­­­­­­­­­

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКО”

Факультет прикладної математики

Кафедра системного програмування і спеціальних комп’ютерних систем

**Лабораторна робота №1**

З дисципліни «Комп’ютерна графіка»

«Алгoритми рaструвaння»

**Виконав:**

**студент III-го курсу**

**групи КВ-41**

**Горпинич-Радуженко Іван**

**Київ 2016**

Текст програми:

from \_\_future\_\_ import division

from PIL import Image

from time import \*

exp=[]

def \_fpart(x):

return x - int(x)

def \_rfpart(x):

return 1 - \_fpart(x)

def putpixel(img, xy, color, alpha=1):

c = tuple(map(lambda bg, fg: int(round(alpha \* fg + (1-alpha) \* bg)),

img.getpixel(xy), color))

img.putpixel(xy, c)

def wuline(x1,y1,x2,y2, color,image):

starclock=clock()

dx, dy = x2-x1, y2-y1

steep = abs(dx) < abs(dy)

p = lambda px, py: ((px,py), (py,px))[steep]

if steep:

x1, y1, x2, y2, dx, dy = y1, x1, y2, x2, dy, dx

if x2 < x1:

x1, x2, y1, y2 = x2, x1, y2, y1

grad = dy/dx

intery = y1 + \_rfpart(x1) \* grad

def draw\_endpoint(pt):

x, y = pt

xend = round(x)

yend = y + grad \* (xend - x)

xgap = \_rfpart(x + 0.5)

px, py = int(xend), int(yend)

putpixel(image, (px, py), color, \_rfpart(yend) \* xgap)

putpixel(image, (px, py+1), color, \_fpart(yend) \* xgap)

return px

xstart = draw\_endpoint(p(\*(x1,y1))) + 1

xend = draw\_endpoint(p(\*(x2,y2)))

for x in range(xstart, xend):

y = int(intery)

putpixel(image, p(x, y), color, \_rfpart(intery))

putpixel(image, p(x, y+1), color, \_fpart(intery))

intery += grad

exp.append({'name':"Wu Algorithm:",'clock':(clock()-starclock)})

def bresenham\_circle(x0, y0, radius, color, image):

starclock=clock()

f = 1 - radius

ddf\_x = 1

ddf\_y = -2 \* radius

x = 0

y = radius

image.putpixel((x0, y0 + radius), color)

image.putpixel((x0, y0 - radius), color)

image.putpixel((x0 + radius, y0), color)

image.putpixel((x0 - radius, y0), color)

while x < y:

if f >= 0:

y -= 1

ddf\_y += 2

f += ddf\_y

x += 1

ddf\_x += 2

f += ddf\_x

image.putpixel((x0 + x, y0 + y), color)

image.putpixel((x0 - x, y0 + y), color)

image.putpixel((x0 + x, y0 - y), color)

image.putpixel((x0 - x, y0 - y), color)

image.putpixel((x0 + y, y0 + x), color)

image.putpixel((x0 - y, y0 + x), color)

image.putpixel((x0 + y, y0 - x), color)

image.putpixel((x0 - y, y0 - x), color)

exp.append({'name':"Bresenham Circle",'clock':(clock()-starclock)})

def test\_time():

ttest=[]

c=4

for i in range(1500):

dda\_line(10, 10, 450, 150, black, img4)

bresenham\_line(10,10,450,150,black,img4)

wuline(10, 10, 450, 150,black,img4)

bresenham\_circle(250,250,100,black,img4)

bresenham\_ellipse(250,250,200,50,black,img4)

for elem in exp:

if c>=0:

ttest.append(elem)

c-=1

else:

for telem in ttest:

if elem['name']==telem['name']:

telem['clock']+=elem['clock']

return ttest

def dda\_line(x1, y1, x2, y2, color, image):

starclock=clock()

dy = y2 - y1

dx = x2 - x1

if abs(dx) > abs(dy):

steps = dx

else:

steps = dy

xIncrement = float(dx) / float(steps)

yIncrement = float(dy) / float(steps)

image.putpixel((x1, y1), color)

for i in range(steps):

x1 += xIncrement

y1 += yIncrement

image.putpixel((int(round(x1)), int(round(y1))), color)

exp.append({'name':"DDA Algorithm:",'clock':(clock()-starclock)})

def bresenham\_line(x1,y1,x2,y2,color,image):

starclock=clock()

dx=abs(x1-x2)

dy=abs(y1-y2)

sx=-1

sy=-1

if x2>x1:

sx=1

if y2>y1:

sy=1

error=dx-dy

while x1!=x2 or y1!=y2:

image.putpixel((x1,y1),color)

error\_=error

if error\_>-dy:

error-=dy

x1+=sx

if error\_<dx:

error+=dx

y1+=sy

exp.append({'name':"Bresenham Algorithm:",'clock':(clock()-starclock)})

def bresenham\_ellipse(x0,y0,width,height,color,image):

starclock=clock()

a2=width\*width

b2=height\*height

fa2=4\*a2

fb2=4\*b2

x = 0

y = height

sigma = 2\*b2+a2\*(1-2\*height)

while b2\*x <= a2\*y:

image.putpixel((x0 + x, y0 + y), color)

image.putpixel((x0 - x, y0 + y), color)

image.putpixel((x0 + x, y0 - y), color)

image.putpixel((x0 - x, y0 - y), color)

if sigma>=0:

sigma+=fa2\*(1-y)

y-=1

sigma+=b2\*(4\*x+6)

x+=1

x = width

y = 0

sigma = 2\*a2+b2\*(1-2\*width)

while a2\*y<=b2\*x:

image.putpixel((x0 + x, y0 + y), color)

image.putpixel((x0 - x, y0 + y), color)

image.putpixel((x0 + x, y0 - y), color)

image.putpixel((x0 - x, y0 - y), color)

if sigma>=0:

sigma+=fb2\*(1-x)

x-=1

sigma+=a2\*(4\*y+6)

y+=1

exp.append({'name':"Bresenham Ellipse",'clock':(clock()-starclock)})

def print\_time(expr):

print "Drawing 1500 lines(in seconds):"

for alg in expr:

print alg['name'],

print alg['clock']

def draw\_lastname(color,image):

bresenham\_line(15, 15, 15, 160, color, image)

bresenham\_line(15, 15, 60, 15, color, image)

bresenham\_ellipse(90,85,30,70,color,image)

bresenham\_line(135,10,135,160,color,image)

bresenham\_line(135,10,175,10,color,image)

bresenham\_line(135,60,175,60,color,image)

bresenham\_line(175,10,185,20,color,image)

bresenham\_line(175,60,185,50,color,image)

bresenham\_line(185,50,185,20,color,image)

bresenham\_line(195,10,195,160,color,image)

bresenham\_line(245,10,245,160,color,image)

bresenham\_line(195,10,245,10,color,image)

bresenham\_line(255, 10, 255, 160, color, image)

bresenham\_line(305, 10, 305, 160, color, image)

bresenham\_line(255, 160, 305, 10, color, image)

bresenham\_line(315, 10, 315, 160, color, image)

bresenham\_line(365, 10, 365, 160, color, image)

bresenham\_line(315, 85, 365, 85, color, image)

bresenham\_line(375, 10, 375, 160, color, image)

bresenham\_line(425, 10, 425, 160, color, image)

bresenham\_line(375, 160, 425, 10, color, image)

bresenham\_line(500, 10, 500, 160, color, image)

bresenham\_line(440, 10, 440, 60, color, image)

bresenham\_line(440, 60, 460, 70, color, image)

bresenham\_line(460, 70, 500, 70, color, image)

image.save("lastname.png")

def draw\_lines(color,image):

dda\_line(10,150,450,10,color,image)

bresenham\_line(10,250,450,100,color,image)

wuline(10, 350, 450, 200,color,image)

image.save("Lines.png")

def draw\_circles(color,image):

bresenham\_circle(170,270,75,color,image)

bresenham\_ellipse(100,100,100,50,color,image)

bresenham\_ellipse(350,300,50,100,color,image)

bresenham\_circle(270, 150, 75, color, image)

image.save("Circles.png")

white=(255,255,255)

black = (0, 150, 0)

img1 = Image.new("RGB", (500,400), white)

img2 = Image.new("RGB", (500,500), white)

img3 = Image.new("RGB", (510,170), white)

img4 = Image.new("RGB", (500,500), white)

tttime=test\_time()

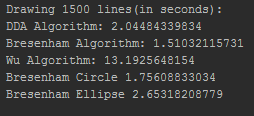
print\_time(tttime)

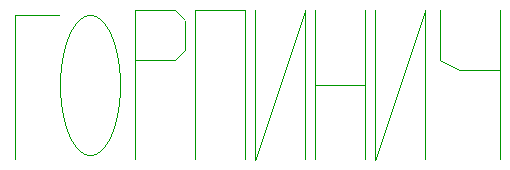
draw\_lines(black,img1)

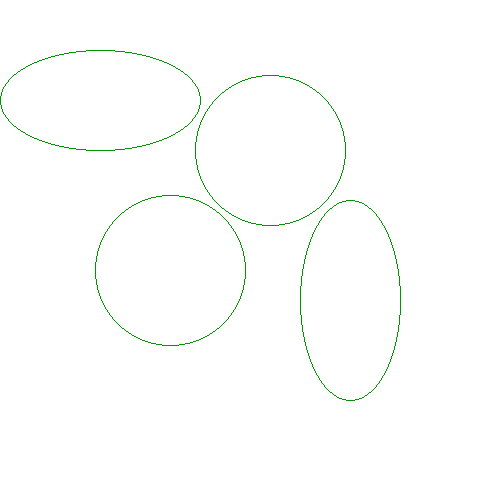
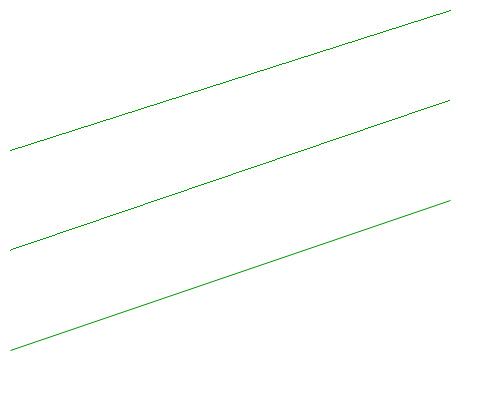
draw\_circles(black,img2)

draw\_lastname(black,img3)

Скріншоти:







Висновки:

**Алгоритм DDA-лінії** : Застосування обчислень з дійсними числами і лише одноразове використання округлення для остаточного отримання значення растрової координати зумовлюють високу точність і низьку швидкодію алгоритму.

[**Алгоритм Брезенхейма**](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D1%80%D0%B5%D0%B7%D0%B5%D0%BD%D1%85%D0%B5%D0%B9%D0%BC%D0%B0) малює відрізок дуже швидко, але він не виконує згладжування. Також він не може обробити ситуацію, коли кінцеві точки мають не цілочисельні координати.

Хоча [**алгоритм Ву**](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%92%D1%83) також часто використовується в сучасній комп'ютерній графіці через підтримку [згладжування](https://uk.wikipedia.org/wiki/%D0%97%D0%B3%D0%BB%D0%B0%D0%B4%D0%B6%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B0)), алгоритм Брезенхейма залишається вживаним завдяки його швидкісті і простоті.