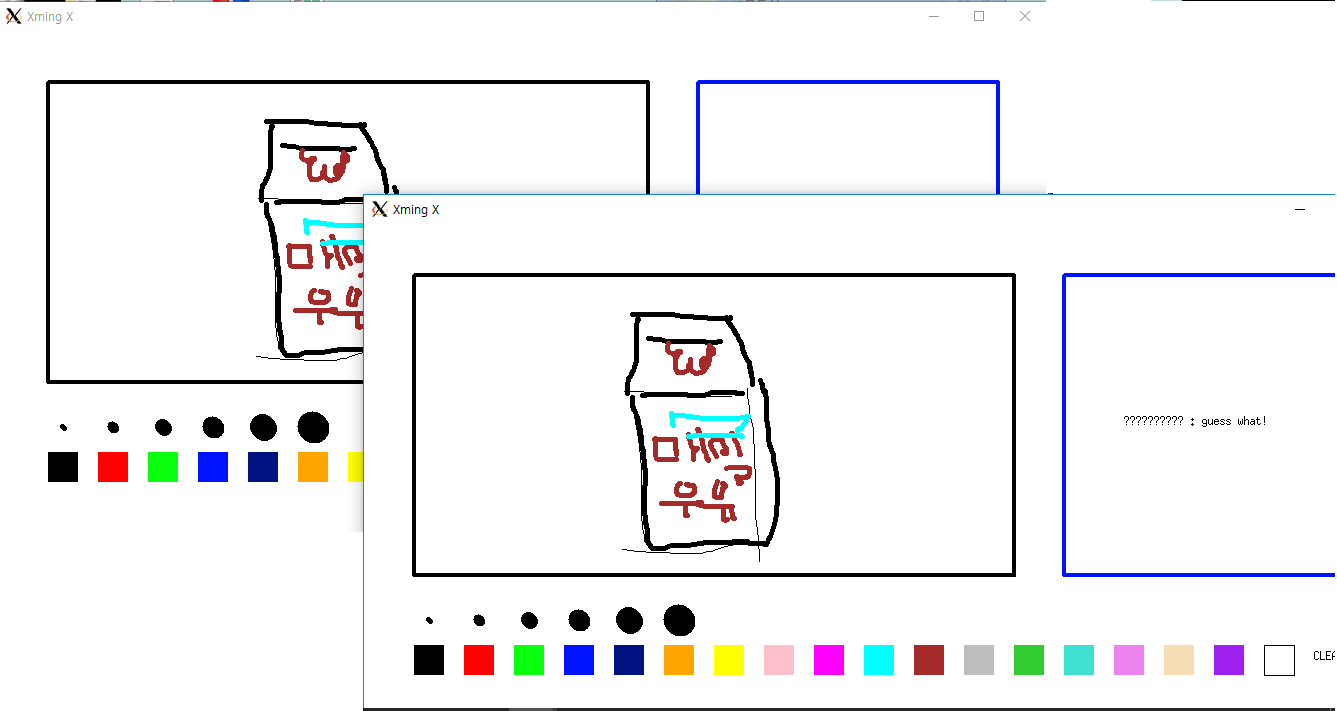
****

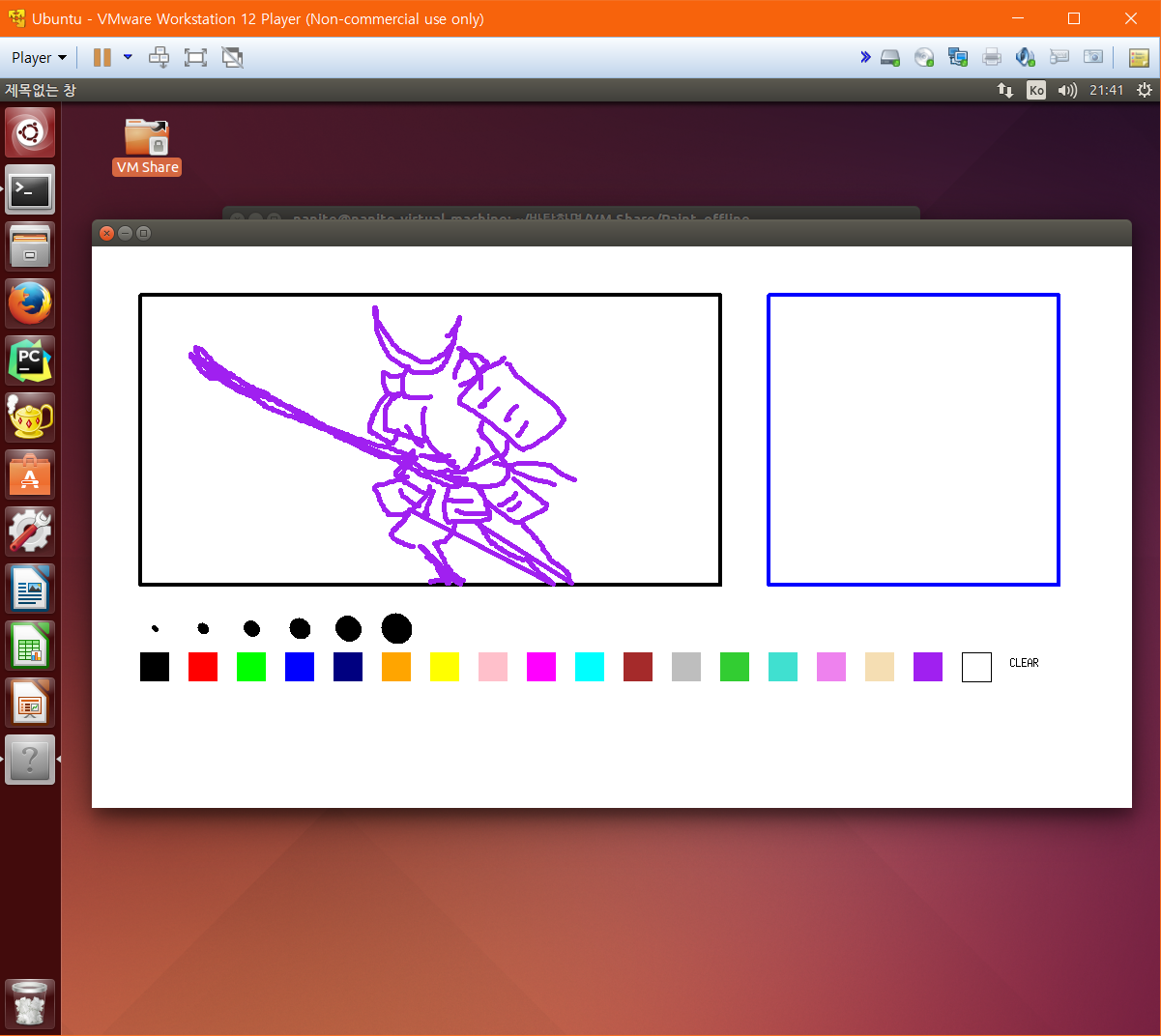
**7. 프로그램 실행 방법**

* + 1. Marsboard에서 소스를 복사한 뒤 터미널에서 make 명령어를 수행하여 프로그램을 빌드한다. 또 페도라 환경에서 소스를 복사한 뒤 터미널에서 make 명령어를 수행하여 프로그램을 빌드한다.
    2. 최종 생성된 프로그램의 이름은 paint이다. 그리는 측은 –w 적고 IP번호를 적는다. 포트번호는 9000번으로 정해져 있다. 문제를 맞추는 측은 -r 옵션을 주고 포트번호 9000을 적는다. 문제를 맞추는 측이 서버이기 때문에 서버부터 실행시키는게 중요하다.
    3. 그림을 그리는 측(client)에서 제시어를 그리면, 정답을 맞히는 측(server) 에 클라이언트의 그림 데이터들이 전송이 된다.
    4. client 가 그림을 그리면 server 는 그 과정을 실시간으로 확인한다.

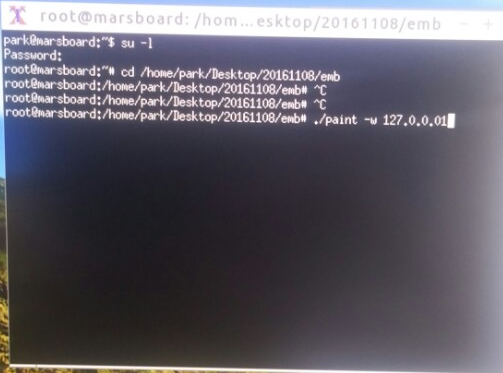
8. 프로그램 실행 화면

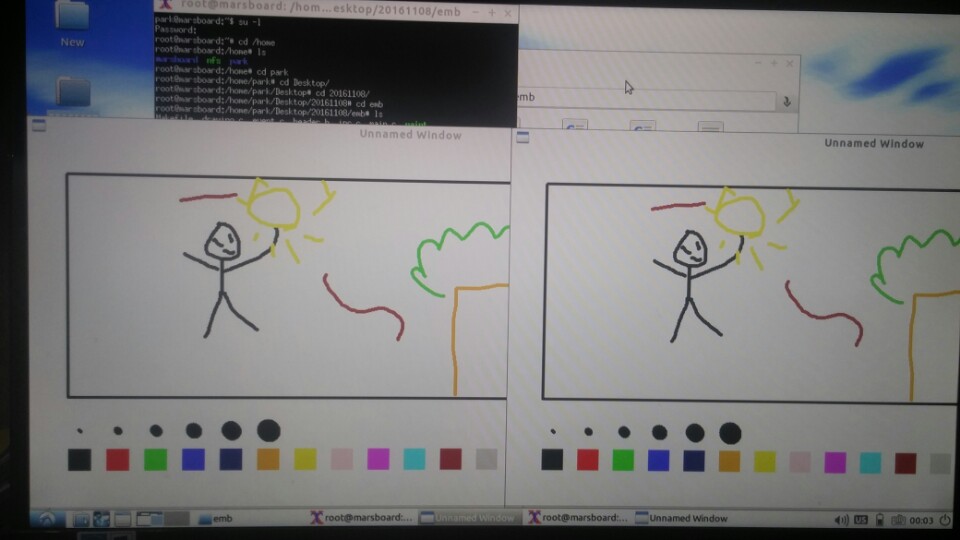






9. 보드 실제 동작 화면





10. 소스코드

**1) main.c**

// main.c : 주함수와 스레드 관리

#include "header.h"

#define N\_THREADS 2

static pthread\_t thid[N\_THREADS];

// externs

char strAnswerCorrect[100] = "nn45nn3c8n";

bool isDrawer = false;

bool bGameOver = false;

extern int main(int argc, char\* argv[]) {

// 옵션 지정하지 않았을 때 에러 출력하고 종료

if (argc < 3) {

printf(

"not enough argument. \n"

"usage: \n"

"[%s -w <IP>] for writer \n"

"[%s -r <whatever>] for reader \n",

argv[0], argv[0]

);

return 1;

}

// writer와 reader 인자 구분

if (strcmp(argv[1],"-w") == 0) {

isDrawer = true;

printf("starting as a writer. \n");

} else if (strcmp(argv[1], "-r") == 0) {

isDrawer = false;

printf("starting as a reader. \n");

} else {

printf("invalid argument \n");

return 1;

}

// 본편 실행

printf("creating threads... \n");

if (isDrawer) {

ClientInitSock(argv[2]); //클라이언트 socket 초기화

InitDisplay(); // x window 초기화

pthread\_create(&thid[0], NULL, Thread1, NULL); // 스레드0: x window 입력 받음

printf("hello there \n");

} else /\* if (!isDrawer) \*/ {

InitDisplay();

pthread\_create(&thid[0], NULL, Thread1, NULL); // x window

pthread\_create(&thid[1], NULL, Thread2Reader, NULL); //Read Socket 초기화 및 메시지 수신

printf("hello there \n");

}

for (;;);

/\*

for (i = 0; i < N\_THREADS; ++i) {

if (pthread\_join(thid[i], NULL) != 0) {

printf("end of process (-1) \n");

return -1;

}

}

\*/

// 종료

printf("end of process (0) \n");

return 0;

}

**2) socket.c**

#include "header.h"

#define MSGTYPE\_PATH 1

#define MSGTYPE\_ANSWER 2

#define MSGTYPE\_GAMEOVER 3

//포트번호

#define PORT 9000

//스레드

static pthread\_t t\_id;

//서버소켓

int sock;

int serv\_sock;

//클라이언트 소켓

#define MAX\_CLNT 256

static int clnt\_cnt = 0;

static int clnt\_socks[MAX\_CLNT];

static pthread\_mutex\_t mutx;

// 주고 받을 데이터: 그린 점

struct path {

int index;

int x;

int y;

int color;

int width;

};

// 오류

static void fatal(char \*err) {

perror(err);

exit(0);

}

//클라이언트 소켓 초기화

extern void ClientInitSock(char\* ip) {

struct sockaddr\_in serv\_addr;

sock = socket(PF\_INET, SOCK\_STREAM, 0);

memset(&serv\_addr, 0, sizeof(serv\_addr));

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = inet\_addr(ip);

serv\_addr.sin\_port = htons(PORT);

if (connect(sock, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) == -1)

fatal("connect() error");

}

// 그린 지점 데이터를 서버로 보냄 클라이언트가 할일

extern void SndPath(int \_index, int \_x, int \_y, int \_color, int \_width) {

struct path buf;

buf.index = \_index;

buf.x = \_x;

buf.y = \_y;

buf.color = \_color;

buf.width = \_width;

write(sock, &buf, sizeof(buf));

}

// 받은 그린 데이터(구조체 메시지; 점 하나)를 풀어서 저장 서버가 할일

static void RcvPath(struct path rcvpath) {

if (indexPath >= MAX\_INDEX\_PATH) { return; } // 여분의 공간이 없으면 추가할 수 없음

// 현재위치 저장

path[rcvpath.index].x = rcvpath.x;

path[rcvpath.index].y = rcvpath.y;

// 현재위치와 함께 다른 (추가) 정보 저장

pathColor[rcvpath.index] = rcvpath.color;

pathWidth[rcvpath.index] = rcvpath.width;

// CLEAR 입력 받은 경우

if (rcvpath.index == -1) {

Clear(); // 지운다

indexPath = 0;

}

// 다음으로

if (indexPath <= rcvpath.index) {

indexPath = rcvpath.index + 1;

}

}

//서버가 여러 클라이언트 데이터 수신

static void \*handle\_clnt(void \*arg) {

int i, str\_len = 0;

struct path buf;

int clnt\_sock = 0;

//데이터 수신

for (;(str\_len = read(\*((int\*)arg), &buf, sizeof(buf))) != 0;) {

RcvPath(buf);

RepaintPath();

}

pthread\_mutex\_lock(&mutx); //동기화 시켜줌

for (i = 0; i<clnt\_cnt; i++) {

if (clnt\_sock == clnt\_socks[i])

{

while (i++<clnt\_cnt - 1)

clnt\_socks[i] = clnt\_socks[i + 1];

break;

}

}

clnt\_cnt--;

pthread\_mutex\_unlock(&mutx);

close(clnt\_sock);

return NULL;

}

//여러 클라이언트 접속시 서버는 클라이언트를 여러쓰레드로 받는다.

static void ServerLoopAccept() {

unsigned int clnt\_adr\_sz;

struct sockaddr\_in clnt\_adr;

int clnt\_sock = 0;

for (;;) {

clnt\_adr\_sz = sizeof(clnt\_adr);

clnt\_sock = accept(serv\_sock, (struct sockaddr\*)&clnt\_adr, &clnt\_adr\_sz);

pthread\_mutex\_lock(&mutx);

clnt\_socks[clnt\_cnt++] = clnt\_sock;

pthread\_mutex\_unlock(&mutx);

pthread\_create(&t\_id, NULL, handle\_clnt, (void\*)&clnt\_sock);

pthread\_detach(t\_id);

printf("Connected client IP: %s \n", inet\_ntoa(clnt\_adr.sin\_addr));

}

}

//서버 초기화

static void ServerInitSock() {

struct sockaddr\_in serv\_adr;

pthread\_mutex\_init(&mutx, NULL); //뮤텍스 초기화

serv\_sock = socket(PF\_INET, SOCK\_STREAM, 0);

memset(&serv\_adr, 0, sizeof(serv\_adr));

serv\_adr.sin\_family = AF\_INET;

serv\_adr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

serv\_adr.sin\_port = htons(PORT); //포트 9000

//바인드, 리슨

if (bind(serv\_sock, (struct sockaddr\*) &serv\_adr, sizeof(serv\_adr)) == -1)

fatal("bind() error");

if (listen(serv\_sock, 5) == -1)

fatal("listen() error");

}

extern void \*Thread2Reader() {

ServerInitSock();

ServerLoopAccept();

pthread\_exit(NULL);

}

**3) header.h**

#include <stdio.h>

#include <pthread.h>

#include <X11/Xlib.h>

#include <X11/Xutil.h>

#include <X11/keysymdef.h>

#include <termios.h>

#include <stdio.h>

#include <stdlib.h>

// header.h : 유일한 헤더

#include <string.h>

#include <fcntl.h>

#include <unistd.h>

#include <errno.h>

#include <time.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

typedef enum bool{ false, true } bool;

// 창에 요소가 배치될 때 창의 왼쪽이나 위쪽 여백

#define MARGIN 50

// 캔버스 영역: 그림 그리는 곳

#define CANVAS\_X MARGIN

#define CANVAS\_Y MARGIN

#define CANVAS\_WIDTH 600

#define CANVAS\_HEIGHT 300

// 플레이어 영역: 오른쪽에 그려지는 사각형 두 개

#define PLAYER\_X MARGIN + CANVAS\_WIDTH + MARGIN

#define PLAYER\_Y MARGIN

#define PLAYER\_WIDTH 300

#define PLAYER\_HEIGHT CANVAS\_HEIGHT

// 물감 (색상 선택) 영역

#define PALET\_AREA\_X MARGIN

#define PALET\_AREA\_Y 370

#define PALET\_ITEM\_WIDTH 30

#define PALET\_ITEM\_HEIGHT 30

// 붓의 굵기 선택 영역

#define BRUSH\_ITEM\_HEIGHT 30

#define BRUSH\_ITEM\_WIDTH 30

#define BRUSH\_AREA\_X MARGIN

#define BRUSH\_AREA\_Y MARGIN + CANVAS\_HEIGHT + BRUSH\_ITEM\_HEIGHT

// main window

extern Display \*dpy;

extern Window w;

extern XEvent xe;

// other ui stuff (context)

extern GC gc;

extern int lineWidth;

extern int color;

// socket

extern int sock;

extern int serv\_sock, clnt\_sock;

extern char strAnswerCorrect[100];

extern bool isDrawer;

extern bool bGameOver;

// drawn line

#define MAX\_INDEX\_PATH 100000

extern int indexPath;

extern XPoint path[MAX\_INDEX\_PATH];

extern int pathColor[MAX\_INDEX\_PATH];

extern int pathWidth[MAX\_INDEX\_PATH];

// drawing

extern void ContinuePath(int x, int y);

extern void DrawPallete();

extern void RepaintPath();

extern void Clear();

extern bool EventCursorIsWithinCanvas(XEvent);

extern bool EventCursorIsWithinColorPick(XEvent);

extern bool EventCursorIsWithinWidthPick(XEvent);

extern bool EventCursorIsBeingClicked(XEvent);

extern bool GetClearPick(XEvent);

extern int GetColorPick(XEvent);

extern int GetWidthPick(XEvent);

extern void SetForegroundToColorIndex(int i);

extern void SetLineWidth(int width);

//socket

extern void SndPath(int \_index, int \_x, int \_y, int \_color, int \_width);

extern void ClientInitSock(char\* ip);

// init

extern void InitDisplay();

// threads

extern void \*Thread1();

extern void \*Thread2Reader();

1. **event.c**

// event.c : 창의 초기화와 이벤트 처리 거시적 관점

#include "header.h"

// main window

Display \*dpy;

Window w;

XEvent xe;

// other ui stuff (context)

GC gc;

int lineWidth;

int color;

// drawn line

int indexPath;

XPoint path[MAX\_INDEX\_PATH];

int pathColor[MAX\_INDEX\_PATH];

int pathWidth[MAX\_INDEX\_PATH];

extern void InitDisplay() {

// 1. 창 띄우고 이벤트(입력) 연결

dpy = XOpenDisplay(NULL);

w = XCreateSimpleWindow(dpy, RootWindow(dpy, 0), 50, 50, 1050, 500, // Display, Window, x, y, width, height

5, BlackPixel(dpy, 0), WhitePixel(dpy, 0)); // border width, border color, bg color

XSelectInput(dpy, w, ExposureMask |

ButtonPressMask | ButtonReleaseMask | // 마우스 버튼 눌림, 뗌

// EnterWindowMask | LeaveWindowMask | // 창에 들어옴, 나감

PointerMotionMask // 커서 이동 (MotionNotify 발생)

);

XMapWindow(dpy, w);

XFlush(dpy);

// 2. gc, font, line, width, color 설정

gc = XCreateGC(dpy, w, 0L, (XGCValues \*)NULL);

color = BlackPixel(dpy, 0);

lineWidth = 5;

XSetFont(dpy, gc, XLoadFont(dpy, "fixed"));

XSetLineAttributes(dpy, gc, lineWidth, LineSolid, CapRound, JoinRound);

XFlush(dpy);

} // func

// (XEvent xe)

static void onExpose() {

DrawPallete();

RepaintPath();

} // func

static void onMouseMoved(XEvent xe) {

if (!isDrawer) { return; } // 이 프로세스가 그림 그리는 측인 경우에만 그린다

// 캔버스 안에서 커서가 눌린 채 움직였고 그림 데이터 저장공간이 여유 있다면

if (EventCursorIsWithinCanvas(xe) && EventCursorIsBeingClicked(xe) && indexPath < MAX\_INDEX\_PATH) {

// 현재위치 저장

path[indexPath].x = xe.xmotion.x;

path[indexPath].y = xe.xmotion.y;

// 직전위치부터 현재위치까지 잇는 선을 그린다 (드로우)

if (indexPath == 0) { return; }

XDrawLine(dpy, w, gc,

path[indexPath - 1].x, path[indexPath - 1].y,

path[indexPath].x, path[indexPath].y);

// 다음으로 (현재위치는 직전위치가 된다)

++indexPath;

// 기본 UI 그려주기

// DrawPallete();

// 통신 추가

if (isDrawer) {

SndPath(indexPath - 1, xe.xmotion.x, xe.xmotion.y, color, lineWidth);

}

} // if

} // func

static void onButtonPress(XEvent xe) {

if (!isDrawer) { return; } // 이 프로세스가 그림 그리는 측인 경우에만 그린다

if (EventCursorIsWithinCanvas(xe) && indexPath < MAX\_INDEX\_PATH) {

ContinuePath(xe.xmotion.x, xe.xmotion.y);

}

} // func

static void onButtonRelease(XEvent xe) {

int result;

if (!isDrawer) { return; } // 이 프로세스가 그림 그리는 측인 경우에만 그린다

XEvent local\_xe = xe;

if (EventCursorIsWithinColorPick(local\_xe)) { // 색상 선택

// printf("EventCursorIsWithinColorPick\n");

if ((result = GetColorPick(local\_xe)) != -1) { // 몇 번째 요소인지 확인하고

SetForegroundToColorIndex(result); // 그 요소의 색상을 적용한다

} else if (GetClearPick(local\_xe)) { // "CLEAR" 선택시

Clear();

}

} else if (EventCursorIsWithinWidthPick(local\_xe)) { // 붓 굵기 선택

// printf("EventCursorIsWithinWidthPick\n");

if ((result = GetWidthPick(local\_xe)) != -1) { // 몇 번째 요소인지 확인하고

SetLineWidth(result \* 5 + 5); // 그 요소의 굵기를 적용한다

}

} else if (EventCursorIsWithinCanvas(local\_xe)) {

ContinuePath(-1, -1);

} // elif

} // func

// input processing routine for main thread

static void UserInputMessageLoop() {

for (;;) {

XNextEvent(dpy, &xe);

switch (xe.type) {

case Expose: // 창 크기 변경 (resize)

onExpose(xe);

break;

case MotionNotify: // 커서 이동

onMouseMoved(xe);

break;

case ButtonPress: // 마우스 클릭: 누름

onButtonPress(xe);

break;

case ButtonRelease: // 마우스 클릭: 뗌

onButtonRelease(xe);

break;

case EnterNotify: // 커서가 창 안으로 들어옴

break;

case LeaveNotify: // 커서가 창 밖으로 나감

break;

default: // Neither of above

printf("xe.type unknown: check either XSelectInput() or switch case (warning) \n");

break;

}

} // for(;;)

} // func

extern void \*Thread1() {

UserInputMessageLoop();

pthread\_exit(NULL);

} // func

1. **drawing.c**

// drawing.c 모든 자잘한 (미시적인) 그리는 처리와 그에 연관된 것

#include "header.h"

//////////////////// color ////////////////////

// color related

#define MAX\_COLOR 18 // number of basic colors available to pick

#define WHITE 17

static char \*color\_name[MAX\_COLOR] = {

"Black", "Red", "Green", "Blue", "Navy",

"Orange", "Yellow", "Pink", "Magenta", "Cyan",

"Brown", "Grey", "LimeGreen", "Turquoise", "Violet",

"Wheat", "Purple" , "White"

};

// "Black", "Red", "Green", "Blue", "Navy",

// "Orange", "Yellow", "Pink", "Magenta", "Cyan",

// "Brown", "Grey", "LimeGreen", "Turquoise", "Violet",

// "Wheat", "Purple", "White"

// ex) XSetForeground(dpy, gc, UsrColorPixel("blue"));

static unsigned long UsrColorPixel(char \*name) { // get color

XColor c0, c1;

Colormap cmap;

cmap = DefaultColormap(dpy, 0);

XAllocNamedColor(dpy, cmap, name, &c0, &c1);

return (c0.pixel);

}

// GC의 선의 색상을 지정

extern void SetForegroundToColorIndex(int i) {

color = UsrColorPixel(color\_name[i]);

XSetForeground(dpy, gc, color);

}

/\*

// GC의 선의 색상을 지정

static void SetForegroundToColorName(char\* name) {

color = UsrColorPixel(name);

XSetForeground(dpy, gc, color);

}

\*/

// GC의 선의 색상을 지정

static void SetForegroundToColor(unsigned long \_color) {

color = \_color;

XSetForeground(dpy, gc, color);

}

//////////////////// ////////////////////

// GC의 선의 굵기를 지정

extern void SetLineWidth(int width) {

XGCValues gv;

lineWidth = width;

gv.line\_width = width;

XChangeGC(dpy, gc, GCLineWidth, &gv);

}

// path에 점 하나를 추가함

extern void ContinuePath(int x, int y) {

if (indexPath >= MAX\_INDEX\_PATH) { return; } // 여분의 공간이 없으면 추가할 수 없음

// 현재위치 저장

path[indexPath].x = x;

path[indexPath].y = y;

// 현재위치와 함께 다른 (추가) 정보 저장

pathColor[indexPath] = color;

pathWidth[indexPath] = lineWidth;

// 다음으로 (현재위치는 직전위치가 된다)

++indexPath;

// 통신 추가

if (isDrawer) {

SndPath(indexPath - 1, x, y, color, lineWidth);

}

}

#define MAX\_NPICK\_WIDTH 6

// 창에 기본 UI 그리는 함수

extern void DrawPallete() {

int i;

XGCValues gv;

// 캔버스 영역 드로우

XSetForeground(dpy, gc, BlackPixel(dpy, 0)); // 임시로 그릴 거니까 static void SetForegroundToColor() 사용하지 않음

gv.line\_width = 4;

XChangeGC(dpy, gc, GCLineWidth, &gv);

XDrawRectangle(dpy, w, gc, CANVAS\_X, CANVAS\_Y, CANVAS\_WIDTH, CANVAS\_HEIGHT);

// 정답 영역 드로우

XSetForeground(dpy, gc, UsrColorPixel("blue"));

gv.line\_width = 4;

XChangeGC(dpy, gc, GCLineWidth, &gv);

XDrawRectangle(dpy, w, gc, PLAYER\_X, PLAYER\_Y, PLAYER\_WIDTH, PLAYER\_HEIGHT);

// 정답 영역 문자

XSetForeground(dpy, gc, BlackPixel(dpy, 0)); // 임시로 그릴 거니까 static void SetForegroundToColor() 사용하지 않음

if (isDrawer) {

XDrawString(dpy, w, gc,

PLAYER\_X + (PLAYER\_WIDTH / 5),

PLAYER\_Y + (PLAYER\_HEIGHT / 2),

strAnswerCorrect, strlen(strAnswerCorrect));

} else {

XDrawString(dpy, w, gc,

PLAYER\_X + (PLAYER\_WIDTH / 5),

PLAYER\_Y + (PLAYER\_HEIGHT / 2),

"????????????? : guess what!", strlen("????????????? : guess what!"));

}

// 물감 영역 드로우: 흰색 차례 직전까지 색깔 순서대로

for (i = 0; i< WHITE; i++) {

XSetForeground(dpy, gc, UsrColorPixel(color\_name[i]));

XFillRectangle(dpy, w, gc,

MARGIN + (PALET\_AREA\_X \* i), MARGIN + PALET\_AREA\_Y,

PALET\_ITEM\_WIDTH, PALET\_ITEM\_HEIGHT);

}

// 물감 영역 드로우: 흰색

i = WHITE;

XSetForeground(dpy, gc, BlackPixel(dpy, 0));

gv.line\_width = 1;

XChangeGC(dpy, gc, GCLineWidth, &gv);

XDrawRectangle(dpy, w, gc,

MARGIN + (PALET\_AREA\_X \* i), MARGIN + PALET\_AREA\_Y,

PALET\_ITEM\_WIDTH, PALET\_ITEM\_HEIGHT);

// 물감 영역 드로우: CLEAR

XSetFont(dpy, gc, XLoadFont(dpy, "fixed"));

XDrawString(dpy, w, gc,

MARGIN + (PALET\_AREA\_X \* (i + 1)), MARGIN + PALET\_AREA\_Y + 15,

"CLEAR", 5);

// 붓의 굵기 드로우

for (i = 0; i< MAX\_NPICK\_WIDTH; i++) {

XSetForeground(dpy, gc, BlackPixel(dpy, 0));

gv.line\_width = i \* 5 + 5;

XChangeGC(dpy, gc, GCLineWidth, &gv);

XDrawLine(dpy, w, gc,

MARGIN + (BRUSH\_AREA\_X \* i) + (BRUSH\_ITEM\_WIDTH / 2) - 1,

BRUSH\_AREA\_Y + (BRUSH\_ITEM\_HEIGHT / 2) - 1,

MARGIN + (BRUSH\_AREA\_X \* i) + (BRUSH\_ITEM\_WIDTH / 2) + 1,

BRUSH\_AREA\_Y + (BRUSH\_ITEM\_HEIGHT / 2) + 1);

}

// 이전 콘텍스트로 되돌아간다

SetForegroundToColor(color);

gv.line\_width = lineWidth;

XChangeGC(dpy, gc, GCLineWidth, &gv);

}

// 다시 따라 그린다

extern void RepaintPath() {

int i; // 반복문 제어 변수

int contextColor = color; // 뒷정리 준비: 이 함수가 끝나면 돌아갈 곳 1 (스택 프레임처럼 시작할 때 한다)

int contextWidth = lineWidth; // 뒷정리 준비: 이 함수가 끝나면 돌아갈 곳 2

SetForegroundToColor(pathColor[0]);

SetLineWidth(pathWidth[0]);

for (i = 1; i<indexPath; i++) {

if (path[i].x != -1) { // 지금 지점이 끊어진 점(-1)이 아닌 유효한 점이니까, 그리기를 시작하는 중이던지 그리는 한 중간이던지 둘 중 하나다.

if (path[i - 1].x != -1) { // 이전 지점도 지금 지점도 끊어진 점(-1)이 아니니까, 붓칠을 하는 중간이다.

XDrawLine(dpy, w, gc, path[i - 1].x, path[i - 1].y, path[i].x, path[i].y); // 두 점을 잇는 선을 따라 붓칠한다.

} else if (path[i - 1].x == -1) { // 이전 지점은 끊어졌고 지금 지점은 있으니까, 두점을 따라 잇는 선을 그리기 시작하는 때다.

SetForegroundToColor(pathColor[i]); // 붓을 든다.

SetLineWidth(pathWidth[i]);

}

} else { /\* 지금 지점이 끊어진 점(-1)이 아닌 유효한 점이니까, 그리기를 중단한 상태다. 붓을 내린다. \*/ }

}

SetForegroundToColor(contextColor); // 뒷정리

SetLineWidth(contextWidth);

} // func

// 캔버스에 그린 것을 지운다

extern void Clear() {

XClearArea(dpy, w, CANVAS\_X, CANVAS\_Y, CANVAS\_WIDTH, CANVAS\_HEIGHT, 0);

indexPath = 0;

// 통신 추가

if (isDrawer) {

SndPath(-1, -1, -1, color, lineWidth);

}

}

// 판별

extern bool EventCursorIsWithinCanvas(XEvent xe) {

return (xe.xmotion.x > CANVAS\_X && xe.xmotion.x < CANVAS\_WIDTH + CANVAS\_X)

&& (xe.xmotion.y > CANVAS\_Y && xe.xmotion.y < CANVAS\_HEIGHT + CANVAS\_Y);

}

extern bool EventCursorIsWithinColorPick(XEvent xe) {

return xe.xmotion.y > MARGIN + PALET\_AREA\_Y && xe.xmotion.y < MARGIN + PALET\_AREA\_Y + PALET\_ITEM\_HEIGHT;

}

extern bool EventCursorIsWithinWidthPick(XEvent xe) {

return xe.xmotion.y > BRUSH\_AREA\_Y && xe.xmotion.y < BRUSH\_AREA\_Y + BRUSH\_ITEM\_HEIGHT;

}

extern bool EventCursorIsBeingClicked(XEvent xe) {

return xe.xmotion.state & Button1Mask;

}

// 버튼이 눌렸는지 여부 반환

extern bool GetClearPick(XEvent xe) {

int i = MAX\_COLOR;

return xe.xmotion.x > MARGIN + (BRUSH\_AREA\_X \* i) && xe.xmotion.x < MARGIN + (BRUSH\_AREA\_X \* i) + BRUSH\_ITEM\_WIDTH;

}

// 몇 번째 버튼이 눌렸는지 반환

extern int GetColorPick(XEvent xe) {

int i;

for (i = 0; i < MAX\_COLOR + 1; ++i) {

if (xe.xmotion.x > MARGIN + (BRUSH\_AREA\_X \* i)

&& xe.xmotion.x < MARGIN + (BRUSH\_AREA\_X \* i) + BRUSH\_ITEM\_WIDTH) {

break;

} // if

} // for

if (i >= MAX\_COLOR) { return -1; }

// printf("color %d \n", i);

return i;

}

// 몇 번째 버튼이 눌렸는지 반환

extern int GetWidthPick(XEvent xe) {

int i;

for (i = 0; i < MAX\_COLOR + 1; ++i) {

if (xe.xmotion.x > MARGIN + (PALET\_AREA\_X \* i)

&& xe.xmotion.x < MARGIN + (PALET\_AREA\_X \* i) + PALET\_ITEM\_WIDTH) {

break;

} // if

} // for

if (i >= MAX\_NPICK\_WIDTH) { return -1; }

// printf("width arr n %d \n", i);

return i;

}