《线画图元生成算法实验》

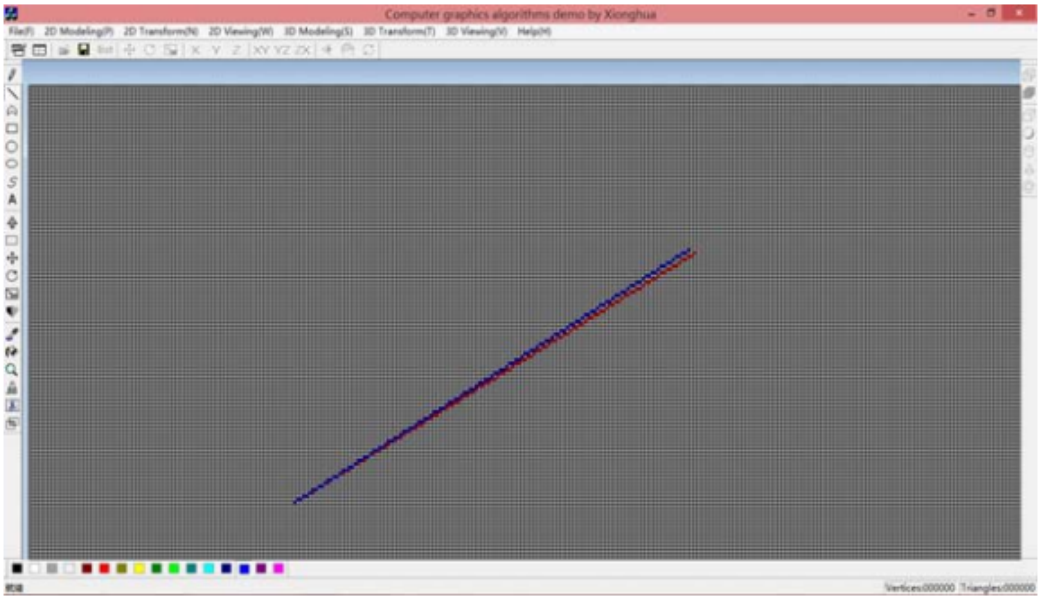
姓名 王红阳
学号 3019244233
专业 计算机科学与技术
班级 3班

天津大学智能与计算学部
2021年 09月27 日

一、实验目的

- 学习如何使用编程语言生成直线

二、实验内容

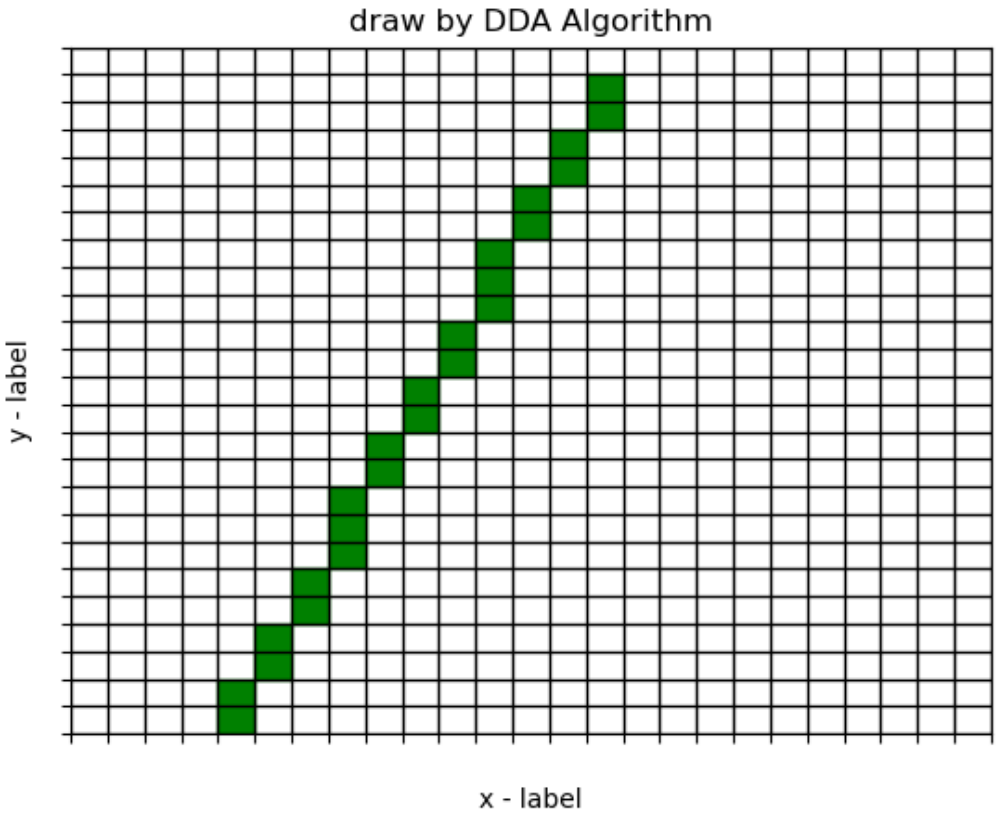
- 内容：
  - 实现 DDA 直线生成算法
  - 实现 Bresenham 直线生成算法
- 要求：
  - 自定义直线段起始点和终点坐标
  - 采用不同的彩色显示两种算法生成的直线结果
  - 为突出显示效果，采用网格表示像素，实现如下绘制效果。
  - 

三、实验结果

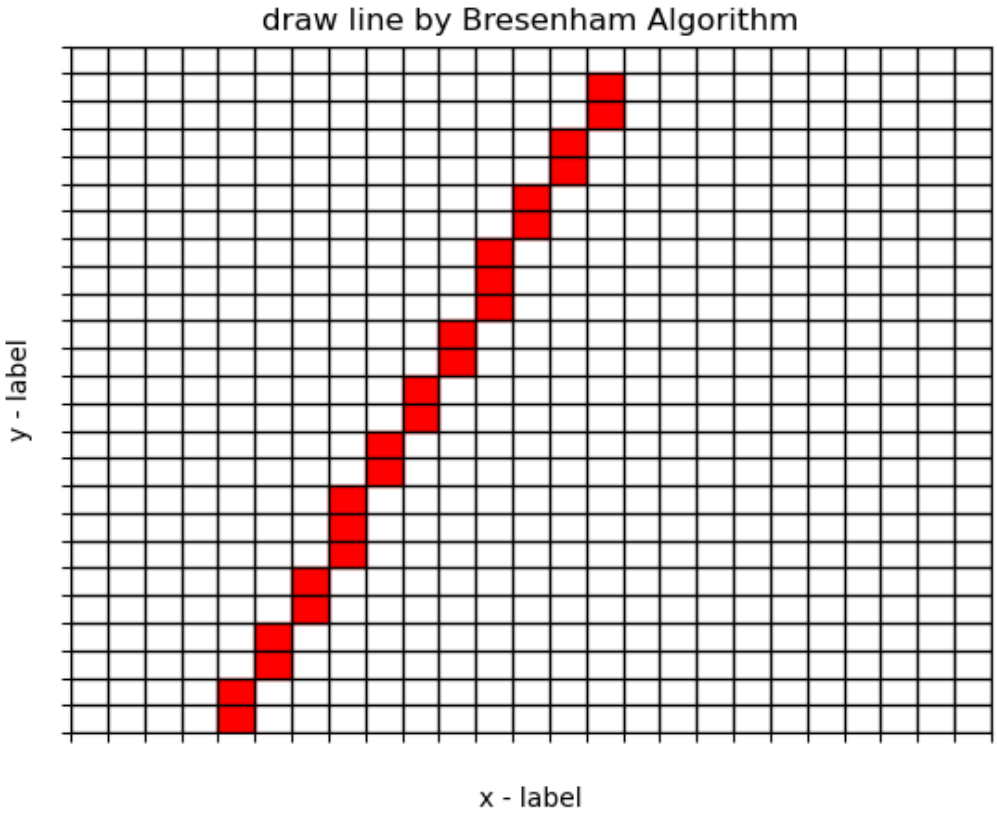
我本打算同时输入一组起点和终点，然后用不同颜色显示DDA算法和Bresenham算法生成的线段。

但是实践之后，发现两条线段经常是重合的，所以我只能分开显示这两个算法生成的线段了。

使用DDA算法生成的线段（起点为（5,1），终点为（15,24））：



使用Bresenham算法生成的线段（起点为（5,1），终点为（15,24））：



四、实验分析和总结

由于我电脑上的python语言的环境搭建的比较完善，经过调查后，我发现matplotlib模块可以满足我的要求。

代码运行过程为：启动代码，然后调用input函数，用于输入起点和终点，然后调用draw_line_by_DDA算法和draw_line_by_Bresenham算法算出需要画的一系列像素坐标，然后将这些算出来的像素坐标传入draw_line函数，画出线段，然后选择另存为保存到本机

在本次实验中我分别使用了 `DDA` 算法和 `Bresenham` 算法生成直线图形，并对原始算法进行升级改造，使其适应直线的不同斜率。

通过本次实验，我更加深入地了解了如何使用计算机编程语言画出图形，为之后的进一步深入了解计算机图形学领域奠定了基础。

五、源代码

main.py:

```
import numpy as np
from pylab import *
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
import matplotlib.patches as patches


def Round(x):
    """四舍五入取整函数，与python自带的取整函数round()不同

    Args:
        x (int): 一个点

    Returns:
        取整后的x
    """
    if x >= 0:
        return int(x + 0.5)
    else:
        return int(x - 0.5)


def Input():
    """输入起点和终点
    """
    x_start, y_start = map(int, input("请输入起点: ").split())
    x_end, y_end = map(int, input("请输入终点: ").split())
    return x_start, y_start, x_end, y_end


def draw_line_by_DDA(x_start, y_start, x_end, y_end):

    # 对x,y 求出min和max
    min_x = min(x_start, x_end)
    max_x = max(x_start, x_end)
```

```
    min_y = min(y_start, y_end)
    max_y = max(y_start, y_end)

    # 定义像素点列表
    x_list = []
    y_list = []

    if x_start == x_end:
        # 斜率不存在时
        yi = min_y
        for i in range(max_y - min_y + 1):
            x_list.append(x_start)
            y_list.append(yi)
            yi = yi + 1
    else:
        k = (y_end - y_start) / (x_end - x_start)
        if abs(k) <= 1:
            # k的正负无所谓
            xi = min_x
            yi = min_y if k >= 0 else max_y
            # 0 <= k <= 1时,直线递增, 当x最小时, y最小
            # -1<= k <= 0时, 直线递减, 当x最小时, y最大
            for i in range(max_x - min_x + 1):
                x_list.append(xi)
                y_list.append(Round(yi))
                xi = xi + 1  # x增1
                yi = yi + k  # y增k
        else:
            # abs(k)大于1的情况
            t = 1 / k
            yi = min_y
            xi = min_x if k > 0 else max_x
            for i in range(max_y - min_y + 1):
                x_list.append(Round(xi))
                y_list.append(yi)
                xi = xi + t
                yi = yi + 1

    return x_list, y_list


def draw_line_by_Bresenham(x_start, y_start, x_end, y_end):

    dx = abs(x_end - x_start)
    dy = abs(y_end - y_start)

    # 根据直线的走势方向, 设置变化的单位是正是负
    s1 = 1 if ((x_end - x_start) > 0) else -1
    s2 = 1 if ((y_end - y_start) > 0) else -1

    # 根据斜率的大小, 交换dx和dy, 可以理解为变化x轴和y轴使得斜率的绝对值为[0,1]
    change_xy = False
```

```
    if dy > dx:
        dx, dy = dy, dx
        change_xy = True

    # 初始误差
    p = 2 * dy - dx
    x = x_start
    y = y_start

    # 定义像素点列表
    x_list = []
    y_list = []

    # 循环，以求出所有像素点
    for i in range(0, int(dx + 1)):
        x_list.append(x)
        y_list.append(y)
        # plt.plot(x, y, color)
        if p >= 0:
            # 此时要选择横纵坐标都不同的点，根据斜率的不同，让变化小的一边变化一个单位
            if change_xy:
                x += s1
            else:
                y += s2
            p -= 2 * dx
        # 根据斜率的不同，让变化大的方向改变一单位，保证两边的变化小于等于1单位，让直线更加均匀
        if change_xy:
            y += s2
        else:
            x += s1
        p += 2 * dy

    return x_list, y_list


def add_pixel(x_list, y_list, ax, color):
    """在画布上画上像素点

    Args:
        x_list和y_list:  像素点位置
    """
    for i in range(len(x_list)):
        x = x_list[i]
        y = y_list[i]
        ax.add_patch(patches.Rectangle((x - 1, y - 1), 1, 1, color=color))


def draw_line(x_list, y_list, color, title):
```

```
    # 定义画布
    fig = plt.figure()
    ax = fig.add_subplot(111)

    # 定义 title和 label名称
    plt.title(title)
    plt.xlabel("x - label")
    plt.ylabel("y - label")

    # -----------设置网格环境-------------
    # 设置长宽
    min_width = min(x_list[0], y_list[0]) - 1
    max_width = max(x_list[-1], y_list[-1]) + 1
    ax.axis([min_width, max_width, min_width, max_width])

    # 设置主刻度标签的位置
    majorLocator = MultipleLocator(1)
    ax.xaxis.set_major_locator(majorLocator)
    ax.yaxis.set_major_locator(majorLocator)

    # 设置标签文本的格式
    majorFormatter = FormatStrFormatter(' ')
    ax.xaxis.set_major_formatter(majorFormatter)
    ax.yaxis.set_major_formatter(majorFormatter)

    ax.grid(True, color='black', linewidth=1)
    # ----------------------------------

    # 画点
    add_pixel(x_list, y_list, ax, color)

    plt.show()


def main():
    x_start, y_start, x_end, y_end = Input()

    color_for_DDA = 'g'
    color_for_Bresenham = 'r'

    DDA_x_list, DDA_y_list = draw_line_by_DDA(x_start, y_start, x_end,
y_end)
    Bresenham_x_list, Bresenham_y_list = draw_line_by_Bresenham(
        x_start, y_start, x_end, y_end)

    draw_line(DDA_x_list, DDA_y_list, color_for_DDA, "draw by DDA
Algorithm")
    draw_line(Bresenham_x_list, Bresenham_y_list, color_for_Bresenham,
              "draw line by Bresenham Algorithm")


if __name__ == '__main__':
```

```
    main()
```