Generování základních objektů v rastru

2. cvičení předmětu IZG

Jan Tomešek

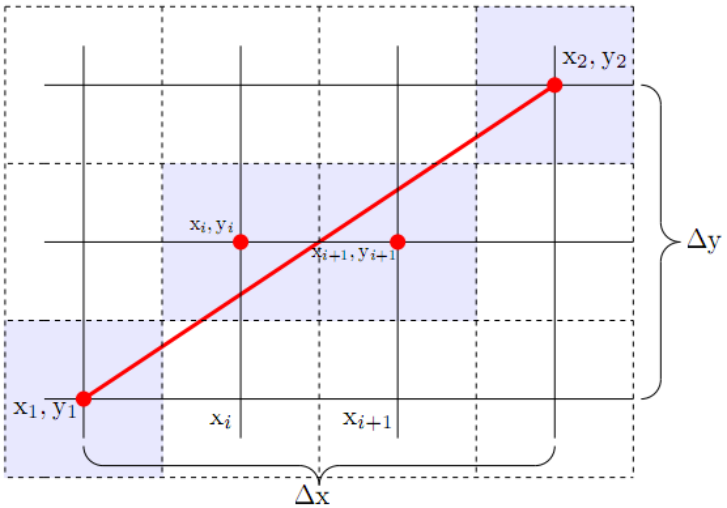
itomesek@fit.vutbr.cz

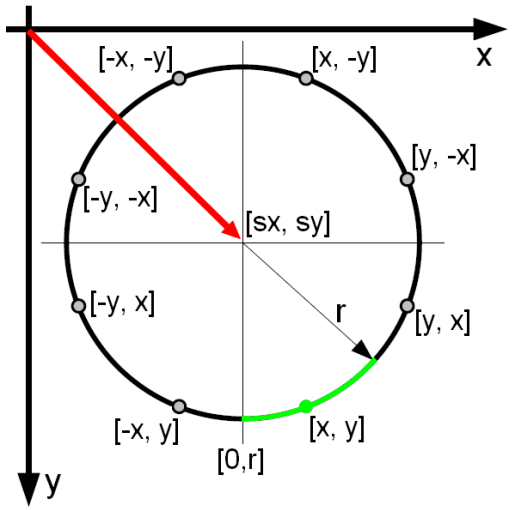
(s využitím materiálů M. Španěla, M. Šolonyho, M. Veľase a M. Kuli)

26. února 2018

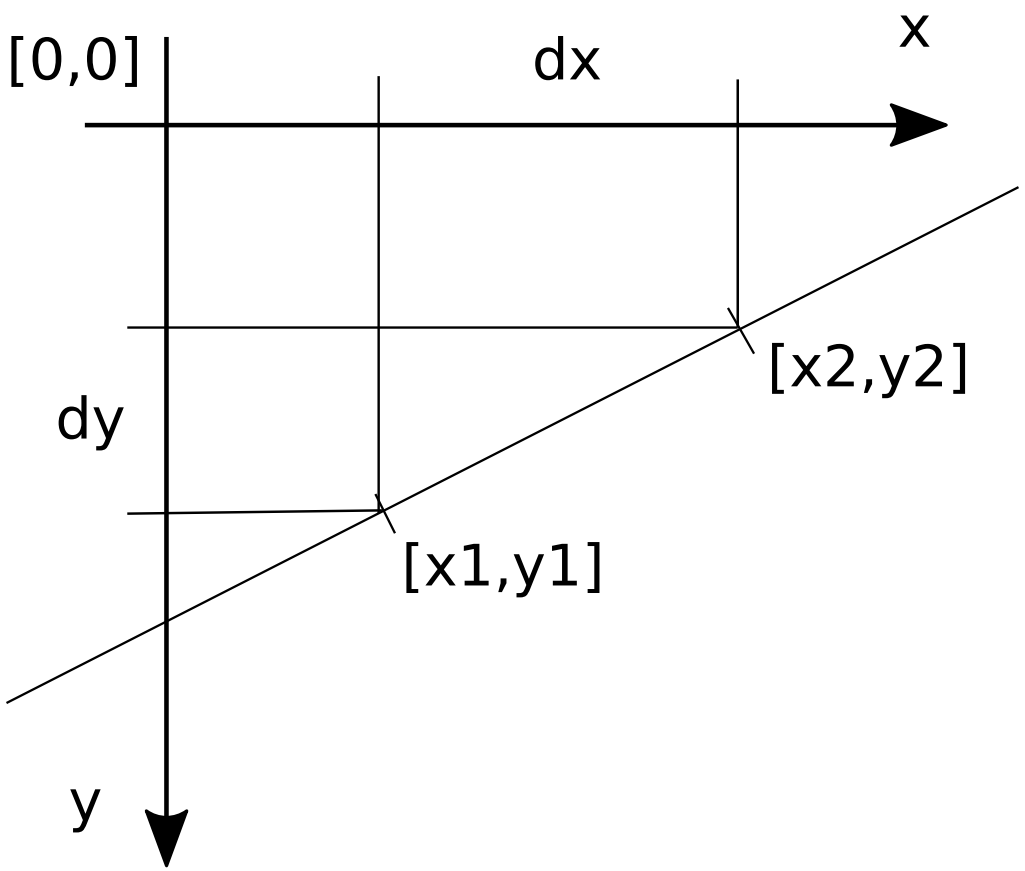
Náplň cvičení

1. rasterizace *úsečky*
   - přehled algoritmů
   - samostatná úloha – DDA algoritmus s fixed-point aritmetikou
     - 2 body
2. rasterizace *kružnice*
   - přehled algoritmů
   - samostatná úloha – MidPoint algoritmus
     - 1 bod





Popis přímky v rovině



Směrnicový tvar přímky

- $y = kx + q$
- směrnice $k = \frac{dy}{dx} = \tan(\alpha)$

Algoritmy

Algoritmy rasterizace úsečky

- DDA
- DDA s kontrolou chyby
- Bresenhamův algoritmus
- DDA s fixed-point aritmetikou

Algoritmy

Algoritmy rasterizace úsečky

- DDA ✗
- DDA s kontrolou chyby ✗
- Bresenhamův algoritmus ✓
- DDA s fixed-point aritmetikou ✓

DDA - Digital Differential Analyser

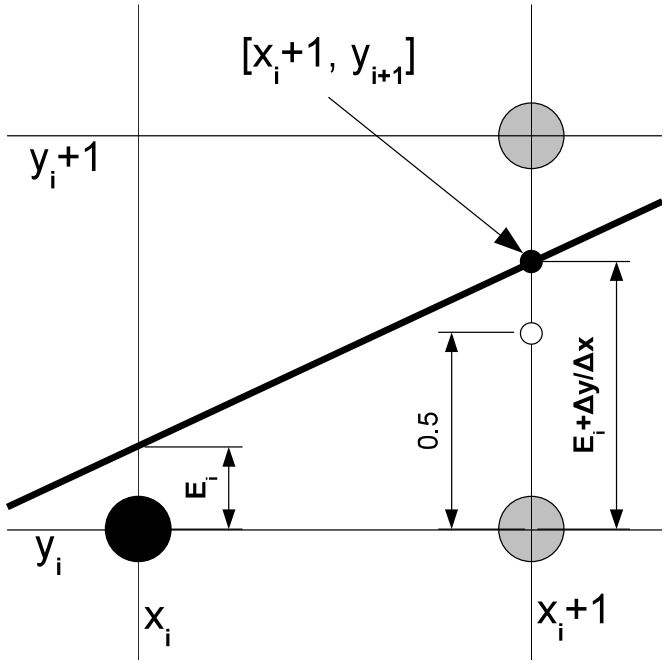
Vlastnosti

- jednoduchý
- používá floating-point aritmetiku ⇒ neefektivní

```
LineDDA(int x1, int y1, int x2, int y2) {
    const double k = (y2-y1) / (x2-x1);
    double y = y1;
    for(int x = x1; x <= x2; x++) {
        putPixel(x, round(y));
        y += k;
    }
}
```

- modifikace: DDA s kontrolou chyby – stále floating-point

Bresenhamův algoritmus



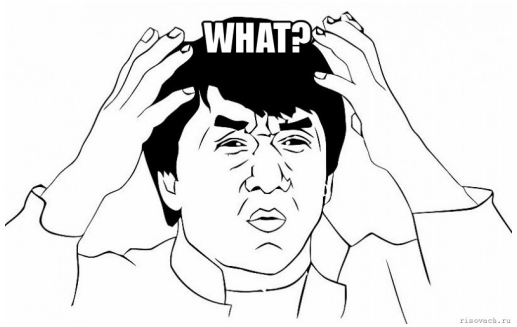
Rozhodování a výpočet chyby vykreslování E

$$E_i + \frac{\Delta y}{\Delta x} \begin{cases} < 0.5 & krok\ (x_i + 1, y_i) \quad E_{i+1} = E_i + \frac{\Delta y}{\Delta x} \\ \geq 0.5 & krok\ (x_i + 1, y_i + 1) \quad E_{i+1} = E_i + \frac{\Delta y}{\Delta x} - 1 \end{cases}$$

Bresenhamův algoritmus

```
LineBresenham(int x1, int y1, int x2, int y2) {
    const int dx = x2-x1 , dy = y2-y1;
    const int P1 = 2*dy, P2 = P1 - 2*dx;
    int P = 2*dy - dx;
    int y = y1;
    for (int x = x1 ; x <= x2 ; x++) {
        putPixel(x, y);
        if (P >= 0) { P += P2 ; y++; }
        else        { P += P1 ; }
}
```

- *prediktor* $\Rightarrow$ jen celá čísla, efektivní, ale ...



DDA s fixed-point aritmetikou

Zlatý Grál rasterizace úsečky

- kódovaní do celých čísel (fixed-point)
- rychlý, efektivní, používaný v GPU
- jednoduchý (téměř)

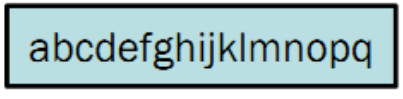
```
#define FRAC_BITS 8
LineDDAFixedPoint(int x1, int y1, int x2, int y2) {
    int y = y1 << FRAC_BITS;
    const int k = ((y2-y1) << FRAC_BITS)/(x2-x1);
    for(int x = x1; x <= x2; x++) {
        putPixel(x, y >> FRAC_BITS);
        y += k;
    }
}
```

DDA s fixed-point aritmetikou

Floating-point aritmetika

- S . . . znaménko
- M . . . mantisa
- E . . . exponent

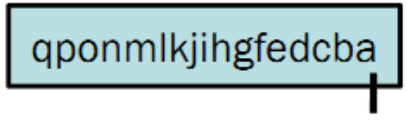
$$X = S \cdot M \cdot 2^E$$



=0,abcdefghijklmnopq

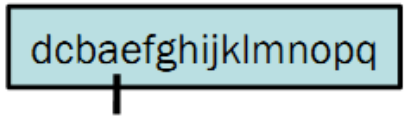
=a*1/2 + b*1/4 + c*1/8 + d*1/16 + …

Fixed-point aritmetika



=qponmlkjihgfedcba,0

=a*1 + b*2 + c*4 + d*8 + …



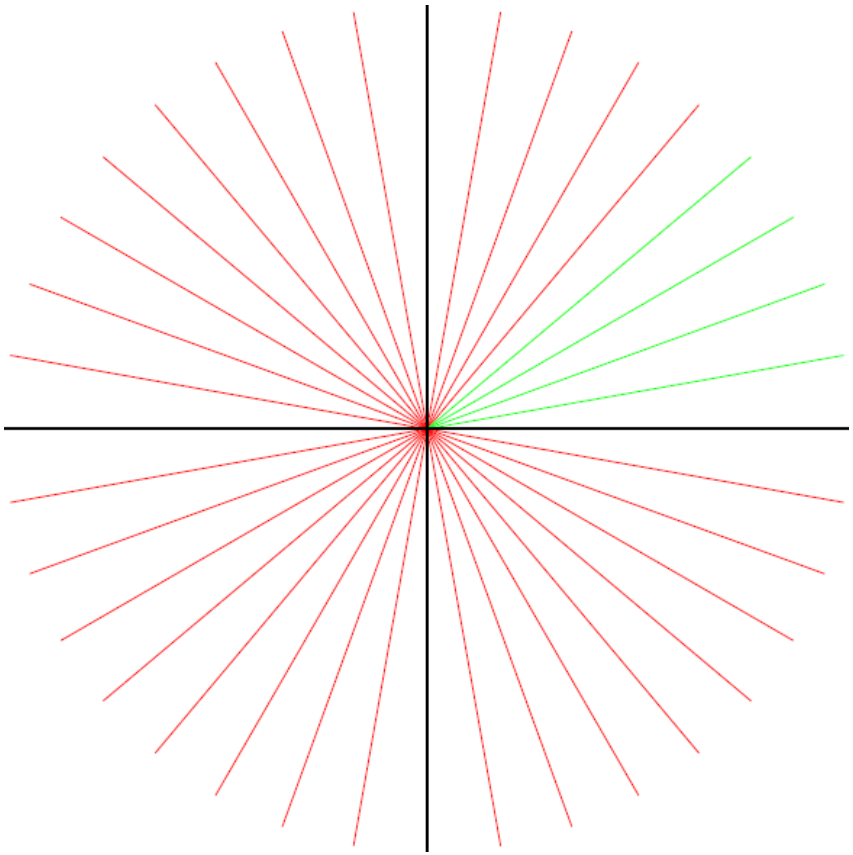
=dcba,efghijklmnopq

=a*1 + b*2 + c*3 + d*4 +
e*1/2 + f*1/4 + g*1/8 + h*1/16 + …

Problém uvedených algoritmů

Problém

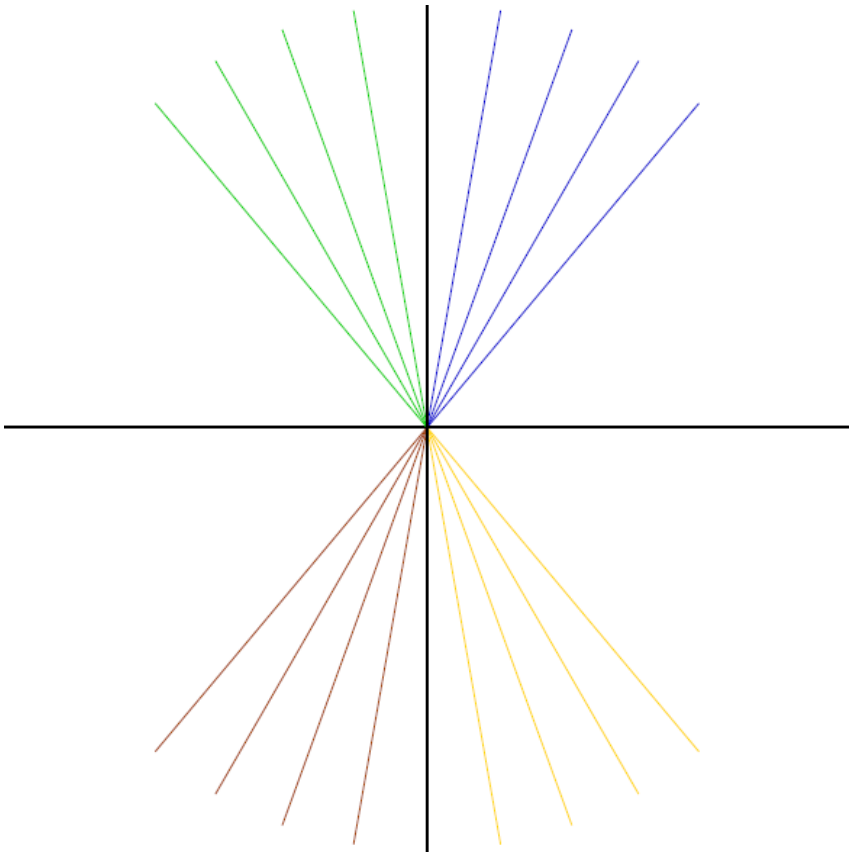
- většina algoritmů jen pro první polovinu *I*. kvadrantu
- potřebujeme vykreslení úsečky v libovolném směru
- zadání první bodované úlohy



Úsečka rychle roste/klesá

$|\mathbf{dy}| > |\mathbf{dx}|$

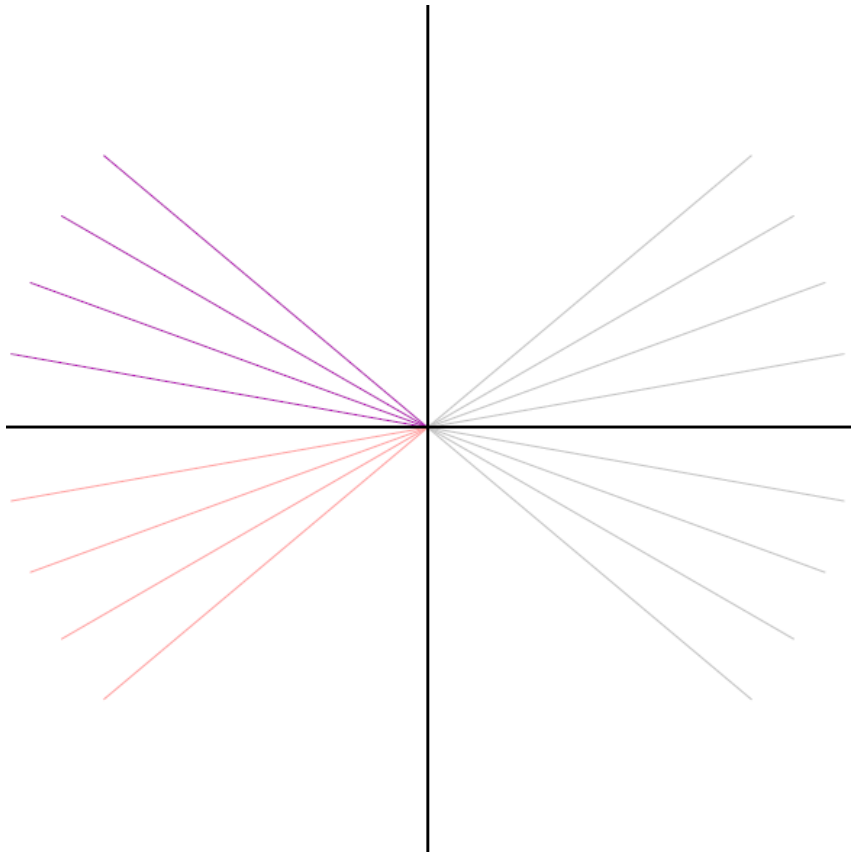
- výměna X-ových a Y-ových souřadnic
- znovu vyměnit při zápisu do framebufferu (`putPixel`)



Úsečka směřuje zprava doleva

$x_1 > x_2$

- výměna počátečního a koncového bodu
- zbytek beze změny



Počáteční bod shodný s koncovým

$[x1, y1] = [x2, y2]$

- není úsečka, ale jen 1 bod
- `Division by zero!`

Úloha 1 (2b)

Doplnění a úprava funkce `drawLine()`

- v souboru `student.cpp`
- vykreslení úsečky algoritmem DDA s FX aritmetikou každým směrem
- funkce nespadne ☺

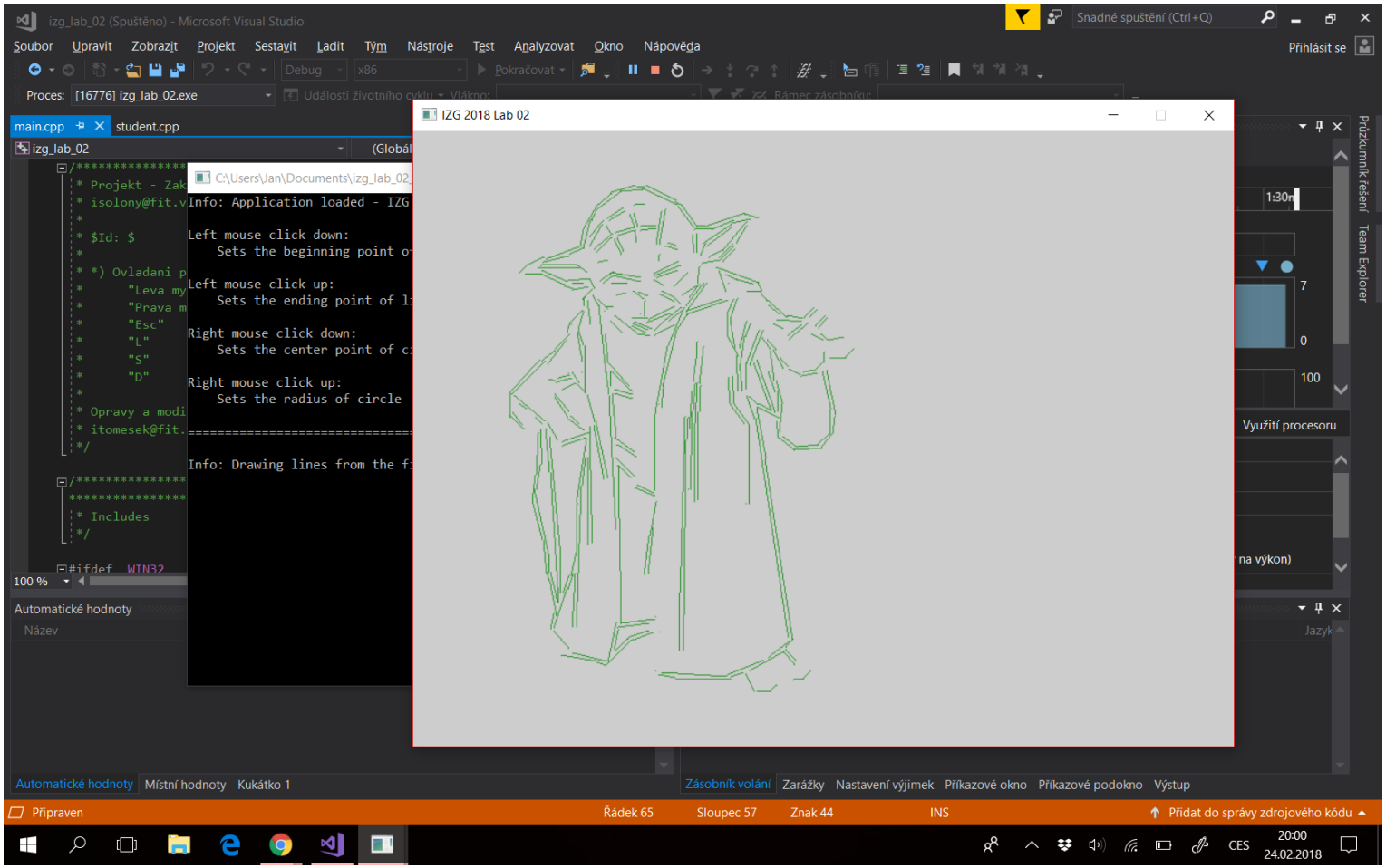
Tipy

- vykreslení pomocí levého tlačítka myši
- makro `SWAP(a,b)` pro výměnu,
  funkce `putPixel(x,y,color)` pro zápis do FB
- dodržujte pořadí uvedených úprav

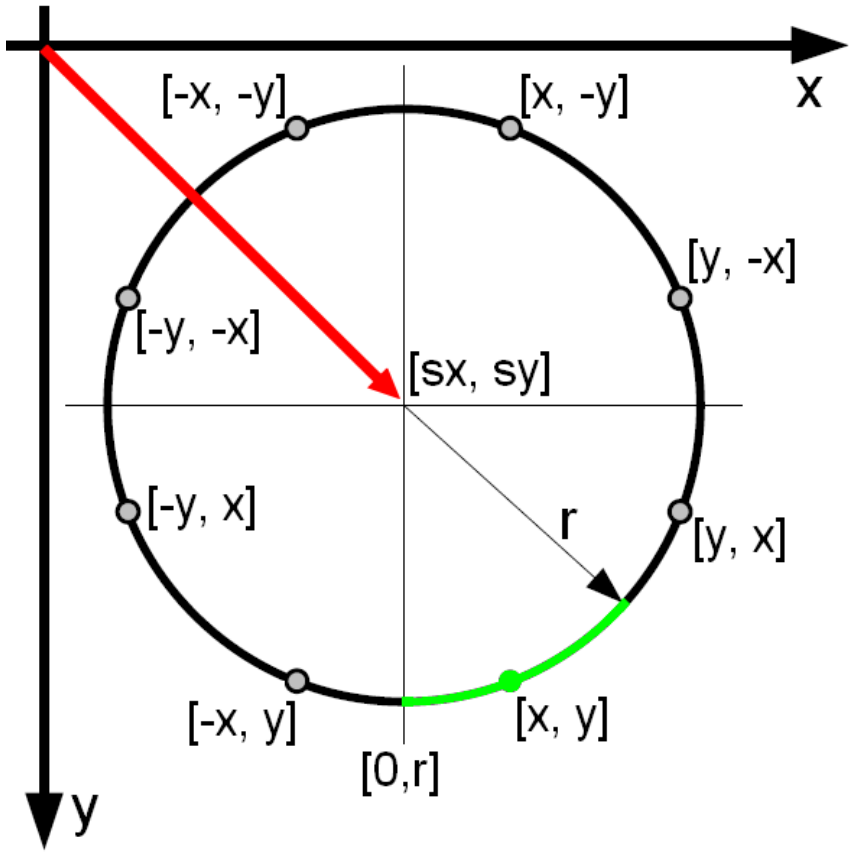
- a jak bude všechno fungovat ...

Found the right solution, you have, hmmm?

- ... a máte představivost, vykreslí se Master Yoda
- klávesa "D"



Popis kružnice v rovině



Středová rovnice kružnice v rovině

- $(x - s_x)^2 + (y - s_y)^2 = r^2$
- střed kružnice $[s_x, s_y]$ a poloměr r

Algoritmy rasterizace kružnice

- Vykreslení kružnice po bodech
- Vykreslení křužnice jako N-úhelník
- MidPoint algoritmus

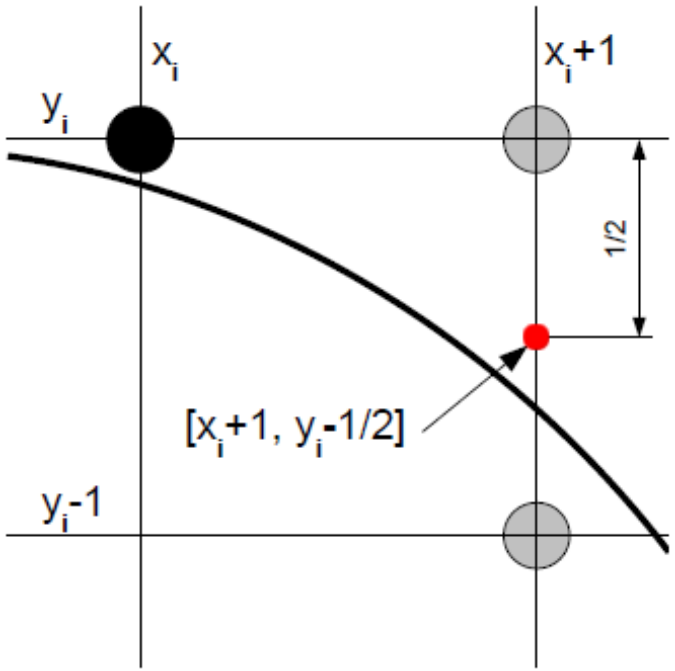
Algoritmy rasterizace kružnice

- Vykreslení kružnice po bodech ✗
- Vykreslení křužnice jako N-úhelník ✗
- MidPoint algoritmus ✓

MidPoint algoritmus

Vlastnosti

- alá Bresenhamův algoritmus
- prediktor, celočíselná aritmetika



MidPoint algoritmus

Vlastnosti

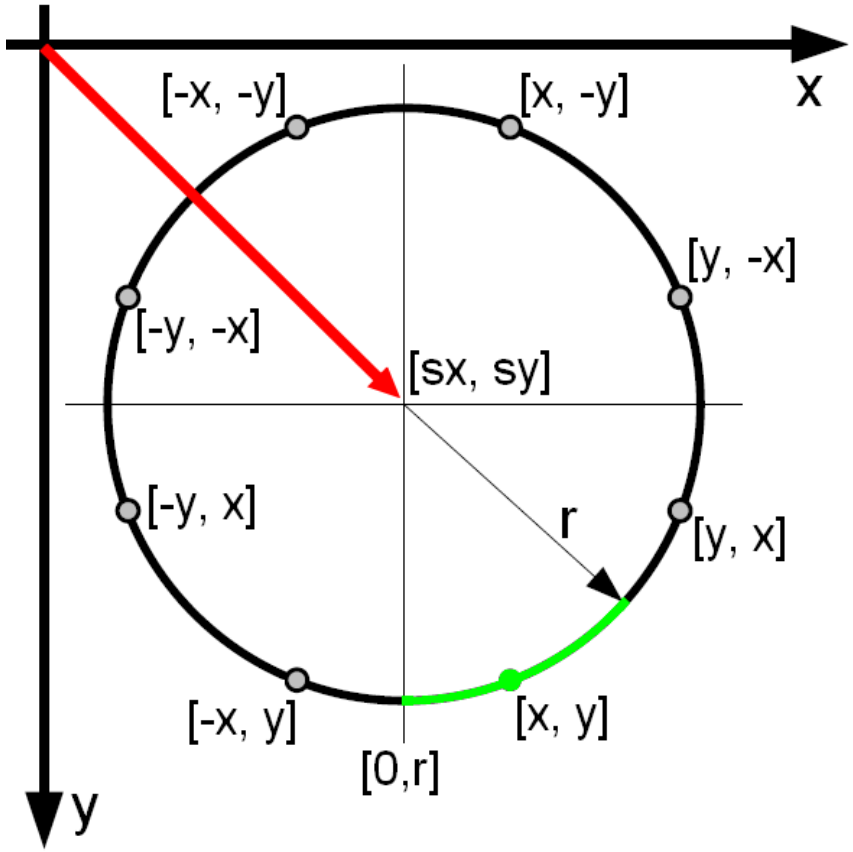
- alá Bresenhamův algoritmus
- prediktor, celočíselná aritmetika

```
void drawCircleMidpoint(int sx, int sy, int R) {
    int x = 0, y = R;
    int P = 1-R, X2 = 3, Y2 = 2*R-2;
    while (x <= y) {
        if (P >= 0) {
            P += -Y2; Y2 -= 2;
            y--;
        }
        P += X2; X2 += 2;
        x++;
    }
}
```

MidPoint algoritmus

Potřebné úpravy

1. celá kružnice $\Rightarrow$ záměna $x \Leftrightarrow y$ a znamínek
2. střed kružnice $[s1, s2] \Rightarrow$ posun každého vykreslovaného bodu



Úloha 2 (1b)

Doplnění těla funkce `put8PixelsOfCircle()`

- v souboru `student.cpp`
- vykreslení celé kružnice (všechny osminy kružnice)

Tipy

- vykreslení pomocí pravého tlačítka myši
- použijte makro `SWAP` a funkci `putPixel()`