**西安电子科技大学**

**计算机学院**

**图**

**形**

**学**

**上**

**机**

**报**

**告**

**班 级：** 1403013

**姓 名：** 刘欢

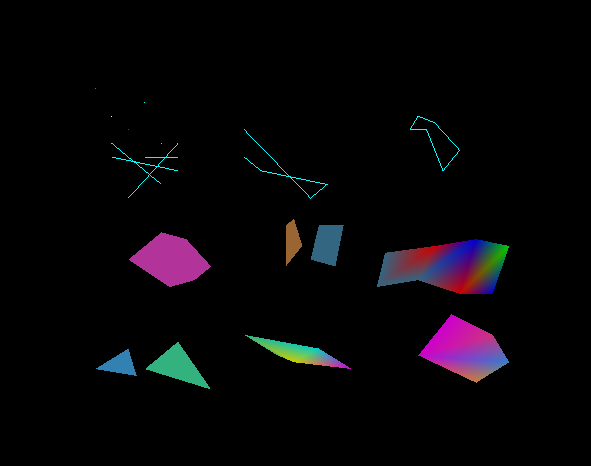
**学 号：** 14030130097

**完成时间：** 2017.5.15

1. 实验目的
2. 理解glut程序框架，能使用opengl绘制基本几何元件
3. 实现直线段的中点算法，加深对算法的理解
4. 实现生成圆的中点算法，加深对算法的理解

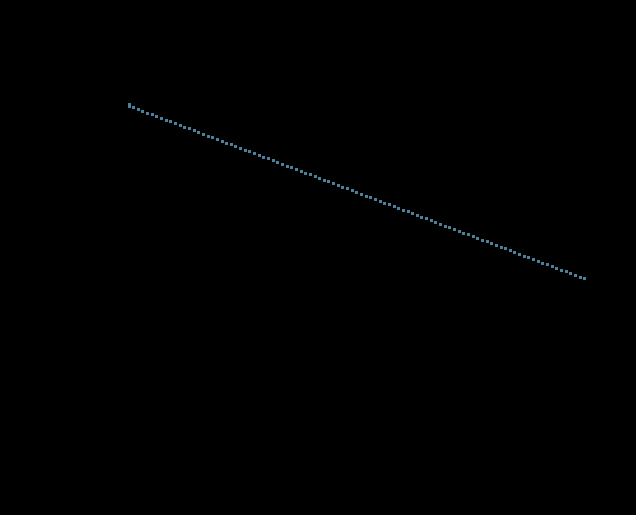
题目一：在屏幕上绘制几何图元，自定义坐标和颜色

实验结果



**二.算法模拟题（可二选一实现）**

1. 采用中点线算法在屏幕上画一条直线。
2. 采用中点圆算法在屏幕上画一个圆。

****

**二．心得体会**

通过本次上机实验，学会了OpenGL中关于绘制几何图元的各个函数的应用，同时通过自己亲手实现了中点线算法，加深了对其原理的理解，和课堂上学到的知识相互印证，巩固了学习成果。

**三．源代码**

**题目一：**

#defineGLUT\_DISABLE\_ATEXIT\_HACK

#include<GL\glut.h>

voidmyinit()

{

glClearColor(0.0,0.0,0.0,0.0);

}

voidchangeSize(GLsizeiw,GLsizeih)

{

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

glOrtho(-20.0,20.0,-20.0\*(GLfloat)h/(GLfloat)w,20.0\*(GLfloat)h/(GLfloat)w,-50.0,50.0);

else

glOrtho(-20.0\*(GLfloat)h/(GLfloat)w,20.0\*(GLfloat)h/(GLfloat)w,-20.0,20.0,-50.0,50.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

voidDrawMyObjects()

{

//画点

glBegin(GL\_POINTS);

glColor3f(2.0,0.0,0.0);

glVertex2f(-10.0,10.0);

glColor3f(1.0,2.0,0.0);

glVertex2f(-11.0,11.0);

glColor3f(0.0,1.0,2.0);

glVertex2f(-9.0,12.0);

glColor3f(3.0,0.0,1.0);

glVertex2f(-12.0,13.0);

glColor3f(0.0,2.0,2.0);

glVertex2f(-8.0,9.0);

glEnd();

//画线段

glBegin(GL\_LINES);

glVertex2f(-11.0,8.0);

glVertex2f(-7.0,7.0);

glVertex2f(-11.0,9.0);

glVertex2f(-8.0,6.0);

glVertex2f(-10.0,5.0);

glVertex2f(-7.0,9.0);

glVertex2f(-9.0,8.0);

glVertex2f(-7.0,8.0);

glEnd();

//画开折线

glBegin(GL\_LINE\_STRIP);

glVertex2f(-3.0,10.0);

glVertex2f(1.0,5.0);

glVertex2f(2.0,6.0);

glVertex2f(-2.0,7.0);

glVertex2f(-3.0,8.0);

glEnd();

//画闭折线

glBegin(GL\_LINE\_LOOP);

glVertex2f(7.0,10.0);

glVertex2f(8.0,10.0);

glVertex2f(9.0,7.0);

glVertex2f(10.0,8.5);

glVertex2f(8.5,10.5);

glVertex2f(7.5,11.0);

glEnd();

//画填充多边形

glBegin(GL\_POLYGON);

glColor3f(0.7,0.2,0.6);

glVertex2f(-6.5,2.0);

glVertex2f(-8.0,2.5);

glVertex2f(-10.0,0.5);

glVertex2f(-7.5,-1.5);

glVertex2f(-6.0,-1.0);

glVertex2f(-5.0,0.0);

glEnd();

//画四边形

glBegin(GL\_QUADS);

glColor3f(0.6,0.4,0.2);

glVertex2f(0.5,1.5);

glVertex2f(0.0,3.5);

glVertex2f(-0.5,3.0);

glVertex2f(-0.5,0.0);

glColor3f(0.2,0.4,0.5);

glVertex2f(3.0,3.0);

glVertex2f(1.5,3.0);

glVertex2f(1.0,0.5);

glVertex2f(2.5,0.0);

glEnd();

//画连接四边形

glBegin(GL\_QUAD\_STRIP);

glVertex2f(5.0,-1.5);

glVertex2f(5.5,1.0);

glVertex2f(7.5,-1.0);

glColor3f(0.8,0.0,0.0);

glVertex2f(8.5,1.5);

glVertex2f(10.0,-2.0);

glColor3f(0.0,0.0,0.8);

glVertex2f(11.0,2.0);

glVertex2f(12.0,-2.0);

glColor3f(0.0,0.8,0.0);

glVertex2f(13.0,1.5);

glEnd();

//画三角形

glBegin(GL\_TRIANGLES);

glColor3f(0.2,0.5,0.7);

glVertex2f(-10.0,-6.0);

glVertex2f(-12.0,-7.5);

glVertex2f(-9.5,-8.0);

glColor3f(0.2,0.7,0.5);

glVertex2f(-9.0,-7.5);

glVertex2f(-7.0,-5.5);

glVertex2f(-5.0,-9.0);

glEnd();

//画连续三角形

glBegin(GL\_TRIANGLE\_STRIP);

glVertex2f(-1.0,-6.5);

glVertex2f(-3.0,-5.0);

glColor3f(0.8,0.8,0.0);

glVertex2f(0.0,-7.0);

glColor3f(0.0,0.8,0.8);

glVertex2f(1.5,-6.0);

glColor3f(0.8,0.0,0.8);

glVertex2f(3.5,-7.5);

glEnd();

//画扇形三角形

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(7.5,-6.5);

glVertex2f(9.5,-3.5);

glColor3f(0.8,0.2,0.5);

glVertex2f(12.0,-5.0);

glColor3f(0.2,0.5,0.8);

glVertex2f(13.0,-7.0);

glColor3f(0.8,0.5,0.2);

glVertex2f(11.0,-8.5);

glEnd();

}

voidRenderScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,1.0,0.0);

DrawMyObjects();

glFlush();

}

intmain()

{

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("opengl画几何图元");

myinit();

glutReshapeFunc(changeSize);

glutDisplayFunc(RenderScene);

glutMainLoop();

return0;

}

**题目二：**  
#defineGLUT\_DISABLE\_ATEXIT\_HACK

#include<GL\glut.h>

#include<math.h>

voidinit()

{

glClearColor(0.0,0.0,0.0,0.0);

}

voidchange\_Size(GLsizeiw,GLsizeih)

{

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

glOrtho(-200.0,200.0,-200.0\*(GLfloat)h/(GLfloat)w,200.0\*(GLfloat)h/(GLfloat)w,-50.0,50.0);

else

glOrtho(-200.0\*(GLfloat)h/(GLfloat)w,200.0\*(GLfloat)h/(GLfloat)w,-200.0,200.0,-50.0,50.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

voidLine(GLintx0,GLinty0,GLintxn,GLintyn)

{

GLinta=yn-y0;

GLintb=-(xn-x0);

GLintc=xn\*y0-x0\*yn;

GLintdeltaA=0;

GLintdeltaB=0;

if(a!=0){

deltaA=a/abs(a);

}

if(b!=0){

deltaB=-1\*b/abs(b);

}

GLintx=x0;

GLinty=y0;

if(abs(a)<abs(b)){

GLintsteps=abs(b);

for(GLinti=0;i<steps;i++){

if(0<(a\*(x+1)+b\*(y+0.5)+c)){

y+=deltaA;

}

x+=deltaB;

glBegin(GL\_POINTS);

glColor3f(0.3,0.5,0.6);

glPointSize(3.0);

glVertex2i(x,y);

glEnd();

}

}

else{

GLintsteps=abs(a);

for(GLinti=0;i<steps;i++){

if(0<=(a\*(x+1)+b\*(y+0.5)+c)){

x+=deltaB;

}

y+=deltaA;

glBegin(GL\_POINTS);

glColor3f(0.3,0.5,0.6);

glPointSize(3.0);

glVertex2i(x,y);

glEnd();

}

}

}

voidRender\_Scene()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.8,0.5);

glPointSize(3.0);

Line(0,100,100,0);

glFlush();

}

intmain()

{

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("中点线算法");

init();

glutReshapeFunc(change\_Size);

glutDisplayFunc(Render\_Scene);

glutMainLoop();

return0;

**}**