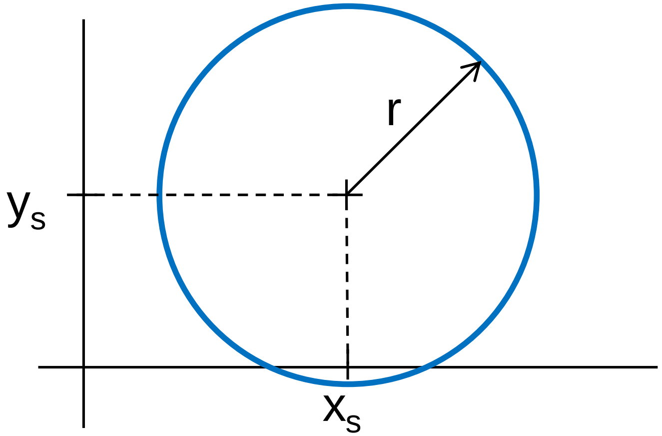
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

APG – Digitizing circle

JIŘÍ ŽÁRA

Description of a circle – option 1



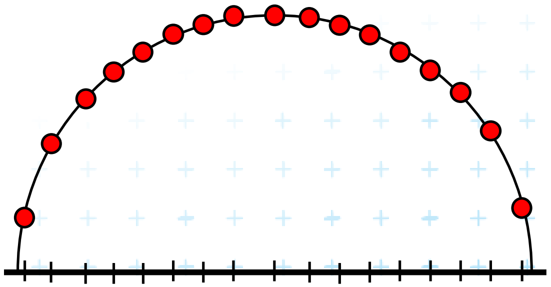
$$(x-x_s)^2 + (y-y_s)^2 = r^2$$

- a) from implicit equation

for (int x = xs - r; i < xs + r; i++) {

$$\boldsymbol{y = ys \pm \sqrt{r^2 - (x - xs)^2}}$$

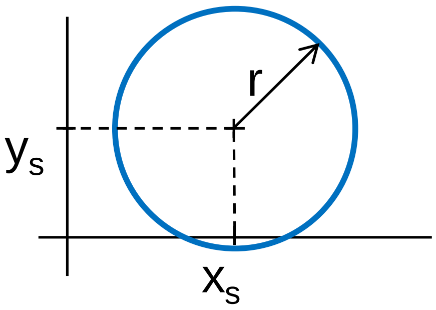
}



Example of wrong sampling

Description of a circle – option 2

- b) Parametric description



for (int φ = 0; φ < 2π; φ += step){

$$\boldsymbol{x = xs + r \cdot \cos\varphi}$$

$$\boldsymbol{y = y_s + r \cdot \sin\varphi}$$

}

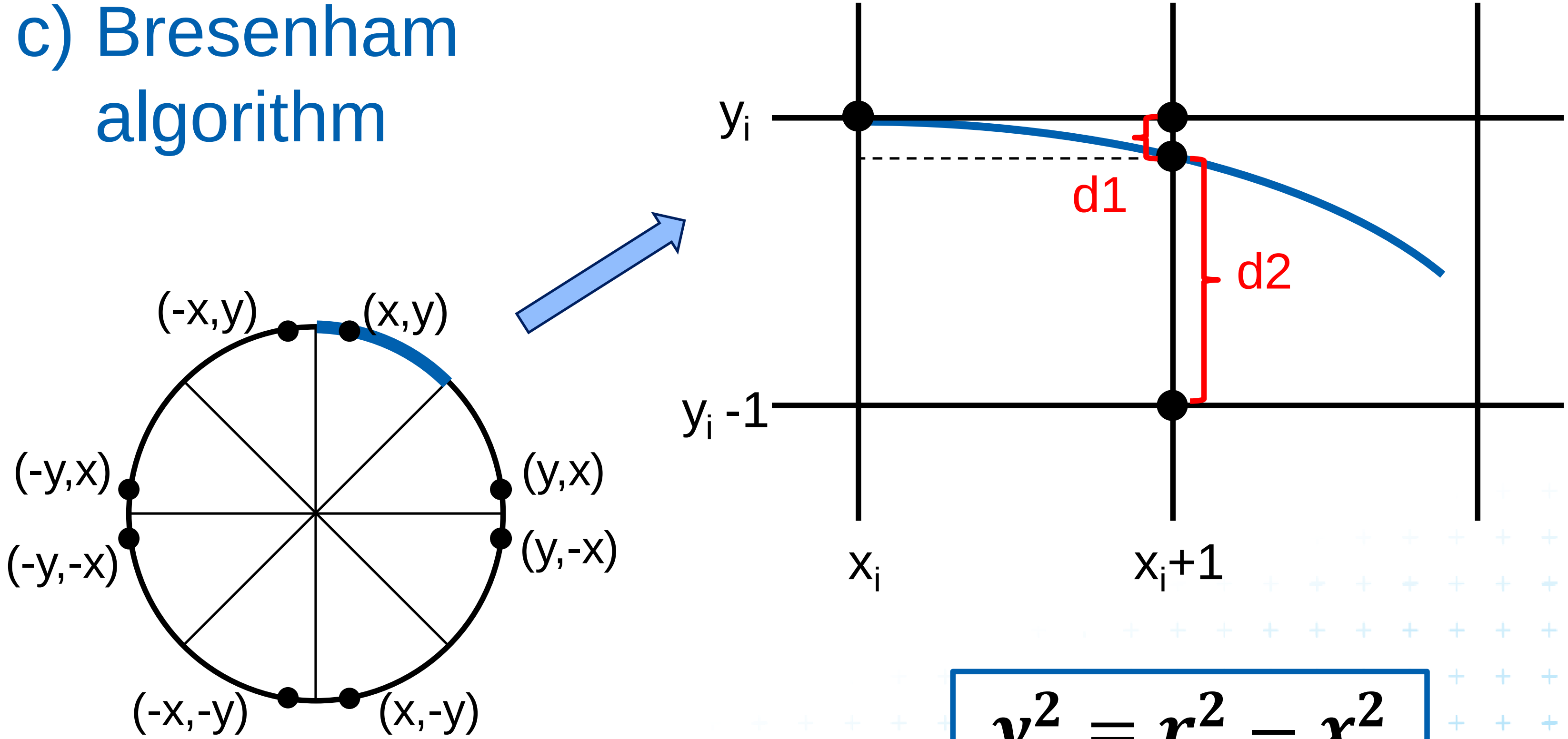
Notes:

step $(\sim 1/r)$ … float number

goniometric function … in a loop body

Digitizing circle

- c) Bresenham algorithm

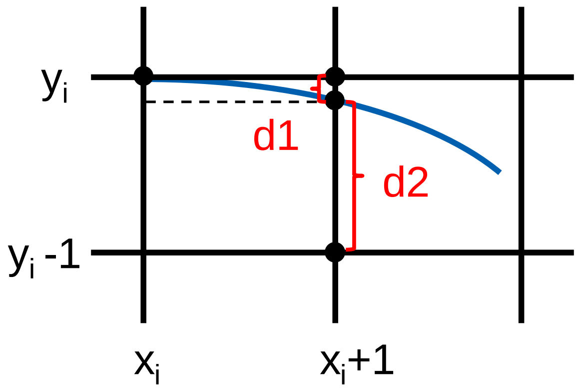


$$\boxed{y^2 = r^2 - x^2}$$

Idea: Use squares (power of two) instead of square roots

Bresenham – circle (1/2)

$$D_1 = {y_i}^2 - y^2 = {y_i}^2 - r^2 + (x_i \ + 1)^2$$
$$D_2 = y^2 - (y_i - 1)^2 =$$
$$= r^2 - \left(x_i \ + 1\right)^2 - \left(y_i \ - 1\right)^2$$



$$p_i = D_1 - D_2$$
$$= 2 \cdot (x_i \ + 1)^2 + {y_i}^2 + (y_i \ - 1)^2 - 2 \cdot r^2$$

Note: D_1 is not exactly d_1

$$p_{i+1} = p_i + 4 \cdot x_i + 6 + 2 \cdot \left({y_{i+1}}^2 - {y_i}^2\right) - 2 \cdot (y_{i+1} - y_i)$$

$$[x_0, y_0] = [0, r] \Rightarrow p_0 = 3 - 2r$$

Bresenham – circle (2/2)

$$[x_i, y_i, p_i] \rightarrow [x_{i+1}, y_{i+1}, p_{i+1}]$$

$a)\ \ x_{i+1} = x_i + 1$

$b)\ \ p_i < 0 \Rightarrow\ y_{i+1} = y_i$

$\qquad\qquad p_{i+1} = p_i + 4 \cdot x_i + 6$

$\quad p_i \geq 0 \Rightarrow y_{i+1} = y_i - 1$

$\qquad\qquad p_{i+1} = p_i + 4 \cdot (x_i - y_i) + 10$

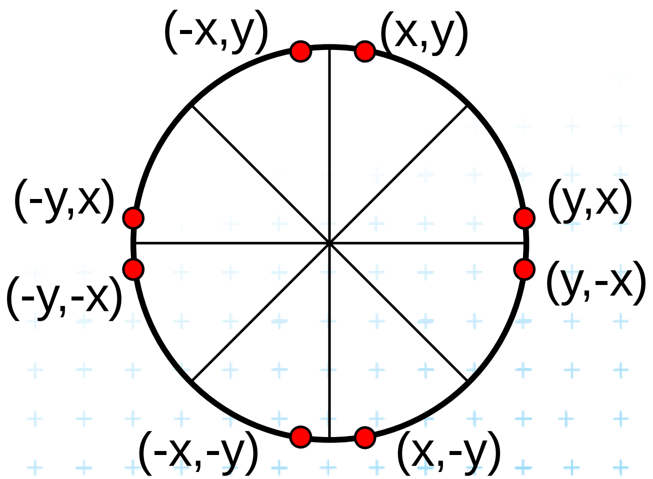
integer, +, -, shift

Tip: Instead of multiplication by 4 (i.e. „ $4 \cdot x_i$ "), keep additional variable incremented by 4 in a loop.

Bresenham - code

```
Bres_circle(int xs, int ys, int r) {
    int x, y, p, fourX, fourY;
    x = 0; y = r;
    p = 3 - 2*r;
    fourX = 0; fourY = 4*r;
    while (x <= y) {
        set_sym_pixel (x,y);
        if (p > 0) {
            p = p - fourY + 4;
            fourY = fourY - 4;
            y = y - 1;
        }
        p = p + fourX + 6;
        fourX = fourX + 4;
        x = x + 1;
    }
}
```

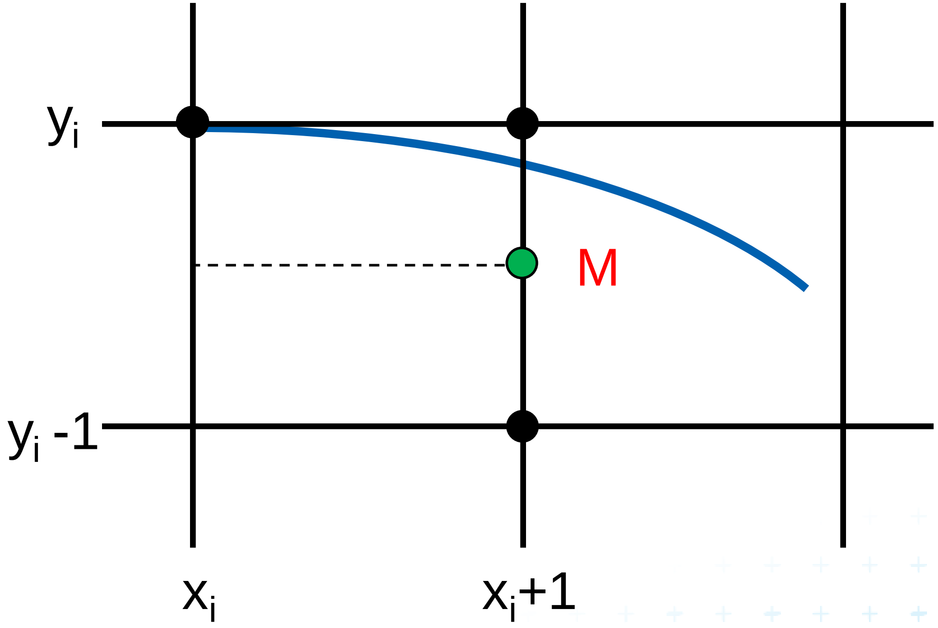
```
set_sym_pixel (int x, int y) {
    // 8 symmetrical pixels
    // around the center xs, ys
}
```



Variant: d) Midpoint algorithm (1/2)

$$F = x^2 + y^2 - r^2$$

- F=0, on circle
- F<0, inside
- F>0, outside

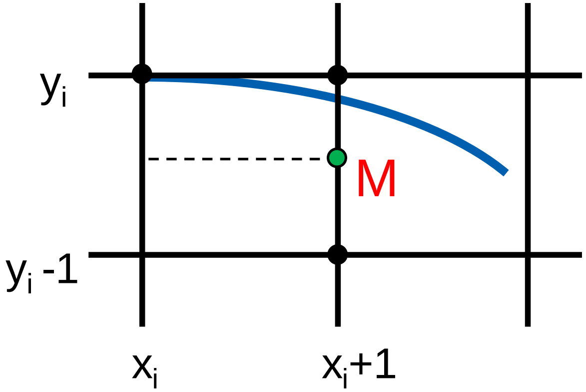


$$\boxed{F(M) = (x_i + 1)^2 + (y_i - 1/2)^2 - r^2}$$

Midpoint algorithm (2/2)

$$p_i = F(M) = (x_i + 1)^2 + (y_i - 1/2)^2 - r^2$$

$$p_{i+1} = (x_i + 2)^2 + (y_{i+1} - 1/2)^2 - r^2$$



$$p_{i+1} = p_i + 2x_i + (y_{i+1})^2 - (y_i)^2 - y_{i+1} + y_i + 3$$

$$[x_0, y_0] \Rightarrow F[0, r] \Rightarrow p_0 = 1 - r + \cancel{1/4}$$ ← **Ignore!**

$$p_i < 0 \Rightarrow y_{i+1} = y_i$$
$$p_{i+1} = p_i + 2 \cdot x_i + 3$$

$$p_i \geq 0 \Rightarrow y_{i+1} = y_i - 1$$
$$p_{i+1} = p_i + 2 \cdot x_i + 3 - 2 \cdot y_i + 2$$

DCGI

Midpoint algorithm - code

```
Midpoint_circle(int xs, int ys, int r) {
    int x, y, p, twoX, twoY;
    x = 0; y = r;
    p = 1 - r;
    twoX = 0; twoY = 2*r;
    while (x <= y) {
        set_sym_pixel (x,y);
        if (p > 0) {
            p = p - twoY + 2;
            twoY = twoY - 2;
            y = y - 1;
        }
        p = p + twoX + 3;
        twoX = twoX + 2;
        x = x + 1;
    }
}
```

Further efficiency improvements
(move constants from a loop
to the initial values)

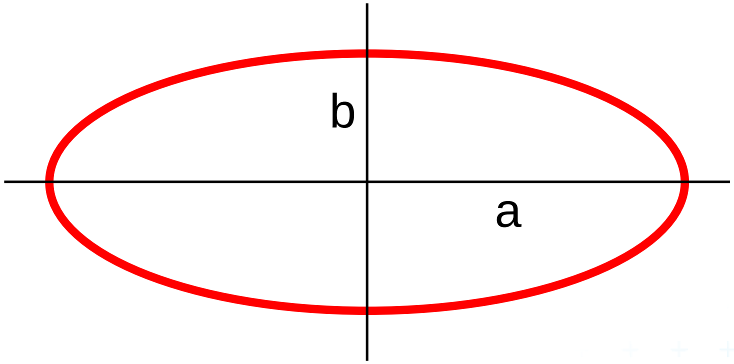
Ellipse

- Raster is not always equidistant grid

  $\Longrightarrow$ circle turns to ellipse (and vice versa)

- Ellipse equation:

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



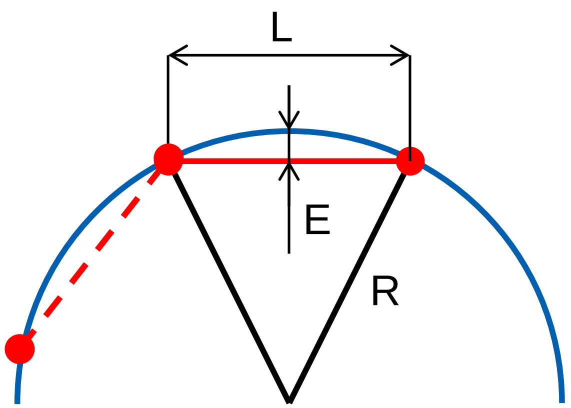
- Bresenham/Midpoint:

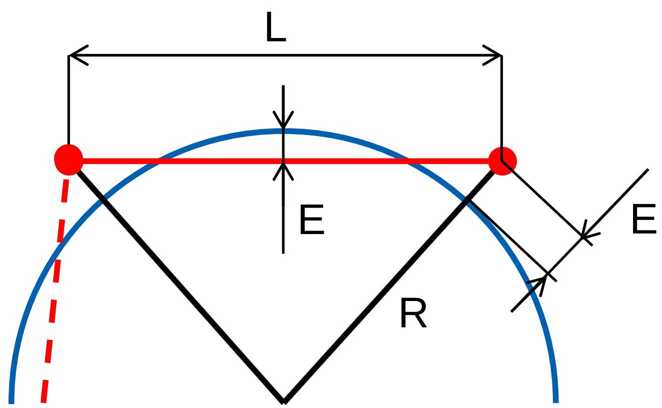
$$y^2 = b^2 \cdot \left(1 - \frac{x^2}{a^2}\right)$$

$$p_{i+1} = p_i + \cdots$$

Fast drawing circle as a polyline

- Rotate line segment: length L, tolerance E





Circumscribed circle

$$L = \sqrt{8 \cdot R \cdot E}$$

Inscribed circle

$$L = 4 \cdot \sqrt{R \cdot E}$$

- Number of line segments = **N** $\sim round(2\pi \cdot \frac{R}{L})$

Implementation – Naive solution

```
x1 = R;
y1 = 0;
alpha = 2 · pi / N;      // circle replaced by N line segments


for (int i = 1; i < N; i++) {
    x2 = R · cos (i · alpha);
    y2 = R · sin (i · alpha);

    Draw_Line (x1, y1, x2, y2);
    x1 = x2;
    y1 = y2;
}
```

Goniometric functions in a loop body

Implementation – Rotation transformation

```
x1 = R;
y1 = 0;
alpha = 2 · pi / N;
CA = cos (alpha); SA = sin (alpha);

for (int i = 1; i < N; i++) {

    x2 = CA · x1 – SA · y1;
    y2 = SA · x1 + CA · y1;

    Draw_Line (x1, y1, x2, y2);
    x1 = x2;
    y1 = y2;
}
```

Rotation in 2D:

$$[x2, y2] = [x1, y1] \cdot \begin{bmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{bmatrix}$$

Implementation – Variation

```
x1 = R;
y1 = 0;
alpha = 2 · pi / N;
SA = sin (alpha);

for (int i = 1; i < N; i++) {
    x2 =        x1 – SA · y1;
    y2 = SA · x2 +       y1;

    úsečka(x1, y1, x2, y2);
    x1 = x2;
    y1 = y2;
}
```

- Approximate values
- Less multiplications

Correction of a spiral

Thank you for your attention

Jiří Žára, 15.12.2020