//GLRect.cpp

#include <gl/glut.h>

void init()

{

glClearColor(0.0f,0.0f,1.0f,1.0f);

}

void RenderScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT);//用当前的清除色清除窗口

glColor3f(1.0f,0.0f,0.0f);//把当前的绘图颜色设置为红色

glRectf(-25.0f,25.0f,25.0f,-25.0f);//用当前颜色绘制一个填充矩形

glFlush();//刷新绘图命令

}

//当窗口改变大小时由GLUT函数库调用

void ChangeSize(GLsizei w,GLsizei h)

{

GLfloat aspectRatio;

// 防止被0除

if(h==0)

h=1;

//把视口设置为窗口的大小

glViewport(0,0,w,h);

//重置坐标系统

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//建立裁剪区域（左、右、底、顶、近、远）

aspectRatio = (GLfloat)w/(GLfloat)h;

if(w<=h)

glOrtho(-100.0,100.0,-100/aspectRatio,100.0/aspectRatio,1.0,-1.0);

else

glOrtho(-100.0\*aspectRatio,100.0\*aspectRatio,-100.0,100.0,1.0,-1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void main()

{

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("GLRect");

init();

glutDisplayFunc(RenderScene);

glutReshapeFunc(ChangeSize);

glutMainLoop();

}

//GLRectFlash.cpp

#include <gl/glut.h>

//方块的初始位置和大小

GLfloat x1=0.0f;

GLfloat y1 = 0.0f;

GLfloat rsize = 25;

//在x和y方向的步进大小

GLfloat xstep = 1.0f;

GLfloat ystep = 1.0f;

//追踪窗口宽度和高度的变化

GLfloat windowWidth;

GLfloat windowHeight;

void init()

{

glClearColor(0.0f,0.0f,1.0f,1.0f);

}

void RenderScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT);//用当前的清除色清除窗口

glColor3f(1.0f,0.0f,0.0f);//把当前的绘图颜色设置为红色

glRectf(x1,y1,x1+rsize,y1-rsize);//用当前颜色绘制一个填充矩形

glutSwapBuffers();//刷新绘图命令并进行交换

}

//当窗口改变大小时由GLUT函数库调用

void ChangeSize(GLsizei w,GLsizei h)

{

GLfloat aspectRatio;

// 防止被0除

if(h==0)

h=1;

//把视口设置为窗口的大小

glViewport(0,0,w,h);

//重置坐标系统

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//建立裁剪区域（左、右、底、顶、近、远）

aspectRatio = (GLfloat)w/(GLfloat)h;

if(w<=h)

{

windowWidth = 100;

windowHeight = 100/aspectRatio;

glOrtho(-100.0,100.0,-windowHeight,windowHeight,1.0,-1.0);

}

else

{

windowWidth = 100\*aspectRatio;

windowHeight = 100;

glOrtho(-windowWidth,windowWidth,-100.0,100.0,1.0,-1.0);

}

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

//当空闲时由GLUT函数库调用（窗口未改变大小或移动时）

void TimerFunction(int value)

{

//在到达左边或右边时翻转方向

if(x1>windowWidth-rsize||x1<-windowWidth)

xstep = -xstep;

//在到达顶边或底边时翻转方向

if(y1>windowHeight||y1<-windowHeight+rsize)

ystep = -ystep;

//实际移动方块

x1 += xstep;

y1 += ystep;

//检查边界。这是为了防止方块在反弹时窗口变小，使方块出现在新的裁剪区域之外

if(x1>(windowWidth-rsize+xstep))

x1 = windowWidth-rsize-1;

else if(x1<-(windowWidth+xstep))

x1 = -windowWidth-1;

if(y1>(windowHeight+ystep))

y1 = windowHeight-1;

else if(y1<-(windowHeight-rsize+ystep))

y1 = -windowHeight+rsize-1;

//用新的坐标重新绘制场景

glutPostRedisplay();

glutTimerFunc(33,TimerFunction,1);

}

void main()

{

glutInitDisplayMode(GLUT\_RGB|GLUT\_DOUBLE);

glutCreateWindow("GLRectFlash");

init();

glutDisplayFunc(RenderScene);

glutReshapeFunc(ChangeSize);

glutTimerFunc(33,TimerFunction,1);

glutMainLoop();

}

//drawGeom.cpp

#include <gl/glut.h>

void myinit()

{

glClearColor(0.0,0.0,0.0,0.0);

}

void changeSize(GLsizei w,GLsizei h)

{

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();

}

void DrawMyObjects()

{

//画点

glBegin(GL\_POINTS);

glColor3f(1.0,0.0,0.0);

glVertex2f(-10.0,11.0);

glColor3f(1.0,1.0,0.0);

glVertex2f(-9.0,10.0);

glColor3f(0.0,1.0,1.0);

glVertex2f(-8.0,12.0);

glEnd();

//画线段

glBegin(GL\_LINES);

glColor3f(1.0,1.0,0.0);

glVertex2f(-11.0,8.0);

glVertex2f(-7.0,7.0);

glColor3f(1.0,0.0,1.0);

glVertex2f(-11.0,9.0);

glVertex2f(-8.0,6.0);

glEnd();

//画开折线

glBegin(GL\_LINE\_STRIP);

glColor3f(0.0,1.0,0.0);

// glPointSize(5);

glVertex2f(-3.0,9.0);

glVertex2f(2.0,6.0);

glVertex2f(3.0,8.0);

glVertex2f(-2.5,6.5);

glEnd();

//画闭折线

glBegin(GL\_LINE\_LOOP);

glColor3f(0.0,1.0,1.0);

glVertex2f(7.0,7.0);

glVertex2f(8.0,8.0);

glVertex2f(9.0,6.5);

glVertex2f(10.3,7.5);

glVertex2f(11.5,6.0);

glVertex2f(7.5,6.0);

glEnd();

//画填充多边形

glBegin(GL\_POLYGON);

glColor3f(0.5,0.3,0.7);

glVertex2f(-7.0,2.0);

glVertex2f(-8.0,3.0);

glVertex2f(-10.3,0.5);

glVertex2f(-7.5,-2.0);

glVertex2f(-6.0,-1.0);

glEnd();

//画四边形

glBegin(GL\_QUADS);

glColor3f(0.7,0.5,0.2);

glVertex2f(0.0,2.0);

glVertex2f(-1.0,3.0);

glVertex2f(-3.3,0.5);

glVertex2f(-0.5,-1.0);

glColor3f(0.5,0.7,0.2);

glVertex2f(3.0,2.0);

glVertex2f(2.0,3.0);

glVertex2f(0.0,0.5);

glVertex2f(2.5,-1.0);

glEnd();

//画连接四边形

glBegin(GL\_QUAD\_STRIP);

glVertex2f(6.0,-2.0);

glVertex2f(5.5,1.0);

glVertex2f(8.0,-1.0);

glColor3f(0.8,0.0,0.0);

glVertex2f(9.0,2.0);

glVertex2f(11.0,-2.0);

glColor3f(0.0,0.0,0.8);

glVertex2f(11.0,2.0);

glVertex2f(13.0,-1.0);

glColor3f(0.0,0.8,0.0);

glVertex2f(14.0,1.0);

glEnd();

//画三角形

glBegin(GL\_TRIANGLES);

glColor3f(0.2,0.5,0.7);

glVertex2f(-10.0,-5.0);

glVertex2f(-12.3,-7.5);

glVertex2f(-8.5,-6.0);

glColor3f(0.2,0.7,0.5);

glVertex2f(-8.0,-7.0);

glVertex2f(-7.0,-4.5);

glVertex2f(-5.5,-9.0);

glEnd();

//画连续三角形

glBegin(GL\_TRIANGLE\_STRIP);

glVertex2f(-1.0,-8.0);

glVertex2f(-2.5,-5.0);

glColor3f(0.8,0.8,0.0);

glVertex2f(1.0,-7.0);

glColor3f(0.0,0.8,0.8);

glVertex2f(2.0,-4.0);

glColor3f(0.8,0.0,0.8);

glVertex2f(4.0,-6.0);

glEnd();

//画扇形三角形

glBegin(GL\_TRIANGLE\_FAN);

glVertex2f(8.0,-6.0);

glVertex2f(10.0,-3.0);

glColor3f(0.8,0.2,0.5);

glVertex2f(12.5,-4.5);

glColor3f(0.2,0.5,0.8);

glVertex2f(13.0,-7.5);

glColor3f(0.8,0.5,0.2);

glVertex2f(10.5,-9.0);

glEnd();

}

void RenderScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,1.0,0.0);

DrawMyObjects();

glFlush();

}

int main()

{

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Geometric Primitive Types");

myinit();

glutReshapeFunc(RenderScene);

glutDisplayFunc(changeSize);

glutMainLoop();

return 0;

}

//LineDDA.cpp

#include <gl/glut.h>

#include <math.h>

typedef struct {

float x,y;

}point;

void init()

{

glClearColor(0.0f,0.0f,0.0f,1.0f);

}

int LineDDA(int x0,int y0,int x1,int y1,point pixels[])

{

/\*int x,num=0;

float dy,dx,y,m;

dx = x1-x0;

dy = y1-y0;

m = dy/dx;

y = y0;

for(x=x0;x<=x1;x++)

{

pixels[num].x = x;

pixels[num].y = y;

y += m;

num ++;

}

return num;\*/

int num,x,y,dx,dy;

float k,xf,yf;

dx = x1-x0;

dy = y1-y0;

num = 0;

k = dy\*1.0/dx;

if(dx==0) //垂直线

{

for(y=y0;y<=y1;y++)

{

pixels[num].x = x0;

pixels[num].y = y;

num ++;

}

}

else if(dy==0) //水平线

{

for(x=x0;x<=x1;x++)

{

pixels[num].x = x;

pixels[num].y = y0;

num ++;

}

}

else if(fabs(k)<=1)

{

yf = y0;

for(x=x0;x<=x1;x++)

{

pixels[num].x = x;

pixels[num].y = yf;

num ++;

yf += k;

}

}

else

{

xf = x0;

for(y=y0;y<=y1;y++)

{

pixels[num].x = xf;

pixels[num].y = y;

num ++;

xf += 1/k;

}

}

return num;

}

void drawLine(int x1,int y1,int x2,int y2)

{

point pixels[100];

int num;

int i;

num = LineDDA(x1,y1,x2,y2,pixels);

glBegin(GL\_POINTS);

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

glVertex2f(pixels[i].x,pixels[i].y);

glEnd();

}

void RenderScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0f,0.0f,0.f);

drawLine(10,10,80,60);

glFlush();

}

void ChangeSize(GLsizei w,GLsizei h)

{

GLfloat aspectRatio;

if(h==0)

h = 1;

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

aspectRatio = (GLfloat)w/(GLfloat)h;

if(w<=h)

glOrtho(-100.0,100.0,-100.0/aspectRatio,100.0/aspectRatio,1.0,-1.0);

else

glOrtho(-100.0\*aspectRatio,100.0\*aspectRatio,-100.0,100.0,1.0,-1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void main()

{

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("LineDDA");

init();

glutDisplayFunc(RenderScene);

glutReshapeFunc(ChangeSize);

glutMainLoop();

}