ZÁKLADNÍ BITMAPOVÁ GRAFIKA

Vnímání jasu a barev

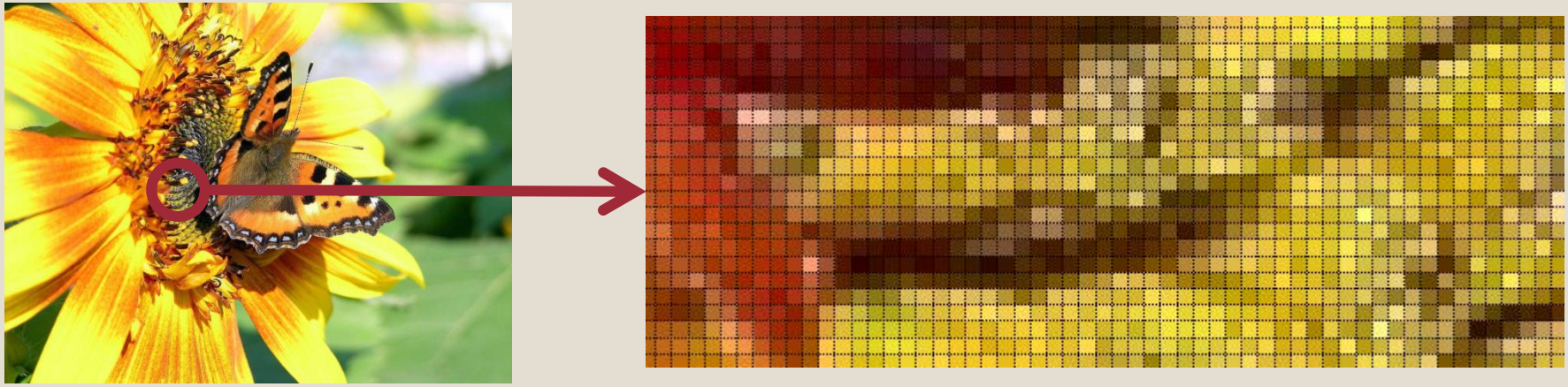
Barevné modely

Užívání rastrových obrázků

Manipulace s obrazem

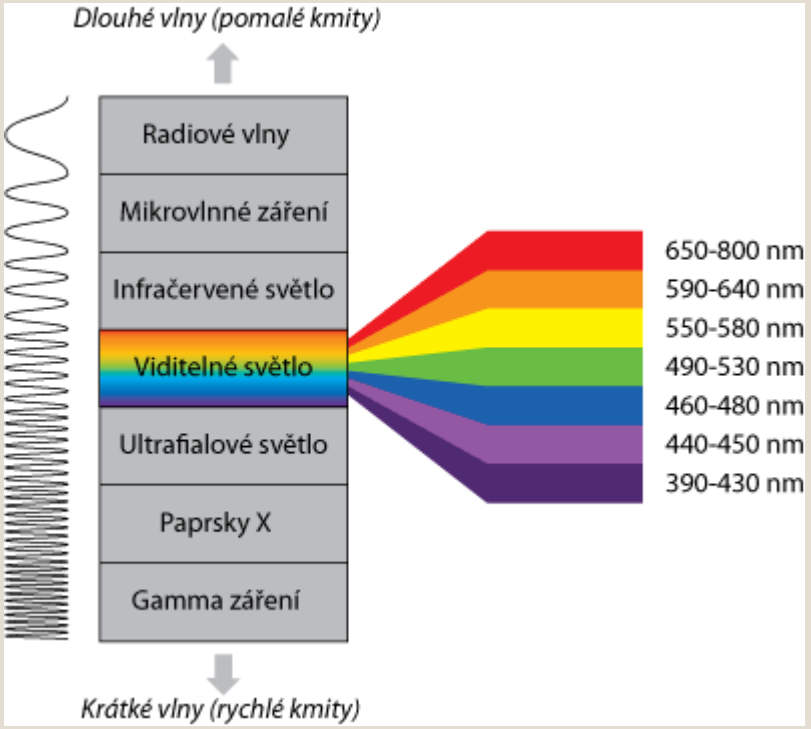
BITMAPOVÁ GRAFIKA

- Bitmapová (rastrová grafika)
  - Obrázek složen z 2D matice diskrétních pixelů
    - Pixel = Picture Element = obrazový element
  - Pixel nese informaci o barevné hodnotě
    - Může se měnit v čase => video
    - Informace kódována na fixní počet bitů



SVĚTLO

- Viditelné světlo = oblast, na kterou je lidské oko citlivé
- Monochromatické světlo
  - Fotony jen jednoho typu → jen jedna vlnová délka
    - Např. 540 nm
  - Zdroj nelze prakticky vyrobit
- Achromatické světlo
  - Fotony různého typu, ale ve světle stejný počet od každého
  - Bílé světlo = 390 – 800 nm
- Polychromatické světlo
  - Fotony různých typů i počtů

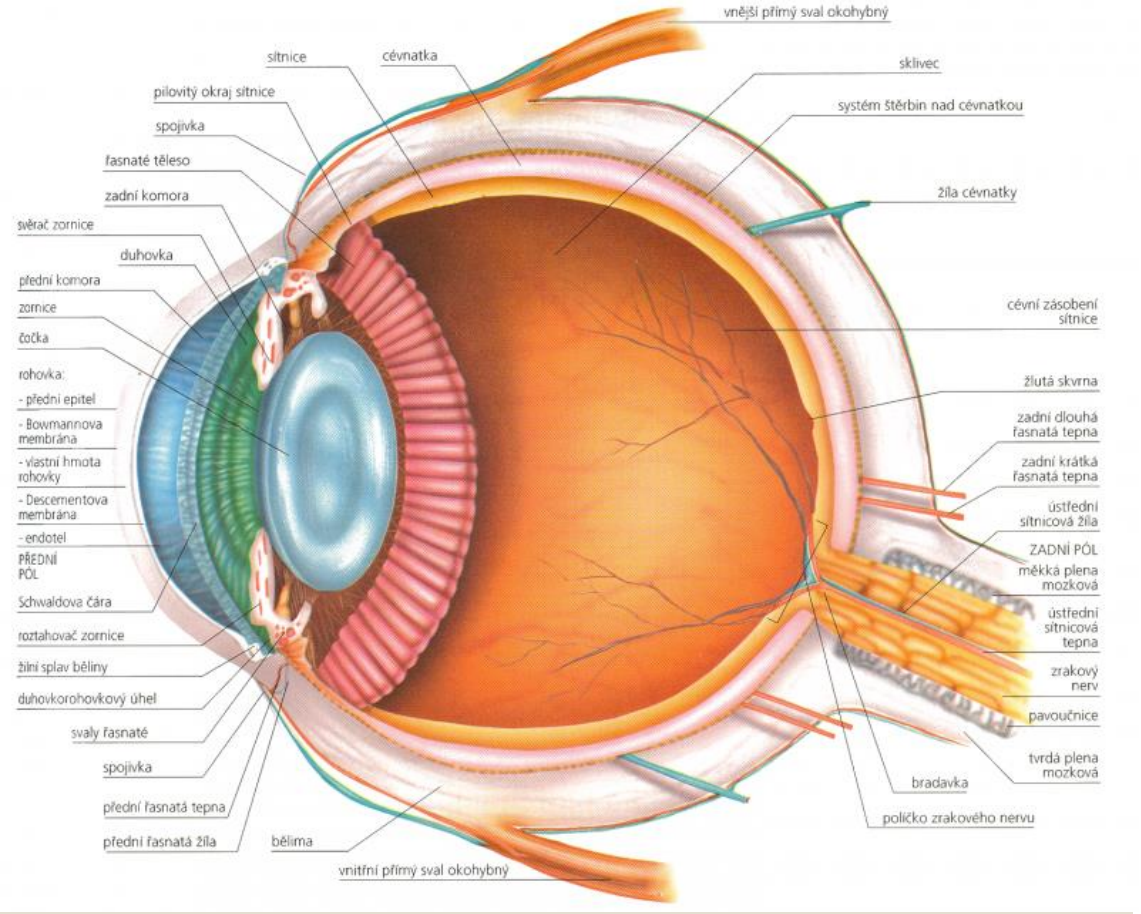


SVĚTLO

- Fotony dopadají do oka (resp. na senzory zařízení)
  - Mohou pocházet buď přímo ze světelných zdrojů nebo jsou odraženy od objektů
  - Dochází k vizuálnímu vjemu

LIDSKÉ OKO

- Sítnice (retina)
  - V zadní části oka
  - Tvořena buňkami citlivých na světlo
  - Propojena zrakovým nervem přímo s mozkem
  - Tři typy buněk:
    - Tyčinky
    - Čípky
    - ipRGC

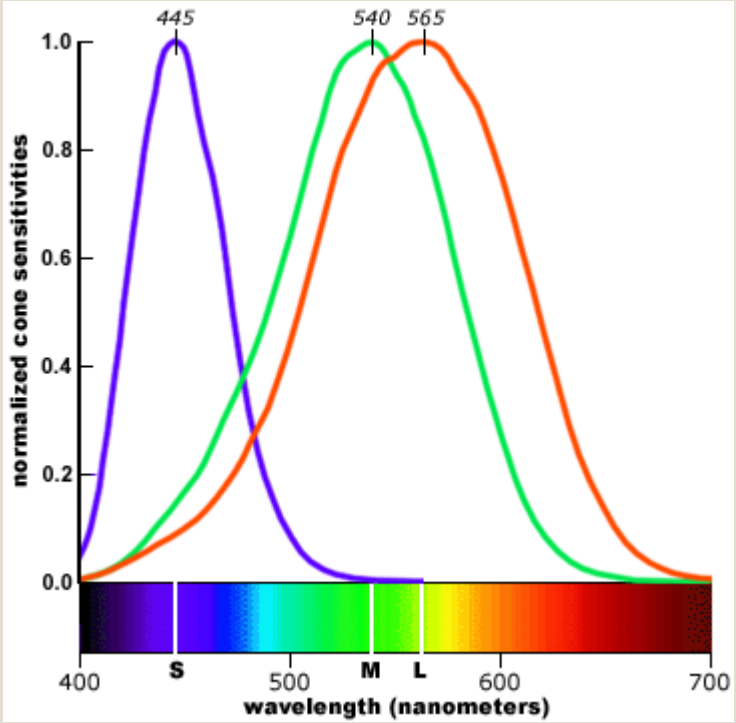


LIDSKÉ OKO

- Tyčinky
  - Extrémní citlivost
    - Podráždění již při 1 fotonu
    - Citlivost klesá s množstvím fotonů
  - Nejcitlivější na světlo modrozelené (cca 505 nm)
    - Přesná hodnota vlnové délky individuální
      - Různé hodnoty v literatuře
  - Dlouhá setrvačnost
  - "Noční vidění"

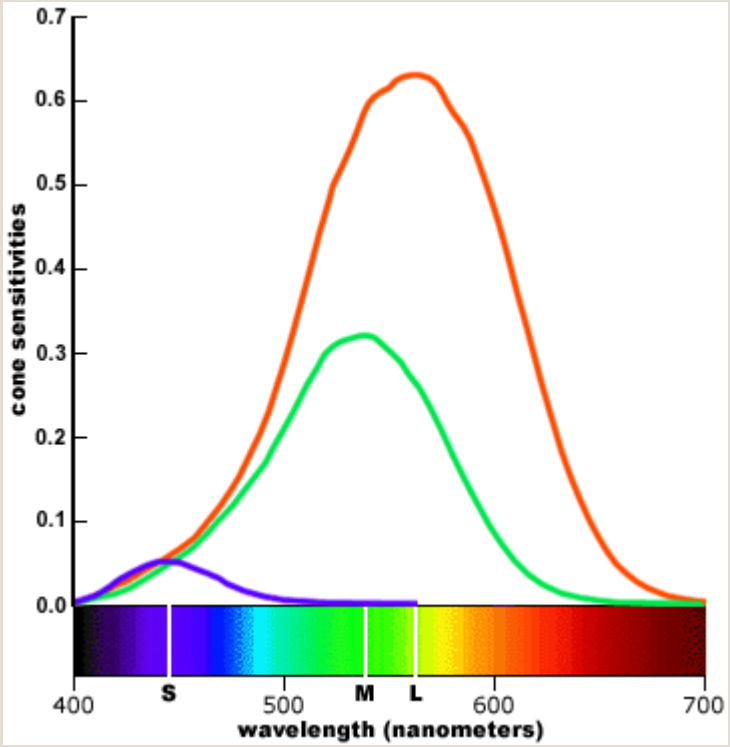
LIDSKÉ OKO

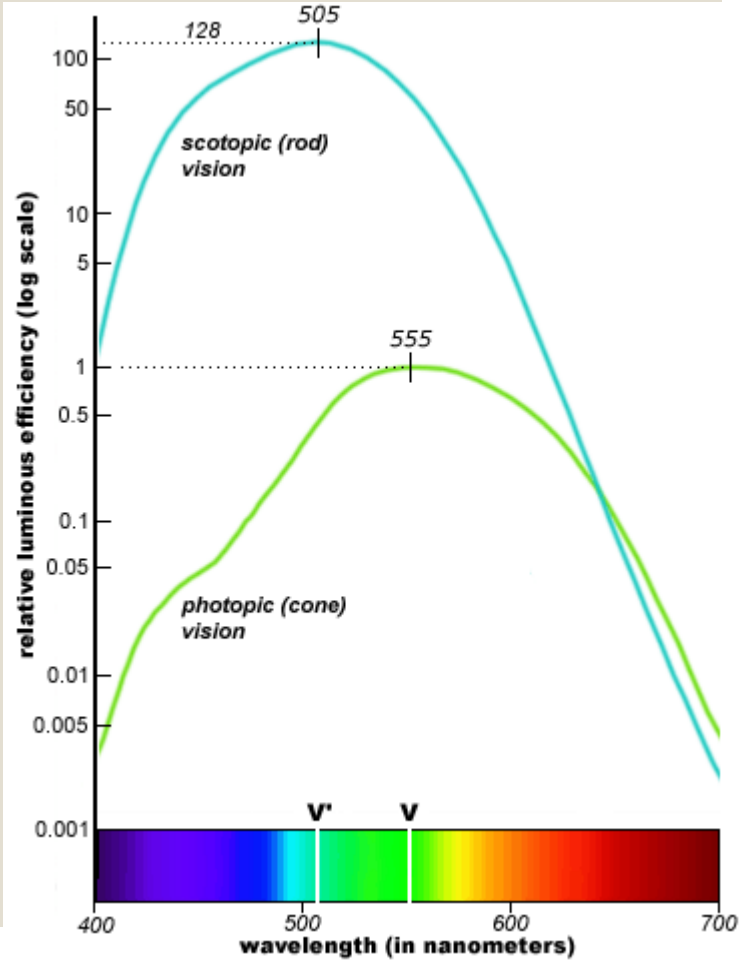
- Čípky
  - Malá citlivost
    - Podráždění až při stovkách fotonů
    - "Denní vidění"
  - Relativně krátká setrvačnost
  - Tři typy čípků: L, S, M
    - Každý citlivý na jinou vlnovou délku
    - Rozmístěny nerovnoměrně
    - Počty odlišné



LIDSKÉ OKO

- Normalizovaná citlivost čípků a tyčinek



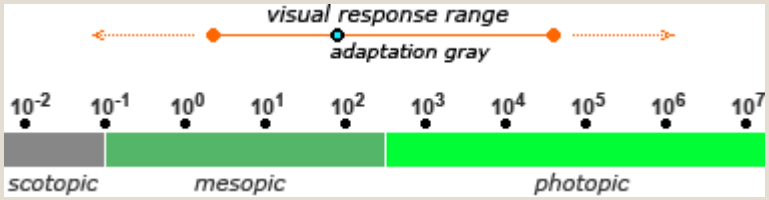


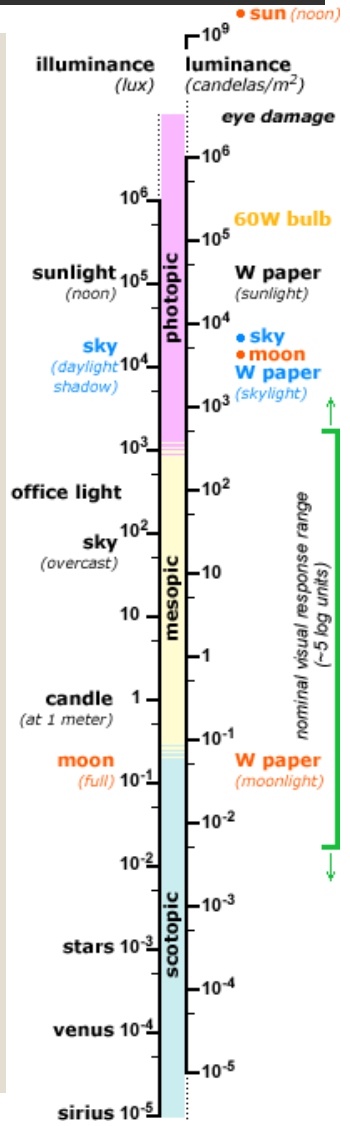
JAS A JEHO VNÍMÁNÍ

- Jas = skalární veličina
  - Určuje, za se nám předmět jeví tmavší nebo světlejší
  - Dán zejména počtem fotonů
  - Ovlivněn vlnovou délkou
    - Modrá 100W žárovka se nám jeví tmavší než bílá
    - Žlutá se jeví světlejší než modrá

JAS A JEHO VNÍMÁNÍ

- Lidské oko dokáže vnímat hodnoty jasu od tisícin po tisíce $Cd \cdot m^{-2}$
  - Celkem asi 9-14 řádů
  - V daném prostředí asi 5 řádů
    - Tj. v jeden okamžik naráz
  - Adaptace na prostředí





JAS A JEHO VNÍMÁNÍ

Řád	Hodnota	Objekt
10^{-6}	1 µcd/m^2	Práh viditelnosti
10^{-4} – 10^{-3}	400 µcd/m^2 – 1 mcd/m^2	Noční obloha
10^{0}	0.5 – 2 cd/m^2	Osvícená silnice
10^{1}	25 cd/m^2	Scéna při západu / východu slunce
10^{2}	700 cd/m^2	Scéna ve dne, kdy je zataženo

JAS A JEHO VNÍMÁNÍ

Řád	Hodnota	Objekt
10^3	2 kcd/m^2 5 kcd/m^2	Obloha s mraky Scéna za plného slunečního svitu
10^4	5-15 kcd/m^2 75 kcd/m^2	Zářivka Sodíková výbojka
10^5	130 kcd/m^2	Mléčná 60W klasická žárovka
10^6	7 Mcd/m^2	Průhledná žárovka
10^9	1.6 Gcd/m^2	Slunce v poledne

JAS A JEHO VNÍMÁNÍ

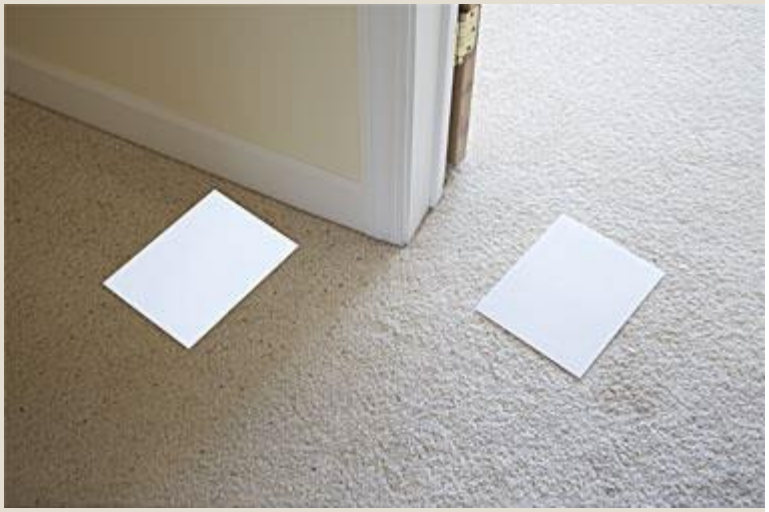
- Současná zařízení (LCD, tiskárny, …) však produkují hodnoty jasu jen velmi omezené
  - 2-3 řády v oblasti pokojového osvětlení ("indoor")
  - Starší LCD monitor 0.5 až 250-300 $Cd \cdot m^{-2}$
  - LCD TV až 300-600 $Cd \cdot m^{-2}$
  - Plasma až 1000 $Cd \cdot m^{-2}$
  - Výtisk z tiskárny cca 1 – 50 $Cd \cdot m^{-2}$
    - Při pokojovém osvětlení
- Kontrast = poměr mezi nejsvětlejší a nejtmavší hodnotou, kterou zařízení zvládne

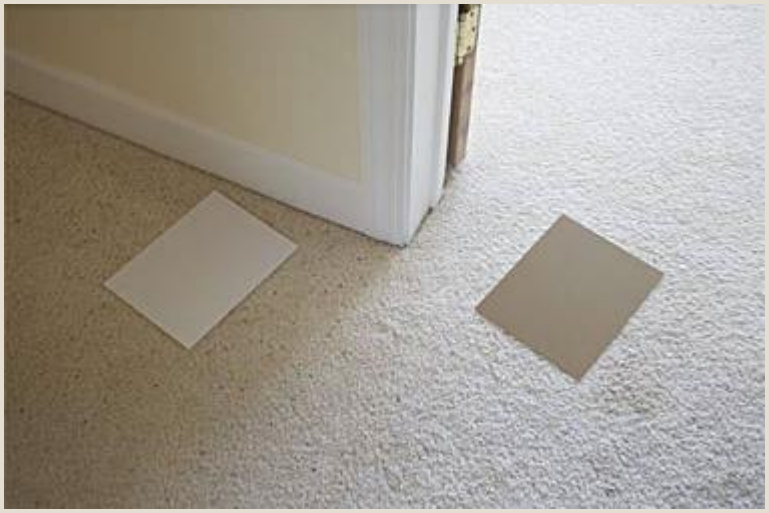
JAS A JEHO VNÍMÁNÍ

- Důsledek:
  - "Černou" a "bílou" vnímáme na různých zařízeních nebo v různých prostředí různě
- Jas (I) není vnímán lineárně
  - Vnímaný jas $\mathrm{L} = I^{0.4}$
- Důsledek:
  - Dvě tmavé plošky s rozdílem jasu 9 $Cd \cdot m^{-2}$ vnímány v pokojovém světle jako zcela odlišné, zatímco při denním světle jako téměř stejné

JAS A JEHO VNÍMÁNÍ

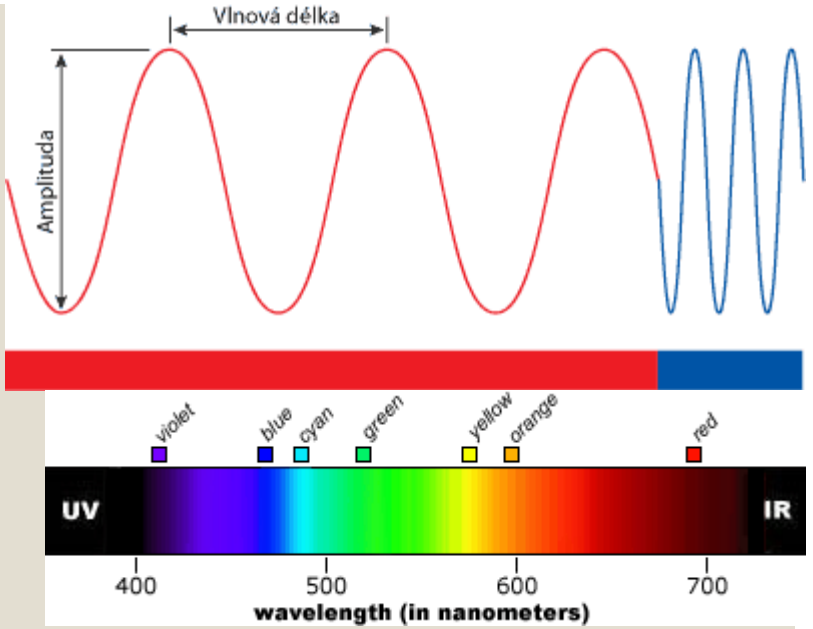
- Jas vnímán v kontextu okolí





BARVA A JEJÍ VNÍMÁNÍ

- Fotony různé vlnové délky vnímány jako různě barevné
- Výsledná vnímaná barva složena mozkem podle toho, jak podráždění jednotlivých typů čípků



BARVA A JEJÍ VNÍMÁNÍ

- Barva vnímána v závislosti na kontextu

BARVA A JEJÍ JAS

- Relativní podráždění čípků: hodnoty L, M, S
- Vnímaný jas B = L+M+S
- Vnímaná barevnost C = (L/B, M/B, S/B)
- Příklad

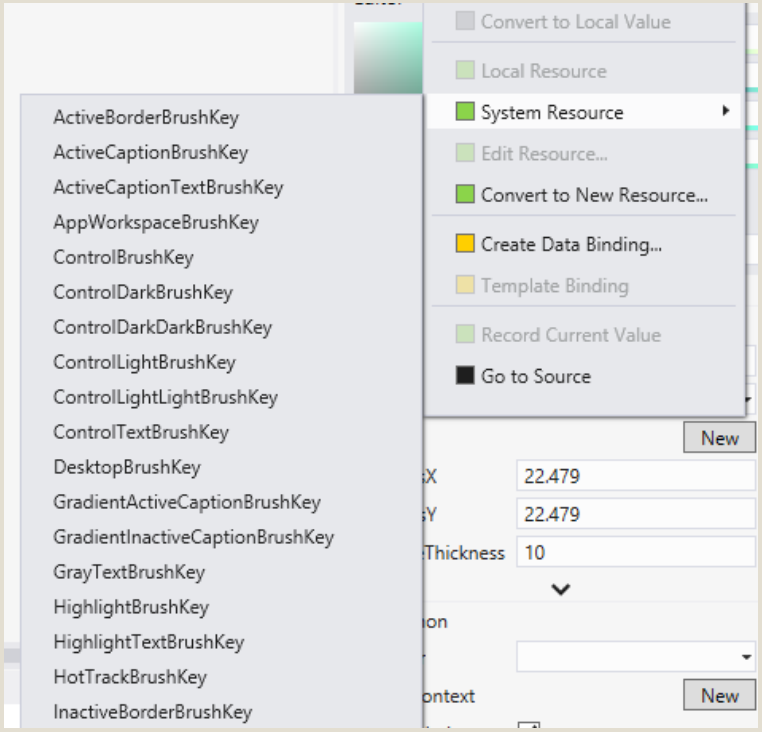
- Důsledek: lidské oko je podstatně citlivější na jas než na barevnost

VOLBA BAREV V APLIKACI

- Barevné prvky aplikace musí ladit
  - Volba teplých vs. studených barev
  - Rozlišitelnost barev
    - Barvoslepí (cca 15%)
  - Menší únava očí

VOLBA BAREV V APLIKACI

- Barvy nesmí být přepálené
  - Zejména u aplikací, které jsou užívány několik hodin denně
  - Záleží rovněž na okolním osvětlení
- Obecné pravidlo:
  - Neexperimentovat
  - Vhodné užívat tlumené barvy
    - Skoro neutrální
- Menší aplikace by neměly mít každá jiný vzhled
  - Vhodné užívat systémové barvy

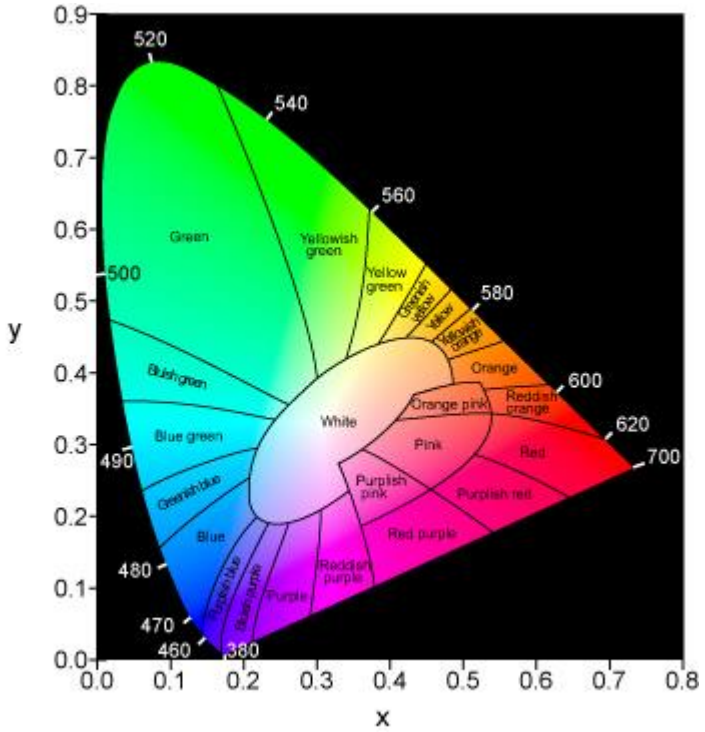


POČÍTAČOVÁ GRAFIKA

- Cílem modelování reálného světa
  - Fotorealistické vs. nefotorealistické
- Úkoly:
  - Reprezentovat barvu (jas a barevnost) ve vhodné digitální podobě
    - Podoba se liší dle požadavků na věrnost reprezentace
  - Zobrazit model (obrázek) na výstupním zařízení
    - LCD obrazovka, plasma, tiskárna, …
  - Zachytit reálný svět vstupním zařízením
    - Digitální fotoaparát, kamera, skener, …
  - Snaha minimalizovat zkreslení dané konstrukčním omezením vstupních a výstupních zařízení

REPREZENTACE BARVY

- CIE xy
  - Standardizovaný matematický model barevného prostoru
  - Libovolný barevný odstín popsatelný dvojicí reálných čísel x, y
    - Vyjadřuje směs různých vlnových délek
  - Používá se pro definování alternativních (praktičtějších) reprezentací barvy



ZOBRAZOVACÍ ZAŘÍZENÍ

- Obrazovka složena ze zobrazovacích elementů
  - Vyzařují světlo
- Ideální obrazovka
  - 1 element může vysílat (emitovat) fotony o libovolné vlnové délce (dle požadavku)
    - 1 pixel zobrazitelný jediným zobrazovacím elementem
  - Rychlost emitování fotonů lze velmi dobře řídit
    - Libovolný jasový rozsah
  - Fyzikálně obtížně realizovatelné
    - Laditelný barvivový laser

ZOBRAZOVACÍ ZAŘÍZENÍ

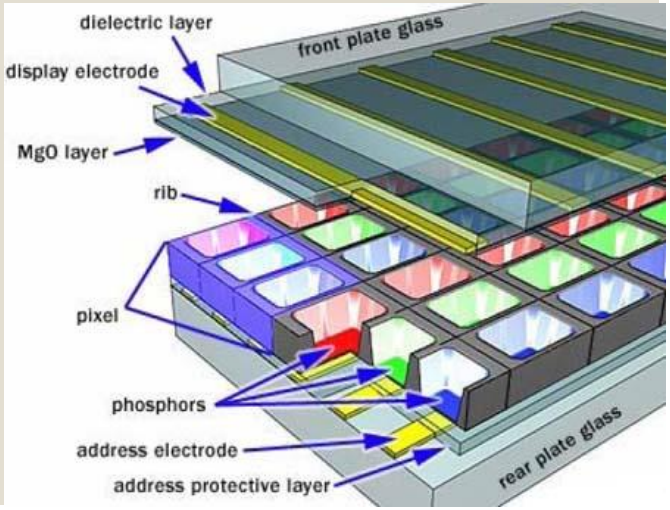
- Fyzikálně realizovatelná obrazovka
  - Jeden pixel zobrazován prostřednictvím více zobrazovacích elementů
  - Element emituje fotony o jedné vlnové délce
  - Je třeba "nekonečně mnoho" zobrazovacích elementů
  - Konstrukčně nerealizovatelné

ZOBRAZOVACÍ ZAŘÍZENÍ

- Reálná obrazovka současnosti
  - Pixel zobrazován prostřednictvím několika málo zobrazovacích elementů
  - Realističnost zobrazení závisí na:
    - volbě typů zobrazovacích elementů, tj. jaké vlnové délky mají fotony, které zobrazovací elementy emitují
    - rozložení zobrazovacích elementů různého typu
    - schopnosti zobrazovacích elementů emitovat různě velké množství fotonů za jednotku času
      - Jasový rozsah

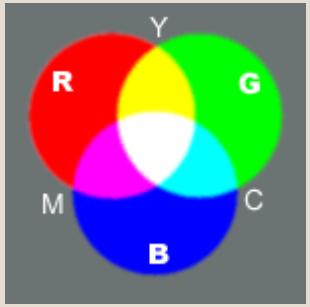
ZOBRAZOVACÍ ZAŘÍZENÍ

- Experimentálně zjištěno, že pro napodobení většiny reálných barev postačí červený (R), zelený (G) a modrý (B) zobrazovací element
- Počet fotonů, který má zobrazovací element emitovat za jednotku času, tj. jeho aktivita určena relativně na intervalu 0.0 – 1.0



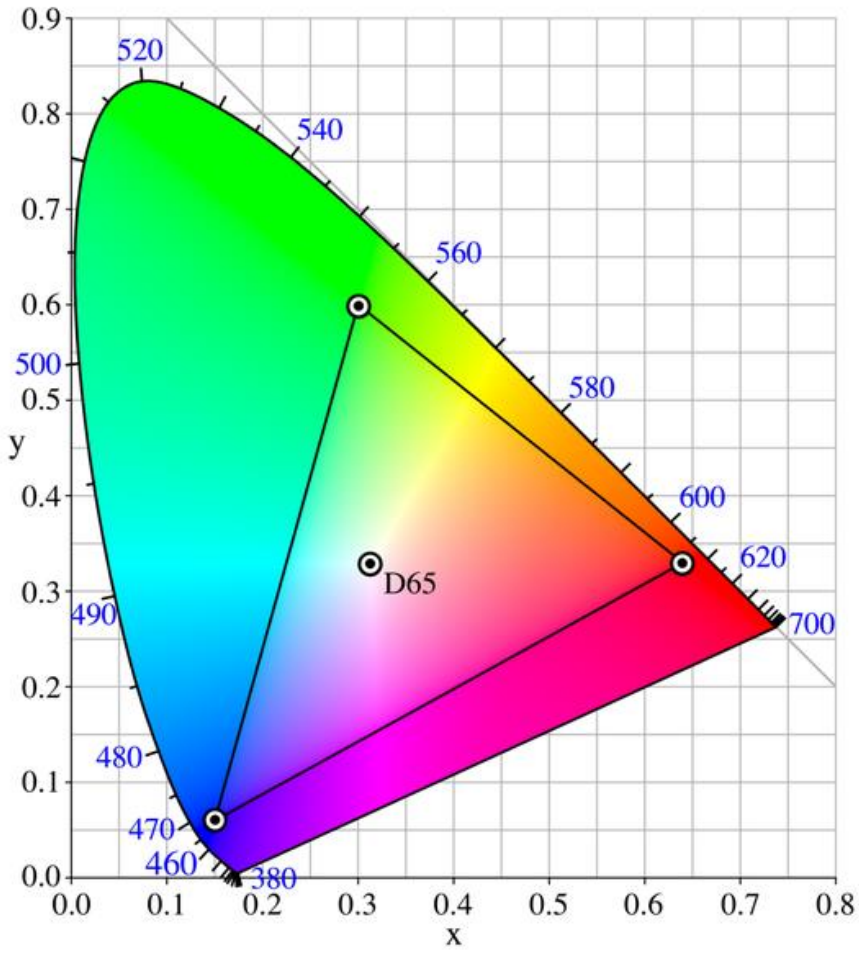
ZOBRAZOVACÍ ZAŘÍZENÍ

- Barvu objektu (pixelu) lze tedy definovat jednoduše trojicí čísel (r, g, b)
  - (1, 1, 1) = 1*(R) + 1*(G) + 1*(B) = bílá
  - (0, 0, 0) = černá
  - Obecná barva = r*(R) + g*(G) + b*(B)
  - Jedná se o tzv. RGB barevný prostor
    - Masivně rozšířená reprezentace barvy



ZOBRAZOVACÍ ZAŘÍZENÍ

- Problém:
  - R, G, B nejsou mono-chromatická světla
  - Převažující vlnová délka se liší dle výrobce
  - "Čistota" základních barev bývá odlišná



ZOBRAZOVACÍ ZAŘÍZENÍ

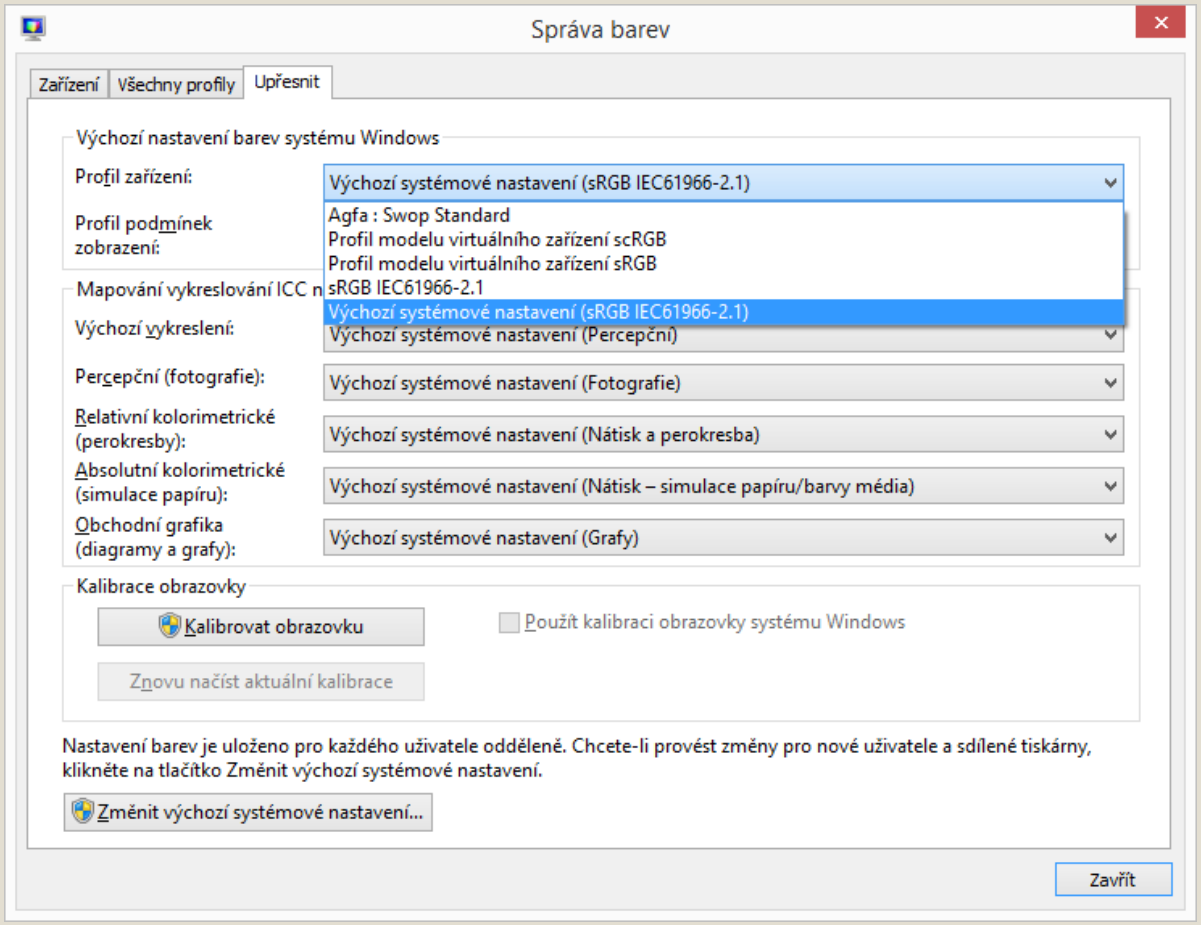
- Nepříjemné důsledky:
  - Stejná barva (r, g, b) vypadá na různých zobrazovacích zařízeních zcela odlišně
    - Např. zelený (0, 1, 0) text na starém monitoru je relativně OK, ale na novém může "pálit" do očí
- Částečným řešením je zavedení standardizovaného teoretického barevného systému RGB_{teorie}
  - Základní barvy přesně nadefinovány
    - Nutně nemusí být v praxi realizovatelné

ZOBRAZOVACÍ ZAŘÍZENÍ

- Barva vyjádřena jako $(r, g, b)_{teorie}$
- Pro zobrazení $(r, g, b)_{teorie}$ aplikace provede:
  - Transformaci $(r, g, b)_{teorie}$ na $(r, g, b)_{displej}$
  - Zašle zobrazovacímu zařízení $(r, g, b)_{displej}$
- Transformace musí zajistit, aby zobrazená barva odpovídala požadované
  - Je nutné znát charakteristiku systému RGB_{teorie}
    - Popsáno v tzv. ICC (barevném) profilu systému
  - Je nutné znát charakteristiku zobrazovacího zařízení
    - Popsáno v ICC profilu zařízení

ZOBRAZOVACÍ ZAŘÍZENÍ

- Správa barev
  - Windows 8.1



ZOBRAZOVACÍ ZAŘÍZENÍ

- Transformace prováděna typicky automaticky grafickou knihovnou
  - Postačí specifikovat ICC profil barevného systému, ve kterém barvu reprezentujeme

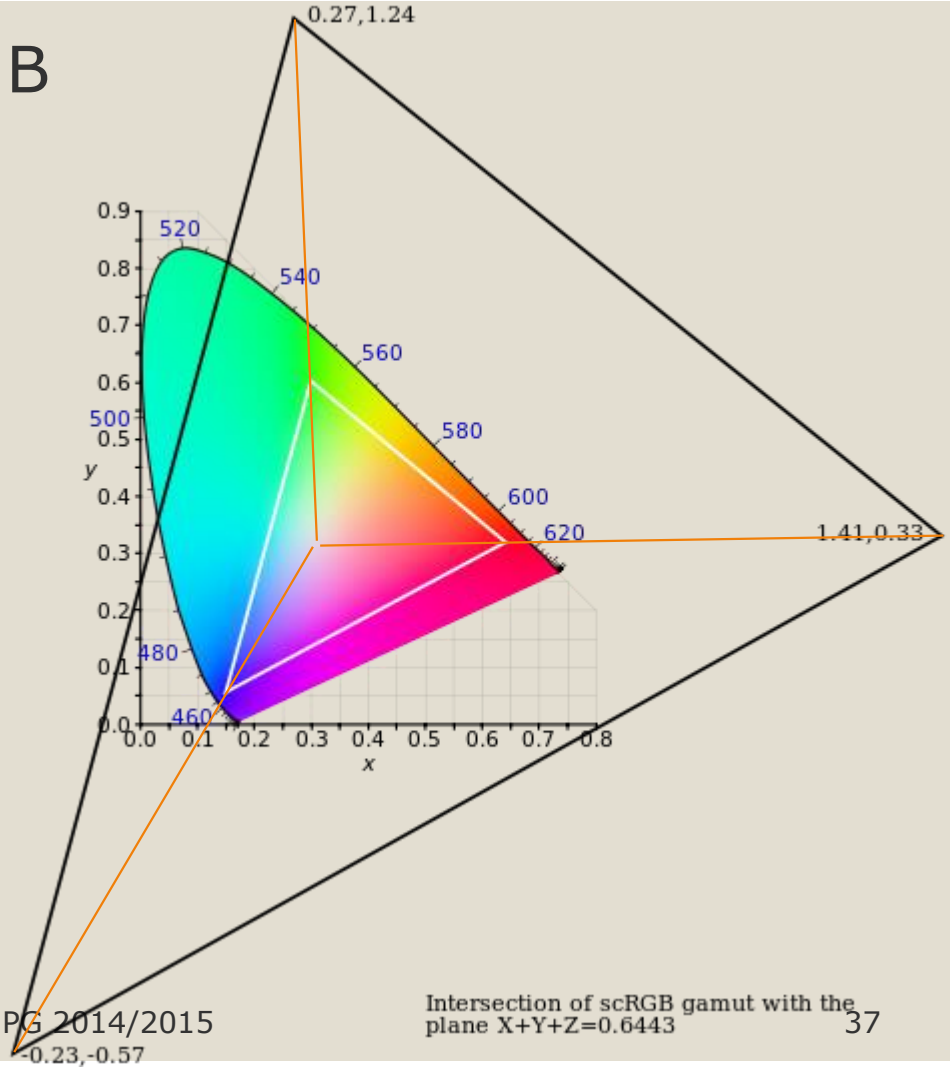
```
ICC_Profile ip = ICC_Profile.getInstance( ColorSpace.CS_sRGB );

ICC_ColorSpace ics = new ICC_ColorSpace( ip );

ColorConvertOp cco = new ColorConvertOp( ics, null );
BufferedImage result = cco.filter( sourceImage, null );
```

ZOBRAZOVACÍ ZAŘÍZENÍ

- Problémy:
  - Barva popsatelná v teoretickém RGB systému nemusí být na daném zařízení reprodukovatelná
  - Většina aplikací se ICC profilem nezabývá
    - Dochází ke zkreslení barvy
      - Tím větší, čím RGB_{teorie} se liší od $RGB_{displej}$
    - Předpokládá se, že RGB_{teorie} = sRGB
- sRGB = standardizovaný barevný systém
  - Základní barvy odpovídají "průměrným" základním barvám v zobrazovacích zařízeních
    - sRGB je podobný CRT $RGB_{displej}$
    - Přílišná omezenost na moderních zařízeních

ZOBRAZOVACÍ ZAŘÍZENÍ

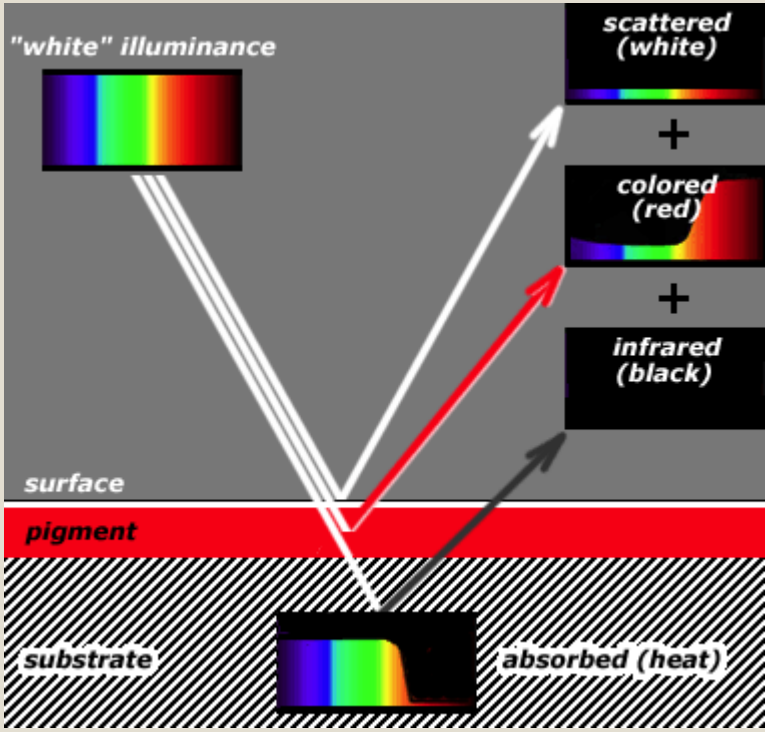
- Barevný systém scRGB
  - Microsoft + HP
  - Definice vlnových délek a bílého bodu totožná s sRGB
  - Hodnoty r, g, b nejsou omezeny na rozsah 0.0 - 1.0, ale na rozsah cca -0.5 – 7.5
  - Bohatší barvy
  - Windows Vista+
  - Podpora ve WPF



Intersection of scRGB gamut with the plane X+Y+Z=0.6443

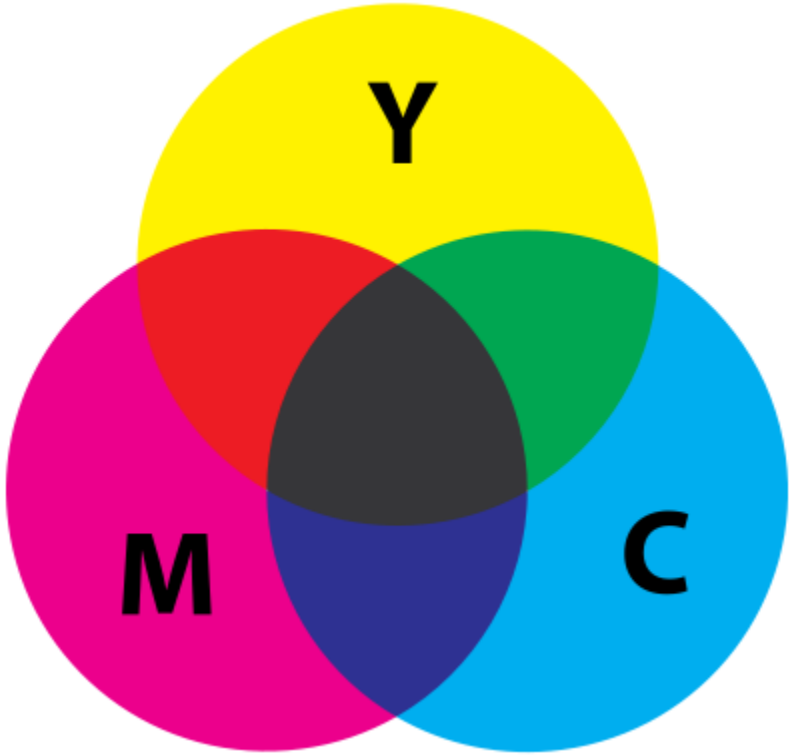
TISKÁRNY

- Zcela odlišný přístup
  - Papír je bílý => potřebujeme modifikovat množství odraženého světla
  - Potřebujeme doplněk k RGB
- Barviva CMY
  - C (cyan, azurová)
    - Pohltí zcela R
  - M (magenta, purpurová)
    - Pohltí zcela G
  - Y (yellow, žlutá)
    - Pohltí zcela B



TISKÁRNY

- Problémy:
  - CMY barviva nejsou čistá
  - Smíšení nedokonalé
  - Vjem bude záviset na osvětlení
- Nepříjemný důsledek:
  - Nelze namíchat černou
    - 100/100/100 = hnědočerná
- Řešení: přidání černé (K)



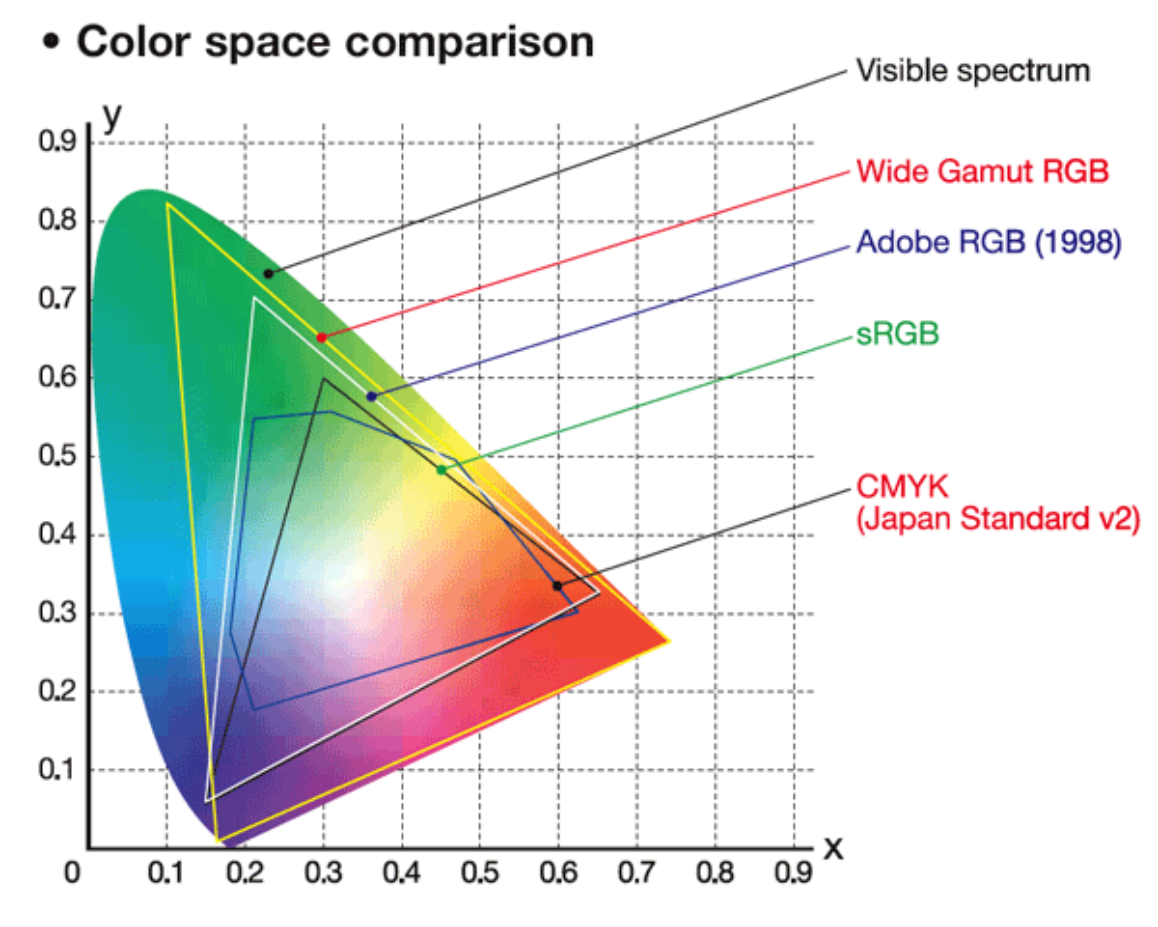
TISKÁRNY

- Barvu objektu (pixelu) lze tedy reprezentovat trojicí (resp. čtveřicí) čísel (c, m, y, k)
  - Jedná se o tzv. CMY(K) barevný prostor
  - (0, 0, 0, 1) = černá
  - (0, 0, 0, 0) = bílá

REPREZENTACE BAREV

- Barva v počítači nemůže být reprezentována jedním univerzálním způsobem
- Barevný systém = způsob reprezentace barvy
  - Má svůj gamut = prostor barev věrně reprezentovatelných barevným systémem
- Gamuty různých barevných systémů mohou být velmi odlišné
  - Např. sRGB a standardizovaný CMY(K)
- Důsledek:"hezký obrázek" na monitoru může po vytištění vypadat příšerně

REPREZENTACE BAREV



REPREZENTACE BAREV

- Barvu reprezentovanou v jednom barevném systému lze převést do jiného
  - ICC profily
  - Může dojít ke ztrátě
- Často používané barevné systémy:
  - RGB, zejména pak sRGB (resp. scRGB)
  - CMY(K), zejména pak Fogra CMYK
  - HSV resp. HSL
  - YCbCr, YCoCg, CIE Lab

REPREZENTACE BAREV

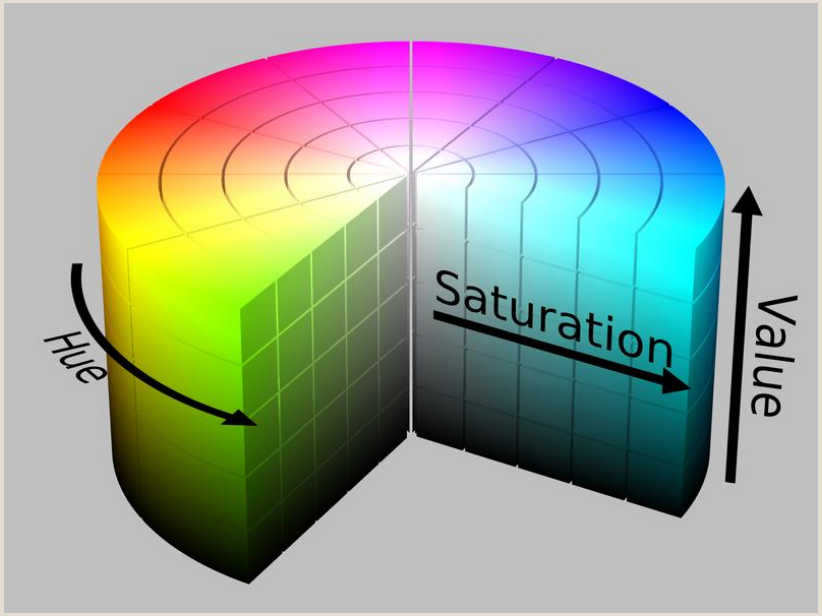
- Barevný systém RGB
  - Používán zobrazovacími i vstupními zařízeními
  - Nativně užíván v bitmapových obrázcích
    - V nekomprimované podobě
  - Hojně rozšířený
    - zejména sRGB
  - Problém: specifikace barvy není intuitivní

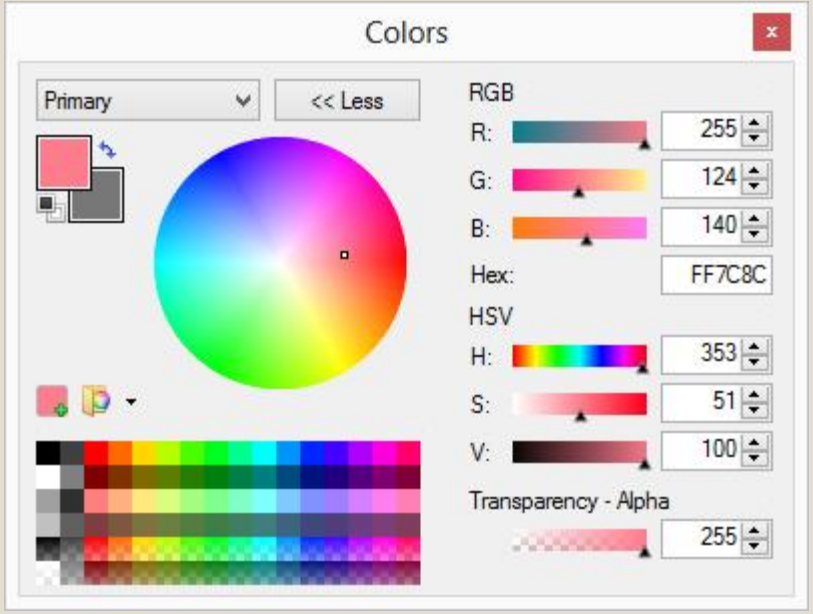
REPREZENTACE BAREV

- Barevný systém CMY(K)
  - Používá se pro tiskové účely
  - DTP aplikace
- Barevný systém HSV
  - Barva reprezentována trojicí Hue, Saturation a Value
  - Hue = odstín barvy (0 – 360 stupňů)
  - Saturation = sytost (0 - 1)
  - Value = jas (0 - 1)
  - Jednodušší na zadání
  - Problematické pro analýzu barvy
    - H zatíženo velkou chybou pro skoro neutrální barvy

REPREZENTACE BAREV

- Snadné řešení přechodů (gradient)
  - Např. výška terénu kódována barvou





REPREZENTACE BAREV

- Grafické knihovny obvykle podporují specifikaci barvy jak zadáním RGB složek, tak zadání HSV složek
- Barevné systémy YCbCr, YCoCg, CIE Lab
  - Jasová složka separována (Y, L)
    - Lze využít pro převod obrázku na šedotónový
  - Barevnost ve zbývajících složkách
  - Systémy se liší způsobem, jak stanoveno
  - Výhodné pro kompresní účely
    - Využívá citlivosti oka na jas

DIGITALIZACE BARVY

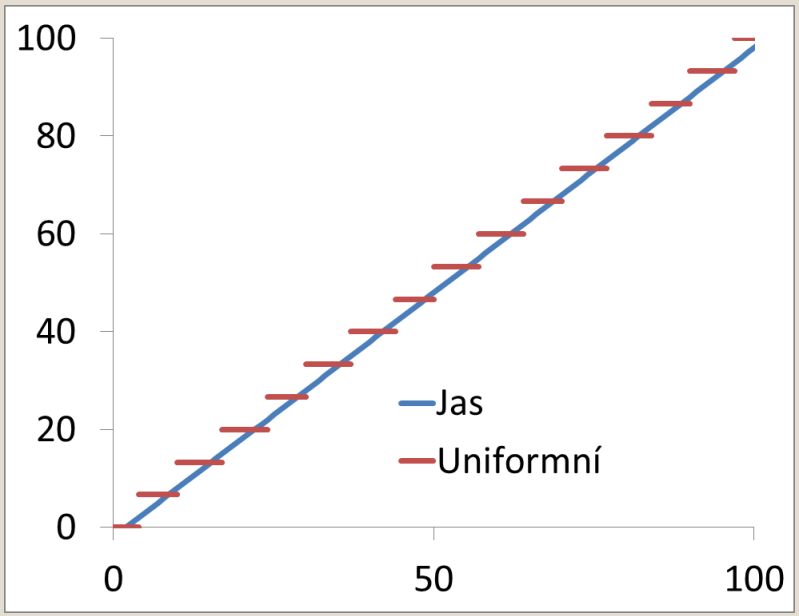
- Vstupní zařízení obsahují senzory fotocitlivé na nějaké „základní“ barvy
  - Nejčastěji opět R, G, B
  - Aktivita senzoru úměrná počtu fotonů
    - Hodnoty jasu mimo jasový rozsah zařízení zkreslené

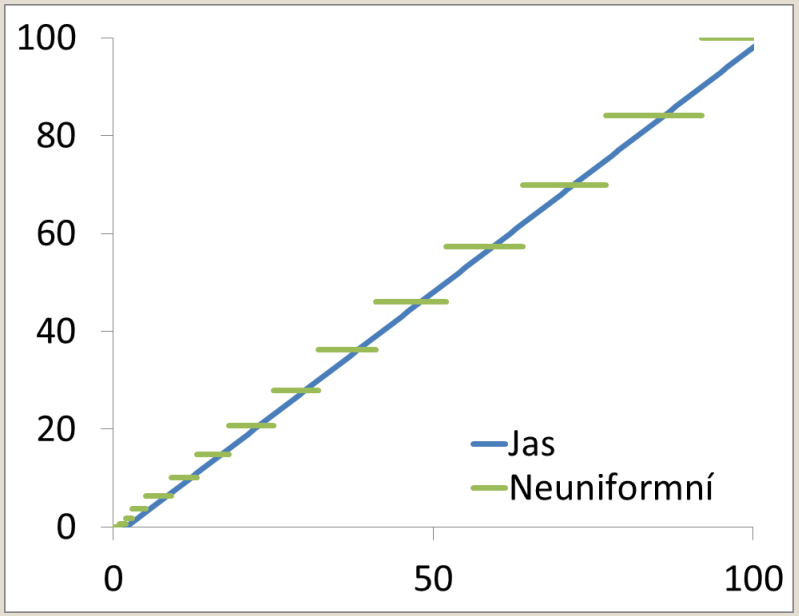
DIGITALIZACE BARVY

- Diskretizace jasového rozsahu:
  - Jedná se o proces tzv. kvantizace
  - Jasový rozsah rozdělen na *m* reprezentativních úrovní
  - Skutečná jasová hodnota zaokrouhlena na hodnotu nejbližší úrovně
- Uniformní diskretizace
  - Nejjednodušší
  - Nebere v úvahu lidské vnímání

DIGITALIZACE JASOVÉ HODNOTY

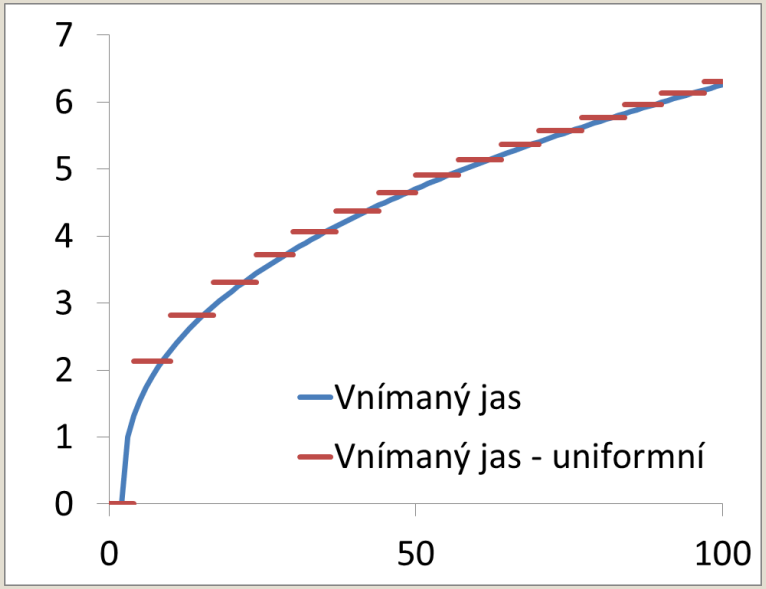
- Neuniformní diskretizace
  - Jasová hodnota podrobena gama korekci $\mathrm{L} = I^{0.4}$
  - Ukládám L namísto I
    - Kvantifikace uniformní v jasovém rozsahu L

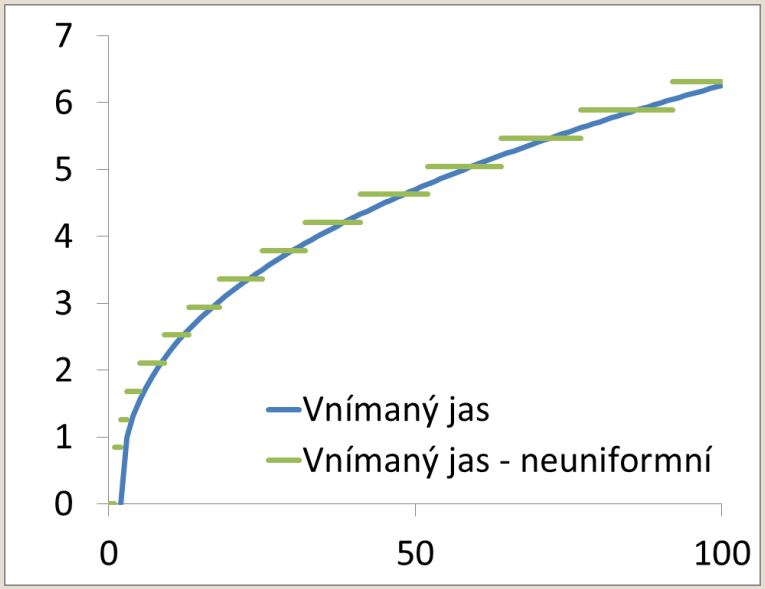




DIGITALIZACE JASOVÉ HODNOTY

- Neuniformní diskretizace
  - "Skutečnou" hodnotu lze určit z kódované jako $I = L^{2.5}$
  - Prováděno automaticky většinou běžných zařízení
  - Vede na menší chybu reprezentace obrazu





UKLÁDÁNÍ BARVY

- Hodnoty barevných komponent nutno uložit v nějakém podporovaném formátu
  - Omezená přesnost
  - Dochází ke ztrátě
- Desetinné datové typy
  - Fixní desetinná čárka
    - Aplikačně závislé
  - Plovoucí desetinná čárka
    - Výhradně IEEE float
    - Hodnoty obvykle v rozsahu 0.0 – 1.0 nebo alternativně 0.0 – 255.0 (maximální aktivace)
    - Např. grafické subsystémy OpenGL, DirectX, …

UKLÁDÁNÍ BARVY

- Celočíselné datové typy
  - Použité ve většině případů
  - Omezenější přesnost
    - Úspornější
  - Fixní počet bitů
    - Určuje číselný rozsah
  - Hodnota barevné komponenta specifikována absolutně v rámci možného číselného rozsahu
    - Maximum = požadavek na maximální intenzitu

UKLÁDÁNÍ BARVY

- 8 bitů (byte)
  - Rozsah 0-255
    - 256 různých hodnot
  - Např. v případě RGB umožňuje realizovat 256*256*256 (= 16.7 miliónu) různých barev
    - Jedná se o tzv. True-Color
      - 24 bitů na pixel (resp. 32 bitů pokud užíváme 8bitů na zadefinování poloprůhlednosti, tzv. alfa kanál)
  - Nejrozšířenější
  - Postačuje pro většinu aplikací
  - Nativní pro všechny grafické knihovny
    - Barva se specifikuje trojicí celých čísel (0-255)
  - Užívá např. formát PNG

```
var mySRgbColor = Color.FromArgb(255, 0, 0, 255);
```

UKLÁDÁNÍ BARVY

- Menší počet bitů se dnes používá již jen ve speciálních případech
- Bitová mapa
  - 1 bit => 2 možné úrovně
  - Černobílý obrázek
  - „Raw“ předloha pro tisk
- 4 bity
  - 16 úrovní
  - Prehistorické
  - 1/2 bytu
    - Dobře se čte

UKLÁDÁNÍ BARVY

- High-Color
  - RGB barva uložena na 2 byty
    - Pixel se snadno přečte
  - Dvě možnosti:
    - 1 bit volný (resp. použit na rozlišení průhledný / neprůhledný) + 5 bitů pro každou barevnou komponentů
      - 32*32*32 = 32768 barev
    - 5 bitů pro R a B, 6 bitů pro G
      - Lidské oko nejcitlivější na G

    Celkem 65536 barev (64K)
  - Požívá se dnes např. při přenosu obsahu obrazovky vzdálené plochy v případě pomalého připojení

UKLÁDÁNÍ BARVY

- scRGB
  - Hodnota $x \in (-0.5 – cca\ 7.5)$ uložena na 16 bitech jako 8192*x + 4096
  - 48 bitů na pixel (resp. 64 použijeme-li 16 bitů pro zadefinování průhlednosti)
    - Lze realizovat $2^{16}*2^{16}*2^{16}$ = cca $2.8*10^{14}$ různých barev
      - Mnohé však mimo viditelné spektrum
  - Nativně podporováno od Windows 7
  - Podporuje např. grafická knihovna WPF
    - Zadává se jako reálné číslo (float)

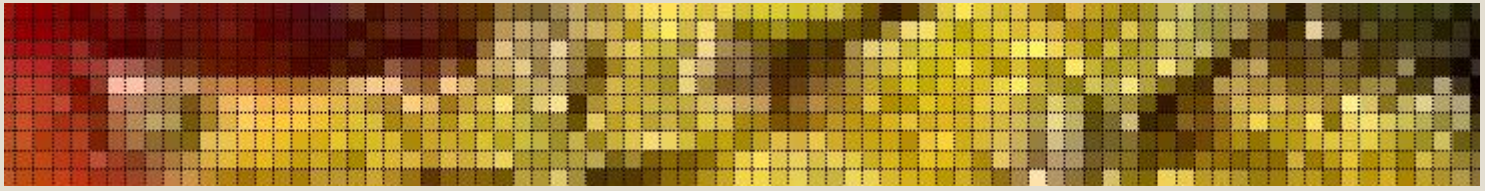
```
var myScRgbColor = Color.FromScRgb(1, 0, 0, 1);
```

UKLÁDÁNÍ BARVY

- 12, 14 nebo 16 bitů
  - 4096, 16384 nebo 65536 úrovní
  - Obvykle ukládáno na 2 byty
    - Nejvyšší bity se nepoužijí
  - Používá se např. ve filmovém průmyslu
    - "raw" výstup z digitálního fotoaparátu má 14 bitů
      - typicky
  - Často používáno v medicínských obrázcích
    - Jen jedna barevná komponenta
    - Pseudo-barva

BITMAPOVÝ OBRAZ

- Matice pixelů MxN
- Interně jednorozměrné pole
  - Pro přístup k pixelu [i, j] nutno spočítat index
- Barva v pixelu uložena přímo nebo nepřímo
- Přímé uložení
  - Hodnoty komponent uloženy za sebou v poli
  - Pořadí (pro stejný barevný model) se může lišit
    - Např. RGB nebo BGR
  - Naprostá většina dnes používaných formátů



BITMAPOVÝ OBRAZ

- Nepřímé uložení
  - Hodnota pixelu v poli je index do tabulky barev
    - Tabulka barev = tzv. paleta
  - Vhodné pro obrázky s malým počtem unikátních barev
  - Používá např. GIF, ale také takto se obarvují medicínská a jiná vědecká data
  - Nejčastěji 256 nebo 4096 unikátních barev + 8bitový nebo 12bitový index

BITMAPOVÝ OBRAZ

- Mnoho bitmapových formátů či grafických knihoven z důvodu urychlení požaduje, aby 1. pixel i-tého řádku začínal vždy na začátku nějakého bytu
  - Často též na adrese zarovnané na 32 bitů
- Důsledek:
  - Za posledním pixelem i-tého řádku může být v poli několik nevyužitých bitů
    - A to dokonce i v případě 24bitového True-Color

```
byte[] obrazek;
pixel_index = i*(M*bitu_na_pixel + bitu_zarovnani)/8
              + j*bitu_na_pixel/8;
```

BITMAPOVÝ OBRAZ

- Grafické knihovny obvykle bitmapový obraz reprezentují nějakou třídou
- Obsahuje metody pro
  - vytvoření nového obrazu MxN pixelů
    - V požadovaném způsobu zápisu pixelů
  - čtení a zápis barvy (obvykle v RGB) pixelu [i, j]
    - POMALÉ!
  - přístup k poli pixelů pro čtení i zápis

BITMAPOVÝ OBRAZ

- Grafické knihovny dále poskytují metody pro
  - Načtení obrazu z nějakého bitmapového formátu
    - Obvykle podporován: BMP, GIF, PNG, TGA, JPEG
  - Uložení obrazu do nějakého bitmapového formátu
  - Vykreslení obrazu na grafický kontext
    - V zadaném měřítku

BITMAPOVÝ OBRAZ V JAVĚ

- Bitmapový obrázek = třída `BufferedImage`
  - `java.awt.image`

```
BufferedImage myImage = new BufferedImage(
        100, 100, BufferedImage. TYPE_3BYTE_BGR);
…
for (int y = 0; y < myImage.getHeight(); y++) {
  for (int x = 0; x < myImage.getWidth(); x++) {
        myImage.setRGB(x, y, Color.red.getRGB());
  }
}

Color pixel =  new Color(myImage.getRGB(x, y), false);
```

Tento příklad je neefektivní!!! Zejména v některých grafických knihovnách. Takhle to nedělejte ☺

BITMAPOVÝ OBRAZ V JAVĚ

- Lepší řešení

```
int w = m_image.getWidth();

int red = Color.red.getRGB();
int[] rgbArray = new int[w];
for (int x = 0; x < w; x++) {
  rgbArray[x] = red;
}

int h = m_image.getHeight();
for (int y = 0; y < h; y++) {
  m_image.setRGB(0, y, w, 1, rgbArray, 0, w);
}
```

BITMAPOVÝ OBRAZ V JAVĚ

- Načtení a uložení obrazu: třída `ImageIO`
  - `javax.imageio`

```
BufferedImage img = ImageIO.read(new File("obrazek.png"));

…

try {
    File outputfile = new File("saved.png");
    ImageIO.write(img, "png", outputfile);
} catch (IOException e) {
    ...
}
```

BITMAPOVÝ OBRAZ V JAVĚ

- Zobrazení obrazu: metoda `drawImage`
  - Poměr 1:1 nebo jiné měřítko
    - Rychlá lineární interpolace

```
g2.drawImage(m_image, 0, 0, null);//normal 1:1
g2.drawImage(m_image, 0, 0,
       this.getWidth(), this.getHeight(), null); //scaled
```

BITMAPOVÝ OBRAZ V JAVĚ

- Lepší (bikubická) interpolace
  - Pomalejší, ale defekty nejsou tak vidět

```
AffineTransform at = AffineTransform.getScaleInstance(
        this.getWidth() / (double)this.m_image.getWidth(),
        this.getHeight()/ (double)this.m_image.getHeight()
);

AffineTransformOp op = new AffineTransformOp(at,
        AffineTransformOp.TYPE_BICUBIC );

g2.drawImage(m_image, op, 0, 0);
```

KONEC

- Příště: Pokročilá bitmapová grafika