

Seminář 8

Připomenutí - sousednost

- 4-sousednost, 8-sousednost
- 4-cesta, 8-cesta
- komponenty

Přímky (úsečky)

Rasterizace přímky - DDA algoritmus

DDA algoritmus

1. Z koncových bodů $[x_1, y_1]$ a $[x_2, y_2]$ urči směrnici m .
2. Inicializuj bod $[x, y]$ hodnotou $[x_1, y_1]$.
3. Dokud je $x \leq x_2$ opakuj:

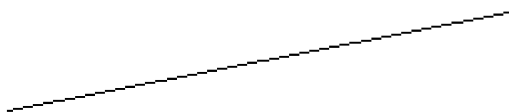
- Vykresli bod $[x, \text{zaokrouhlene}(y)]$

- $x = x + 1$

- $y = y + m$

Tento algoritmus je jen pro $0 \leq m \leq 1$ pro ostatní m je potřeba upravit. pro $|m| > 1$ se úsečka přimyká k y a tak se y zvetsuje o 1 a x o $1/m$

```
usecka = DDA([-100,1], [150,50],[60,300]);  
figure, imshow(usecka);
```



Úkol 1

Jak bychom upravili funkci `DDA()`, aby se místo 8-sousedné přímky vykreslovala 4-sousedná.

Úkol 2

Simulujte (na papír) rasterizaci úsečky $A = [0, 0]$, $B = [10, 5]$ pomocí algoritmu DDA.

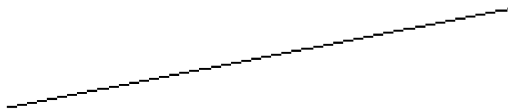
Úkol 3

Simulujte (na papír) rasterizaci úsečky $A = [0, 0]$, $B = [5, 10]$ pomocí algoritmu DDA, přičemž vezměte jako řídící osu osu x .

Bresenhamův algoritmus

`bresenhamuv_algoritmus()` -- pro názornost jen pro přímky přichylující se k ose x .

```
usecka = bresenhamuv_algoritmus([-100,1], [150,50],[60,300]);  
figure, imshow(usecka);
```



Bresenhamův algoritmus - přerušovaná čára

`ba_prerusovana()` -- pro názornost jen pro přímky přichylující se k ose x .

úseky jsou vždy stejné délky ve smyslu počtu pixelů

```
usecka = ba_prerusovana([-100,1], [100,1],[300,300], 10); % posledni argument =  
delka useku  
usecka2 = ba_prerusovana([-100,-100], [100,99],[300,300], 10); % posledni  
argument = delka useku  
  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```



Bresenhamův algoritmus - přerušovaná čára

`ba_prerusovana2()` -- pro názornost jen pro přímky přichylující se k ose x.

úseky jsou vždy stejné délky

```
usecka = ba_prerusovana2([-100,1], [100,1],[300,300], 10); % posledni argument =  
delka useku  
usecka2 = ba_prerusovana2([-100,-100], [100,99],[300,300], 10); % posledni  
argument = delka useku  
  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```



Bresenhamův algoritmus - silná čára

`ba_silna()` -- pro názornost jen pro přímky přichylující se k ose x.

tloušťka čáry je dána počtem pixelů

```
usecka = ba_silna([-100,1], [100,1],[300,300], 10); % posledni argument = tloustka
usecka2 = ba_silna([-100,-100], [100,99],[300,300], 10); % posledni argument =
tloustka

figure,
subplot(1,2,1), imshow(usecka);
subplot(1,2,2), imshow(usecka2);
```



Bresenhamův algoritmus - silná čára

`ba_silna2()` -- pro názornost jen pro přímky přichylující se k ose x.

tloušťka čáry je dána šířkou

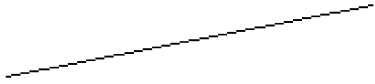
```
usecka = ba_silna2([-100,1], [100,1],[300,300], 10); % posledni argument =  
tloustka  
usecka2 = ba_silna2([-100,-100], [100,99],[300,300], 10); % posledni argument =  
tloustka  
  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```



DDA antialias

vyhlazení úsečky

```
usecka = DDA([-100,1], [100,40],[60,205]);  
usecka2 = DDAalias([-100,1], [100,40],[60,205]);  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```

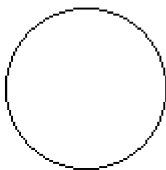


Kružnice

Bresenhamův algoritmus

ba_kruznice()

```
kruznice = ba_kruznice(40, [50,50], [100,100]);  
figure, imshow(kruznice);
```



Pomocné funkce

DDA algoritmus

```
function out = DDA(A,B,velikost)
```

```

out = ones(velikost);
% nastaveni offsetu pro primky se souradnicema <=0

if (A(1) <=0 || B(1) <= 0)
    offset_x = - min(A(1),B(1)) + 1;
else
    offset_x = 0;
end

if (A(2) <=0 || B(2) <= 0)
    offset_y = - min(A(2),B(2)) + 1;
else
    offset_y = 0;
end

% z koncovych bodu se urci smernice
dx = B(1)-A(1);
dy = B(2)-A(2);

m = dy/dx;

%inicializace [x,y]
x = A(1);
y = A(2);

while(x <= B(1))
    out(round(y)+offset_y, x+offset_x) = 0;
    x = x +1;
    y = y + m;
end

out = flipud(out); % otoceni obrazku
end

```

Bresengamův algoritmus

```

function out = bresenhamuv_algoritmus(A,B,velikost)
    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

```



```

end

% z koncovych bodu se urci konstanty
dx = B(1)-A(1);
dy = B(2)-A(2);

k1 = 2*dy;
k2 = 2*(dy-dx);

% rozhodovaci clen
p = 2*(dy-dx);

%inicializace [x,y]
x = A(1);
y = A(2);

while(x <= B(1))
    x = x +1;
    if (p >0)    % p je kladne
        y = y +1;
        p = p + k2;
    else        % p neni kladne
        p = p + k1;
    end
    out(y+offset_y, x+offset_x) = 0; %vykresleni bodu    (v obrazcich je prvni
souradnice sloupec a druhy radek)
end

    out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - přerušovaná čára (naivní)

```

function out = ba_prerusovana(A,B,velikost, delka)
    aktualni_delka = delka;
    plny = 1;

    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0

    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else

```

```

        offset_y = 0;
end

% z koncovych bodu se urci konstanty
dx = B(1)-A(1);
dy = B(2)-A(2);

k1 = 2*dy;
k2 = 2*(dy-dx);

% rozhodovaci clen
p = 2*(dy-dx);

%inicializace [x,y]
x = A(1);
y = A(2);

while(x <= B(1))
    x = x +1;
    if (p >0)    % p je kladne
        y = y +1;
        p = p + k2;
    else        % p neni kladne
        p = p + k1;
    end

    %vykresleni jen, pokud je v plnem useku
    if(plny == 1)
        out(y+offset_y, x+offset_x) = 0; %vykresleni bodu    (v obrazcich je
prvni souradnice sloupec a druhy radek)
    end

    aktualni_delka = aktualni_delka -1;
    if(aktualni_delka ==0)
        if(plny == 1)
            plny = 0;
        else
            plny = 1;
        end
        aktualni_delka = delka;
    end

end

out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - přerušovaná čára

```

function out = ba_prerusovana2(A,B,velikost, delka)
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    delka = round((abs(dx) / sqrt(dx^2 + dy^2) )*delka);
    aktualni_delka = delka;
    plny = 1;

    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci konstanty
    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovaci clen
    p = 2*(dy-dx);

    %inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        x = x +1;
        if (p >0) % p je kladne
            y = y +1;
            p = p + k2;
        else % p neni kladne
            p = p + k1;
        end

        %vykresleni jen, pokud je v plnem useku
        if(plny == 1)
            out(y+offset_y, x+offset_x) = 0; %vykresleni bodu (v obrazcich je
            prvni souradnice sloupec a druhy radek)
        end
    end

```

```

    aktualni_delka = aktualni_delka -1;
    if(aktualni_delka ==0)
        if(plny == 1)
            plny = 0;
        else
            plny = 1;
        end
        aktualni_delka = delka;
    end
end
out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - silná čára (naivní)

```

function out = ba_silna(A,B,velikost, tloustka)
    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci konstanty
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovaci clen
    p = 2*(dy-dx);

    %inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        x = x +1;
        if (p >0) % p je kladne
            y = y +1;

```

```

        p = p + k2;
    else % p není kladné
        p = p + k1;
    end
    out(y+offset_y: y+offset_y + tloušťka-1, x+offset_x) = 0; % vykreslení
    tloušťka bodů

end
out = flipud(out); % otočení obrázku
end

```

Bresenhamův algoritmus - silná čára

```

function out = ba_silna2(A,B,velikost, tloušťka)
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    tloušťka = round((sqrt(dx^2 + dy^2) / abs(dx))*tloušťka);

    out = ones(velikost);

    % nastavení offsetu pro přímky se souřadnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncových bodů se určí konstanty
    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovací člen
    p = 2*(dy-dx);

    % inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        x = x +1;
        if (p >0) % p je kladné
            y = y +1;

```

```

        p = p + k2;
    else          % p není kladné
        p = p + k1;
    end

    out(y+offset_y: y+offset_y + tloušťka-1, x+offset_x) = 0; %vykreslení
    tloušťka bodu

end

out = flipud(out); % otočení obrázku
end

```

DDA antialias

```

function out = DDAalias(A,B,velikost)
    out = ones(velikost);

    % nastavení offsetu pro primky se souřadnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncových bodů se určí směrnice
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    m = dy/dx;

    %inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        out(floor(y)+offset_y, x+offset_x) = y - floor(y);
        out(ceil(y)+offset_y, x+offset_x) = ceil(y) - y;
        x = x +1;
        y = y + m;
    end

    out = flipud(out); % otočení obrázku
end

```

Bresenhamův algoritmus - kružnice

```
function out = ba_kruznice(r, stred, velikost)
    out = ones(velikost);

    % offset, aby byla kruznice uprostred. Pripadne by zde
    offset_x = stred(1);
    offset_y = stred(2);

    devx = 3;
    devy = 2*r - 2;

    % rozhodovaci clen
    p = 1 - r;

    x = 0;
    y = r;

    while(x <= y)
        out(x+offset_x, y+offset_y) = 0;
        out(-x+offset_x, y+offset_y) = 0;
        out(x+offset_x, -y+offset_y) = 0;
        out(-x+offset_x, -y+offset_y) = 0;
        out(y+offset_y, x+offset_x) = 0;
        out(-y+offset_y, x+offset_x) = 0;
        out(y+offset_y, -x+offset_x) = 0;
        out(-y+offset_y, -x+offset_x) = 0;
        if (p >= 0)
            p = p - devy;
            devy = devy - 2;
            y = y - 1;
        end
        p = p + devx;
        devx = devx + 2;
        x = x+1;
    end
end
```