1. 프로그램 개요

int main()

{

**while**(1)

{

ip = readtouch(); *// read touch*

cmd = readcommand(ip); *// convert to command*

**if**(cmd>=0 && cmd<=5) *// set drawmode*

*line, rect, oval, freedraw, select, erase 모드로 설정*

**else** **if**(cmd == 6) *// clear*

*화면을 전부 지움*

**else** **if**(cmd == 10 || cmd == 11) *// set pen or fill mode*

*pen, fill 중 맞는 모드로 설정*

**else** **if**(cmd>=20 && cmd<=27) *// set color*

*색깔 설정*

**else** **if**(cmd==100) *// drawing*

*현재 설정된 모드에 맞춰 그림을 그림*

}

}

*/\* 터치입력을 받는 함수 \*/*

input readtouch();

*/\* 터치입력을 명령으로 변환 \*/*

int readcommand(input);

*/\* 현재 모드를 바탕으로 그림을 그리는 함수 \*/*

void drawing(input);

*/\* 화면을 지우는 함수 \*/*

void clearpaper();

*/\* 배경을 만드는 함수 \*/*

void createbg(paint\* p);

*/\* 배경을 화면에 뿌려주는 함수 \*/*

void printbg(int,int,int,int);

*/\* 그림을 화면에 뿌려주는 함수 \*/*

void printmap(paint\*);

*/\* 그림을 선택하는 함수 \*/*

paint\* selectpaint(int,int);

*/\* 지우개로 지우는 함수 \*/*

void erasemap();

*/\* Bezier 곡선 알고리즘에 따라 점을 반환하는 함수 \*/*

point getbezier(point,point,point,double);

1. 전역변수 및 구조체

*/\* 그림의 크기 \*/*

#define PAPER\_SIZE 192

*/\* 각 그림을 저장할 구조체 \*/*

**typedef** **struct** tag\_paint

{

point min\_pos;

point max\_pos;

point flag;

unsigned char color;

unsigned int map[PAPER\_SIZE][6];

}paint;

*/\* 그림을 저장할 버퍼와 버퍼의 크기 \*/*

paint\* buffer[1000];

int buffer\_count;

*/\* 배경을 저장할 버퍼 \*/*

unsigned int bgdot[PAPER\_SIZE][6];

unsigned char bgcolor[PAPER\_SIZE][PAPER\_SIZE];

*/\* 새로 그려지는 그림을 가리키는 변수 \*/*

paint\* selected;

*/\* 지우개 역할을 하는 변수 \*/*

paint eraser;

1. 프로그램 흐름과 구조

이 그림판 프로그램에서 가장 중요한 자료구조는 paint 구조체입니다. 각 그림마다 하나씩 할당되며 완성된 그림 객체는 buffer에 저장되고 관리됩니다.

**typedef** **struct** tag\_paint

{

point min\_pos;

point max\_pos;

point flag;

unsigned char color;

unsigned int map[PAPER\_SIZE][6];

}paint;

* map 배열은 192\*192 크기의 비트를 저장하기 위한 배열입니다.
* color는 그림의 색깔을 가리킵니다.
* min\_pos와 max\_pos는 그림의 최소, 최대 좌표를 가리킵니다.
* flag는 그림 위치를 조정하기 위한 변수입니다.

flag는 min\_pos와 max\_pos에 더해져서 그림의 위치를 결정합니다. 기본값은 (0, 0)이지만 select기능으로 그림이 이동하면 그에 맞춰서 flag변수가 조정됩니다. flag와 min\_pos, max\_pos을 더한 값은 그림의 크기를 넘을 수 없습니다. (0, 0) ~ (191, 191)

만들어진 그림은 buffer에 저장됩니다.

paint\* buffer[1000];

높은 인덱스에 저장되는 그림이 화면의 가장 위에 있는 그림입니다. 높은 index에 저장된 그림은 select기능에 있어서 우선순위가 있습니다.

초기화 함수 init()를 제외하고 프로그램은 가장 먼저 터치입력을 받는 것으로 시작합니다.

ip = readtouch();

입력받은 좌표는 readcommand() 함수로 위치와 동작을 고려하여 알맞은 명령으로 변환됩니다.

cmd = readcommand(ip); *// convert to command*

이 명령을 바탕으로 모드설정, 색깔설정, 화면지우기, 그리기 등 작업을 구분하여 수행합니다.

프로그램의 중심인 drawing() 함수를 중심으로 설명하겠습니다.

void drawing(input ip)

{

**if**(drawmode>=0 && drawmode <=3)

{

**if**(prev\_pressure == 0 && pressure > 0)

*// 그림 객체 생성*

**else** **if**(prev\_pressure > 0 && pressure == 0 && newpaint != NULL)

*// 객체 생성 마무리*

**else** **if**(prev\_pressure > 0 & pressure > 0 && newpaint != NULL){

**if**(drawmode == 0) *// line 기능*

**else** **if**(drawmode == 1)

*// rect 기능*

**else** **if**(drawmode == 2)

*// oval 기능*

**else** **if**(drawmode == 3)

*// freedraw 기능*

}

}

**else** **if**(drawmode == 4)

*// select 기능*

**else** **if**(drawmode == 5)

*// erase 기능*

}

line, rect, oval, freedraw같은 그림모드에선 첫 터치와 동시에 객체를 생성합니다. 객체를 생성하고 드래그하면 입력에 맞춰서 각 기능을 수행합니다. 터치를 떼면 그림 객체생성을 완료한 것으로 간주하고 buffer에 그림객체를 추가하는 등 마무리작업을 합니다.

line은 화면에 직선을 그리는 기능입니다. line은 1차방정식을 푸는 것을 기반으로 작동합니다. 첫 터치 좌표와 실시간으로 입력되는 터치 좌표를 이용하여 1차방정식을 풀고 이를 이용하여 화면에 선을 그립니다.

a = (ypos2 - ypos1)/(double)(xpos2 - xpos1);

b = ypos1 - a\*xpos1;

*// . . .*

result =(int)(i\*a+b);

dotinmap(i, result);

rect 기능은 화면에 사각형을 그리는 기능입니다. pen모드와 full모드로 나뉘는데 pen모드는 사각형 외각 프레임만 그리고 full모드는 사각형 내부가 전부 채워진 사각형을 그립니다.

**for**(i=xlow; i<=xhigh; i=i+1) *// 상* dotinmap(i, ylow);

**for**(i=xlow; i<=xhigh; i=i+1) *// 하*

dotinmap(i, yhigh);

**for**(i=ylow; i<=yhigh; i=i+1) *// 좌*

dotinmap(xlow, i);

**for**(i=ylow; i<=yhigh; i=i+1) *// 우*

dotinmap(xhigh, i);

*// . . .*

**for**(i=ylow; i<=yhigh; i=i+1){

**for**(j=xlow; j<=xhigh; j=j+1)

dotinmap(j, i);

}

oval 기능은 화면에 타원을 그리는 기능입니다. rect와 마찬가지로 pen모드와 full모드로 나뉘고 각각 외곽 프레임만, 내부가 전부 채워진 타원을 그립니다. oval은 다음과 같은 타원의 방정식을 푸는 것을 기반으로 작동합니다.

**for**(i=xlow; i<=xhigh; i++){

result = (b/a)\*(sqrt(a\*a-(i-center\_x)\*(i-center\_x)));

dotinmap(i, (int)(result+center\_y));

dotinmap(i, (int)(-result+center\_y));

}

**for**(i=ylow; i<yhigh; i++){

int t1, t2;

result = (a/b)\*(sqrt(b\*b-(i-center\_y)\*(i-center\_y)));

t1 = (int)(result+center\_x);

t2 = (int)(-result+center\_x);

dotinmap(t1, i);

dotinmap(t2, i);

**if**(penmode == 1){

**for**(j=t2; j<=t1; j++)

dotinmap(j, i);

}

}

line, rect, oval은 rubber-band를 지원하기 위해 객체 생성할 때마다 createbg()함수로 배경을 만듭니다. 배경은 지금까지 그려진 모든 그림을 취합하여 만듭니다. 이렇게 만들어진 배경으로 화면을 실시간으로 갱신합니다. 배경은 bgdot와 bgcolor배열에 저장합니다.

freedraw 기능은 화면에 임의의 도형을 그리는 기능입니다. 입력좌표에 그대로 점을 찍는 방법은 불연속적인 선이 만들어질 수 있습니다. 따라서 Bezier곡선 알고리즘을 사용하여 연속적인 선을 그립니다.

double t = 0;

**for**(t=0; t<=1.0; t+=(1.0/10.0)){

result = getbezier(p1, p2, p3, t);

dotinmap(result.x, result.y);

}

*// . . .*

point getbezier(point p1, point p2, point p3, double t){

point result;

result.x = p3.x\*(t\*t) + p2.x\*(2\*t\*(1-t)) + p3.x\*((1-t)\*(1-t));

result.y = p3.y\*(t\*t) + p2.y\*(2\*t\*(1-t)) + p3.y\*((1-t)\*(1-t));

**return** result;

}

입력좌표에 그대로 찍는 방식보다는 개선되었지만, 매우 빠르게 그리면 계단현상이 발생하는 것이 문제점입니다.

select 기능은 원하는 그림을 선택하는 기능입니다. 이 기능으로 그림을 움직이거나 원하는 그림만 삭제할 수 있습니다. 그림을 움직일 땐 화면을 갱신하기 위해 createbg() 함수로 배경을 만듭니다. 배경은 선택된 그림을 제외한 다른 모든 그림을 취합하여 만듭니다.

첫 터치시에 selectpaint() 함수로 그림을 선택합니다. 이 때 선택된 그림을 뺀 나머지 그림으로 배경을 만듭니다.

selected = selectpaint(xpos1, ypos1);

**if**(selected != NULL){

createbg(selected);

}

드래그 과정에서 배경과 객체 그림을 실시간으로 갱신합니다.

printbg(minx+fx, miny+fy, maxx+fx, maxy+fy);

printmap(selected);

erase 기능은 지우개로 화면을 지우는 기능입니다. 작동방식은 freedraw와 유사하여 Bezier 곡선 알고리즘을 이용합니다. 실시간으로 입력좌표 위치의 그림을 지우지 않고, 드래그 하면서 eraser객체를 만들고 마지막에 모든 그림을 갱신하는 방법을 사용합니다. 완전히 제거된 그림은 프로그램에서 삭제됩니다.

**for**(t=0; t<=1.0; t+=(1.0/10.0)){

result = getbezier(p1, p2, p3, t);

dotineraser(result.x,result.y);

}

*// . . .*

erasemap();

지우개 기능 외에도 select와 연계하여 선택된 그림을 삭제하는 기능도 제공합니다.

*/\* select와 연계하여 선택된 그림 객체 삭제 \*/*

createbg(e\_selected);

printbg(xpos1, ypos1, xpos2, ypos2);

free(e\_selected);