

# Generování Základních Objektů v Rastru

## 2. cvičení z předmětu IZG

Tomáš Polášek

[ipolasek@fit.vutbr.cz](mailto:ipolasek@fit.vutbr.cz)

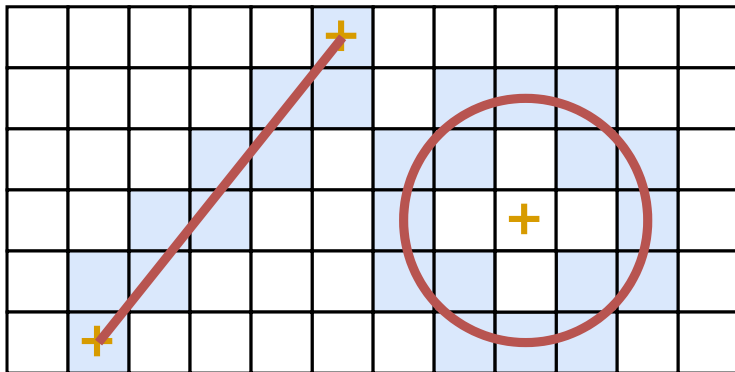
Fakulta informačních technologií Vysokého učení technického v Brně

Božetěchova 1/2. 612 66 Brno - Královo Pole

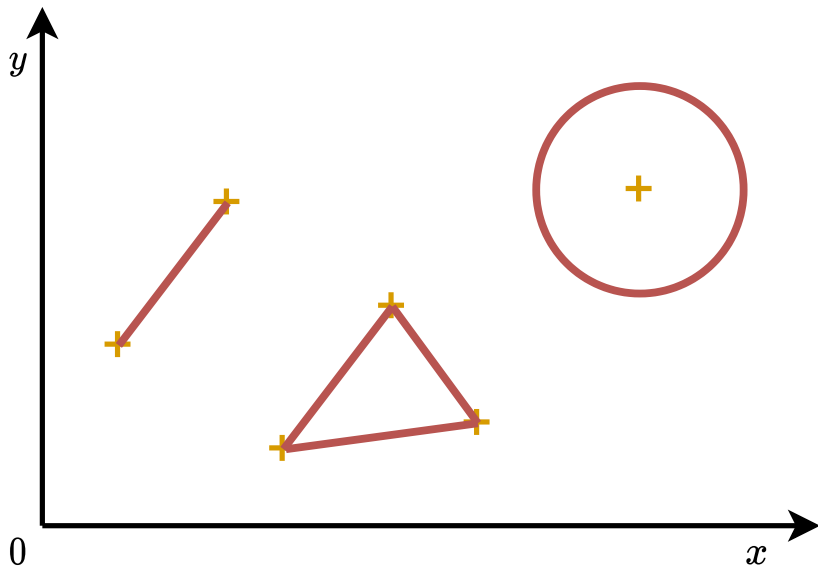


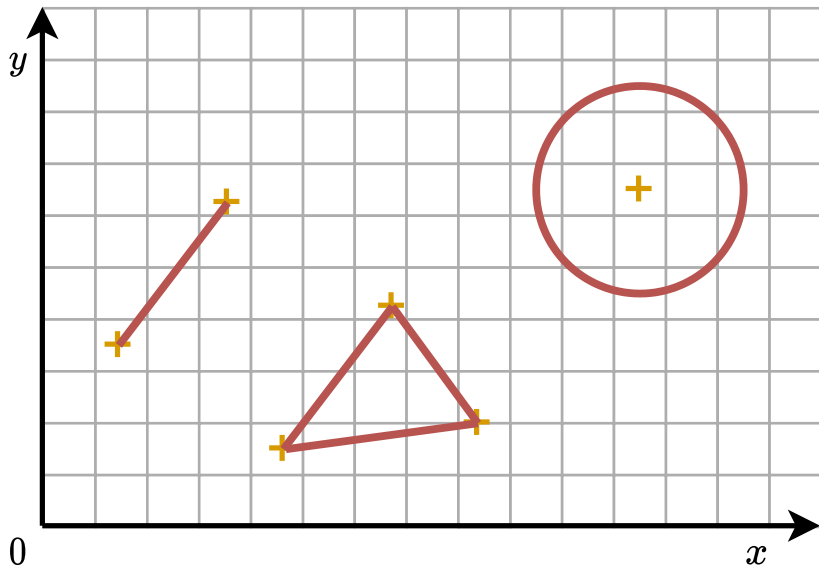
8. března 2022

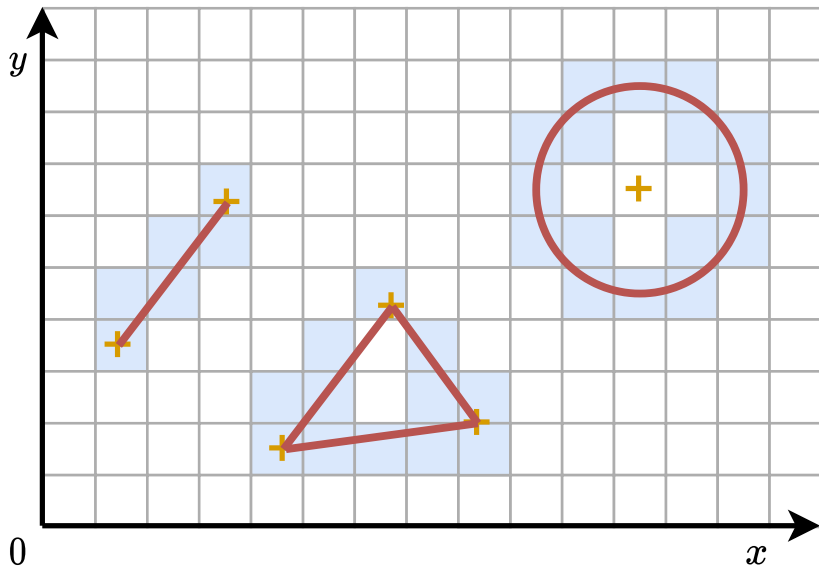
- 1 Osnova cvičení
- 2 Přehled problematiky
- 3 Rasterizace úsečky:
  - Samostatná úloha (2b) *FX DDA*
- 4 Rasterizace kružnice:
  - Samostatná úloha (1b) *MidPoint*

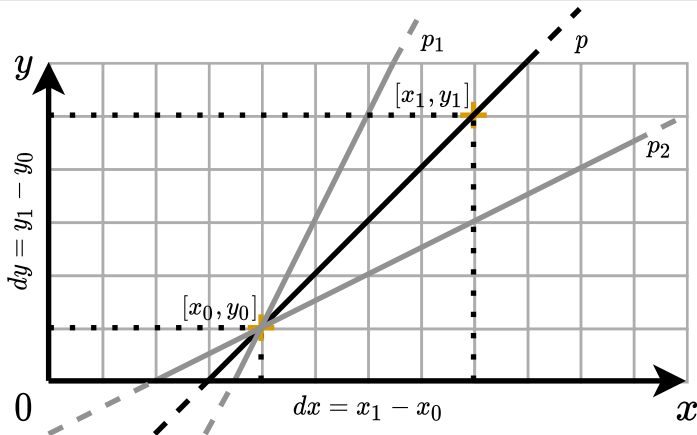












## Směrnice tvar **přímky**

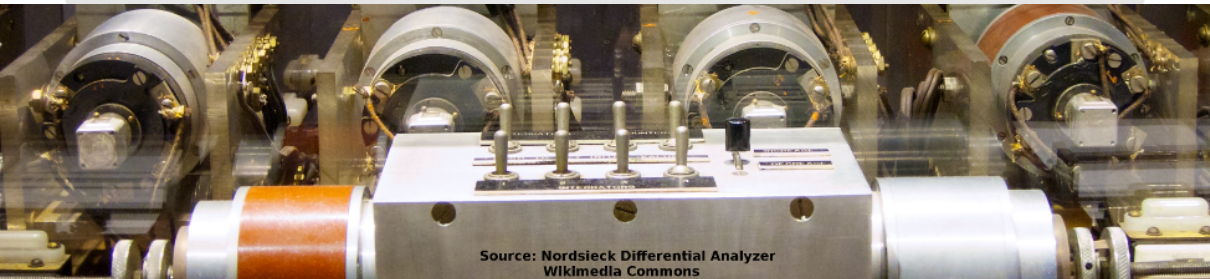
- Vektorový popis:  $y = kx + q$ , kde  $k = \frac{dy}{dx}$  je směrnice přímky
- „Iterace“ přes  $x$

## Cíle

- Vektorový  $\Rightarrow$  Rastrový
- Efektivní & Jednoduchý

## Algoritmy

- 1 Digital Differential Analyzer (DDA) + EC
- 2 Bresenhamův algoritmus
- 3 Fixed Point Digital Differential Analyzer (FX DDA)





## Vlastnosti

- Efektivní (?) & Jednoduchý
- Floating Point (FP) operace
- Alternativa s kontrolou chyby  
( $E \geq 0.5 \Rightarrow y = y + 1; E = E - 1$ )

```
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)  
    { setPixel(x, round(y)); y += k; }  
}
```

## Vlastnosti

- Efektivní HW implementace, složitější na pochopení
- Celočíselné operace
- Prediktor

```
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, y = y1;  
    for (int x = x1; x <= x2; ++x) {  
        setPixel(x, y);  
        if (P >= 0) { P += P2; ++y; }  
        else        { P += P1; }  
    }  
}
```

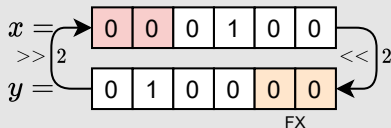
## Vlastnosti

- Optimalizace DDA pro Fixed Point (FX)
- Efektivní implementace, jednoduchý

```
LineDDAFX(int x1, int y1, int x2, int y2) {  
    int y = y1 << FX_BITS;  
    const int k = ((y2-y1) << FX_BITS)/(x2-x1);  
    for(int x = x1; x <= x2; ++x) {  
        setPixel(x, y >> FX_BITS); y += k;  
    }  
}
```

## Vlastnosti

- Optimalizace DDA pro Fixed Point (FX)
- Efektivní implementace, jednoduchý
- Meta-parametr **FX\_BITS = 8**



$$1.75_{10} \cdot 10^2 = 175.0 \Rightarrow 175 \text{ div } 2 = 87 \Rightarrow 87.0 \cdot 10^{-2} = 0.87 \approx \frac{1.75}{2.0} = 0.875$$

$$1.11_2 \cdot 2^2 = 111.0 \Rightarrow 111 \text{ div } 10 = 11 \Rightarrow 11.0 \cdot 2^{-2} = 0.11 \approx \frac{1.11}{10.0} = 0.111$$

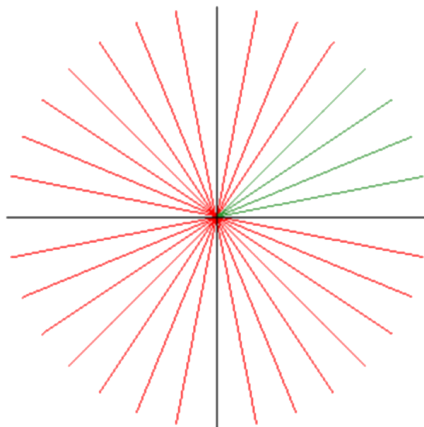
```

LineDDAFX(int x1, int y1, int x2, int y2) {
    int y = y1 << FX_BITS;
    const int k = ((y2-y1) << FX_BITS)/(x2-x1);
    for(int x = x1; x <= x2; ++x) {
        setPixel(x, y >> FX_BITS); y += k;
    }
}

```

## Omezený rozsah

- Jen pro první polovinu I. kvadrantu
- Obecné řešení  $\Rightarrow$  bodovaná úloha (**2b**)



## #0 $\frac{1}{2}$ l. kvadrantu

- Předem připravené

## #1 $[x1, y1] = [x2, y2]$

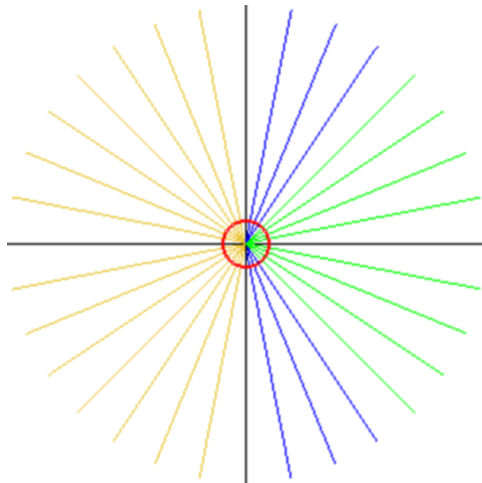
- Neplatná úsečka
- Před opravou pád aplikace!

## #2 $|dy| > |dx|$

- Záměna  $X \Leftrightarrow Y$  souřadnic
- Inverzní operace při zápisu do framebufferu!

## #3 $x_1 > x_2$

- Výměna bodů  $[x1, y1]$  a  $[x2, y2]$

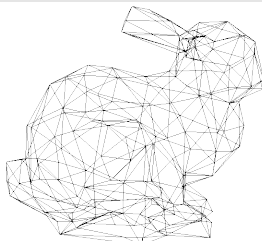
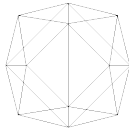
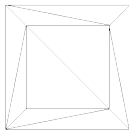


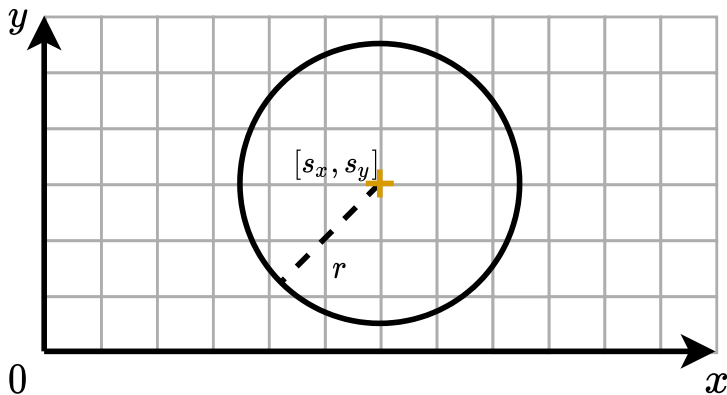
## Doplnit a upravit funkci `drawLine()`

- V souboru **student.cpp**
- Vykreslení testovacího vzoru v celém kruhu

## Tips & Tricks

- Pro testování kreslení levým tlačítkem myši + klávesa **T**
- Postupujte od bodu **#0** po **#3**
- Po dokončení vyzkoušejte klávesy **D**, **B**, **N**, **M** a pohyb s prostředním tlačítkem myši





## Středová rovnice **kružnice**

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



## Cíle

- Vektorový  $\Rightarrow$  Rastrový
- Efektivní & Jednoduchý
- Využití **8-symetrie**

## Algoritmy

- 1 Vykreslení kružnice po bodech
- 2 Vykreslení kružnice jako N-úhelník
- 3 MidPoint algoritmus

```
void circleByPoints(int R) {  
    const int R2 = R * R;  
    int x = R;  
    int y = 0;  
    while (x >= y)  
    {  
        put8CirclePixels(x, y);  
        x--;  
        y = sqrt(R2 - x * x);  
    }  
}
```

```
void circleByPolygon(int R, int N) {  
    const double cosa = cos(-PI/4 * 1/N);  
    const double sina = sin(-PI/4 * 1/N);  
    int x1 = R, y1 = 0, x2, y2;  
    for (int i = 0; i < N; ++i) {  
        x2 = x1 * cosa - y1 * sina;  
        y2 = x1 * sina + y1 * cosa;  
        put8CircleLines(x1, y1, x2, y2);  
        x1 = x2; y1 = y2;  
    }  
}
```

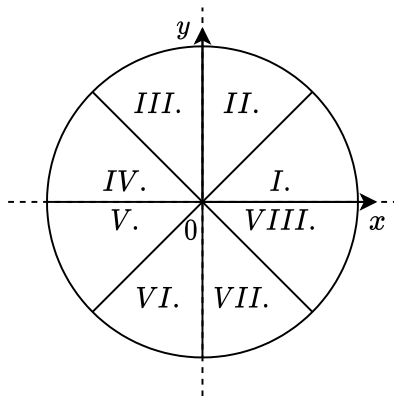
## Vlastnosti

- Celočíselná aritmetika
- Efektivní implementace, jednoduchý
- Základní varianta pro  $s = [0, 0]$  a polovinu I. kvadrantu

```
void circleMidPoint(int R) {
    int x = R, y = 0;
    int P = 1 - R, X2 = 2 - 2 * R, Y2 = 3;
    while (x >= y) {
        put8CirclePixels(x, y);
        if (P >= 0)
            { P += X2; X2 += 2; --x; }
        P += Y2; Y2 += 2; ++y;
    }
}
```

## Omezený rozsah

- Střed v počátku  $s = [0, 0]$
- Jen pro první polovinu I. kvadrantu
- Obecné řešení  $\Rightarrow$  bodovaná úloha (**1b**)



## #1 $[0, 0] \Rightarrow [s_x, s_y]$

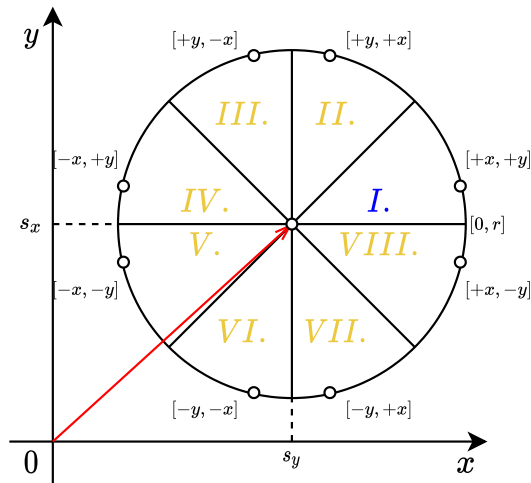
- Posun středu souřadné soustavy

## #2 I. Oktant

- Vykreslení jednoho bodu

## #3 II. – VIII. Oktant

- Duplikace bodu pro zbytek
- Prohození souřadnic

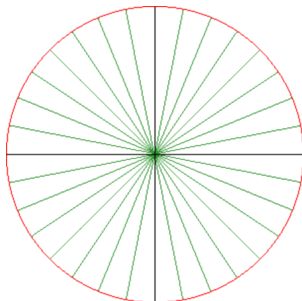


Doplnit a upravit funkci `put8PixelsOfCircle()`

- V souboru **student.cpp**
- Vykreslení celého kruhu v testovacím vzoru

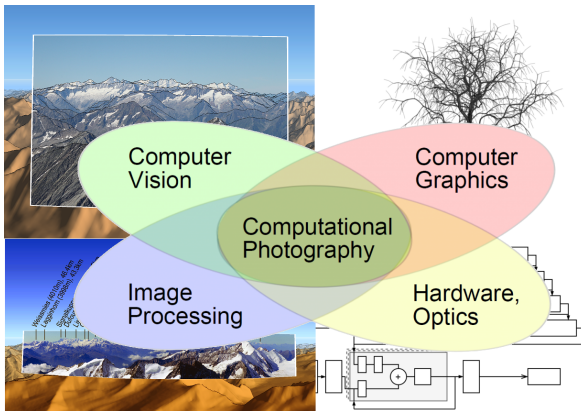
## Tips & Tricks

- Pro testování kreslení pravým tlačítkem myši + klávesa **T**
- Postupujte od bodu **#1** po **#3**



# Generování Základních Objektů v Rastru

## 2. cvičení z předmětu IZG



[cphoto.fit.vutbr.cz](http://cphoto.fit.vutbr.cz)

## Introduction to Game Development



[cphoto.fit.vutbr.cz/ludo](http://cphoto.fit.vutbr.cz/ludo)

Doplnit a upravit funkci `drawLine()`

- Vykreslení testovacího vzoru v celém kruhu

Doplnit a upravit funkci `put8PixelsOfCircle()`

- Vykreslení celého kruhu v testovacím vzoru

## Hodnocení úkolu

- Ukázka a zodpovězení dotazu
- **student.cpp**  $\Rightarrow$  **<LOGIN>.cpp**  $\Rightarrow$  `ipolasek@fit.vutbr.cz`  
Subject = "[IZG] <LOGIN> : Cviceni #2"

