POKROČILÁ BITMAPOVÁ GRAFIKA

Změna měřítka

Konvoluce

Kreslení do obrázku

Průhlednost

Kompozice

Vodoznak

Bitmapové formáty

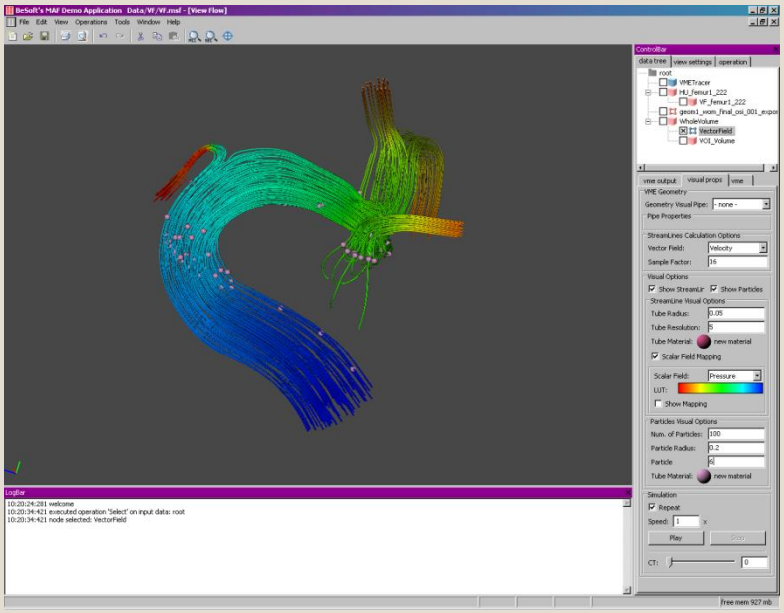
Dávkové zpracování bitmapové grafiky

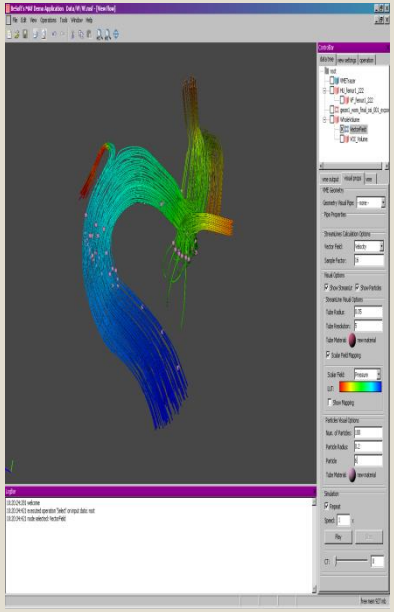
ZMĚNA MĚŘÍTKA

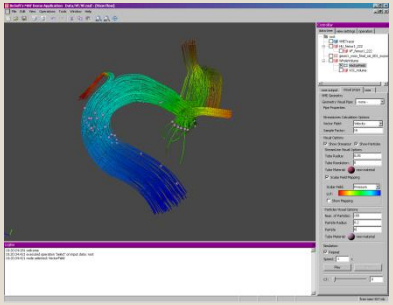
- Změna velikosti = změna počtu pixelů

ZMĚNA MĚŘÍTKA

- Obvyklý postup:
  - Změna na ose x
  - Změna na ose y







ZMĚNA MĚŘÍTKA

- Předpoklad (pro jednoduchost):
  - $M \times N \leftrightarrow k \cdot M \times l \cdot N; k, l \in N$
- Zvětšení 4× metodou nejbližšího souseda
  - Nearest neighbour
  - [00 40 80 80] →
    [**00** 00 00 00 **40** 40 40 40 **80** 80 80 80 **80** 80 80 80]
  - Nejjednodušší, nejrychlejší
  - Obvykle nevhodné

ZMĚNA MĚŘÍTKA

- Zvětšení 4× metodou (bi)lineární interpolace
  - [00 40 80 80] →
    [**00** 10 20 30 **40** 50 60 70 **80** 80 80 80 **80** 80 80 80]
  - Jednoduché a rychlé
  - Standardní volba pro většinu grafických knihoven
  - Vhodné jen pro malá zvětšení

ZMĚNA MĚŘÍTKA

- Zvětšení 4× metodou (bi)kubické interpolace
  - Pro každou dvojici sousedních pixelů najít kubickou funkci f(t) takovou, že f(0) = první pixel, f(1) = druhý
    - Krajní pixely typicky nutné duplikovat
  - Stanovit hodnoty f(0.25), f(0.5) a f(0.75)
  - Zaokrouhlit hodnoty na celá čísla
  - [00 40 80 80] →
    [**00** 04 15 28 **40** 52 65 76 **80** 86 85 82 **80** 80 80 80]
  - Vhodné zejména pro větší zvětšení
  - Lépe zachovává hrany
  - Podstatně pomalejší

ZMĚNA MĚŘÍTKA

- Zvětšení k×
  - Analogické
- Zmenšení 2× triviální metodou
  - Jedná se o princip vzorkování
  - Každý druhý pixel vynechat
    - [**00** 00 **40** 40 **00** 00 **40** 40] → [00 40 00 40]
  - Nejjednodušší možné
    - Totéž provádí standardně grafická karta pro převod 3D scény na rastrovou obrazovku
    - Dtto digitální fotoaparát

ZMĚNA MĚŘÍTKA

- Použitelné v praxi pro velmi rychlé náhledy
- Může vést k aliasingu
  - [**00** 00 **00** 40 **40** 40 **00** 00 **00** 40 **40** 40] → [00 00 40 00 00 40]
  - [**00** 40 **00** 40 **00** 40] → [00 00 00]

ZMĚNA MĚŘÍTKA

- Aliasing
  - vzniká při vzorkování obrazu s opakujícím se vzorem, pokud vzorkovací frekvence f_s je "srovnatelná" s frekvencí f_p opakování vzoru
  - f_s = 1 / vzdálenost středů sousedních pixelů
  - f_p = 1 / vzdálenost začátků opakovaného vzoru

ZMĚNA MĚŘÍTKA

- Odstranění aliasingu
  - Úplné odstranění není prakticky možné
- Potlačení aliasingu (tzv. anti-aliasing)
  - Aliasing nebude tolik rušivý
    - Ideálně ho ani nepostřehneme
  - Metoda tzv. super-samplingu:
    - Vytvoř bitmapový obraz k× větší, než požadovaný
    - Změň měřítko, přičemž proveď "průměrování" sousedních pixelů pro stanovení barvy výsledného pixelu

ZMĚNA MĚŘÍTKA

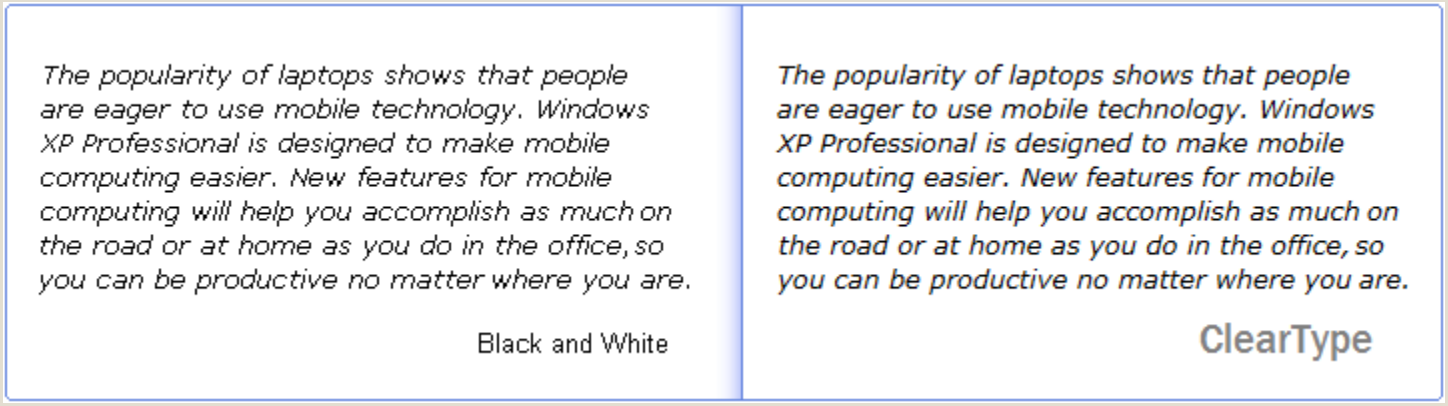
- Zmenšení obrazu 2× (bi)lineární filtrací
  - Nejjednodušší je průměrováním dvojic pixelů
    - [**00** 40 **00** 40 **00** 40] →
      [20 20 20]
  - Vhodnější je vážený průměr přes sousedy
    - Např. $I'(x) = 0.25 \cdot I(x-1) + 0.5 \cdot I(x) + 0.25 \cdot I(x+1)$
    - [00 40 00 40 00 40] → [*00* 00 40 00 40 00 40 *40*] →
      [**10** 20 **20** 20 **20** 30] → [10 20 20]
    - [0 0 0 0 4 4 0 0 4 4 0 0 4 4] → [0 1 3 1 3 1]
  - Snížení kontrastu a rozmazání původních hran
    - Problém zejména u textu

ZMĚNA MĚŘÍTKA

- Relativně rychlé
  - Grafické knihovny provádí často na požádání pro rasterizaci vektorové grafiky
  - `g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);`

ZMĚNA MĚŘÍTKA

- ClearType
  - Microsoft
    - Automaticky používáno od Windows Vista+
  - Využívá znalostí rozložení R, G, B prvků displeje
    - Většinou umístěny vedle sebe
  - Změna R, G, B pro dosažení efektu "průměrování"
    - Není průměrován tedy celý pixel



ZMĚNA MĚŘÍTKA

- Bez anti-aliasingu, Anti-aliasing, ClearType

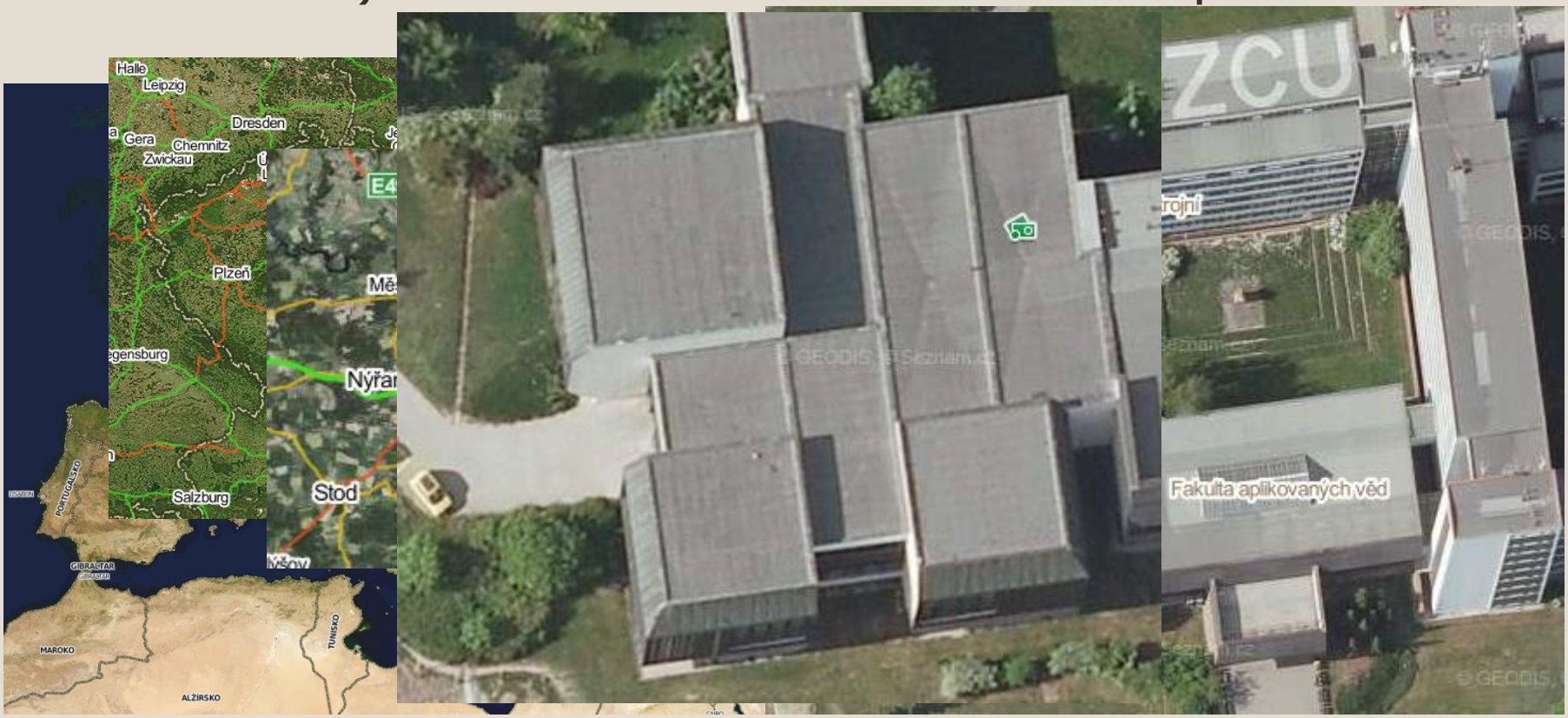
```
g2.setRenderingHint(
 RenderingHints.
  KEY_TEXT_ANTIALIASING,
 RenderingHints.
  VALUE_TEXT_ANTIALIAS_ON
);
```

ZMĚNA MĚŘÍTKA

- Zmenšení obrazu 2× (bi)kubickou filtrací
  - Váhy nejsou lineární (odvozeny z kubické funkce)
  - Lépe zachovává hrany
  - Pomalejší
- Zmenšení obrazu k×
  - Analogické, jen je třeba vzít více sousedů
    - 2k-1
  - Váhy pro bilineární filtraci jsou:
    [1 2 3 … k … 3 2 1] / k^2

MIP-MAPING

- Technika pro zobrazování velkých (např. 10K×10K) obrázků na různé úrovni přiblížení



MIP-MAPING

- Obrázek rozřezán na rozumně velké bloky
  - Např. 256×256 pixelů
- Bloky zmenšeny na různé rozměry
  - Probíhá v pre-procesingu
  - Lze použít sofistikovanou metodu změny měřítka
- Bloky nejsou podmínkou

MIP-MAPING

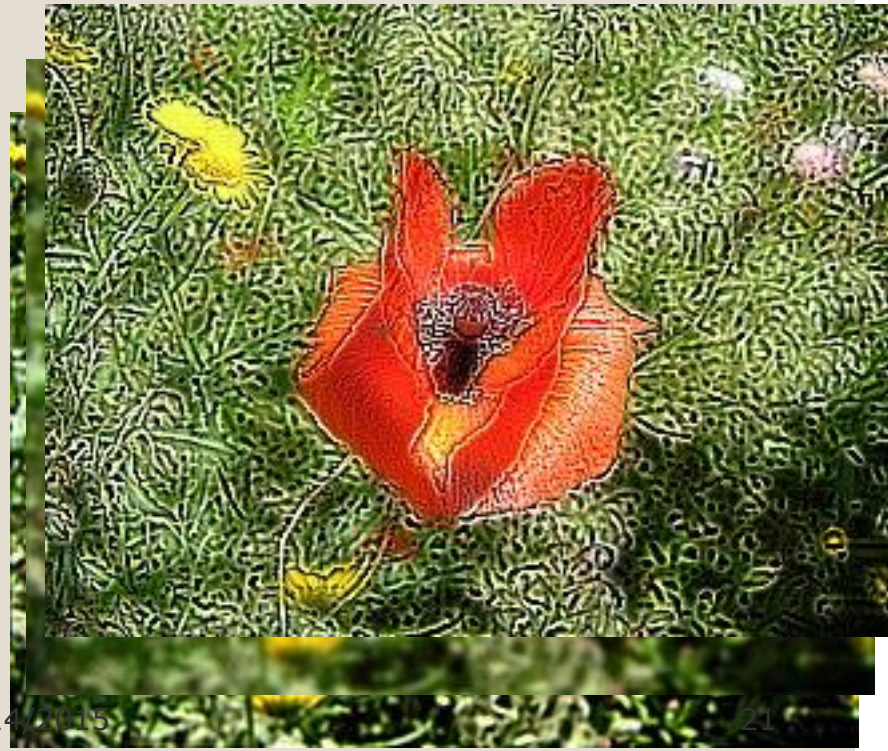
- Požadovaný výřez obrázku:
  - $[x_1, y_1, x_2, y_2]_{bmp}$
- Má být zobrazen na obrazovce do oblasti:
  - $[x_1, y_1, x_2, y_2]_{displej}$
- Dvě možnosti:
  - Vyberu nejvhodnější vygenerované rozlišení a jeho výřez zobrazím (např. umístěním bloků vedle sebe)
    - Může být zobrazen větší výřez, než požadován
    - Oblast může být nezaplněna
    - Skoková změna

MIP-MAPING

- Vyberu nejvhodnější vyšší rozlišení a změním jeho měřítko tak, aby to vyhovovalo požadavkům
  - Složitější a časově náročnější
  - Vhodné pro animace
  - Změny nejsou patrné

KONVOLUČNÍ FILTROVÁNÍ

- Používá se např. pro:
  - Rozmazání načteného obrázku pro eliminaci šumu
    - Nejčastěji Gaussovský filtr
  - Detekce hran v obrázku
  - Vytvoření vrženého stínu

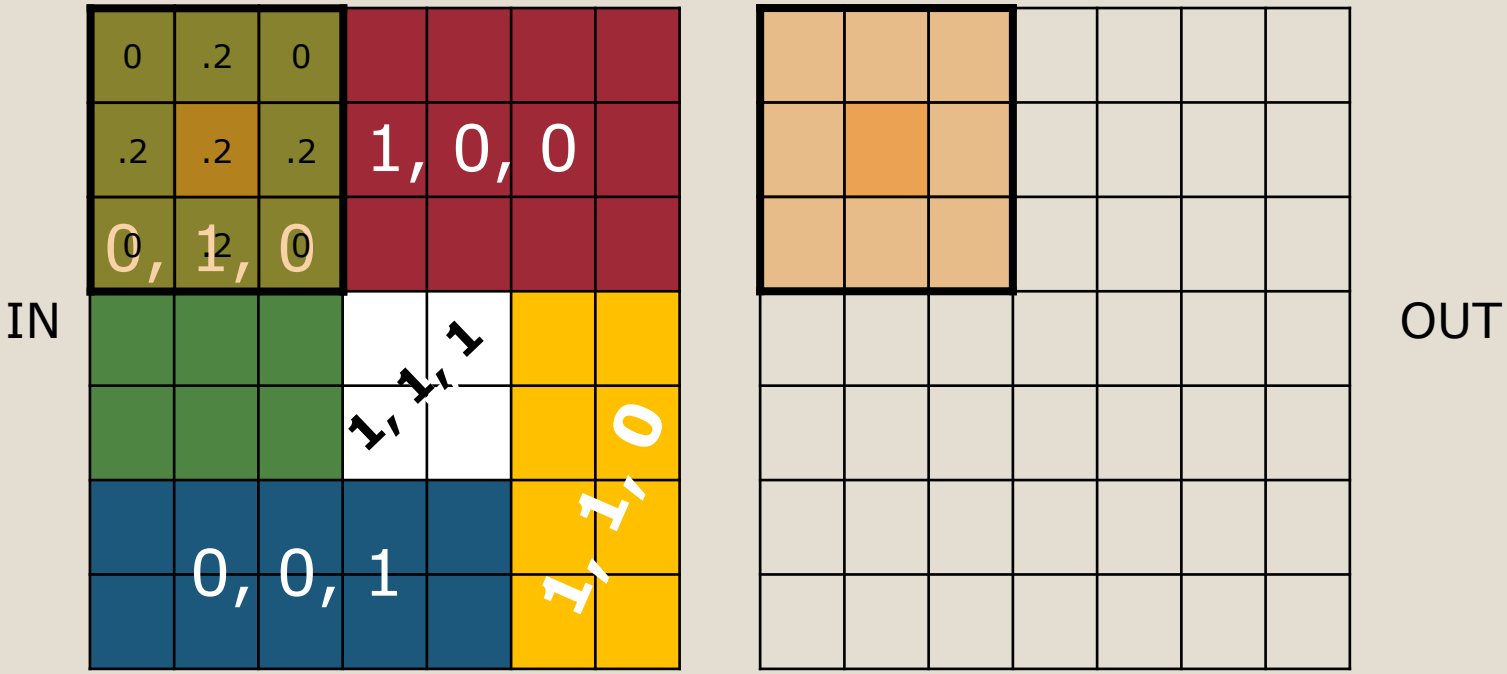


KONVOLUČNÍ FILTROVÁNÍ

- Obdoba bilineární filtrace
  - Žádné pixely nezahazujeme
  - $I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w_{i,j} \cdot I(x+i, y+j)$
  - Váhy $w_{i,j}$ určují výsledný efekt
    - Váhám se říká někdy též konvoluční jádro

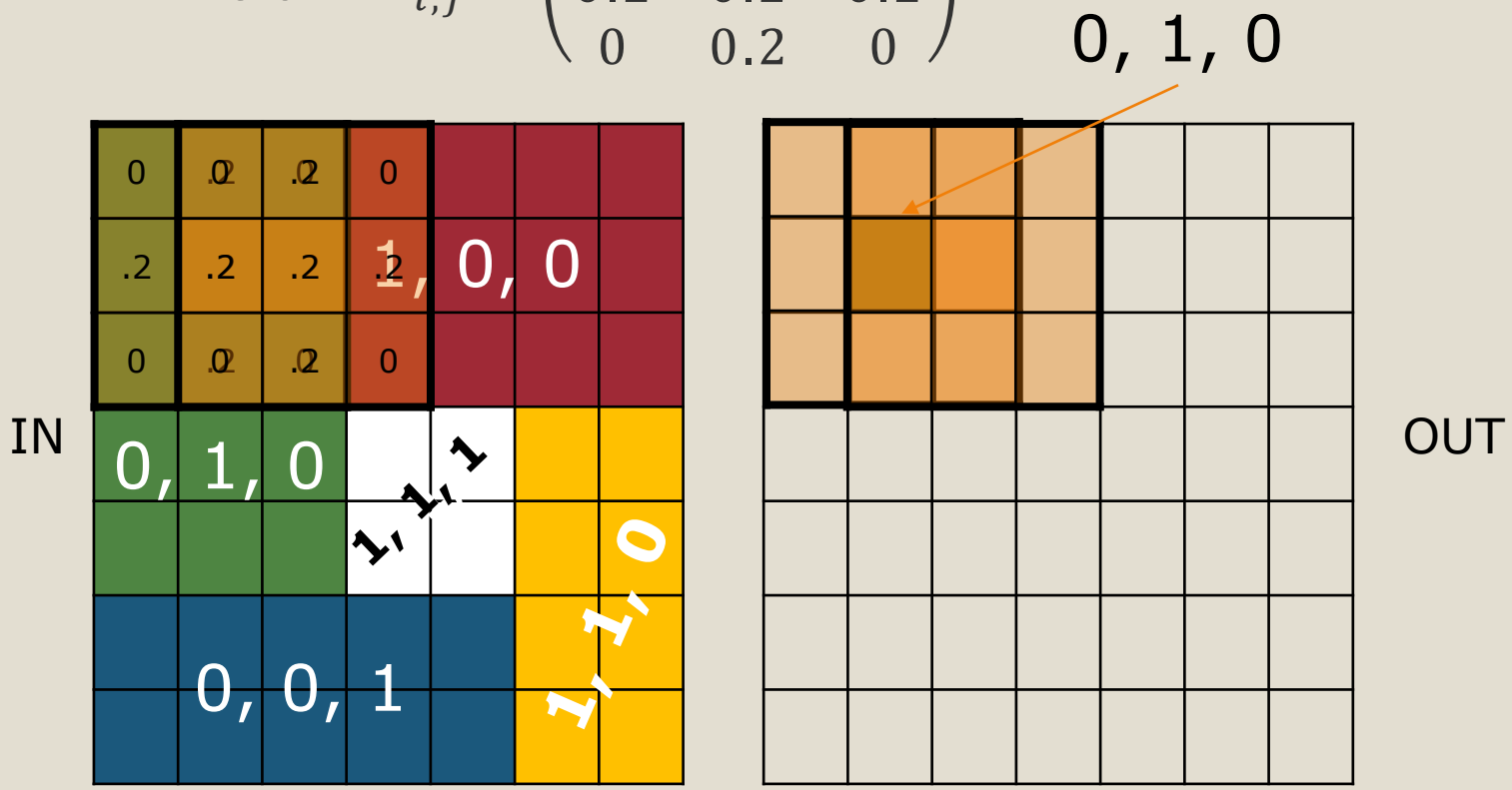
KONVOLUČNÍ FILTROVÁNÍ

- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$



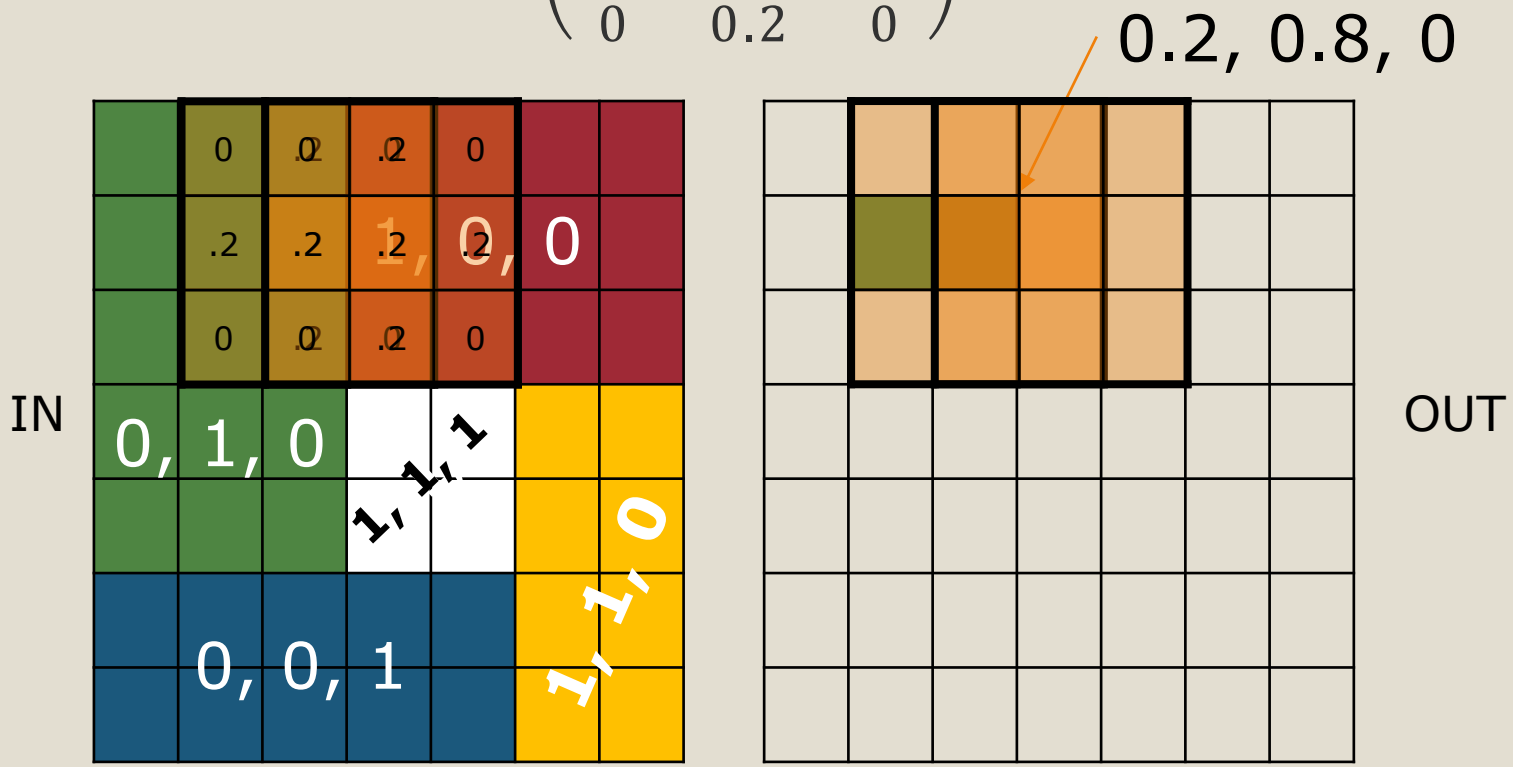
KONVOLUČNÍ FILTROVÁNÍ

- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$



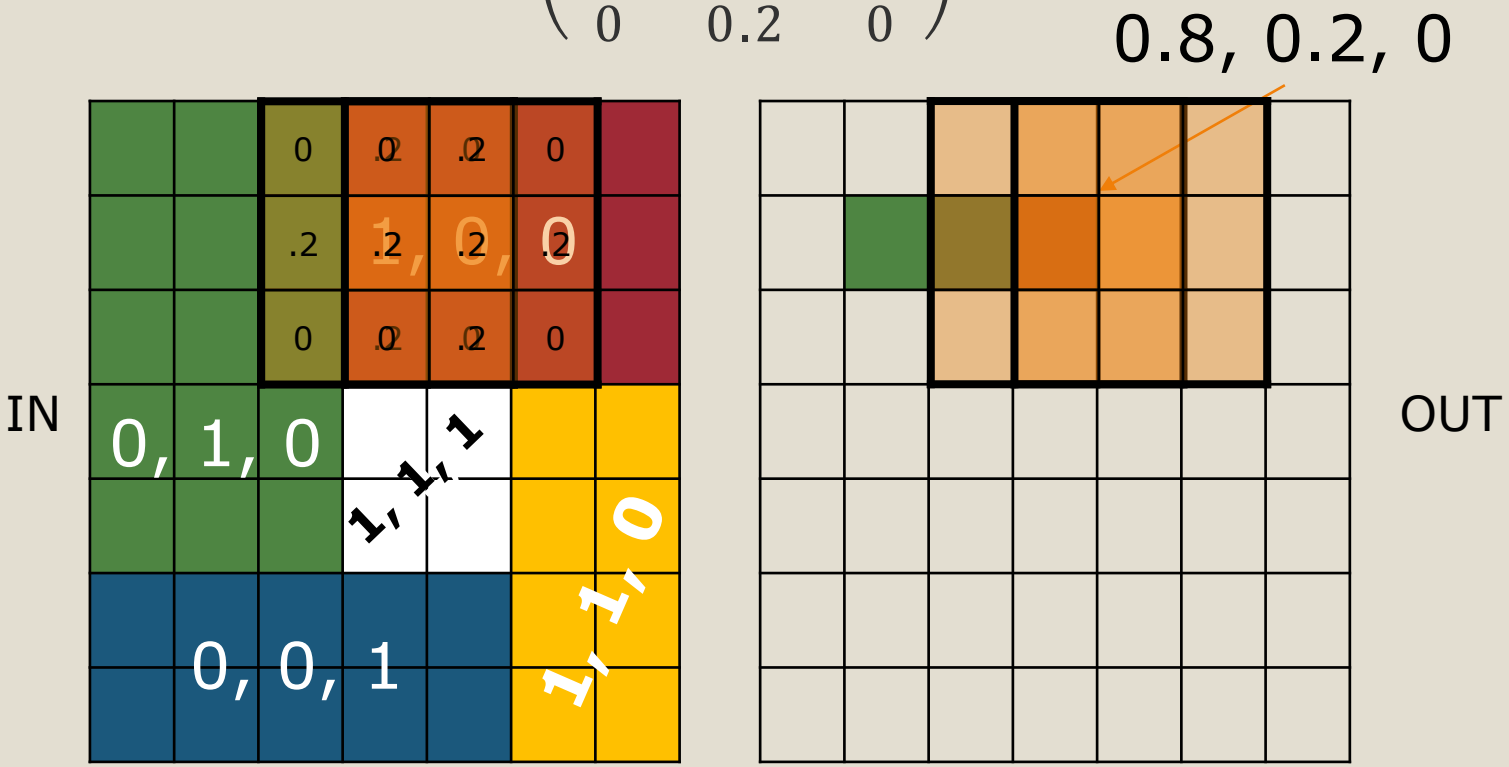
KONVOLUČNÍ FILTROVÁNÍ

- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$



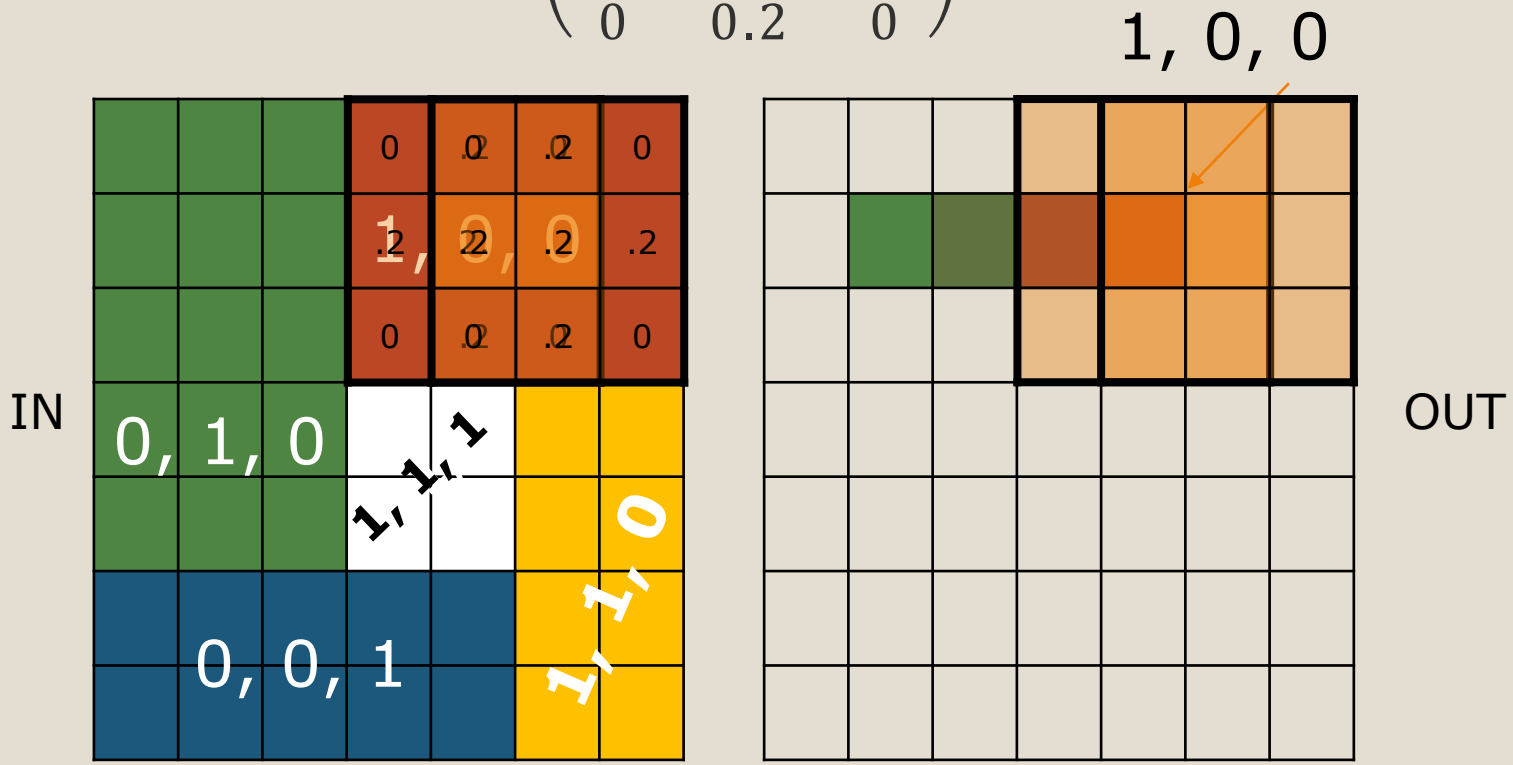
KONVOLUČNÍ FILTROVÁNÍ

- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$



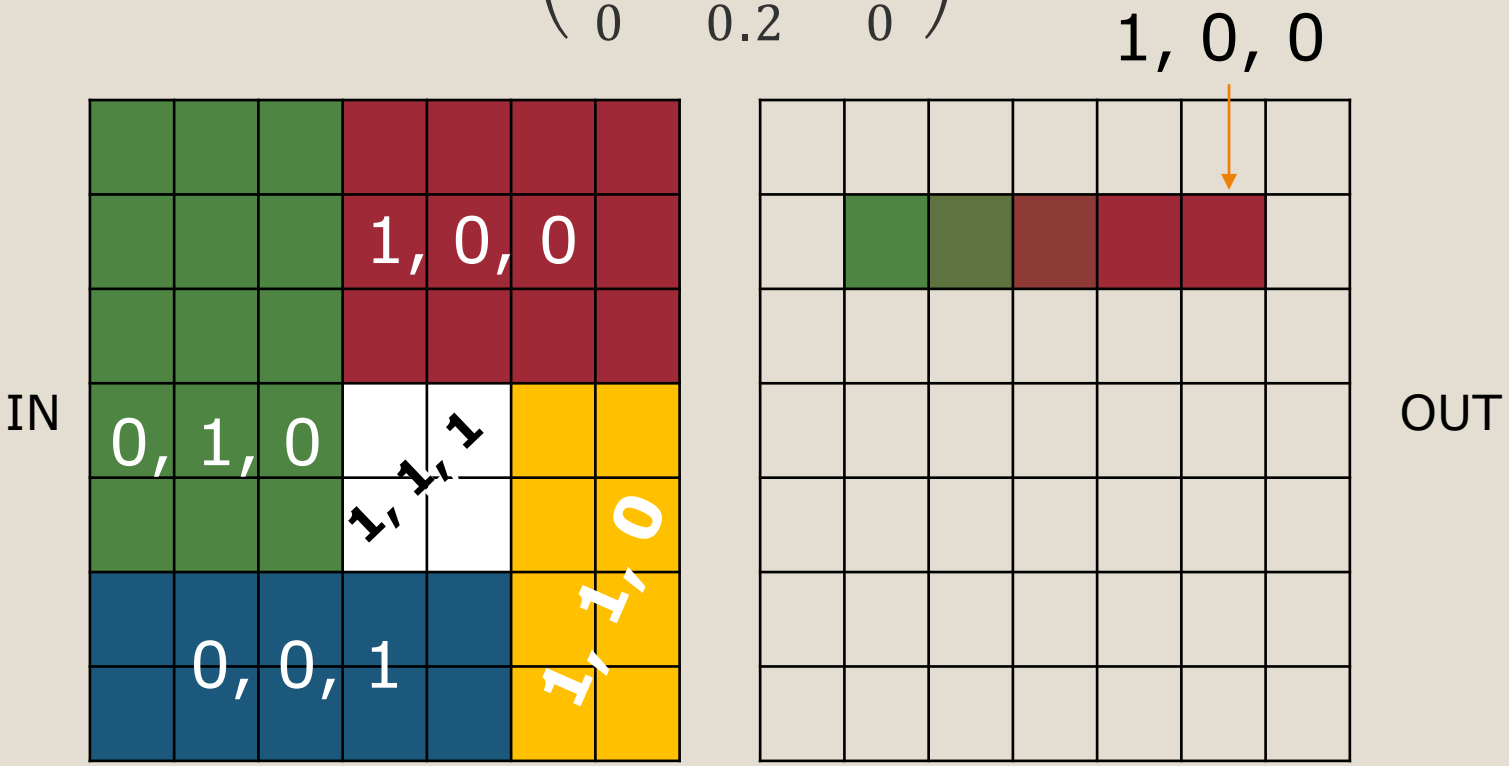
KONVOLUČNÍ FILTROVÁNÍ

- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$



KONVOLUČNÍ FILTROVÁNÍ

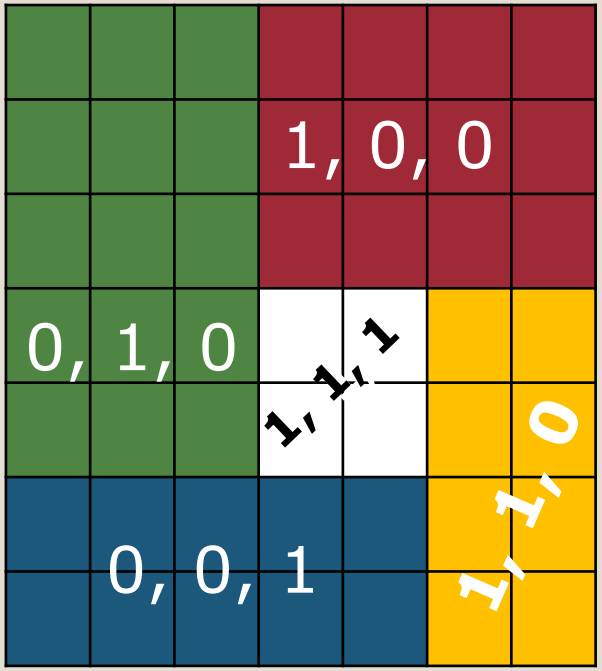
- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$

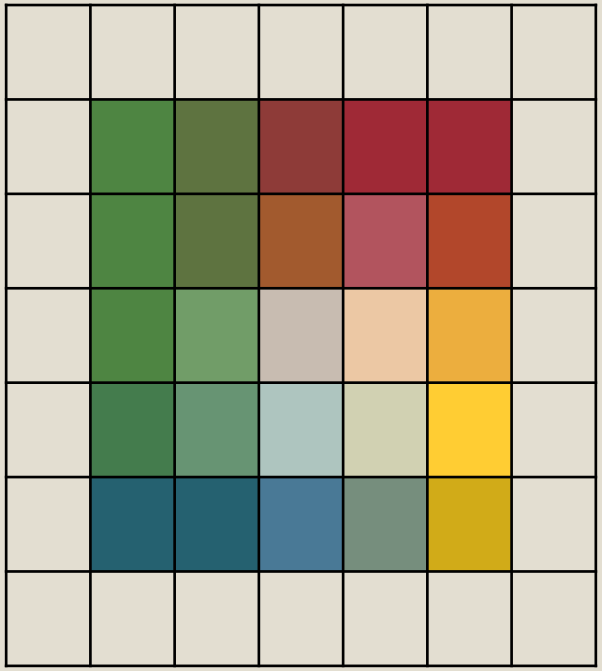


KONVOLUČNÍ FILTROVÁNÍ

- Příklad: $w_{i,j} = \begin{pmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{pmatrix}$

IN

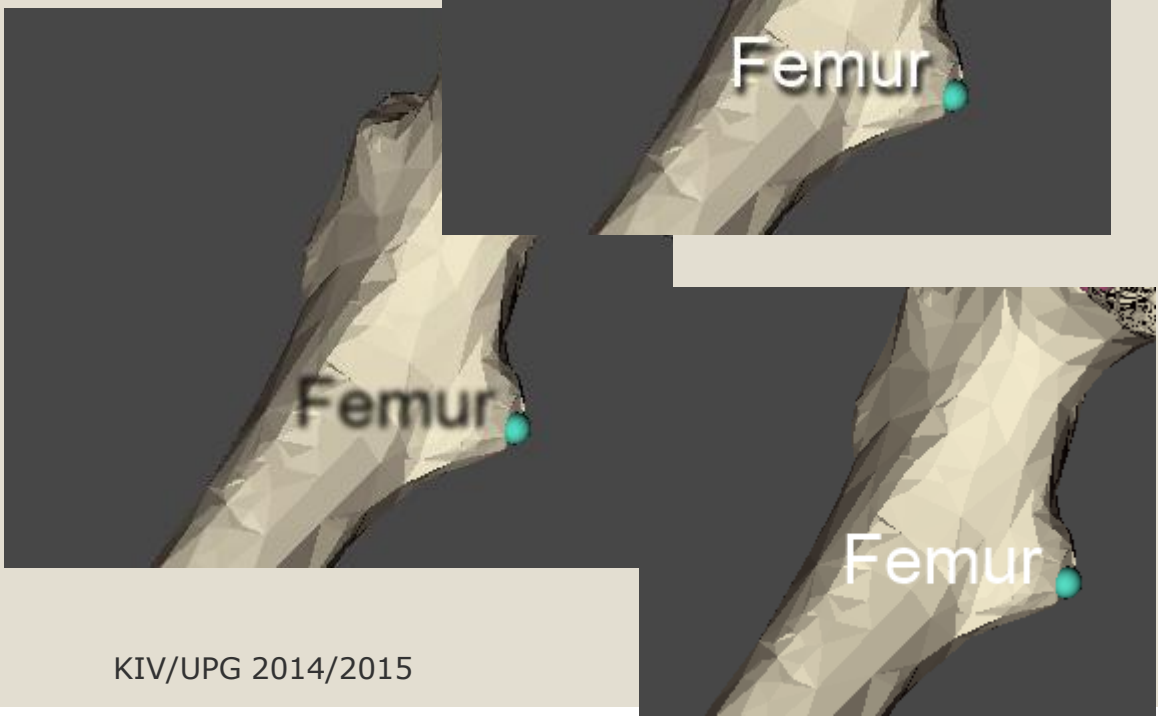




OUT

KONVOLUČNÍ FILTROVÁNÍ

- Gaussovský filtr
  - $w_{i,j} = e^{-\lambda \cdot \left(\frac{i}{k}\right)^2} \cdot e^{-\lambda \cdot \left(\frac{j}{k}\right)^2}$
  - $\frac{1}{16} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

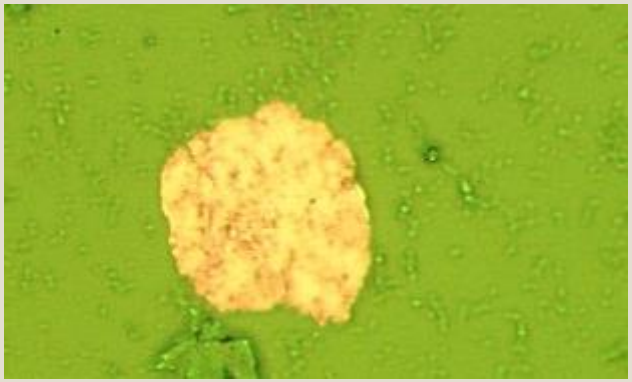


KONVOLUČNÍ FILTROVÁNÍ

```
float[] matrix = {
    0.111f, 0.111f, 0.111f,
    0.111f, 0.111f, 0.111f,
    0.111f, 0.111f, 0.111f,
};

ConvolveOp op = new ConvolveOp(
      new Kernel(3, 3, matrix));

BufferedImage result = op.filter(m_image, null);

g2.drawImage(result, 0, 0, null);     //normal 1:1
```

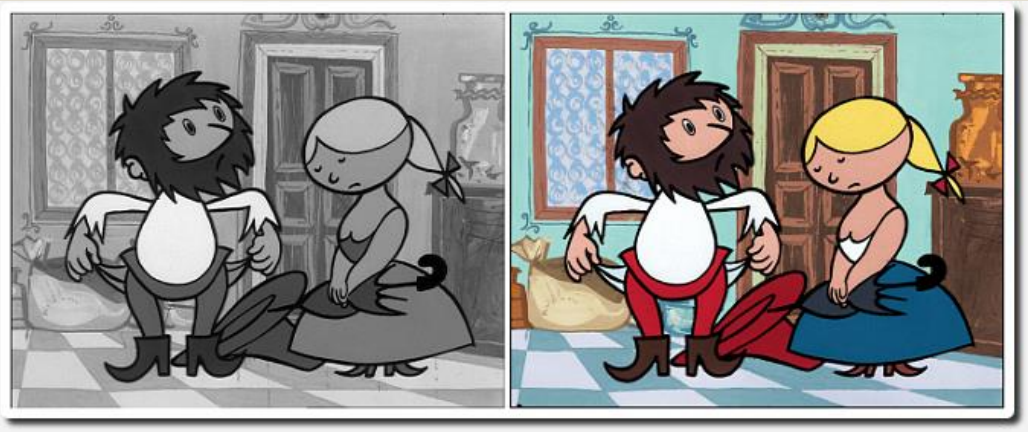
KRESLENÍ DO BITMAPY

- Např. doplnění popisků, anotace, …
- Dvě možnosti
  - Manuálně (programově)
  - Automaticky přes grafický kontext
- Manuálně
  - Programátor mění obsah jednotlivých pixelů bitmapy
  - Např. rozpoznávání obrazu
    - Chci označit pixely patřící cizí částici ve snímku oleje
    - Automatické orámování částice
  - Např. přebarvení obrázku



KRESLENÍ DO BITMAPY

- Např. automatické přebarvení obrázku
- Odebrání červených očí



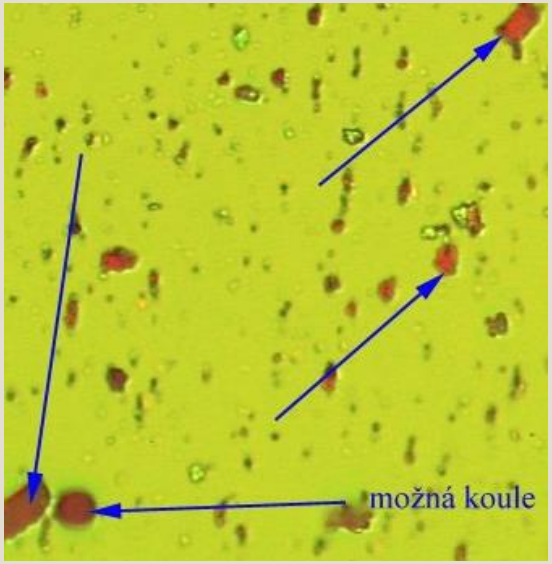
KRESLENÍ DO BITMAPY

- Automaticky
  - Nad bitmapou se vytvoří grafický kontext
    - Obvykle existuje metoda pro zavolání nad bitmapou
  - Kreslím na grafický kontext voláním metod pro kreslení vektorové grafiky


```
BufferedImage img = new BufferedImage(
  640, 480, BufferedImage.TYPE_3BYTE_BGR);

Graphics2D gimg = img.createGraphics();

gimg.setPaint(Color.RED);
gimg.fillRect(0, 0, img.getWidth() / 2, img.getHeight());

g2.drawImage(img, 0, 0, this.getWidth(), this.getHeight(), null); //scaled
```

KRESLENÍ DO BITMAPY

- Např. doplnění vektorové grafiky do obrázku
  - Popisky, křivky silnic, šipky, …
- Klíčové pro tzv. double-buffering



DOUBLE BUFFERING

- Namísto kreslení rovnou na obrazovku, kreslím do bitmapy (o stejné velikosti) a výsledek pak zobrazím na obrazovku (1:1)
- Pomalejší, ale nezbytné pro animace
- Animace bez double-bufferingu:
  1. Smaž obrazovku (např. bílou)
  2. Vykresli objekt v aktuálním čase
  3. Zvyš aktuální čas
  4. Zpět na 1

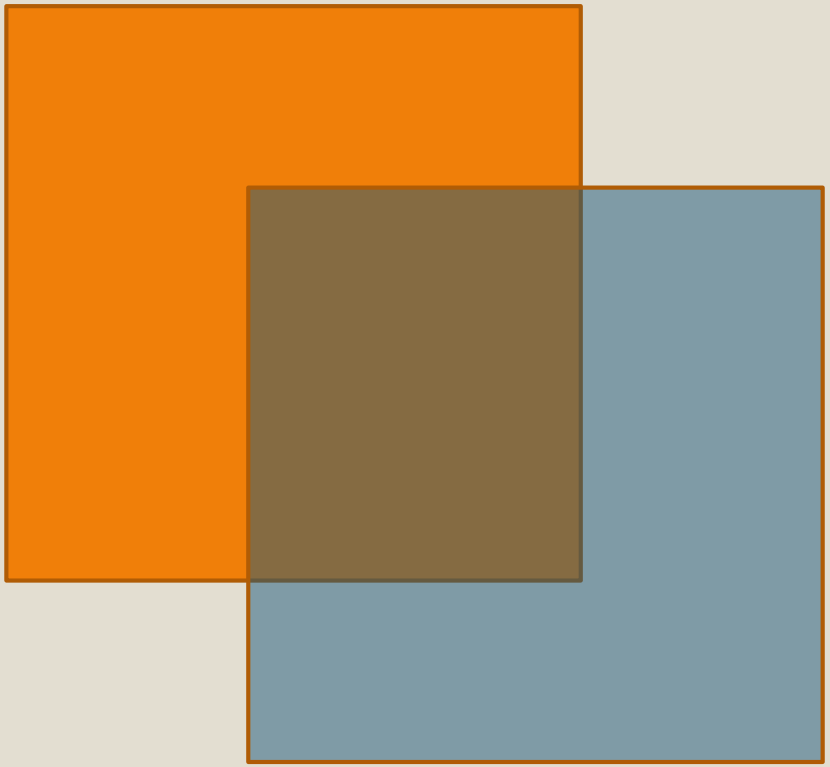
DOUBLE BUFFERING

- Animace s double-bufferingem:
  1. Vykresli objekt v aktuálním čase do bitmapy
  2. Vykresli bitmapu na obrazovku
  3. Zvyš aktuální čas
  4. Zpět na 1

PRŮHLEDNOST

- Výhodné pro snadnou specifikaci bitmapy obecného (neobdélníkového) tvaru
  - Např. spojení dvou bitmap do sebe
- Alfa (alpha) kanál = informace, zda pixel je průhledný či nikoliv
  - 0.0 = průhledný
  - 1.0 = neprůhledný
- Částečná průhlednost:
  - Alfa z intervalu 0.0 - 1.0
- V praxi alfa ukládána nejčastěji na 8 bitů
  - 256 úrovní, např. RGBA

PRŮHLEDNOST



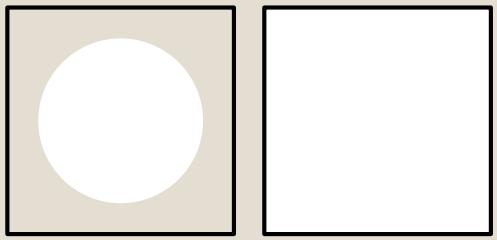
50% průhlednost

PRŮHLEDNOST

- Neprůhledný pixel v obrázku na pozadí má barvu X (r, g, b)
- Chceme přes něj zobrazit pixel barvy Y (r, g, b), který má nastavenou alfu A
- Pixel bude mít barvu $Z = (1 - A) \cdot X + A \cdot Y$
  - Pixel v obrázku bude neprůhledný

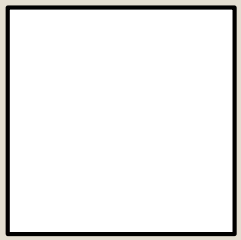
PRŮHLEDNOST

- Pozor na skládání!
- Obrázek obsahuje bílý kruh (1, 1, 1, 1) na zcela průhledném pozadí (0, 0, 0, 0)
- Obrázek zobrazuji přes neprůhledné pozadí bílé barvy (1, 1, 1)



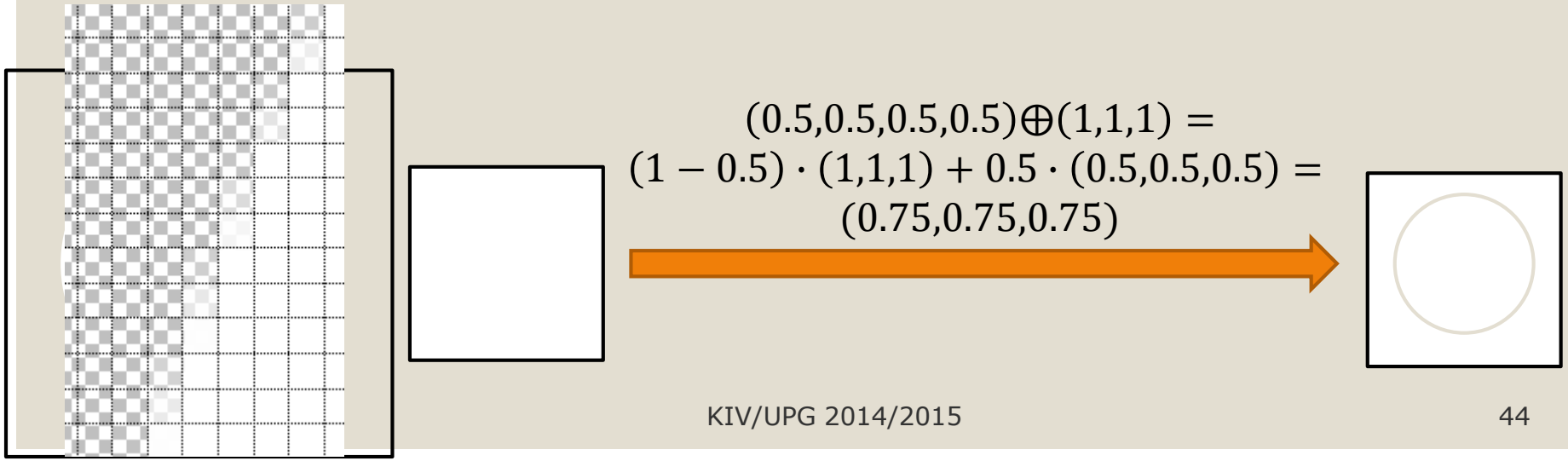
$$(1,1,1,1) \oplus (1,1,1) =$$
$$(1-1) \cdot (1,1,1) + 1 \cdot (1,1,1) = (1,1,1)$$

$$(0,0,0,0) \oplus (1,1,1) =$$
$$(1-0) \cdot (1,1,1) + 0 \cdot (1,1,1) = (1,1,1)$$



PRŮHLEDNOST

- Obrázek je větší než ten, do kterého ho chci na nějaké místo zobrazit
- Provedu jeho zmenšení bilineární filtrací
  - V obrázku vzniknou pixely (0.25, 0.25, 0.25, 0.25), (0.5, 0.5, 0.5, 0.5) a (0.75, 0.75, 0.75, 0.75)
    - Redukce na 1/4

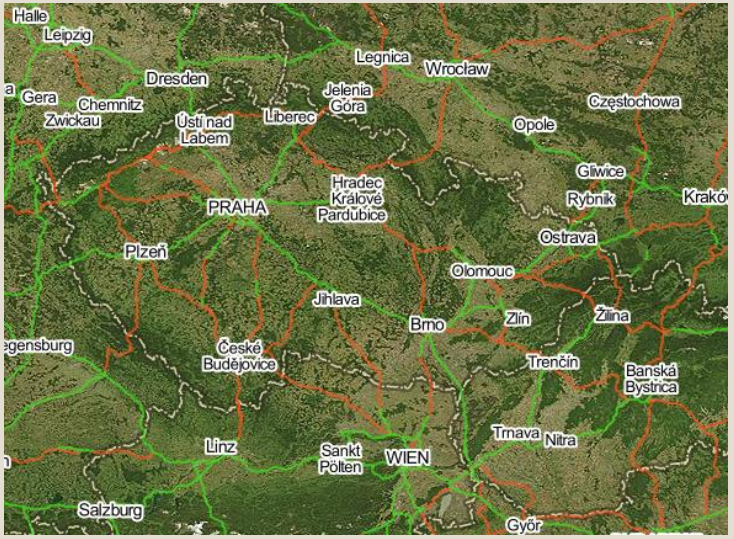


PRŮHLEDNOST

- Důvod:
  - Obrázek již obsahuje vážení alfou
    - Díky bilineární filtraci
  - Jedná se o tzv. pre-multiplied alpha obrázek
- Řešení: $Z = (1 - A) \cdot X + Y$

VRSTVY

- Výsledný obraz (např. na grafickém kontextu) vznikne složením obrázků přes sebe
  - Např. bitmapový obrázek ze satelitu na němž je rasterizovaný vektorový obrázek silnic a popisků
- Vrstvy lze ukládat / načítat odděleně
- Skládání (kompozice):
  - Z = op(X, Y)
    - op = kompoziční (blend) operátor



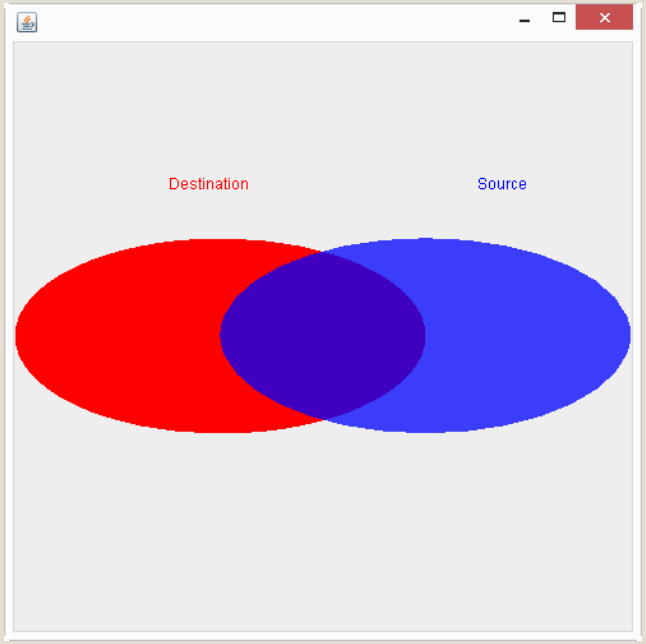
KOMPOZICE OBRAZU

- Alfa kompozice
  - Skládá se na základě alfa
  - Alfa nemusí být specifikována pro každý pixel, ale specifikuje se pro celý obrázek

```
//vykresli "dest"

//nastav, jak provádět kompozici obrazu
g2.setComposite(AlphaComposite.getInstance(
        compositeRule, alphaValue));

//vykresli "src"
```

KOMPOZICE OBRAZU

- Pravidla kompozice určují, jak se naloží s pixely z "dest" a pixely ze "src"
  - Co z toho je X a co Y
  - Jedná o pre-multiplied alpha?
- Nejčastější volba SRC_OVER
  - **DEMO**

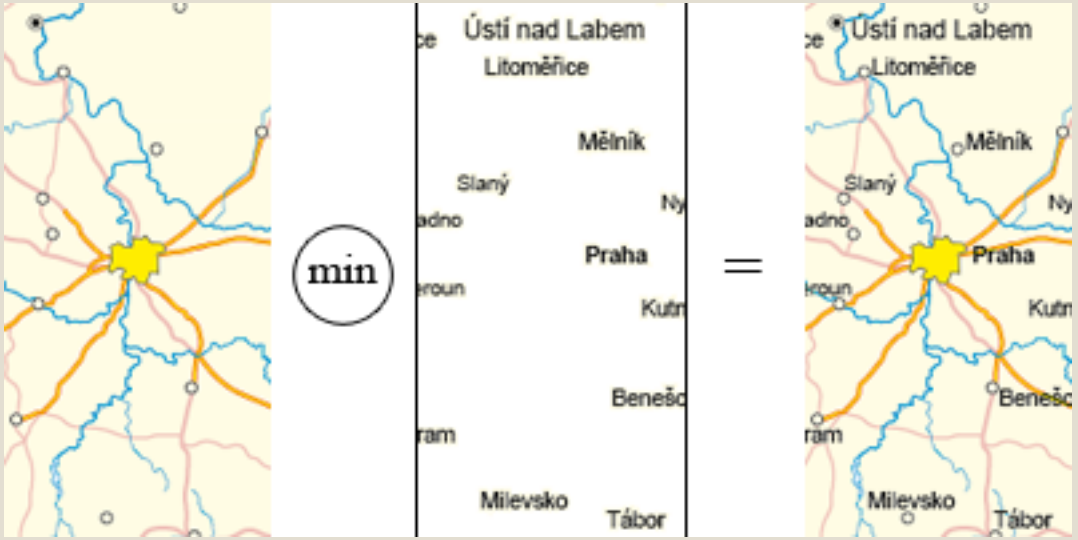


KOMPOZICE OBRAZU

- Maximum (lighten)
  - Z = max(X, Y)
    - Pozor: X, Y = <0, 1>
  - Typicky: Y je černobílý, černá = pozadí, bílá = to, co má být vidět (např. text, čáry, …)

KOMPOZICE OBRAZU

- Minimum (darken)
  - Z = min(X, Y)
  - Analogie maximum



KOMPOZICE OBRAZU

- Multiply
  - Z = X * Y
  - Lokální ztmavení obrázku X, Y určuje míru ztmavení
  - Často se též volí Y = X

KOMPOZICE OBRAZU

- Existují rovněž další způsoby kompozice
  - Scr (X, Y) = (1 – MUL(1-X, 1-Y))

VODOZNAK

- Nejčastěji pro účely copyrightu
  - Mnoho obrázků na internetu nemůže být jen tak použito pro libovolný účel
  - Obsahuje typicky jméno resp. logo autora
- Vodoznak může být viditelný
  - Neměl by jít snadno zaretušovat

VODOZNAK

- Lze snadno vytvořit alfa kompozicí (alfa např. 10%)
- Vhodné pro DEMO verze, fotobanky

- Vodoznak může být neviditelný
  - Obrázek není viditelně poškozen
  - Data vodoznaku skryta
    - Modifikují se barvy pixelů
  - Různé algoritmy pro vytvoření takového vodoznaku
    - Vodoznak nelze přečíst bez znalosti algoritmu
  - Vodoznak často vytvořen tak, že data lze zrekonstruovat, ačkoliv uživatel obrázek silně pozmění
    - Oříznutí, přidání šumu, gama korekce
    - Tzv. robustní vodoznak

VODOZNAK

- Např. barevné kopírky automaticky umisťují do produkovaných kopií neviditelný vodoznak obsahující informace o kopírce, která ho vyrobila
  - Lze dohledat místo, kde byl padělek vytvořen
- Objeví-li autor svůj vodoznakem opatřený obrázek použitý někým jiným bez jeho souhlasu, může dokázat, že je to skutečně jeho obrázek

- Neviditelný vodoznak lze rovněž použít pro ověření pravosti dat, se kterými pracuji
  - Změna jediného pixelu se pozná
  - Jedná se o tzv. křehký vodoznak

FORMÁTY BITMAPOVÉ GRAFIKY

- Bitmapovou grafiku lze obvykle
  - Uložit na disk v nějakém bitmapovém formátu
  - Načíst z nějakého bitmapovém formátu
  - Zobrazit na grafickém kontextu
    - Původní nebo odlišné měřítko
- Nejčastěji používané bitmapové formáty:
  - Windows Bitmap (BMP)
  - Graphics Interchange Format (GIF)
  - Portable Network Graphics (PNG)
  - Joint Photographic Experts Group (JPEG)
  - JPEG 2000

FORMÁTY BITMAPOVÉ GRAFIKY

- Bitmapový formát typicky obsahuje hlavičku následovanou vlastními daty (pixely)
- Hlavička
  - Signatura formátu
  - Rozlišení obrázku
  - Počet barevných složek
  - Počet bitů na jednotlivé barevné složky
  - Způsob uložení vlastních dat
    - Způsob komprese
    - Little vs. Big Endian

FORMÁTY BITMAPOVÉ GRAFIKY

- Hlavička může obsahovat rovněž informaci o barveném prostoru (ICC) a metadata
- Metadata = popisné a technické údaje
- IPTC struktura
  - Popisek, co je na obrázku
  - Jméno autora, resp. copyright
- EXIF struktura
  - Datum a čas pořízení snímku
  - Typ a nastavení fotoaparátu
    - Doba otevření závěrky, zaostření, …
  - A prakticky vše jako IPTC

FORMÁTY BITMAPOVÉ GRAFIKY: BMP

- Podporován většinou grafických knihoven
- Velká variabilita možností
  - Poslední verze 5: Windows 98+
  - Většina knihoven a aplikací podporuje jen základ
  - Literatura typicky uvádí vlastnosti jen základu
- Základ (nejrozšířenější část)
  - Bezeztrátový formát
  - RGB barevný systém
  - Podpora palety o 1, 16 nebo 256 barev
  - Podpora přímé reprezentace barev 16 bitů na pixel (HighColor), 24 bpp (TrueColor) nebo 32 bpp (TrueColor)

FORMÁTY BITMAPOVÉ GRAFIKY: BMP

- Data pixelů uložena v nekomprimované formě nebo pro obraz s paletou 16 nebo 256 barev komprimována jednoduchou RLE kompresí
  - [**00** 00 00 00 **40** 40 40 40 **80** 80 80 80 80 80 80 80] → [04 00 | 04 40 | 08 80]
  - Mnoho knihoven/aplikací RLE kompresi nepodporuje
- Velké soubory (zejména pro kvalitní grafiku)
- Vhodné např. pro ikony
- Podpora průhlednosti je možná v režimu 32 bpp, ale není standardní

FORMÁTY BITMAPOVÉ GRAFIKY: BMP

- Úplná specifikace
  - Data za hlavičkou mohou být samostatný obrázek ve formátu PNG nebo JPEG
    - Obrázek tudíž může být podroben ztrátě
    - Soubory mohou být malé

FORMÁTY BITMAPOVÉ GRAFIKY: GIF

- Podporován často grafickými knihovnami
- Bezeztrátový formát
- Data komprimovaná LZW slovníkovou metodou (delší sekvence nahrazeny kratšími)
- RGB barevný systém s paletou 2, 4, 8, 16, 32, 64, 128 nebo 256 barev
- Podpora průhlednosti (binární)
- Podpora animací
- Malá velikost
- Vhodné pro internet
  - Pokud nelze použít vhodnější SVG

FORMÁTY BITMAPOVÉ GRAFIKY: PNG

- Menší podpora staršími knihovnami
- Mnoho aplikací podporuje jen částečně
- Bezeztrátový formát
- Data komprimovaná slovníkovou metodou
- RGB(A) barevný systém
  - Podpora palety o 2, 4, 16 nebo 256 barev
  - Podpora přímé reprezentace o 1, 2, 4, 8 nebo 16 bitů na barevnou složku
    - Nejčastěji 8 bitů na složku

FORMÁTY BITMAPOVÉ GRAFIKY: PNG

- Podpora úplné průhlednosti
  - Výhradně 8 bitů na alfu
  - Různé aplikace mají s alfou problémy (např. IE6)
- Vhodné zejména pro obrázky rastrované vektorové grafiky
  - Texty, ostré přechody barev, jednobarevné plochy
  - Kompresní poměr může být klidně 25:1+
- Nevhodné pro zašuměná data
  - Např. reálná fotografie
  - Kompresní poměr typicky 2:1

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

- Ztrátový formát
  - Využívá se "nedokonalosti" lidského zraku
    - Oko "integruje" takže náhlé změny (šum) nepostřehne
    - Jemný šum v jasu nepostřehnutelný
    - Středně velký šum v barevnosti nepostřehnutelný
  - Degradace kvality obrazu
    - Lze řídit poměr mezi kvalitou a velikostí souboru
  - Obvyklé kompresní poměry 5:1 – 20:1

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

1:1

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

17:1

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

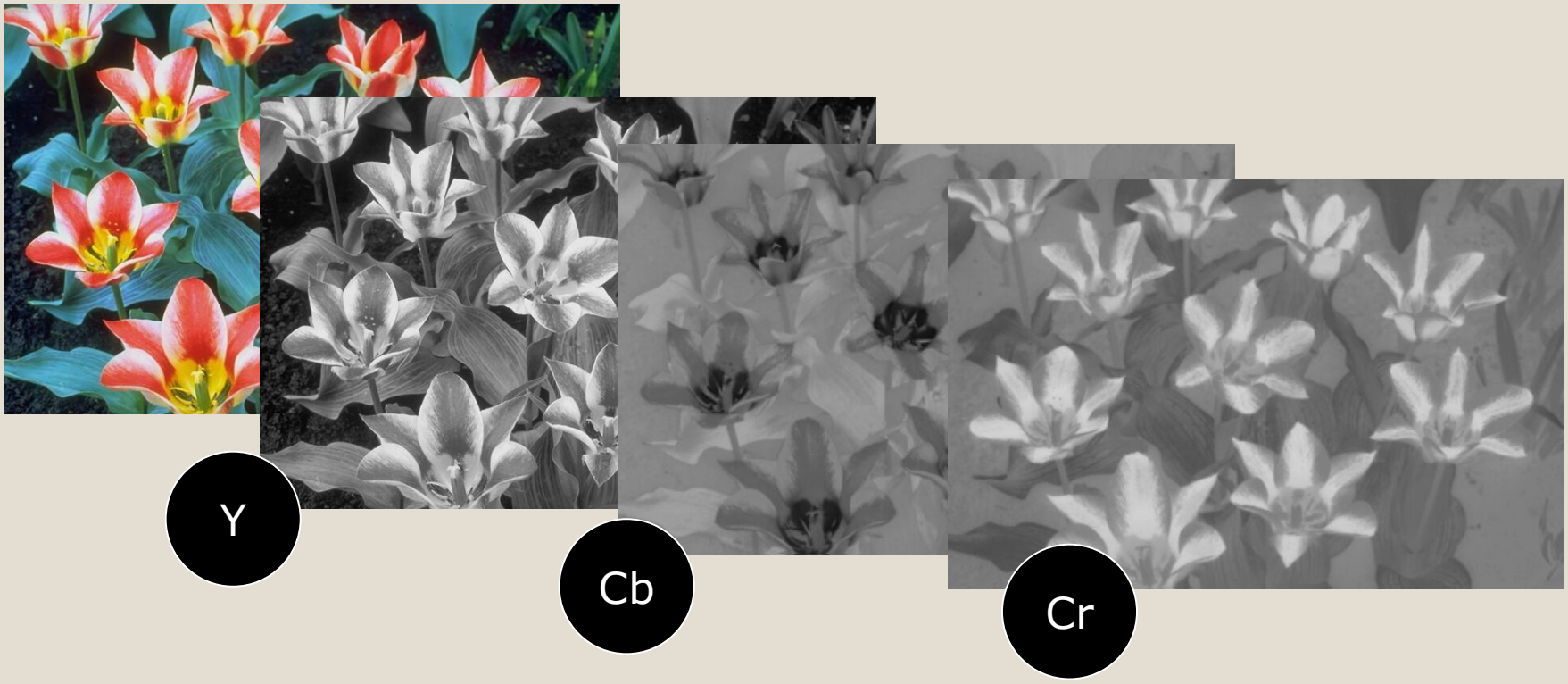
27:1

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

72:1

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

- Barevné podvzorkování:
- RGB na YCbCr



FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

Y

Cb

Cr

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

- Při velké kompresi viditelné artefakty
  - Rozmazané ostré hrany
  - Vizuální rozbití obrazu na bloky 8x8 pixelů
- Barevný systém YCbCr
  - Přímá reprezentace barvy, TrueColor
- Komprese založená na diskrétní kosinové transformaci (DCT) následované kvantizací a RLE + slovníkovou kompresní metodou
- Průhlednost není podporována

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG

- Vhodné pro fotografie určené k archivaci
  - JPEG (s EXIF) obvyklým výstupem z digitálních fotoaparátů
- NEVHODNÉ pro uložení obrazu, který:
  - Budu postupně upravovat
  - Bude rozpoznáván počítačem
    - Problematika Computer vision

FORMÁTY BITMAPOVÉ GRAFIKY: JPEG200

- Obdoba a konkurent formátu JPEG
- Bezeztrátová i ztrátová varianta
- Komprese využívá waveletovou transformaci
  - Vyšší kompresní poměry při stejné kvalitě
    - Pro vizuálně bezeztrátové uložení obrazu je asi 2x lepší
  - Lze dosáhnout vysokých kompresních poměrů bez rozpadu obrazu na malé ostré bloky
  - Podstatně pomalejší na kompresi/dekompresi
- Licenční problematika složitá
- Není příliš rozšířen
  - Malá podpora v aplikacích i knihovnách

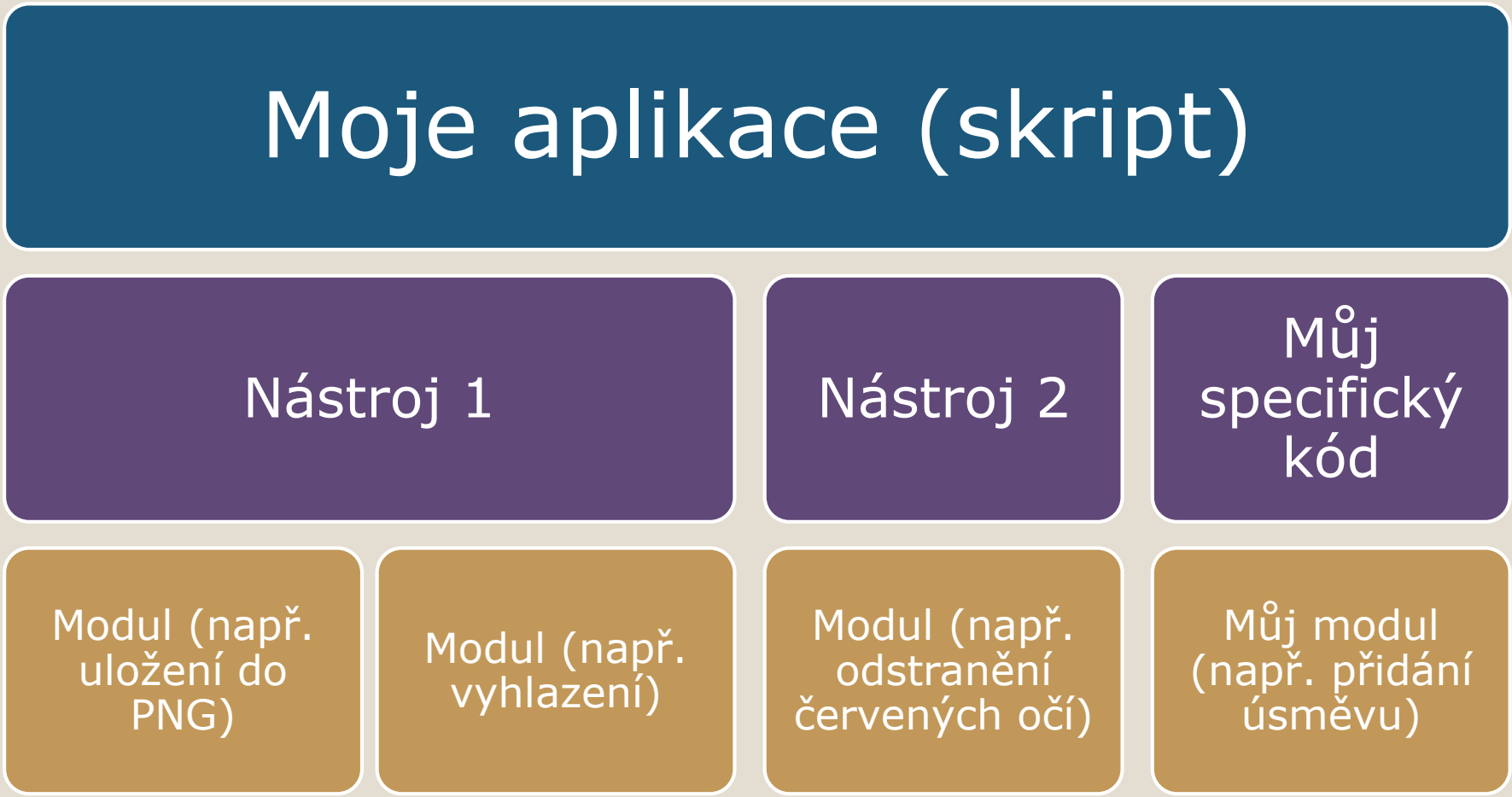
DÁVKOVÉ ZPRACOVÁNÍ BITMAP

- Více obrázků se zpracuje stejným způsobem
- Příklady užití:
  - Webová galerie zobrazuje náhledy, tj. je třeba každý obrázek původní kolekce zmenšit
  - Fotografie z dovolené se nevejdou na CD, je tedy třeba změnit formát, ve kterém jsou uloženy
  - Bezpečnostní kamera automaticky snímá místnost, ostraha se nemusí dívat na všechny obrázky, ale jen na ty, kde došlo k nějakému pohybu
  - Doplnění vodoznaku do obrázků nějaké kolekce

DÁVKOVÉ ZPRACOVÁNÍ BITMAP

- Dávkové zpracování = aplikace volána z příkazové řádky s parametry, co má udělat
- Mnoho aplikací dávkové zpracování buď nepodporuje, nebo jen nejtypičtější úkony
  - Zmenšení, změna formátu obrázku je obvyklá
  - Automatické obarvení k dispozici nebývá
- Řešení:
  - Vzít nástroje, které dohromady podporují, co chceme
  - Napsat skript (BATCH, Python, bash, JavaScript, …), který nástroje postupně nad obrázky zavolá
    - Případně plnohodnotnou aplikaci
  - Doplnit modul, který v nástrojích chybí

DÁVKOVÉ ZPRACOVÁNÍ BITMAP



DÁVKOVÉ ZPRACOVÁNÍ BITMAP

- Obrázky mezi moduly předávány:
  - Prostřednictvím souboru
    - Modul obrázek načte, upraví a uloží
  - Přímo prostřednictvím IPC (např. roury v Linuxu)
    - Nástroje spuštěny současně
    - Upravený obrázek jde z paměti jednoho nástroje do paměti dalšího modulu
    - Lepší efektivita
    - Nemusí být k dispozici

DÁVKOVÉ ZPRACOVÁNÍ BITMAP

- Nástroje pro zpracování bitmapy:
  - netpbm
  - imagemagick
  - **graphicsmagick**
  - …

```
try {
  Runtime.getRuntime().exec(
        "gm mogrify -format jpeg *.png");
} catch (IOException e) {
  //obsluha neúspěšného volání
}
```

KONEC

- Příště: Základní vizualizace informace