计算机图形学第二次作业——任意宽度的直线生成算法

> 1851197 周楷彬

- 文件目录：
  - `bresenham.py`：Bresenham画线算法
  - `bresenham_line_brush.py`：基于Bresenham算法的线刷子算法
  - `bresenham_square_brush.py`：基于Bresenham算法的方形刷子算法
  - `utils.py`：一些辅助函数
  - `outputs/`：结果图片

线刷子

算法思想

- 按照原始的直线段的扫描转换算法点亮栅格中的像素，与此同时在每次点亮的时候用一个具有一定宽度的“线刷子”点亮构成宽度的像素
- 垂直线刷子：直线斜率在[-1, 1]时，把刷子置成垂直方向
- 水平线刷子：直线斜率不在[-1, 1]时，把刷子置成水平方向

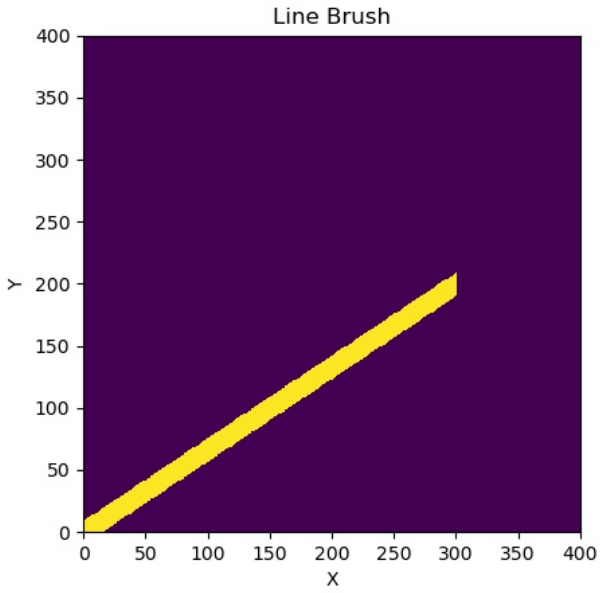
算法实现

在Bresenham画线算法的基础上，实现有宽度的直线生成算法

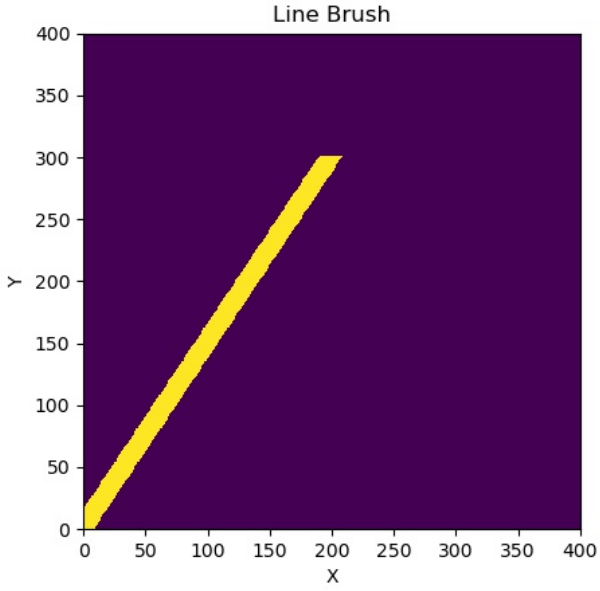
```
def BresenhamLine_line_brush(img,x0,y0,x1,y1,width,color):
    dx = x1 - x0
    dy = y1 - y0
    if abs(dy)<=abs(dx):          #k∈[-1,1]，垂直线刷子
        for i in range(-width//2,width//2):
            img = BresenhamLine(img,x0,y0+i,x1,y1+i,color)
    else:                         #k∉[-1,1]，水平线刷子
        for i in range(-width//2,width//2):
            img = BresenhamLine(img,x0+i,y0,x1+i,y1,color)
    return img
```

算法结果

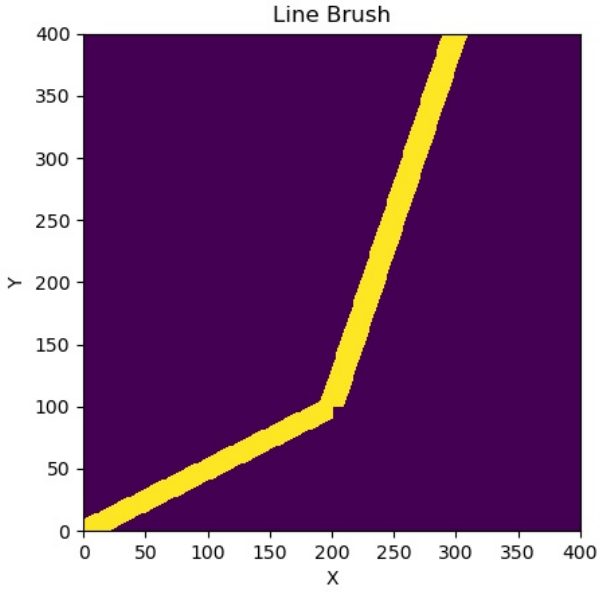
- $k \in [-1, 1]$:



- $k \notin [-1, 1]$:



- 两线相接:



算法评价

- 优点：算法简单、效率高
- 缺点：
  - 线的始末端总是水平或垂直的。因此，当线宽较大时，看起来很不自然。
  - 当比较接近水平的线与比较接近垂直的线汇合时，汇合处将有缺口。
  - 对于宽度为偶数个像素的直线会产生偏移

方形刷子

算法思想

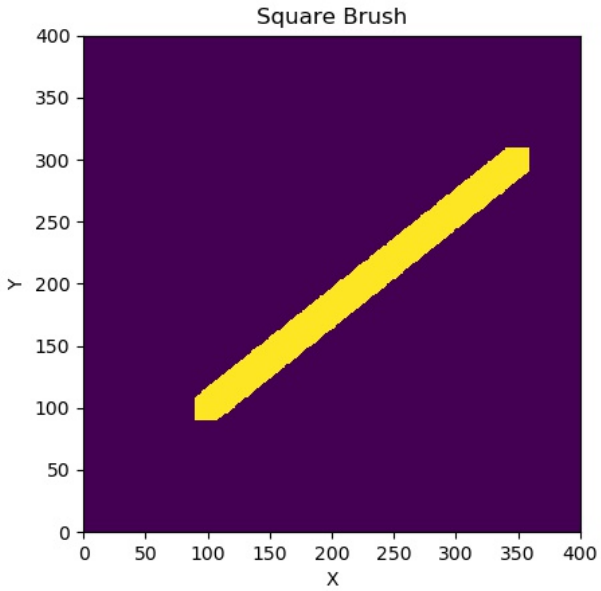
将边宽为指定线宽的方形刷子的中心放在直线的一个端点，方形刷子的中心沿着直线移动，直到直线的另一个端点

算法实现

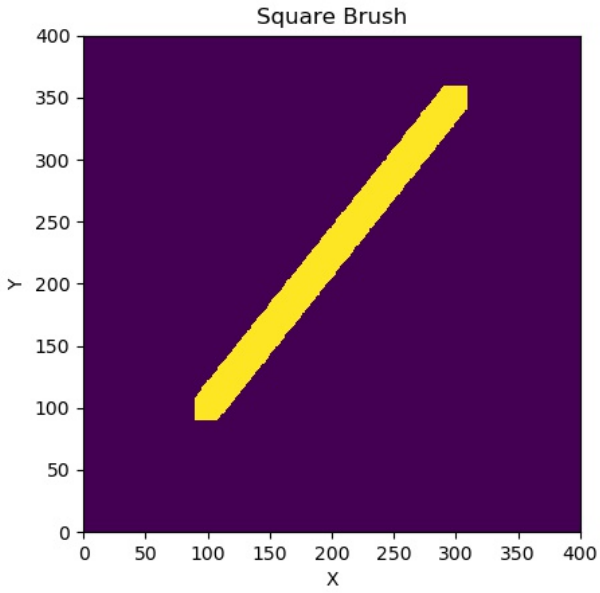
```
def BresenhamLine_square_brush(img,x0,y0,x1,y1,width,color):
    dx = x1 - x0
    dy = y1 - y0
    for i in range(-width//2,width//2):
        for j in range(-width//2,width//2):
            img = BresenhamLine(img,x0+i,y0+j,x1+i,y1+j,color)
    return img
```

算法结果

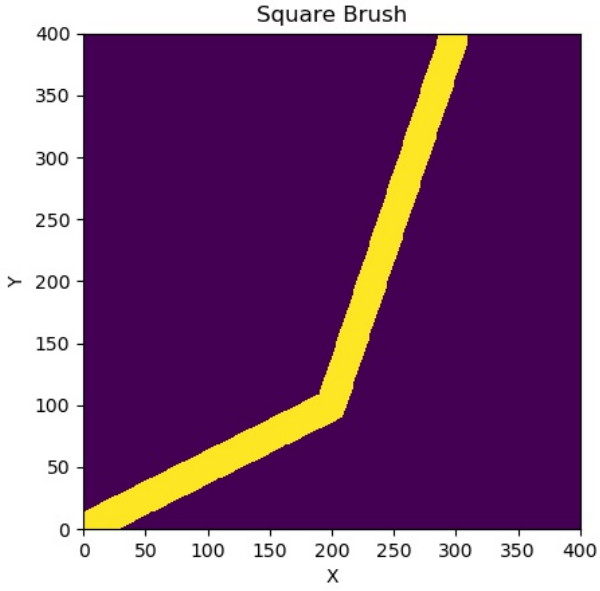
- $k \in [-1, 1]$:



- $k \notin [-1, 1]$:



- 两线相接：



算法评价

- 优点：实现简单
- 缺点：
  - 效率低：正方形的扫描方式导致相邻两个正方形有重叠
  - 用方刷子绘制的线条末端也是水平或垂直的，且线宽与线条方向有关
  - 对于宽度为偶数个像素的直线会产生偏移。