**北京师范大学珠海分校**

**【计算机图形学】实验报告**

**班级：** 数媒2班 **学号：** 1801050056  **姓名：** 何金城

|  |  |  |  |
| --- | --- | --- | --- |
| **实验五 （名称）基本图元的生成算法** | | | |
| **成 绩** |  | **（日期、星期、节次）** |  |
| **指导教师** |  | **地点** |  |
| **一、实验目的：**  1、掌握DDA直线生成算法。  2、掌握中点画圆法的生成算法。  3、掌握反走样思想和算法。  4、进一步提高综合编程的能力 | | | |
| **二、实验要求：**  已知基本图元程序basic\_primitive\_generation\_algorithm.cpp，  完成以下实验内容：  用DDA算法替代程序中的直线绘制语句，验证DDA直线生成算法，  用中点画圆法 替代程序中圆的绘制方法，验证中点画圆生成算法，  提交实验报告：学号姓名-lab5-实验报告.docx | | | |
| **三、基本图元的生成算法程序设计（包括：程序代码、注释和运行结果截图等，将新增或改写的代码标红显示）**  // circle algorithm.cpp : Defines the entry point for the console application.  //  //This program was updated by Ms Huang Jing in Beijing Normal Unbersity Zhuhai Campus  // in Oct 20 in 2012,Contact email: 306744855@qq.com  #include "stdafx.h"  #include <glut.h>  //#include <stdio.h>  //#include <stdlib.h>  #include <math.h>  #define N 1000 //maximum line numbers  int cx = 150, cy = 150, radius = 80, ri = 50, ro = 80, n = 3, samples = 1, flag = 1;  int ww, hh; // for display window width and height  int line[N][4], k = 0; //for line's endpoint coordinates and line number  void Myinit(void);  void plotC(int x, int y, int xc, int yc);  void Bresenham\_Circle\_Algorithm(int cx, int cy, int radius);  void DrawOneLine(int x1, int y1, int x2, int y2);  void NSidedPolygon(int n, int cx, int cy, int radius);  void SampleCircle(int inner, int outer, int samples, int cx, int cy);  void Keyboard(unsigned char key, int x, int y);  void Display(void);  void Reshape(int w, int h);  void Drawlines();  void myMouse(int button, int state, int x, int y);  void myMotion(int x, int y);  void CirPot(int cx, int cy, int x, int y);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  char \*argv[] = { "hello ", " " };  int argc = 2; // must/should match the number of strings in argv  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);  glutCreateWindow("Basic\_Primitive\_Algorithm");  Myinit(); //初始设置  glutDisplayFunc(Display); //注册绘制响应回调函数  glutKeyboardFunc(Keyboard); //注册键盘响应回调函数  glutReshapeFunc(Reshape); //注册窗口改变回调函数  glutMouseFunc(myMouse); //注册鼠标按钮回调函数  glutMotionFunc(myMotion); //注册鼠标移动回调函数  glutMainLoop();  return 0;  }  void Myinit(void)  {  glClearColor(0.0, 0.0, 0.0, 0.0); //背景色  glLineWidth(4.0); //线宽  }  void Display(void)  {  glClear(GL\_COLOR\_BUFFER\_BIT);  glMatrixMode(GL\_MODELVIEW); //设置矩阵模式为模型变换模式，表示在世界坐标系下  glLoadIdentity(); //将当前矩阵设置为单位矩阵  if (flag == 1) //画圆  Bresenham\_Circle\_Algorithm(cx, cy, radius);  if (flag == 2) //画多边形  NSidedPolygon(n, cx, cy, radius);  if (flag == 3) //画胖圆  SampleCircle(ri, radius, samples, cx, cy);  if (flag == 4) //鼠标划线  Drawlines();  glutSwapBuffers();  }  void Keyboard(unsigned char key, int x, int y)  {  switch (key)  {  case 'r':  if ((flag == 1) && (radius>1)) radius--; //画单个圆时半径变小  if ((flag == 3) && (radius>ri)) radius--; //画胖圆时外径变小  break;  case 'R':  if ((radius<ww / 2) && (radius<hh / 2))  radius++; //半径增加  break;  case 'n':  if ((flag == 2) && (n>3)) n--; //多边形边数递减  break;  case 'N':  if (flag == 2) n++; //多边形边数递增  break;  case 'i':  if ((flag == 3) && (ri>1)) ri--; //画胖圆时内径变小  break;  case 'I':  if ((flag == 3) && (ri<radius)) ri++; //画胖圆时内径变大  break;  case 's':  if (samples>1) samples--; //采样数减少  break;  case 'S':  samples++; //采样数增加  break;  case 'a':  //cancel antialias  glDisable(GL\_BLEND);  glDisable(GL\_LINE\_SMOOTH);  glDisable(GL\_POINT\_SMOOTH);  glDisable(GL\_POLYGON\_SMOOTH);  break;  case 'A':  //enable antialias  glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);  glEnable(GL\_BLEND);  glEnable(GL\_POINT\_SMOOTH);  glHint(GL\_POINT\_SMOOTH\_HINT, GL\_NICEST);  glEnable(GL\_LINE\_SMOOTH);  glHint(GL\_LINE\_SMOOTH\_HINT, GL\_NICEST);  glEnable(GL\_POLYGON\_SMOOTH);  break;  case '1':  flag = 1; //画圆  break;  case '2':  flag = 2; //画多边形  break;  case '3':  flag = 3; //画胖圆  //cancel antialias  glDisable(GL\_BLEND);  glDisable(GL\_LINE\_SMOOTH);  glDisable(GL\_POINT\_SMOOTH);  glDisable(GL\_POLYGON\_SMOOTH);  break;  case '4':  flag = 4; //鼠标划线  for (int i = 0; i <= k; i++)  {  line[i][0] = 0;  line[i][1] = 0;  line[i][2] = 0;  line[i][3] = 0;  }  k = 0;  break;  case 27:  exit(0);  // default:  // return;  }  glutPostRedisplay();  }  void Reshape(int w, int h)  {  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  glViewport(0, 0, w, h);  gluOrtho2D(0, w, 0, h);  ww = w;  hh = h;  cx = w / 2;  cy = h / 2;  }  void Bresenham\_Circle\_Algorithm(int cx, int cy, int radius)  {  /\* YOUR CODE HERE ,  Update the followling codes with Bresenham\_Circle\_Algorithm \*/  glColor3f(1, 1, 1);  /\*glTranslatef(cx, cy, 0);  glutWireSphere(radius, 40, 40);\*/  int x, y, h;  x = 0; y = int(radius); h = 1 - int(radius);  CirPot(cx, cy, x, y);  while (x<y)  {  if (h<0) h += 2 \* x + 3;  else { h += 2 \* (x - y) + 5; y--; }  x++;  CirPot(cx, cy, x, y);  }  }  void CirPot(int cx, int cy, int x, int y)  {  //根据圆的对称性绘出8个对称点  glBegin(GL\_POINTS);  glVertex2f(cx + x, cy + y); //用color颜色在cx+x,y+y处绘制一点  glVertex2f(cx + x, cy - y);  glVertex2f(cx - x, cy + y);  glVertex2f(cx - x, cy - y);  glVertex2f(cx + y, cy + x);  glVertex2f(cx + y, cy - x);  glVertex2f(cx - y, cy + x);  glVertex2f(cx - y, cy - x);  glEnd();  }  void NSidedPolygon(int n, int cx, int cy, int radius)  {  int x[100], y[100];  double angle;  angle = 2 \* 3.1415926 / n;  glColor3f(1, 1, 1);  for (int i = 0; i <= n; i++)  {  x[i] = (int)(cx + radius\*cos(i\*angle));  y[i] = (int)(cy + radius\*sin(i\*angle));  }  for (int i = 0; i <= (n - 1); i++)  {  DrawOneLine(x[i], y[i], x[i + 1], y[i + 1]);  }  }  void DrawOneLine(int xa, int ya, int xb, int yb)  {  /\* YOUR CODE HERE \*/  //update the following codes with DDA line algorithm  glColor3f(1, 1, 1);  /\*glBegin(GL\_LINES);  glVertex2f(xa, ya);  glVertex2f(xb, yb);  glEnd();\*/  int dx = xb-xa, dy = yb-ya, k;  float xIncrement, yIncrement, steps, x = xa, y = ya;  if (abs(dx) > abs(dy)) steps = abs(dx);  else steps = abs(dy);  xIncrement = (float)(dx) / steps;  yIncrement = (float)(dy) / steps;  for (k = 0; k<steps; k++)  {  //用color颜色在round(x), round(y)处绘制一点  //Putpixel(round(x), round(y), color);  glBegin(GL\_POINTS);  glVertex2f((int)(x+0.5), (int)(y+0.5));  glEnd();  x += xIncrement; y += yIncrement;  }  }  void Drawlines()  {  for (int i = 0; i <= k; i++) //\*\*\*\*\*\*\*\*  {  DrawOneLine(line[i][0], line[i][1], line[i][2], line[i][3]);  }  }  void SampleCircle(int inner, int outer, int samples, int cx, int cy)  {  float r2, ro2, ri2; //r2为子像素的半径  int x, y, nx, ny; //x,y 为胖圆内的一点  float color\_value; //颜色亮度  int count; //计数合格的子像素  //######### Calculate outer radius's square and inner's square  ro2 = outer\*outer; //胖圆外径平方  ri2 = inner\*inner; //胖圆内径平方  //######### for each pixel(x,y), testing its sample  //1/4胖圆内的点测试  for (x = 0; x<outer; x++)  for (y = 0; y <= outer; y++)  {  //######## count available subpixels for each pixel  count = 0;  //每个点切分成smaples\*samples个子像素  //判断子像素是否在胖圆内  for (nx = 0; nx <= samples; nx++)  for (ny = 0; ny <= samples; ny++)  {  r2 = (x + (float)nx / (float)samples)\*(x + (float)nx / (float)samples);  r2 = r2 + (y + (float)ny / (float)samples)\*(y + (float)ny / (float)samples);  if ((r2 <= ro2) && (r2 >= ri2)) count++; //计数合格的子像素  }  //########## calculate color gray level for each pixel  color\_value = ((float)count) / (((float)samples + 1.0)\*((float)samples + 1.0));  //########## draw pixel by using the color calculated above  glColor3f(color\_value, color\_value, color\_value);  //对称画点  glBegin(GL\_POINTS);  glVertex2f(cx + x, cy + y);  glVertex2f(cx + x, cy - y);  glVertex2f(cx - x, cy + y);  glVertex2f(cx - x, cy - y);  glEnd();  }  }  //鼠标按钮响应事件..  void myMouse(int button, int state, int x, int y)  {  if (button == GLUT\_LEFT\_BUTTON&&state == GLUT\_DOWN)  {  line[k][0] = x; //线段起点x坐标  line[k][1] = hh - y; //线段起点y坐标  }  if (button == GLUT\_LEFT\_BUTTON&&state == GLUT\_UP)  {  line[k][2] = x; //线段终点x坐标  line[k][3] = hh - y; //线段终点y坐标  k++;  glutPostRedisplay();  }  }  //鼠标移动时获得鼠标移动中的坐标-----------------------------------------------------  void myMotion(int x, int y)  {  //get the line's motion point  line[k][2] = x; //动态终点的x坐标  line[k][3] = hh - y; //动态终点的y坐标  glutPostRedisplay();  }  0102 | | | |
| **四、实验总结：**  了解，明白 | | | |