Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

Комп’ютерна графіка

Лабораторна робота №1

Растрова графіка.

Алгоритми растрового подання відрізка.

Виконав:

студент групи ФІ-91

Корешков Михайло

каф. ММАД

Перевірив:

Півень О.Б. каф. ІБ

Київ – 2022

Лабораторна робота №1.

Растрова графіка.

Алгоритми растрового подання відрізка.

**Мета**: ознайомитися з алгоритмом растрового подання відрізка.

**Завдання**: написати програму, що реалізує алгоритм растрового малювання відрізка.

« Реалізуйте за допомогою мови програмування алгоритм цифрового диференціального аналізатора для малювання відрізка. »

# 1. Вихідний код алгоритма растеризації

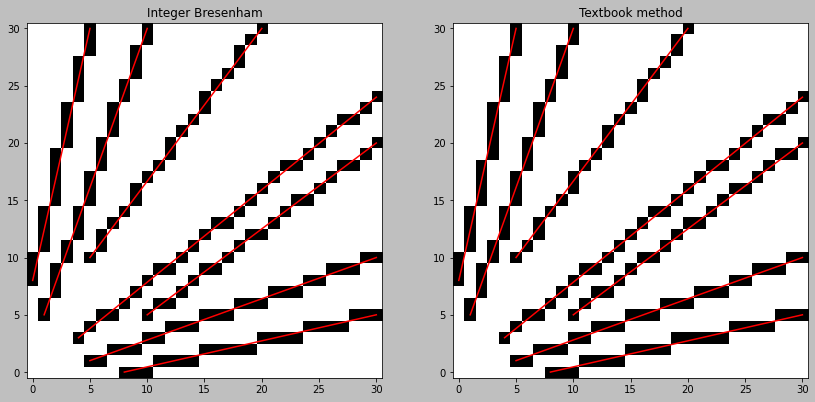
import numpy as np  
import matplotlib.pyplot as plt  
  
W = 1  
B = 0  
  
  
def build\_simple\_line(x1, x2, y1, y2):  
 if x1 != x2 and y1 != y2:  
 raise *ValueError*("Simple line must be parallel to X or Y axis")  
  
 if x1 == x2:  
 return np.stack([np.full(y2 - y1 + 1, x1), np.arange(y1, y2 + 1)]).T  
  
 return np.stack([np.arange(x1, x2 + 1), np.full(x2 - x1 + 1, y1)]).T  
  
  
def build\_line\_0(x1, y1, x2, y2):  
 if x1 == x2 or y1 == y2:  
 return build\_simple\_line(x1, x2, y1, y2)  
 dir = np.array([x2 - x1, y2 - y1])  
 dir = dir / np.linalg.norm(dir)  
 *# print("direction:", dir)* p = np.array([x1 + 0.0, y1 + 0.0])  
 line = [p.astype(*int*)]  
  
 while p[0] <= x2:  
 p += dir  
 pfloor = p.astype(*int*)  
 if (pfloor != line[-1]).any():  
 line.append(pfloor)  
  
 return line  
  
  
def build\_line\_1(x1, y1, x2, y2):  
 if x1 == x2 or y1 == y2:  
 return build\_simple\_line(x1, x2, y1, y2)  
  
 dir = np.array([x2 - x1, y2 - y1])  
 if dir[0] > dir[1]:  
 *# possible gaps in x* k = dir[1] / dir[0]  
 xs = np.arange(x1, x2 + 1)  
 ys = np.linspace(y1, y2, (x2 - x1 + 1), endpoint=True).astype(*int*)  
 else:  
 *# possible gaps in y* k = dir[0] / dir[1]  
 ys = np.arange(y1, y2 + 1)  
 xs = np.linspace(x1, x2, (y2 - y1 + 1), endpoint=True).astype(*int*)  
  
 return np.stack([xs, ys]).T  
  
  
def build\_line\_textbook(x1, y1, x2, y2):  
 if x1 == x2 or y1 == y2:  
 return build\_simple\_line(x1, x2, y1, y2)  
  
 dir = np.array([x2 - x1, y2 - y1])  
 if dir[0] > dir[1]:  
 *# possible gaps in x* k = dir[1] / dir[0]  
 xs = np.arange(x1, x2 + 1)  
 ys = np.linspace(y1, y2, (x2 - x1 + 1), endpoint=True)  
 ys = (ys + 0.5).astype(*int*)  
 else:  
 *# possible gaps in y* k = dir[0] / dir[1]  
 ys = np.arange(y1, y2 + 1)  
 xs = np.linspace(x1, x2, (y2 - y1 + 1), endpoint=True)  
 xs = (xs + 0.5).astype(*int*)  
  
 return np.stack([xs, ys]).T  
  
  
def build\_line\_integer\_bresenham(x1, y1, x2, y2):  
 if x1 == x2 or y1 == y2:  
 return build\_simple\_line(x1, x2, y1, y2)  
  
 dx = x2-x1  
 dy = y2-y1  
 line = []  
  
 if dx > dy:  
 *# possible gaps in x* D = (dy << 1) - dx  
 y = y1  
  
 for x in *range*(x1, x2+1):  
 line.append((x,y))  
 if D > 0:  
 y += 1  
 D -= dx << 1  
 D += dy << 1  
 else:  
 *# possible gaps in y* D = (dx << 1) - dy  
 x = x1  
  
 for y in *range*(y1, y2+1):  
 line.append((x,y))  
 if D > 0:  
 x += 1  
 D -= dy << 1  
 D += dx << 1  
  
 return line  
  
  
def rect(canvas, color=0, x1=0, y1=0, x2=None, y2=None):  
 if x2 is None:  
 x2 = canvas.shape[0]  
 if y2 is None:  
 y2 = canvas.shape[1]  
  
 canvas[x1:x2 + 1, y1:y2 + 1] = color  
 return canvas  
  
  
def draw(canvas, points, color=1):  
 for p in points:  
 *# print(p)* canvas[p[0], p[1]] = color  
 return canvas  
  
  
def show(canvas, \*args, \*\*kwargs):  
 return plt.imshow(canvas.T, interpolation='none', origin='lower', \*args, \*\*kwargs)

# 2. Результати роботи алгоритму

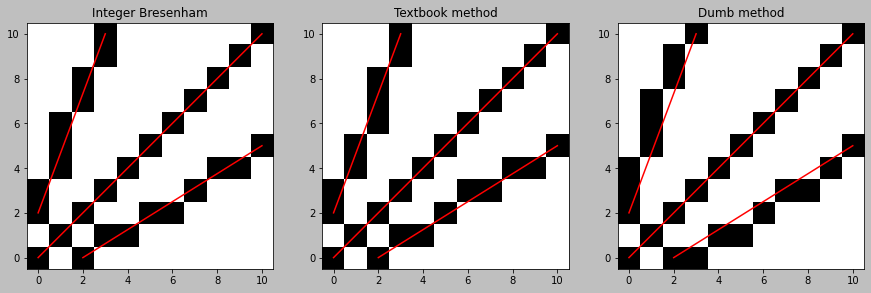
import numpy as np  
import matplotlib.pyplot as plt  
from lab1 import \*

WIDTH = 30  
HEIGHT = 30

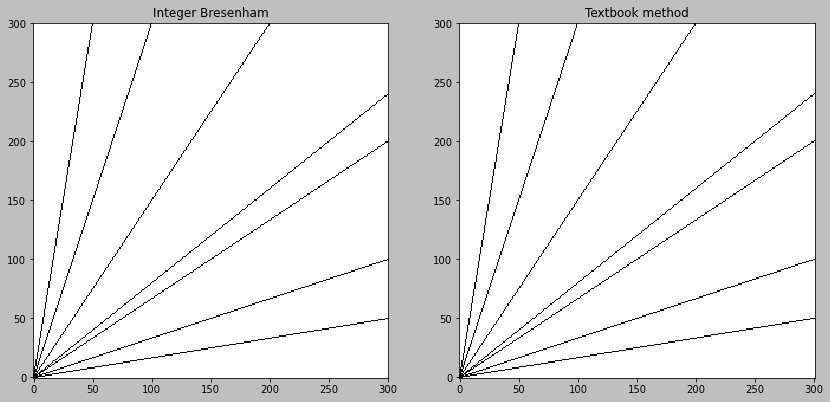
lines = [  
 (0,8,5,30),  
 (8,0,30,5),  
 (1,5,10,30),  
 (5,1,30,10),  
 (5,10,20,30),  
 (10,5,30,20),  
 (4,3,30,24),  
]  
  
canvas1 = np.full((WIDTH+1,HEIGHT+1), 1)  
canvas2 = np.full((WIDTH+1,HEIGHT+1), 1)  
for L in lines:  
 canvas1 = draw(canvas1, build\_line\_integer\_bresenham(\*L), 0)  
 canvas2 = draw(canvas2, build\_line\_textbook(\*L), 0)  
   
plt.style.use('grayscale')  
fig, axs = plt.subplots(1,2, figsize=(14,14))  
  
axs[0].imshow(canvas1.T, interpolation='none', origin='lower')  
axs[0].set\_title("Integer Bresenham")  
axs[1].imshow(canvas2.T, interpolation='none', origin='lower')  
axs[1].set\_title("Textbook method")  
  
for L in lines:  
 axs[0].plot([L[0],L[2]],[L[1],L[3]], 'r-', lw=1.5)  
 axs[1].plot([L[0],L[2]],[L[1],L[3]], 'r-', lw=1.5)



canvas1 = np.full((11,11), 1)  
canvas2 = np.full((11,11), 1)  
canvas3 = np.full((11,11), 1)  
  
lines = [  
 (0, 2, 3, 10),  
 (2, 0, 10, 5),  
 (0,0,10,10)  
]  
  
for L in lines:  
 canvas1 = draw(canvas1, build\_line\_integer\_bresenham(\*L), 0)  
 canvas2 = draw(canvas2, build\_line\_textbook(\*L), 0)  
 canvas3 = draw(canvas3, build\_line\_1(\*L), 0)  
  
plt.style.use('grayscale')  
fig, axs = plt.subplots(1,3, figsize=(15,15))  
  
axs[0].imshow(canvas1.T, interpolation='none', origin='lower')  
axs[0].set\_title("Integer Bresenham")  
axs[1].imshow(canvas2.T, interpolation='none', origin='lower')  
axs[1].set\_title("Textbook method")  
axs[2].imshow(canvas3.T, interpolation='none', origin='lower')  
axs[2].set\_title("Dumb method")  
  
for L in lines:  
 axs[0].plot([L[0],L[2]],[L[1],L[3]], 'r-', lw=1.5)  
 axs[1].plot([L[0],L[2]],[L[1],L[3]], 'r-', lw=1.5)  
 axs[2].plot([L[0],L[2]],[L[1],L[3]], 'r-', lw=1.5)

**

lines = np.array([  
 (0,0,5,30),  
 (0,0,30,5),  
 (0,0,10,30),  
 (0,0,30,10),  
 (0,0,20,30),  
 (0,0,30,20),  
 (0,0,30,24),  
])  
  
lines \*= 10  
  
canvas1 = np.full((WIDTH\*10+1,HEIGHT\*10+1), 1)  
canvas2 = np.full((WIDTH\*10+1,HEIGHT\*10+1), 1)  
for L in lines:  
 canvas1 = draw(canvas1, build\_line\_integer\_bresenham(\*L), 0)  
 canvas2 = draw(canvas2, build\_line\_textbook(\*L), 0)  
  
plt.style.use('grayscale')  
fig, axs = plt.subplots(1,2, figsize=(14,14))  
  
axs[0].imshow(canvas1.T, interpolation='none', origin='lower')  
axs[0].set\_title("Integer Bresenham")  
axs[1].imshow(canvas2.T, interpolation='none', origin='lower')  
axs[1].set\_title("Textbook method")



# Висновки

Реалізовано алгоритми растеризації та відрисовки прямих.