KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Transformace barev, komprese, rasterizace

Vybrané partie z IT

Mgr. Markéta Trnečková, Ph.D.

Transformace barev

- polotónování (halftoning)
- rozptylování (dithering)
  - náhodné rozptýlení
  - pravidelné (maticové) rozptýlení
  - distribuce zaokrouhlovací chyby

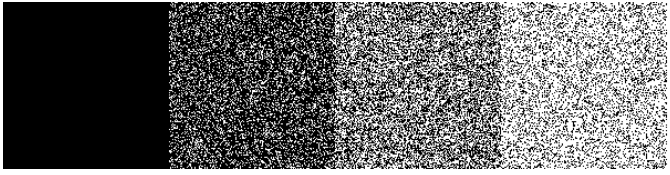
Transformace barev

- **Vstup**: f ($[0, f_{max}]$)
- **Výstup**: g ($\{0, 1\}$)

Transformace barev

Náhodné rozptýlení

1. $g(x, y) = 0$
2. Pokud $f(x, y) > random(f_{max})$
   $g(x, y) = g(x, y) + 1$

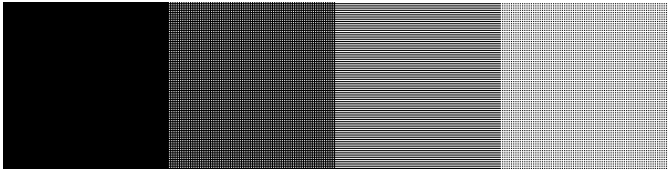


Transformace barev

Maticové rozptýlení

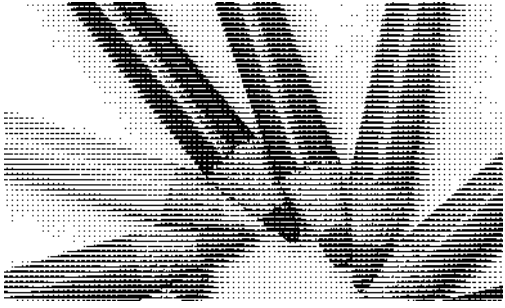
$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 3 & 2 \\ 1 & 0 \end{bmatrix}$$

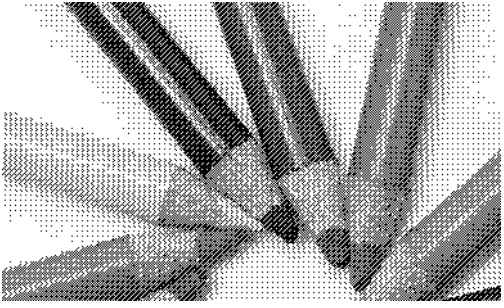
$f = 0$ $f = 1$ $f = 2$ $f = 3$ $f = 4$ zkráceně

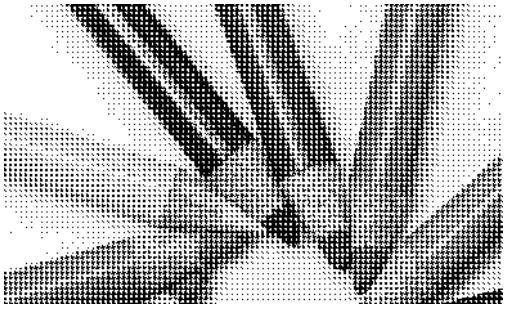


Transformace barev

Maticové rozptýlení







Transformace barev

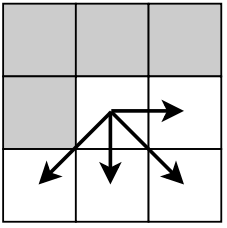
Maticové rozptýlení

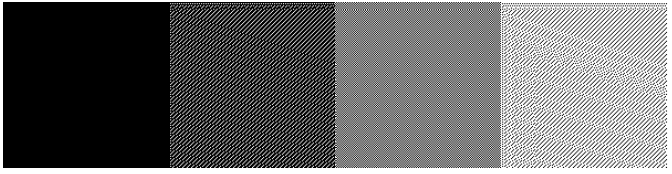
1. $g(x, y) = 0$
2. Pokud $f(x, y) > M(x \mod n, y \mod n)$
   $g(x, y) = g(x, y) + 1$

Transformace barev

Distribuce zaokrouhlovací chyby

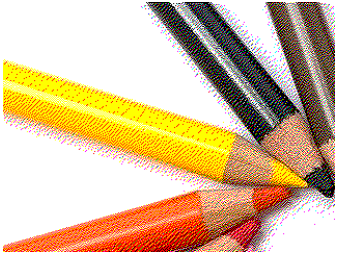
- **Chyba**: $e = f(x, y) - g(x, y)$
- **Floyd-Steinberg**:
  7/16, 3/16, 5/16 a 1/16

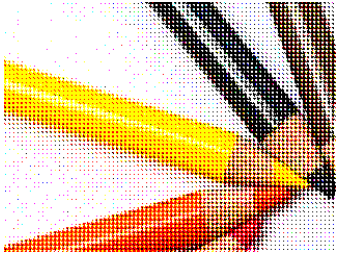




Transformace barev

Barevné rozptylování



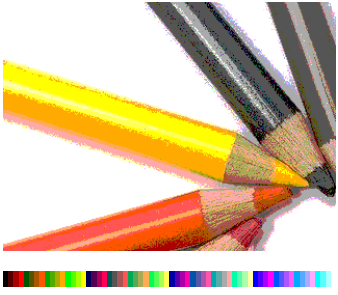


Transformace barev

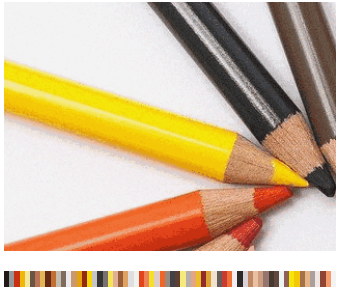
Barevná paleta

- univerzální
- přizpůsobená obrazu

Původní obraz



Univerzální paleta

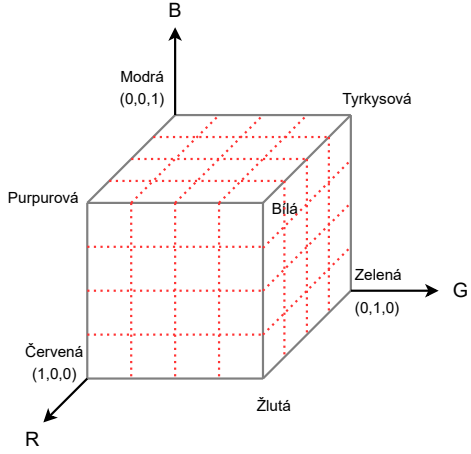


Přizpůsobená paleta

Transformace barev

Přednastavené barevné palety

- Vybrané barvy
- Pravidelně rozdělený barevný prostor



Transformace barev

3-3-2 paleta

- **Barev**: 256
- **Výpočet**:

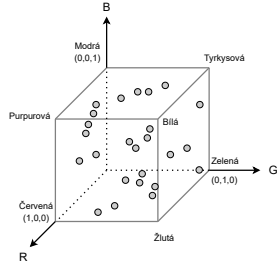
$$r = ((i \gg 5) \cdot 255)/7$$
$$g = (((i \gg 2)\&7) \cdot 255)/7$$
$$b = ((i\&3) \cdot 255)/3$$

- **Index**: $i = (((r \cdot 7)/255) \ll 5) + (((g \cdot 7)/255) \ll 2) + ((b \cdot 3)/255)$

Transformace barev

Paleta přizpůsobená obrazu

- **Tečky**: četnost barvy
- Dělíme prostor řezy rovnoběžnými s osami:
  - Začátek: 1 oblast
  - dokud nemáme předepsaný počet oblastí:
    - najdeme oblast s největším rozměrem v jedné z os
    - rozdělíme tuto oblast řezem kolmým na vybranou souřadnicovou osu
  - každou oblast nahradíme jednou barvou

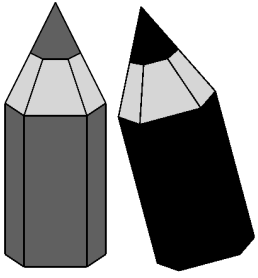


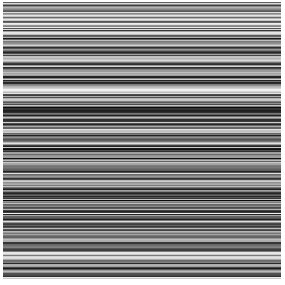
Typy obrazů

- *binární (B/W) – 1 bit/pixel*
- v odstínech šedi (gray scale) – 1 byte/pixel
- *indexový (pseudo color) – 1 byte/pixel*
- indexový (direct color) – 3 byte/pixel
- *plně barevný (color) – 3-4 složky"*
  - *low color (*15 *bit)*
  - *high color (*16 *bit)*
  - *true color (*24 *bit)*
  - *super true color (*32 *bit)*
  - *deep color (*48 *bit)*

Redundance

- *Redundance kódování – informaci kódujeme více bity, než je potřeba*
- Redundance prostorová – korelace mezi pixely
- *Nerelevantní informace – informace, kterou lidské oko nedokáže zpracovat*





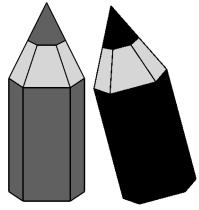
Redundance

- **Redundantní data** – reprezentace obsahuje opakující se nebo nerelevantní informaci
- **Kód nesoucí informaci** – b, b'
- *Relativní redundance dat*
  $R = 1 - \frac{1}{C}$
- Kompresní poměr
  $C = \frac{b}{b'}$

Redundance

Redundance kódování

- **Délka** – počet bitů každé informace
- 8-**bitový kód** – každá barva je kódována 8 bity
- Pro obrázek obsahující 4 barvy je 8-bitový kód zbytečný
- **Fixní** *m*-**bitový kód** – každý kus informace kódován *m* bity



Redundance

Redundance kódování

- Fixní m-bitový kód není vždy optimální
- **Variabilní délka kódu** – Huffmanovo kódování
- $r_k \in [0, L-1]$ – intenzity v obraze
- velikost obrazu: $M \times N$
- n_k – počet výskytů intenzity r_k
- $P(r_k) = \frac{n_k}{M \cdot N}$ – pravděpodobnost výskytu intenzity
- $L(r_k)$ – počet bitů potřebných k reprezentaci hodnoty r_k
- **Průměrný počet bitů potřebných k reprezentaci každého pixelu** – $L_{avg} = \sum_{k=0}^{L-1} L(r_k)P(r_k)$
- **Celkový počet bitů potřebných k reprezentaci každého pixelu** – $M \cdot N \cdot L_{avg}$

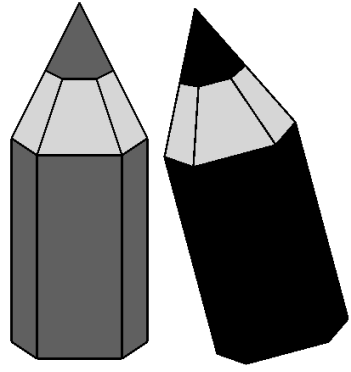
Redundance

Redundance kódování – Příklad

- Velikost: 440×440
- Intenzity: 0, 96, 214 a 255

Fixní 8-kód

intenzita	kód
r_0	00000000
r_{96}	01100000
r_{214}	11010110
r_{255}	11111111



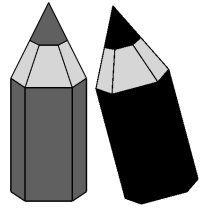
Redundance

Redundance kódování – Příklad

Kód s proměnlivou délkou

intenzita	$P(r_k)$	kód	délka kódu
r_0	0.25	01	2
r_{96}	0.2	000	3
r_{214}	0.1	001	3
r_{255}	0.45	1	1

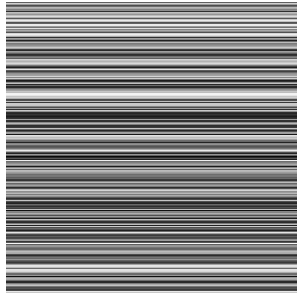
- Průměrná délka:
  $L_{avg} = 0.25 \cdot 2 + 0.2 \cdot 3 + 0.1 \cdot 3 + 0.45 \cdot 1 = 1.85$
- Komprese a relativní redundance 8-kódu:
  $C = \frac{440 \cdot 440 \cdot 8}{440 \cdot 440 \cdot 1.85} = \frac{8}{1.85} \approx 4.32$
  $R = 1 - \frac{1}{4.32} \approx 0.77$



Redundance

Redundance prostorová

- Velikost: 256×256
- Intenzity: $0, \ldots, 255$
- Kód: Každý řádek intenzita + počet opakování
- Každý pixel kódován 2 byty
- RLE komprese



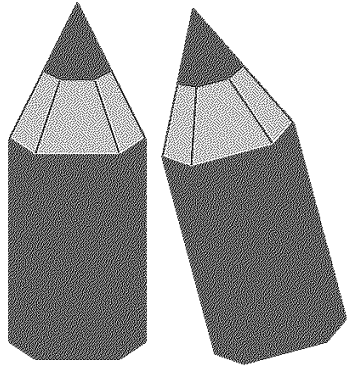
Redundance

Nerelevantní informace

- Velikost: 440×440
- Fixní 8-kód – celková délka: $440 \cdot 440 \cdot 8$
- Zaokrouhlení: 1 intenzita

Redundance

Nerelevantní informace



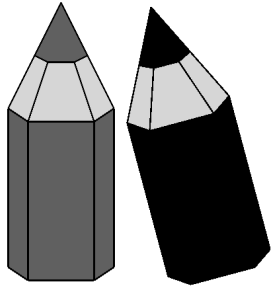
Redundance
Informace

- **Náhodná událost** – E
- **Pravděpodobnost náhodné události** – E
- **Informace** – $I(E) = \log \frac{1}{P(E)} = -\log P(E)$
- **Základ logaritmu** = jednotky (v obraze 2)
- **Entropie** = průměrná informace
  $H = -\sum_{j=1}^{J} P(a_j) \log P(a_j)$
- **Matlab**: `J = entropy(I)`

Redundance

Redundance kódování – Příklad

intenzita	$P(r_k)$
r_0	0.25
r_{96}	0.2
r_{214}	0.1
r_{255}	0.45



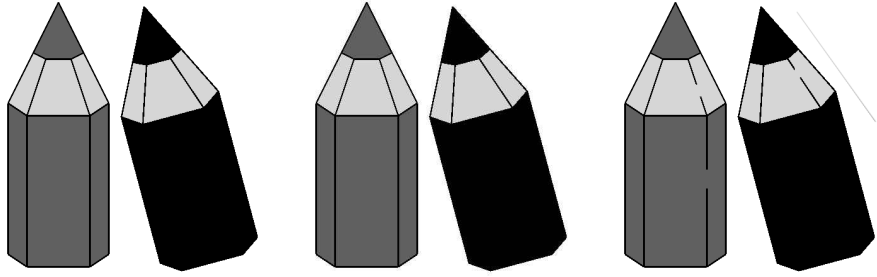
$$H = -[0.25 \cdot \log_2 0.25 + 0.2 \cdot \log_2 0.2 + 0.1 \cdot \log_2 0.1 + 0.45 \cdot \log_2 0.45] \approx 1.815 \text{ bit/pixel.}$$

Měření kvality komprese

- Objektivní hodnocení
- Subjektivní hodnocení – fidelity kriteria

Měření kvality komprese

Subjektivní hodnocení



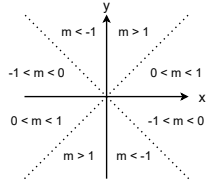
Dvourozměrné objekty

- **Liniový charakter**:
  - úsečky, lomené čáry
  - křivky
- **Plošný charakter**:
  - kružnice, elipsy
  - mnohoúhelníky
- **Rasterizace** – vykreslení objektu do rastru

Dvourozměrné objekty

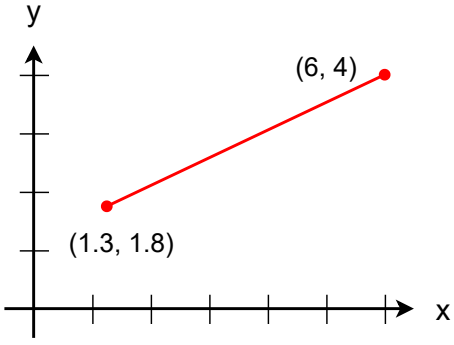
Úsečka a lomená čára

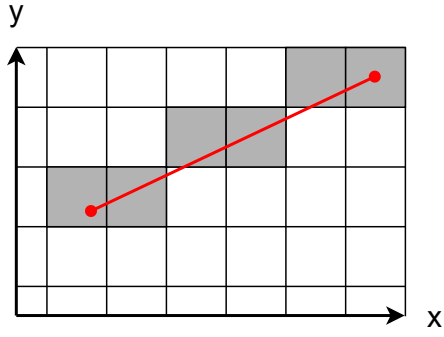
- **Úsečka** – část přímky
- **Koncové body úsečky**: $[x_1, y_1]$ a $[x_2, y_2]$
- Při rasterizaci zadáváme – barvu, tloušťku a styl
- Sousednost pixelů hraje roli
- **Obecná rovnice přímky**: $ax + by + c = 0$
- **Parametrická rovnice přímky**: $x = a_1 + t \cdot u_1$, $y = a_2 + t \cdot u_2$
- **Směrnice přímky**: $m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$



Dvourozměrné objekty

Spojité a rastrové zobrazení čáry





Dvourozměrné objekty – úsečky

Algoritmus DDA

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:
   1. Vykresli bod $[x, zaokrouhlene(y)]$
   2. $x = x + 1$
   3. $y = y + m$

Z parametrické rovnice přímky – iterační zápis

$x_{k+1} = x_k + 1,$

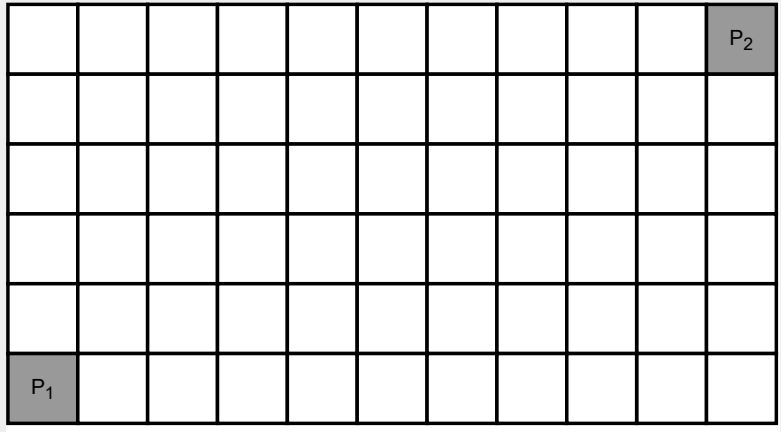
$y_{k+1} = y_k + m$

Dvourozměrné objekty – úsečky

Algoritmus DDA

Příklad

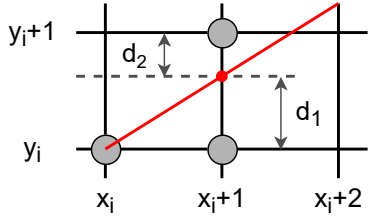
Simulujte vykreslení úsečky P_1P_2 pomocí algoritmu DDA.



Dvourozměrné objekty – úsečky

Bresenhamův algoritmus

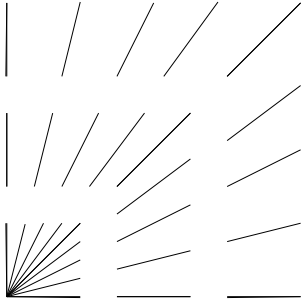
- Bod na úsečce $[x_i, y_i]$
- Následující bod $[x_i + 1, y_i]$ nebo $[x_i + 1, y_i + 1]$
- **Midpoint** – reálný bod s x-ovou souřadnicí $x_i + 1$
- Vybereme ten bod, který je blíže

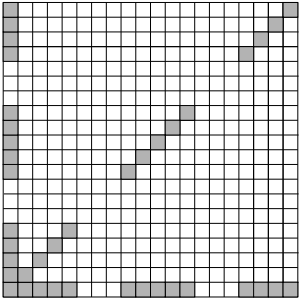


Dvourozměrné objekty – úsečky

Přerušovaná čára

- **Zadání**: pomocí úseků – plný a prázdný
- **Vzor**: několik úseků

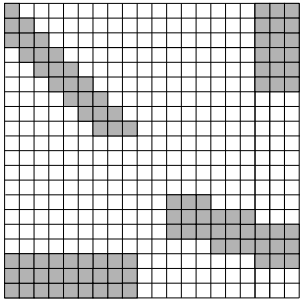


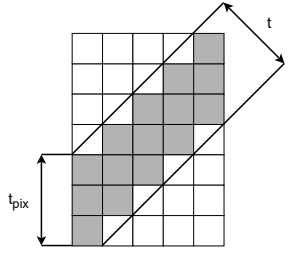


Dvourozměrné objekty – úsečky

Silná čára

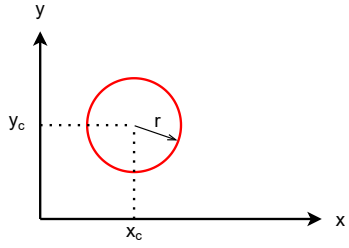
- **Naivní přístup**: vykreslení více pixelů nad (vedle) sebe
- **Výpočet tloušťky ze sklonu**: $t_{pix} = t \frac{\sqrt{(\Delta x)^2+(\Delta y)^2}}{|\Delta x|}$





Dvourozměrné objekty – kružnice

- **Zadání**: střed $[x_c, y_c]$ a poloměr r
- Rasterizujeme kružnici v počátku a posuneme

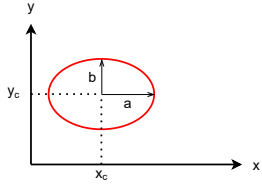


Dvourozměrné objekty – kružnice

Elipsa

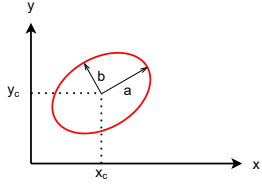
Elipsa – hlavní a vedlejší osy rovnoběžné s osami x a y

- **Zadání**: střed $[x_c, y_c]$ a velikost obou poloos a a b
- Rasterizujeme v počátku a pak posouváme



Elipsa – obecně orientovaná hlavní a vedlejší osa

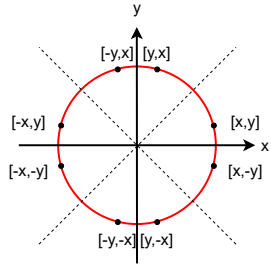
- **Zadání**: střed $[x_c, y_c]$ a poloměr r
- Rasterizujeme jako elipsu s osami rovnoběžnými a pak otáčíme



Dvourozměrné objekty – kružnice

Rasterizace kružnice

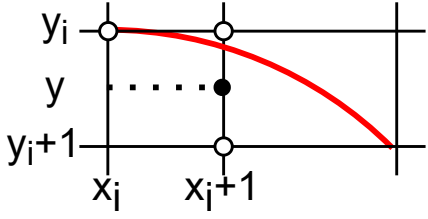
- Rasterizace:
  - Série lomených čar
  - Rasterizace kružnice
- Kružnice je symetrická podle středu
- Stačí rasterizovat 1/8 bodů



Dvourozměrné objekty – kružnice

Bresenhamův algoritmus

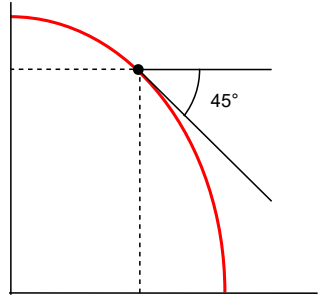
- **Midpoint algoritmus**
- rasterizujeme 1. oktant
- x-ová souřadnice bodů na kružnici se vždy zvyšuje o 1
- y-ová je stejná nebo o 1 menší
- začínáme v bodě $[0, r]$
- končíme v bodě, kde $x = y$



Dvourozměrné objekty – kružnice

Rasterizace elipsy

- Elipsa je symetrická podle středu
- Stačí rasterizovat 1/4 bodů
- Řídící osa se v průběhu výpočtu mění
- V bodě, kde je směrnice tečny -1
- **Bod**: $[\frac{a^2}{\sqrt{a^2+b^2}}, \frac{b^2}{\sqrt{a^2+b^2}}]$
- Rasterizace obdobná jako u kružnice



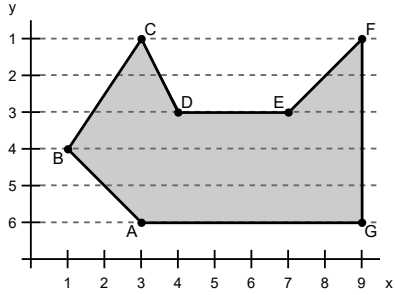
Oblast

- **Oblast** (area):
  - hranice
  - vnitřní body
- **Zadání hranice**:
  - **geometricky určená hranice** – posloupnost bodů, na sebe navazující křivky
  - **hranice nakreslená v rastru** – zadává se vlastností hranice
- **Rasterizace**:
  - Vykreslení hranice
  - Vyplnění (obarvení) vnitřních bodů (plná, šrafovaná, vzorek)

Oblast

Geometricky zadaná oblast – Řádkové vyplňování

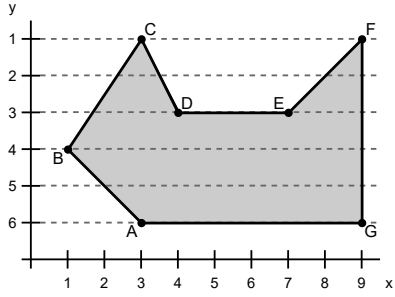
- Vyplňování rozkladovými řádky
- Každým řádkem vedeme vodorovnou čáru
- Hledáme průsečíky, které seřadíme podle x
- Vyplníme oblasti mezi lichými a sudými průsečíky



Oblast

Řádkové vyplňování

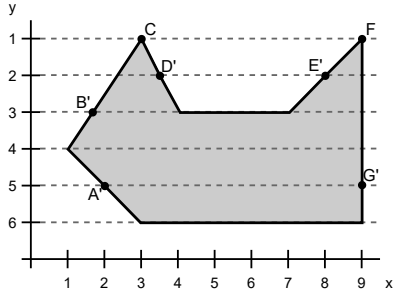
- 1. řádek protíná BC, CD, EF a FG
- dvojice průsečíků (BC a CD; EF a FG)
- 2. řádek protíná BC, CD, EF a FG
- 3. řádek hrana DE nekonečně mnoho průsečíků
- vodorovné hrany vynecháváme
- 4. řádek protíná BC, AB a FG
- zkracujeme úsečky o 1 pixel zdola



Oblast

Řádkové vyplňování

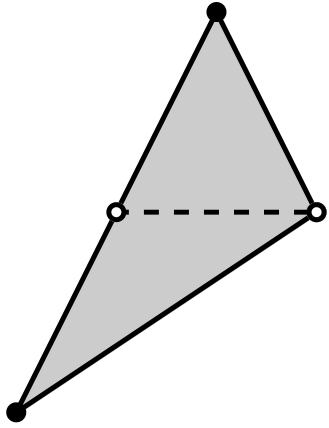
- 1. řádek protíná BC, CD, EF a FG
- dvojice průsečíků (BC a CD; EF a FG)
- 2. řádek protíná BC, CD, EF a FG
- 3. řádek protíná BC a FG
- 4. řádek protíná AB a FG



Oblast

Vyplňování trojúhelníka

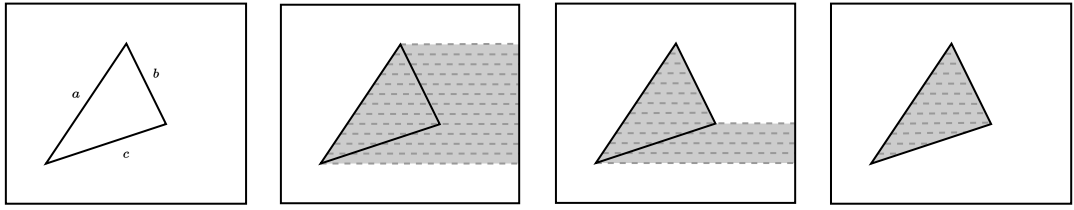
- Řádkové vyplňování
- Pokud je jedna z hran rovnoběžná s x – velice snadné
- Obecné trojúhelníky dělíme na dva



Oblast

Inverzní vyplňování

- **Šablona** – pomocný binární obraz
- Postupně zpracováváme hrany a měníme šablonu



Oblast

Vyplnění oblasti vzorem

- Pravidelný vzor – šrafování
- Vzor zadaný rastrem

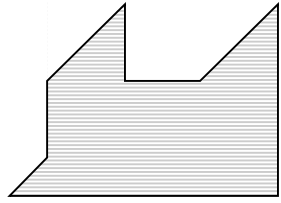
Oblast

Vzor zadaný rastrem

- **Vzor** – V, (velikost $m \times n$)
- pixelu oblasti na souřadnicích (x, y) přiřadíme hodnotu $V(x \mod m, y \mod n)$

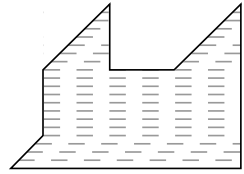
Oblast
Šrafování

- **Vodorovné šrafy** – řádkové vyplňování
- změna kroku změny v y souřadnici z hodnoty 1 na m

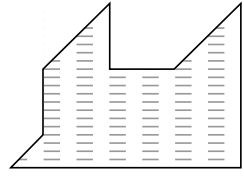


Oblast
Šrafování

- **Šrafování úseček na řádcích** – řádkové vyplňování
- **Naivní přístup** – od průsečíku vykreslujeme přerušovanou čáru



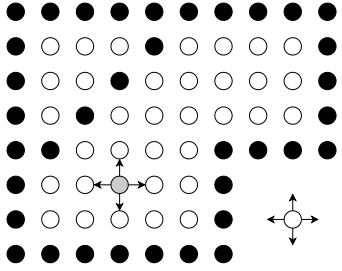
- **Lepší přístup** – šrafujeme relativně od zadaného bodu
- **Šrafování pod obecným úhlem** – otočení oblasti

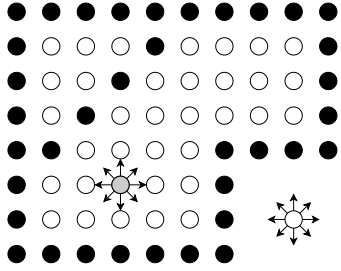


Oblast

Oblast s hranicí zadanou v rastru

- **Zadání hranice**:
  - definovaná nějakou vlastností (například barvou)
  - oblast jako pixely splňující nějakou vlastnost
- od zadaného bodu – **semínko** – vyplňujeme oblast než narazíme na hranici
- záleží na sousednosti – 4-sousedné a 8-sousedné oblasti

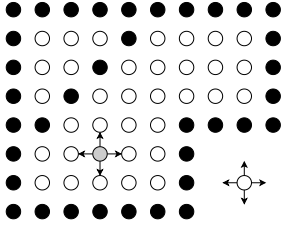


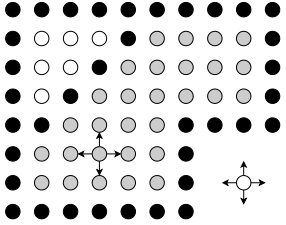


Oblast

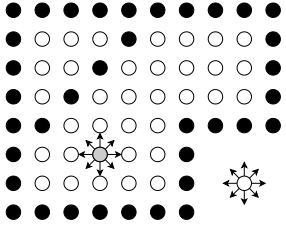
Oblast s hranicí zadanou v rastru

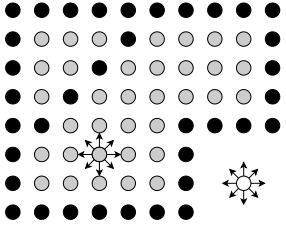
4-sousedná oblast





8-sousedná oblast





Oblast

Rekurzivní semínkové vyplňování

UmístiSemínko(x,y)

1. Pokud je bod $[x, y]$ vnitřním bodem a dosud nebyl obarven, pak
   1. Obarvi bod $[x, y]$
   2. UmístiSemínko(x+1,y)
   3. UmístiSemínko(x-1,y)
   4. UmístiSemínko(x,y+1)
   5. UmístiSemínko(x,y-1)

Oblast

Vyplňování oblasti vzorem

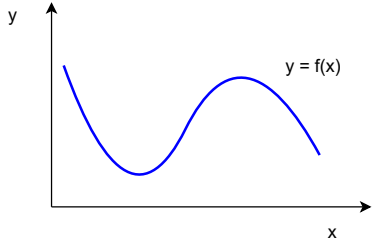
- **Vzor v rastru** – stejný princip, jako u hranice zadané geometricky
- Problém – vzor obsahuje pixel hodnoty, jakou je definovaná hranice
- Používá se maska pro uchování informace, zda byl pixel zpracován
- **Šrafování** – zpravidla převádíme na vzor v rastru

Křivky
Zadání křivky

- **Zadání**:
  - Explicitní
  - Implicitní
  - Parametrické

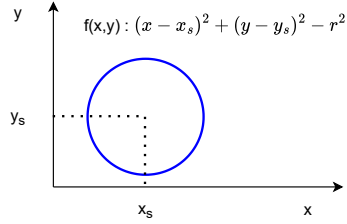
Křivky
Explicitní křivky

- $y = f(x)$



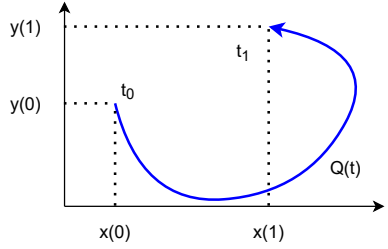
Křivky
Implicitní křivky

- $f(x, y) = 0$
- **Příklad**: kružnice
  $(x - x_s)^2 + (y - y_s)^2 - r^2 = 0$
- obtížně zobrazitelné



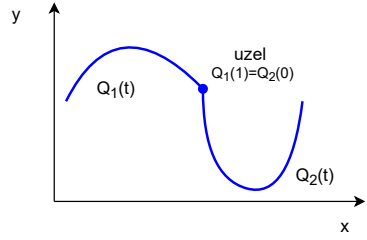
Křivky
Parametrické zadání

- Dráha bodu v čase $t \in < t_{min}, t_{max} >$
- $x = x(t),\ y = y(t)$
- **Bodová rovnice**: $Q(t) = [x(t), y(t)]$
- **Vektorová rovnice**: $\overrightarrow{q}(t) = [x(t), y(t)]$
- **Polohový vektor** $\overrightarrow{q}(t) = Q(t) - [0, 0]$
- Jednoduché zobrazení – závisí jen na jednom parametru t



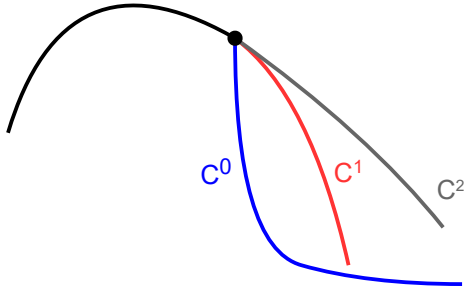
Křivky
Složitější křivky

- Složitější křivky dělíme na **segmenty**
- je nutné řešit návaznost – **uzel**
- **Tečný vektor** $\overrightarrow{q'}(t_0) = (x'(t_0), y'(t_0))$
- **Rovnice tečny**
  $P(u) = Q(t_0) + u\overrightarrow{q'}(t_0) = (x'(t_0), y'(t_0))$
- **směrový vektor přímky** $\vec{q}'(t_0)$
- **Spojitost**:
  - Parametrická
  - Geometrická
- **Třída křivky**



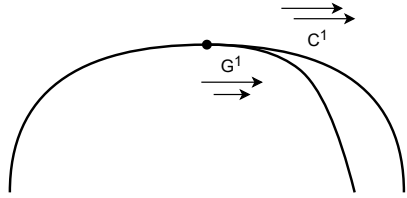
Křivky
Parametrická spojitost

- Parametrické spojitost stupně n
- **Označení**: C^n
- Ve všech bodech má spojité derivace podle parametru t do řádu n
- C^0 = segmenty jsou spojitě navázány
- C^1 = tečný vektor v koncovém bodě segmentu Q_1 je roven tečnému vektoru segmentu Q_2
- $\vec{q_1}^{(i)}(1) = \vec{q_2}^{(i)}(0),\ \forall i = 0, \ldots, n$



Křivky
Geometrická spojitost

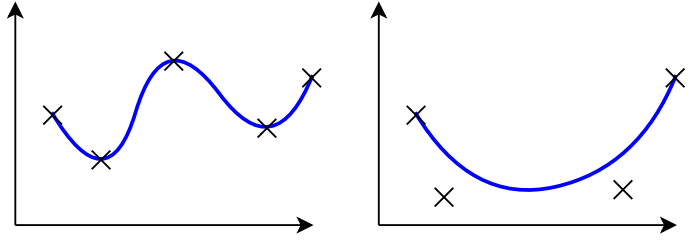
- Geometrická spojitost stupně n
- **Označení**: G^n
- Ve všech bodech má spojité derivace podle parametru t do řádu n
- C^0 = koncový bod prvního segmentu je počátečním bodem druhého
- C^1 = tečné vektory $\vec{q}_1(1)$ a $\vec{q}_2(0)$ jsou souhlasně kolineární
- $\vec{q}_1(1) = k \cdot \vec{q}_2(0)$, pro $k > 0$



Křivky

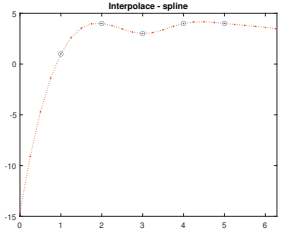
Modelování křivek

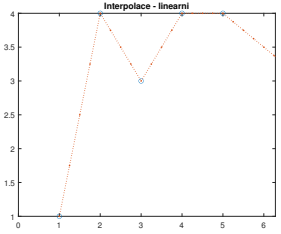
- **Polynomiální křivky**
- $Q_n(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$
- Modelují se pomocí **řídících bodů**
- Řídící body tvoří **řídící polynom**
- Řídící body se:
  - Interpolují
  - Aproximují

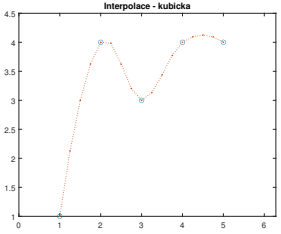


Křivky
Interpolační křivky

- **Matlab**: `vystup_y = interp1(vstup_x, vstup_y, vystup_x, metoda);`



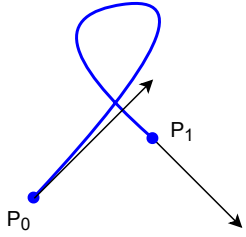


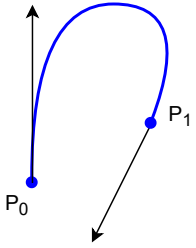


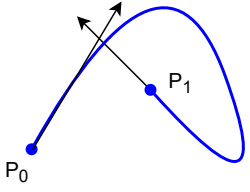
Křivky

Hermitovské kubiky

- Interpolační křivky
- **Řídící body**: P_0 a P_1
- **Tečné vektory koncových bodů**: $\vec{p_0}'$ a $\vec{p_1}'$
- $P(t) = (2t^3 - 3t^2 + 1)P_0 + (t^3 - 2t^2 + t)\vec{p_0}' + (-2t^3 + 3t^2)P_1 + (t^3 - t^2)\vec{p_1}'$







Křivky
Aproximační křivky

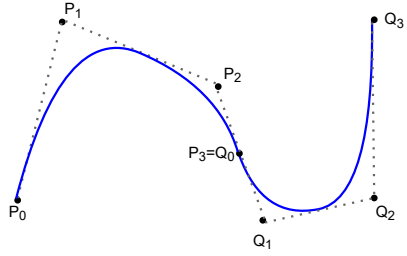
- Křivka nemusí procházet body, jen se k nim blížit
- **Příklad**:
  - Beziérovy křivky
  - Coonsovy křivky
  - Spline křivky

Křivky
Beziérovy křivky

- **Použití** – definice fontů
- Křivka prochází prvním a posledním bodem a leží v konvexní obálce řídících bodů
- Invariantní vůči posunu, změně měřítka a otočení
- Změna polohy bodu má vliv na tvar celé křivky
- **Beziérova křivka stupně** n: $Q(t) = \sum_{i=1}^{n} P_i B_i^n(t)$
- B_i jsou *Bernsteinovy polynomy n-tého stupně*
- $B_i^n(t) = \binom{n}{i} t^i (1 - t^{n-i})$
- Pro $t \in < 0, 1 >$, $i = 0, \ldots, n$

Křivky
Beziérovy křivky

- Složitější křivky dělíme nejčastěji na kubiky
- **Spojitost** C^0: snadná
- **Spojitost** C^1: bod $P_n = Q_0$ středem úsečky určené body P_{n-1} a Q_1
- $\vec{q_0}' = n(Q_1 - Q_0)$
  $\vec{p_1}' = n(P_n - P_{n-1})$
- **Spojitost** G^1: P_{n-1}, $P_n = Q_0$ a Q_1 leží na jedné přímce



Křivky

Vykreslení Beziérových křivek v rastru

- **Algoritmy**:
  - naivní (neadaptivní)
  - rekurzivní algoritmus de Castlejau

Křivky

Naivní algoritmus pro vykreslení Beziérových křivek

- **Zadání křivky**: $Q(t) = \sum_{i=1}^{n} P_i B_i^n(t)$
- Postupně dosazujeme t a body spojíme úsečkami
- Δt – konstantní (nestejně dlouhé úseky křivky)

Křivky

Algoritmus de Castlejau pro vykreslení Beziérových křivek

- Výpočet bodu $Q(t) = P_{n,n}$
- $P_{j,i}(t) = (1-t)P_{j-1,i-1} + tP_{j,i-1}$
- $i = 1, \ldots, n$ a $j = i, i+1, \ldots, n$
- **Vstup**: body řídícího polygonu ($P_{i,0} = P_i$)
- **Postup výpočtu** $Q(1/2)$ ($P_{3,3}$)
- Každé rozdělení generuje dva nové řídící polygony