**Flood fill**

#include<GL/glut.h>

#include<iostream>

#include<math.h>

using namespace std;

void myInit(void)

{

glClearColor(0.0,0.0,0.0,0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(1.0);

glLineWidth(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,500,0,500);

}

void display(void)

{

}

struct COLOR

{

float r;

float b;

float g;

};

class polygon\_fill

{

int compare\_color(COLOR a,COLOR b)

{

if(a.r==b.r&&a.b==b.b&&a.g==b.g)

return 1;

else

return 0;

}

void draw(int x,int y,COLOR c)

{

glColor3f(c.r,c.b,c.g);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

glFlush();

}

COLOR getpixelcolor(int x,int y)

{

COLOR c;

glReadPixels(x,y,1,1,GL\_RGB,GL\_FLOAT,&c);

return c;

}

public:

/\*void bound\_fill(int x,int y,COLOR fill,COLOR border\_col)

{

COLOR color;

color=getpixelcolor(x,y);

if(compare\_color(color,border\_col)==0&&compare\_color(color,fill)==0)

{

draw(x,y,fill);

bound\_fill(x+1,y,fill,border\_col);

bound\_fill(x,y+1,fill,border\_col);

bound\_fill(x-1,y,fill,border\_col);

bound\_fill(x,y-1,fill,border\_col);

bound\_fill(x-1,y-1,fill,border\_col);

bound\_fill(x-1,y+1,fill,border\_col);

bound\_fill(x+1,y-1,fill,border\_col);

bound\_fill(x+1,y+1,fill,border\_col);

}

} \*/

/\* void flood\_fill(int x,int y,COLOR fill,COLOR bg\_color)

{

COLOR color;

color=getpixelcolor(x,y);

if(compare\_color(color,bg\_color)==1)

{

draw(x,y,fill);

flood\_fill(x+1,y,fill,bg\_color);

flood\_fill(x,y+1,fill,bg\_color);

flood\_fill(x-1,y,fill,bg\_color);

flood\_fill(x,y-1,fill,bg\_color);

}

} \*/

void flood\_fill(int x, int y, COLOR fill, COLOR bg\_color, COLOR border\_col)

{

COLOR color;

color = getpixelcolor(x, y);

if (compare\_color(color, bg\_color) == 1 && !compare\_color(color, border\_col))

{

draw(x, y, border\_col); // Fill the pixel with the boundary color

// Recursively fill neighboring pixels

flood\_fill(x + 1, y, fill, bg\_color, border\_col);

flood\_fill(x, y + 1, fill, bg\_color, border\_col);

flood\_fill(x - 1, y, fill, bg\_color, border\_col);

flood\_fill(x, y - 1, fill, bg\_color, border\_col);

}

}

};

int xc,yc,i=0;

int xarr[100];

int yarr[100];

void menu(int value)

{

polygon\_fill p;

COLOR fill,bcol,back\_col,c;

fill.r=0.0;

fill.b=0.0;

fill.g=1.0;

bcol.r=1.0;

bcol.b=0.0;

bcol.g=0.0;

back\_col.r=0.0;

back\_col.b=0.0;

back\_col.g=0.0;

switch(value)

{

case 1:

p.flood\_fill(xc,500-yc,fill,back\_col,bcol);

cout<<"COMPLETED FLOOD FILL"<<"\n";

break;

/\*p.bound\_fill(xc,500-yc,fill,bcol);

cout<<"COMPLETED BOUNDARY FILL"<<"\n";

break;\*/

case 2:

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_POINTS);

glVertex2i(0,0);

glEnd();

glFlush();

i=0;

cout<<"SCREEN CLEARED"<<"\n";

break;

}

}

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

{

if(key==27)

exit(0);

else

cout<<"you pressed "<<"\n";

}

void mouse(int button,int state,int x,int y)

{

if(state==GLUT\_DOWN)

{

if(button==GLUT\_LEFT\_BUTTON)

{

xarr[i]=x;

yarr[i]=y;

i++;

if(i>1)

{

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINE\_STRIP);

glVertex2i(xarr[i-2],500-yarr[i-2]);

glVertex2i(xarr[i-1],500-yarr[i-1]);

glEnd();

glFlush();

}

}

if(button==GLUT\_RIGHT\_BUTTON)

{

xc=x;

yc=y;

}

}

return;

}

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

{

glutInit(&argc,argv);

glutInitWindowSize(500,500);

glutInitWindowPosition(200,100);

glutCreateWindow("POLYGON FILL");

glutDisplayFunc(display);

glutMouseFunc(mouse);

glutKeyboardFunc(keyboard);

glutCreateMenu(menu);

glutAddMenuEntry("FLOOD FILL",1);

glutAddMenuEntry("CLEAR SCREEN",2);

glutAttachMenu(GLUT\_MIDDLE\_BUTTON);

cout<<"PLEASE FOLLOW THESE STEPS:"<<endl;

cout<<"1. MAKE PLOYGON by USING LEFT CLICK"<<endl;

cout<<"2. SELECT SEED PIXEL BY RIGHT CLICK"<<endl;

cout<<"FOR MENU, use the middle button of the mouse"<<endl;

myInit();

glutMainLoop();

return 0;

}