实验一　椭圆的扫描转换算法的实现

一．实验目的

1、学习椭圆的中点 Bresenham 算法。

2、实现交互式绘制椭圆。

二．实验工具与设备

Clion

三、实验内容

1、在本科学习平台（s.ecust.edu.cn）资料栏下，下载椭圆的中点 Bresenham 算法代码文件 onMidPointEllispe.cpp；

2、参考教材 3.4 节和 3.6 节内容，添加“椭圆”菜单项，通过橡皮筋技术，交互确定矩形对角线位置，根据矩形宽、高，确定椭圆的长、短轴长度。

1） 如果是橡皮筋时，显示矩形和内切于矩形的椭圆

2） 如果是橡皮筋结束，仅显示椭圆

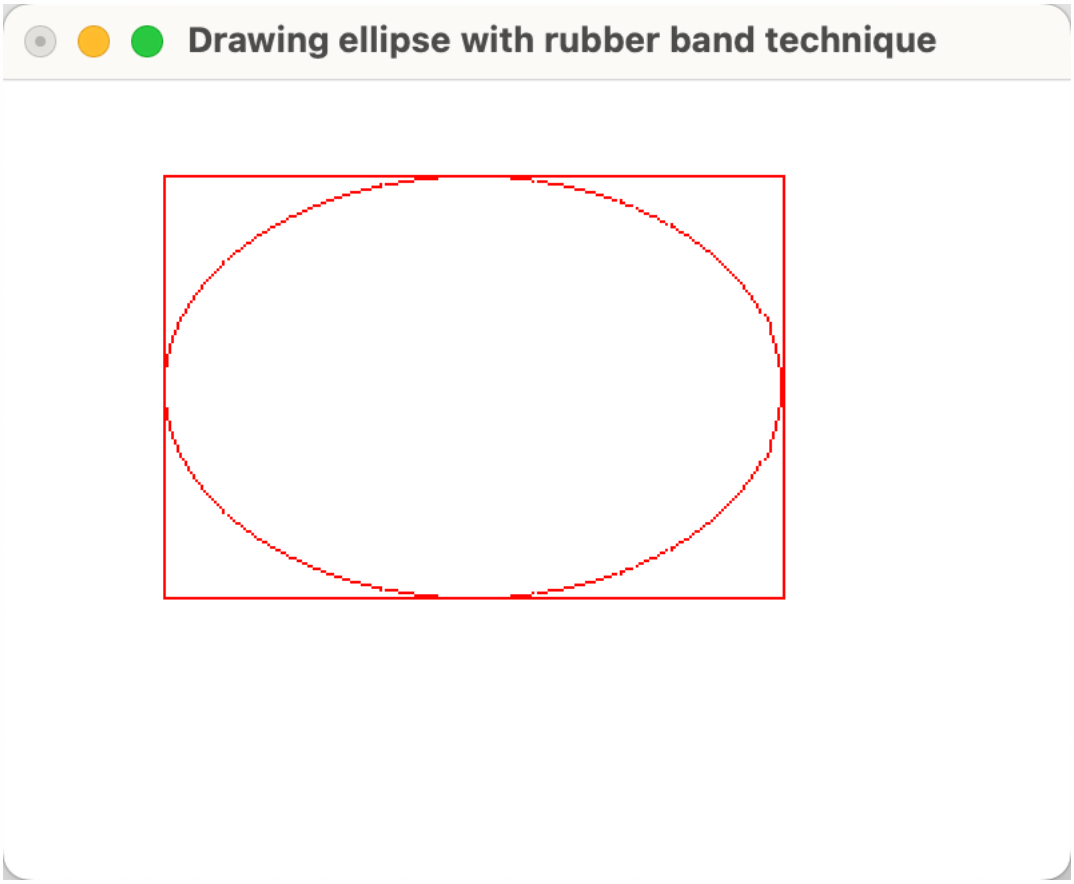
3、由于椭圆的中点 Bresenham 算法要求椭圆的圆心须在坐标系原点上，可以利用 glTranslated (GLdouble x, GLdouble y, GLdouble z);将世界坐标原点平移到矩形中心，椭圆绘制结束，再反平移，恢复世界坐标原点的原来位置。

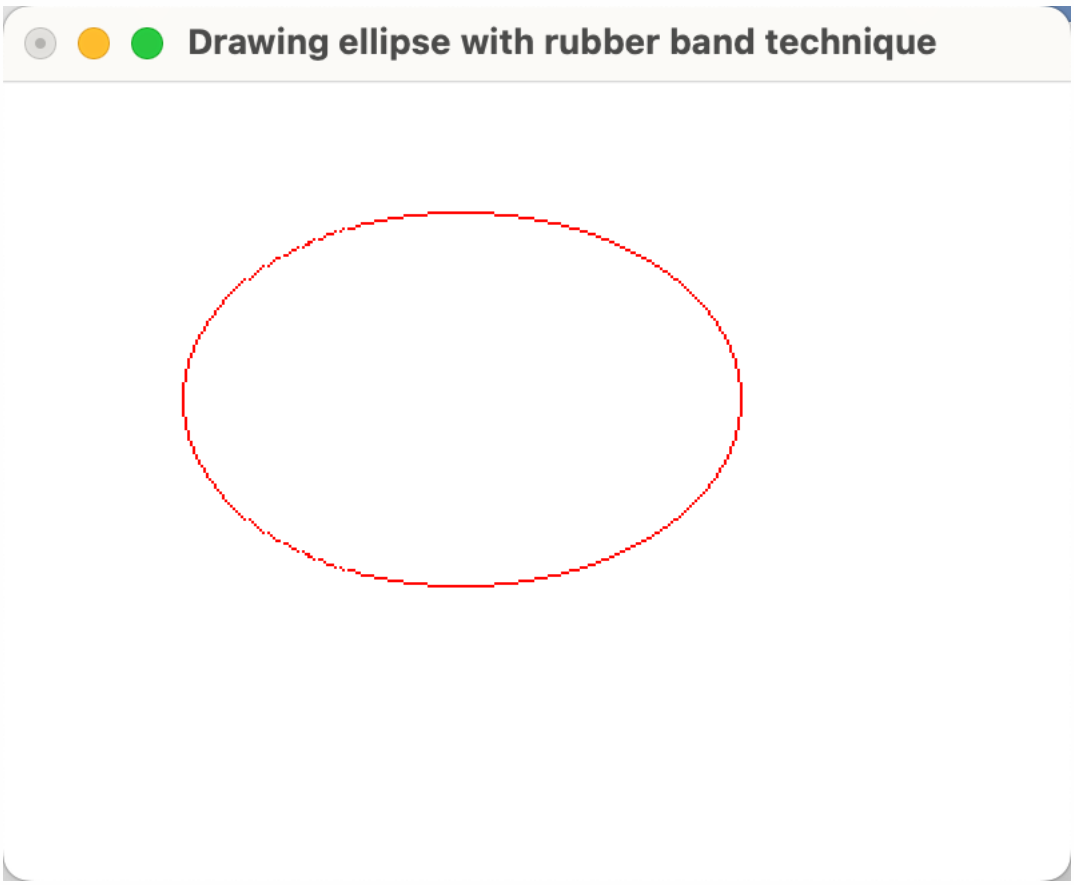
```
#include <iostream>
#include <GLUT/glut.h>//在 windows 系统上运行请修改 glut 存储路径

using namespace std;
int iPointNum = 0;
int x1=0,x2=0,y1=0,y2=0;
int pos_x = 0,pos_y = 0;
int winWidth = 400, winHeight = 300;
void Initial(void){
    glClearColor(1.0f,1.0f,1.0f,1.0f);
}
void ChangeSize(int w,int h){
    winWidth = w;
    winHeight = h;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
```

```
    glLoadIdentity();
    gluOrtho2D(0.0,winWidth,0.0,winHeight);
}
void onMidPointEllispe(){
    int x,y;
    float d1,d2;
    int a = abs(0.5*(x2-x1));
    int b = abs(0.5*(y2-y1));
    pos_x = 0.5*(x2+x1);
    pos_y = 0.5*(y2+y1);
    glTranslated(0.5*(x2+x1),0.5*(y2+y1),0);
    glBegin(GL_POINTS);
    x=0;y=b;//起始点坐标
    d1=b*b+a*a*(-b+0.25);
    glVertex2i(x,y);
    glVertex2i(-x,-y);
    glVertex2i(-x,y);
    glVertex2i(x,-y);
    while(b*b*(x+1)<a*a*(y-0.5))
    {
        if(d1<=0){
            d1+=b*b*(2*x+3);
            x++;
        }
        else{
            d1+=b*b*(2*x+3)+a*a*(-2*y+2);
            x++; y--;
        }
        glVertex2f(x,y);
        glVertex2f(-x,-y);
        glVertex2f(-x,y);
        glVertex2f(x,-y);
    }/*while上半部分*/
    d2=b*b*(x+0.5)*(x+0.5)+a*a*(y-1)*(y-1)-a*a*b*b;
    while(y>0)
    {
        if(d2<=0){
            d2+=b*b*(2*x+2)+a*a*(-2*y+3);
            x++; y--;
        }
        else{
            d2+=a*a*(-2*y+3);
            y--;
        }
```

```
            glVertex2f(x,y);
            glVertex2f(-x,-y);
            glVertex2f(-x,y);
            glVertex2f(x,-y);
        }

        glEnd();
        glTranslated(-0.5*(x2+x1),-0.5*(y2+y1),0);
    }
    void Display(void){
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0f,0.0f,0.0f);
        if(iPointNum >=1){
            onMidPointEllispe();
        }
        if(iPointNum == 1){
            glBegin(GL_LINE_STRIP);//start to draw something
            glVertex2i(x1, y1);
            glVertex2i(x2, y1);
            glVertex2i(x2, y2);
            glVertex2i(x1, y2);
            glVertex2i(x1, y1);
            glEnd();
        }

        glutSwapBuffers();
    }
    void MousePlot(GLint button, GLint action, GLint xMouse,GLint yMouse){
        if(button == GLUT_LEFT_BUTTON && action == GLUT_DOWN){
            if(iPointNum == 0 || iPointNum == 2){
                iPointNum = 1;
                x1 = xMouse;
                y1 = winHeight - yMouse;
            }
            else(iPointNum = 2);
            x2 = xMouse;
            y2 = winHeight - yMouse;
            glutPostRedisplay();
        }
        if(button == GLUT_RIGHT_BUTTON && action == GLUT_DOWN){
            iPointNum = 0;
            glutPostRedisplay();
        }
    }
```

```
void PassiveMouseMove(GLint xMouse,GLint yMouse){
    if(iPointNum == 1){
        x2 = xMouse;
        y2 = winHeight - yMouse;
        glutPostRedisplay();
    }
}
void ProcessMenu(int value){
    if(value == 1){
        glutMouseFunc(MousePlot);
        glutPassiveMotionFunc(PassiveMouseMove);
    }
}
void Key(unsigned char key, int i, int i1){
    if(key == 'r'){
        iPointNum = 0;
        glutPostRedisplay();
    }
}
int main(int argc,char *argv[]){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); //初始化窗口的显示模式
    glutInitWindowSize(400, 300);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("name");
    glutDisplayFunc(Display);
    glutReshapeFunc(ChangeSize);

    glutCreateMenu(ProcessMenu);
    glutAddMenuEntry("Ellipse",1);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutKeyboardFunc(Key);//绘制完成后按下 r 键可以清除
    Initial();
    glutMainLoop();
}
```


Drawing ellipse with rubber band technique


Drawing ellipse with rubber band technique

四、拓展实验

利用 glutGetModifiers()函数，通过组合 SHIFT 键，控制正方形边长第 2 个位置，根据正方形长度确定正圆直径，实现正圆绘制。

int glutGetModifiers(void);

这个函数仅仅只能在处理按键消息或者鼠标消息函数里被调用，通过函数的返回值检测是否有组合键被按下。

这里函数的返回值是三个 glut.h 里预定义的常量里的一个，或它们的或组合。这三个常量是：

1）GLUT_ACTIVE_SHIFT: 当按下 SHIFT 键或以按下 CAPS LOCK，注意两者同时按下时，不会返回这个值。

2）GLUT_ACTIVE_CTRL: 返回它，当按下 CTRL 键。

3）GLUT_ACTIVE_ATL：返回它,当按下 ATL 键。

例如：GLUT_ACTIVE_CTRL|GLUT_ACTIVE_ALT：按下组合键 CTRL+ALT

按 shift+c 可以画正圆

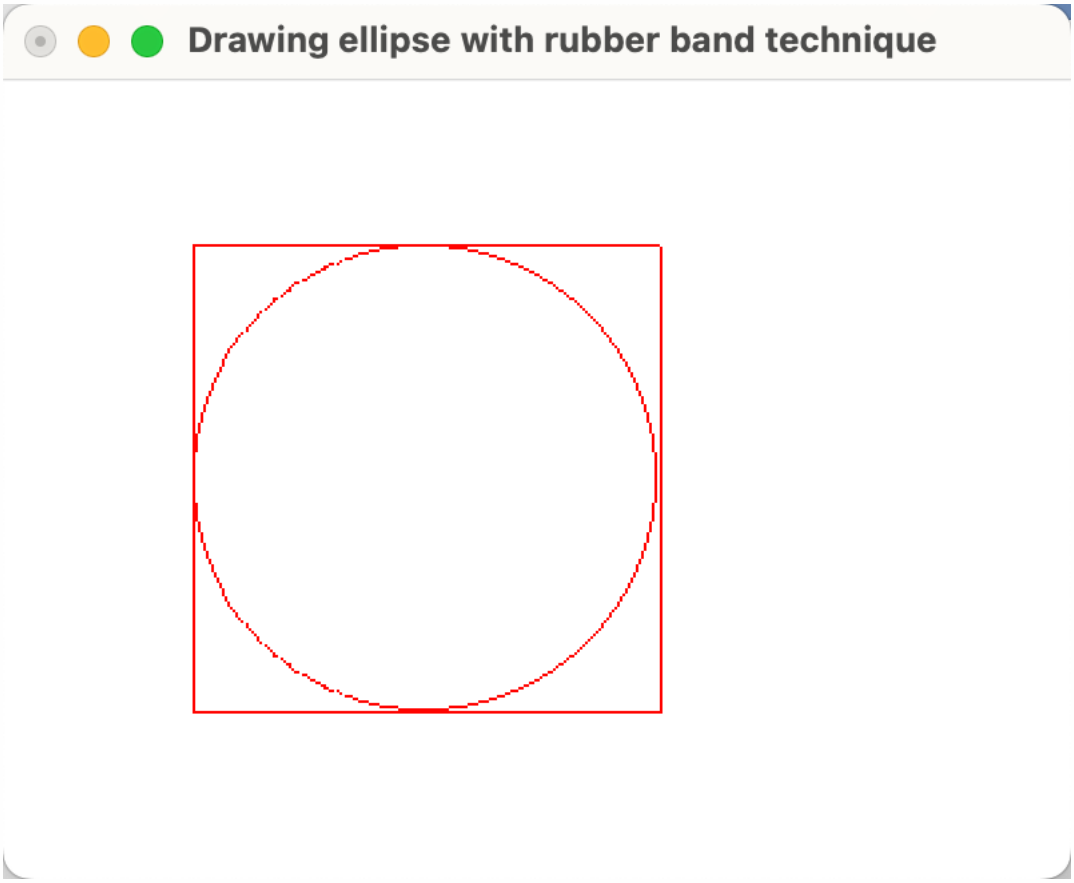
```
#include <iostream>
#include <GLUT/glut.h>//在 windows 系统上运行请修改 glut 存储路径

//
//author
//This code is finished with Macos whose document code is 'UTF-8',
//so if you find any grabled code when you open it, please change the document
code from 'GB2312' or else to 'UTF-8'.
//

using namespace std;
int iPointNum = 0;
int a0=0,a1=0;
int x1=0,x2=0,y1=0,y2=0;
int pos_x = 0,pos_y = 0;
int judge;
int winWidth = 400, winHeight = 300;
void Initial(void){
    glClearColor(1.0f,1.0f,1.0f,1.0f);
}
void ChangeSize(int w,int h){
    winWidth = w;
    winHeight = h;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,winWidth,0.0,winHeight);
}
void onMidPointEllispe(int a,int b){
```

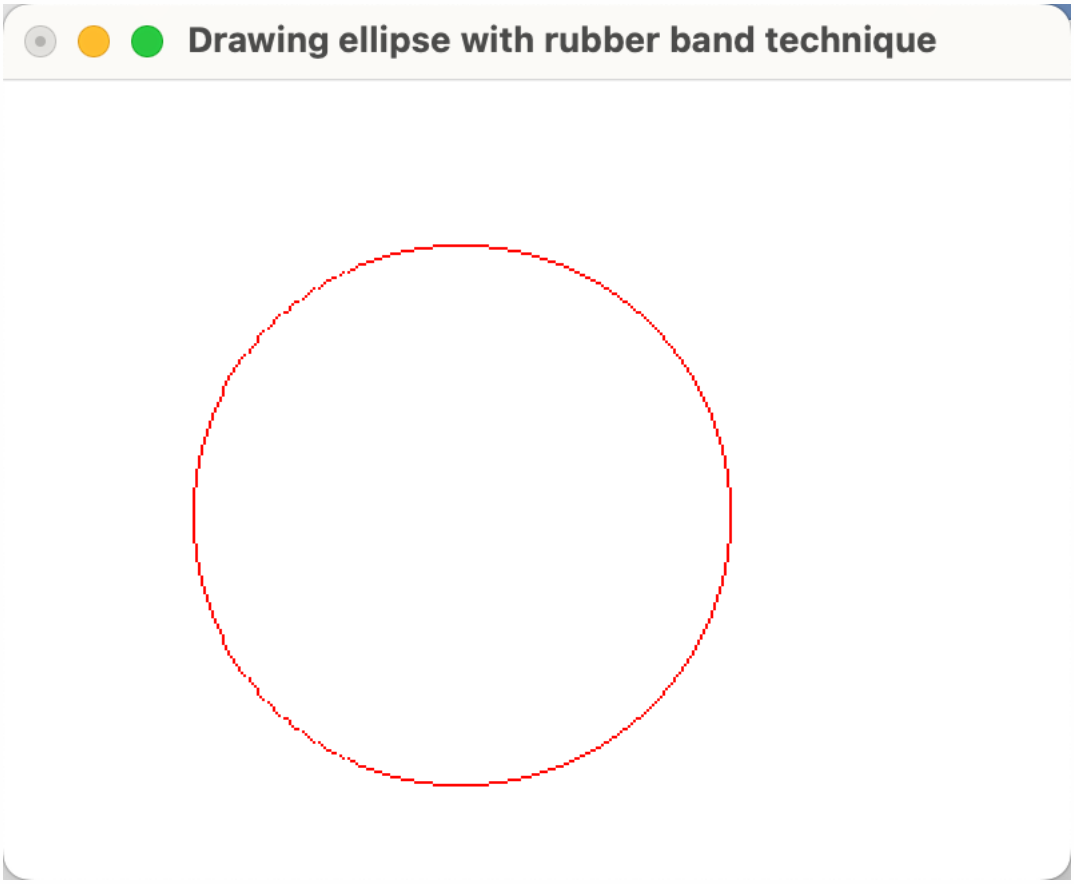
```
int x,y;
float d1,d2;
pos_x = 0.5*(x2+x1);
pos_y = 0.5*(y2+y1);
if(judge == 0)glTranslated(pos_x,pos_y,0);
else if(judge == 1) glTranslated(pos_x,y1-b,0);
glBegin(GL_POINTS);
x=0;y=b;//起始点坐标
d1=b*b+a*a*(-b+0.25);
glVertex2i(x,y);
glVertex2i(-x,-y);
glVertex2i(-x,y);
glVertex2i(x,-y);
while(b*b*(x+1)<a*a*(y-0.5))
{
    if(d1<=0){
        d1+=b*b*(2*x+3);
        x++;
    }
    else{
        d1+=b*b*(2*x+3)+a*a*(-2*y+2);
        x++; y--;
    }
    glVertex2f(x,y);
    glVertex2f(-x,-y);
    glVertex2f(-x,y);
    glVertex2f(x,-y);
}/*while上半部分*/
d2=b*b*(x+0.5)*(x+0.5)+a*a*(y-1)*(y-1)-a*a*b*b;
while(y>0)
{
    if(d2<=0){
        d2+=b*b*(2*x+2)+a*a*(-2*y+3);
        x++; y--;
    }
    else{
        d2+=a*a*(-2*y+3);
        y--;
    }
    glVertex2f(x,y);
    glVertex2f(-x,-y);
    glVertex2f(-x,y);
    glVertex2f(x,-y);
}
```

```
    glEnd();
    if(judge == 0)glTranslated(-pos_x,-pos_y,0);
    else if(judge == 1) glTranslated(-pos_x,-(y1-b),0);
}
void Display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f,0.0f,0.0f);
    if(judge == 0) {
        if (iPointNum >= 1) {
            onMidPointEllispe(abs(0.5 * (x2 - x1)), abs(0.5 * (y2 - y1)));
        }
        if (iPointNum == 1) {
            glBegin(GL_LINE_STRIP);//start to draw something
            glVertex2i(x1, y1);
            glVertex2i(x2, y1);
            glVertex2i(x2, y2);
            glVertex2i(x1, y2);
            glVertex2i(x1, y1);
            glEnd();
        }
    }
    else{
        if (iPointNum >= 1) {
            onMidPointEllispe(abs(0.5 * (x2 - x1)), abs(0.5 * (x2 - x1)));
        }
        if (iPointNum == 1) {
            glBegin(GL_LINE_STRIP);//start to draw something
            glVertex2i(x1, y1);
            glVertex2i(x2, y1);
            glVertex2i(x2, y1-abs((x2 - x1)));
            glVertex2i(x1, y1-abs((x2 - x1)));
            glVertex2i(x1, y1);
            glEnd();
        }
    }

    glutSwapBuffers();
}
void MousePlot(GLint button, GLint action, GLint xMouse,GLint yMouse){
    if(button == GLUT_LEFT_BUTTON && action == GLUT_DOWN){
        if(iPointNum == 0 || iPointNum == 2){
            iPointNum = 1;
            x1 = xMouse;
            y1 = winHeight - yMouse;
```

```
        }
        else(iPointNum = 2);
        x2 = xMouse;
        y2 = winHeight - yMouse;
        glutPostRedisplay();
    }

}
void PassiveMouseMove(GLint xMouse,GLint yMouse){
    if(iPointNum == 1){
        x2 = xMouse;
        y2 = winHeight - yMouse;
        glutPostRedisplay();
    }
}
void ProcessMenu(int value){
    if(value == 1){
        glutMouseFunc(MousePlot);
        glutPassiveMotionFunc(PassiveMouseMove);
    }
}
void Key(unsigned char key, int i, int i1){
    if(key == 'r'){
        iPointNum = 0;
        judge = 0;
        glutPostRedisplay();
    }
    int mod = glutGetModifiers();
    if (mod == GLUT_ACTIVE_SHIFT and key == 'C') {
        judge = 1;
    }
}

int main(int argc,char *argv[]){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); //初始化窗口的显示模式
    glutInitWindowSize(400, 300);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Drawing ellipse with rubber band technique");
    glutDisplayFunc(Display);
    glutReshapeFunc(ChangeSize);

    glutCreateMenu(ProcessMenu);
    glutAddMenuEntry("Ellipse",1);
```

```
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutKeyboardFunc(Key);//按 shift+c 画正圆，绘制完成后按下 r 键可以清除和重置
    Initial();
    glutMainLoop();
}
```





五．思考题

1．什么是“扫描转换”？一般什么时候需要扫描转换？

扫描转换就是将图形对象表示为像素集合的过程。图形对象是连续的。使用的像素是离散的。每个像素可以具有开或关状态。

在将函数图像绘制在屏幕时需要用到扫描转换。