**Appendix A - Sample source Code**

**Paint.cpp**

#include<math.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#include<malloc.h>

#include<glut.h>

#define CANVASX0 100

#define CANVASX1 800

#define CANVASY 650

struct buffer

{

float r,g,b;

};

struct buffer buf[900][700];

enum shapes {NONE,DRAW\_RECT,FILL\_RECT,LINE,POINT,CIRCLE,FILLED\_CIRCLE,TRIANGLE,FILLED\_TRIANGLE,FILLED\_POLYGON,POLYGON,CLEAR,PENCIL,ERASER,INTERNEL\_CLIP,EXTERNEL\_CLIP,TRANSLATE,SCALE,COPY,PASTE,CUT,FILESAVED,TEXT,BRUSH};

shapes STATUS=NONE;

int char\_xi=0;

float store\_pixels[900][700][3];

float colors[8][3]={{0,0,0},{0,0,1},{0,1,0},{0,1,1},{1,0,0},{1,0,1},{1,1,0},{1,1,1}};

int color\_index=0;

int ttimes=0,ptimes=0,fptimes=0;

int xc,yc;

int xr,yr;

int xps[15],yps[15];

int point\_size=1;

int cw,ch;

void myinit()

{

glClearColor(1.0,1.0,1.0,1.0);

glPointSize(point\_size);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,900,0.0,700);

}

void init\_glut(int argc,char \*\*argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(900,700);

glutInitWindowPosition(0,0);

glutCreateWindow("PAINT");

}

void init\_listeners()

{

void display();

void mousefunc(int,int,int,int);

void mousemotion(int,int);

void keyfunc(unsigned char ,int ,int );

glutDisplayFunc(display);

glutMouseFunc(mousefunc);

glutMotionFunc(mousemotion);

glutKeyboardFunc(keyfunc);

}

//creates a box for each menu

void create\_item(int x,int y,int w,int h,char \* label)

{

glBegin(GL\_LINE\_LOOP);

glVertex2i(x,y);

glVertex2i(x+w,y);

glVertex2i(x+w,y+h);

glVertex2i(x,y+h);

glEnd();

int tempx=x+2;

glRasterPos2i(tempx,y+5);

while(\*label)

{

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13,(\*label));

label++;

tempx+=8;

glRasterPos2i(tempx,y+5);

}

glFlush();

}

void top\_menus()

{

int tempx=110,tempy=660,ii=0,count=7;;

char labels[][7]={"CLEAR","SAVE","OPEN","COPY","CUT","PASTE","EXIT"};

while(ii<count)

{

create\_item(tempx,tempy,80,20,labels[ii]);

ii++;

tempx=tempx+90;

}

}

void right\_menus()

{

int tempx=810,tempy=620,i=0,count=8;

char labels[][10]={"PENCIL","ERASER","iCLIP","eCLIP","Translate","Scale","Text","Brush"};

while(i<count)

{

create\_item(tempx,tempy,80,20,labels[i]);

i++;

tempy=tempy-40;

}

}

void bottom\_colors()

{

int tempx=410,tempy=90,w=30,h=30,count=0;

//different colors

while(count<8)

{

glColor3fv(colors[count]);

glBegin(GL\_POLYGON);

glVertex2i(tempx,tempy);

glVertex2i(tempx+w,tempy);

glVertex2i(tempx+w,tempy-h);

glVertex2i(tempx,tempy-h);

glEnd();

tempx+=50;

count++;

}

glColor3f(0.0,0.0,0.0);

//choosen color

glBegin(GL\_POLYGON);

glVertex2i(410,10);

glVertex2i(710,10);

glVertex2i(710,50);

glVertex2i(410,50);

glEnd();

glFlush();

}

void bottom\_down()

{

int tempx=805,tempy=50,i=0,count=3;

char labels[][10]={{"<<"},{point\_size+'0'},{">>"}};

while(i<count)

{

create\_item(tempx,tempy,25,20,labels[i]);

i++;

tempx=tempx+35;

}

create\_item(805,20,90,20,"Font Size");

}

void place\_items()

{

int tempx=0,tempy=600,i=0,count=10;

char labels[][9]={"FRect","Rect","Line","Point","FCle","Cle","FTri","Tri","Poly","FillP"};

while(i<count)

{

create\_item(tempx,tempy,40,40,labels[i]);

i++;

if(tempx==0)

{

tempx=tempx+60;

}

else

{

tempy=tempy-60;

tempx=0;

}

}

}

int isInsideCanvas(float x,float y)

{

if((x>CANVASX0 & x<CANVASX1) & (y<CANVASY)) return 1;

else return 0;

}

void draw\_point(float x,float y)

{

printf("pointing 1");

glColor3fv(colors[color\_index]);

if(STATUS==ERASER)

{

glColor3f(1.0,1.0,1.0);

}

printf("pointing 2");

if(!isInsideCanvas(x,y)) return;

printf("pointing 3");

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

glFlush();

}

void draw\_lines(int x1,int y1,int x2,int y2)

{

glBegin(GL\_LINES);

glVertex2i(x1,y1);

glVertex2i(x2,y2);

glEnd();

glFlush();

}

void fill\_rect(int x,int y)

{

static int times=0,xt,yt;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

xr=x;yr=y;

return;

}

else{

printf("Entered");

glBegin(GL\_POLYGON);

glVertex2i(xt,yt);

glVertex2i(x,yt);

glVertex2i(x,y);

glVertex2i(xt,y);

glEnd();

glFlush();

STATUS=NONE;

times=0;

}

}

void draw\_rect(int x,int y)

{

static int times=0,xt,yt;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

xr=x;yr=y;

return;

}

else{

printf("Entered");

glBegin(GL\_LINE\_LOOP);

glVertex2i(xt,yt);

glVertex2i(x,yt);

glVertex2i(x,y);

glVertex2i(xt,y);

glEnd();

glFlush();

STATUS=NONE;

times=0;

}

}

void draw\_line(int x,int y)

{

static int times=0,xt,yt;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

return;

}

else{

printf("Entered");

glBegin(GL\_LINES);

glVertex2i(xt,yt);

glVertex2i(x,y);

glEnd();

glFlush();

STATUS=NONE;

times=0;

}

}

float distance(int x1,int y1,int x2,int y2)

{

return sqrt(abs(((double)(x2-x1)\*(x2-x1))+((y2-y1)\*(y2-y1))));

}

void construct\_circle(int xc,int yc,float radius)

{

double theta=0;

while(theta<=360)

{

//printf("circle");

draw\_point(xc+(radius\*cos((double)theta)),yc+(radius\*sin((double)theta)));

theta+=0.5;

}

}

void draw\_circle(int x,int y)

{

static int times=0,xt,yt;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

return;

}

else{

float r=abs(distance(xt,yt,x,y));

construct\_circle(xt,yt,r);

STATUS=NONE;

times=0;

glFlush();

}

}

void fill\_circle(int x,int y)

{

static int times=0,xt,yt;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

return;

}

else{

float radius=abs(distance(xt,yt,x,y));

double theta=0;

glBegin(GL\_POLYGON);

while(theta<=360)

{

glVertex2i(xt+(radius\*cos((double)theta)),yt+(radius\*sin((double)theta)));

theta+=0.5;

}

glEnd();

STATUS=NONE;

times=0;

glFlush();

}

}

void draw\_triangle(int x,int y)

{

static int times=0,xt1,yt1,xt2,yt2;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt1=x;yt1=y;

return;

}

if(times==2)

{

xt2=x;yt2=y;

return;

}

else

{

glBegin(GL\_LINE\_LOOP);

glVertex2i(xt1,yt1);

glVertex2i(xt2,yt2);

glVertex2i(x,y);

glEnd();

glFlush();

STATUS=NONE;

times=0;

}

}

void fill\_triangle(int x,int y)

{

static int times=0,xt1,yt1,xt2,yt2;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt1=x;yt1=y;

return;

}

if(times==2)

{

xt2=x;yt2=y;

return;

}

else

{

glBegin(GL\_POLYGON);

glVertex2i(xt1,yt1);

glVertex2i(xt2,yt2);

glVertex2i(x,y);

glEnd();

glFlush();

STATUS=NONE;

times=0;

}

}

void draw\_polygon(int x,int y)

{

if(!isInsideCanvas(x,y)){

//STATUS=NONE;

//times=0;

return;

}

xps[ptimes]=x;

yps[ptimes]=y;

ptimes++;

fptimes=ptimes;

if(ptimes==1) return;

int i=ptimes-2;

for(i;(i+1)<ptimes;i++)

{

draw\_lines(xps[i],yps[i],xps[i+1],yps[i+1]);

glFlush();

}

//draw\_lines(xps[i],yps[i],xps[0],yps[0]);

glFlush();

}

void fill\_polygon()

{

int i=0;

glBegin(GL\_POLYGON);

for(;i<fptimes;i++)

{

glVertex2i(xps[i],yps[i]);

}

//glVertex2i(xps[0],yps[0]);

glEnd();

glFlush();

fptimes=0;

STATUS=NONE;

}

void select\_rect(int x1,int y1,int x2,int y2)

{

glBegin(GL\_LINE\_LOOP);

glVertex2i(x1,y1);

glVertex2i(x2,y1);

glVertex2i(x2,y2);

glVertex2i(x1,y2);

glEnd();

glFlush();

}

void clear\_rect(int x1,int y1,int x2,int y2)

{

glColor3f(1.0,1.0,1.0);

glBegin(GL\_POLYGON);

glVertex2i(x1,y1);

glVertex2i(x2,y1);

glVertex2i(x2,y2);

glVertex2i(x1,y2);

glEnd();

glFlush();

glColor3fv(colors[color\_index]);

}

void clear\_parameters()

{

if(!(STATUS==FILESAVED))

STATUS==NONE;

ptimes=0;

}

void internal\_clip(int x,int y)

{

static int times=0,xt,yt;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

return;

}

glColor3f(1.0,1.0,1.0);

glBegin(GL\_POLYGON);

glVertex2i(xt,yt);

glVertex2i(x,yt);

glVertex2i(x,y);

glVertex2i(xt,y);

glEnd();

glColor3fv(colors[color\_index]);

glFlush();

times=0;

STATUS=NONE;

}

void externel\_clip(int x ,int y)

{

void clear\_canvas();

static int times=0,xt,yt;

static int pixels[900][700];

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;

return;

}

int width=abs(x-xt);

int height=abs(y-yt);

int gx=(x<xt)?x:xt;

int gy=(y>yt)?y:yt;

int lx=(x>xt)?x:xt;

int ly=(y<yt)?y:yt;

clear\_rect(100,650,799,gy);

clear\_rect(100,ly,799,101);

clear\_rect(100,650,gx,101);

clear\_rect(lx,650,799,101);

//glReadPixels(x,y,width,height,GL\_RGB,GL\_FLOAT,pixels);

//clear\_canvas();

//glDrawPixels(00,00,GL\_RGB,GL\_FLOAT,pixels);

glFlush();

times=0;

STATUS=NONE;

}

void translate(int x,int y)

{

static int xt=0,yt=0,times=0;

static float pixels[900][700][3];

int tx=0,ty=0;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;return;

}

printf("\n Enter the translating factors tx,ty: ");

scanf("%d %d",&tx,&ty);

glReadPixels(xt,y,abs(xt-x),abs(yt-y),GL\_RGB,GL\_FLOAT,pixels);

clear\_rect(xt,yt,x,y);

glRasterPos2i(xt+tx,y+ty);

glDrawPixels(abs(xt-x),abs(yt-y),GL\_RGB,GL\_FLOAT,pixels);

glFlush();

times=0;

STATUS=NONE;

}

void copy(int x,int y)

{

static int xt=0,yt=0,times=0;

int tx=0,ty=0;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;return;

}

cw=abs(xt-x);

ch=abs(yt-y);

glReadPixels(xt,y,abs(xt-x),abs(yt-y),GL\_RGB,GL\_FLOAT,store\_pixels);

if(STATUS==CUT)

{

clear\_rect(xt,yt,x,y);

}

//glRasterPos2i(xt+tx,y+ty);

//glDrawPixels(abs(xt-x),abs(yt-y),GL\_RGB,GL\_FLOAT,pixels);

glFlush();

times=0;

STATUS=NONE;

}

void paste(int x,int y)

{

if(!isInsideCanvas(x,y)) return;

glRasterPos2i(x,y);

glDrawPixels(cw,ch,GL\_RGB,GL\_FLOAT,store\_pixels);

glFlush();

STATUS=NONE;

}

void rotate(int x,int y)

{

static int xt=0,yt=0,times=0;

static float pixels[900][700];

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;return;

}

int w=abs(xt-x);

int h=abs(yt-y);

glReadPixels(xt,y,w,h,GL\_RGB,GL\_FLOAT,pixels);

clear\_rect(xt,yt,x,y);

glTranslatef(xt,y,0);

glRotatef(60,0,0,1);

glTranslatef(-xt,-y,0);

glRasterPos2i(xt,y);

glDrawPixels(w,h,GL\_RGB,GL\_FLOAT,pixels);

glFlush();

STATUS=NONE;

times=0;

}

void scale(int x,int y)

{

static int xt=0,yt=0,times=0;

static int pixels [900][700];

int sx=2,sy=2;

if(!isInsideCanvas(x,y)) return;

times++;

if(times==1)

{

xt=x;yt=y;return;

}

int w=abs(xt-x)\*sx;

int h=abs(yt-y)\*sy;

glReadPixels(xt,y,w,h,GL\_RGB,GL\_FLOAT,pixels);

clear\_rect(xt,yt,x,y);

printf("\n Enter the scale factors sx and sy:");

scanf("%d%d",&sx,&sy);

//printf("\nThe position where the scaled portion should b placed from the current point(x,y):");

//int tx,ty;

//scanf("%d%d",&tx,&ty);

glRasterPos2i(xt+20,y+30);

glPixelZoom(sx,sy);

glDrawPixels(w,h,GL\_RGB,GL\_FLOAT,pixels);

glPixelZoom(1,1);

glFlush();

times=0;

STATUS=NONE;

}

void brushit()

{

point\_size+=15;

glPointSize(point\_size);

}

void clear\_canvas()

{

glColor3f(1.0,1.0,1.0);

glBegin(GL\_POLYGON);

glVertex2i(100,101);

glVertex2i(799,101);

glVertex2i(799,649);

glVertex2i(100,649);

glEnd();

glFlush();

glColor3fv(colors[color\_index]);

}

void draw\_layout()

{

glColor3f(0.0,0.0,0.0);

draw\_lines(0,100,900,100);

draw\_lines(400,0,400,100);

draw\_lines(100,0,100,700);

draw\_lines(800,0,800,700);

draw\_lines(0,650,900,650);

place\_items();

top\_menus();

right\_menus();

bottom\_colors();

bottom\_down();

glFlush();

}

void refresh()

{

//draw layout

draw\_layout();

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

refresh();

glFlush();

}

void post\_message(char \*msg,int x,int y)

{

glRasterPos2i(x,y);

while(\*msg)

{

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13,(\*msg));

msg++;

x+=8;

glRasterPos2i(x,y);

}

glFlush();

}

void keyfunc(unsigned char key,int x,int y)

{

y=700-y;

static int yi=y;

static int xi=x;

if(STATUS==TEXT)

{

if(y!=yi)

{

xi=x;

yi=y;

}

glRasterPos2i(xi,yi);

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13,key);

xi+=8;

glFlush();

}

}

void mousefunc(int btn,int state,int x,int y)

{

static int xp=0,yp=0;

y=700-y;

if(xp==x & yp==y) return;//to prevent from duplicate points

printf("%d %d\n",x,y);

if(btn==GLUT\_LEFT\_BUTTON & state==GLUT\_DOWN)

{

if(isInsideCanvas(x,y))

{

xc=x;yc=y;

if(STATUS==POINT)

{

draw\_point(x,y);

STATUS=NONE;

}

}

else

{

if(STATUS==PENCIL | STATUS==ERASER |STATUS==BRUSH)

{

if(STATUS==ERASER)//RESET THE BLACK FONT

{

glColor3fv(colors[color\_index]);

printf("\nBck\n");

}

if(STATUS==BRUSH)

{

printf("\nPZBck\n");

point\_size=1;

glPointSize(point\_size);

}

STATUS=NONE;

}

}

if(y>100 & y<660 & x>0 & x<100)//check for left menu actions

{

clear\_parameters();

if(y>=600 & y<660)//rectangles

{

//first left menu row

if(x>=0 & x<40)

{

//first option

printf("R");

STATUS=FILL\_RECT;

clear\_rect(101,99,399,1);

post\_message("Filled Rectangle",150,50);

}

else if(x>=60)

{

//second option

printf("r");

STATUS=DRAW\_RECT;

clear\_rect(101,99,399,1);

post\_message("Rectangle",150,50);

}

}

else if(y>=540 & y<600)

{

if(x>=0 & x<40)

{

printf("L");

STATUS=LINE;

clear\_rect(101,99,399,1);

post\_message("Line",150,50);

}

else if(x>=60)

{

printf("p");

STATUS=POINT;

clear\_rect(101,99,399,1);

post\_message("Point",150,50);

}

}

else if(y>=480 & y<540)

{

if(x>=0 & x<40)

{

printf("C");

STATUS=FILLED\_CIRCLE;

clear\_rect(101,99,399,1);

post\_message("Filled Circle",150,50);

}

else if(x>=60)

{

printf("c");

STATUS=CIRCLE;

clear\_rect(101,99,399,1);

post\_message("Circle",150,50);

}

}

else if(y>=420 & y<480)

{

if(x>=0 & x<40)

{

printf("T");

STATUS=FILLED\_TRIANGLE;

clear\_rect(101,99,399,1);

post\_message("Filled Triangle",150,50);

}

else if(x>=60)

{

printf("t");

STATUS=TRIANGLE;

clear\_rect(101,99,399,1);

post\_message("Triangle",150,50);

}

}

else if(y>=340 & y<420)

{

if(x>=0 & x<40)

{

printf("Poly");

STATUS=FILLED\_POLYGON;

clear\_rect(101,99,399,1);

post\_message("Polygon",150,50);

}

else if(x>=60)

{

printf("po");

STATUS=POLYGON;

clear\_rect(101,99,399,1);

post\_message("Fill Polygon",150,50);

}

}

}

if(y>660 & y<680 & x>110 & x<800)//check for top menus

{

clear\_parameters();

if(x>110 & x<190)

{

STATUS=CLEAR;

clear\_rect(101,99,399,1);

post\_message("Clear",150,50);

}

if(x>200 & x<280)

{

printf("\nEnter the filename:");

char fname[50];

scanf("%s",fname);

FILE \*fp=fopen(fname,"w");

glReadPixels(100,100,700,550,GL\_RGB,GL\_FLOAT,buf);

fwrite(&buf,sizeof(buf),1,fp);

fclose(fp);

printf("File saved successfully");

glFlush();

STATUS=FILESAVED;

}

if(x>290 & x<370)

{

printf("\nEnter the filename:");

char fname[50];

scanf("%s",fname);

FILE \*fp=fopen(fname,"r");

fread(&buf,sizeof(buf),1,fp);

glRasterPos2i(100,100);

glDrawPixels(700,550,GL\_RGB,GL\_FLOAT,buf);

fclose(fp);

printf("File retrieved successfully");

glFlush();

STATUS=NONE;

}

if(x>380 & x<460)//copy

{

STATUS=COPY;

}

if(x>470 & x<550)//cut

{

STATUS=CUT;

}

if(x>560 & x<640)//paste

{

STATUS=PASTE;

}

if(x>650 & x<730)

{

exit(0);

}

}

if(y<650 & y>100 & x<900 & x>800) //check for right menus

{

clear\_parameters();

if(y>620 & y<640)

{ printf("PENCIL");

STATUS=PENCIL;

clear\_rect(101,99,399,1);

post\_message("Pencil",150,50);

}

if(y>580 & y<600)

{ printf("ERASER");

STATUS=ERASER;

clear\_rect(101,99,399,1);

post\_message("Eraser",150,50);

}

if( y>540 & y< 560)

{

STATUS=INTERNEL\_CLIP;

printf("iCLIP");

clear\_rect(101,99,399,1);

post\_message("Internal Clipping",150,50);

}

if(y>500 & y<520)

{

STATUS=EXTERNEL\_CLIP;

printf("eClip");

clear\_rect(101,99,399,1);

post\_message("Externel Clipping",150,50);

}

if(y>460 & y<480)

{

STATUS=TRANSLATE;

printf("translate");

clear\_rect(101,99,399,1);

post\_message("Translate",150,50);

}

if(y>420 & y<440)

{

STATUS=SCALE;

printf("scale");

clear\_rect(101,99,399,1);

post\_message("Scale",150,50);

}

if(y>380 & y<400)

{

STATUS=TEXT;

printf("text");

clear\_rect(101,99,399,1);

post\_message("Text",150,50);

}

if(y>340 & y<360)

{

printf("Brush");

STATUS=BRUSH;

brushit();

clear\_rect(101,99,399,1);

post\_message("Brush",150,50);

}

}

//check for color menus

if(x>410 & x<800 & y<90 & y>60)

{

clear\_rect(101,99,399,1);

post\_message("Choose Color",150,50);

if(x<440) { color\_index=0; printf("0");}

else if(x<490) { color\_index=1;printf("1");}

else if(x<540) { color\_index=2;printf("2");}

else if(x<590) { color\_index=3;printf("3");}

else if(x<640) {color\_index=4;printf("4");}

else if(x<690) { color\_index=5;printf("5");}

else if(x<740) {color\_index=6;printf("6");}

else if(x<790) {color\_index=7;printf("7");}

//choosen color

glColor3fv(colors[color\_index]);

glBegin(GL\_POLYGON);

glVertex2i(410,10);

glVertex2i(710,10);

glVertex2i(710,50);

glVertex2i(410,50);

glEnd();

glFlush();

}

//check for font size options

if(x>=805 & x<=900 & y>=50 &y<=70)

{

if(x>=805 & x<=830) //increase font

{

if(point\_size==1) return;

if(point\_size>=8)

{

point\_size-=4;

}

else

{

point\_size--;

}

glPointSize(point\_size);

int tempx=840;

int no=point\_size;

while(no>0)

{

char pz[][5]={{(no%10)+'0'}};

clear\_rect(839,71,866,49);

create\_item(tempx,50,25,20,pz[0]);

no/=10;

tempx+=5;

}

glFlush();

}

if(x>=875 & x<=900)

{

if(point\_size>=8)

{

point\_size+=4;

}

else

{

point\_size++;

}

glPointSize(point\_size);

int tempx=840;

int no=point\_size;

while(no>0)

{

char pz[][5]={{(no%10)+'0'}};

clear\_rect(839,71,866,49);

create\_item(tempx,50,25,20,pz[0]);

no/=10;

tempx+=5;

}

glFlush();

}

}

if(STATUS==FILL\_RECT)

{

fill\_rect(x,y);

glFlush();

}

if(STATUS==DRAW\_RECT)

{

draw\_rect(x,y);

}

if(STATUS==LINE)

{

draw\_line(x,y);

}

if(STATUS==POINT)

{

draw\_point(x,y);

}

if(STATUS==FILLED\_CIRCLE)

{

fill\_circle(x,y);

}

if(STATUS==CIRCLE)

{

draw\_circle(x,y);

}

if(STATUS==TRIANGLE)

{

draw\_triangle(x,y);

}

if(STATUS==FILLED\_TRIANGLE)

{

fill\_triangle(x,y);

}

if(STATUS==FILLED\_POLYGON)

{

draw\_polygon(x,y);

}

if(STATUS==POLYGON)

{

fill\_polygon();

}

if(STATUS==CLEAR)

{

clear\_canvas();

}

if(STATUS==PENCIL)

{

draw\_point(x,y);

}

if(STATUS==ERASER)

{

draw\_point(x,y);

}

if(STATUS==INTERNEL\_CLIP)

{

internal\_clip(x,y);

}

if(STATUS==EXTERNEL\_CLIP)

{

externel\_clip(x,y);

}

if(STATUS==TRANSLATE)

{

translate(x,y);

}

if(STATUS==SCALE)

{

scale(x,y);

}

if(STATUS==COPY)

{

copy(x,y);

}

if(STATUS==PASTE)

{

paste(x,y);

}

if(STATUS==CUT)

{

copy(x,y);

}

if(STATUS==BRUSH)

{

draw\_point(x,y);

}

xp=x;yp=y;

}

}

void mousemotion(int x,int y)

{

y=700-y;

if(STATUS==PENCIL | STATUS==BRUSH)

{

draw\_point(x,y);

}

if(STATUS==ERASER)

{

draw\_point(x,y);

}

}

int main(int argc,char \*\* argv)

{

init\_glut(argc,argv);

init\_listeners();

myinit();

glutMainLoop();

}