《计算机图形学》实验报告

《裁剪算法实验》

姓名 王红阳

学号 3019244233

专业 计算机科学与技术

班级 3班

天津大学智能与计算学部

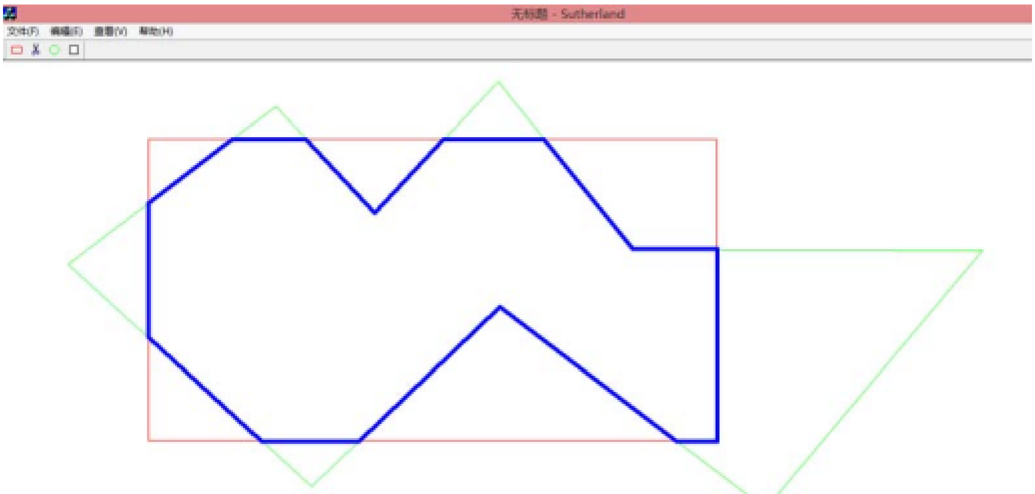
2021年 09月27 日

一、实验目的

- 用编程语言实现如何裁剪直线和裁剪多边形

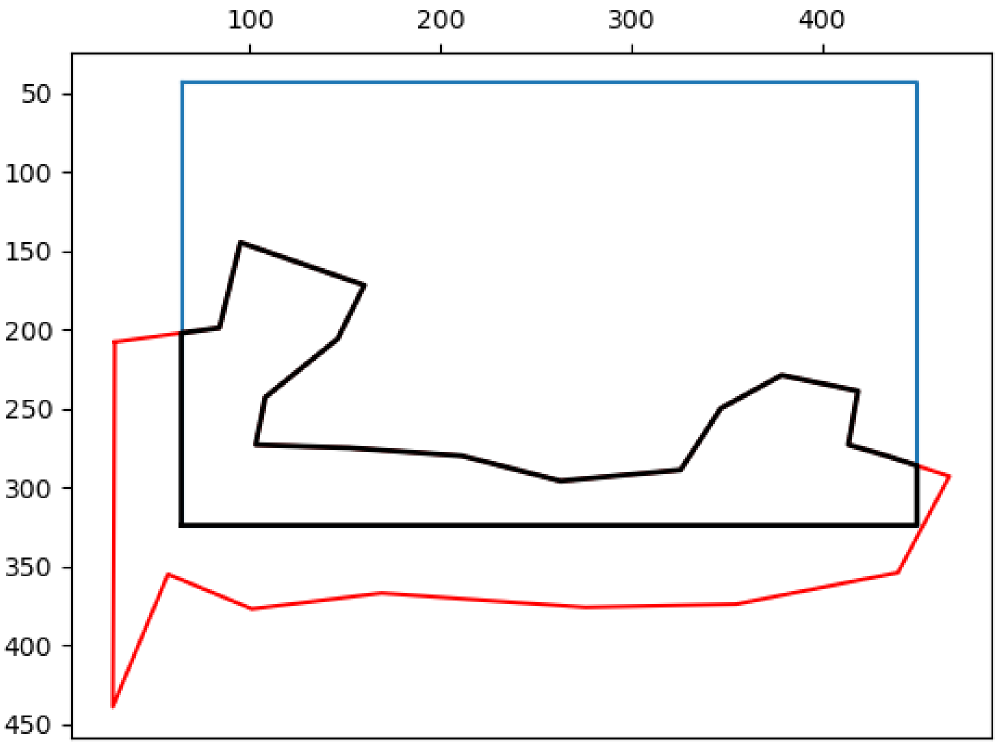
二、实验内容

- 内容:
  - 实现 Cohen-Sutherland 直线裁剪算法（选做）
  - 实现 Sutherland-Hodgman 多边形裁剪算法
- 要求:
  - 自定义裁剪窗口和待裁剪直线段(或多边形);
  - 采用不同颜色突出显示裁剪结果，如下图所示



三、实验结果

使用Sutherland_Hodgman算法对多边形进行裁剪:



四、实验分析和总结

上次实验，我使用python中的matplotlib实现了画线，但是matplotlib并不能很好地画出多边形，经过考察，我决定使用python中的opencv模块进行实现。

使用流程：使用鼠标左键画矩形裁剪框，鼠标中键标注多边形顶点，鼠标右键勾画多边形，之后按ESC退出，然后就会出现裁剪后的图形了

代码流程：使用opencv工具自定义矩形裁剪框和多边形待裁剪图形 ，然后将每一线段的一对顶点送入一组裁剪器(左、右、下、上）一个裁剪器完成一对顶点的处理后，该边裁剪后留下的坐标值送给下一个裁剪器。最终将裁剪完成的图形，使用matplotlib进行显示。

通过本次实验，我更加深入地了解了如何使用python画出图形，如何使用opencv模块进行鼠标事件的监听和使用matplotlib模块画图，为后续进一步深入实现计算机图形学领域的经典算法奠定了基础。

五、源代码

Sutherland_Hodgman.py:

```
import matplotlib.pyplot as plt
import numpy as np
import cv2

drawing = False  # 鼠标按下为真
notdone = True
ix, iy = -1, -1  # 左下角
```

```python
px, py = -1, -1  # 右上角
l = []  # 多边形顶点的列表


def pointInRec(p):
    """判断点P是否在区域内
    """
    if ix <= p[0] <= px and iy <= p[1] <= py:
        return True
    return False


def draw_rectangle(event, x, y, flags, param):
    """响应鼠标事件，画矩形
    """
    global ix, iy, drawing, px, py, l, notdone

    if event == cv2.EVENT_LBUTTONDOWN and notdone == True:  # 鼠标左键画矩形
        drawing = True
        ix, iy = x, y
    elif event == cv2.EVENT_MOUSEMOVE and notdone == True:
        if drawing == True:
            cv2.rectangle(img, (ix, iy), (px, py), (0, 0, 0), 0)  # 将刚刚拖拽的矩形涂黑
            cv2.rectangle(img, (ix, iy), (x, y), (0, 255, 0), 0)
            px, py = x, y
    elif event == cv2.EVENT_LBUTTONUP and notdone == True:
        drawing = False
        cv2.rectangle(img, (ix, iy), (x, y), (0, 255, 0), 0)
        px, py = x, y
        notdone = False
    elif event == cv2.EVENT_MBUTTONDOWN:  # 鼠标中键标记多边形顶点
        print((x, y))
        l.append([x, y])
        cv2.circle(img, (x, y), 1, (255, 255, 255))
    elif event == cv2.EVENT_RBUTTONDOWN:  # 鼠标右键生成多边形
        pts = np.array(l, np.int32)
        pts = pts.reshape((-1, 1, 2))
        cv2.polylines(img, [pts], True, (255, 255, 255))


def line_intersection(line1, line2):
    """计算两条线的交点
    """
    xdiff = (line1[0][0] - line1[1][0], line2[0][0] - line2[1][0])
    ydiff = (line1[0][1] - line1[1][1], line2[0][1] - line2[1][1])

    def det(a, b):
        return a[0] * b[1] - a[1] * b[0]

    div = det(xdiff, ydiff)
    if div == 0:
        return 99999, 99999

    d = (det(*line1), det(*line2))
    x = det(d, xdiff) / div
    y = det(d, ydiff) / div
    return x, y


def fun(p1, p2):
    """求出两个点的delta y,delta  x, 叉积
    """
    x1 = p1[0]
    y1 = p1[1]
    x2 = p2[0]
    y2 = p2[1]
    a = y2 - y1
    b = x1 - x2
    c = x2 * y1 - x1 * y2
    return a, b, c


def clip_left(pointList):
    global ix, iy, px, py
    newList = []
    for i in range(len(pointList)):
        p1 = pointList[i - 1]
        p2 = pointList[i]
        if p1[0] < ix and p2[0] > ix:  # 由外到内
            a, b, c = fun(p1, p2)
            y = (-c - a * ix) / b
            intersection = [ix, y]
            newList.append(intersection)
            newList.append(p2)
        elif p1[0] > ix and p2[0] > ix:  #由内到内
            newList.append(p2)
        elif p1[0] > ix and p2[0] < ix:  #由内到外
            a, b, c = fun(p1, p2)
            y = (-c - a * ix) / b
            intersection = [ix, y]
            newList.append(intersection)
    return newList


def clip_bottom(pointList):
    pointList = clip_left(pointList)
    global ix, iy, px, py
    newList = []
    for i in range(len(pointList)):
        p1 = pointList[i - 1]
        p2 = pointList[i]
        if p1[1] < iy and p2[1] > iy:  # 由外到内
            a, b, c = fun(p1, p2)
            x = (-c - b * iy) / a
            intersection = [x, iy]
            newList.append(intersection)
            newList.append(p2)
        elif p1[1] > iy and p2[1] > iy:  #由内到内
            newList.append(p2)
        elif p1[1] > iy and p2[1] < iy:  #由内到外
            a, b, c = fun(p1, p2)
            x = (-c - b * iy) / a
            intersection = [x, iy]
            newList.append(intersection)
    return newList


def clip_right(pointList):
    pointList = clip_bottom(pointList)
    global ix, iy, px, py
    newList = []
    for i in range(len(pointList)):
        p1 = pointList[i - 1]
        p2 = pointList[i]
        if p1[0] > px and p2[0] < px:  # 由外到内
            a, b, c = fun(p1, p2)
            y = (-c - a * px) / b
            intersection = [px, y]
            newList.append(intersection)
            newList.append(p2)
        elif p1[0] < px and p2[0] < px:  #由内到内
            newList.append(p2)
        elif p1[0] < px and p2[0] > px:  #由内到外
            a, b, c = fun(p1, p2)
            y = (-c - a * px) / b
            intersection = [px, y]
            newList.append(intersection)
    return newList


def clip_top(pointList):
    pointList = clip_right(pointList)
    global ix, iy, px, py
    newList = []
    for i in range(len(pointList)):
        p1 = pointList[i - 1]
        p2 = pointList[i]
        if p1[1] > py and p2[1] < py:  # 由外到内
            a, b, c = fun(p1, p2)
            x = (-c - b * py) / a
            intersection = [x, py]
            newList.append(intersection)
            newList.append(p2)
        elif p1[1] < py and p2[1] < py:  #由内到内
            newList.append(p2)
        elif p1[1] < py and p2[1] > py:  #由内到外
            a, b, c = fun(p1, p2)
            x = (-c - b * py) / a
            intersection = [x, py]
            newList.append(intersection)
    return newList


if __name__ == '__main__':
    img = np.zeros((512, 512, 3), np.uint8)
    cv2.namedWindow('image')
    cv2.setMouseCallback('image', draw_rectangle)
    while (1):
        cv2.imshow('image', img)
        k = cv2.waitKey(1) & 0xFF
        if k == ord('q'):
            break
        elif k == 27:
            break
    cv2.destroyAllWindows()

    recList = [[ix, iy], [px, iy], [px, py], [ix, py], [ix, iy]]  # 裁剪窗口
    flag = []
    pointList = l
    pointList.append(l[0])
    newList = clip_top(pointList)
    newList.append(newList[0])
    x = [x[0] for x in recList]
    y = [x[1] for x in recList]
    x1 = [x[0] for x in pointList]
    y1 = [x[1] for x in pointList]
    x2 = [x[0] for x in newList]
    y2 = [x[1] for x in newList]

    # 把坐标系原点设置为左上角，使得plt与cv2保持一致
    ax = plt.gca()  # 获取到当前坐标轴信息
    ax.xaxis.set_ticks_position('top')  # 将X坐标轴移到上面
    ax.invert_yaxis()  # 反转Y坐标轴
    plt.plot(x, y)
    plt.plot(x1, y1, color='red')
    plt.plot(x2, y2, color='black', linewidth='2')
    plt.show()
```