实验三 中点Bresenham算法生成直线段

时间：2022年3月30日

地点：信息学院2202机房

1、实验内容

通过中点Bresenham算法以及改进的Bresenham算法生成直线段

2、实验目的

熟悉OpenGL并且根据原理实现Bresenham算法

3、实验代码

|  |
| --- |
| #include <windows.h>  #include <GL/glut.h>  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  void setPixel(int x, int y)  {  glPointSize(5.0f);  glBegin(GL\_POINTS);  glVertex2i(x, y);  glEnd();  glFlush();  }  void lineBres(int x0, int y0, int xEnd, int yEnd)  {  int dx = fabs(xEnd - x0), dy = fabs(yEnd - y0);  int p = 2 \* dy - dx;  int twoDy = 2 \* dy, twoDyMinusDx = 2 \* (dy - dx);  int x, y;  if (x0 > xEnd)  {  x = xEnd;  y = yEnd;  xEnd = x0;  }  else  {  x = x0;  y = y0;  }  setPixel(x, y);  while (x < xEnd)  {  x++;  if (p < 0)  p += twoDy;  else  {  y++;  p += twoDyMinusDx;  }  setPixel(x, y);  }  }  void init(void)  {  glClearColor(1.0, 1.0, 1.0, 0.0); //设置显示窗口颜色为白色  glMatrixMode(GL\_PROJECTION);  gluOrtho2D(0.0, 200.0, 0.0, 150.0); //设置窗口位置  }  void myDisplay()  {  int x0, y0, xEnd, yEnd;  //接收起点终点位置  scanf\_s("%d %d %d %d", &x0, &y0, &xEnd, &yEnd);  lineBres(x0, y0, xEnd, yEnd);  }  void main(int argc, char \*\*argv)  {  glutInit(&argc, argv); //初始化  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); //设置显示模式  glutInitWindowPosition(50, 100); //设置窗口位置  glutInitWindowSize(400, 300); //设置窗口长宽  glutCreateWindow("Bresenham算法"); //设置标题  init();  glutDisplayFunc(&myDisplay);  glutMainLoop();  } |

改进后的算法：

|  |
| --- |
| #include <windows.h>  #include <GL/glut.h>  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  void setPixel(int x, int y)  {  glPointSize(5.0f);  glBegin(GL\_POINTS);  glVertex2i(x, y);  glEnd();  glFlush();  }  void init(void)  {  glClearColor(1.0f, 1.0f, 1.0f, 0.0f); //设置显示窗口颜色为白色  glMatrixMode(GL\_PROJECTION);  gluOrtho2D(0.0f, 200.0f, 0.0f, 150.0f); //设置规划参数  }  void proBresenhamLine(int x0, int y0, int x1, int y1)  {  int dx, dy, e, x, y;  dx = x1 - x0;  dy = y1 - y0;  e = -dx;  x = x0;  y = y0;  while (x <= x1)  {  setPixel(x, y);  x++;  e = e + 2 \* dy;  if (e > 0)  {  y++;  e = e - 2 \* dx;  }  }  }  void myDisplay()  {  int x0, y0, xEnd, yEnd;  //s输入起点中点位置  scanf\_s("%d %d %d %d", &x0, &y0, &xEnd, &yEnd);  proBresenhamLine(x0, y0, xEnd, yEnd);  }  void main(int argc, char \*\*argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); //初始化显示模式  glutInitWindowSize(600, 600); //设置窗口尺寸  glutInitWindowPosition(200, 100); //设置窗口位置  glutCreateWindow("Bresenham算法"); //创建窗口  glutDisplayFunc(&myDisplay); //注册显示函数  glClearColor(1.0f, 1.0f, 1.0f, 1.0f); //设置背景颜色  glColor3f(1.0f, 0.0f, 0.0f); //设置绘制颜色  glMatrixMode(GL\_PROJECTION);  gluOrtho2D(0.0, 600, 0.0, 600); //设置投影区域  glutMainLoop(); //进入程序循环  } |

4、实验总结

本次实验主要进行的是使用Bresenham算法实现了直线的绘制。在上个实验实现的DDA算法可以看到，由于在循环中涉及实型数据的加减运算，因此直线的生成速度较慢。生成直线的算法中，Bresenham算法是最有效的算法之一。Bresenham算法是一种基于误差判别式来生成直线的方法。通过中点和判别式的计算快速得到下一个选择的点的位置，减少了循环的计算量。