**МГТУ им. Н.Э. Баумана**

**КОМПЬЮТЕРНАЯ ГРАФИКА**

**Лабораторный практикум №4**

**по теме: «*Построение окружности и эллипса*»**

***Студент: Нгуен Фыок Санг***

***Группa: ИУ7И-46Б***

***Работу проверил:***

2020

Цель работы: реализация алгоритмов построения окружности, исследование и сравнение визуальных и временных характеристик алгоритмов.

1. **Реализовать алгоритмы построения окружности на основе** 
   1. Канонического уравнени
   2. Параметрического уравнения
   3. Алгоритма Брезенхема
   4. Алгоритма средней точки
   5. Построение окружности с помощью библиотечной функции

Пользователь выбирает из списка определенный алгоритм, задает координаты центра, радиус, цвет рисования.

Визуальные характеристики исследуются путем рисования той же окружности цветом фона, но с помощью другого алгоритма.

1. **Реализовать алгоритмы построения эллипса на основе** 
   1. Канонического уравнения
   2. Параметрического уравнения
   3. Алгоритма Брезенхема (модифицировать самостоятельно)
   4. Алгоритма средней точки
   5. Построение эллипса с помощью библиотечной функции

Пользователь выбирает из списка определенный алгоритм, задает координаты центра, полуоси, цвет рисования.

Визуальные характеристики исследуются путем рисования того же эллипса цветом фона, но с помощью другого алгоритма.

1. **Реализовать алгоритмы построения окружности на основе**

Вход: радиус R, центр C (xC, yC)

* 1. ***Канонического уравнения***
     1. Идея алгоритма:

Разделите четвертую часть на 2 части по точке M (dx / dy = -1):

* 0 < x ≤ xM: смещение на х равно 1(x = x + 1)
* xM < x ≤ R: смещение на y равно 1(y = y - 1)

Начальное условие:

* x = 0, y = R
  + 1. Код программы:

def circle\_canon(xc, yc, R):

r2 = R \* R

x\_M = round(R / sqrt(2))

x = 0

while x <= x\_M:

y = round(sqrt(r2 - x \* x))

plot\_4\_point(x, y, xc, yc)

x += 1

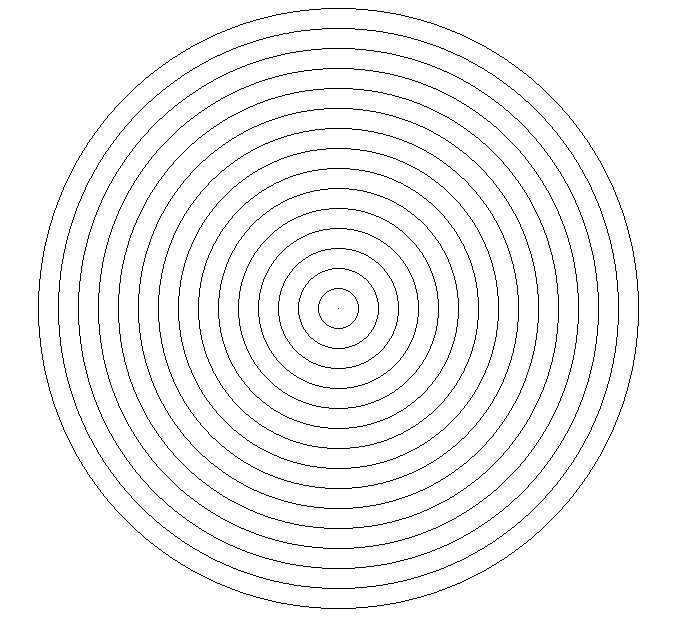
y = round(sqrt(r2 - x\_M \* x\_M))

while (y >= 0):

x = round(sqrt(r2 - y \* y))

plot\_4\_point(x, y, xc, yc)

y -= 1



* 1. ***Параметрического уравнения***
     1. Идея алгоритма:
     2. Код программы

def circle\_param(xc, yc, R):

dt = 1 / R

m = pi / 2 \* 1.01

alpha = 0

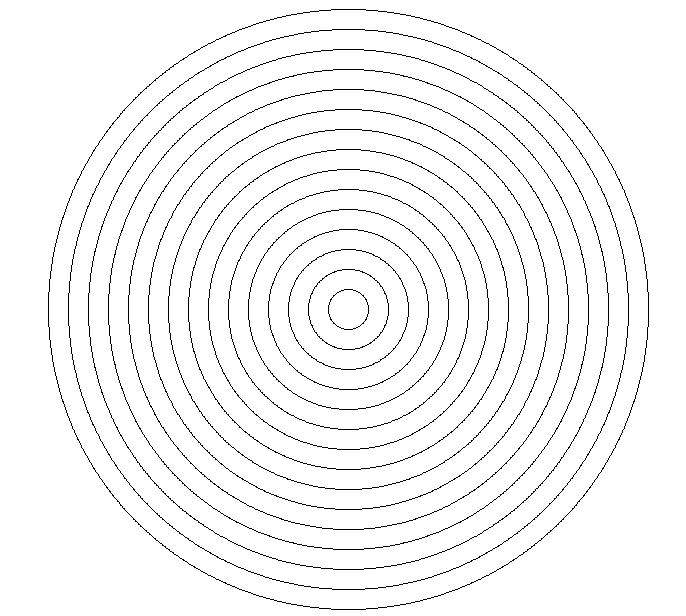
while alpha <= m:

x = round(R \* cos(alpha))

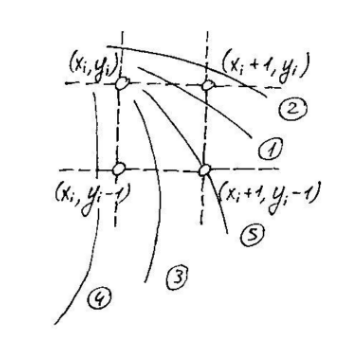
y = round(R \* sin(alpha))

plot\_4\_point(x, y, xc, yc)

alpha += dt



* 1. ***Алгоритма Брезенхема:***
     1. Идея алгоритма:



* Δi < 0: Диагональный пиксель находится внутри ок­ружности

=> необходимо выбрать диагональный или горизонтальный пиксель (случаи 1 и 2)

* Δi = 0: Диагональный пиксель лежит на окружности (случай 5)
* Δi > 0: Диагональный пиксель находится внеш. ок­ружности

=> необходимо выбрать диагональный пиксель или вертикальный пиксель (случаи 3 и 4)

* **Δi < 0:**
* Случай 1:

* Случай 2:

< 0

* **Δi > 0**
* Случай 3:

* Случай 4:

> 0

* **Δi = 0**
* Случай 5:
* Горизонтальный шаг:
* Вертикальный шаг:
* Диагональный шаг:

Начальное условие:

x = 0

y = R

* + 1. Код программы

def circle\_bresenham(xc, yc, R):

x = 0

y = R

deli = 2 \* (1 - R)

while (y >= 0):

plot\_4\_point(x, y, xc, yc)

if (deli < 0):

err1 = 2 \* deli + 2 \* y - 1

if (err1 <= 0): #gorizontal

x += 1

deli += 2 \* x + 1

else: #diag

x += 1

y -= 1

deli += 2 \* (x - y + 1)

elif (deli > 0):

err2 = 2 \* deli - 2 \* x - 1

if (err2 <=0): # diag

x += 1

y -= 1

deli += 2 \* (x - y + 1)

else: # vertical

y -= 1

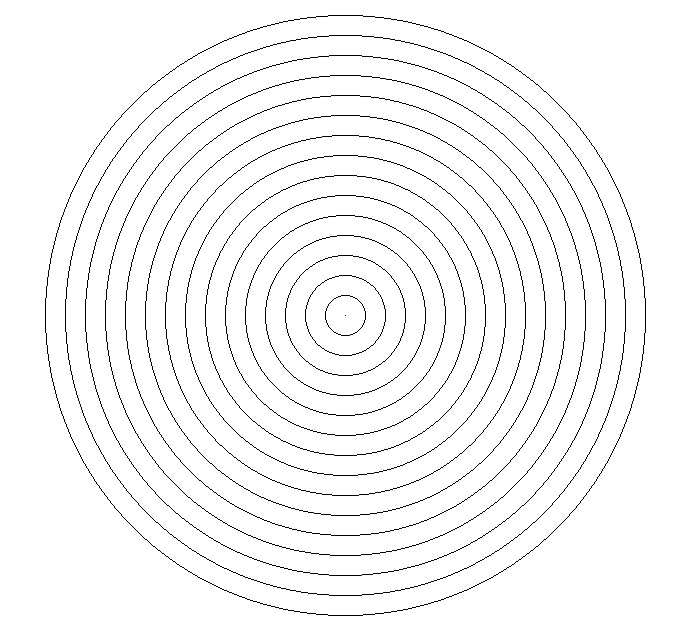
deli += 1 - 2 \* y

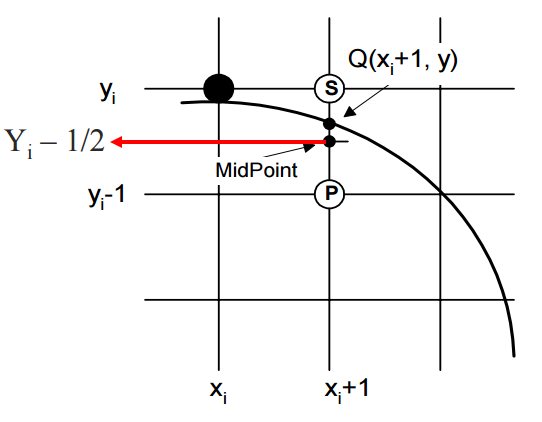
else: #diag

x += 1

y -= 1

deli += 2 \* (x - y + 1)



* 1. ***Алгоритма средней точки***
     1. Идея алгоритма:

:

* + 1. Код программы

def circle\_midpoint(xc, yc, R):

x = 0

y = R

plot\_4\_point(x, y, xc, yc)

plot\_4\_point(y, x, xc, yc)

f = int(5/4 - R) # f = (0 + 1)\*\*2 + (R - 1/2) \*\*2 - R\*\*2 = -R + 5/4

while (x < y):

x += 1

if (f < 0): # gorizontal

f += 2 \* x + 1

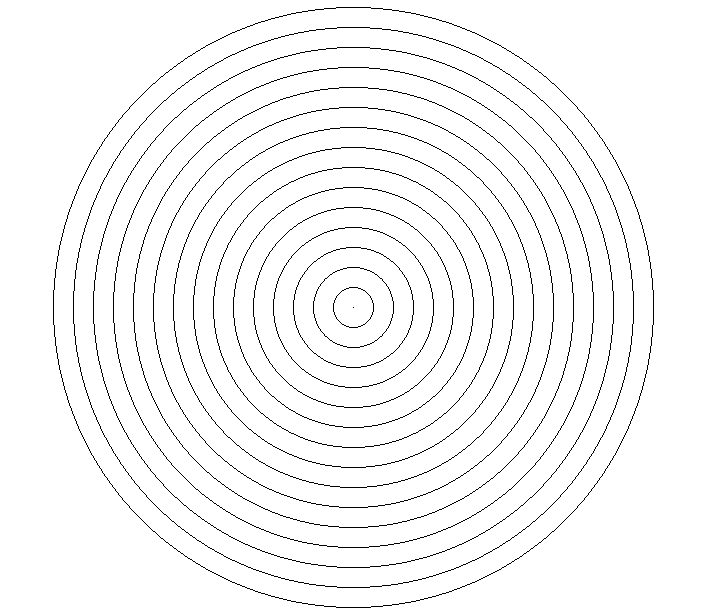
else:

y -= 1

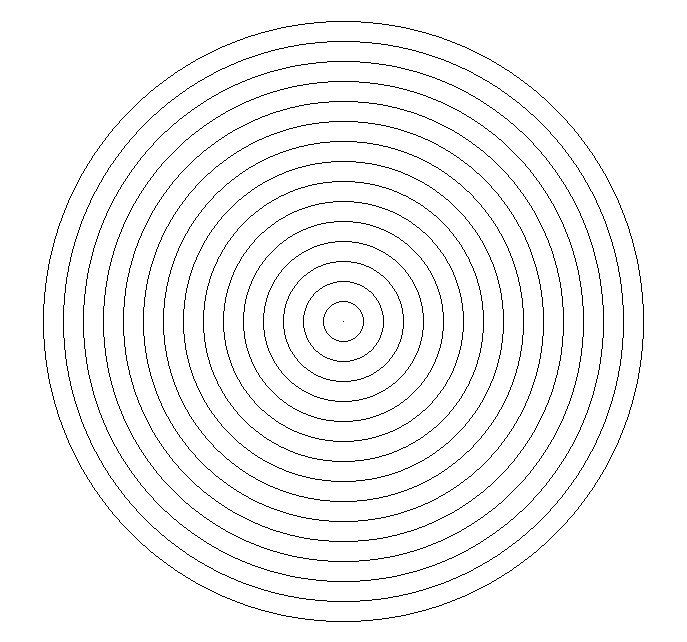
f += 2 \* (x - y) + 1

plot\_4\_point(x, y, xc, yc)

plot\_4\_point(y, x, xc, yc)



* 1. Построение окружности с помощью библиотечной функции



1. **Реализовать алгоритмы построения эллипса на основе** 
   1. ***Канонического уравнения:***
      1. Идея алгоритма:

Разделите четвертую часть на 2 части по точке M (dx / dy = -1):

* 0 < x ≤ xM: смещение на х равно 1(x = x + 1)
* xM < x ≤ a: смещение на y равно 1(y = y - 1)

Начальное условие:

* x = 0
* y = b
  + 1. Код программы:

def elip\_canon(xc, yc, a, b):

a2 = a \* a

b2 = b \* b

c = b / a

x\_C = round(a2 / sqrt(a2 + b2))

x = 0

while x <= x\_C:

y = round(c \* sqrt(a2 - x\*x))

plot\_4\_point(x, y, xc, yc)

x += 1

y = round(c \* sqrt(a2 - x\_C \* x\_C))

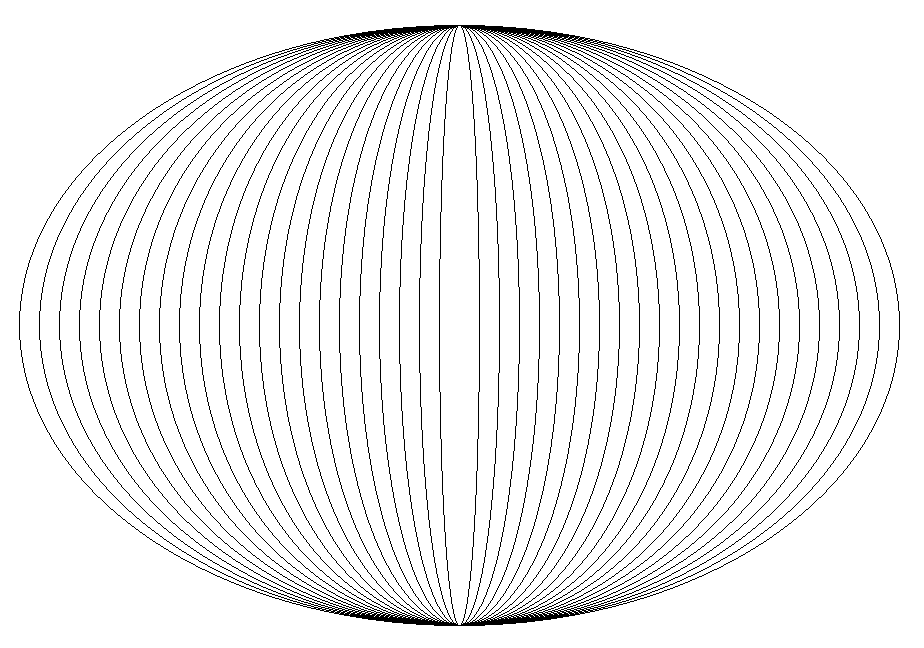
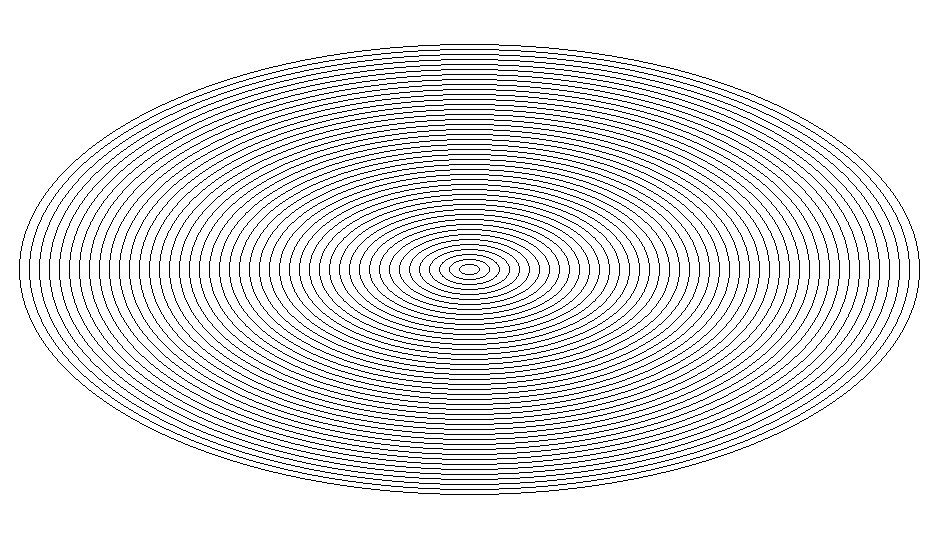
c = a / b

while (y >= 0):

x = round(c \* sqrt(b2 - y \* y))

plot\_4\_point(x, y, xc, yc)

y -= 1



* 1. ***Параметрического уравнения***
     1. Идея алгоритма:
* dx = 1 / a
* dy = 1 / b

Разделите четвертую часть на 2 части по точке M (dx / dy = -1):

* 0 < x ≤ xM: t = t - dx
* xM < x ≤ a: t = t – dy
  + 1. Код программы

def elip\_param(xc, yc, a, b):

alpha = pi / 2

dx = 1 / a

dy = 1 / b

a2 = a \* a

b2 = b \* b

t\_x\_C = acos(a2 / (a2 + b2))

while alpha >= t\_x\_C:

x = round(a \* cos(alpha))

y = round(b \* sin(alpha))

plot\_4\_point(x, y, xc, yc)

alpha -= dx

alpha = t\_x\_C

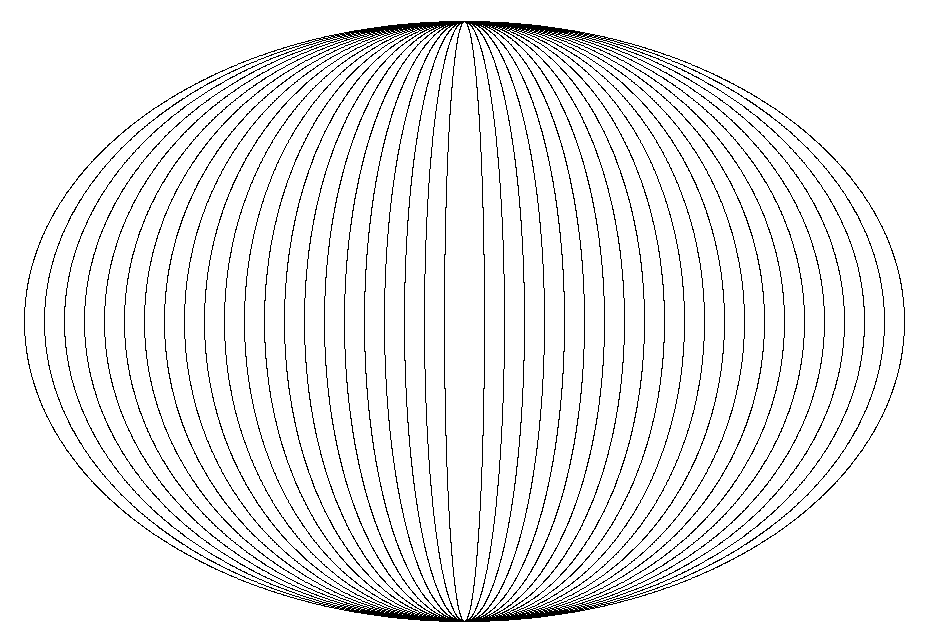
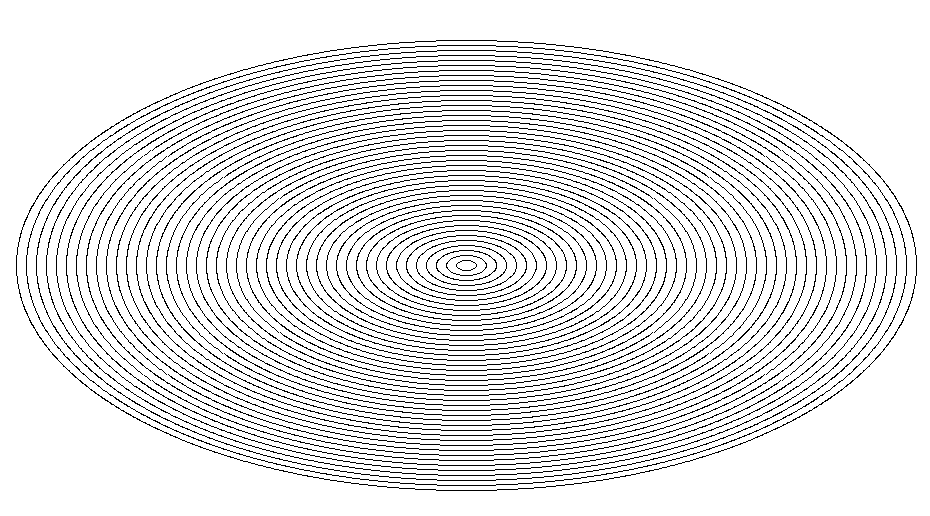
while alpha >= -1:

x = round(a \* cos(alpha))

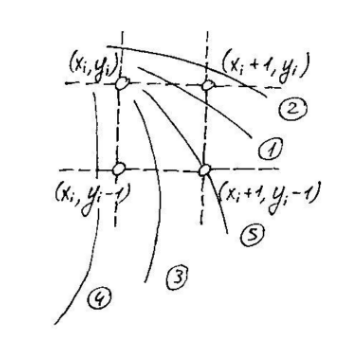
y = round(b \* sin(alpha))

plot\_4\_point(x, y, xc, yc)

alpha -= dy



* 1. ***Алгоритма Брезенхема:***
     1. Идея алгоритма:



* Δi < 0: Диагональный пиксель находится внутри ок­ружности

=>необходимо выбрать диагональный или горизонтальный пиксель (случаи 1 и 2)

* Δi = 0: Диагональный пиксель лежит на окружности (случай 5)
* Δi > 0: Диагональный пиксель находится внеш. ок­ружности

=>необходимо выбрать диагональный пиксель или вертикальный пиксель (случаи 3 и 4)

* **Δi < 0:**
* Случай 1:

* Случай 2:

* **Δi > 0**
* Случай 3:

* Случай 4:

* **Δi = 0**
* Случай 5:
* Горизонтальный шаг:
* Вертикальный шаг:
* Диагональный шаг:

Начальное условие:

x = 0

y = b

* + 1. Код программы:

def elip\_bresenham(xc, yc, a, b):

x = 0

y = b

b2 = b \* b

a2 = a \* a

bd = 2 \* b2

ad = 2 \* a2

deli = b2 + a2 -ad \* b

while (y >= 0):

plot\_4\_point(x, y, xc, yc)

if (deli < 0):

err1 = 2 \* deli + ad \* y - a2

if (err1 < 0): # gorizontal

x += 1

deli += bd \* x + b2

else: #diag

x += 1

y -= 1

deli += bd \* x + b2 + a2 - ad \* y

elif (deli > 0):

err2 = 2 \* deli - bd \* x - b2

if (err2 <= 0): #diag

y -= 1

x += 1

deli += bd \* x + b2 + a2 - ad \* y

else: # vertical

y -= 1

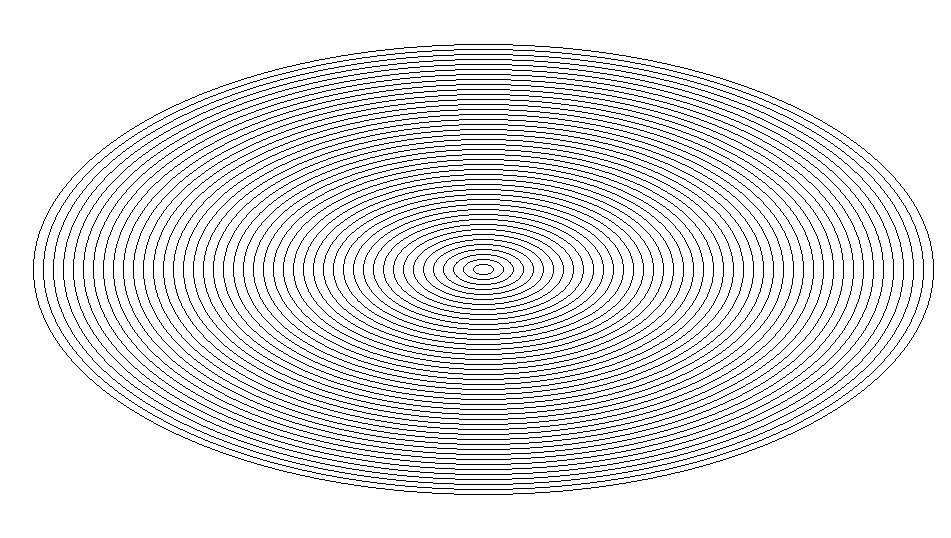
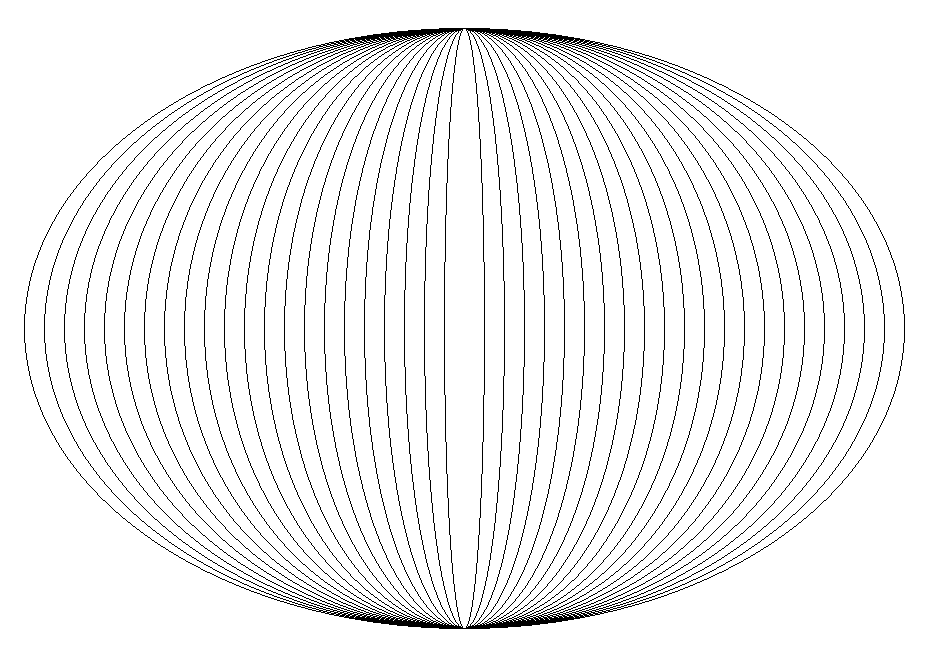
deli += a2 - ad \* y

else:

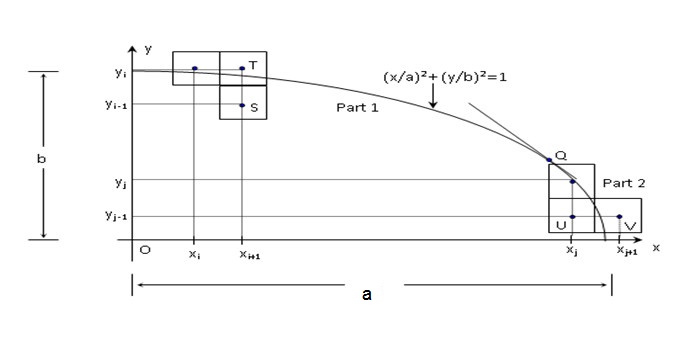
y -= 1

x += 1

deli += bd \* x + b2 + a2 - ad \* y



* 1. ***Алгоритма средней точки***
     1. Идея алгоритма:



Разделите четвертую часть на 2 части по точке M (dx / dy = -1):

* 0 < x ≤ xM: смещение на х равно 1(x = x + 1)
* xM < x ≤ a: смещение на y равно 1(y = y - 1)

**В первой части:**

df

Если выбрана точка S, значение функции должно быть скорректировано:

Во второй части:

dy = -ad \* y

* + 1. Код программы:

def elip\_midpoint(xc, yc, a, b):

b2 = b\*b

a2 = a\*a

bd = 2 \* b2

ad = 2 \* a2

x = 0

y = b

dx = 0

f = b2 - a2 \* y + a2 / 4 # = (b2 \* (0 + 1)\*\*2 + a2 \* (b - 1/2)\*\*2 - a2\*b2

delf = -ad \* y

x\_C = int(a2 / sqrt(a2 + b2))

y\_C = int(b2 / sqrt(a2 + b2))

# В первой части

while (x <= x\_C):

plot\_4\_point(x, y, xc, yc)

if (f > 0): # выбрана точка S

y -= 1

delf += ad

f += delf

dx += bd

x += 1

f += dx + b2

f += 0.75 \* (a2 - b2) - (b2 \* x + a2 \* y)

delf = bd \* x

dy = -ad \* y

# Во второй части

while y >= 0:

plot\_4\_point(x, y, xc, yc)

if (f < 0):

x += 1

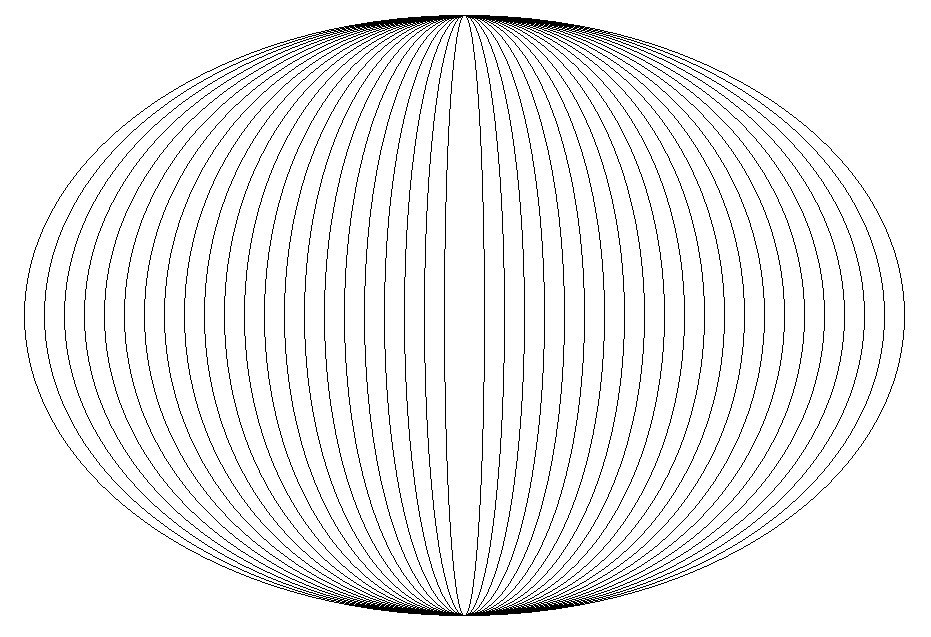
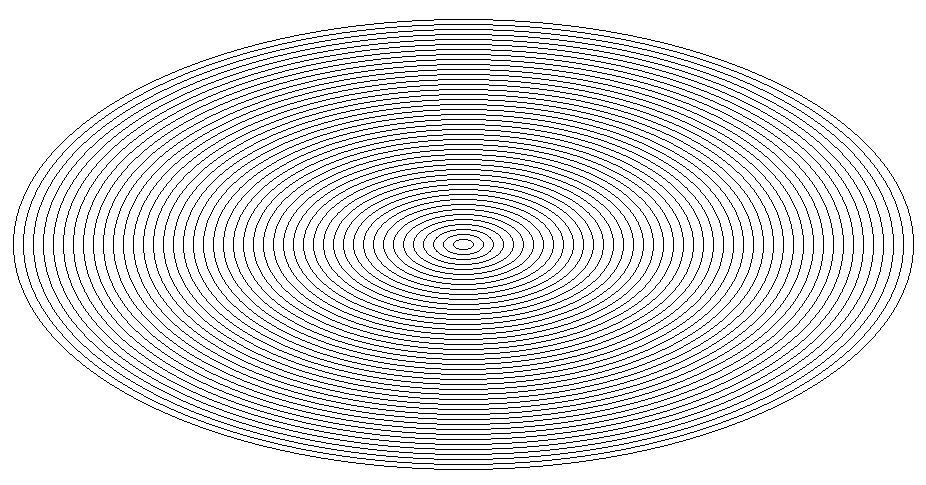
delf += bd

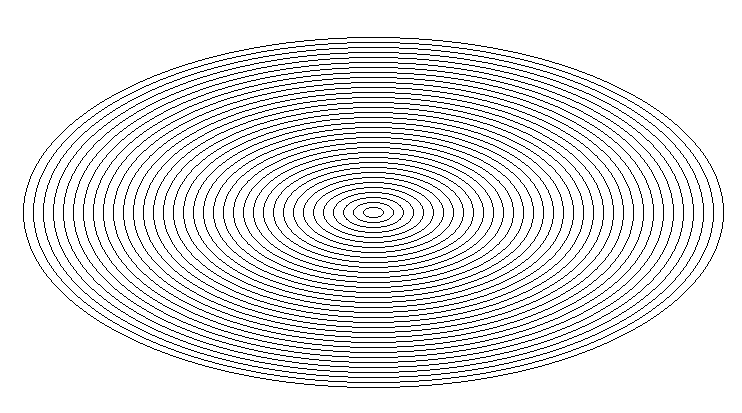
f += delf

dy += ad

f += dy + a2

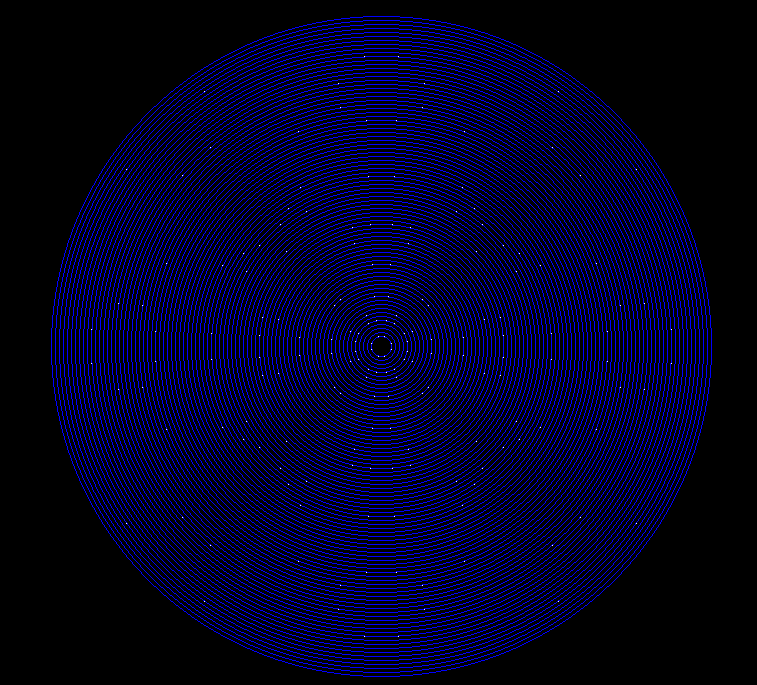
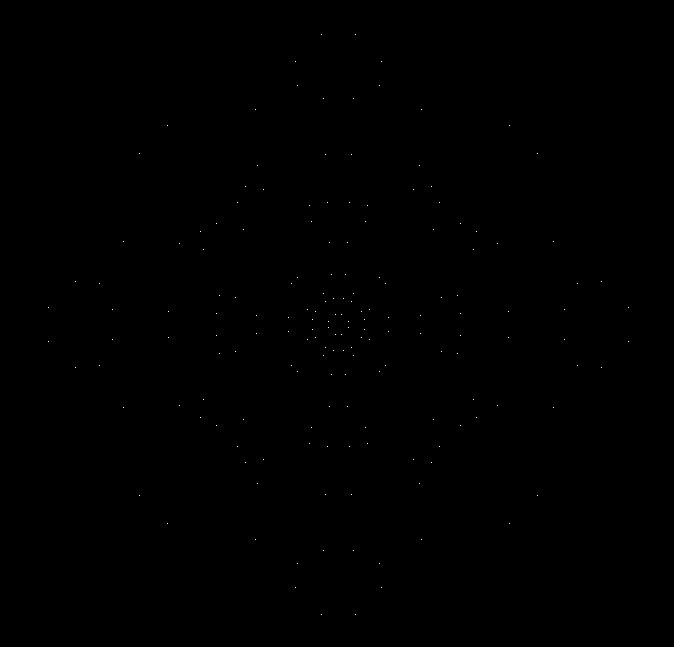
y -= 1

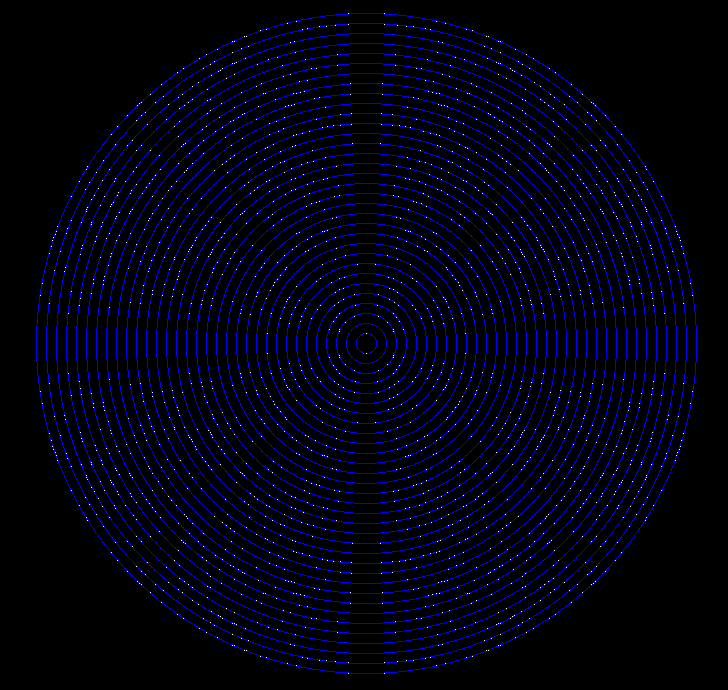


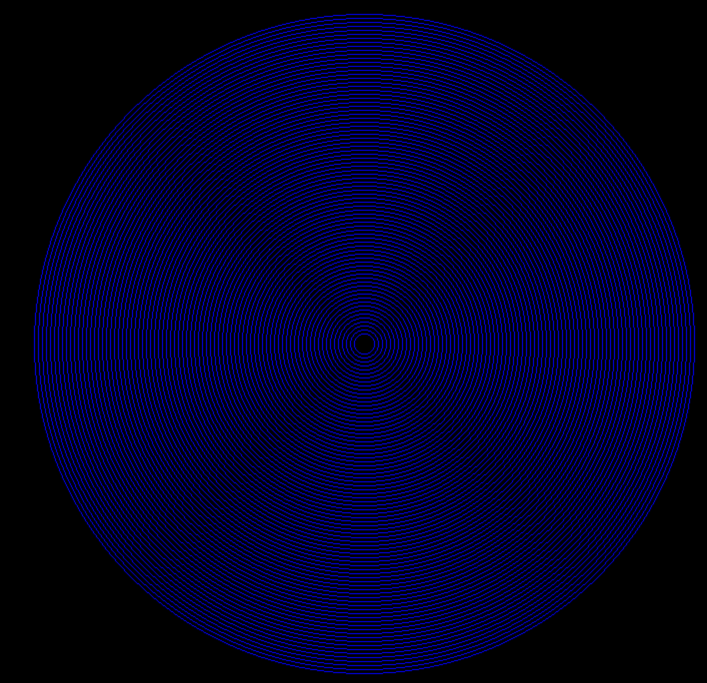
* 1. Построение эллипса с помощью библиотечной функции

1. Сравнение визуальных характеристик разных алгоритмов при рисовании спектра концентрических окружностей.
   1. Канонического уравнени
   2. Параметрического уравнения
   3. Алгоритма Брезенхема
   4. Алгоритма средней точки

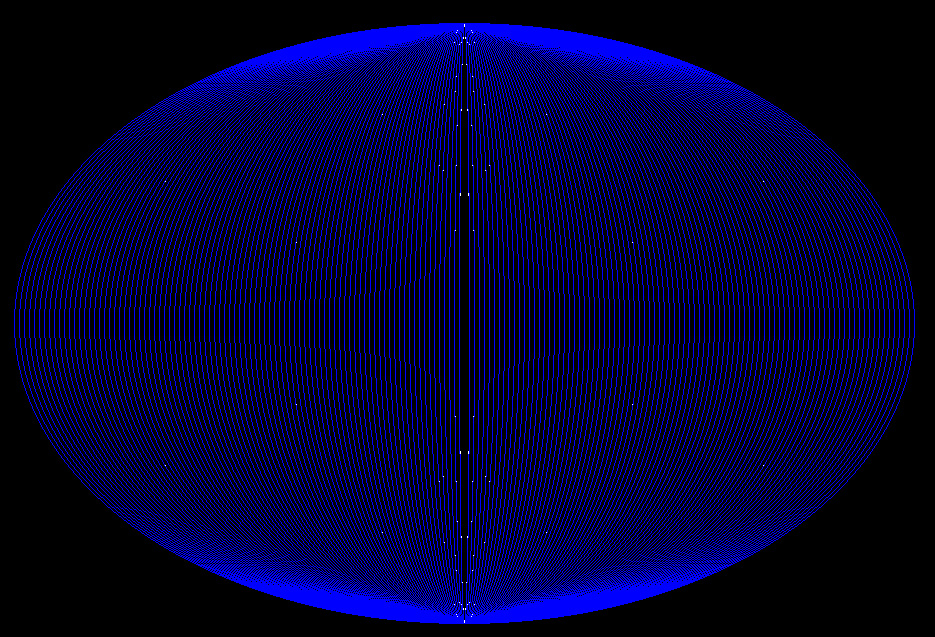
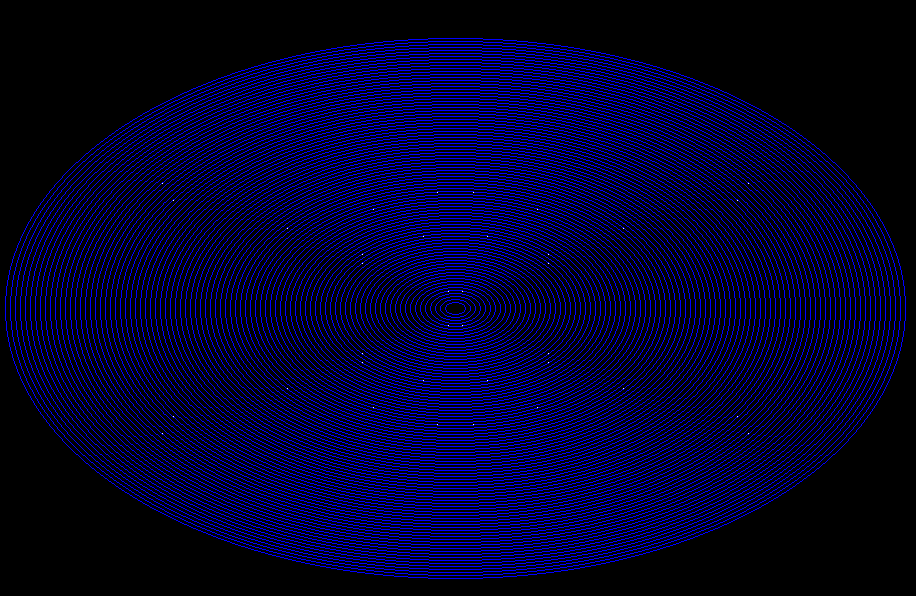
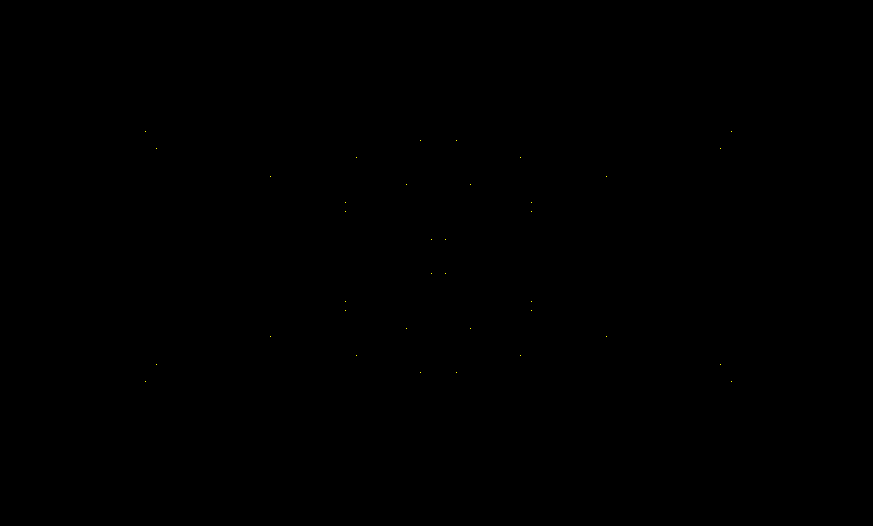
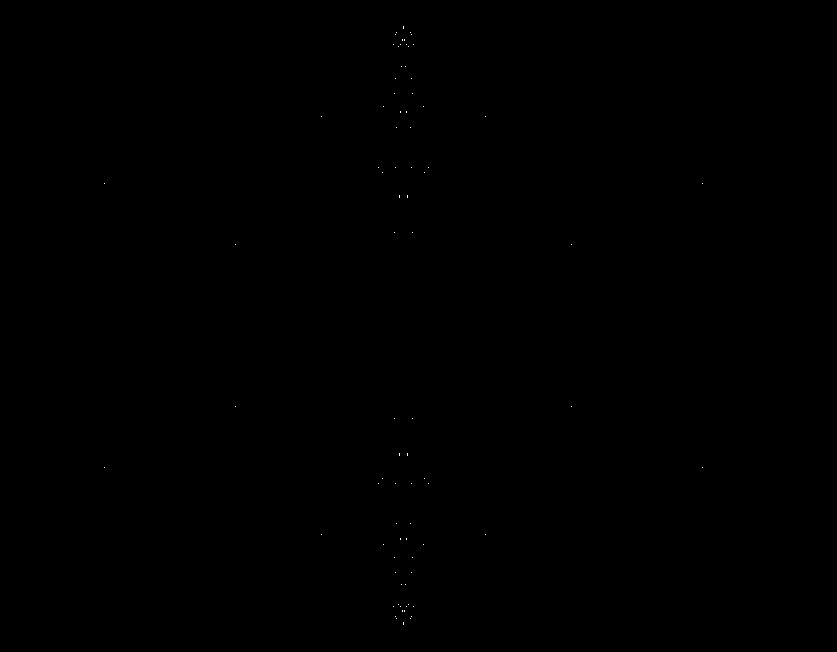
Алгоритма Брезенхема( Белая ) - Алгоритма средней точки ( Синяя - Черная )



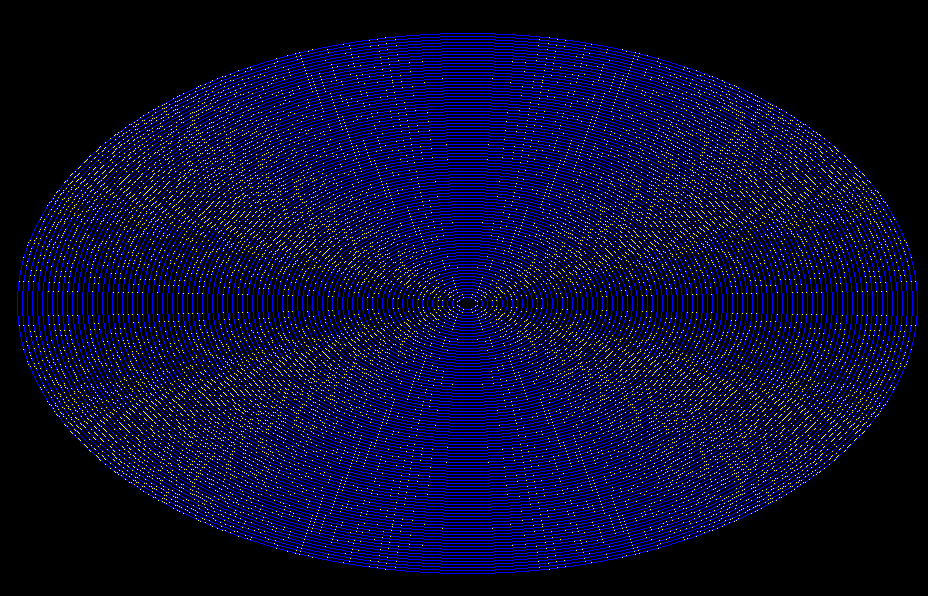
Параметрического уравнения ( Белая ) - Канонического уравнени ( Синяя - Черная )

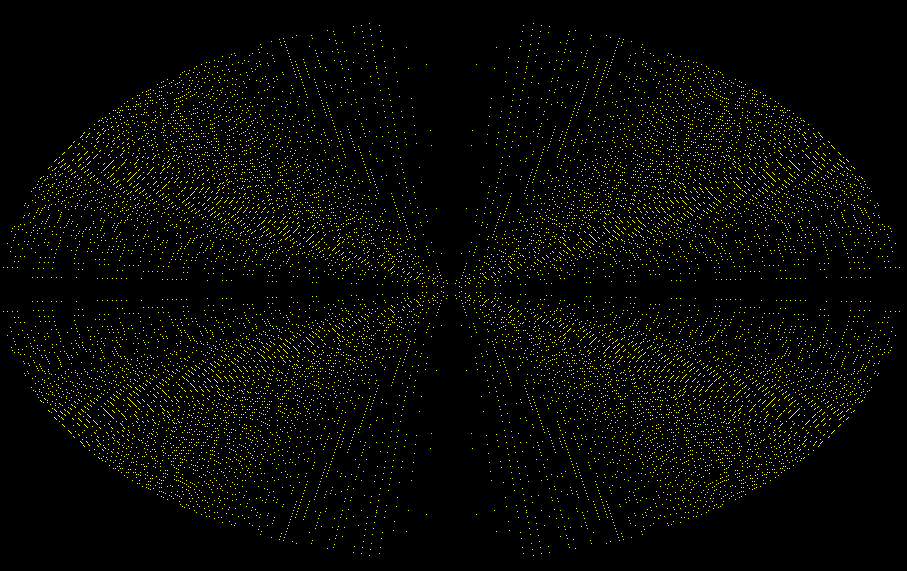
Алгоритма Брезенхема ( Белая ) - Канонического уравнени ( Синяя )

1. Сравнение визуальных характеристик разных алгоритмов при рисовании спектра концентрических эллипсов.

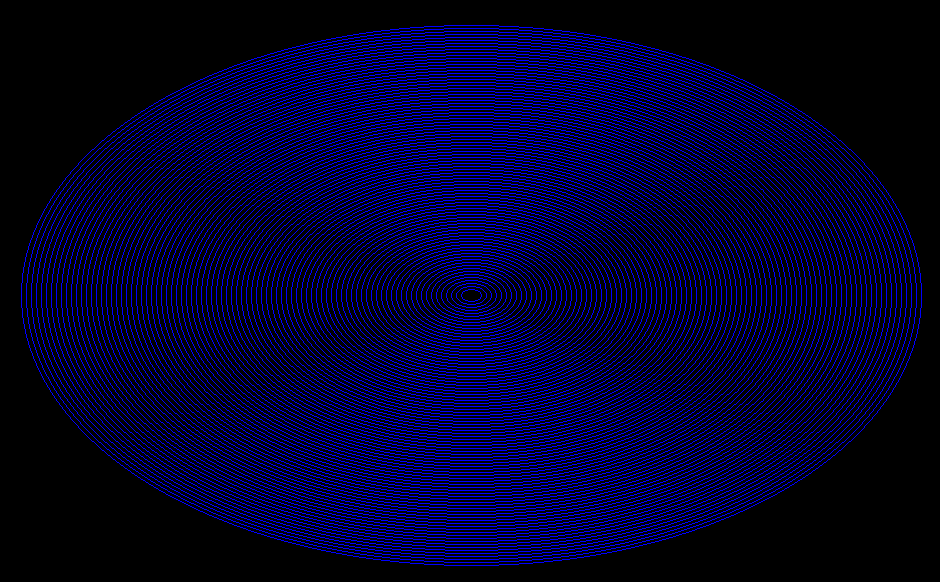
Алгоритма Брезенхема( Белая ) - Алгоритма средней точки ( Синяя - Черная ) 

Параметрического уравнения ( Белая ) - Канонического уравнени ( Синяя - Черная )

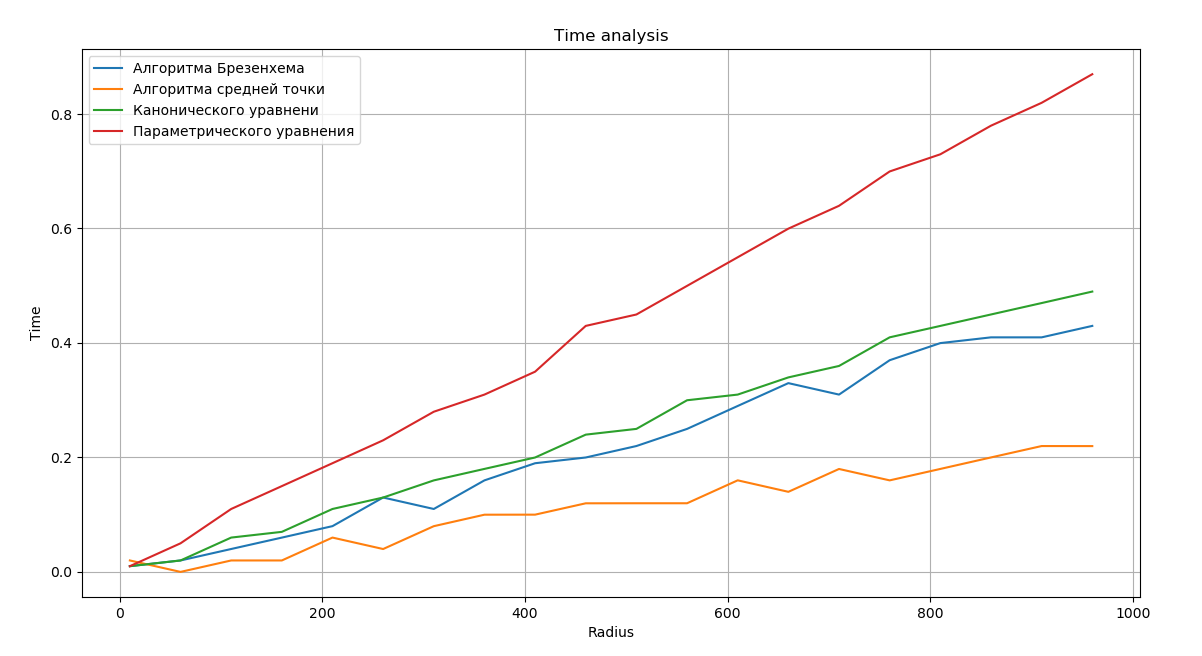




Алгоритма средней точки ( Белая ) - Канонического уравнени ( Синяя )



Сравнить временные характеристики разных алгоритмов окружности:



Сравнить временные характеристики разных алгоритмов (b = 0..500)

