**МГТУ им. Н.Э. Баумана**

**КОМПЬЮТЕРНАЯ ГРАФИКА**

**Лабораторный практикум №3**

**по теме: «*Построение отрезка*»**

***Студент: Нгуен Фыок Санг***

***Группa: ИУ7И-46Б***

***Работу проверил:***

2020

**Построение отрезка с использованием разных алгоритмов:**

* ЦДА
* Брезенхэма
* Ву

**I. Алгоритм ЦДА:**

1. ***Идея алгоритма:***
   1. Начальная точка A(xA, yA) и конечная точка B(xB, yB)
   2. Рассчитать количество пикселей линии, нарисованной на экране.
      1. Δx = xB – xA, Δy = yB – yA
      2. dx = |Δx|, dy = |Δy|
      3. l = max(dx, dy) //if (dx > dy) then l = dx, else l = dy
   3. Определить значения, чтобы добавить к х и у.
      1. Δx = Δx / l, Δy = Δy / l
   4. Найти координаты для рисования:
      1. Нач. Условие: x = xA, y = yA
      2. Циклипостроения отрезка:
         1. Высь. Т(E(x), E(y)), где Е функция окруления.
         2. x = x + Δx, y = y + Δy
2. *Схема*:

Δx = xB – xA,

Δy = yB – yA

dx = |Δx|,

dy = |Δy|

T(E(x), E(y))

x = x + Δx

y = y + Δy

i от 1 до l+1

Δx = Δx / l

Δy = Δy / l

x = xA

y = yA

l = dx

l = dy

dx > dy

да

нет

да

нет

1. *Код программы:*

// функция окруления

def ROUND(a):

return int(a + 0.5)

def line\_DDA(x1,y1,x2,y2):

x,y = x1,y1

dx = abs(x2 - x1)

dy = abs(y2 - y1)

length = dx if dx > dy else dy

dx = dx/float(length)

dy = dy/float(length)

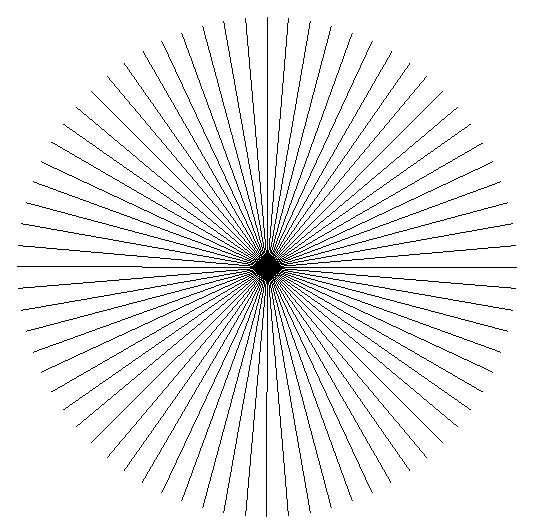
for i in range(int(length)):

img.put(line\_color, (ROUND(x),ROUND(y)))

x += dx

y += dy

1. Результат



**II. Алгоритм Брезенхэма:**

1. *Идея алгоритма:*
   * Построена точка (xi, yi)
   * Наити следушая точка:

Ошибка: **ei = Ygi – Yi**

Если ei < ½: выбирается нижная точка Yi+1 = Yi

Если ei ≥ ½: выбирается верхная точка Yi+1= Yi+1

ei+ 1 = Ygi+1 – Yi+1 = Ygi + m – Yi+1 = Ygi – Yi + m = ei + m

(на пред. Шаге Yi+1 = Yi)

* Если dy больше, чем dx, т.е. y изменяется быстрее, чем x, необходимо изменить x и y

1. Начальная точка A(xA, yA) и конечная точка B(xB, yB)
2. dx = xB – xA, dy = yB – yA
3. Δx = sign(dx), Δy = sign(dy)
4. dx = |dx|, dy = |dy|
5. Если dx > dy, то obmen = 0, иначе obmen = 1, t= dx, dx = dy, dy = t
6. m = dy / dx, e = m – ½ // **e = 2dy - dx**
7. Цикл построение:
8. Выс. T(x, y)
9. Если e > 0, то

Если obmen = 0, то y = y + Δy, иначе x + x + Δx

e = e – 1 // e = e – 2dx

Иначе (e>= 0), то

Если obmen = 0, то y = y + Δy, иначе x + x + Δx

e = e – m // **e = e + 2dy**

* Вышеприведенный алгоритм работает с действительными числами, поэтому он медленный и ошибки могут накапливаться, поэтому лучше преобразовать их в целые числа, чтобы они были быстрее и точнее.
  + или 2dx.e = 2dy – x
  + e = 2dx.e = 2dy – dx
* Алгоритм Брезенхэма с устранением ступенчатости:
  + Отрезок высв. Переменной интенсивностью.
  + Интерсивность писеля выбирается пропорционнально площади части писеля, находится под отрезкам.
  + Нач. писеля всегда высв. c половиной интенсивностью

S

S = Yi + m / 2

Si+1 = Si + m

Si+1 = Si + m - 1

1. Схема:

dx = xB – xA, dy = yB – yA

Δx = sign(dx), Δy = sign(dy)

dx = |dx|, dy = |dy|

dx>dy

да

нет

Obmen = 1

Obmen = 0

x = xA, y = yA

m = dy / dx

e = m – 1/2

нет

i от 1 до dx + 1

да

e > 0

нет

нет

да

obmen = 0

obmen = 1

да

x = x+Δx

x = x+Δx

y = y+Δy

y = y+Δy

e = e - 1

e = e - 1

1. *Код программы:*
2. *Алгоритм Брезенхэма (действ.)*

def line\_Brezenham\_float(x1, y1, x2, y2):

x, y = x1, y1

dx, dy = x2 - x1, y2 - y1

delx, dely = sign(dx), sign(dy)

dx, dy = abs(dx), abs(dy)

if dx > dy:

obmen = 0

else:

obmen = 1

t = dx

dx= dy

dy = t

m = dy / dx

e = m - 1/2

while (x != x2 or y != y2):

img.put(line\_color, (int(x),int(y)))

if e > 0:

if obmen == 0:

y += dely

else:

x += delx

e -= 1

else:

if obmen == 0:

x += delx

else:

y += dely

e += m

1. *Алгоритм Брезенхэма (цел.)*

def line\_Brezenham\_int(x1, y1, x2, y2):

x, y = x1, y1

dx, dy = x2 - x1, y2 - y1

delx, dely = sign(dx), sign(dy)

dx, dy = abs(dx), abs(dy)

if dx > dy:

obmen = 0

else:

obmen = 1

t = dx

dx= dy

dy = t

e = 2 \* dy - dx

while (x != x2 or y != y2):

img.put(line\_color, (int(x),int(y)))

if e > 0:

if obmen == 0:

y += dely

else:

x += delx

e -= 2 \* dx

else:

if obmen == 0:

x += delx

else:

y += dely

e += 2 \* dy

1. *Алгоритм Брезенхэма с устранением ступенчатости*

// Функция интенсивности

def get\_rgb\_intensity(color, intensity):

grad = []

(r1, g1, b1) = canvas.winfo\_rgb(color) # разложение цвета линни на составляющие ргб

(r2, g2, b2) = canvas.winfo\_rgb(bg\_color) # разложение цвета фона на составляющие ргб

r\_ratio = float(r2 - r1) / intensity # получение шага интенсивности

g\_ratio = float(g2 - g1) / intensity

b\_ratio = float(b2 - b1) / intensity

for i in range(intensity):

nr = int(r1 + (r\_ratio \* i)) # заполнение массива разными оттенками

ng = int(g1 + (g\_ratio \* i))

nb = int(b1 + (b\_ratio \* i))

grad.append("#%4.4x%4.4x%4.4x" % (nr, ng, nb))

grad.reverse()

return grad

// Алгоритм Брезенхэма с устранением ступенчатости

def line\_Brezenham(x1, y1, x2, y2):

x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

I = 100

colors = get\_rgb\_intensity(line\_color, I)

dx = x2 - x1

dy = y2 - y1

sx = sign(dx)

sy = sign(dy)

dy = abs(dy)

dx = abs(dx)

if dy >= dx:

dx, dy = dy, dx

obmen = 1

else:

obmen = 0

m = dy / dx \* I

e = I / 2

w = I - m

x = x1

y = y1

while x != x2 or y != y2:

I1 = int(round(e) - 1)

img.put(colors[I1], (int(x),int(y)))

if e < w:

if obmen == 0:

x += sx

else:

y += sy

e += m

elif e >= w:

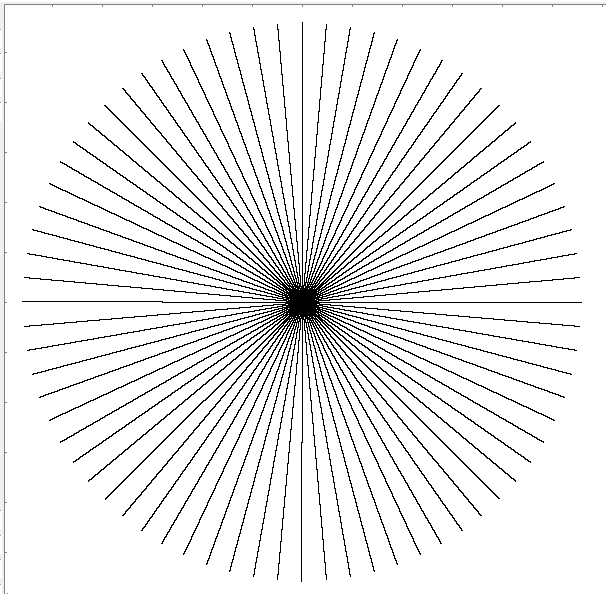
x += sx

y += sy

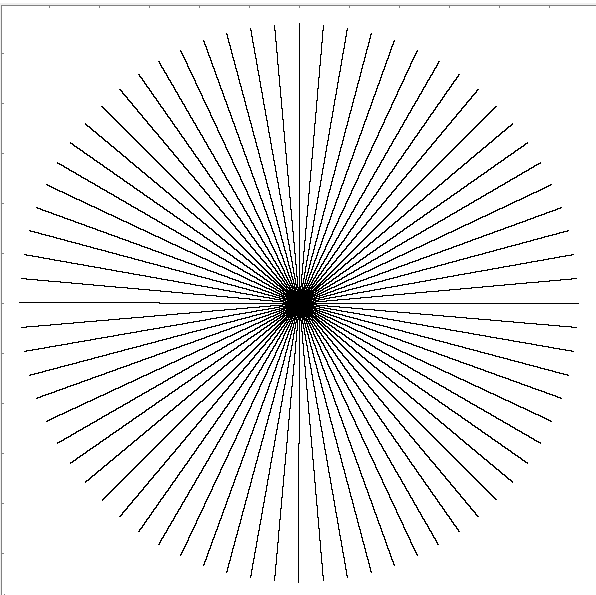
e -= w

1. *Результат*

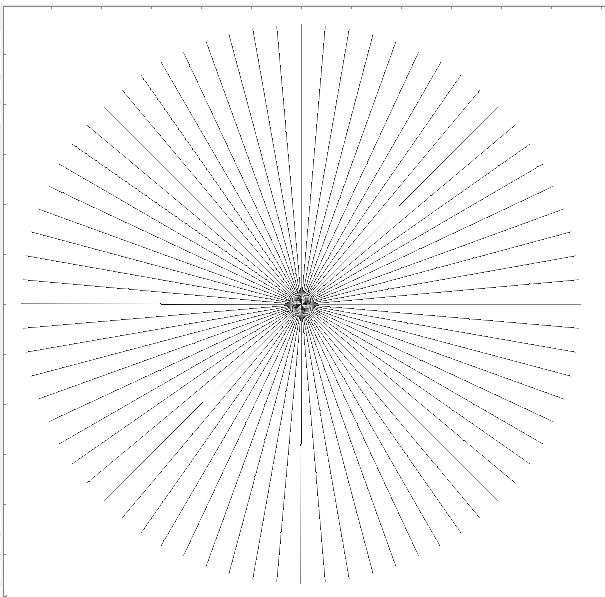
**Алгоритм Брезенхэма (действ.)**



**Алгоритм Брезенхэма (цел.)**



Алгоритм Брезенхэма с устранением ступенчатости



**IV. Алгоритм Ву:**

Алгоритм Wu, хотя и медленнее, чем алгоритм Брезенхэма, исключает наложение имен и может быть нарисован, даже если начальная и конечная координаты являются действительными числами.

1. *Идея алгоритма:*

Интенсивность высв. Писеля зависит от расстояния до отрезка. Чем ближе писель расположен к точке идеального отрезка, тем с большей интенсивностью он должен высвечивается.

* 1. На каждом шаге высвечивается 2 соседных писеля.
  2. Расстояние от точки идеального до нижнего писеля равно дробной части ординаты пиксель из отрезки.

d1 = Yui - ]Yui[

d2 = 1 – d1 = 1 – Yui + ]Yui[

1. *Код программы:*

// целая часть

def ipart(x):

return int(x)

// дробная часть

def fpart(x):

return x - int(x)

def rfpart(x):

return 1 - fpart(x)

def line\_Wu(x1, y1, x2, y2):

I = 100 // макс. интенсивностью

colors = get\_rgb\_intensity(line\_color, I)

if x1 == x2 and y1 == y2:

img.put(line\_color, (int(x1),int(y1)))

print(x1, y1, x2, y2)

steep = abs(y2 - y1) > abs(x2 - x1)

dx = x2 - x1

dy = y2 - y1

if (abs(dx) < abs(dy)):

x1, y1 = y1, x1

x2, y2 = y2, x2

dx, dy = dy, dx

if x1 > x2:

x1, x2 = x2, x1

y1, y2 = y2, y1

if dx == 0:

m = 1

else:

m = dy / dx

# first endpoint

xend = round(x1)

yend = y1 + m \* (xend - x1)

xgap = rfpart(x1 + 0.5)

xpxl1 = xend

ypxl1 = ipart(yend)

I1 = int((I - 1) \* (rfpart(yend) \* xgap))

img.put(colors[I1], (int(xpxl1),int(ypxl1)))

I1 = int((I - 1) \* (fpart(yend) \* xgap))

img.put(colors[I1], (int(xpxl1),int(ypxl1 + 1)))

y = yend + m

# second endpoint

xend = int(x2 + 0.5)

yend = y2 + m \* (xend - x2)

xgap = fpart(x2 + 0.5)

xpxl2 = xend

ypxl2 = ipart(yend)

I1 = int((I - 1) \* rfpart (yend) \* xgap)

img.put(colors[I1], (int(xpxl2),int(ypxl2)))

I1 = int((I - 1) \* fpart (yend) \* xgap)

img.put(colors[I1], (int(xpxl2),int(ypxl2 + 1)))

#main loop

for x in range(xpxl1, xpxl2):

if (steep == 0):

I1 = int((I - 1) \* rfpart(y))

img.put(colors[I1], (int(x),int(ipart(y))))

I2 = int((I - 1) \* fpart(y))

img.put(colors[I2], (int(x),int(ipart(y + 1))))

else:

I1 = int((I - 1) \* rfpart(y))

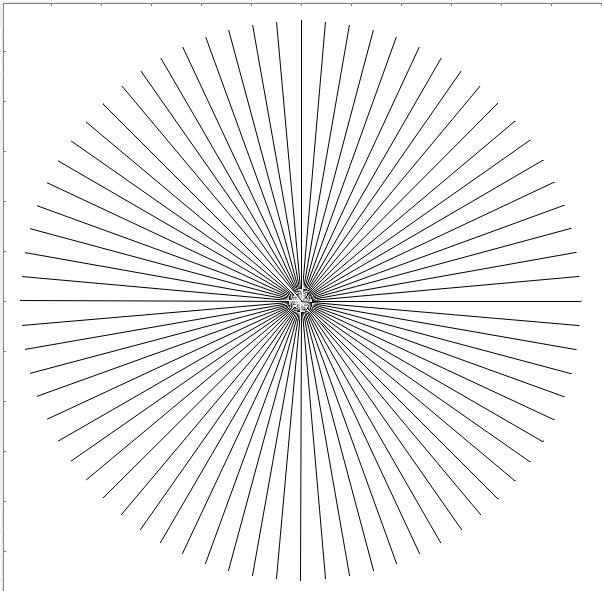
img.put(colors[I1], (int(ipart(y)), int(x)))

I2 = int((I - 1) \* fpart(y))

img.put(colors[I2], (int(ipart(y + 1)), int(x)))

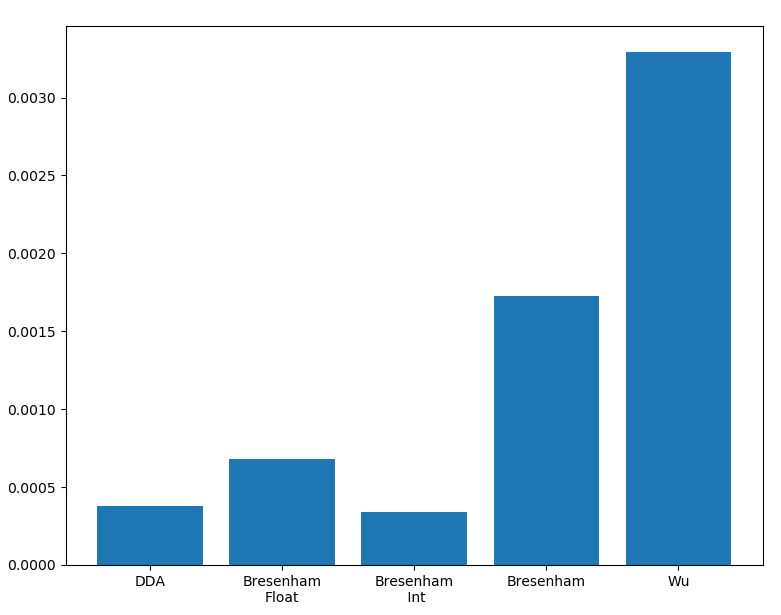
y += m

1. Результат



**V.Исследование:**

Время:



Анализ ступечатости

