实验四 填充算法实验

时间：2022年4月13日

地点：信息学院机房2202

1、实验内容

1.教材P66，填充六边形

2.使用opengl，用扫描线填充算法填充多边形

2、实验目的

验证扫描线填充算法，指定任意的多边形边数，填充多边形

3、实验代码

实验一：填充六边形实验代码

#include <windows.h>

#include <GL/glut.h>

#include <stdlib.h>

#include <math.h>

const double TWO\_PI=6.2831853;

/\* Initial display-window size \*/

GLsizei winWidth = 400,winHeight = 400;

GLuint regHex;

class screenPt{

private:

GLint x,y;

public:

/\* Default Constructor:initializes coordinate position to (0.0).\*/

screenPt(){

x=y=0;

}

void setCoords(GLint xCoord,GLint yCoord){

x=xCoord;

y=yCoord;

}

GLint getx()const{

return x;

}

GLint gety()const{

return y;

}

};

static void init(void)

{

screenPt hexVertex,circCtr;

GLdouble theta;

GLint k;

/\*Set circle center coordinates.\*/

circCtr.setCoords(winWidth/2,winHeight/2);

glClearColor(1.0,1.0,1.0,0.0);//Display-window color = white.

/\*Set up a display list for a red regular hexagon.

\*Vertices for the hexagon are six equally spaced

\*points around the circumference of a circle.

\*/

regHex = glGenLists(1); //Get an identifier for the display list.

glNewList(regHex,GL\_COMPILE);

glColor3f(1.0,0.0,0.0); // Set fill color for haxagon to red.

glBegin(GL\_POLYGON);

for(k=0;k<6;k++)

{

theta = TWO\_PI\*k/6.0;

hexVertex.setCoords(circCtr.getx()+150\*cos(theta),circCtr.gety()+150\*sin(theta));

glVertex2i(hexVertex.getx(),hexVertex.gety());

}

glEnd();

glEndList();

}

void regHexagon(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glCallList(regHex);

glFlush();

}

void winReshapeFcn(int newWidth,int newHeight)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,(GLdouble)newWidth,0.0,(GLdouble)newHeight);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowPosition(100,100);

glutInitWindowSize(winWidth,winHeight);

glutCreateWindow("Reshape\_Function & Display\_List Example");

init();

glutDisplayFunc(regHexagon);

glutReshapeFunc(winReshapeFcn);

glutMainLoop();

}

实验二：填充任意多边形代码

//通用扫描线填充算法

#include <GL/glut.h>

#include <stdlib.h>

#include <windows.h>

#define NULL 0

#define WINDOW\_HEIGHT 400

#define WINDOW\_WIDTH 400

struct dcPt

{

int x;

int y;

};

typedef struct tEdge

{

int yUpper;

float xIntersect, dxPerScan;

struct tEdge \* next;

} Edge;

void setPixel(GLint x,GLint y)

{

glBegin(GL\_POINTS);

glVertex2i(x, y);

glEnd();

}

/\* Inserts edge into list in order of increasing xIntersect field. \*/

void insertEdge (Edge \* list, Edge \* edge)

{

Edge \* p, \* q = list;

p = q->next;

while (p != NULL)

{

if (edge->xIntersect < p->xIntersect)

p = NULL;

else

{

q = p;

p = p->next;

}

}

edge->next = q->next;

q->next = edge;

}

/\* For an index, return y-coordinate of next nonhorizontal line \*/

int yNext (int k, int cnt, dcPt \* pts)

{

int j;

if ((k+1) > (cnt-1))

j = 0;

else

j = k + 1;

while (pts[k].y == pts[j].y)

if ((j+1) > (cnt-1))

j = 0;

else

j++;

return (pts[j].y);

}

/\* Store lower-y coordinate and inverse slope for each edge. Adjust and store upper-y coordinate for edges that are the lower member of a monotically increasing or decreasing pair of edges \*/

void makeEdgeRec

(dcPt lower, dcPt upper, int yComp, Edge \* edge, Edge \* edges[])

{

edge->dxPerScan =

(float) (upper.x - lower.x) / (upper.y - lower.y);

edge->xIntersect = lower.x;

if (upper.y < yComp)

edge->yUpper = upper.y - 1;

else

edge->yUpper = upper.y;

insertEdge (edges[lower.y], edge);

}

void buildEdgeList (int cnt, dcPt \* pts, Edge \* edges[])

{

Edge \* edge;

dcPt v1, v2;

int i, yPrev = pts[cnt - 2].y;

v1.x = pts[cnt-1].x;

v1.y = pts[cnt-1].y;

for (i=0; i<cnt; i++)

{

v2 = pts[i];

if (v1.y != v2.y) /\* nonhorizontal line \*/

{

edge = (Edge \*) malloc (sizeof (Edge));

if (v1.y < v2.y) /\* up-going edge \*/

makeEdgeRec (v1, v2, yNext (i, cnt, pts), edge, edges);

else /\* down-going edge \*/

makeEdgeRec (v2, v1, yPrev, edge, edges);

}

yPrev = v1.y;

v1 = v2;

}

}

void buildActiveList (int scan, Edge \* active, Edge \* edges[])

{

Edge \* p, \* q;

p = edges[scan]->next;

while (p)

{

q = p->next;

insertEdge (active, p);

p = q;

}

}

void fillScan (int scan, Edge \* active)

{

Edge \* p1, \* p2;

int i;

p1 = active->next;

while (p1)

{

p2 = p1->next;

for (i=p1->xIntersect; i<p2->xIntersect; i++)

setPixel ((int) i, scan);

p1 = p2->next;

}

}

void deleteAfter (Edge \* q)

{

Edge \* p = q->next;

q->next = p->next;

free (p);

}

/\* Delete completed edges. Update 'xIntersect' field for others \*/

void updateActiveList (int scan, Edge \* active)

{

Edge \* q = active, \* p = active->next;

while (p)

if (scan >= p->yUpper)

{

p = p->next;

deleteAfter (q);

}

else

{

p->xIntersect = p->xIntersect + p->dxPerScan;

q = p;

p = p->next;

}

}

void resortActiveList (Edge \* active)

{

Edge \* q, \* p = active->next;

active->next = NULL;

while (p)

{

q = p->next;

insertEdge (active, p);

p = q;

}

}

void scanFill (int cnt, dcPt \* pts)

{

Edge \*edges[WINDOW\_HEIGHT], \* active;

int i, scan;

for (i=0; i<WINDOW\_HEIGHT; i++)

{

edges[i] = (Edge \*) malloc (sizeof (Edge));

edges[i]->next = NULL;

}

buildEdgeList (cnt, pts, edges);

active = (Edge \*) malloc (sizeof (Edge));

active->next = NULL;

for (scan=0; scan<WINDOW\_HEIGHT; scan++)

{

buildActiveList (scan, active, edges);

if (active->next)

{

fillScan (scan, active);

updateActiveList (scan, active);

resortActiveList (active);

Sleep(10); //停顿10毫秒，放慢填充速度，便于观看填充过程

}

}

/\* Free edge records that have been malloc'ed ... \*/

}

void init (void)

{

glClearColor (1.0, 1.0, 1.0, 0.0); // Set display-window color to white.

glMatrixMode (GL\_PROJECTION); // Set projection parameters.

gluOrtho2D (0.0, WINDOW\_WIDTH, 0.0, WINDOW\_HEIGHT);

}

void myDraw (void)

{

dcPt pts[] =

{

50, 50,

300, 20,

170, 300,

20,120

};

glClear (GL\_COLOR\_BUFFER\_BIT);

glColor3f (0.4, 0.8, 0.0);

scanFill (4, pts); // 第一个参数为填充图元的顶点数，第二个参数为顶点坐标数组

Sleep (20);

glFlush ( );

}

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

{

glutInit (&argc, argv); // Initialize GLUT.

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB); // Set display mode.

glutInitWindowPosition (50, 100); // Set top-left display-window position.

glutInitWindowSize (WINDOW\_WIDTH, WINDOW\_HEIGHT); // Set display-window width and height.

glutCreateWindow ("通用扫描线填充"); // Create display window.

init ( ); // Execute initialization procedure.

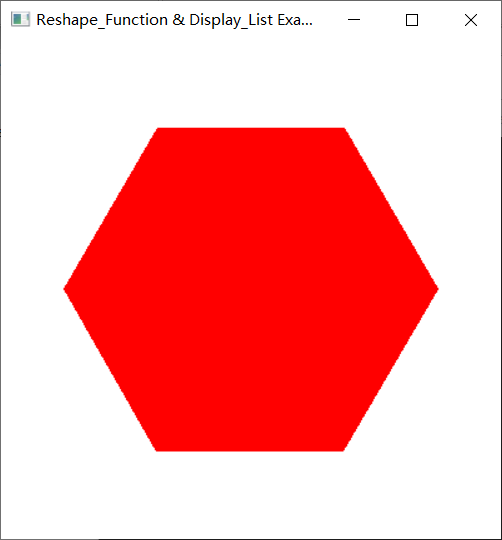
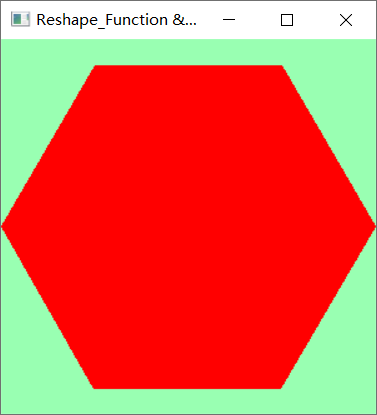
glutDisplayFunc (myDraw); // Send graphics to display window.

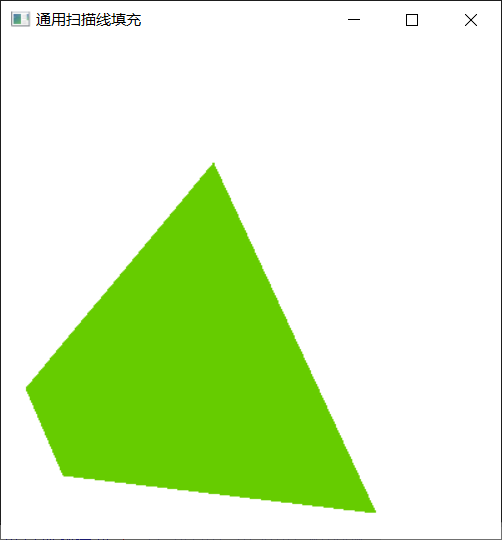
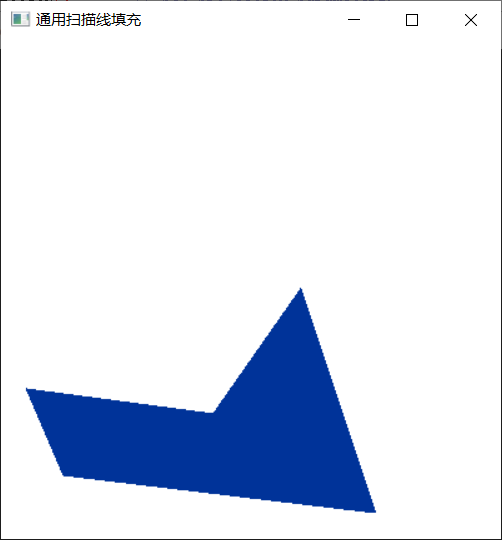
glutMainLoop ( ); // Display everything and wait.

}

4、实验结果

实验一：

实验二：



5、实验总结

填充六边形时，将课本上的代码照着抄下来，在这个过程中，本以为是个很简单的任务，但是由于自己对编译工具的不熟练，在编译运行的时候，还是出现了编译器报错的问题，通过一步一步地排查，解决了问题。

第二个实验需要使用老师给的代码，在看懂的基础上，修改相应的参数，并思考怎么才能把代码优化的更好，运行速度更快。

这次实验总体来说还是比较简单的，但见微知著，把简单的事情做好，也能很好地提高我们的能力。