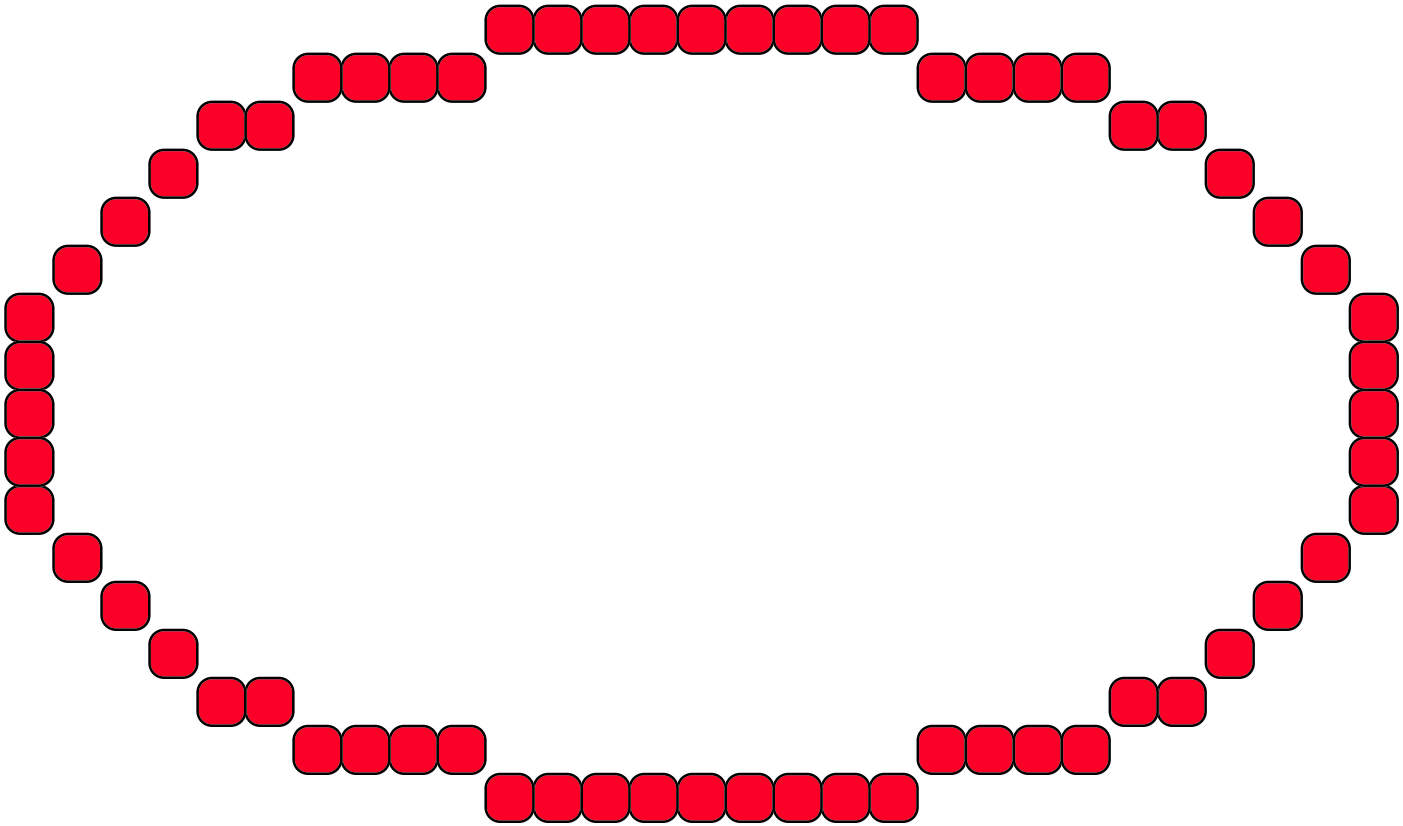
Kreslení křivek

KSVI MFF UK Praha

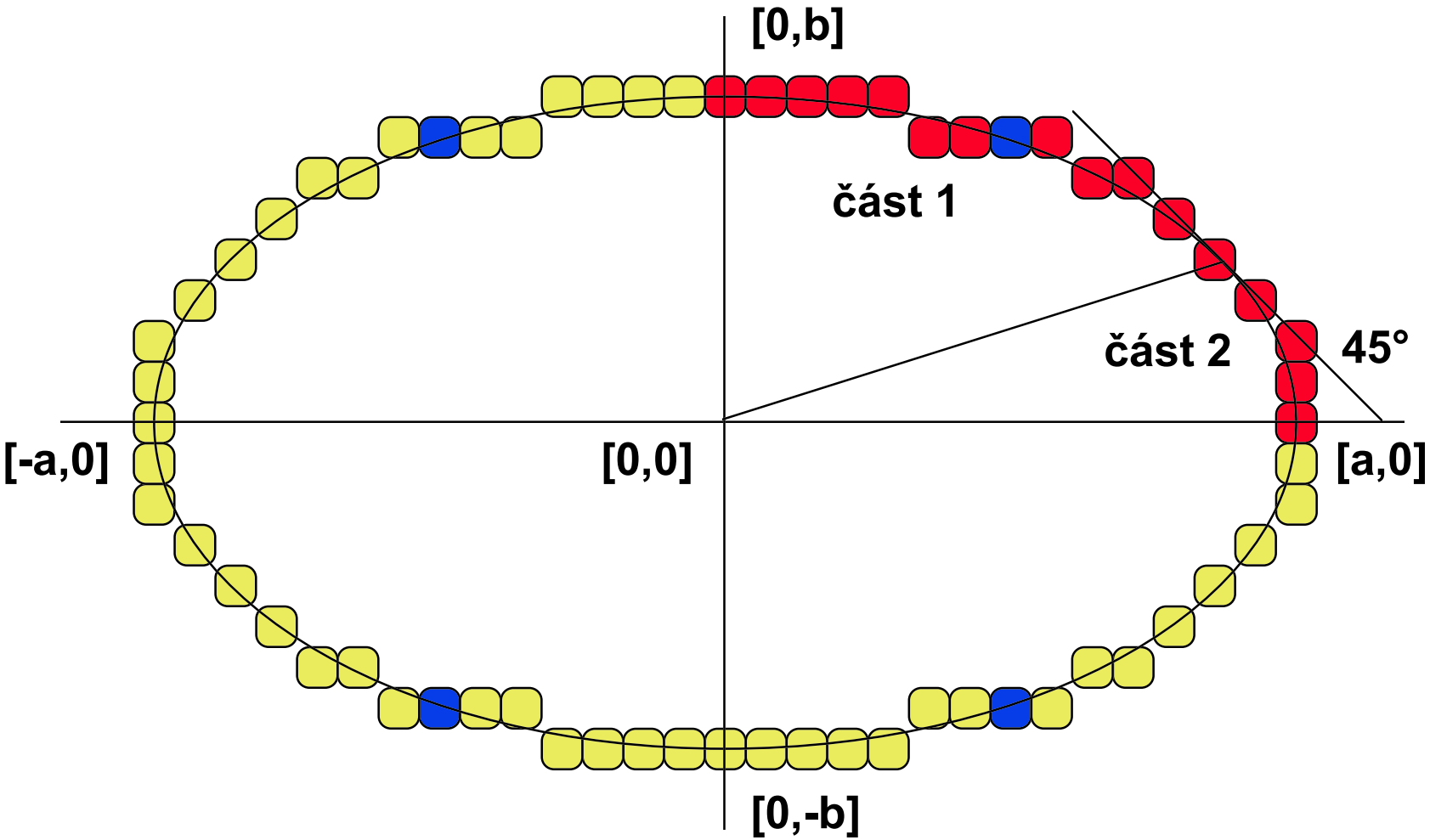
e-mail: Josef.Pelikan@mff.cuni.cz

WWW: http://cgg.ms.mff.cuni.cz/~pepca/

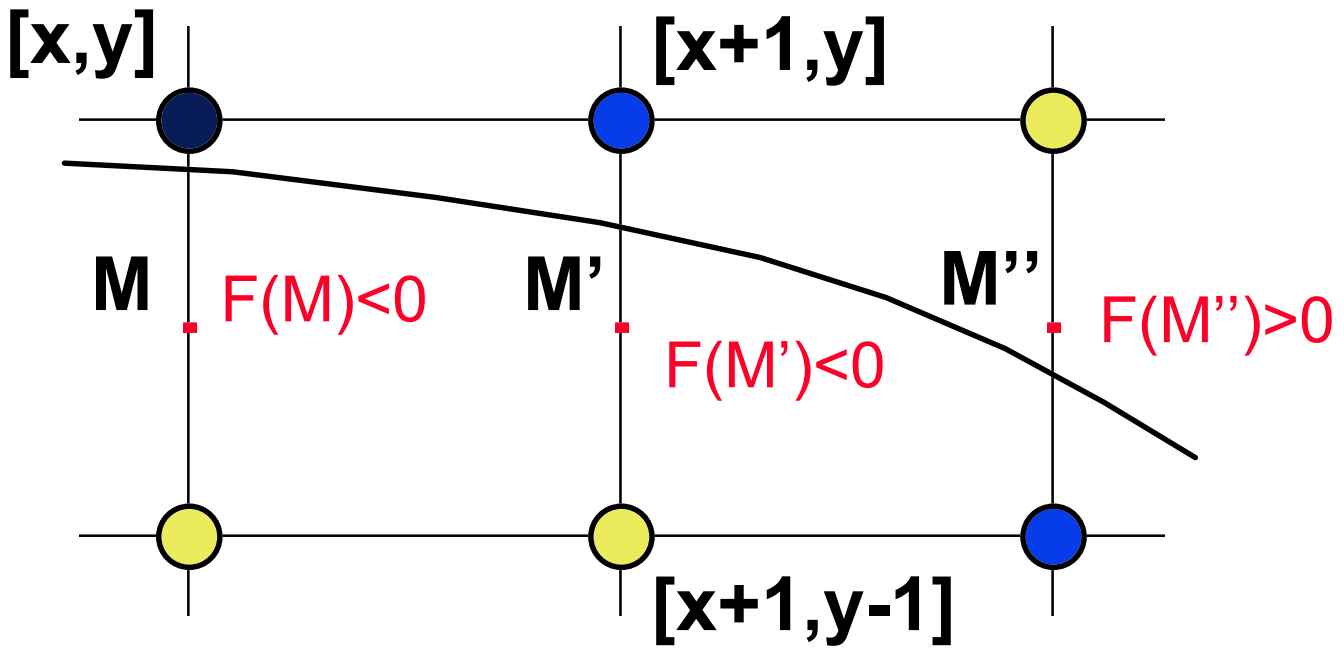
Elipsa



Elipsa se středem v počátku



“midpoint” algoritmus



$$F(M) = b^2 M_x^2 + a^2 M_y^2 - a^2 b^2$$

Inkrementální odvození

1) $F(M') = b^2(x+1)^2 + a^2(y-\frac{1}{2})^2 - a^2b^2 \quad < \quad 0$

$F(M'') = b^2(x+2)^2 + a^2(y-\frac{1}{2})^2 - a^2b^2$

$F(M'') = F(M') + b^2(2x+3)$

2) $F(M') \geq 0$

$F(M'') = b^2(x+2)^2 + a^2(y-\frac{3}{2})^2 - a^2b^2$

$F(M'') = F(M') + b^2(2x+3) + a^2(-2y+2)$

Inkrementální odvození

I) inicializace:

$$F\left[1, b - \tfrac{1}{2}\right] = b^2 - a^2 b + \tfrac{1}{4} a^2$$

II) přechod z části 1 do části 2:

$$\frac{\partial F}{\partial x} = \frac{\partial F}{\partial y}, \quad \text{tj.} \quad 2 b^2 x = 2 a^2 y$$

pomocné diferenční proměnné pro [x+1,y-1/2]:

$$dx = \frac{\partial F}{\partial x} = b^2 (2x + 2), \quad dy = \frac{\partial F}{\partial y} = a^2 (2y - 1)$$

Inkrementální odvození

III) inicializace pro část 2:

$$F\left[x+\tfrac{1}{2}, y-1\right] = b^2\left(x+\tfrac{1}{2}\right)^2 - a^2(y-1)^2 - a^2b^2$$

testy v části 2:

$$F(M') \geq 0 \Rightarrow F(M'') = F(M') + a^2(-2y+3)$$

$$F(M') < 0 \Rightarrow F(M'') = F(M') + b^2(2x+2) + a^2(-2y+3)$$

Kreslení elipsy

```
procedure EllipsePoints ( x, y, color : integer );
                { předpoklad: střed elipsy je v počátku }
begin
  PutPixel( x, y,color);
  PutPixel( x,-y,color);
  PutPixel(-x, y,color);
  PutPixel(-x,-y,color);
end;

procedure EllipseMidpoint ( a, b, color : integer );
var x, y, D, dx, dy, aa, aa2, bb, bb2 : longint;
begin
  x := 0; y := b;          { souřadnice prvního bodu }
  aa := sqr(a); aa2 := 2*aa; bb := sqr(b); bb2 := 2*bb;
  D := bb - aa*b + aa div 4;
  dx := bb2; dy := aa*(2*b - 1);     { dF/dx, dF/dy }
  ...
```

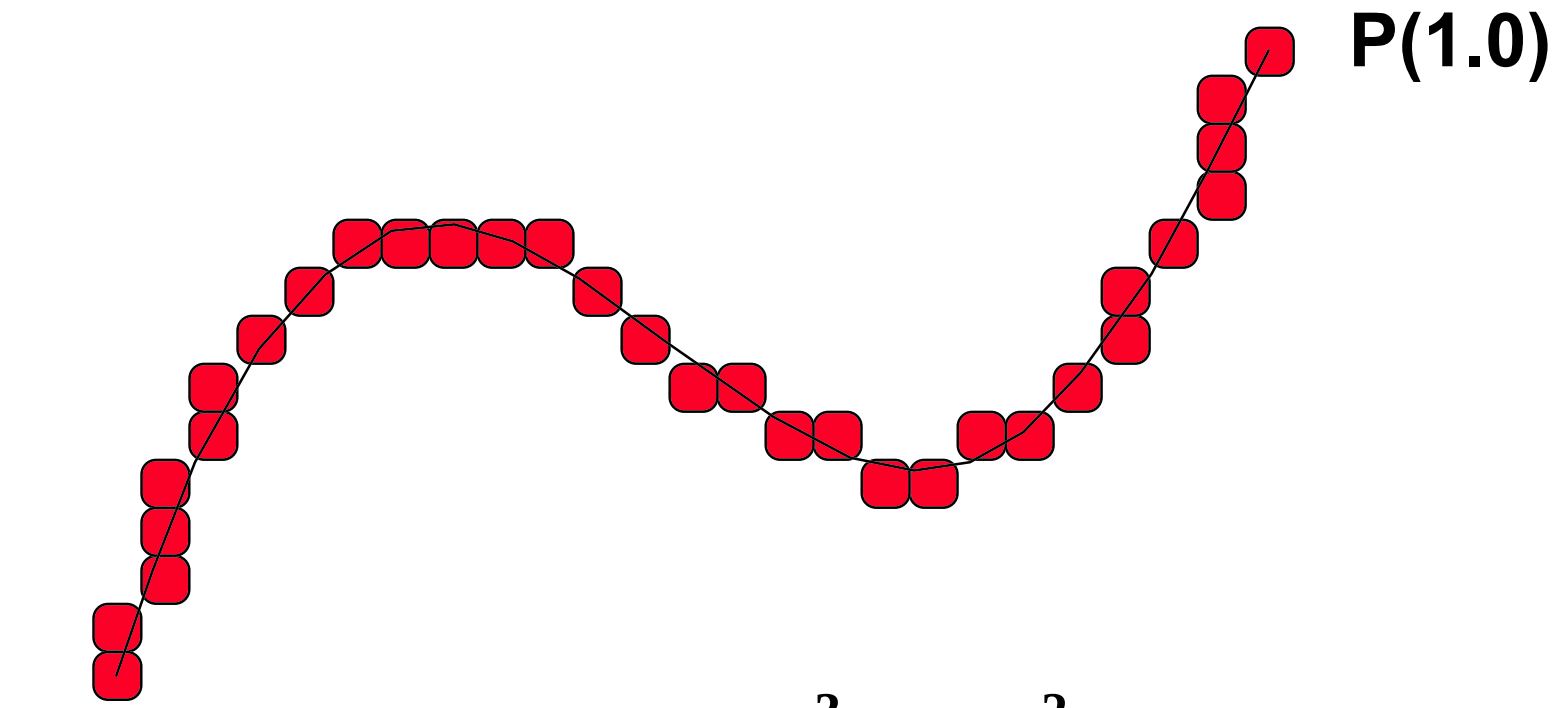
Kreslení elipsy

```
...
EllipsePoints(0,b,color);
while dx < dy do        { část 1 }
  begin
    if D >= 0 then
      begin             { klesám o jednu řádku }
        D := D - dy + aa;
        dy := dy - aa2;
        y := y - 1;
      end;
    D := D + dx + bb;
    dx := dx + bb2;
    x := x + 1;
    EllipsePoints(x,y,color);
  end;
...
```

Kreslení elipsy

```
  ...
  D := bb*(sqr(x)+x) + bb div 4 + aa*(sqr(y-1)-bb);
  while y < 0 do          { část 2 }
    begin
      if D < 0 then
        begin             { posun doprava }
          D := D + dx;
          dx := dx + bb2;
          x := x + 1;
        end;
      D := D - dy + aa2;
      dy := dy - aa2;
      y := y - 1;
      EllipsePoints(x,y,color);
    end;
end;
```

Kubická křivka v rovině



$$P_x(t) = A_x t^3 + B_x t^2 + C_x t + D_x$$

$$P_y(t) = A_y t^3 + B_y t^2 + C_y t + D_y$$

Diferenční algoritmus

- výpočet hodnot **polynomu P** v bodech **P(0)**, **P(h)**, **P(2h)**, .. **P(ih)**, .. pouze pomocí **sčítání**

- pro **kubický polynom**:
  1. inicializace proměnných **a**, **b**, **c**, **d**
  2. krok: **Output(d)**
     d := d + c
     c := c + b
     b := b + a

Maticová notace

- **inicializace**:

$$\begin{bmatrix} d \\ c \\ b \\ a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & h & h^2 & h^3 \\ 0 & 0 & 2h^2 & 6h^3 \\ 0 & 0 & 0 & 6h^3 \end{bmatrix} \cdot \begin{bmatrix} D \\ C \\ B \\ A \end{bmatrix}$$

- **krok**:

$$\begin{bmatrix} d \\ c \\ b \\ a \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d \\ c \\ b \\ a \end{bmatrix}$$

Úpravy kroku **h**:

- **U** - zvětšení kroku na dvojnásobek (“Up”):

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

- **D** - zmenšení kroku na polovinu (“Down”):

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & -1/8 & 1/16 \\ 0 & 0 & 1/4 & -1/8 \\ 0 & 0 & 0 & 1/8 \end{bmatrix}$$

Adaptivní diferenční algoritmus

```
procedure CubicCurve;
var B1, B2, B3 : POINT2D; { x, y : real }
    X, Y : DiffRecord;    { a, b, c, d, h : real }
    t : real;             { parametr křivky }
begin
  "inicializace X, Y"   { inicializace diferencí }
  t := 0.0; B1.x := X.d; B1.y := Y.d;
  B2 := B1; B3 := B1;   { B1=B2=B3=začátek křivky }
  repeat                { jeden krok křivky }
    if |B3-B1| > 1 then Output(B1); B1 := B2
    B2 := B3;           { zapomenu B2 }
    while (X.c > 1) or (Y.c > 1) do Down(X); Down(Y);
    while (X.c < 0.5) and (Y.c < 0.5) do Up(X); Up(Y);
    Step(X); Step(Y); B3.x := X.d; B3.y := Y.d;
    t := t + X.h;
  until t > 1.0;
  Output(B1); Output(B2);  { dokreslím zbytek křivky }
end;
```

Celočíselná implementace

- pro uložení proměnných **a**, **b**, **c**, **d** použiji **32-bitové** registry (v pevné desetinné čárce)

a	celá část	desetinná část
b	12 bitů	20 bitů
c		
d		

- díky **akumulaci chyb** nelze nakreslit křivku delší než cca **100 pixelů**

Zvětšená přesnost členu **d**:

- pro uložení proměnných **a**, **b**, **c** použiji **více desetinných bitů** (28)

- přibyla jedna **operace "shift"**, ale již lze nakreslit až **512 kroků křivky**

Dynamické řízení přesnosti:

- pro uložení proměnných **a**, **b**, **c** použiji **více desetinných bitů** (**n** mohu adaptivně měnit)

- přibyly dvě **operace "shift"**, ale např. pro **n=14** lze již nakreslit **8K kroků křivky**

Literatura

- **J. Foley, A. van Dam, S. Feiner, J. Hughes**: ***Computer Graphics, Principles and Practice***, 88-91
- **D. Da Silva: *Raster Algorithms for 2D Primitives***, Master's Thesis, Brown University, 1989
- **Jiří Žára a kol.**: ***Počítačová grafika***, principy a algoritmy, 112-115

Konec

- **S.-L. Chang, M. Chantz, R. Rocchetti:** ***Rendering Cubic Curves and Surfaces with Integer Adaptive Forward Differencing***, Computer Graphics, vol. 23, #3, 157-166

- **LAN** na Malé Straně:
  - **barbora\usr:\vyuka\pelikan\3**