

Thanaporn Yankomut 60070501018

Phuraefa Rattanatakun 60070501045

Wantanee Saetear 60070501053

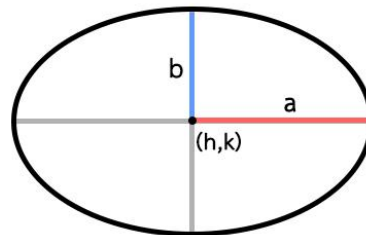
Parattha Weerapong 60070501080

---

## Lab 5 Ellipse Drawing Algorithms

Ellipse Equation is

$$\frac{(x - h)^2}{a^2} + \frac{(y - k)^2}{b^2} = 1$$



And An ellipse has two axes of symmetry. So, we will find only point in quadrant 1 and mirror it to complete ellipse.

quadrant 1 (x, y)

quadrant 2 (-x, y) [mirror by y-axis]

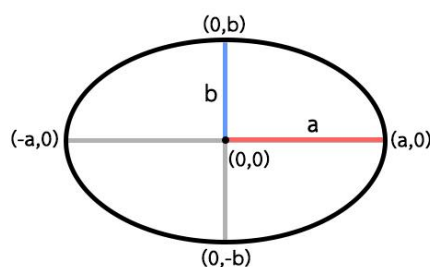
quadrant 3 (-x, -y) [mirror by x-axis]

quadrant 4 (x, -y) [mirror by x-axis and y-axis]

- 
1. The definition and the proof of ellipse drawing algorithms. The students can use the idea from the midpoint algorithm concepts from two papers, those of Carpenter and Kennedy.

Assume the ellipse's center is at the origin (0,0). there for the equation is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$



## Kennedy Algorithm

In Kennedy Algorithm. from

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Multiply the equation with  $a^2b^2$

$$x^2b^2 + y^2a^2 = a^2b^2$$
$$f(x, y) = x^2b^2 + y^2a^2 - a^2b^2$$

If  $f(x, y) = 0$ , then the point  $(x, y)$  is exactly on the ellipse

If  $f(x, y) < 0$ , then the point  $(x, y)$  is inside the ellipse

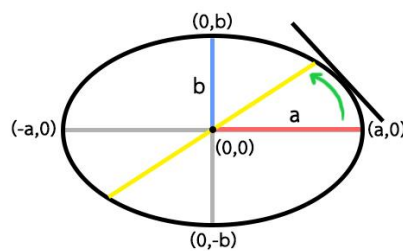
And if  $f(x, y) > 0$ , then the point  $(x, y)$  is outside the ellipse

And value of  $|f(x, y)|$  is for measure the error

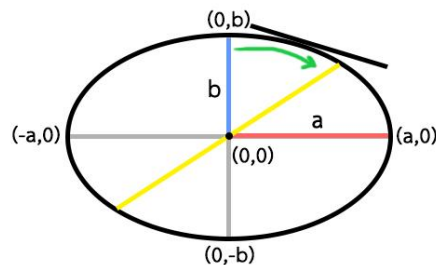
$$Error(x, y) = |x^2b^2 + y^2a^2 - a^2b^2|$$

We will calculate 2 set of point in quadrant 1.

The first set, slope of tangent line is less than -1. Starting render from  $(a, 0)$ . and render in counterclockwise direction until the slope of tangent line reach -1.



The second set, slope of tangent line is greater than -1. Starting render from  $(0, b)$ . and render in clockwise direction until the slope of tangent line reach -1.



For calculate where to tangent line is equals -1. From the equation

$$x^2b^2 + y^2a^2 = a^2b^2$$

Apply  $\frac{dy}{dx}$  to the equation

$$\begin{aligned} 2xb^2 + 2ya^2 \frac{dy}{dx} &= 0 \\ \frac{dy}{dx} &= -\frac{2xb^2}{2ya^2} \\ \text{slope} &= -\frac{2xb^2}{2ya^2} \end{aligned}$$

For the first set that slope less than -1, it will stop when it reaches the point that slope equals -1. So, it will stop when

$$\begin{aligned} \text{slope} &\geq -1 \\ -\frac{2xb^2}{2ya^2} &\geq -1 \\ 2xb^2 &\geq 2ya^2 \end{aligned}$$

For the next point the value will change.  $2xb^2$  is decrease by  $2b^2$  and  $2ya^2$  is increase by  $2a^2$  because x and y value are change.

And for the second set that the slope greater than -1, it will stop when it reaches the point that slope equals -1. So, it will stop when

$$\begin{aligned} \text{slope} &< -1 \\ -\frac{2xb^2}{2ya^2} &< -1 \\ 2xb^2 &< 2ya^2 \end{aligned}$$

For the next point the value will change.  $2xb^2$  is increase by  $2b^2$  and  $2ya^2$  is decrease by  $2a^2$  because x and y value are change.

When plotting the next point.

For first set, tangent slope is less than -1. Which means y is change faster than x. so, y is always increase by 1 and x we have to choose between x or x - 1.

$$\begin{aligned} \text{Error}(x-1, y+1) &= |(x-1)^2 b^2 + (y+1)^2 a^2 - a^2 b^2| \\ \text{Error}(x-1, y+1) &= |x^2 b^2 - 2xb^2 + b^2 + y^2 a^2 + 2ya^2 + a^2 - a^2 b^2| \end{aligned}$$

and

$$\begin{aligned} \text{Error}(x, y+1) &= |x^2 b^2 + (y+1)^2 a^2 - a^2 b^2| \\ \text{Error}(x, y+1) &= |x^2 b^2 + y^2 a^2 + 2ya^2 + a^2 - a^2 b^2| \end{aligned}$$

To decision we will compare these 2 values and choose the less one.

So, we will choose x - 1 when

$$\text{Error}(x-1, y+1) < \text{Error}(x, y+1)$$

and

$$\text{Error}(x-1, y+1) + \text{Error}(x, y+1) > 0 \leftrightarrow \text{Error}(x-1, y+1) < \text{Error}(x, y+1)$$

Therefore, we plot (x-1, y+1) at the next point when

$$\begin{aligned} \text{Error}(x-1, y+1) + \text{Error}(x, y+1) &> 0 \\ 2[b^2 + y^2 a^2 + 2ya^2 + a^2 - a^2 b^2] - 2xb^2 + b^2 &> 0 \end{aligned}$$

For second set, tangent slope is greater than -1. Which means x is change faster than y. so, x is always increase by 1 and y we have to choose between y or y - 1.

$$\begin{aligned} \text{Error}(x+1, y-1) &= |(x+1)^2 b^2 + (y-1)^2 a^2 - a^2 b^2| \\ \text{Error}(x+1, y-1) &= |x^2 b^2 - 2xb^2 + b^2 - y^2 a^2 + 2ya^2 + a^2 - a^2 b^2| \end{aligned}$$

and

$$\begin{aligned} \text{Error}(x+1, y) &= |(x+1)^2 b^2 + y^2 a^2 - a^2 b^2| \\ \text{Error}(x+1, y) &= |x^2 b^2 + 2xb^2 + b^2 + y^2 a^2 - a^2 b^2| \end{aligned}$$

To decision we will compare these 2 values and choose the less one.

So, we will choose y- 1 when

$$\text{Error}(x+1, y-1) < \text{Error}(x+1, y)$$

and

$$\text{Error}(x+1, y-1) + \text{Error}(x+1, y) > 0 \leftrightarrow \text{Error}(x+1, y-1) < \text{Error}(x+1, y)$$

Therefore, we plot (x+1, y-1) at the next point when

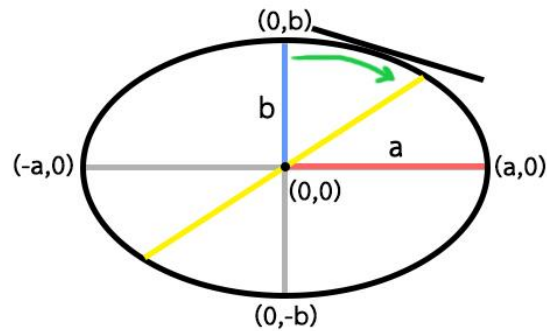
$$\begin{aligned} \text{Error}(x+1, y-1) + \text{Error}(x+1, y) &> 0 \\ 2[b^2 + b^2 x^2 + 2xb^2 + a^2 y^2 - a^2 b^2] - 2ay^2 + a^2 &> 0 \end{aligned}$$

## Carpenter Algorithm

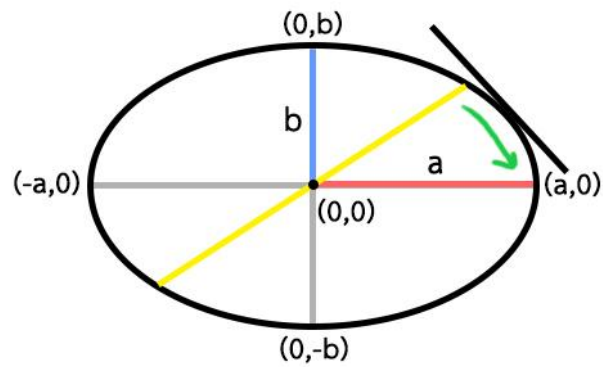
In Carpenter Algorithm.

We will calculate 2 set of point in quadrant 1.

For the first set, slope of tangent line is greater than -1. Starting render from  $(0, b)$ . and render in clockwise direction until the slope of tangent line reach -1.



For the second set, continue render in clockwise direction untill  $y = 0$ .



For calculate where to tangent line is equals -1. From the equation

$$f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1$$

And

$$\frac{dy}{dx} = -\frac{\frac{\partial f}{\partial x}}{\frac{\partial f}{\partial y}}$$

then

$$\begin{aligned}\frac{dy}{dx} &= -\frac{\frac{2x}{a^2}}{\frac{2y}{b^2}} \\ \frac{dy}{dx} &= -\frac{b^2 x}{a^2 y} \\ \text{slope} &= -\frac{b^2 x}{a^2 y}\end{aligned}$$

For the first set that slope less than -1, it will stop when it reaches the point that slope equals -1. So, it will stop when

$$\begin{aligned}\text{slope} &\geq -1 \\ -\frac{b^2 x}{a^2 y} &\geq -1 \\ b^2 x &< a^2 y\end{aligned}$$

And initialize value for (1, b – 0.5)

$$b^2 < a^2 b - \frac{a^2}{2}$$

When plotting the next point.

$$f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1$$

Multiply the equation with  $4a^2b^2$

$$h(x, y) = 4b^2x^2 + 4a^2y^2 - 4a^2b^2$$

If  $h(x, y) = 0$ , then the point  $(x, y)$  is exactly on the ellipse

If  $h(x, y) < 0$ , then the point  $(x, y)$  is inside the ellipse

And if  $h(x, y) > 0$ , then the point  $(x, y)$  is outside the ellipse

Then initialize the value.

$$h\left(1, b - \frac{1}{2}\right) = 4b^2 + a^2(1 - 4b)$$

for the first set, when plot the next point  $x$  is always increase

for the first set, when plot the next point  $y$  is always decrease

if the  $h > 0$ ,  $y$  is decrease and  $h$  value will change to be

$$h(x + dx, y + dy) = h(x, y) + 4b^2dx(2x - dx)$$

if the  $h < 0$ ,  $x$  is increase and  $h$  value will change to be

$$h(x + dx, y + dy) = h(x, y) + 4a^2dy(2y - dy)$$

## 2. Write two programs implementing these algorithms.

Ellipse.py

```
from Kennedy import Kennedy
from Carpenter import Carpenter
import turtle
```

Import Class Kennedy to calculate with Kennedy algorithm

Import Class Carpenter to calculate with Carpenter algorithm

Import turtle to draw graph

```
if __name__ == "__main__":
    main()
```

Let module that have name 'main' be main in program

```
def main():
    # get user input
    print("Ellipse center point")
    x = int(input("x: "))
    y = int(input("y: "))
    print("Ellipse Property")
    a = int(input("x radius: "))
    b = int(input("y radius: "))
    print('Center point is (',x,',',y,')')
    print('Most top point is (',x+a,',0)')
    print('Most right point is (0,',y+b,')')
```

Get center point (x, y) and get x radius (a) and y radius (b).

Then show the center point, (a,0) and (0,b) to user.

```
# translation to origin
x, y, tx, ty = transToOrigin(x, y)
```

Translate center point to the origin with module transToOrigin()

```
# translate center point to origin
def transToOrigin(x, y):
    tx = -x
    ty = -y
    x = x + tx
    y = y + ty
    return x, y, tx, ty
```



```

# kennedy
k = Kennedy(a, b)
k.midpoint()
# carpenter
c = Carpenter(a, b)
c.midpoint()

```

Calculate quadrant 1 of ellipse with Kennedy and Carpenter algorithm.

```

# mirror x and y
k.negResult()
c.negResult()

```

Mirror the result for the other quadrant.

```

# translation back to (x,y) point
k.xResult, k.yResult = transback(k.xResult, k.yResult, tx, ty)
k.xNegResult, k.yNegResult = transback(k.xNegResult, k.yNegResult, tx, ty)
c.xResult, c.yResult = transback(c.xResult, c.yResult, tx, ty)
c.xNegResult, c.yNegResult = transback(c.xNegResult, c.yNegResult, tx, ty)

```

Translate all of ellipse point the to the center point with module transback().

```

# translate from origin to the center point
def transback(x, y, tx, ty):
    i = 0
    j = 0
    for i in range(len(x)):
        temp = x[i]
        x[i] = temp - tx
    for j in range(len(y)):
        temp = y[j]
        y[j] = temp - ty
    return x, y

```

```

print("Kennedy")
print("quadrant 1") # (x, y)
printPoint(k.xResult, k.yResult)
print("")
print("quadrant 2") # (-x, y)
printPoint(k.xNegResult, k.yResult)
print("")
print("quadrant 3") # (-x, -y)
printPoint(k.xNegResult, k.yNegResult)
print("")
print("quadrant 4") # (x, -y)
printPoint(k.xResult, k.yNegResult)
print("")

print("Carpenter")
print("quadrant 1") # (x, y)
printPoint(c.xResult, c.yResult)
print("")
print("quadrant 2") # (-x, y)
printPoint(c.xNegResult, c.yResult)
print("")
print("quadrant 3") # (-x, -y)
printPoint(c.xNegResult, c.yNegResult)
print("")
print("quadrant 4") # (x, -y)
printPoint(c.xResult, c.yNegResult)

```

Show all point to user.

quadrant 1 (x, y)

quadrant 2 (-x, y) [mirror by y-axis]

quadrant 3 (-x, -y) [mirror by x-axis]

quadrant 4 (x, -y) [mirror by x-axis and y-axis]

```

# setup for drawing graph
turtle.setup()
turtle.penup()
turtle.hideturtle()

```

Set up the screen for drawing graph. To show only point the plotted.

```

# kennedy
draw(k.xResult, k.yResult, "blue")
draw(k.xNegResult, k.yResult, "blue")
draw(k.xNegResult, k.yNegResult, "blue")
draw(k.xResult, k.yNegResult, "blue")

# carpenter
draw(c.xResult, c.yResult, "red")
draw(c.xNegResult, c.yResult, "red")
draw(c.xNegResult, c.yNegResult, "red")
draw(c.xResult, c.yNegResult, "red")

```

Draw the graph with module draw().

```

# draw graph
def draw(x,y,color):
    i = 0
    for i in range(len(x)):
        # scaling the point
        temp_x = x[i] * 10
        temp_y = y[i] * 10
        turtle.goto(temp_x,temp_y)
        turtle.dot(8,color)

```

Scaling the graph for clarify and plot it.

Kennedy.py

Initial value of variable in this class.

```
class Kennedy:

    def __init__(this,a,b):
        this.a = a
        this.b = b
        this.x = a
        this.y = 0
        this.xChange = b*b*(1-2*a)
        this.yChange = a*a
        this.error = 0
        this.xStop = 2*b*b*a
        this.yStop = 0
        this.xResult = []
        this.yResult = []
        this.xNegResult = []
        this.yNegResult = []
```

**a** is x radius

**b** is y radius

start at point (x, y) = (a, 0)

**x** is x coordinate of start point

**y** is y coordinate of start point

**xChange** is for update error when the next point is decrease x.

$$\begin{aligned} &Error(x-1, y+1) + Error(x, y+1) > 0 \\ &2[b^2 + y^2a^2 + 2ya^2 + a^2 - a^2b^2] - 2xb^2 + b^2 > 0 \end{aligned}$$

When (x, y) = (a, 0)

**yChange** is for update error when the next point is increase y.

$$\begin{aligned} &Error(x+1, y-1) + Error(x+1, y) > 0 \\ &2[b^2 + b^2x^2 + 2xb^2 + a^2y^2 - a^2b^2] - 2ay^2 + a^2 > 0 \end{aligned}$$

When (x, y) = (a, 0)

**Error** is error of ellipse

$$Error(x, y) = |x^2b^2 + y^2a^2 - a^2b^2|$$

When (x, y) = (a, 0)

Slope is less than -1 when

$$2xb^2 < 2ya^2$$

**xStop** is  $2xb^2$

**yStop** is  $2ya^2$

**xResult** and **yResult** is list for keep the plotted point.

**xNegResult** and **yNegResult** is list for keep the mirror of plotted point.

```

def midpoint(this):
    # octant 1
    while this.xStop >= this.yStop:
        this.plot()
        this.y += 1
        this.yStop += 2*this.a*this.a
        this.error += this.yChange
        this.yChange += 2*this.a*this.a
        if (2*this.error + this.xChange) > 0:
            this.x -= 1
            this.xStop -= 2*this.b*this.b
            this.error += this.xChange
            this.xChange += 2*this.b*this.b

```

While slope is less than -1. Plot the point.

And for the next point, Increment y and choose between x and x -1 then update the other values.

```

# octant 2
this.x = 0
this.y = this.b
this.xChange = this.b*this.b
this.yChange = this.a*this.a*(1 - 2*this.b)
this.error = 0
this.xStop = 0
this.yStop = 2*this.a*this.a*this.b

```

When reach the point that slope = -1, initial value for the second set to start at (0,b)

```

while this.xStop <= this.yStop:
    this.plot()
    this.x += 1
    this.xStop += 2 * this.b * this.b
    this.error += this.xChange
    this.xChange += 2 * this.b * this.b
    if (2 * this.error + this.yChange) > 0:
        this.y -= 1
        this.yStop -= 2 * this.a * this.a
        this.error += this.yChange
        this.yChange += 2 * this.a * this.a

```

While slope is greater than -1. Plot the point.

And for the next point, Increment x and choose between y and y -1 then update the other values.

```
def plot(this):  
    this.xResult.append(this.x)  
    this.yResult.append(this.y)
```

Module plot is for add the point to list xResult and yResult

```
def negResult(this):  
    this.xNegResult = [-x for x in this.xResult]  
    this.yNegResult = [-y for y in this.yResult]
```

Module negResult is for mirror all point in xResult and yResult.

Carpenter.py

Initial value of variable in this class.

```
class Carpenter:

    def __init__(this,a,b):
        this.a = a
        this.b = b
        this.x = 0
        this.y = b
        this.a2t8 = 8*a*a
        this.b2t8 = 8*b*b
        this.h = 4*b*b + a*a*(1-4*b)
        this.d1 = 12*b*b
        this.d2 = -this.a2t8*(b-1)
        this.sn = b*b
        this.sd = a*a*b - a*a/2
        this.xResult = []
        this.yResult = []
        this.xNegResult = []
        this.yNegResult = []
```

**a** is x radius

**b** is y radius

start at point (x, y) = (0, b)

**x** is x coordinate of start point

**y** is y coordinate of start point

**a2t8** and **b2t8** is for make code looks simply.

**h** is decision value

$$h\left(1, b - \frac{1}{2}\right) = 4b^2 + a^2(1 - 4b)$$

**d1** is for update h when the next point is increase x.

$$h(x + dx, y + dy) = h(x, y) + 4b^2 dx(2x - dx)$$

**d2** is for update h when the next point is decrease y.

$$h(x + dx, y + dy) = h(x, y) + 4a^2 dy(2y - dy)$$

When (x, y) = (a, 0)

Slope is less than -1 when

$$b^2 < a^2 b - \frac{a^2}{2}$$

**sn** is  $2xb^2$

**sd** is  $2ya^2$

**xResult** and **yResult** is list for keep the plotted point.

**xNegResult** and **yNegResult** is list for keep the mirror of plotted point.

```

def midpoint(this):
    # slope < 1
    while(this.sn < this.sd):
        if(this.h > 0):
            this.y -= 1
            this.h += this.d2
            this.sd -= this.a*this.a
            this.d2 += this.a2t8
        this.x += 1
        this.h += this.d1
        this.sn += this.b*this.b
        this.d1 += this.b2t8
        this.plot()

```

While slope is greater than -1.

Increment x and choose between y and y – 1 and update the other values then plot the point.

```

# slope >1
this.h -= this.b*this.b*(4*this.x*this.x + 4*this.x + 1) + \
    4*this.a*this.a*(this.y - 1)*(this.y - 1) - \
    4*this.a*this.a*this.b*this.b
this.d1 = this.b2t8*(this.x + 1)
this.d2 = -4*this.a*this.a*(2*this.y - 3)

```

When reach the point that slope is -1. initial value for the second set.

h is

$$h(x + 1, y - 1) = 4b^2(x + 1)^2 + 4a^2(y - 1)^2 - 4a^2b^2$$

d1 is for update h when the next point is increase x.

$$h((x + 1) + 1, y + dy) = h(x, y) + 4b^2(2(x + 1) - 1)$$

$$h((x + 1) + 1, y + dy) = h(x, y) + 8b^2(x + 1)$$

d2 is for update h when the next point is decrease y.

$$h(x + dx, (y - 1) - 1) = h(x, y) - 4a^2(2(y - 1) - 1)$$

$$h(x + dx, (y - 1) - 1) = h(x, y) - 4a^2(2y - 3)$$



```

while(this.y > 1):
    if(this.h < 0):
        this.x += 1
        this.h += this.d1
        this.d1 += this.b2t8
    this.y -= 1
    this.h += this.d2
    this.d2 += this.a2t8
    this.plot()

```

While y is greater than 1

decrement y and choose between x and x + 1 and update the other values then plot the point.

```

def plot(this):
    this.xResult.append(this.x)
    this.yResult.append(this.y)

```

Module plot is for add the point to list xResult and yResult

```

def negResult(this):
    this.xNegResult = [-x for x in this.xResult]
    this.yNegResult = [-y for y in this.yResult]

```

Module negResult is for mirror all point in xResult and yResult.

### 3. Evaluate and compare two algorithms and their results

Compare by The result of 2 algorithm

Ellipse center point

x: 1

y: 2

Ellipse Property

x radius: 5

y radius: 8

Center point is ( 1 , 2 )

Most top point is ( 6 ,0)

Most right point is (0, 10 )

#### Kennedy

Kennedy

quadrant 1

( 6 , 2 ) ( 6 , 3 ) ( 6 , 4 ) ( 6 , 5 ) ( 5 , 6 ) ( 5 , 7 ) ( 4 , 8 ) ( 1 , 10 ) ( 2 , 10 ) ( 3 , 9 )

quadrant 2

( -4 , 2 ) ( -4 , 3 ) ( -4 , 4 ) ( -4 , 5 ) ( -3 , 6 ) ( -3 , 7 ) ( -2 , 8 ) ( 1 , 10 ) ( 0 , 10 ) ( -1 , 9 )

quadrant 3

( -4 , 2 ) ( -4 , 1 ) ( -4 , 0 ) ( -4 , -1 ) ( -3 , -2 ) ( -3 , -3 ) ( -2 , -4 ) ( 1 , -6 ) ( 0 , -6 ) ( -1 , -5 )

quadrant 4

( 6 , 2 ) ( 6 , 1 ) ( 6 , 0 ) ( 6 , -1 ) ( 5 , -2 ) ( 5 , -3 ) ( 4 , -4 ) ( 1 , -6 ) ( 2 , -6 ) ( 3 , -5 )

#### Carpenter

Carpenter

quadrant 1

( 2 , 10 ) ( 3 , 9 ) ( 3 , 8 ) ( 3 , 7 ) ( 4 , 6 ) ( 4 , 5 ) ( 5 , 4 ) ( 5 , 3 )

quadrant 2

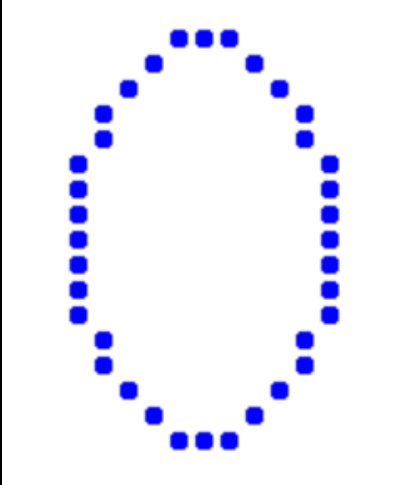
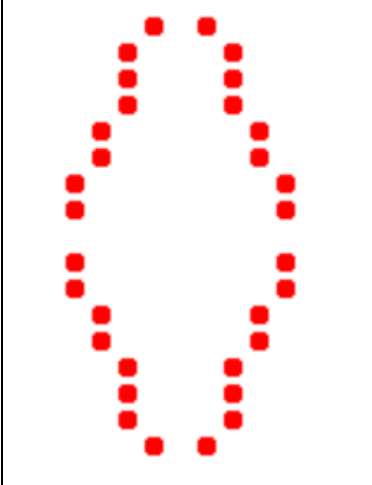
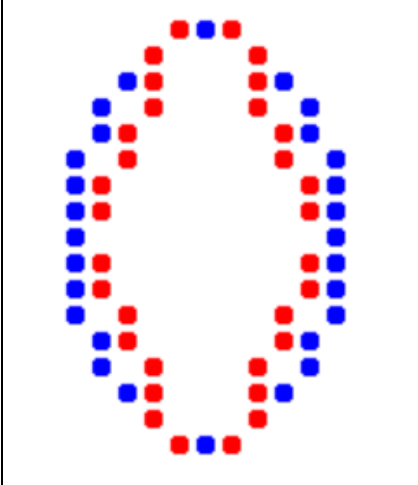
( 0 , 10 ) ( -1 , 9 ) ( -1 , 8 ) ( -1 , 7 ) ( -2 , 6 ) ( -2 , 5 ) ( -3 , 4 ) ( -3 , 3 )

quadrant 3

( 0 , -6 ) ( -1 , -5 ) ( -1 , -4 ) ( -1 , -3 ) ( -2 , -2 ) ( -2 , -1 ) ( -3 , 0 ) ( -3 , 1 )

quadrant 4

( 2 , -6 ) ( 3 , -5 ) ( 3 , -4 ) ( 3 , -3 ) ( 4 , -2 ) ( 4 , -1 ) ( 5 , 0 ) ( 5 , 1 )

Kennedy	Carpenter	both
		

Ellipse center point  
 x: 1  
 y: 2  
 Ellipse Property  
 x radius: 9  
 y radius: 5  
 Center point is ( 1 , 2 )  
 Most top point is ( 10 ,0)  
 Most right point is (0, 7 )

### Kennedy

Kennedy

quadrant 1

( 10 , 2 ) ( 10 , 3 ) ( 9 , 4 ) ( 1 , 7 ) ( 2 , 7 ) ( 3 , 7 ) ( 4 , 7 ) ( 5 , 6 ) ( 6 , 6 ) ( 7 , 6 ) ( 8 , 5 )

quadrant 2

( -8 , 2 ) ( -8 , 3 ) ( -7 , 4 ) ( 1 , 7 ) ( 0 , 7 ) ( -1 , 7 ) ( -2 , 7 ) ( -3 , 6 ) ( -4 , 6 ) ( -5 , 6 ) ( -6 , 5 )

quadrant 3

( -8 , 2 ) ( -8 , 1 ) ( -7 , 0 ) ( 1 , -3 ) ( 0 , -3 ) ( -1 , -3 ) ( -2 , -3 ) ( -3 , -2 ) ( -4 , -2 ) ( -5 , -2 ) ( -6 , -1 )

quadrant 4

( 10 , 2 ) ( 10 , 1 ) ( 9 , 0 ) ( 1 , -3 ) ( 2 , -3 ) ( 3 , -3 ) ( 4 , -3 ) ( 5 , -2 ) ( 6 , -2 ) ( 7 , -2 ) ( 8 , -1 )

### Carpenter

Carpenter

quadrant 1

( 2 , 7 ) ( 3 , 7 ) ( 4 , 7 ) ( 5 , 6 ) ( 6 , 6 ) ( 7 , 6 ) ( 8 , 5 ) ( 9 , 4 ) ( 9 , 3 )

quadrant 2

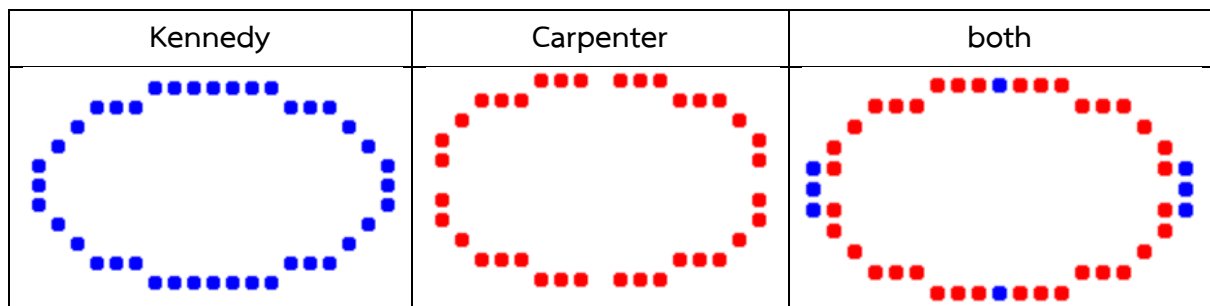
( 0 , 7 ) ( -1 , 7 ) ( -2 , 7 ) ( -3 , 6 ) ( -4 , 6 ) ( -5 , 6 ) ( -6 , 5 ) ( -7 , 4 ) ( -7 , 3 )

quadrant 3

( 0 , -3 ) ( -1 , -3 ) ( -2 , -3 ) ( -3 , -2 ) ( -4 , -2 ) ( -5 , -2 ) ( -6 , -1 ) ( -7 , 0 ) ( -7 , 1 )

quadrant 4

( 2 , -3 ) ( 3 , -3 ) ( 4 , -3 ) ( 5 , -2 ) ( 6 , -2 ) ( 7 , -2 ) ( 8 , -1 ) ( 9 , 0 ) ( 9 , 1 )



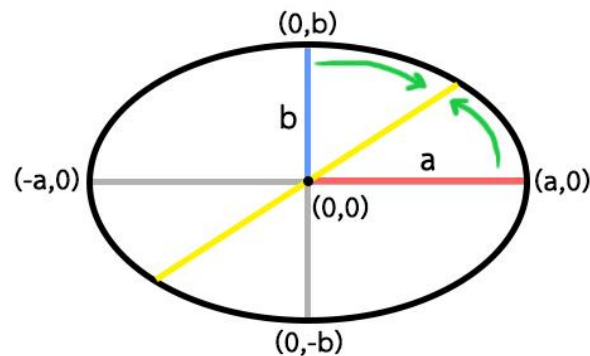
From the result we can see that Kennedy algorithm is generate more pixels than Carpenter algorithm.

And Carpenter algorithm is not plot the point at (a, 0) and (0, b)

Compare by the direction when plotting

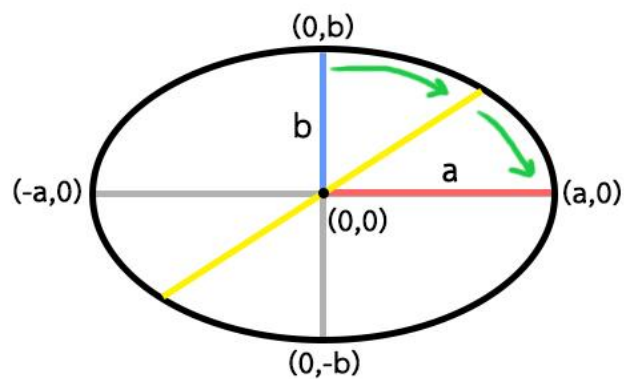
Kennedy

Plot the graph from (a, 0) and (0, b) to the point that slope equals -1.



Carpenter

Plot the graph from (0, b) to the point that slope equals -1 and then continue plot until y equals 0.



Compare by the decision value

Kennedy

Compare between 2 value of error and choose the less one

$$Error(x, y) = |x^2b^2 + y^2a^2 - a^2b^2|$$

Carpenter

Use h value for decision

$$h(x, y) = 4b^2x^2 + 4a^2y^2 - 4a^2b^2$$

If  $h(x, y) = 0$ , then the point (x, y) is exactly on the ellipse

If  $h(x, y) < 0$ , then the point (x, y) is inside the ellipse

And if  $h(x, y) > 0$ , then the point (x, y) is outside the ellipse