Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

Комп’ютерна графіка

Лабораторна робота №2

Побудова графіка функції за алгоритмом Брезенхейма

Виконав:

студент групи ФІ-91

Корешков Михайло

каф. ММАД

Перевірив:

Півень О.Б. каф. ІБ

Київ – 2022

Лабораторна робота №2.

Побудова графіка функції за алгоритмом Брезенхейма

# 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:

        y1, y2 = min(y1, y2), max(y1,y2)

        return list(np.stack([np.full(y2 - y1 + 1, x1), np.arange(y1, y2 + 1)]).T)

    x1, x2 = min(x1, x2), max(x1,x2)

    return list(np.stack([np.arange(x1, x2 + 1), np.full(x2 - x1 + 1, y1)]).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 = np.abs(x2-x1)

    dy = np.abs(y2-y1)

    sx = 1 if x2 > x1 else -1

    sy = 1 if y2 > y1 else -1

    line = []

    if dx > dy:

        # possible gaps in x

        D = (dy << 1) - dx

        y = y1

        for x in range(x1, x2+sx, sx):

            line.append((x,y))

            while D >= 0:

                y += sy

                D -= dx << 1

            D += dy << 1

    else:

        # possible gaps in y

        if x2 < x1:

            step = -1

        D = (dx << 1) - dy

        x = x1

        for y in range(y1, y2+sy, sy):

            line.append((x,y))

            while D >= 0:

                x += sx

                D -= dy << 1

            D += dx << 1

    return line

def f2p\_from\_xywh(w, h, dx, dy, w0, h0):

    kx, ky = w/dx, h/dy

    def f2p(points):

        return (points \* np.array([[kx], [ky]]) + np.array([[w0],[h0]])).astype(int)

    return f2p

def build\_bresenham\_polygon(xs, ys, f2p= lambda x: x):

    curve = []

    xs, ys = f2p(np.stack([xs, ys]))

    for i in range(1,len(xs)):

        curve += build\_line\_integer\_bresenham(xs[i-1], ys[i-1], xs[i], ys[i])

    return curve

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. Подробиці Варіанту 6

Варіант полягає у відрисовці Кардіоїди та Овалу Кассіні.

1. Кардіоіда задається простим параметричним рівнянням
2. Овал Кассіні задається складніше. В залежності від відношення його параметрів, він приймає різні форми.
   1. За a < b він має форму неперервної кривої, яка задає бієкцію між та
   2. За a = b він має форму зв’язкої вісьмірки (лемніскати Бернулі)
   3. За a > b він складається з двох незв’язних частин, симетричних відносно OY

Для останніх двох варіантів необхідно будувати фігуру по частинам. Тому функція генерації координат точок повертає в першу чергу список окремих частин фігури

def get\_cardioid(a=1, b=1):

    def \_cardioid(n):

        t = np.linspace(0, 2\*np.pi, n)

        ct = np.cos(t)

        st = np.sin(t)

        return a\*(ct\*\*2) + b \* ct, ct \* st + b \* st

    return \_cardioid

def get\_cassini(a=0, b=1):

    def \_cassini(n):

        if b > a:

            phi = np.linspace(0, np.pi, n)

            a2cos2phi = a\*a\*np.cos(2\*phi)

            root = np.sqrt(b\*\*4 - (a\*\*2 \* np.sin(2\*phi))\*\*2)

            ans1 = np.sqrt(a2cos2phi + root)

            return [(phi, ans1), (phi+np.pi, ans1)]

        elif b < a:

            print("b/a: ", b/a)

            phi0 = np.arcsin((b/a)\*\*2)/2

            phi\_full = np.linspace(-phi0, phi0, n)

            phi = phi\_full[1:-1]

            a2cos2phi = a\*a\*np.cos(2\*phi)

            root = np.sqrt(b\*\*4 - (a\*\*2 \* np.sin(2\*phi))\*\*2)

            ans1 = np.sqrt(a2cos2phi + root)

            ans2 = np.sqrt(a2cos2phi - root)

            ans3 = np.sqrt(a\*a\*np.cos(2\*np.array([-phi0, phi0])))

            print("shapes: ", ans1.shape, ans3.shape)

            ans1\_full = np.concatenate([[ans3[0]],ans1,[ans3[1]]])

            ans2\_full = np.concatenate([[ans3[0]],ans2,[ans3[1]]])

            phis = [phi\_full, phi\_full, phi\_full+np.pi, phi\_full+np.pi]

            rs = [ans1\_full, ans2\_full, ans1\_full, ans2\_full]

            return list(zip(phis, rs))

        elif b == a:

            phi = np.linspace(-np.pi/4, np.pi/4, n)

            ans = 2\*a\*a\*np.cos(2\*phi)

            return [(phi,ans), (phi+np.pi,ans)]

    return \_cassini

# 3. Дослідження фігур

def demo\_cardioid():

    params = [

        (1,1), (3,1), (1,3), (0.5,1), (1,0.5), (1,0.25)

    ]

    fig, axs = plt.subplots(2,3)

    for ax, (cardioid\_a, cardioid\_b) in zip(axs.flat, params):

        cardioid = get\_cardioid(cardioid\_a, cardioid\_b)

        ax.scatter(\*cardioid(100))

        ax.set\_title(f"Кардіоїда a={cardioid\_a}, b={cardioid\_b}")

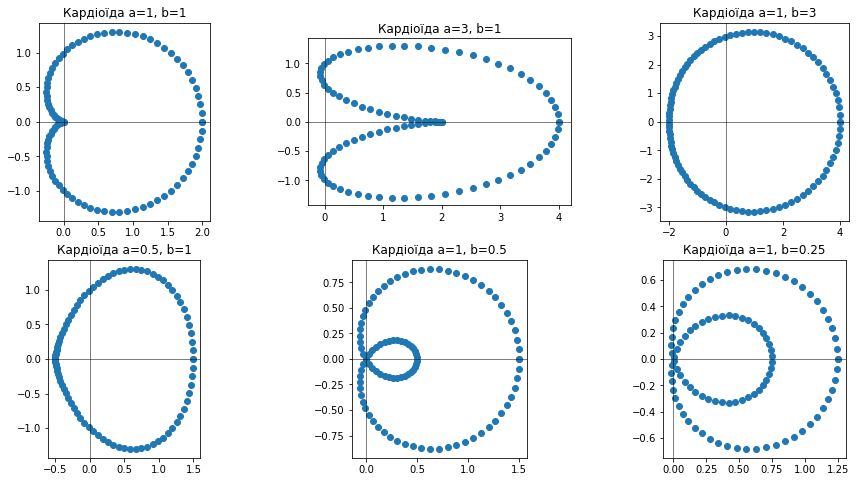
        ax.axhline(0, c='k', lw=0.5)

        ax.axvline(0, c='k', lw=0.5)

        ax.set\_aspect(1)

    fig.set\_size\_inches((16,8))

demo\_cardioid()



def demo\_cassini():

    params = [

        (0.6, 1), (0.8, 1), (1,1), (1.1, 1), (1.4, 1), (1.6, 1)

    ]

    fig, axs = plt.subplots(2,3, subplot\_kw={'projection': 'polar'})

    for ax, (a, b) in zip(axs.flat, params):

        oval = get\_cassini(a, b)

        for block in oval(100):

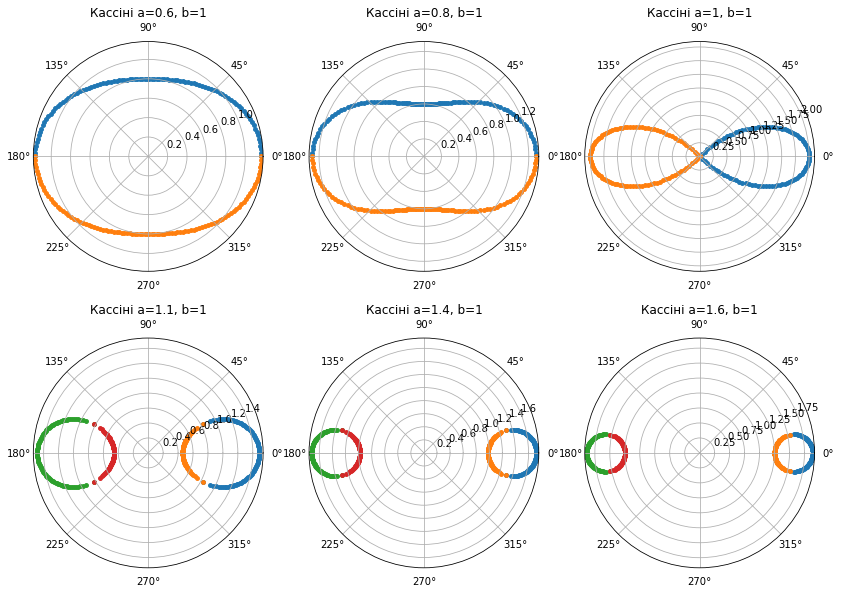
            ax.scatter(\*block, s=15)

        ax.set\_title(f"Кассіні a={a}, b={b}")

        ax.set\_aspect(1)

    fig.set\_size\_inches((14,10))

demo\_cassini()



# 4. Демонстрація роботи алгоритму відрисовки

def plot\_cardioid(a, b, N):

    cardioid = get\_cardioid(a, b)

    f2p = f2p\_from\_xywh(WIDTH, HEIGHT, 2\*(a+b), 2\*(a+b), WIDTH/4, HEIGHT/2)

    canvas = np.zeros((WIDTH+1, HEIGHT+1))

    curve = build\_bresenham\_polygon(\*cardioid(N), f2p)

    canvas = draw(canvas, curve)

    show(canvas)

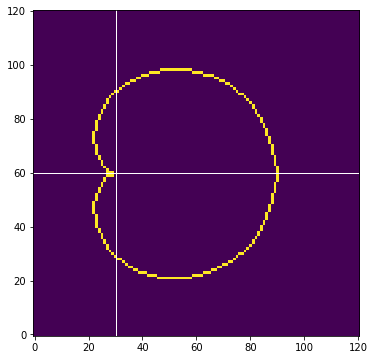
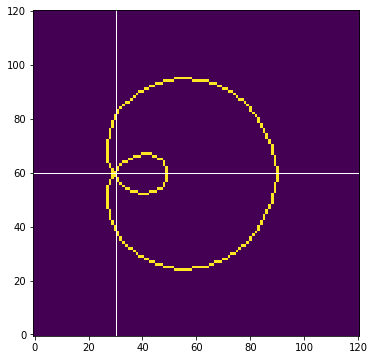
    plt.gcf().set\_size\_inches((6,6))

    plt.gca().axvline(WIDTH/4, c='w', lw=1)

    plt.gca().axhline(HEIGHT/2, c='w', lw=1)

plot\_cardioid(1,0.5,40)

plot\_cardioid(1,1,40)



def plot\_oval(a, b, N):

    oval = get\_cassini(a, b)

    blocks = oval(N)

    f2p = f2p\_from\_xywh(WIDTH, HEIGHT, 2\*(a\*\*2+b\*\*2), 2\*(a\*\*2+b\*\*2), WIDTH/2, HEIGHT/2)

    canvas = np.zeros((WIDTH, HEIGHT))

    for phi, r in blocks:

        xs = np.cos(phi) \* r

        ys = np.sin(phi) \* r

        curve = build\_bresenham\_polygon(xs, ys, f2p)

        canvas = draw(canvas, curve)

    show(canvas)

    plt.gcf().set\_size\_inches((6,6))

    plt.gca().axvline(WIDTH/2, c='w', lw=1)

    plt.gca().axhline(HEIGHT/2, c='w', lw=1)

plot\_oval(1,1,50)

plot\_oval(1.2,1.1,50)

plot\_oval(0.95,1,50)

