山东大学 计算机科学与技术 学院

计算机图形学 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201300130043 | 姓名： 李园 | | 班级： 13级计算机4班 |
| 实验题目： 实验一 实现基本图元的绘制算法 | | | |
| 实验学时：2 | | 实验日期： 2016年3月30日 | |
| 实验目的：  1. 掌握OpenGL环境的配置方法。  2. 熟悉OpenGL应用程序基础架构。  3. 熟练掌握简单OpenGL应用程序的建立、调试与运行。  4. 培养良好的编程习惯和风格，并且学习撰写实验报告。 | | | |
| 硬件环境：  PC电脑 | | | |
| 软件环境：  操作系统：Windows 10  编程软件：Visual Studio 2015 | | | |
| 实验步骤与内容：  1. 实现直线的DDA算法、正负法、Bresenham算法。  实验核心代码如下：   * 1. DDA算法：   void dda\_line(int xa, int ya, int xb, int yb) {  GLfloat delta\_x, delta\_y, x, y;  int dx, dy, steps;  dx = xb - xa;  dy = yb - ya;  if (abs(dx)>abs(dy))  steps = abs(dx);  else  steps = abs(dy);  delta\_x = (GLfloat)dx / (GLfloat)steps;  delta\_y = (GLfloat)dy / (GLfloat)steps;  x = xa;  y = ya;  glClear(GL\_COLOR\_BUFFER\_BIT);  glBegin(GL\_POINTS);  glVertex3f(x, y, 0);  for (int i = 0; i <= steps; i++)  {  x += delta\_x;  y += delta\_y;  glBegin(GL\_POINTS);  glVertex3f(x, y, 0);  glEnd();  }  }  1.2 正负法：  void sign\_line(int xa, int ya, int xb, int yb)  {  int x = xa, y = ya;  int a = ya - yb, b = xb - xa;  int cx, cy;  if (b >= 0)  {  //xb >= xa的情况  cx = 1;  //从左向右画  }  else {  // x2 < x1 的情况  b = -b; //使 b>=0  cx = -1; // 从右往左画  } if (a <= 0) {  //y2 >= y1 的情况  cy = 1; // 从下往上画  }  else {  //y2 < y1 的情况  a = -a; //使 a<=0  cy = -1; // 从上往下画  }  int d, d1, d2;  glBegin(GL\_POINTS);  if (-a <= b) // 斜率绝对值 <= 1  {  d = 2 \* a + b;  d1 = 2 \* a;  d2 = 2 \* (a + b);  while (x != xb)  {  if (d < 0)  {//去右上方像素点  y += cy;  d += d2;  }  else {  //取右下方像素点  d += d1;  }  x += cx;  glVertex2f(x, y);  }  }  else  // 斜率绝对值 > 1  {  //因两种情况以直线y=x为轴对称  //可轻松将斜率绝对值大于1的情况转换为斜率的绝对值小于1的情况  d = 2 \* b + a;  d1 = 2 \* a;  d2 = 2 \* (a + b);  while (y != yb)  {  if (d < 0)  {  d += d1;  }  else  {  x += cx;  d += d2;  }  y += cy;  glVertex2f(x, y);  }  }  glEnd();  }  1.3 Bresenham算法：  void bresenham\_line(int x1, int y1, int x2, int y2)  {  int x = x1;  int y = y1;  int dx = abs(x2 - x1);  int dy = abs(y2 - y1);  int s1 = x2 > x1 ? 1 : -1;  int s2 = y2 > y1 ? 1 : -1;  bool change = false;  // 默认不互换 dx、dy  if (dy > dx)  // 当斜率大于 1 时，dx、dy 互换  {  int temp = dx;  dx = dy;  dy = temp;  change = true;  }  int p = 2 \* dy - dx;  glBegin(GL\_POINTS);  for (int i = 0; i < dx; i++)  {  glVertex2f(x, y);  if (p >= 0)  {  if (!change)  // 当斜率 < 1 时，选取上下象素点  y += s2;  else// 当斜率 > 1 时，选取左右象素点  x += s1;  p -= 2 \* dx;  } if (!change)  x += s1;  // 当斜率 < 1 时，取 x  else y += s2;  // 当斜率 > 1 时，取 y  p += 2 \* dy;  }  glEnd();  }  实验结果：    黄、蓝、红依次代表DDA,正负法和bresenham方法 1  2. 实现圆弧的正负法和Bresenham算法。  方法代码：  //画点  void gl\_Point(int x, int y) {  glBegin(GL\_POINTS);  glVertex2i(x, y);  glEnd();  }  //基于正负法实现画圆弧 传入参数为半径,圆心坐标，象限  void Arc\_Sign(int radius,int centerX,int centerY,int area){  int x,y,f;  x=0;y=0+radius;f=0;  while(y>0){  switch(area){  case 1:  gl\_Point(x+centerX,y+centerY);  break;  case 2:  gl\_Point(-x+centerX,y+centerY);  break;  case 3:  gl\_Point(-x+centerX,-y+centerY);  break;  case 4:  gl\_Point(x+centerX,-y+centerY);  break;  }  if(f>0){  f=f-2\*y+1;  y=y-1;  }else{  f=f+2\*x+1;  x=x+1;  }  }  if(y==centerY){  gl\_Point(x,y);  }  }  //Bresenham画圆算法  void bresenhamArc(int R, int centerX, int centerY, int area) {  int x, y, d;  x = 0; y = R; d = 3 - 2 \* R;  while (x<y) {  switch (area) {  case 1:  gl\_Point(x + centerX, y + centerY);  break;  case 2:  gl\_Point(y + centerX, x + centerY);  break;  case 3:  gl\_Point(y + centerX, -x + centerY);  break;  case 4:  gl\_Point(x + centerX, -y + centerY);  break;  case 5:  gl\_Point(-x + centerX, -y + centerY);  break;  case 6:  gl\_Point(-y + centerX, -x + centerY);  break;  case 7:  gl\_Point(-y + centerX, x + centerY);  break;  case 8:  gl\_Point(-x + centerX, y + centerY);  break;  }  if (d<0) {  d = d + 4 \* x + 6;  }  else {  d = d + 4 \* (x - y) + 10;  y = y - 1;  }  x = x + 1;  }  if (x == y) {  gl\_Point(x, y);  }  }  //display对象  void display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3f(0.0, 0.0, 1.0);  bresenhamArc(500, 0, 0, 1);  glColor3f(0.0, 1.0, 0.0);  bresenhamArc(420, 20, 20, 1);  glColor3f(0.0, 1.0, 1.0);  bresenhamArc(340, 40, 40, 1);  glColor3f(1.0, 0.0, 0.0);  bresenhamArc(260, 60, 60, 1);  glColor3f(1.0, 0.0, 1.0);  bresenhamArc(180, 80, 80, 1);  glColor3f(1.0, 1.0, 0.0);  bresenhamArc(100, 100, 100, 1);  glColor3f(1.0, 1.0, 1.0);  bresenhamArc(20, 120, 120, 1);  glFlush();  }  实验结果截图：    短 bresenham八分弧，长 正负法 四分弧    变形 四分弧 1    变形 四分弧 2  3. 利用上述完成的算法绘制中国象棋的棋盘和棋子。  主要代码：  //display对象  void display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT);    //棋盘格 面  glColor3f(1.0, 1.0, 1.0);  glPointSize(3.0f);  bresenham\_line(45, 20, 45,480);  bresenham\_line(45, 480, 455, 480);  bresenham\_line(455, 480, 455, 20);  bresenham\_line(455, 20, 45, 20);  glPointSize(1.0f);  bresenham\_line(50, 25, 50, 475);  bresenham\_line(50, 475, 450, 475);  bresenham\_line(450, 475, 450, 25);  bresenham\_line(450, 25, 50, 25);  for (int i = 75; i < 426; i=i+50)  {  bresenham\_line(50, i, 450, i);  }  for (int i = 100; i < 401; i+=50)  {  bresenham\_line(i, 25, i, 225);  bresenham\_line(i, 275, i, 475);  }  //面 斜线  bresenham\_line(200, 25, 300, 125);  bresenham\_line(200, 125, 300, 25);  bresenham\_line(200, 475, 300, 475);  bresenham\_line(200, 375, 300, 375);    //棋盘 子  glPointSize(3.0f);  for (int i = 50; i < 451; i=i+50)  {  for (int j = 1; j < 9; j++)  {  bresenhamArc(20, i, 25, j);  bresenhamArc(20, i, 475, j);  }  }  for (int i = 50; i < 451; i = i + 100)  {  for (int j = 1; j < 9; j++)  {  bresenhamArc(20, i, 225, j);  bresenhamArc(20, i, 275, j);  }  }  for (int i = 100; i < 401; i = i + 300)  {  for (int j = 1; j < 9; j++)  {  bresenhamArc(20, i, 125, j);  bresenhamArc(20, i, 375, j);  }  }  glFlush();    }  //主程序入口  int main(int argc, char\*\* argv)  {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);  glutInitWindowSize(500, 500);  glutInitWindowPosition(600, 50);  glutCreateWindow("Chinese ChessBoard");  glutDisplayFunc(display);  myinit();  glutMainLoop();  }  实验结果截图： | | | |
| 结论分析与体会：  经过了对直线和圆弧的生成算法的理解和运用，成功的画出了最后的棋牌的元素，增强了理解和对c++调用opengl库的熟悉度。而且从长远角度考虑，画点的方法宜用3D，方便三维视图的构建。在选点画圆（棋子）的过程中注意简化代码~ | | | |