**实验二 基本图元光栅化**

1. 实验目的

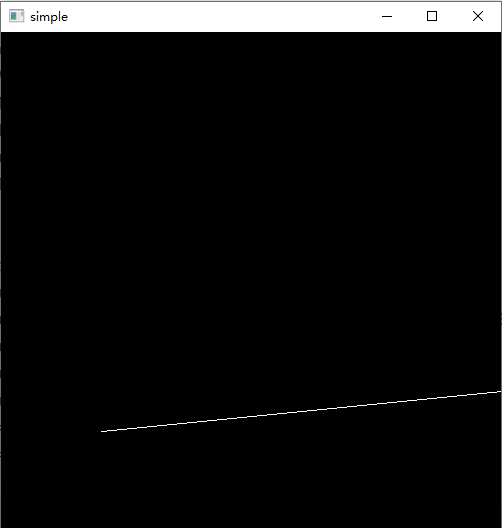
（DDA算法、Bresenham算法、中点画线算法）的实现原理，理解圆的扫描转换原理。理解基本图元光栅化（扫描转换）的概念，掌握直线光栅化的三种基本算法

1. 实验内容

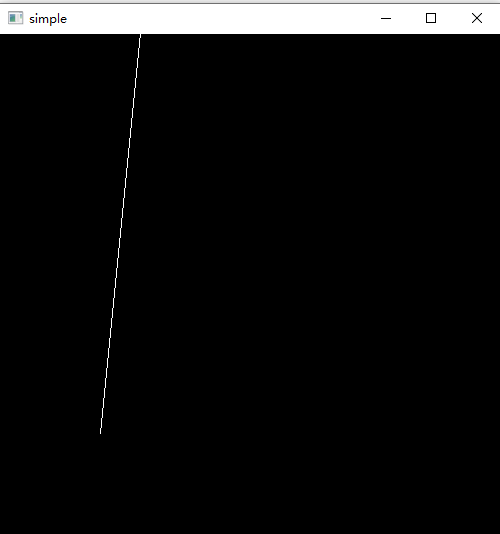
请基于实验一发布的例程Example1程序框架，完成以下实验内容。

1. 【必做】设计函数，实现任意斜率的Bresenham直线绘制算法，并给出不同方向区间直线段的绘制结果，将输出结果及问题分析写入实验报告。

直线斜率大于0小于1：



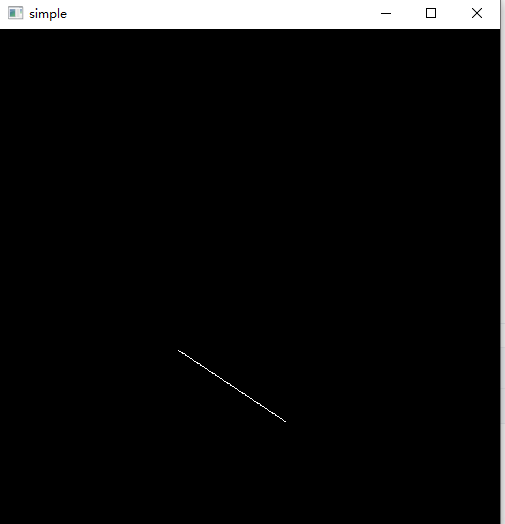
直线斜率大于1：



直线斜率小于-1：



直线斜率大于-1小于0：

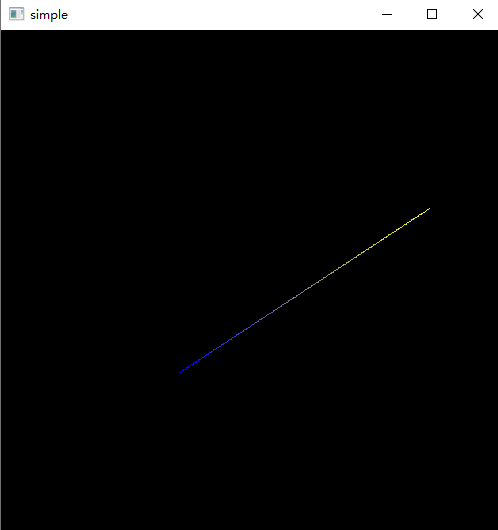


#include <GL/glut.h>  
#include <cmath>  
  
int a, b, c, d;  
  
*//Bresenham*void drawLine(int x1, int y1, int x2, int y2) {  
 int dx = abs(x2 - x1);  
 int dy = abs(y2 - y1);  
 int sx, sy;  
  
 if (x1 < x2) sx = 1;  
 else sx = -1;  
  
 if (y1 < y2) sy = 1;  
 else sy = -1;  
  
 int err = dx - dy;  
  
 while (1) {  
 glVertex2i(x1, y1);  
 if (x1 == x2 && y1 == y2)  
 break;  
 int e2 = 2 \* err;  
 if (e2 > -dy) {  
 err -= dy;  
 x1 += sx;  
 }  
 if (e2 < dx) {  
 err += dx;  
 y1 += sy;  
 }  
 }  
}  
  
void display() {  
 glClear(GL\_COLOR\_BUFFER\_BIT);  
 glBegin(GL\_POINTS);  
 drawLine(a, b, c, d);  
 glEnd();  
 glFlush();  
}  
  
void reshape(int w, int h) {  
 glViewport(0, 0, w, h);  
 glMatrixMode(GL\_PROJECTION);  
 glLoadIdentity();  
 gluOrtho2D(-200, w, -200, h);  
 glMatrixMode(GL\_MODELVIEW);  
}  
  
int main(int argc, char \*\*argv) {  
 **glutInit**(&argc, argv);  
 glutInitDisplayMode(**GLUT\_SINGLE** | **GLUT\_RGB**);  
 glutInitWindowSize(500, 500);  
 glutInitWindowPosition(0, 0);  
 **glutCreateWindow**("simple");  
 a = 50;  
 b = 50;  
 c = 200;  
 d = -50;  
 glutDisplayFunc(display);  
 glutReshapeFunc(reshape);  
 glutMainLoop();  
 return 0;  
}

1. 【选做】改写题目1中任意斜率Bresenham直线段绘制函数，输入直线的起点颜色值、终点颜色值，利用线性插值公式2-1(该公式针对斜率大于0且小于1的直线)，实现渐变色直线的绘制，将实验结果及问题分析写入实验报告。

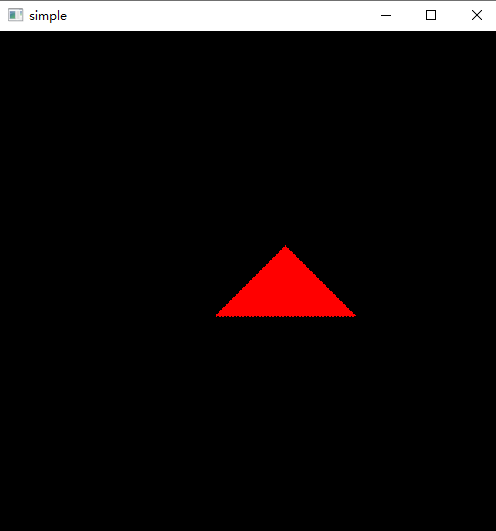
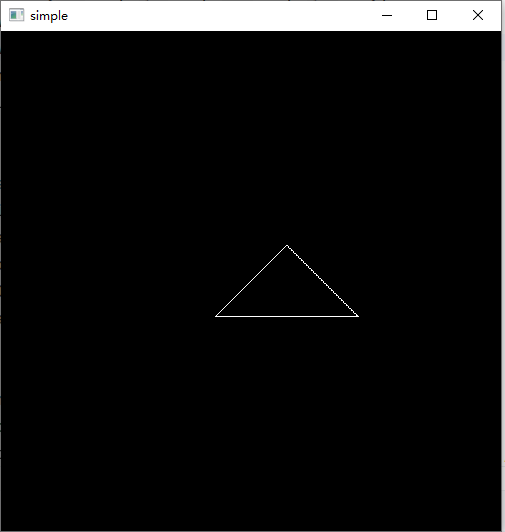
 （公式2-1）

其中公式2-1中，x1为起点x坐标，x2为终点x坐标，c1为起点颜色，c2为终点颜色，C为直线段上任意一点（坐标为x）的颜色。（注意：颜色c1、c2以及C都是通过(r,g,b)三个颜色分量组合来表示的，所以上述公式应该分别应用到(r,g,b)三个分量以计算某点颜色，最后将计算出的颜色通过函数glColor3f设置称为当前颜色。）



#include <GL/glut.h>  
#include <cmath>  
  
int a, b, c, d;  
float c1[] = {0.0f, 0.0f, 1.0f};  
float c2[] = {1.0f, 1.0f, 0.0f};  
  
*//Bresenham*void drawLine(int x1, int y1, int x2, int y2) {  
 int dx = abs(x2 - x1);  
 int dy = abs(y2 - y1);  
 int sx, sy;  
  
 if (x1 < x2) sx = 1;  
 else sx = -1;  
  
 if (y1 < y2) sy = 1;  
 else sy = -1;  
  
 int err = dx - dy;  
  
 int tmp1 = x1;  
 int tmp2 = y1;  
 float color[3];  
  
 while (1) {  
 for(int i = 0;i < 3;i++) {  
 color[i] = (x1-x2)\*c1[i]/(tmp1-x2) + (x1-tmp1)\*c2[i]/(x2-tmp1);  
 }  
 glColor3f(color[0], color[1], color[2]);  
 glVertex2i(x1, y1);  
 if (x1 == x2 && y1 == y2)  
 break;  
 int e2 = 2 \* err;  
 if (e2 > -dy) {  
 err -= dy;  
 x1 += sx;  
 }  
 if (e2 < dx) {  
 err += dx;  
 y1 += sy;  
 }  
 }  
}  
  
void display() {  
 glClear(GL\_COLOR\_BUFFER\_BIT);  
 glBegin(GL\_POINTS);  
 drawLine(50, 20, 400, 250);  
 glEnd();  
 glFlush();  
}  
  
void reshape(int w, int h) {  
 glViewport(0, 0, w, h);  
 glMatrixMode(GL\_PROJECTION);  
 glLoadIdentity();  
 gluOrtho2D(-200, w, -200, h);  
 glMatrixMode(GL\_MODELVIEW);  
}  
  
int main(int argc, char \*\*argv) {  
 **glutInit**(&argc, argv);  
 glutInitDisplayMode(**GLUT\_SINGLE** | **GLUT\_RGB**);  
 glutInitWindowSize(500, 500);  
 glutInitWindowPosition(0, 0);  
 **glutCreateWindow**("simple");  
 glutDisplayFunc(display);  
 glutReshapeFunc(reshape);  
 glutMainLoop();  
 return 0;  
}

1. 【必做】利用题1中的Bresenham直线绘制函数绘制三角形的三条边，并将该三角形内部像素设置为某种颜色（内部颜色可以和边颜色相同）。将实验结果及问题分析写入实验报告。



#include <GL/glut.h>  
#include <cmath>  
  
int a, b, c, d;  
  
void drawLine(int x1, int y1, int x2, int y2) {  
 int dx = abs(x2 - x1);  
 int dy = abs(y2 - y1);  
 int sx, sy;  
  
 if (x1 < x2) sx = 1;  
 else sx = -1;  
  
 if (y1 < y2) sy = 1;  
 else sy = -1;  
  
 int err = dx - dy;  
  
 while (1) {  
 glVertex2i(x1, y1);  
 if (x1 == x2 && y1 == y2)  
 break;  
 int e2 = 2 \* err;  
 if (e2 > -dy) {  
 err -= dy;  
 x1 += sx;  
 }  
 if (e2 < dx) {  
 err += dx;  
 y1 += sy;  
 }  
 }  
}  
  
void display() {  
 glClear(GL\_COLOR\_BUFFER\_BIT);  
  
 *// 画三角形并填充颜色* glBegin(GL\_TRIANGLES);  
 glColor3f(1.0f, 0.0f, 0.0f); *// 设置红色* glVertex2i(100, 100);  
 glVertex2i(200, 200);  
 glVertex2i(300, 100);  
 glEnd();  
  
 glBegin(GL\_LINES); *// 画线框* glColor3f(0.0f, 0.0f, 0.0f); *// 设置黑色* drawLine(100, 100, 200, 200);  
 drawLine(200, 200, 300, 100);  
 drawLine(100, 100, 300, 100);  
 glEnd();  
  
 glFlush();  
}  
  
void reshape(int w, int h) {  
 glViewport(0, 0, w, h);  
 glMatrixMode(GL\_PROJECTION);  
 glLoadIdentity();  
 gluOrtho2D(-200, w, -200, h);  
 glMatrixMode(GL\_MODELVIEW);  
}  
  
int main(int argc, char \*\*argv) {  
 **glutInit**(&argc, argv);  
 glutInitDisplayMode(**GLUT\_SINGLE** | **GLUT\_RGB**);  
 glutInitWindowSize(500, 500);  
 glutInitWindowPosition(0, 0);  
 **glutCreateWindow**("simple");  
 glutDisplayFunc(display);  
 glutReshapeFunc(reshape);  
 glutMainLoop();  
 return 0;  
}