Fundamentals of Computer Graphics and Image Processing

3. LECTURE – CIRCLE AND ELLIPSE LINE DRAWING ALGORITHMS

Lecture plan

Information about the test

Circle line drawing algorithm

- Circle line mathematical description
- Direct approach to circle line drawing
- Mid-point circle line drawing algorithm
- Mathematical calculations for circle line

Ellipse line drawing algorithm

- Mathematical description of the ellipse line
- Midpoint ellipse line drawing algorithm

Information about the test

You will have the test in two weeks (October 27th)

The test includes:

- 5 quiz questions on the course evaluation system (0.25 points)
- 3 open questions on theory (0.75 points)
- 1 mathematical calculation task on straight line or circle line algorithms (1 point). A photo of the handwritten solution is required to pass, typed versions will not be accepted!

The test is taken only ONCE and CANNOT be retaken! It is only available online for 1 hour during the time of the lecture (13:00-14:00). If you are unable to attend it due to some serious circumstances, please inform me at least 3 days before the test using the ortus message system or e-mail (*Katrina.Bolocko@rtu.lv*). You will have to support your circumstances with an authoritative document! We will then negotiate a date and time when you can take the test.

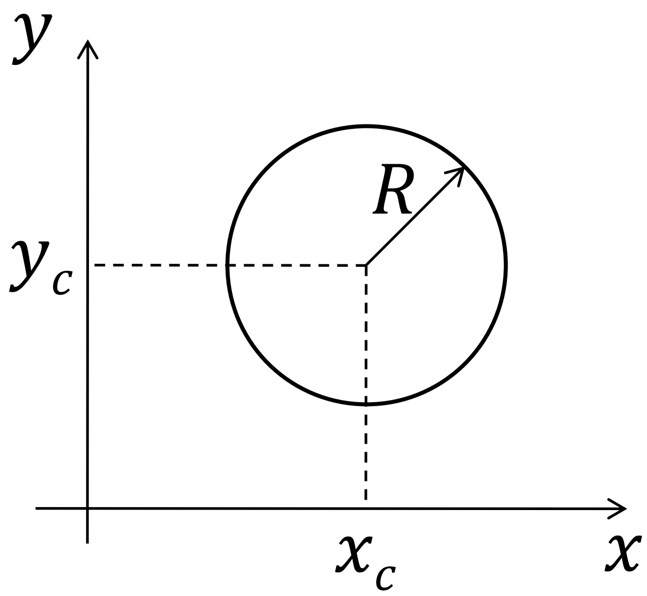
Circle line drawing algorithm

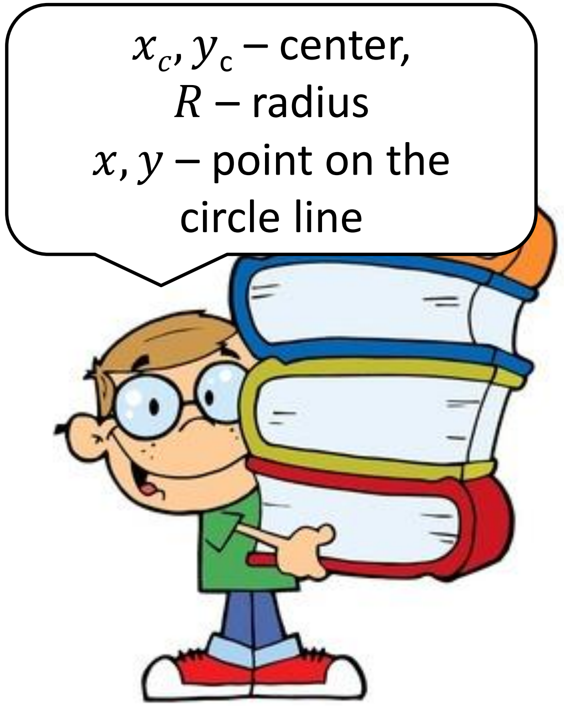
MATH TO ALGORITHM

Circle line mathematical description

In mathematical coordinates system, the circle is described:

$$(x - x_c)^2 + (y - y_c)^2 = R^2 \qquad (1)$$





Circle line mathematical description

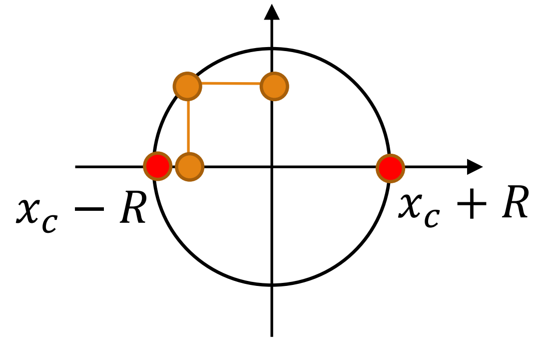
$$(x - x_c)^2 + (y - y_c)^2 = R^2$$

From this can be deduced that in the interval

$$x_c - R \leq x \leq x_c + R$$

value y can be found:

$$y = y_c \pm \sqrt{R^2 - (x_c - x)^2}$$



We can use this formula to calculate the points of the circle.

Direct approach to circle drawing

Direct approach:

The mathematical formula may be used to calculate circle line points.

$$y = y_c \pm \sqrt{R^2 - (x_c - x)^2}$$

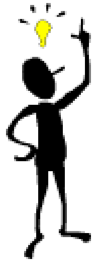
Input: x_c, y_c, R
Algorithm begins from $x = 0$
Until x reaches point $x = R$, repeat:

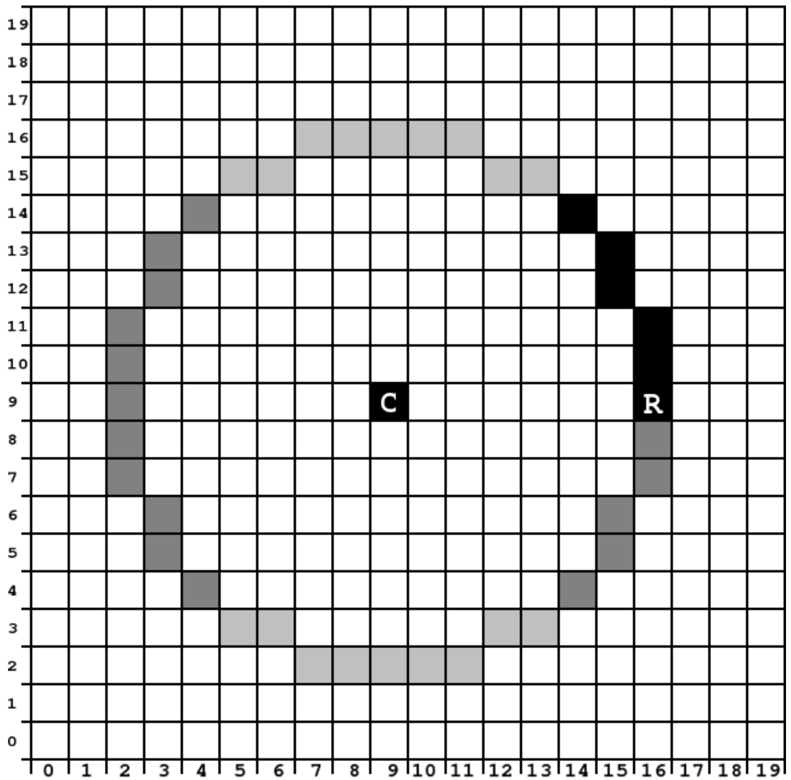
For each x calculate the value y using the mathematical formula:

$$y = round(\sqrt{R^2 - x^2})$$

Draw pixels with coordinates:

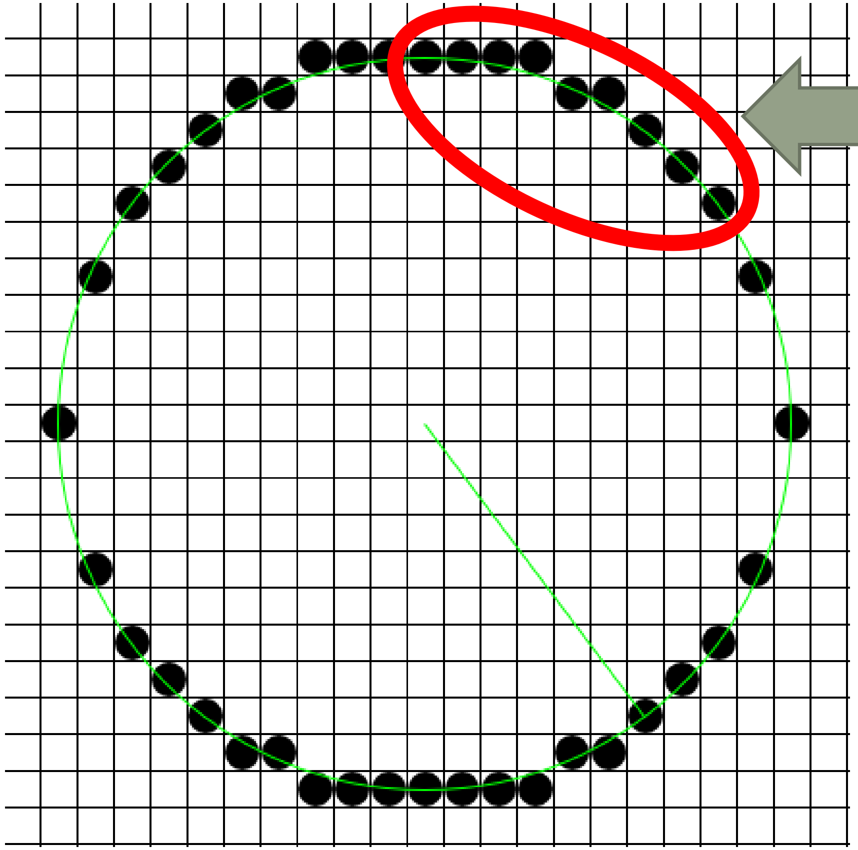
$(x_c + x, y_c + y)$ $(x_c + x, y_c - y)$



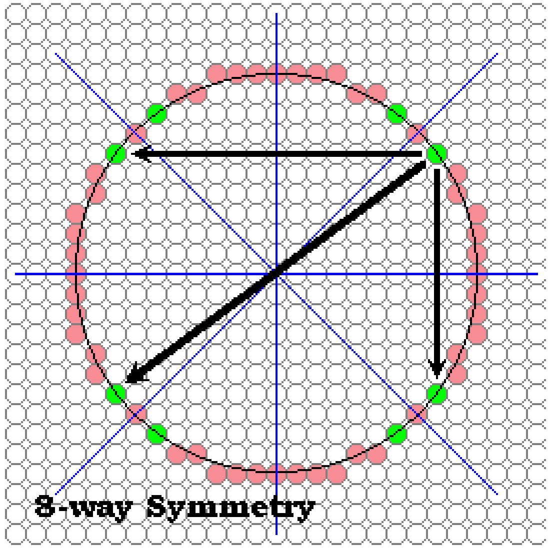


Direct approach to circle drawing

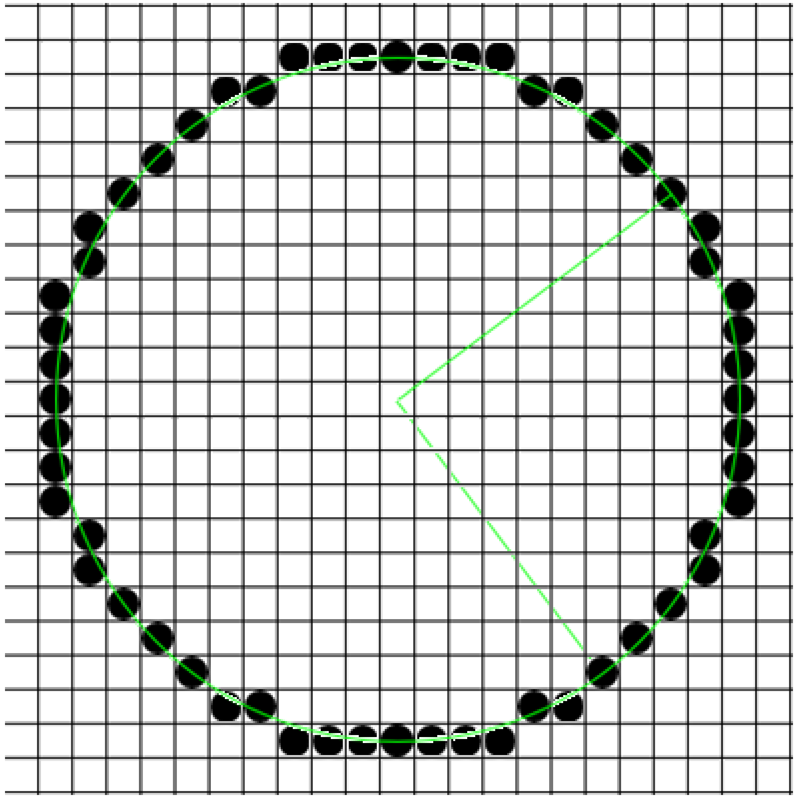
Unfortunately, the resulting circle has holes!

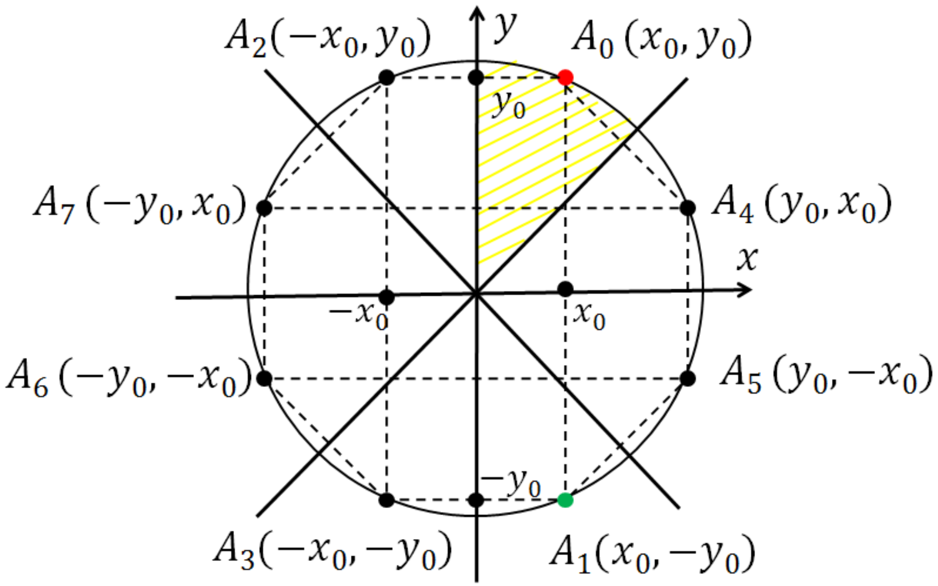


We can use symmetry to remedy this!

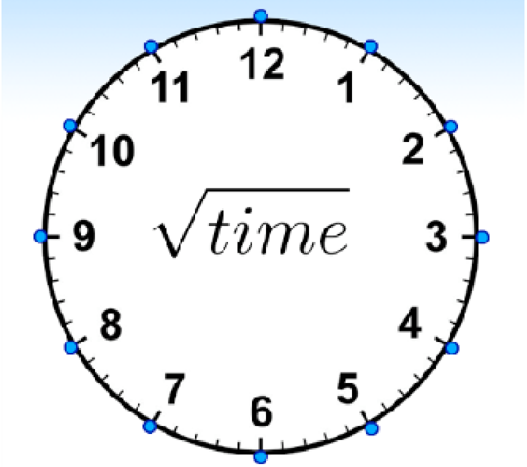


Symmetry of the circle





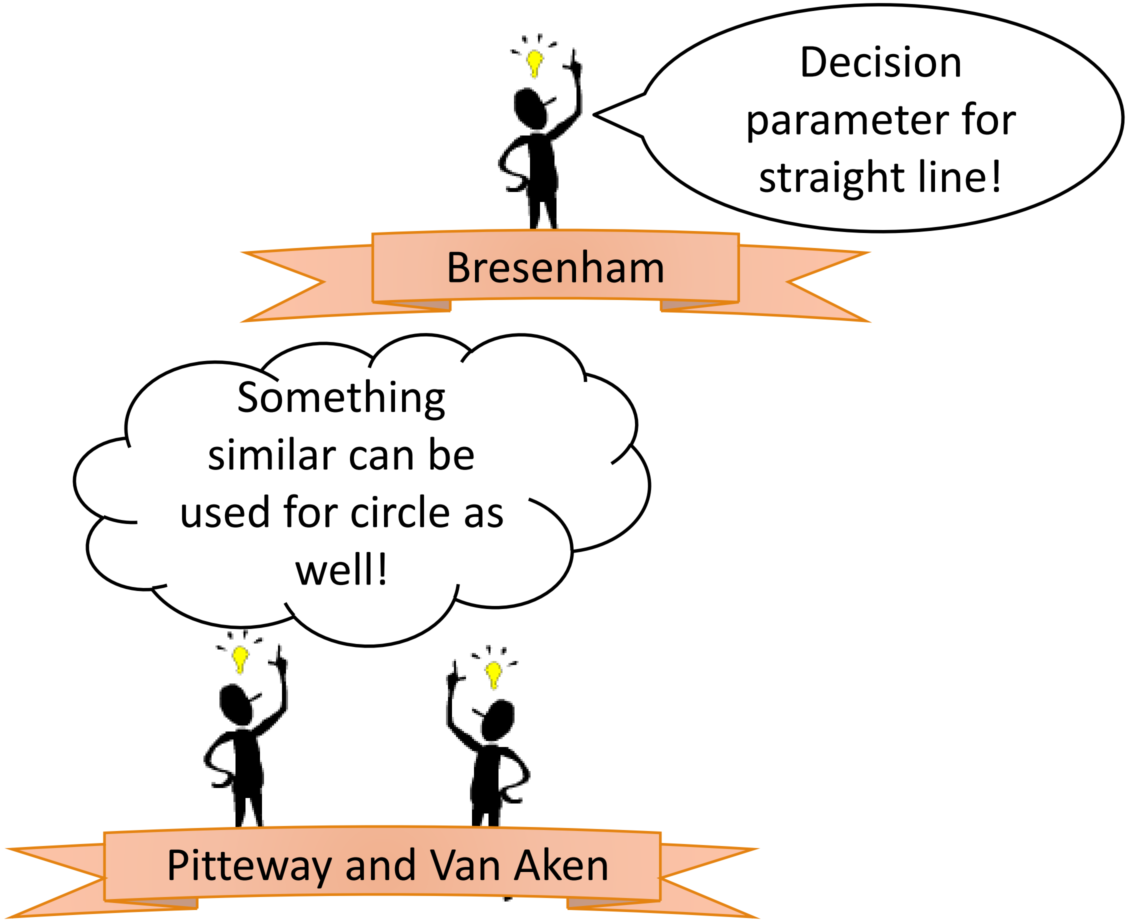
Midpoint circle line drawing algorithm

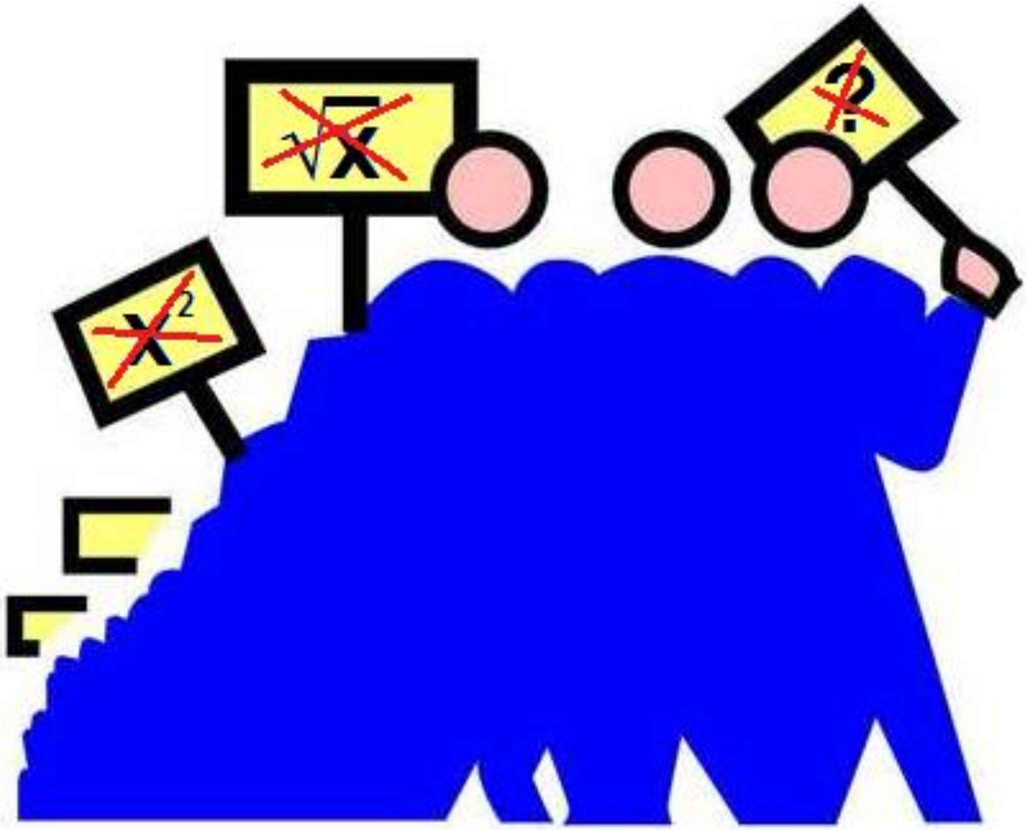


$\sqrt{x} + x^2$

196?. YEAR

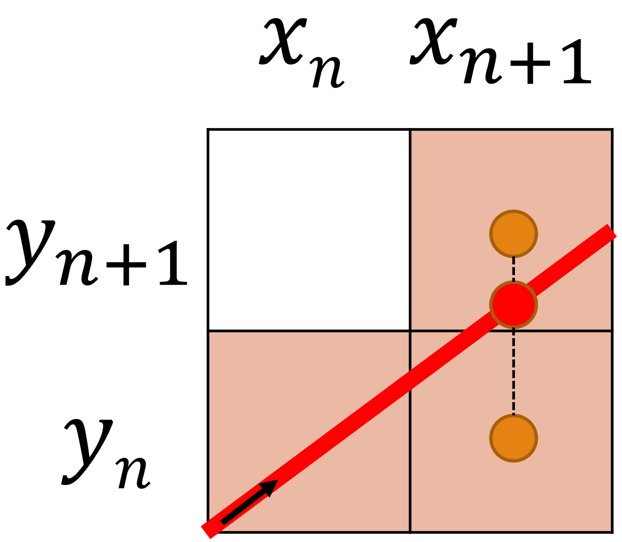
Midpoint circle line drawing algorithm



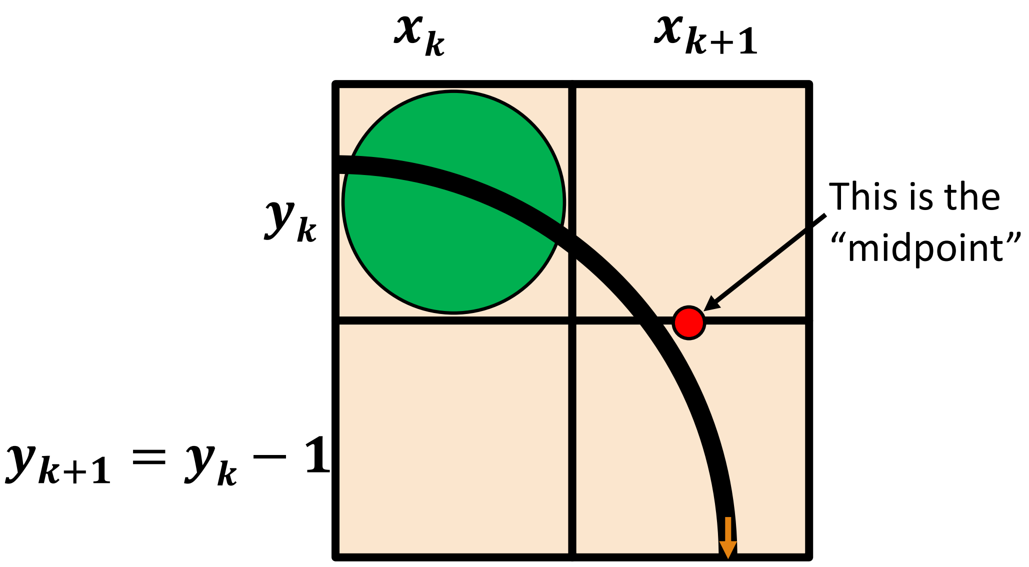


Midpoint circle line drawing algorithm

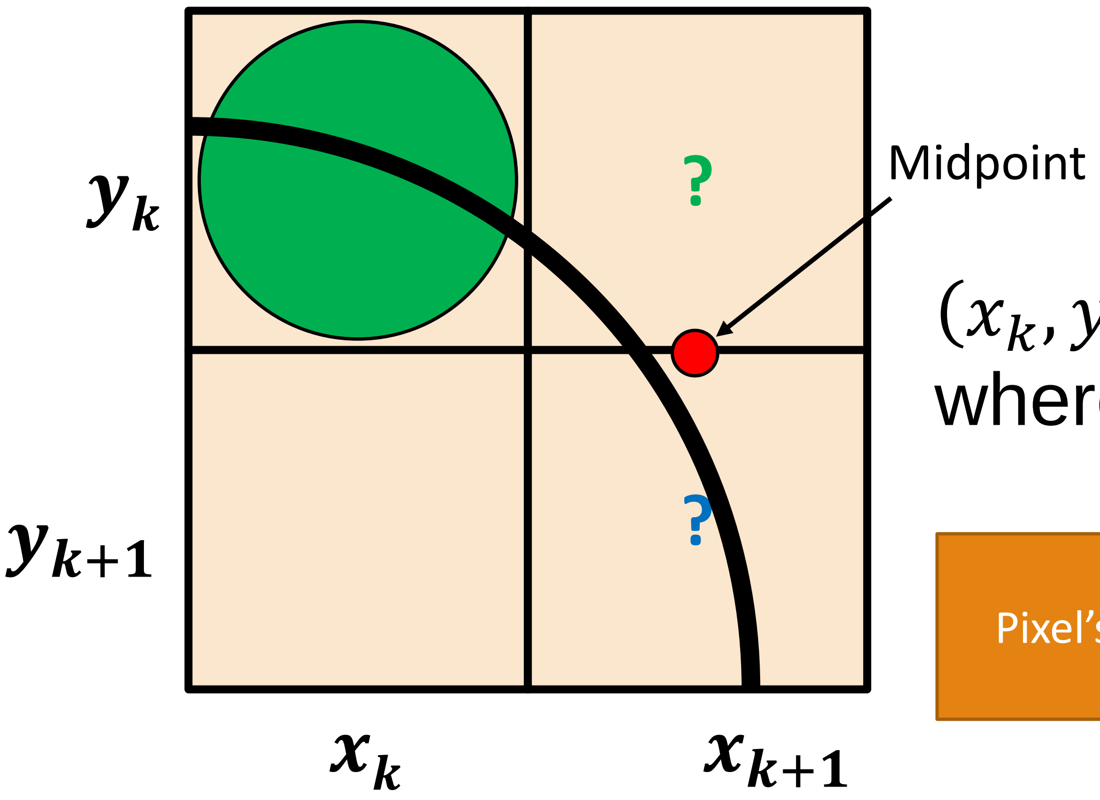
Bresenham's straight line drawing algorithm



Midpoint circle line drawing algorithm



Midpoint circle line drawing algorithm



$(x_k, y_k) \rightarrow (x_{k+1}, y_{k+1})$,
where $y_{k+1} = y_k$ or $y_k - 1$

Pixel's index!

Pixel's value!

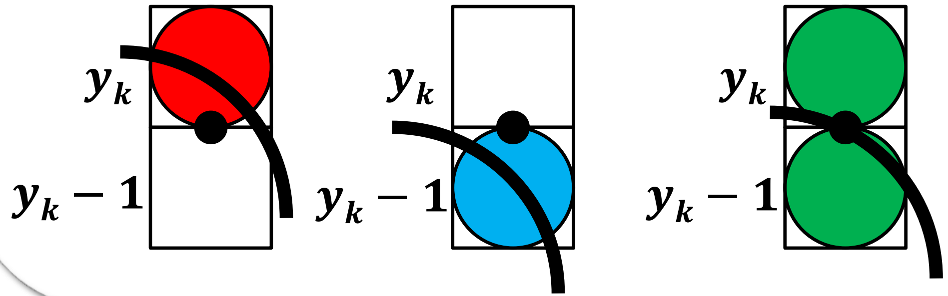
Midpoint circle line drawing algorithm

A decision parameter p_k must work like this:

If $p_k < 0$, then the midpoint is inside the circle and the pixel y_k is closer to the circle line

If $p_k > 0$, then the midpoint is outside the circle and the pixel $y_k - 1$ is closer to the circle line

If $p_k = 0$, then the midpoint is on the circle line



Midpoint circle line drawing algorithm

How to find such a parameter? Well, actually, we can use the circle function $f_r(x, y) = x^2 + y^2 - R^2$. Check out this neat property of the circle function:

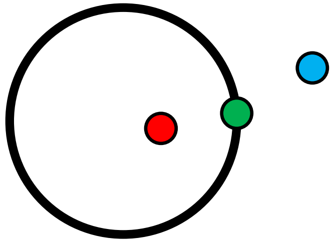
(2)

Property of the circle line function:

$f_r(x, y) < 0$, **if point (x, y) is inside circle line**

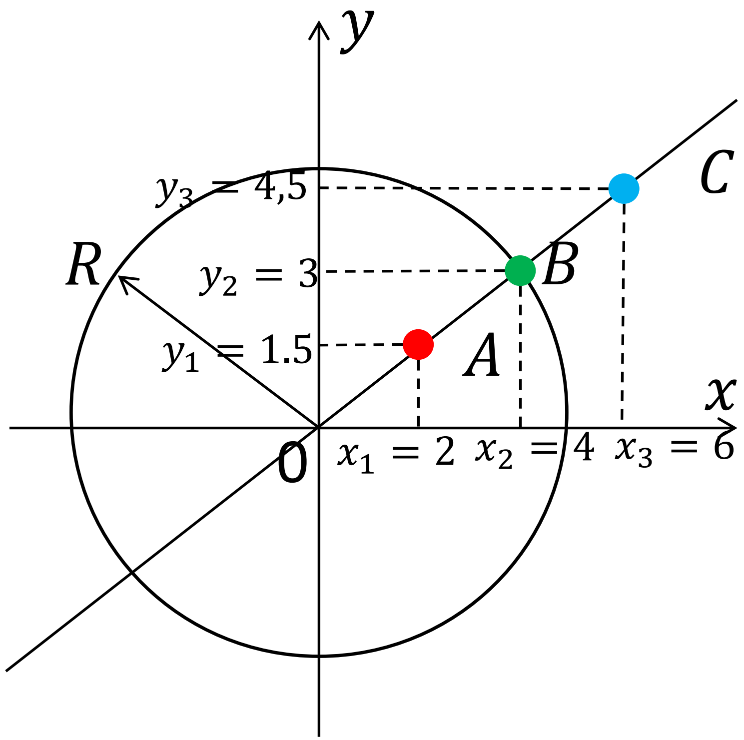
$f_r(x, y) = 0$, **if point (x, y) is on the circle line**

$f_r(x, y) > 0$, **if point (x, y) is outside circle line**



Circle function property:

Let's see if it's true, OK? Let's consider the circle function $x^2 + y^2 - R^2$, and $R = 5$, then:



A: $x_1^2 + y_1^2 - R^2 = 4 + 2.25 - 25 = -18.75 < 0$

B: $x_2^2 + y_2^2 - R^2 = 9 + 16 - 25 = 0$

C: $x_3^2 + y_3^2 - R^2 = 36 + 20{,}25 - 25 = 31{,}25 > 0$

Midpoint circle line drawing algorithm

This means that the function itself may be used as the decision parameter to see where the **midpoint** is!

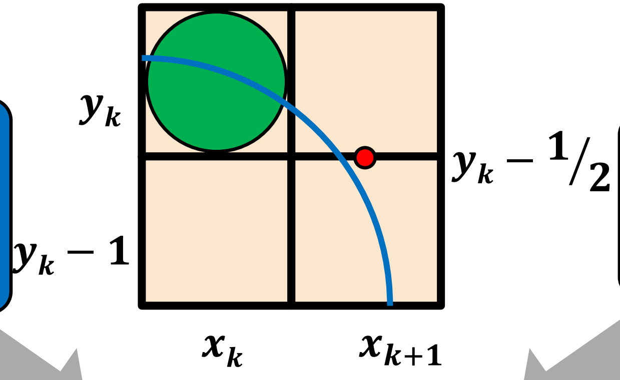
So the positive or negative sign of the **circle function** may show where the point (x, y) is! Just like Bresenham's decision parameter!

So, the decision parameter is the **circle function**, into which we inserted the **middle point** coordinates! It tells us the midpoint's position to relation with the circle line!

Midpoint circle line drawing algorithm

Circle function:

$f_r(x, y) = x^2 + y^2 - R^2$



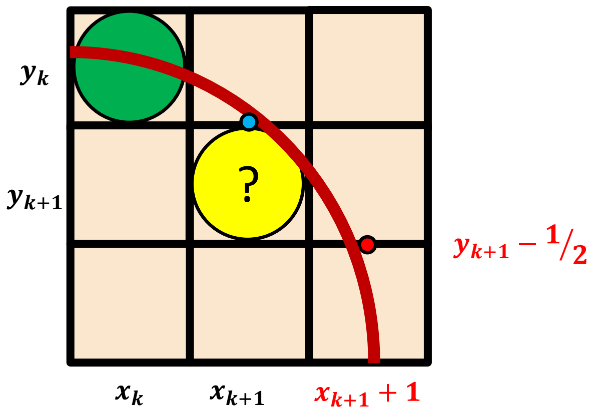
Midpoint:

$(x_k + 1, y_k - ½)$

Decision parameter:

$$p_k = f_r(x_k + 1, y_k - ½) = (x_k + 1)^2 + (y_k - ½)^2 - R^2$$

Midpoint circle line drawing algorithm



It's easy for the initial point: When at point $(\mathbf{0}, \boldsymbol{R})$, we get

$$p_0 = f_r\left(0 + 1, R - \tfrac{1}{2}\right) = 1^2 + (R - \tfrac{1}{2})^2 - R^2 = \tfrac{5}{4} - R$$

But since computers use integers, we use this formula:

$$p_0 = 1 - R$$

But how do we calculate the rest of the decision parameters? Let's look at these two formulas:

- $p_k = f_r(x_k + 1, y_k - ½)$
- $p_{k+1} = f_r(x_{k+1} + 1, y_{k+1} - ½)$

Now just like with Bresenham, let's subtract the $\boldsymbol{p_k}$ from $\boldsymbol{p_{k+1}}$, and check the result:

$$p_{k+1} - p_k = (x_{k+1} + 1)^2 + \left(y_{k+1} - \frac{1}{2}\right)^2 - R^2 - (x_k + 1)^2 - \left(y_k - \frac{1}{2}\right)^2 + R^2$$
$$= p_k + 2x_{k+1} + \left(y_{k+1}^2 - y_k^2\right) - (y_{k+1} - y_k) + 1$$

Depending on the chosen pixel, the coordinate $\boldsymbol{y_{k+1}}$, can take value of $\boldsymbol{y_k}$ or $\boldsymbol{y_k - 1}$, this means that the equation above can take only two forms, depending on $\boldsymbol{p_k}$ sign:

$$p_{k+1} = p_k + 2x_{k+1} + 1 \quad \textbf{OR} \quad p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$$

Midpoint circle line drawing algorithm

Input: x_c, y_c, R

1. Initial values for the 0[th] step:
   - $x_k, y_k = (0, R)$
   - $p_k = 1 - R$
2. While $x_k < y_k$, *{WHILE}* repeat:
   1. If $p_k < 0$ then: *{IF}*

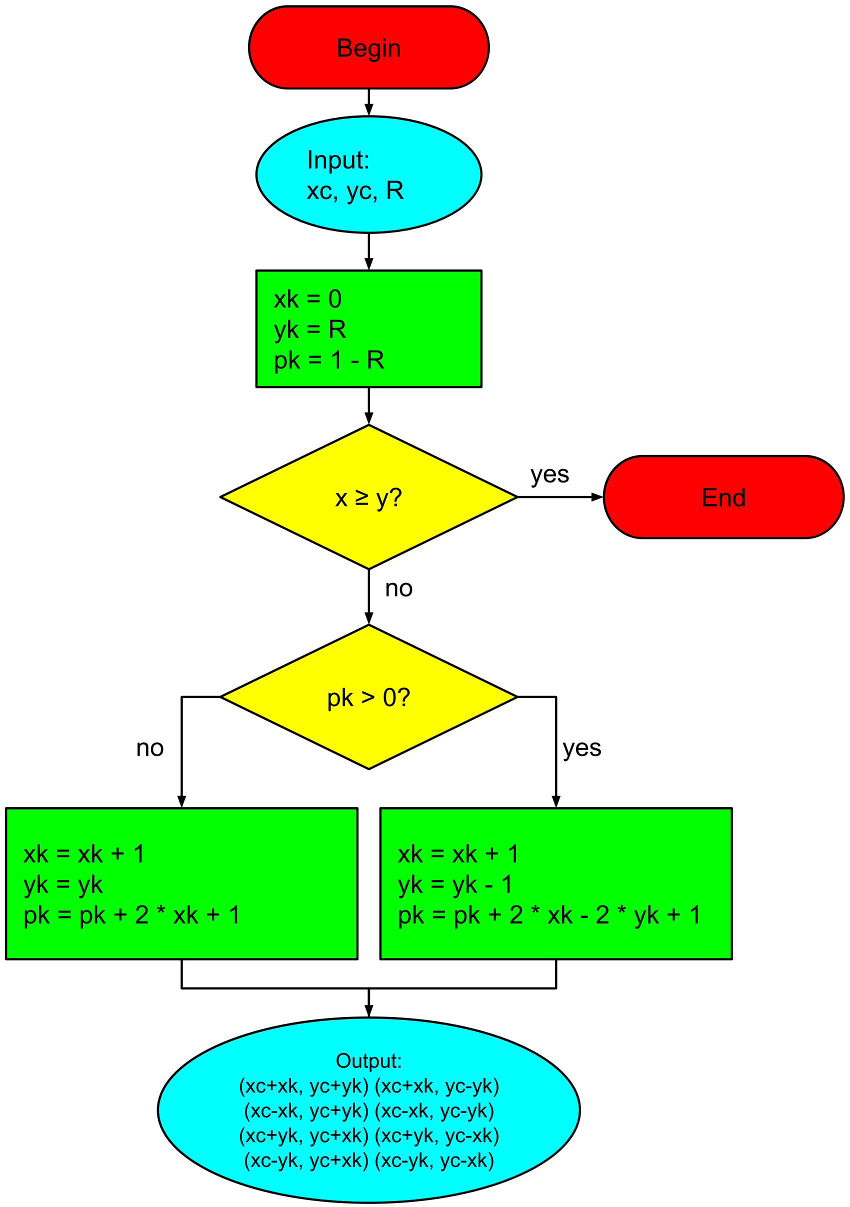
      $x_{k+1} = x_k + 1$ and

      $p_{k+1} = p_k + 2x_{k+1} + 1$
   2. If $p_k \geq 0$ then: *{ELSE}*

      $x_{k+1} = x_k + 1,$

      $y_{k+1} = y_k - 1$ and

      $p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$
   3. Draw pixel x_k, y_k symmetrically in 8 parts, taking the circle's central point into account.



Ellipse midpoint algorithm

MATH TO ALGORITHM

Mathematical description of the ellipse line

In canonical form, the points that are aligned on ellipse line are described as follows:

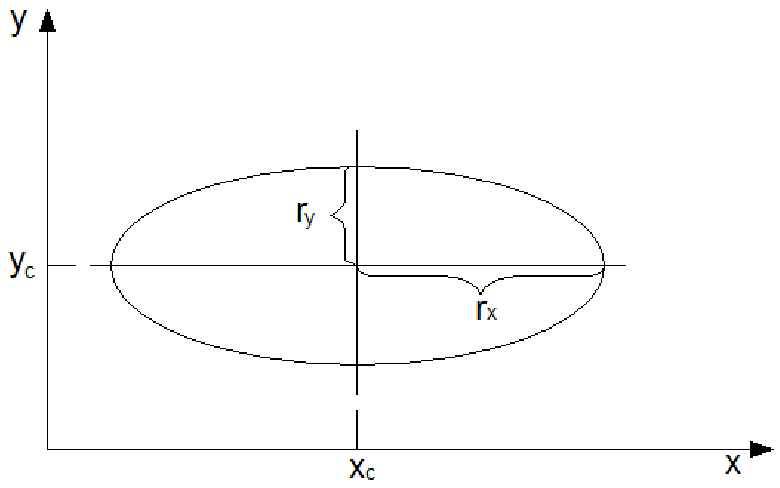
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

If the ellipse center (x_c, y_c) does not match the coordinates system starting point, then:

$$\frac{(x-x_c)^2}{r_x^2} + \frac{(y-y_c)^2}{r_y^2} = 1,$$

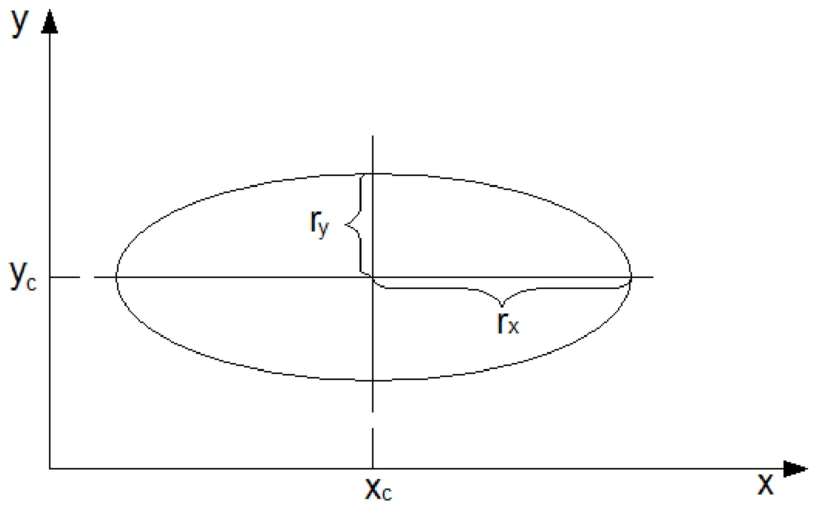
where $r_x = a$ – horizontal half-axis

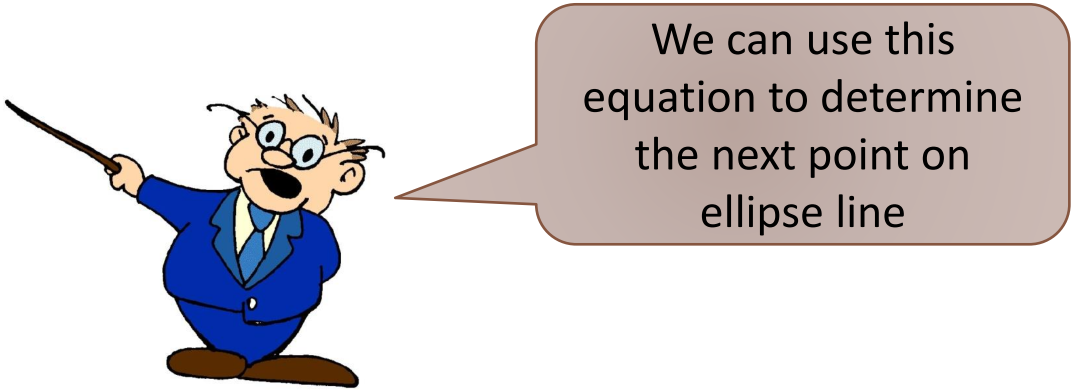
$r_y = b$ – vertical half-axis



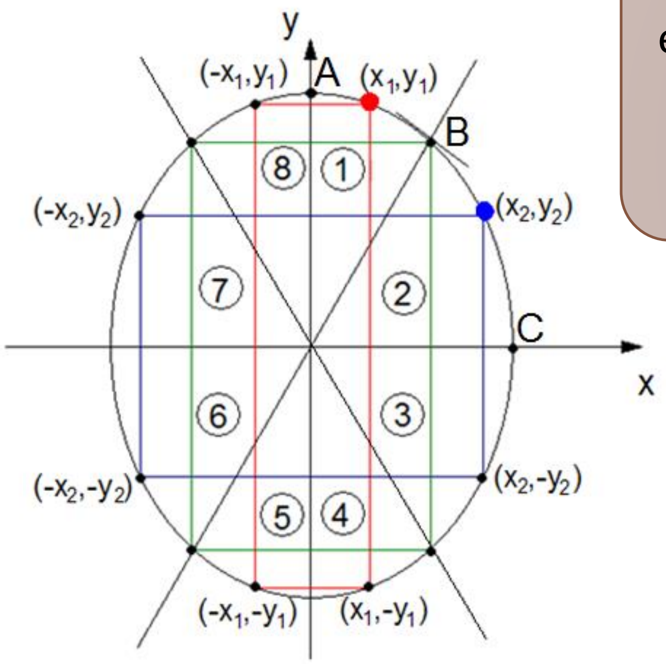
Mathematical description of the ellipse line

$$\frac{(x-x_c)^2}{r_x^2}+\frac{(y-y_c)^2}{r_y^2}=1 \text{ OR } r_y^2(x-x_c)^2+r_x^2(y-y_c)^2=r_x^2r_y^2$$



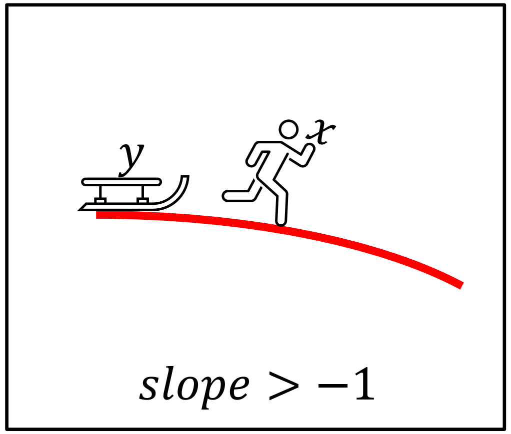


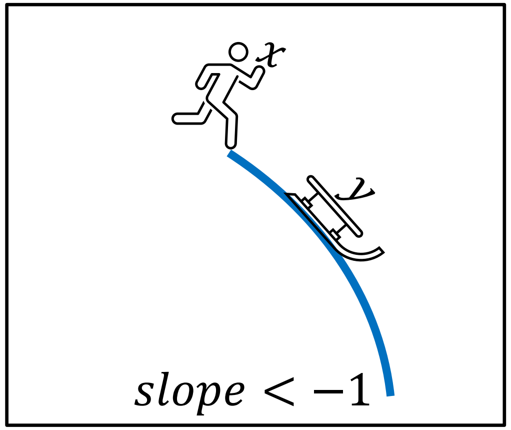
Mathematical description of the ellipse line



We can use 4-way symmetry with the ellipse. But first, the ellipse points should be calculated in 2 parts: from A to B we follow x coordinate and from B to C we use y.

WHY? This is due to the slope of the ellipse. Depending on the slope, it's either the x or the y coordinate that changes more rapidly





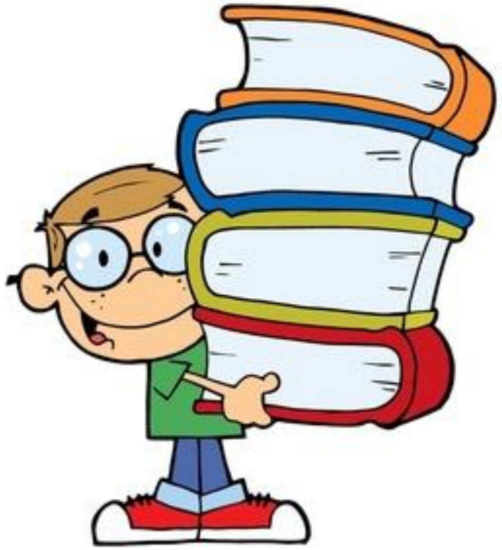
Mathematical description of the ellipse line

So now we know that regions are defined by slopes. For the ellipse to calculate the slope, we can use the formula:

$$\boxed{slope = -\frac{2r_y^2 x}{2r_x^2 y}}$$

At point B the slope's value is -1, so it means that:

$$\boxed{r_y^2 x = r_x^2 y}$$



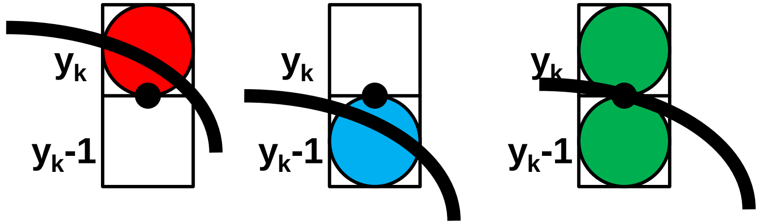
Ellipse line drawing algorithm (Part I, before point B)

The decision parameter p_k must meet the following conditions:

If $p_k < 0$, then midpoint is inside ellipse and pixel y_k is closer to the line

If $p_k > 0$, then midpoint is outside ellipse and the closest pixel is $y_k - 1$

If $p_k = 0$, then midpoint is on the ellipse line



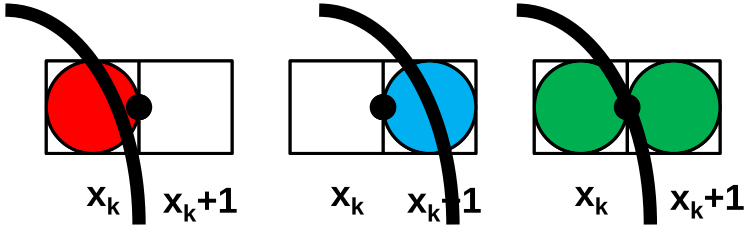
Ellipse line drawing algorithm (Part II, after point B)

The decision parameter p_k must meet the following conditions:

If p_k< 0 then midpoint is inside ellipse and pixel x_k is closer to the line

If p_k > 0, then midpoint is outside ellipse and the closest pixel is $x_k + 1$

If $p_k = 0$, then midpoint is on the ellipse line



Ellipse line drawing algorithm

So just as the circle line, the ellipse line drawing algorithm uses the ellipse function as the decisional parameter:

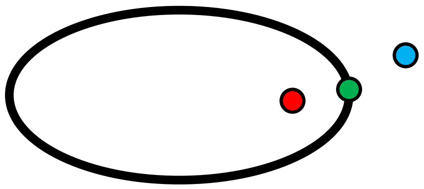
$$f_{el}(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

In this case:

$\boldsymbol{f_r(x, y) < 0}$**, if point** $\boldsymbol{(x, y)}$ **is inside ellipse**

$\boldsymbol{f_r(x, y) = 0}$**, if point** $\boldsymbol{(x, y)}$ **is on the ellipse line**

$\boldsymbol{f_{r(x, y)} > 0}$**, if point** $\boldsymbol{(x, y)}$ **is outside ellipse**

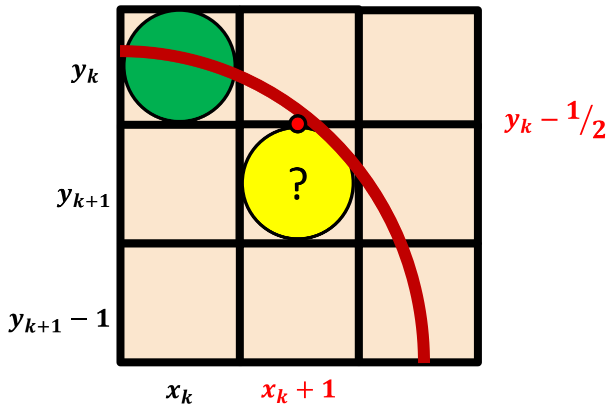


Ellipse line drawing algorithm (Part I)

At part I, the algorithm uses $\boldsymbol{x}$ coordinate as the main variable, each time increasing the $\boldsymbol{x}$ (and does this until $\boldsymbol{r_x^2 y = r_y^2 x}$, when it reaches point **B**)

The decision parameter depends on the midpoint, at the first part the midpoint coordinates are:
$(x_k + 1, y_k - ½)$

So the decision parameter in Part I is:
$p_k = f_{el}\left(x_k + 1, y_k - \frac{1}{2}\right) = r_y^2\left(x_k + 1\right)^2 + r_x^2\left(y_k - \frac{1}{2}\right)^2 - r_x^2 r_y^2$



Ellipse line drawing algorithm (Part I)

Just like with the circle, the next decision parameter p_{k+1} can be described through the previous one p_k:

$$p_{k+1} = p_k + 2r_y^2(x_k + 1) + r_y^2 + r_x^2\left[\left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2\right],$$

where y_{k+1} may be y_k, or $y_k - 1$, depending on the sign of p_k.

At Part I, in the initial point $(0, ry)$ we get

$$p_0 = f_{el}\left(0 + 1, r_y - \frac{1}{2}\right) = r_y^2 - r_x^2 r_y + \frac{1}{4}r_x^2$$

Then, at the beginning of each step it is necessary to assess the formula $r_x^2 y \geq r_y^2 x$ and if it is true, go to Part II.

Ellipse line drawing algorithm (Part II)

At the second part the algorithm uses $\boldsymbol{y}$ coordinate as the main variable, decreasing $\boldsymbol{y}$ at each iterative step until it reaches 0.

The decision parameter is calculated using the midpoint and in the second part it is $(x_k + ½, y_k - 1)$

Then the initial decision parameter for the second part is:

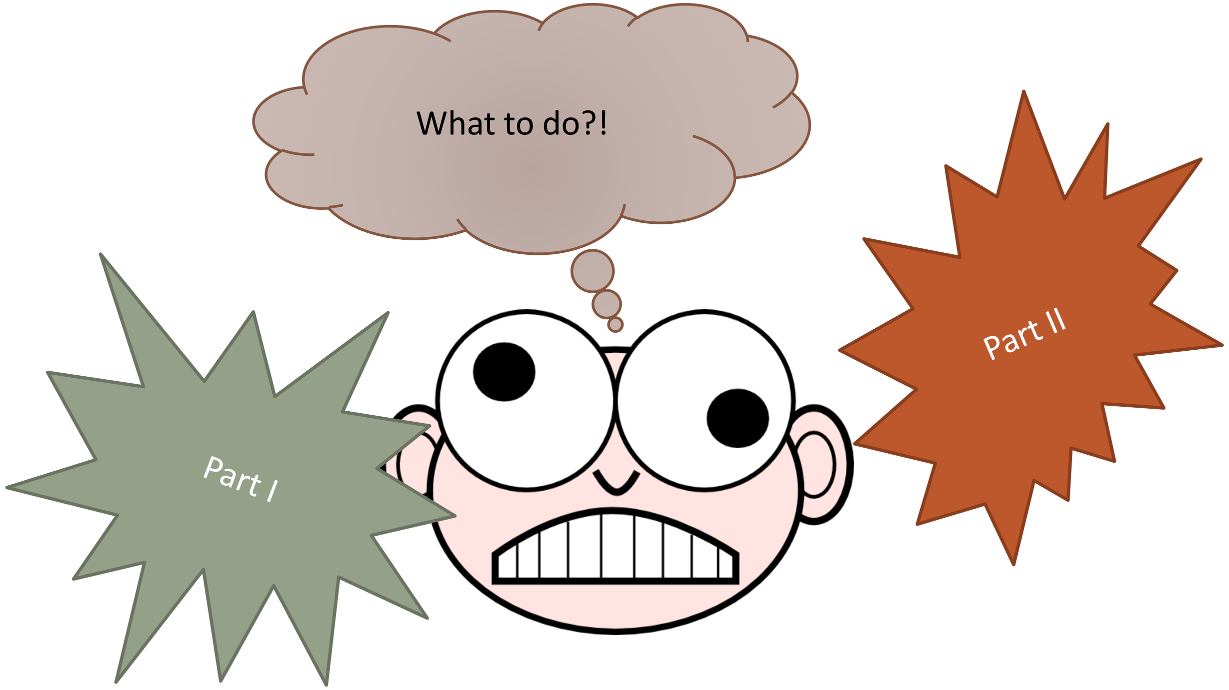
$$p_k = f_{el}\left(x_k + \frac{1}{2}, y_k - 1\right) = r_y^2(x_k + \frac{1}{2})^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2$$

At the next step, the decision parameter, just like in the first part, will be calculated as follows:

$$p_{k+1} = p_k + 2r_x^2(y_k - 1) + r_x^2 + r_y^2[\left(x_{k+1} + \frac{1}{2}\right)^2 - \left(x_k - \frac{1}{2}\right)^2],$$

where x_{k+1} is x_k, or $x_k + 1$, depending on the sign of p_k.

Ellipse line drawing algorithm



Ellipse line drawing algorithm (1)

Input data: r_x, r_y,(x_c,y_c)=(0,0)

1. Define the starting point and initial decision parameter:

   $(x_0, y_0) = (0, r_y)$ and $p_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$

2. While $r_y^2 x < r_x^2 y$, for each x_k, starting with $k = 0$:
   - if $p_k < 0$, then the next pixel (x_{k+1}, y_{k+1}) is $(x_k + 1, y_k)$ and $p_{k+1} = p_k + 2r_y^2 x_{k+1} + r_y^2$
   - Else the next pixel (x_{k+1}, y_{k+1}) is $(x_k + 1, y_k - 1)$ and $p_{k+1} = p_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$
   - Draw all the points, including symmetrical points from other quadrants

Ellipse line drawing algorithm (2)

3. If $r_y^2 x \geq r_x^2 y$, then we must define starting point and initial value of decision parameter for the second part. Remember, we continue from previous part:
   - Starting point is the last calculated (x_k, y_k), because we continue from previous pixel!
   - So we only recalculate the initial parameter for the second part:

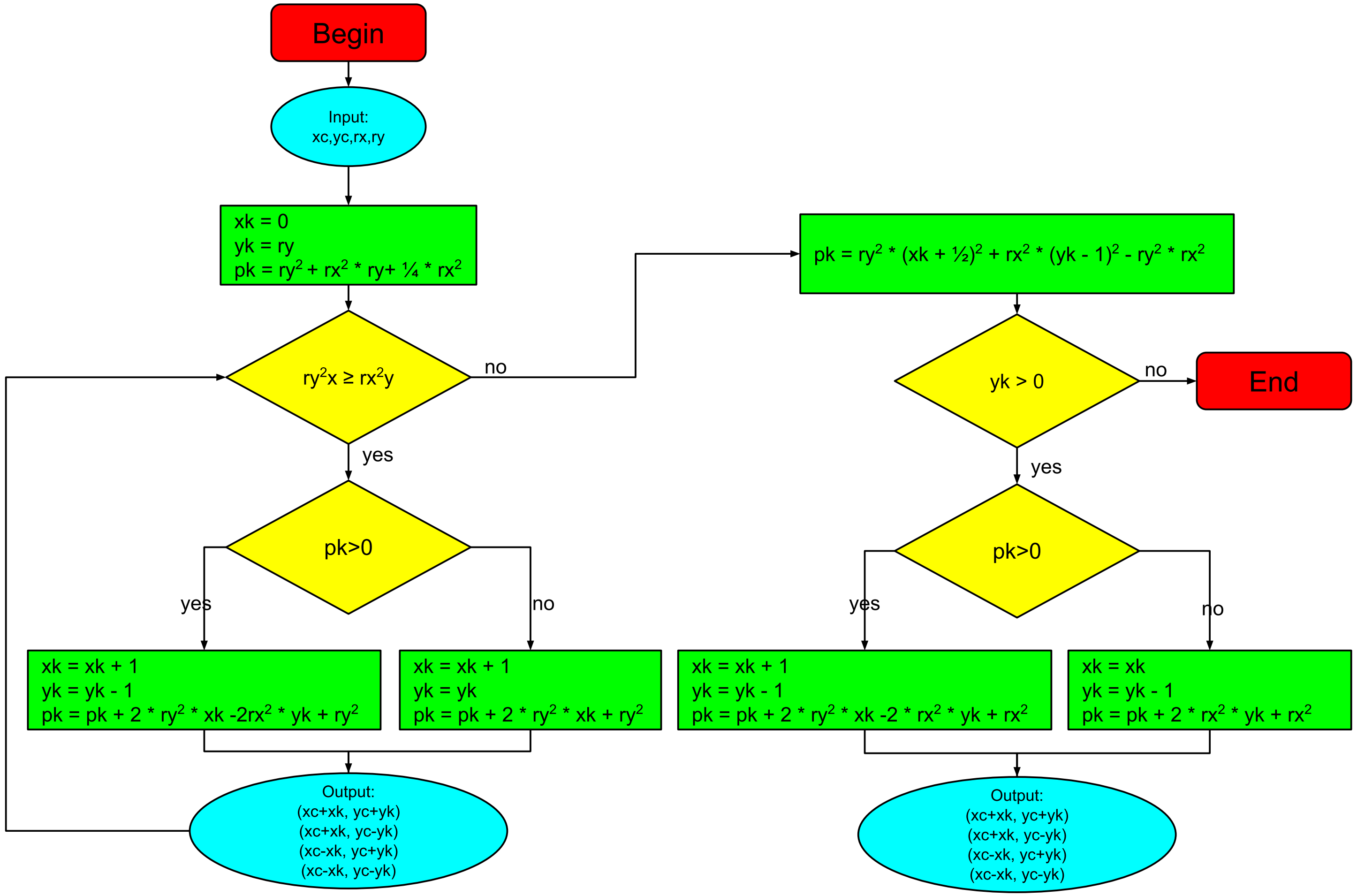
   $p_k = r_y^2\left(x_k + \frac{1}{2}\right)^2 + r_x^2(y_k - 1)^2 - r_x^2 r_y^2$

4. While y>0, for each y_k , starting with $k = 0$:
   - if $p_k > 0$, then the next pixel (x_{k+1}, y_{k+1}) is $(x_k, y_k - 1)$ and

     $p_{k+1} = p_k - 2r_x^2 y_{k+1} + r_x^2$
   - else the next pixel (x_{k+1}, y_{k+1}) is $(x_k + 1, y_k - 1)$ and

     $p_{k+1} = p_k - 2r_x^2 y_{k+1} + r_x^2 + 2r_y^2 x_{k+1}$
   - Draw all the points, including symmetrical points from other quadrants

Ellipse line drawing algorithm



Math calculation of the circle line: example

$x_c = 2, y_c = 3, R = 8$

Initial values:

- $x_0 = 0$
- $y_0 = R = 8$
- $P_0 = 1 - R = 1 - 8 = -7$

While $x < y$, we calculate:

1. $P_0 = -7 < 0 \Rightarrow (x_1, y_1) = (x_0 + 1, y_0) = (0 + 1, 8) = (1, 8)$,
   with central point x_c, y_c: $(x_c + x_1, y_c + y_1) = (2 + 1, 3 + 8) = (3, 11)$
   $is\ x < y?\ 1 < 8\ - yes, continue$
   $P_1 = P_0 + 2x_1 + 1 = -7 + 2 \cdot 1 + 1 = -4$

2. $P_1 = -4 < 0 \Rightarrow (x_2, y_2) = (x_1 + 1, y_1) = (1 + 1, 8) = (2, 8)$,
   with central point x_c, y_c: $(x_c + x_2, y_c + y_2) = (2 + 2, 3 + 8) = (4, 11)$
   $is\ x < y?\ 2 < 8\ - yes, continue$
   $P_2 = P_1 + 2x_2 + 1 = -4 + 2 \cdot 2 + 1 = 1$

3. $P_2 = 1 > 0 \Rightarrow (x_3, y_3) = (x_2 + 1, y_2 - 1) = (2 + 1, 8 - 1) = (3, 7)$
   with central point x_c, y_c: $(x_c + x_3, y_c + y_3) = (2 + 3, 3 + 7) = (5, 10)$
   $is\ x < y?\ 3 < 7\ - yes, continue$
   $P_3 = P_2 + 2x_3 - 2y_3 + 1 = 1 + 2 \cdot 3 - 2 \cdot 7 + 1 = -6$

Math calculation of the circle line: example

4. $P_3 = -6 < 0 \Rightarrow (x_4, y_4) = (x_3 + 1, y_3) = (3 + 1{,}7) = (4, 7)$

with central pointx_c, y_c: $(x_c + x_4, y_c + y_4) = (2 + 4, 3 + 7) = (6{,}10)$
$is\ x < y?\ 4 < 7\ - yes, continue$
$P_4 = P_3 + 2x_4 + 1 = -6 + 2 \cdot 4 + 1 = 3$

5. $P_4 = 3 > 0 \Rightarrow (x_5, y_5) = (x_4 + 1, y_4 - 1) = (4 + 1, 7 - 1) = (5, 6)$

with central point x_c, y_c: $(x_c + x_5, y_c + y_5) = (2 + 5, 3 + 6) = (7{,}9)$
$is\ x < y?\ 5 < 6\ - yes, continue$
$P_5 = P_4 + 2x_5 - 2y_5 + 1 = 3 + 2 \cdot 5 - 2 \cdot 6 + 1 = 2$

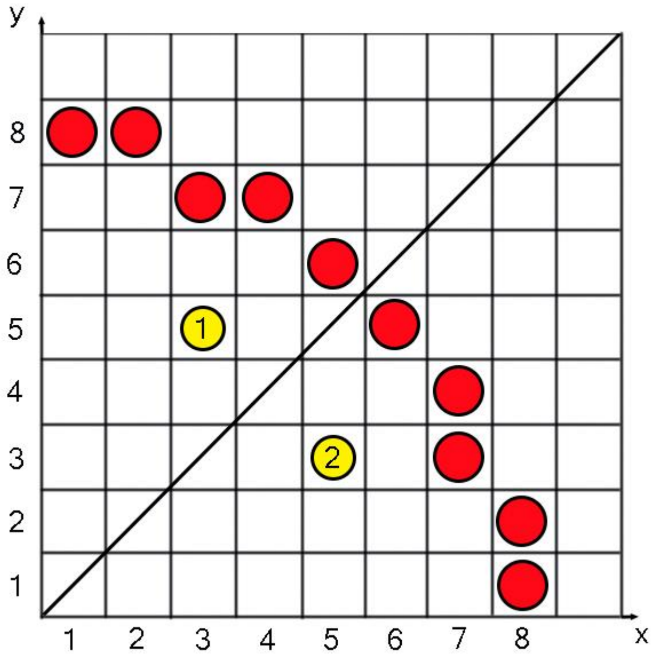
6. $P_5 = 2 > 0 \Rightarrow (x_6, y_6) = (x_5 + 1, y_5 - 1) = (5 + 1, 6 - 1) = (6, 5)$

with central pointx_c, y_c: $(x_c + x_6, y_c + y_6) = (2 + 6, 3 + 5) = (8{,}8)$
$is\ x < y?\ 6 > 5\ - no, stop!$

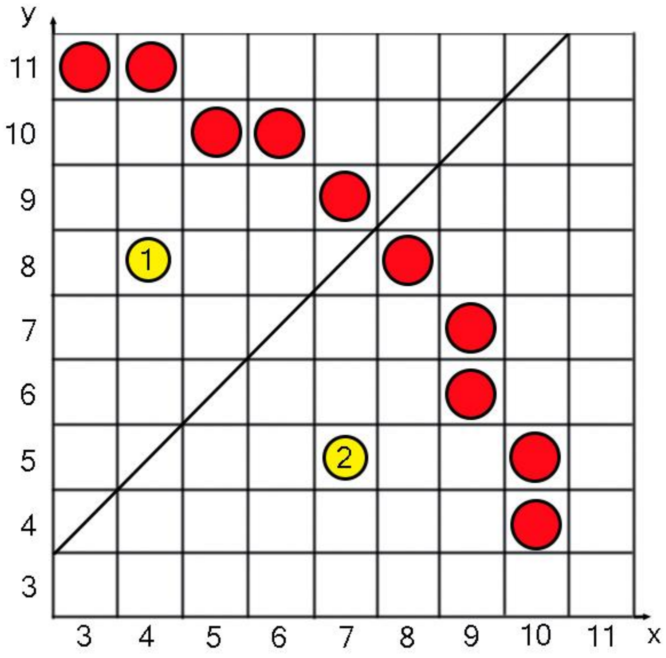
Remember, when $x \geq y$ – we STOP!

Math calculation of the circle line: example

THIS IS HOW THE CALCULATED POINTS LOOK LIKE



AND THIS TAKES INTO ACCOUNT THE CENTRAL POINT COORDINATES



Midpoint circle line drawing algorithm

Input: x_c,y_c,R

1. Initial values for the 0th step:
   - $x_k, y_k = (0, R)$
   - $p_k = 1 - R$
2. While $x_k < y_k$, *{WHILE}* repeat:
   1. If $p_k < 0$ then: *{IF}*

      $x_{k+1} = x_k + 1$ and

      $p_{k+1} = p_k + 2x_{k+1} + 1$
   2. If $p_k \geq 0$ then: *{ELSE}*

      $x_{k+1} = x_k + 1$,

      $y_{k+1} = y_k - 1$ and

      $p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$
   3. Draw pixel x_k, y_k symmetrically in 8 parts, taking the circle's central point into account.