
Barva, světlo, materiály v počítačové grafice

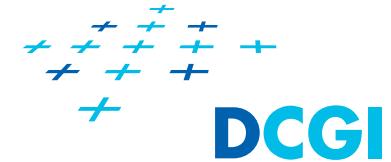
Petr Felkel

Katedra počítačové grafiky a interakce, ČVUT FEL
místnost KN:E-413 (Karlovo náměstí, budova E)

E-mail: felkel@fel.cvut.cz

S použitím knihy [MPG] a materiálů Jaroslava Křivánka,
Jaroslava Sloupa a Vlastimila Havrana

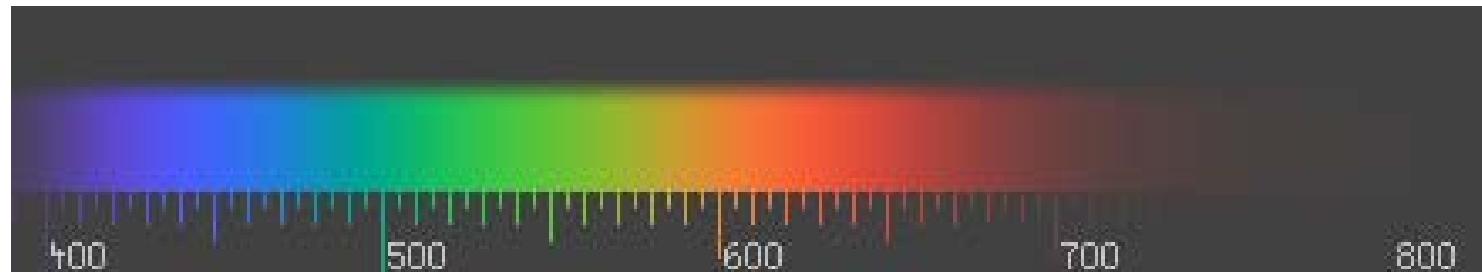
Osnova



- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

Viditelné světlo

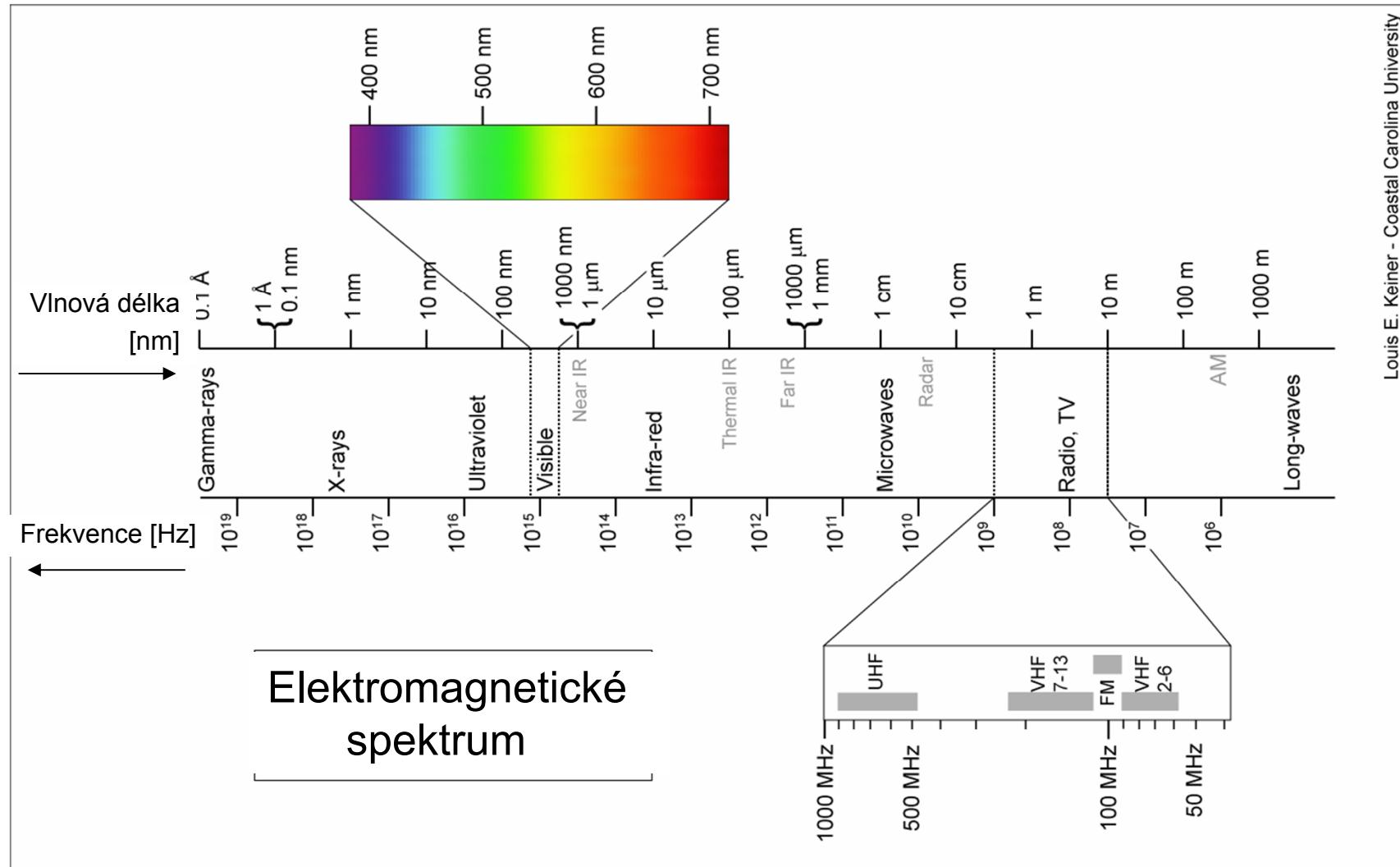
- EM záření cca 380nm – 720nm na něž jsou citlivé buňky sítnice



- Světlo je *fyzikální* jev
- Pro popis světla lze použít fyziku

→ Vlnová délka

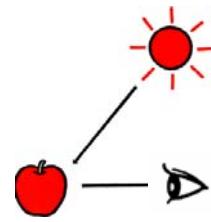
Viditelné světlo



Barva

= Vjem

- Nekoresponduje přímo (1 : 1) s fyzikální skutečností
- Pro popis barvy je třeba vycházet z psychologie (experimentální psychologie, psychofyziky)
 - Tendence posouvat rozsah oka tak, aby vnímána *bílá* Na fotografii vidíme, že večer do modra, žárovka do žluta,... (chromatická adaptace)
 - V noci *posun relativní citlivosti k modré* (modrý papír vypadá v noci jako světlejší než červený a naopak) [Purkyňův efekt]
 - *Metamerismus* = různé materiály vnímány jako stejná barva (díky tomu lze sestrojit televizi či namíchat „stejné“ barvy s jiným rozpouštědlem vodové, acetonové, ...) – záleží ale na spektru světla, které objekt osvětuje!
 - ...

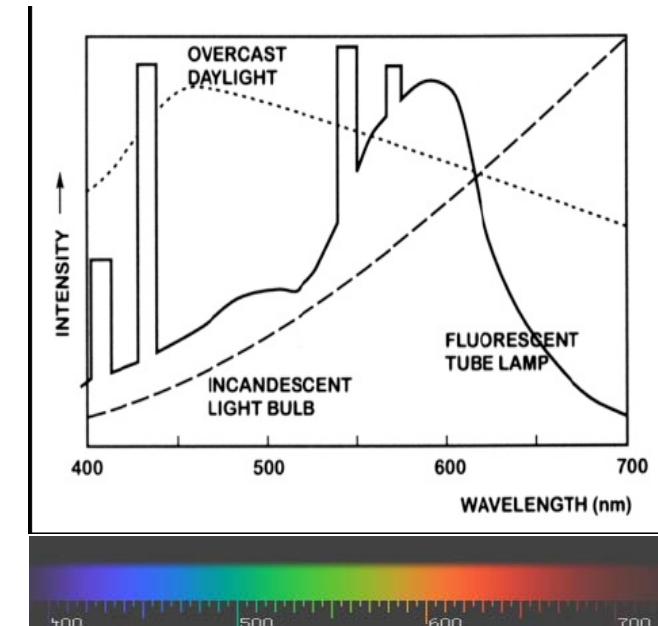


Čím je určen barevný vjem

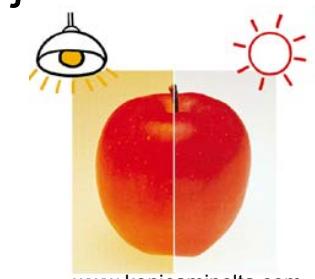


A. Zdroj světla

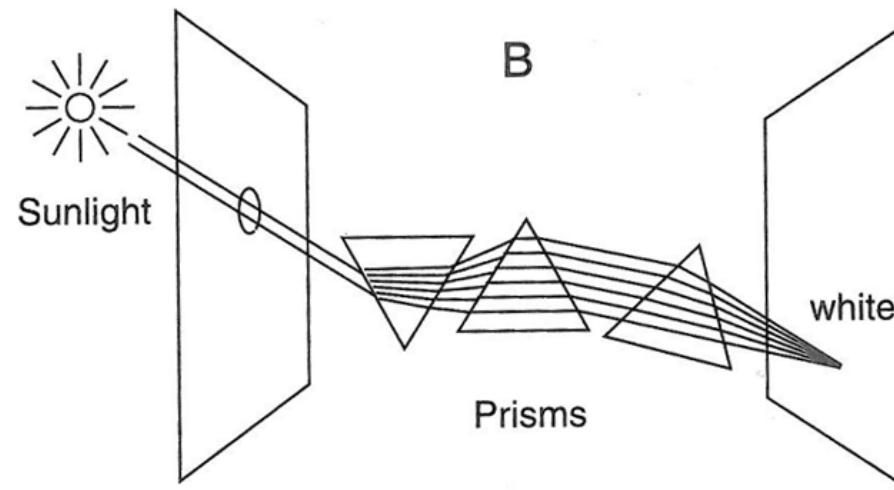
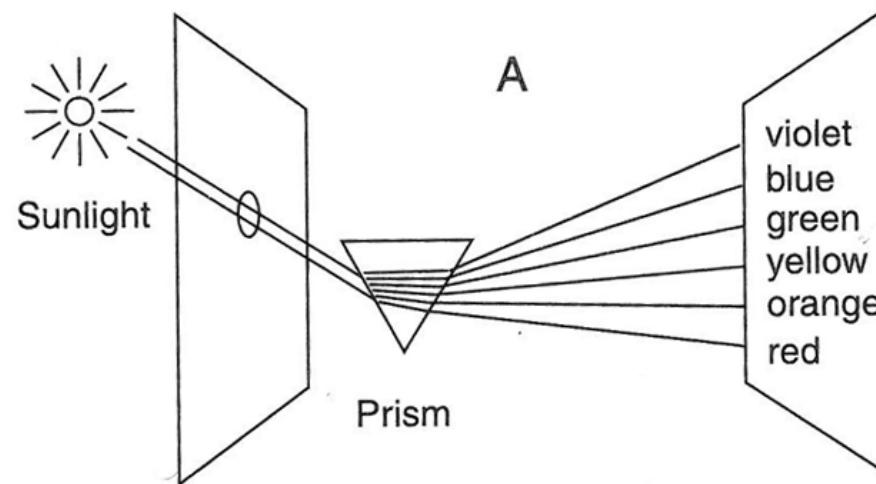
- Hustotu světelného toku v závislosti na vlnové délce λ vyjadřuje **spektrální radiance**, $L(\lambda)$
 - ◆ Pozn.: Oficiální, normou přijatý, český název radiance, používaný v osvětlovacím inženýrství, je „zář“, v grafice se však používá slovo radiance
- Určuje barevnou adaptaci oka
 - ◆ Člověk má tendenci interpretovat barvu světla jako bílou, nezávisle na jejím skutečné odstínu

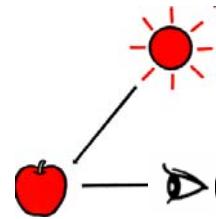


- B. Vlastnosti materiálu pozorovaného objektu
- C. Perceptuální procesy v oku a mozku



Rozklad denního světla (Newton)





Čím je určen barevný vjem

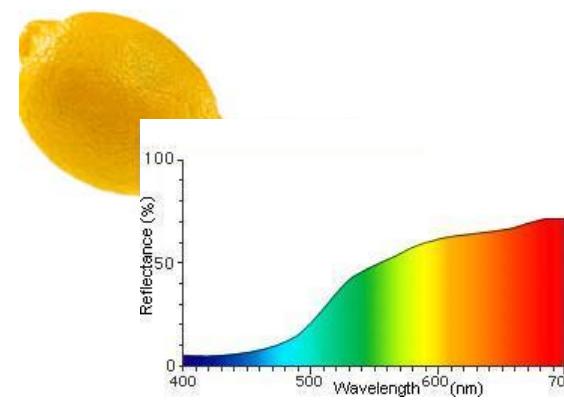
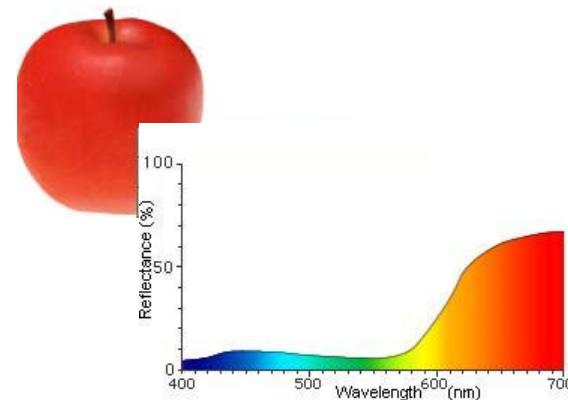


A. Zdroj světla

B. Vlastnosti materiálu pozorovaného objektu

- **Spektrální odrazivost $\rho(\lambda)$**

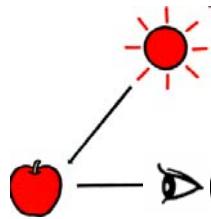
$$\text{Radiance } L_{\text{odražená}}(\lambda) = L_{\text{příchozí}}(\lambda) \rho(\lambda)$$



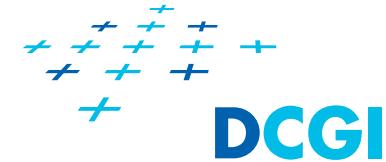
www.konicaminolta.com

- Díky chromatické adaptaci (na bílou) má spektrální odrazivost $\rho(\lambda)$ daleko zásadnější vliv na vnímanou barvu objektu než spektrální radiance $L_{\text{příchozí}}(\lambda)$

C. Perceptuální procesy v oku a mozku



Čím je určen barevný vjem



A. Zdroj světla

B. Vlastnosti materiálu pozorovaného objektu

C. Perceptuální procesy v oku a mozku

- Radiance $L(\lambda)$ vstupující do oka (tj. spojitá funkce vlnové délky) je „dekódována“ na **vědomý vjem**, jež popisujeme v dimenzích:
 - ◆ **Barevný odstín** (Hue)
 - ◆ **Sytost** barvy (Saturation) – jak je barva čirá (užší spektrum)
 - ◆ **Světllost, jas** (Brightness, Lightness, Intensity) – jak je barva světlá

Percepční funkce lidského OKA



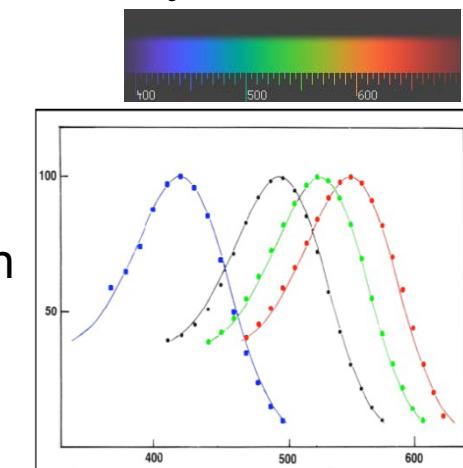
Světlo prochází čočkou a promítá se na sítnici, kde jsou světlocitlivé buňky:

- Tyčinky

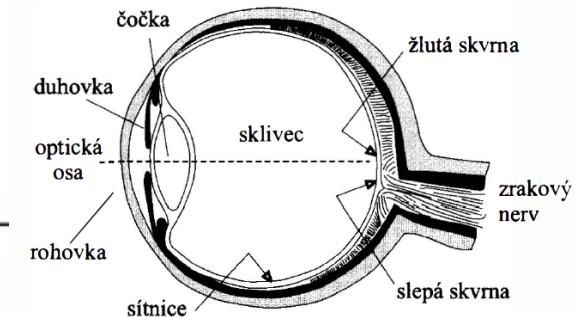
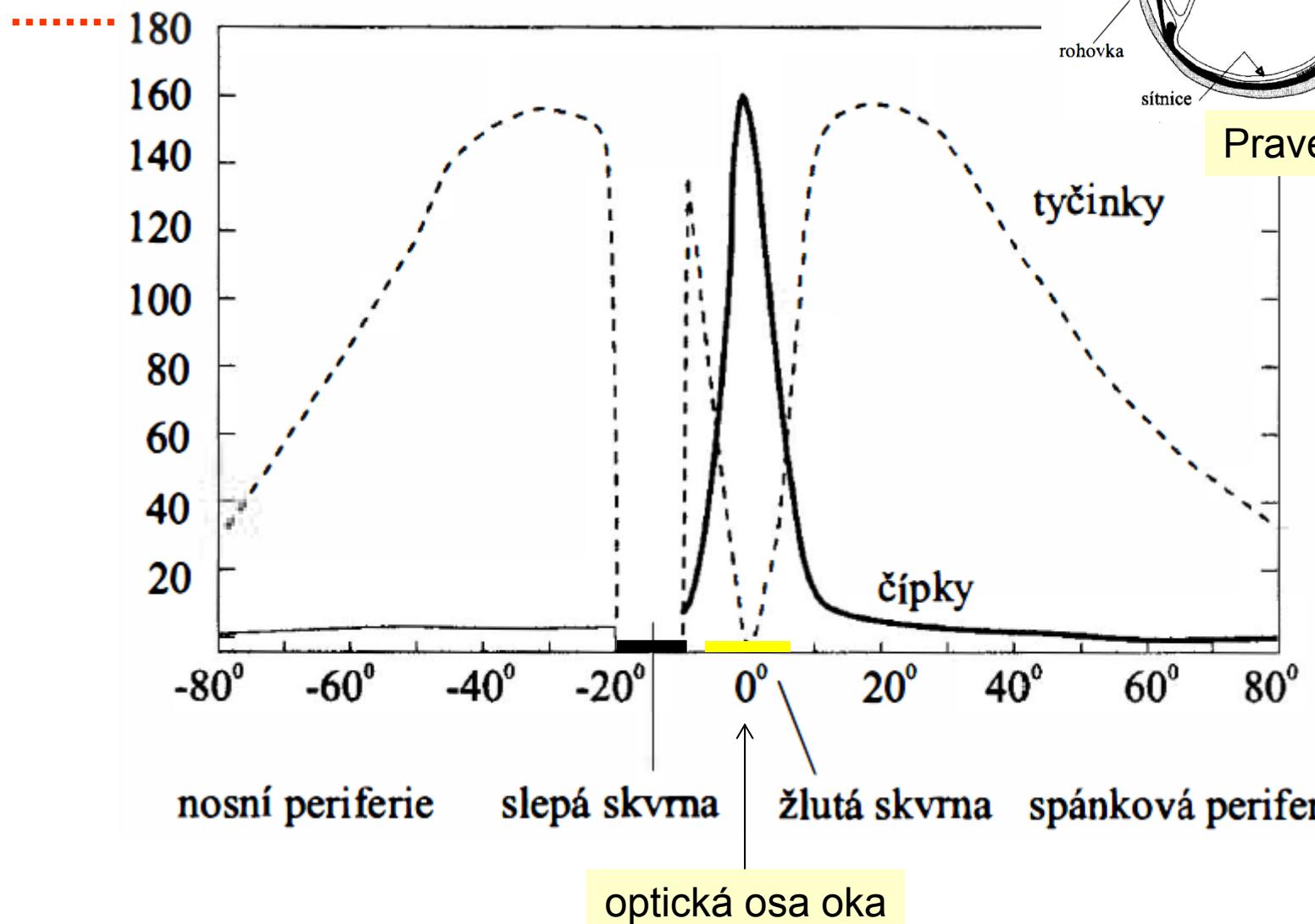
- Jsou aktivovány při nízkých intenzitách světla – noční vidění
- Při denním světle jsou saturovány – nedodávají žádný signál
- Pouze jeden druh tyčinek – černobílé vidění
=> Za tmy je obtížné až nemožné rozeznávat barvy

- Čípky

- 3 druhy reagující různě citlivě na světlo o různých vlnových délkách
 - S / M / L ... short / medium / long wavelength
(krátké, střední a dlouhé vlnové délky)
- Zprostředkovávají barevné vidění

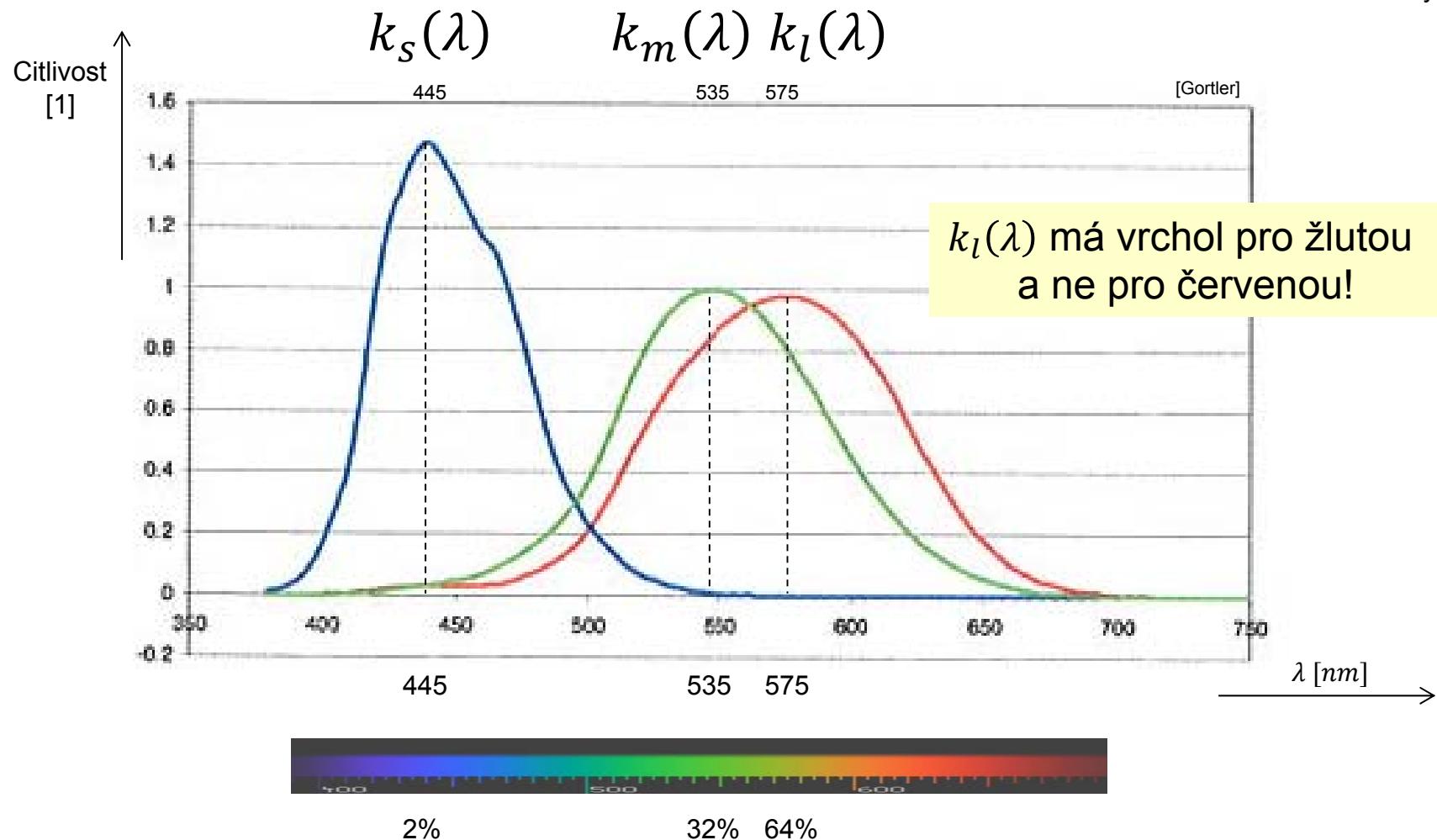


Rozložení fotoreceptorů na sítnici

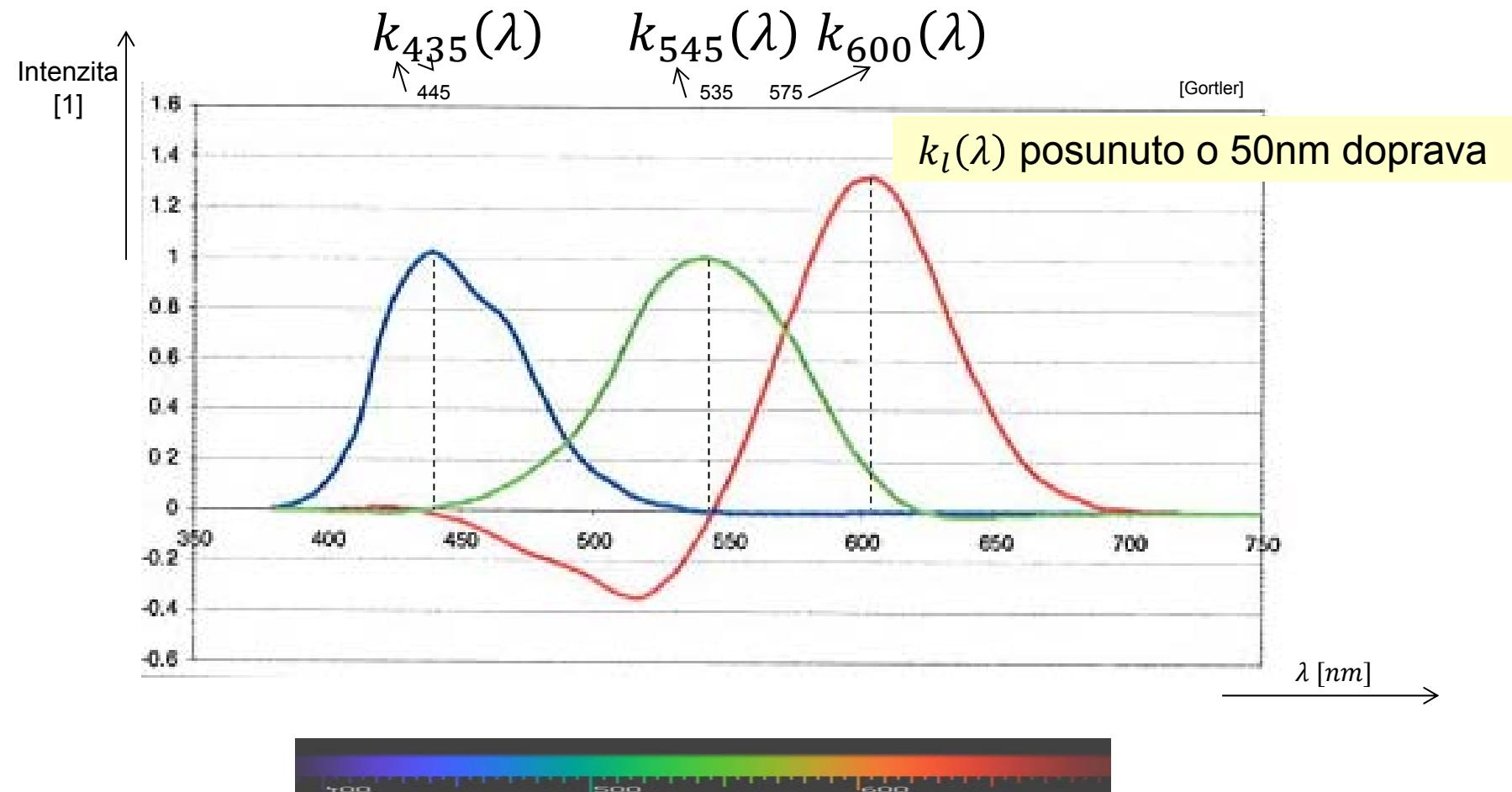
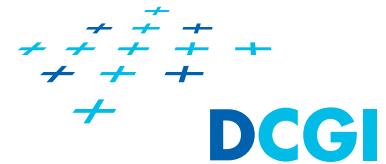


Pravé oko shora

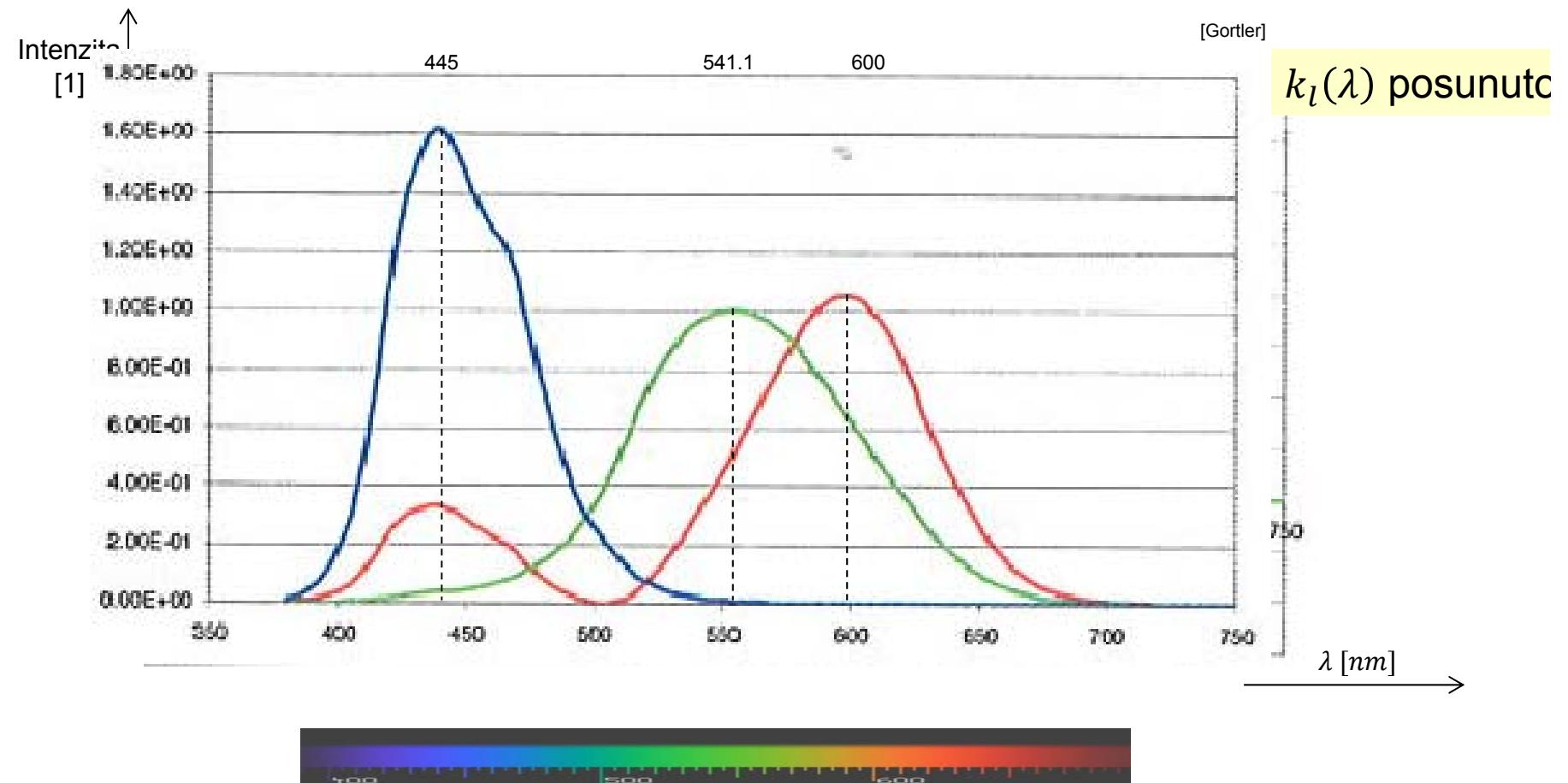
Relativní citlivost čípků na různé frekvence



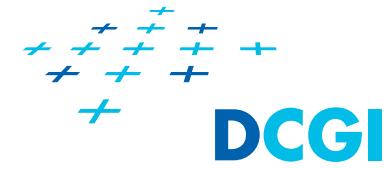
Srovnávací funkce pro XYZ



Světelné zdroje pro porovnávání barev

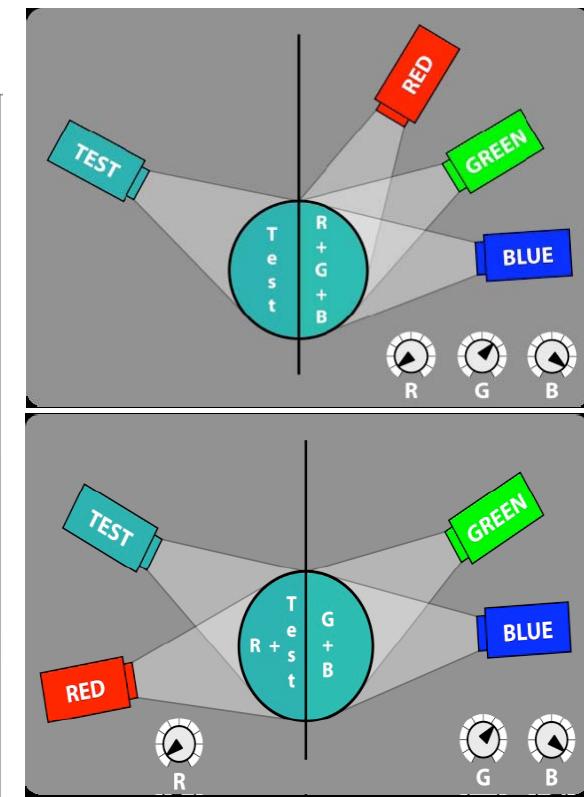
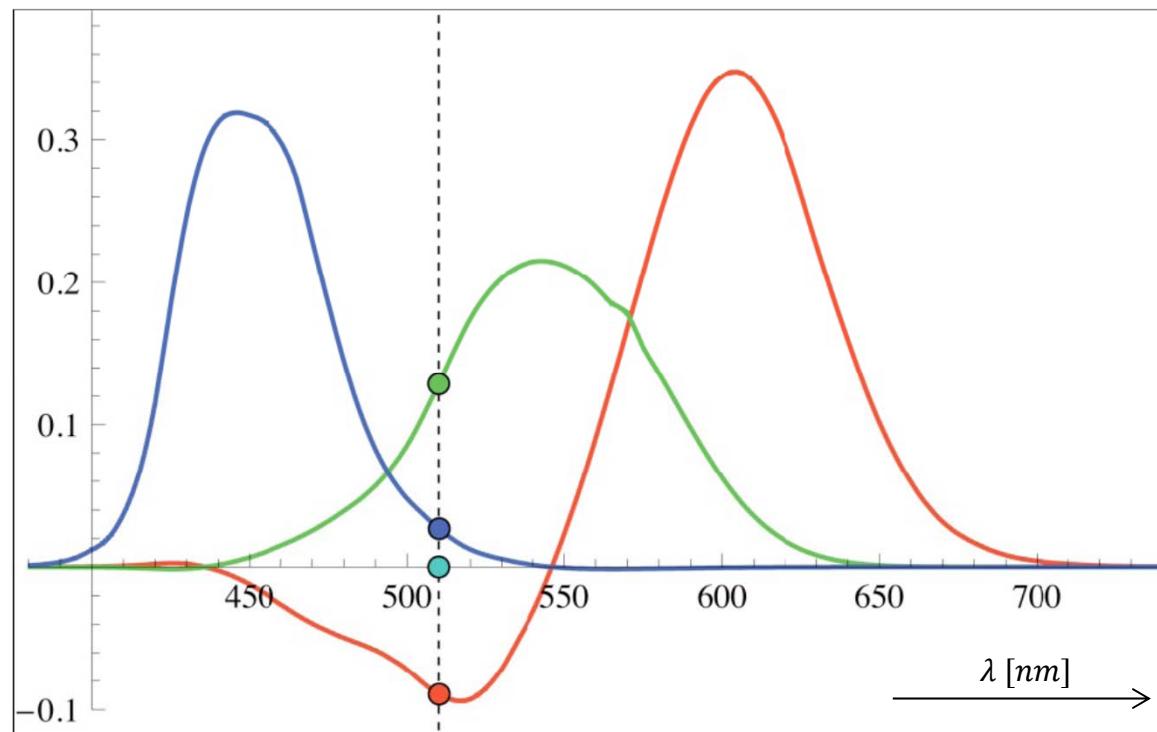


Čím je určen barevný vjem

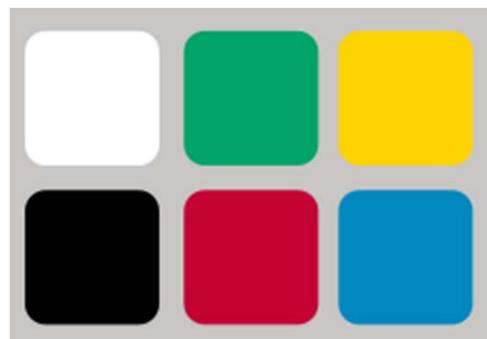
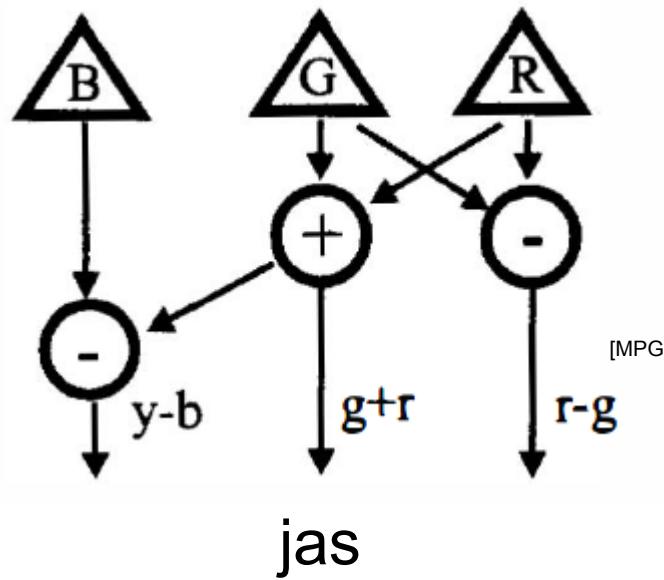


Experiment s porovnáváním barev (CIE1931- 2° a 1964 -10°)

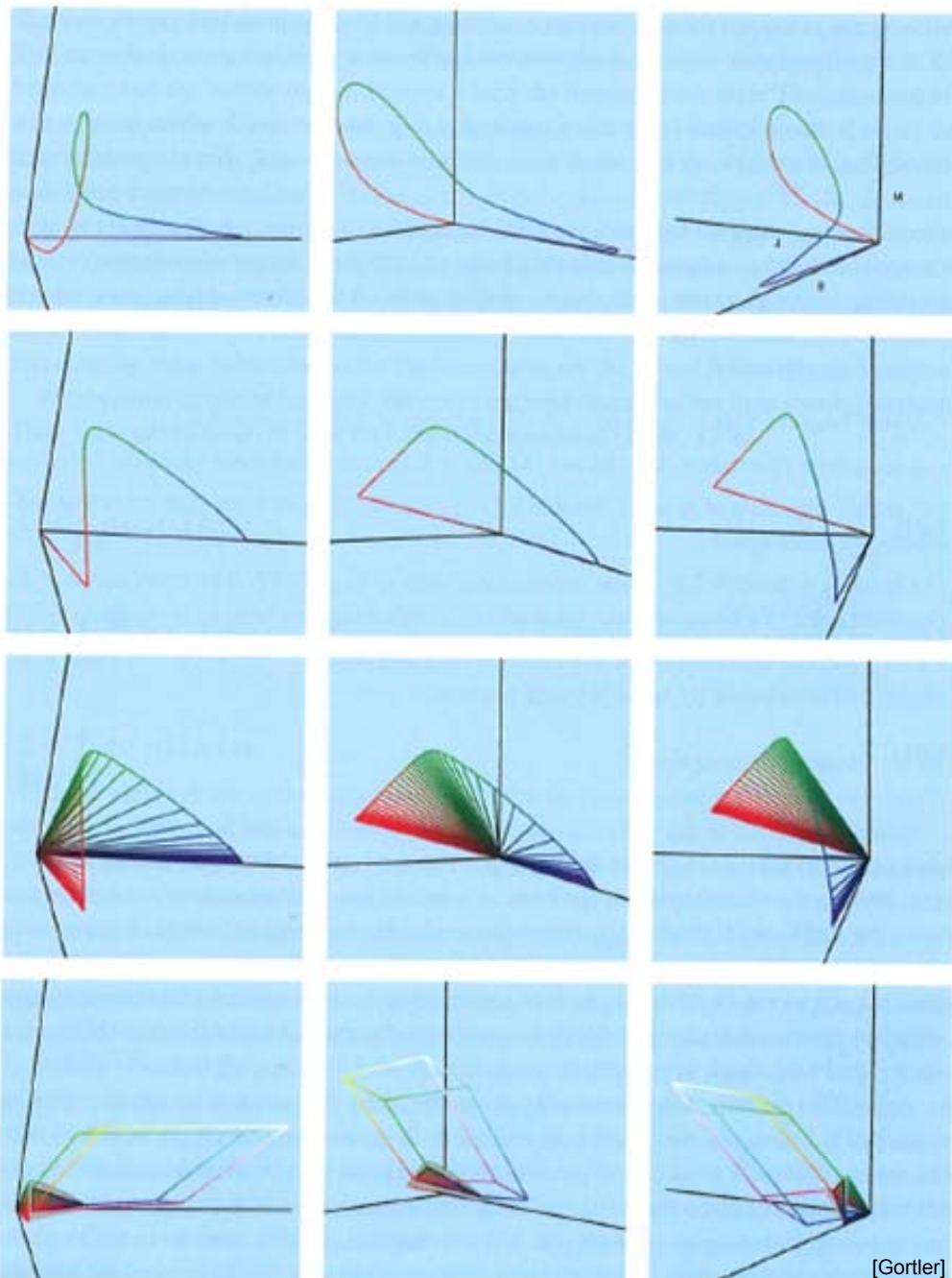
- Monochromatické světlo ~ kombinaci daných R,G,B světel



Rekombinace barevných stimulů



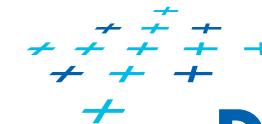
- Ještě v oku se barevná informace rekombinuje před připojením očního nervu
- Kanály
 - r-g červená-zelená
 - y-b žlutá-modrá
 - g+r zelená+červená ~ jas (B/W)
- Neexistuje
 - Nazelenalá červeň
 - Nažloutlá modř
- Podobný princip vkódování TV signálů



Prostor LMS (citlivost čípků oka)

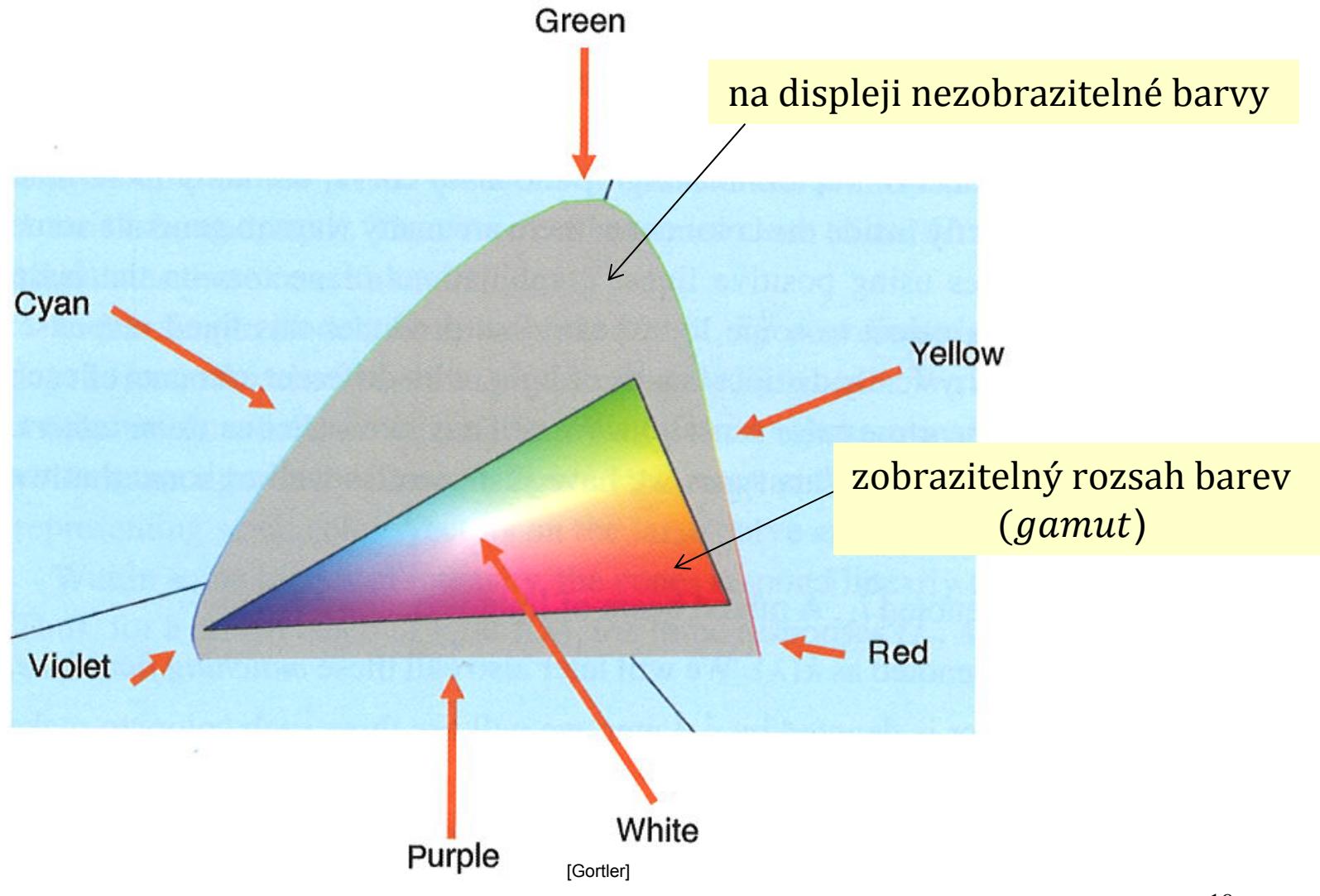
Zelená osa je imaginární

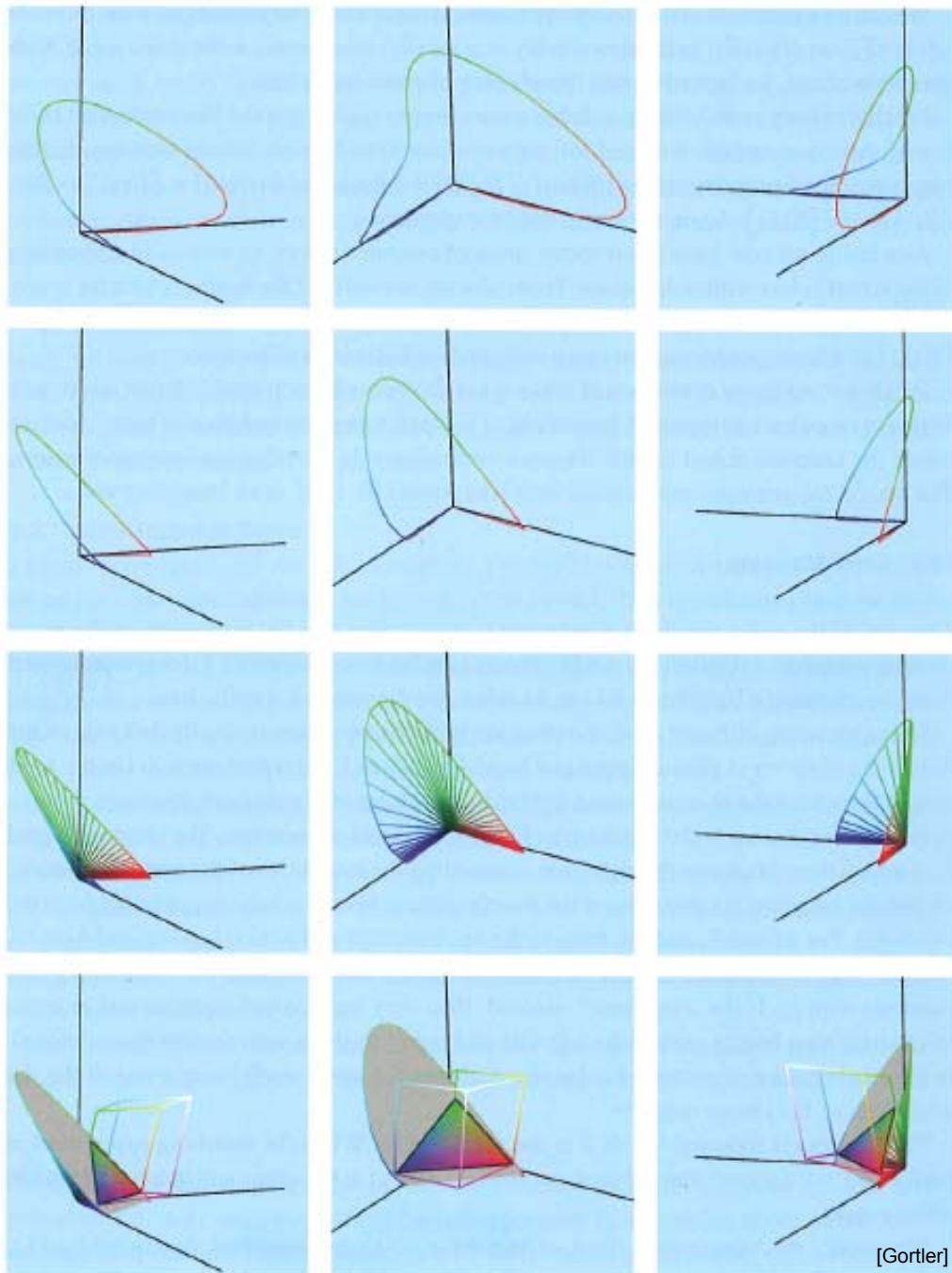
- nelze izolovaně vybudit jen čípky typu **M**
- vždy se vybudí i **S**, nebo **L**



DCGI

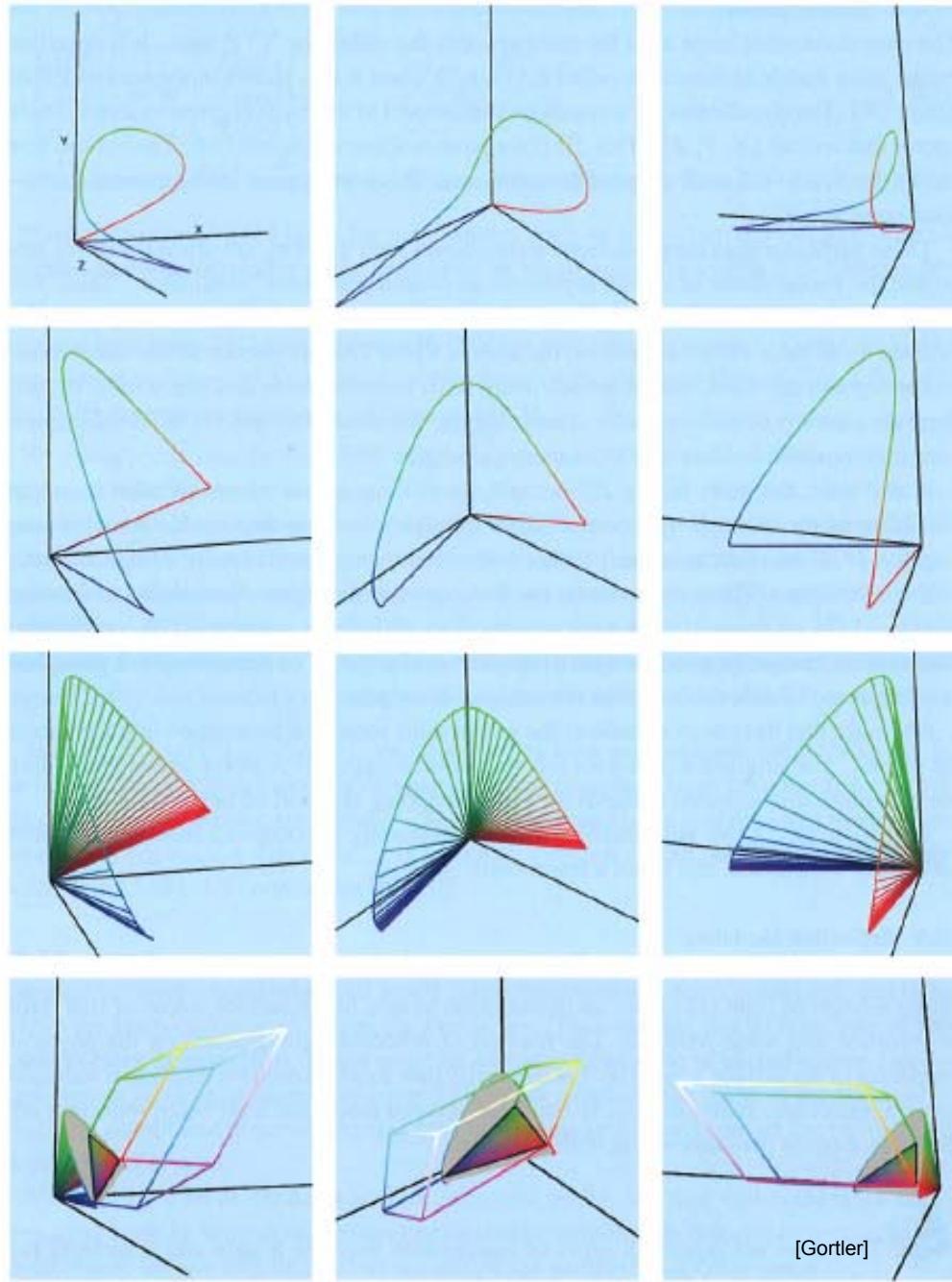
2D diagram barev





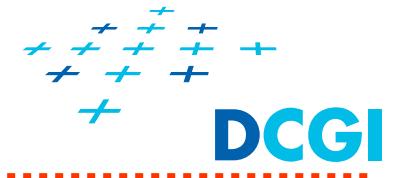
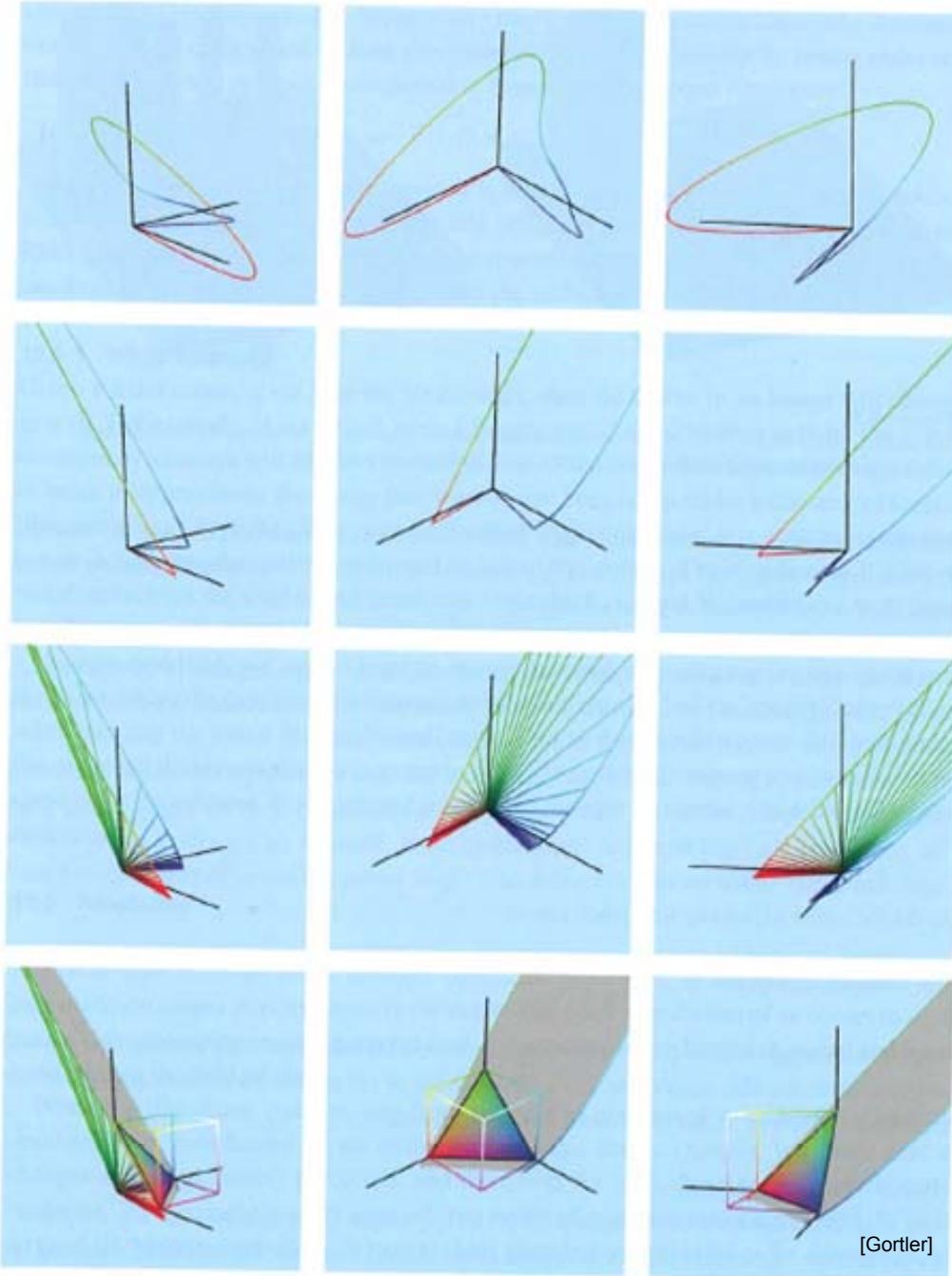
Prostor experimentu
porovnávání barev

(křivka vybíhá z I.
kvadrantu do záporné
červené osy)



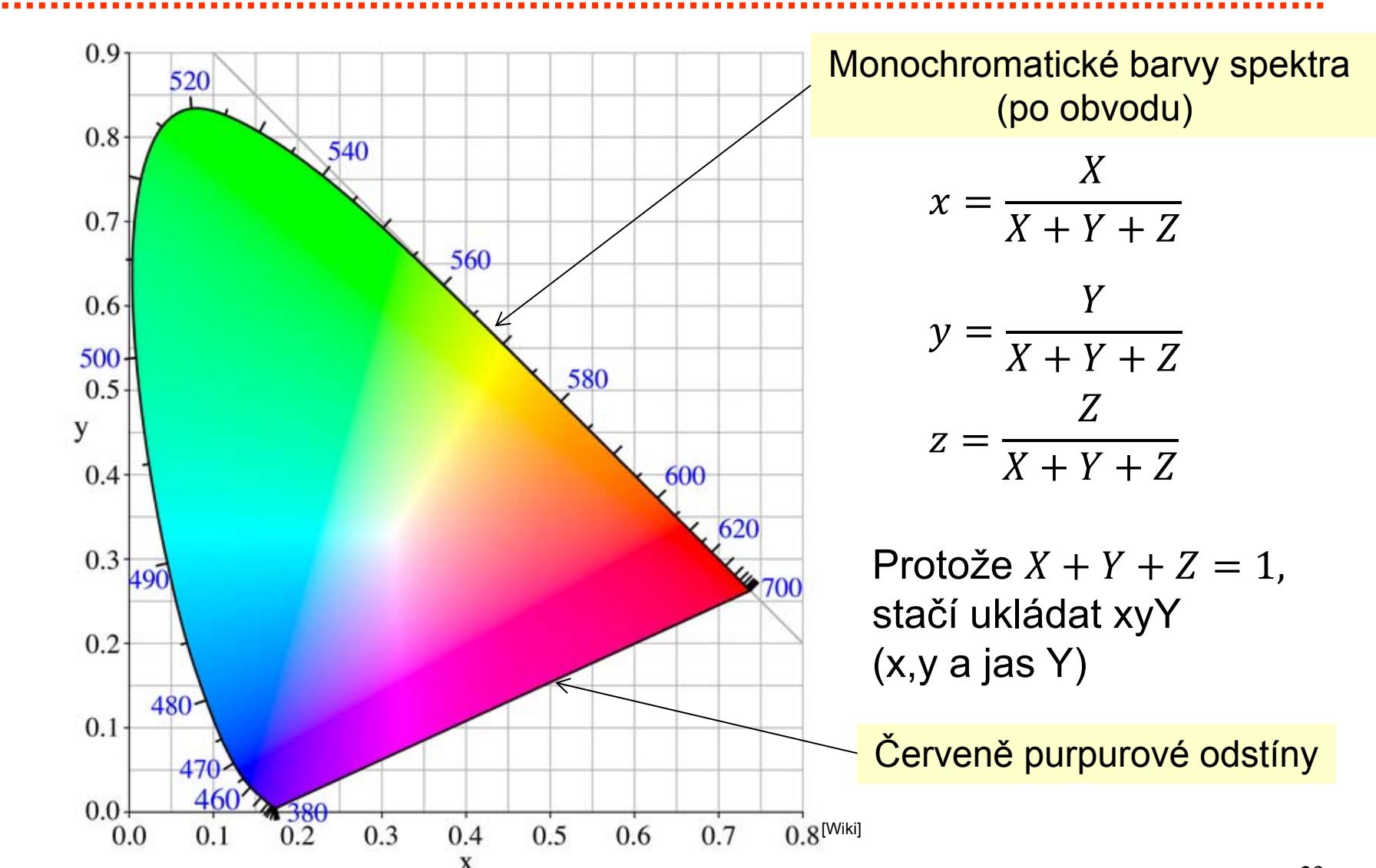
Prostor barev XYZ

(zůstává v I. kvadrantu
Za cenu imaginární
barev)



Prostor barev RGB pro
reálné barvy – reálné
zařízení

Chromatický diagram CIE 1931 xyY



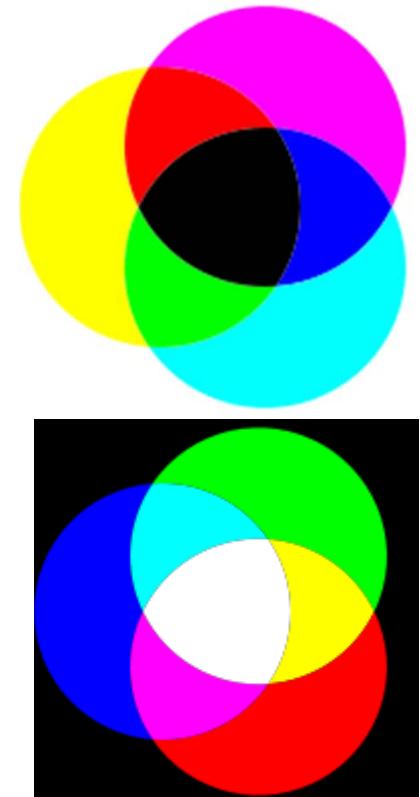
Osnova

- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (flat, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

Barevné modely a OpenGL

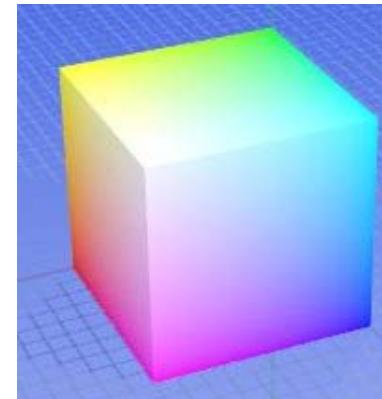
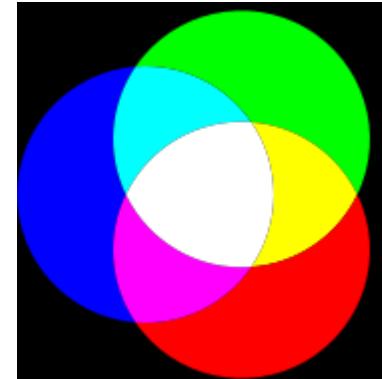
- Různá zařízení mají různý princip generování barev
 - **Tiskárny**
 - překrývají inkousty, každý utlumí část spektra
(subtraktivní skládání barev)
 - CMY, CMYK
 - **Obrazovky**
 - rozsvěcují barevné obrazové elementy
(aditivní skládání barev)
 - RGB, sRGB,...

OpenGL typicky používá RGB
 - barevné modely
 - ♦ RGB, CMY, CMYK, ... = poměrné, pro dané zdroje
 - ♦ XYZ , $sRGB$, (L^*, u^*, v^*) , (L^*, a^*, b^*) , ... – absolutní
– jednoznačný popis barvy



RGB – Red Green Blue

- Pro televizi, monitory, projektoru
- Aditivní skládání barev
 - Odpovídá přidávání světla (např. emitovaného luminoforem v televizi)
 - $[0,0,0]$ = černá
 - $[1,1,1]$ = bílá
- RGB krychle
 - Vizualizace RGB prostoru
- Luminance RGB barvy:
 - $0.299 r + 0.587 g + 0.114 b$
 - V různých aplikacích různé vzorce



RGB – Red Green Blue

Mnoho různých RGB prostorů:

- Standardy
 - CIE RGB 1931
 - sRGB (pro web)
- Každé výstupní zařízení má „svůj“ RGB prostor
- Pokud dostanete RGB obrázek bez informace o kalibraci, nevíte o barvách nic



CMY – Cyan Magenta Yellow

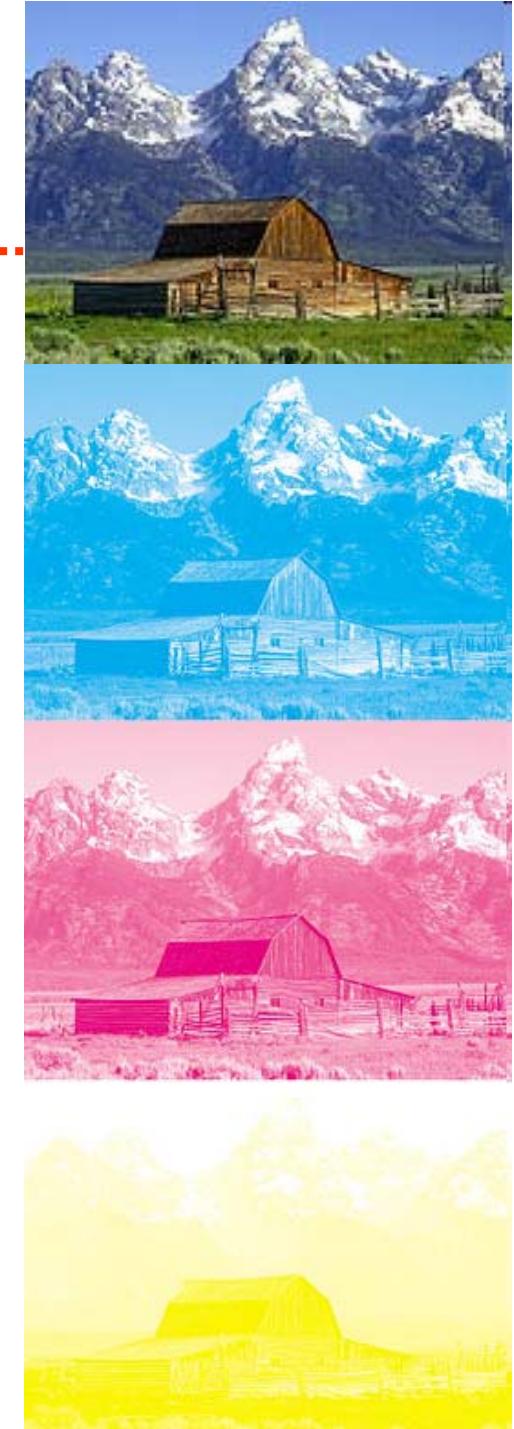
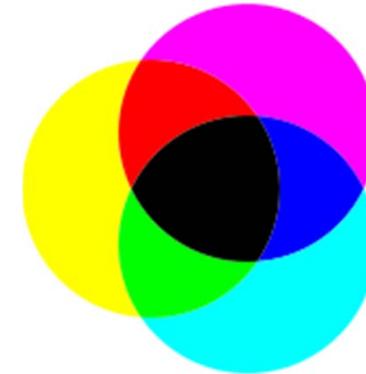
- Pro tisk
- Subtraktivní skládání barev
 - Odpovídá míchání barevných pigmentů v malbě a tisku
 - $[0,0,0]$ = bílá (barva papíru)
 - $[1,1,1]$ = černá (maximum všech pigmentů)
- Převod RGB => CMY

$$[C, M, Y] = [1 - R, 1 - G, 1 - B]$$

(POZOR – tento vztah platí jen **velmi přibližně**. Pokud chceme vytisknout „stejnou“ barvu, jako na obrazovce, musíme obě zařízení zkalibrovat a vzít v úvahu konkrétní barevný rozsah tiskárny i monitoru - gamut)

Proto se převádí pomocí systémů pro správu barev za použití barevných profilů zařízení.

Navíc to nikdy úplně nejde - mají značně **odlišný gamut**



CMYK = CMY + Key (blacK)

- Kvalita černé není dobrá, vzniká-li mísením barevných pigmentů
 - Při tisku 100% CMY barvy rozmáčí papír, pomalu schnou
 - CMY černá je spíše tmavě hnědá
 - Černý pigment je levnější než barevný
- Proto se přidává černý pigment mezi základní barvy
- Převod C'M'Y' => CMYK

$$K = \min(C', M', Y')$$

$$[CMYK] = \left[\frac{C' - K}{1 - K}, \quad \frac{M' - K}{1 - K}, \quad \frac{Y' - K}{1 - K}, \quad K \right]$$

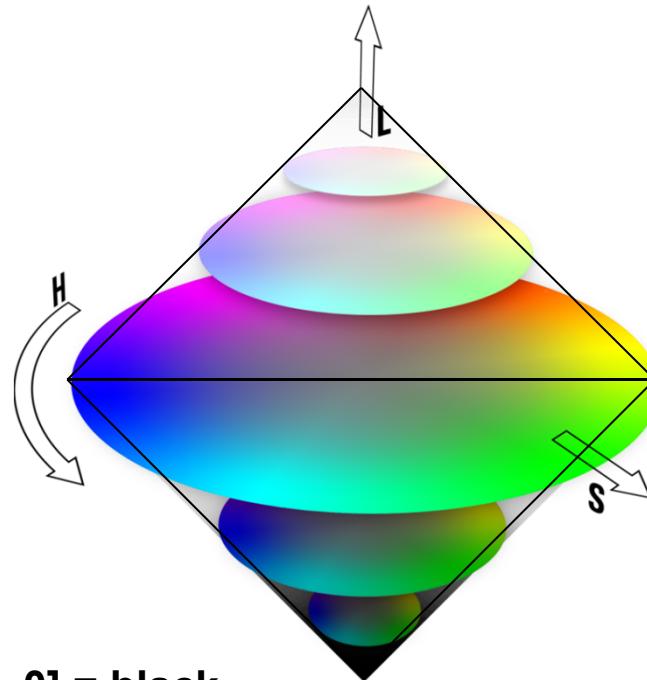
(POZOR – tento vztah platí jen velmi přibližně – viz předchozí str.)



HLS color model

HLS - Hue, Lightness, and Saturation

- Úhel okolo osy (jehlanů) odpovídá odstínu (*hue*)
- Vzdálenost od osy sytosti (*saturation*)
- Vzdálenost podél osy jasu (*lightness*)



[170, x, 0] = greys

[170, 1, 0] = white

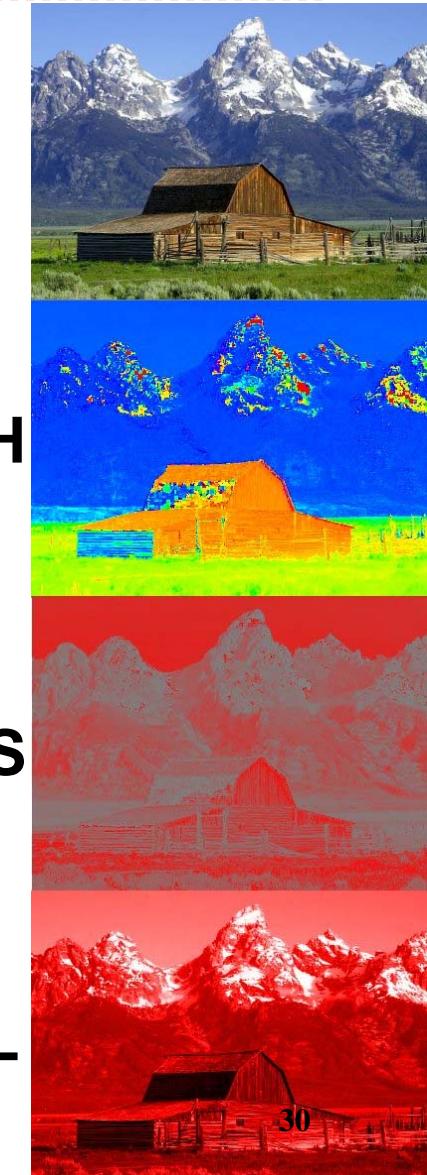
[127, 0.5, 1] = cyan

[170, 0, 0] = black

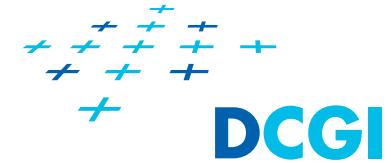
[213, 0.5, 1] = magenta

[42, 0.5, 1] = yellow

- *HLS model lépe odpovídá intuitivnímu vnímání sytosti a jasu, jako dvou oddělených parametrů*



Barva a světlo podrobněji



[MPG] Žára a kol. Moderní počítačová grafika,
Kap. 1 „Světlo a barvy v počítačové grafice“. Computer Press, 2004

[Hunterlab] Obsažnější úvod do teorie barev:

<http://www.hunterlab.com/ColorEducation/ColorTheory>
<http://www.hunterlab.com/pdf/color.pdf>

[Wikipedia]

<http://en.wikipedia.org/>, heslo “cie xyz”, “color space” a odkazy

[SKALA] Skala V. „Světlo, barvy a barevné systémy v počítačové
grafice“. Praha: ČVUT, 1993

Osnova

- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

Osvětlování - *lighting*

- Osvětlení přidá na realističnosti objektů ve scéně

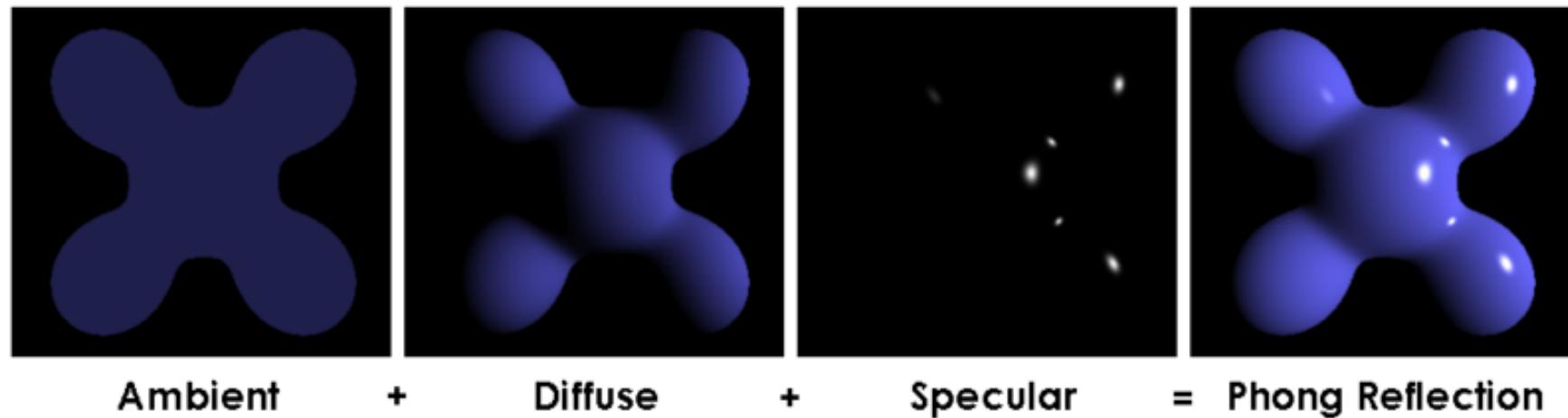


Výpočet osvětlení v bodě scény

Phongův osvětlovací model (*reflection model*) v OpenGL

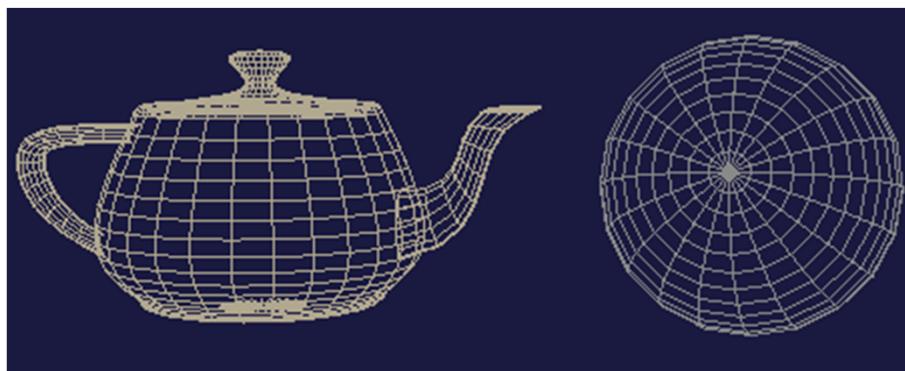
- Ambientní (globální od prostředí a od světel)
- Difúzní (od světel)
- Zrcadlově odražené (od světel)
- Vyzářené (tělesem)

[Wikipedia] http://en.wikipedia.org/wiki/Phong_reflection_model



Ambientní světlo (ambient)

- **Ambientní světlo** – rozptýleno prostředím tolikrát, že není možné určit směr, odkud přichází (je všesměrové)
(není to obecně úplně pravda - ambient occlusion)
- Odráží se od povrchu také všesměrově (do všech směrů)
- Bez něj jsou odvrácené plochy úplně černé!
- **Je to pouze velmi hrubá approximace reality.**



jen ambientní světlo
=> 3D objekty vypadají jako placaté
(Sníh v mlze)
PGR

Ambientní světlo (ambient)

- **Ambientní světlo od světel**

Rovnice pro odražené ambientní světlo od jednoho sv. zdroje

$$\text{ambient}_{\text{reflected}} = \text{ambient}_{\text{light}} * \text{ambient}_{\text{material}}$$

(nezávisí na poloze světla)

- **Ambientní světlo od prostředí**

Rovnice pro odražené ambientní světlo od prostředí

$$\text{global_ambient}_{\text{reflected}} = \text{global_ambient}_{\text{light_model}} * \text{ambient}_{\text{material}}$$

(nezávisí na světlech)

Difúzní světlo (diffuse)

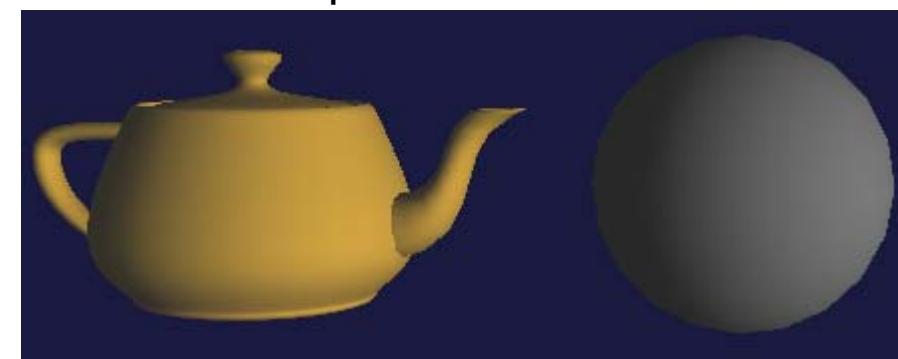
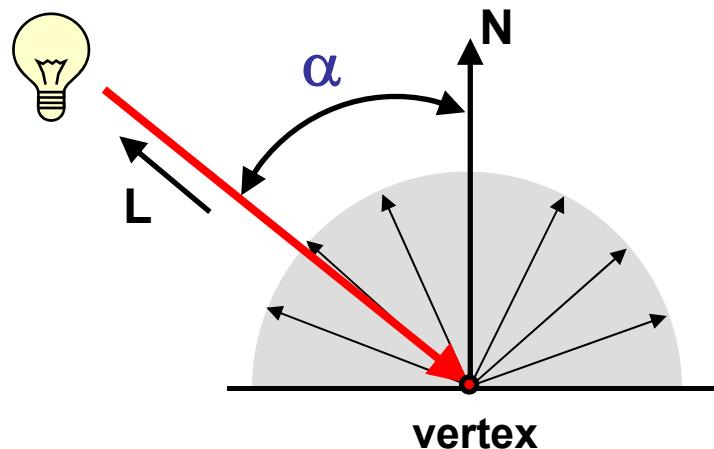
- **Difúzní světlo** je směrové světlo vysílané světelným zdrojem
- Po dopadu je odraženo do všech směrů, proto je stejně jasné, nezávisle na poloze kamery
- Množství rozpáleného světla je úměrné úhlu mezi směrem ke světlu a normálou povrchu \Rightarrow **mění se s cosinem úhlu dopadu**

Rovnice odraženého difúzního světla

$$\text{diffuse}_{\text{reflected}} = (\max[\cos \alpha, 0]) * \text{diffuse}_{\text{light}} * \text{diffuse}_{\text{material}}$$

$\cos \alpha = \mathbf{L} \cdot \mathbf{N}$ skalární součin \mathbf{L}, \mathbf{N} jsou jednotkové vektory

$\cos \alpha$ je maximální při dopadu kolmo na plochu

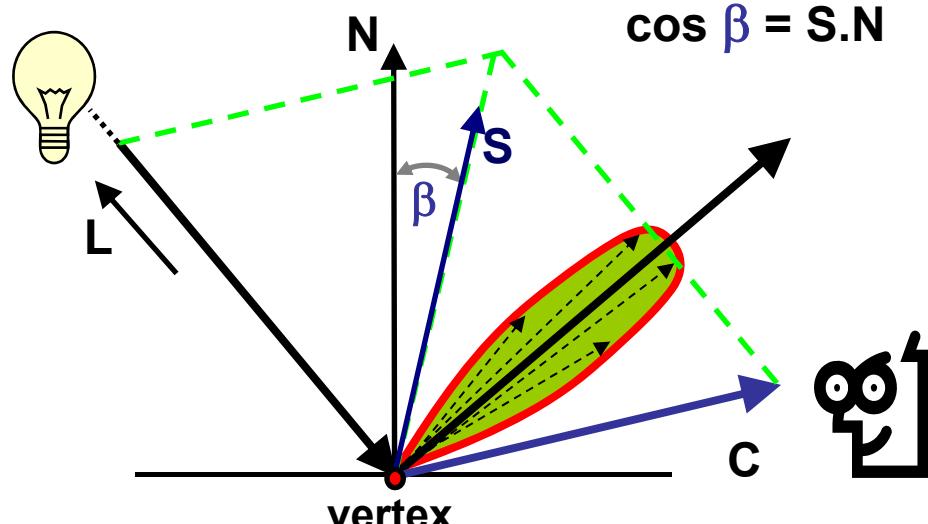


Zrcadlově odražené světlo (specular)

- Přichází z jednoho směru a odráží se do jednoho směru
- Závisí **na úhlu mezi světlem a pozorovatelem (dopadu i oka)**
- Vytváří jasné plochy (specular highlight nebo specular reflection)

Rovnice odraženého zrcadlového světla pro tzv. Blinn-Phongův model

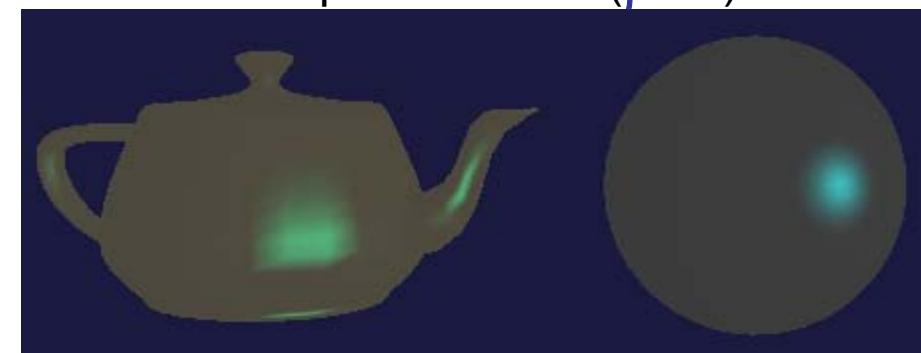
$$\text{specular}_{\text{reflected}} = (\max[\cos \beta, 0])^{\text{shininess_material}} * \text{specular}_{\text{light}} * \text{specular}_{\text{material}}$$



$$S = (C+L) / |C+L|$$

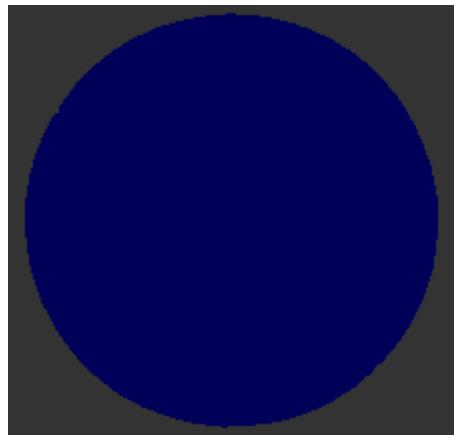
half angle

$\cos \beta = S \cdot N$
 S, N jsou jednotkové vektory
 $\cos \beta$ je maximální když se světlo odráží do oka pozorovatele ($\beta = 0$).



Vyzářené světlo (emissive)

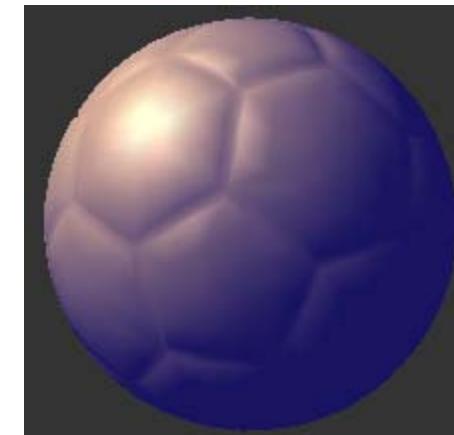
- Simuluje světlo vyzářené objektem, ale
 - nedodává světlo jiným objektům ve scéně
 - není ovlivněno světelnými zdroji
- Objekt osvětlený pouze vyzářeným světlem (a žádným sv. zdrojem) vypadá jako objekt osvětlený ambientním osvětlením



jen vyzářené



zrcadlové + difúzní
+ ambientní



vše dohromady

Osvětlení v bodě scény

- Příspěvky světla se počítají **odděleně pro barevné složky R,G,B**
- Složky ambientní, difúzní, zrcadlově odražené, vyzářené **se sečtou** a oříznou (*clamp*) do intervalu $\langle 0.0, 1.0 \rangle$

Osvětlení v bodě scény (Phongův osvětlovací model)

$$\text{color} = \text{emission} + \text{global_ambient}_{\text{reflected}} + \sum \text{light}_i$$

$$\text{light}_i = \text{spotlightEffect} * \text{attenuationFactor} * (\text{ambient}_{\text{reflected}} + \text{diffuse}_{\text{reflected}} + \text{specular}_{\text{reflected}})$$

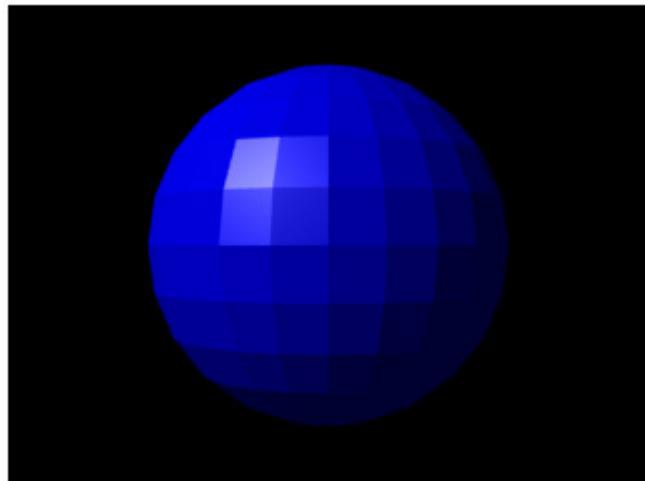
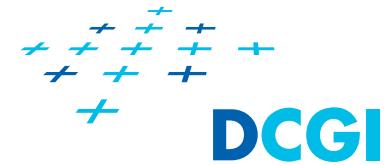
Osnova

- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

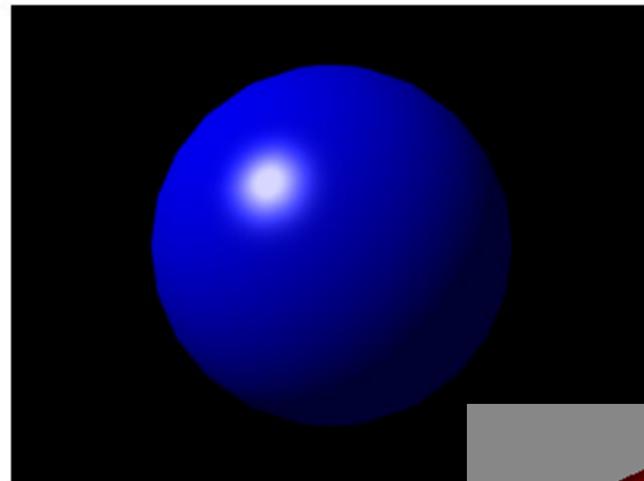
Stínování (shading) = postup vybarvení plošek

- **Konstantní (Flat shading)**: Celý polygon vybarví stejně dle jedné normály. Ploškované.
- **Gouraud shading**: vypočítá osvětlení (barvy) ve vrcholech. V ostatních bodech *interpoluje barvu* bodu podél povrchu polygonu.
Je velmi rychlá a úsporná. Hot spot jen ve vrcholech.
- **Phong shading**: zná normály ve vrcholech, *interpoluje směr normály* podél povrchu polygonu. Osvětlení se počítá pro každý pixel.

Stínování – výpočet osvětlení plošek

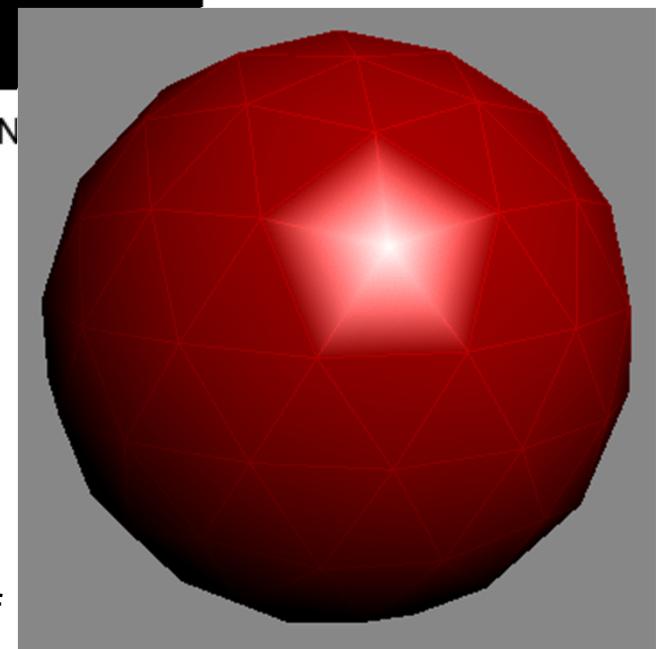


FLAT SHADING



PHONG SHADING

GOURAUD SHADING



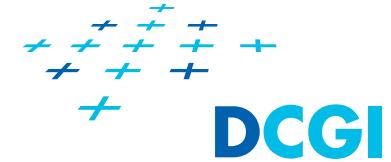
Obrázky převzaty z [Wikipedia]

Gouraud_low_anim.gif

Osnova

- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

Osvětlování v OpenGL



Moderní OpenGL podporuje unifikovaný způsob práce s barevnými a osvětlovacími modely

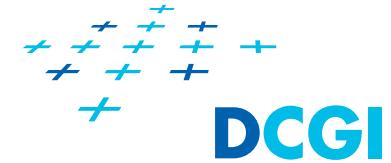
- normály pro jednotlivé vrcholy se předávají do vertex shaderu
 - barva se určí ve složkách RGB například pro každý vrchol
 - při výpočtu ve fragment shaderu lze spočítat cokoliv je potřeba
- Obecnou otázkou je rychlosť výpočtu, náročnost je potřeba omezit podle složitosti scény, počtu světel, osvětlovacího modelu atd.

Nastavení osvětlování, komponenty

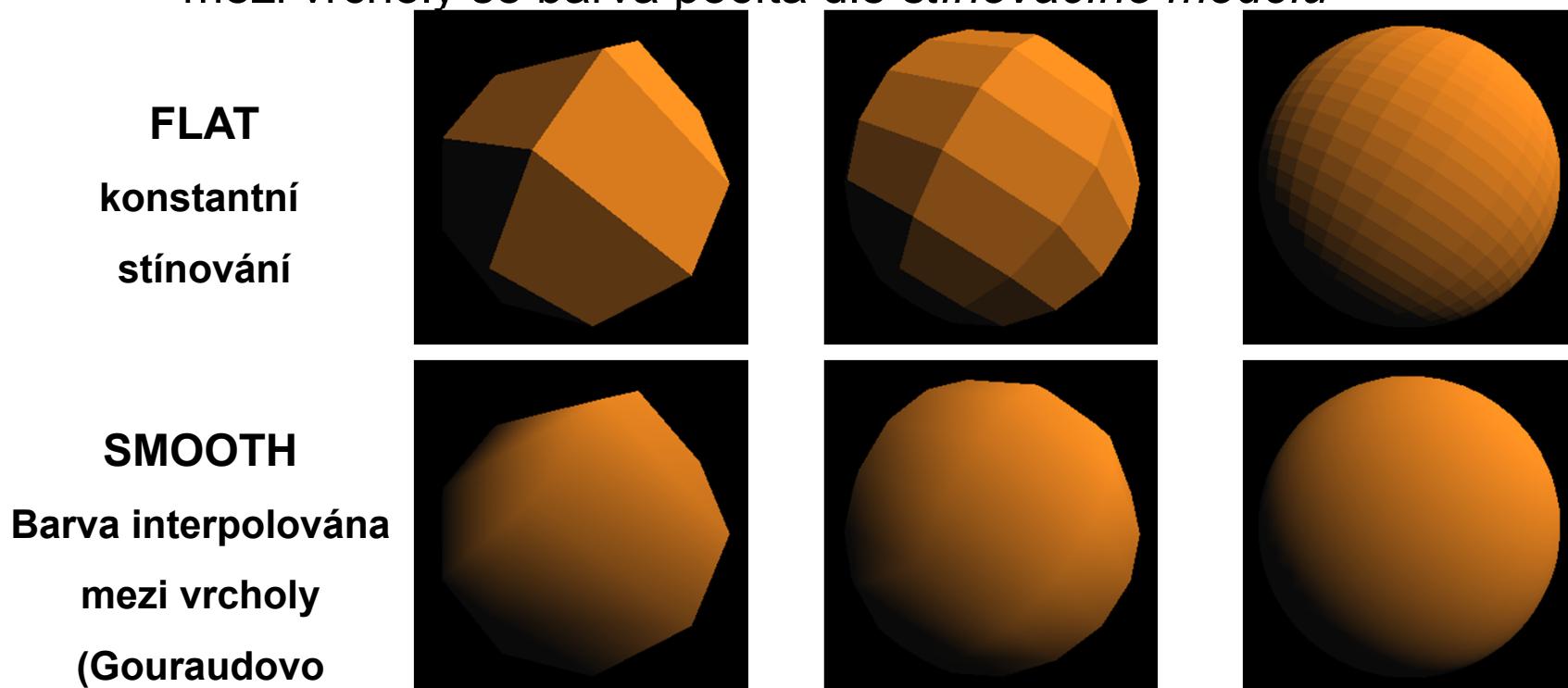
- poloha a vlastnosti světelných zdrojů
- vlastnosti materiálů
- normálové vektory ve vrcholech

⇒ barva objektu dána množstvím světla odraženého k pozorovateli

Osvětlování v OpenGL



- barva (osvětlení) se počítá ve vrcholech
- mezi vrcholy se barva počítá dle *stínovacího modelu*



- Při nízkém rozlišení vznikají artefakty na obrysových hranách
⇒ vyšší rozlišení modelu vede na kvalitnější osvětlení

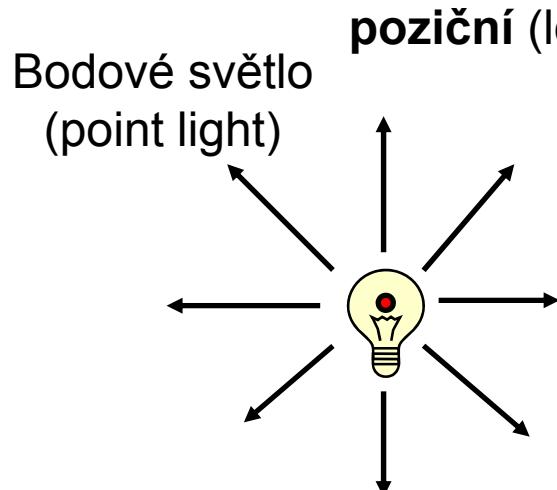
Osnova

- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

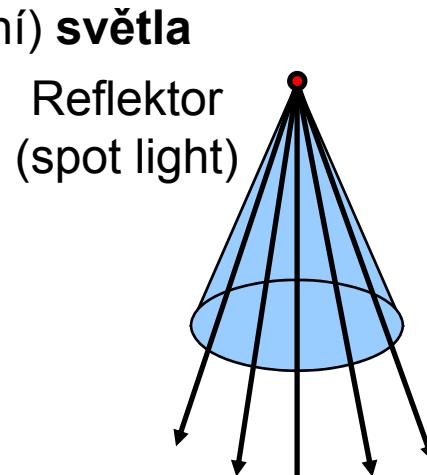
Světelné zdroje

- v OpenGL jsou světla implementována ve fragment shaderu
- Vypnutí a zapnutí, změna jasu světel lze pomocí dat předáváných dat
- **Nezobrazují se**, jen osvětlují okolní objekty !!!
- Každý vrchol osvětlen všemi zapnutými světly
⇒ OpenGL **nepočítá zastínění okolními objekty**
- Barva světla určena množstvím červené, zelené a modré
⇒ RGB intenzita

Typické světelné zdroje v OpenGL



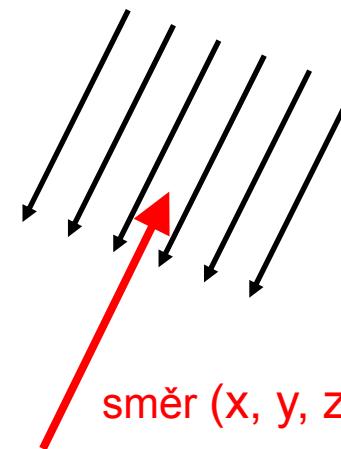
Bodové světlo
(point light)



poziční (lokální) světla

- pozice $[x, y, z]$

směrová (paralelní)
světla

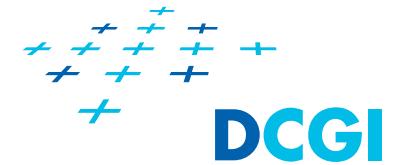


Světlo
umístěno v
nekonečné
vzdálenosti
od scény

typ světelného zdroje se určí parametrem **POSITION** = (x, y, z, w)

- směrové světlo $w = 0$
 (x, y, z) je vektor ukazující ke světlu
- poziční světlo $w <> 0$
 $[x, y, z]$ je pozice světla (v homogenních souřadnicích)

Světelné zdroje



V reálném prostředí klesá intensita světla se vzdáleností

Zeslabení je modelováno faktorem:

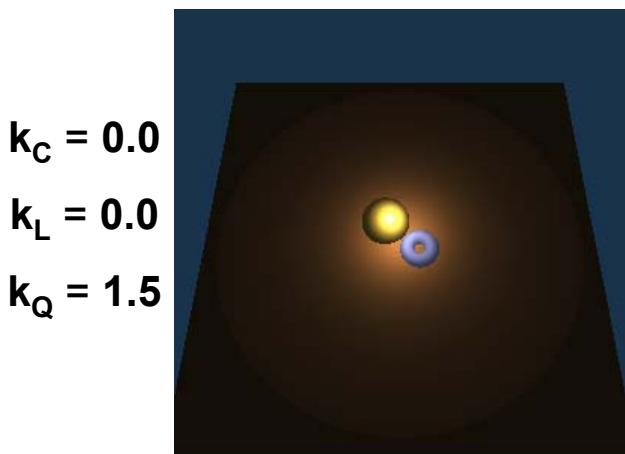
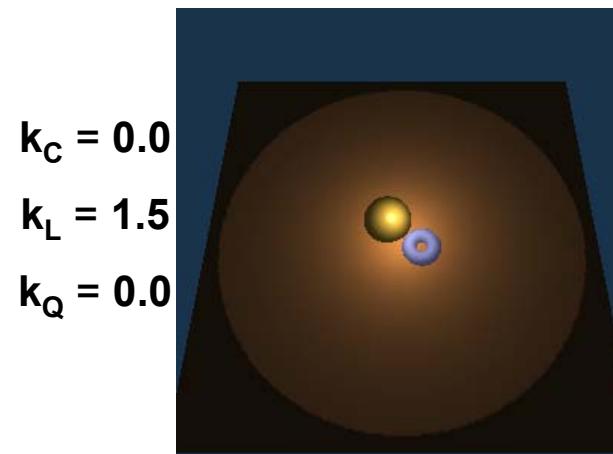
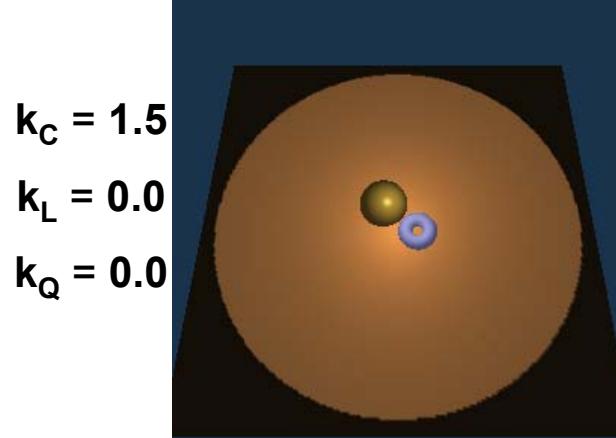
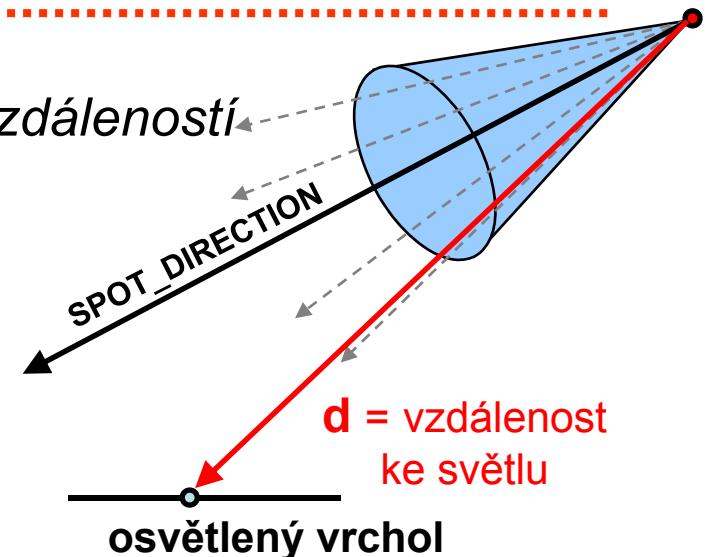
$$\text{attenuationFactor} = 1.0 / (k_C + k_L * d + k_Q * d^2)$$

d = vzdálenost mezi světlem a vrcholem

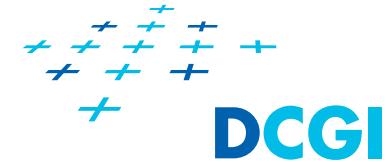
k_C = constant attenuation

k_L = linear attenuation

k_Q = quadratic attenuation

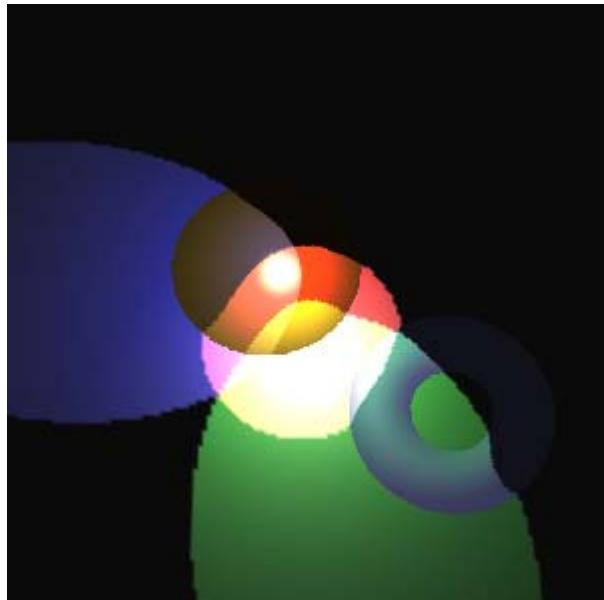


Světelné zdroje

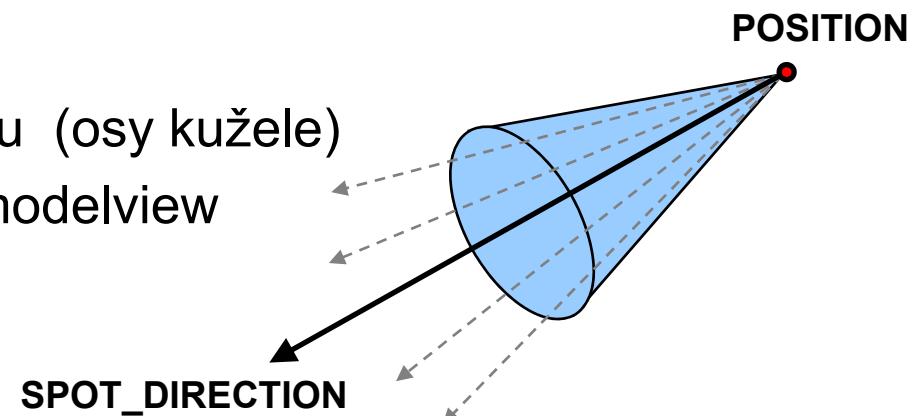
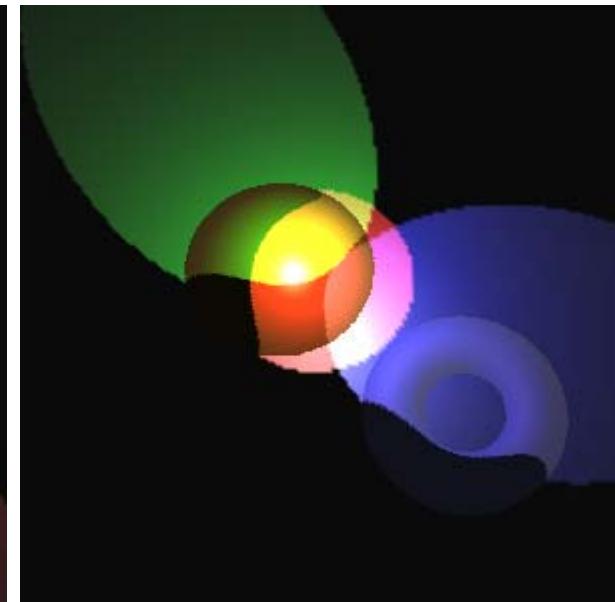
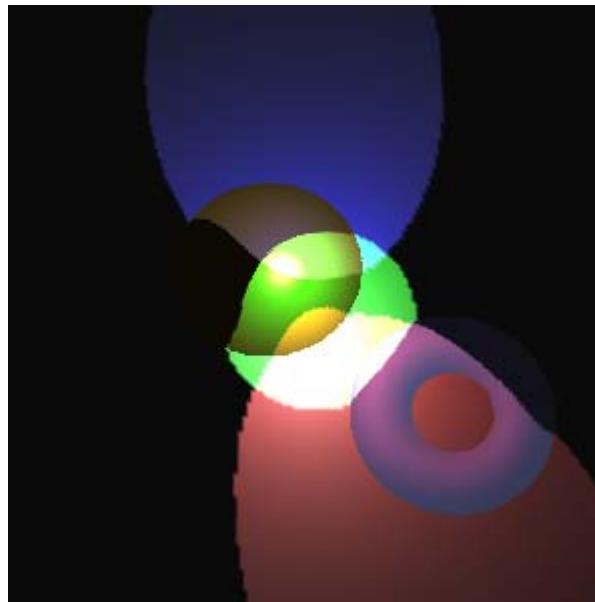


SPOT_DIRECTION

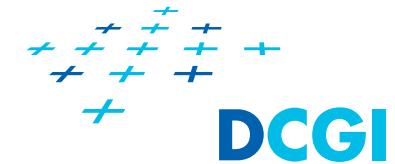
- definuje směr osy reflektoru (osy kuželeta)
- je transformována maticí modelview



*red, green,
and blue spot
lights*



Světelné zdroje

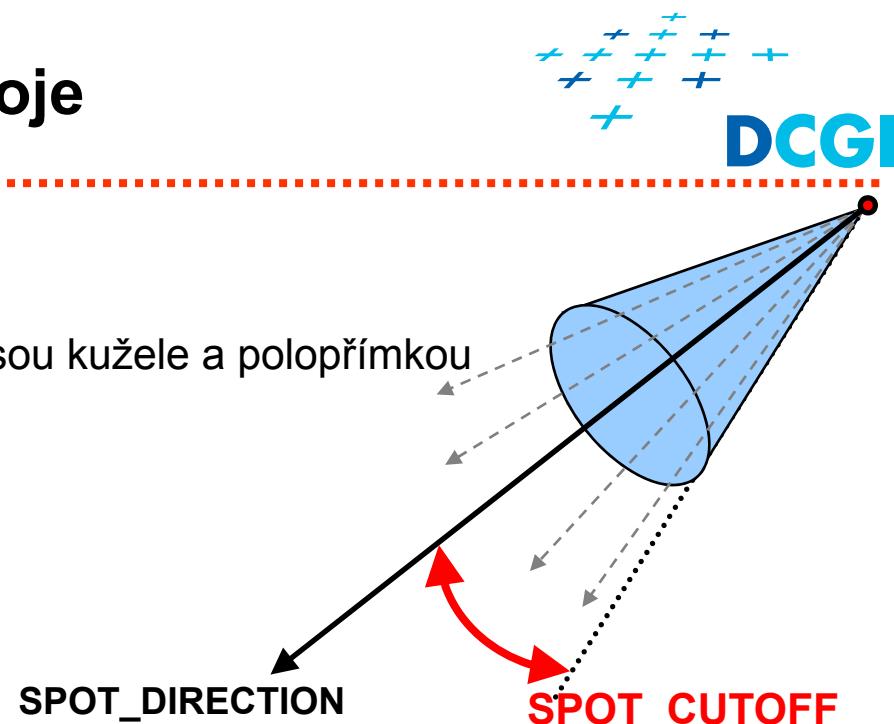


SPOT_CUTOFF

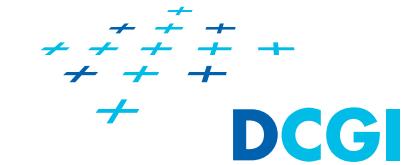
- Popisuje rozevření kužele (úhel mezi osou kužele a polopřímkou podél stěny kužele)

Povolené hodnoty pro **SPOT_CUTOFF**

- 180.0** *bodové světlo*
- $\langle 0.0, 90.0 \rangle$** *reflektor*



Světelné zdroje

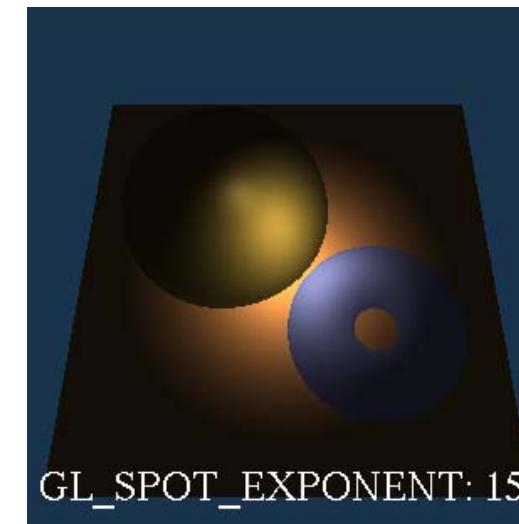
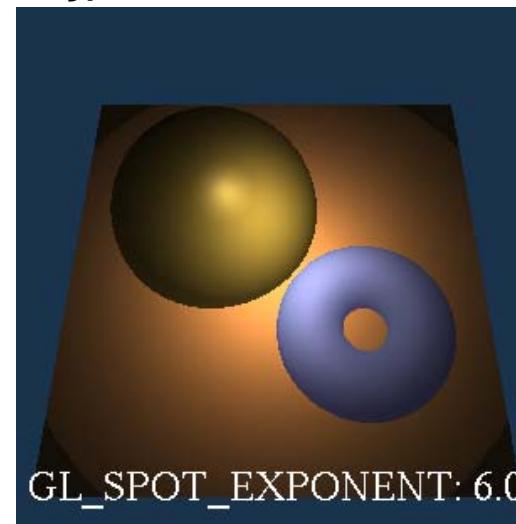
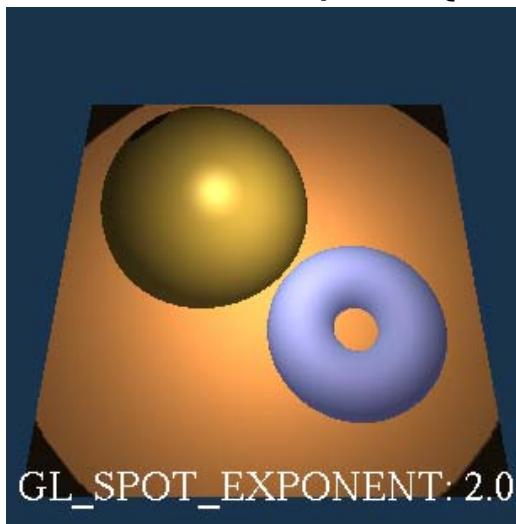
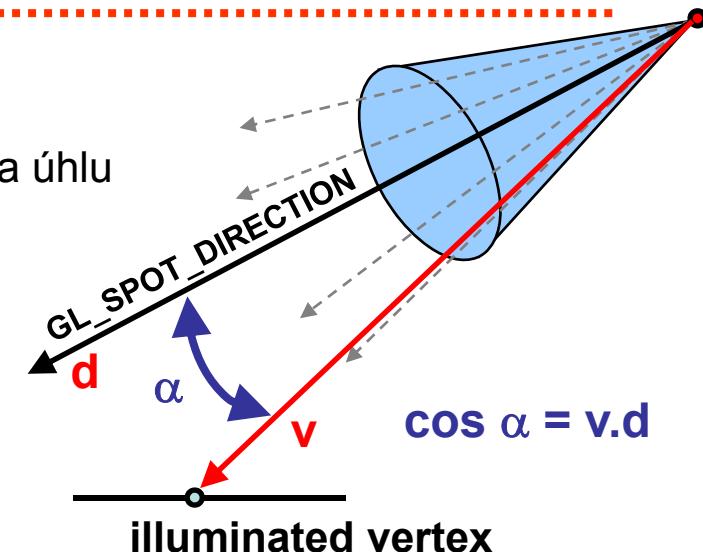


SPOT_EXPONENT

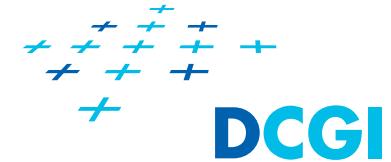
- Popisuje rozložení světla v rámci kuželeta v závislosti na úhlu od osy kuželeta (zaostření světla)
- Nejvyšší intenzita je ve středu kuželeta a klesá směrem ke stěnám

Efekt reflektoru se projeví následujícími hodnotami:

- Světelný paprsek leží mimo kužel **0.0**
- Světlo není reflektor **1.0**
- Jinak **(max {cos α, 0})^{SPOT_EXPONENT}**



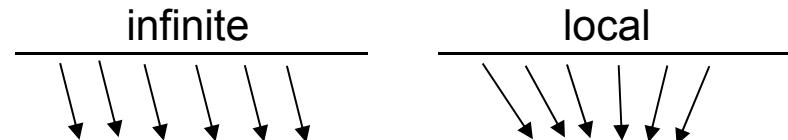
Globální nastavení



Viewpoint („poloha pozorovatele“) - Lokální a v nekonečnu

- poloha viewpointu ovlivňuje rychlosť výpočtů highlights vytvárené zrcadlovým odrazem
- **infinite viewpoint** (implicitný) – smér mezi pozorovatelem a libovolným vrcholom scény zůstává konstantní
- **local viewpoint** vede k realističtějším výsledkům, smér k pozorovateli se počítá pro každý vrchol, proto snížení celkového výkonu zobrazování

Změna na lokální viewpoint



- Umístí viewpoint do bodu $(0, 0, 0)$ v souřadnicích oka (approximace eye vektoru)

Definice světel, standardní notace a názvosloví



Jméno parametru	Iniciální hodnota	Význam
AMBIENT	(0.0,0.0,0.0,1.0)	ambientní složka barvy světla
DIFFUSE	(1.0,1.0,1.0,1.0)	difúzní složka barvy světla (LIGHT0!)
SPECULAR	(1.0,1.0,1.0,1.0)	zrcadlová složka barvy světla (LIGHT0!)
POSITION	(0.0,0.0,1.0,0.0)	(x, y, z, w) poloha světla
SPOT_DIRECTION	(0.0,0.0,-1.0)	(x, y, z) směr bodového světla
SPOT_EXPONENT	0.0	exponent bodového světla
SPOT_CUTOFF	180.0	hraniční úhel výřezu bodového světla
CONSTANT_ATTENUATION	1.0	konstantní faktor útlumu
LINEAR_ATTENUATION	0.0	lineární faktor útlumu
QUADRATIC_ATTENUATION	0.0	kvadratický faktor útlumu

Poznámka: parametry pro barvu jsou 4-složkové.

Osnova

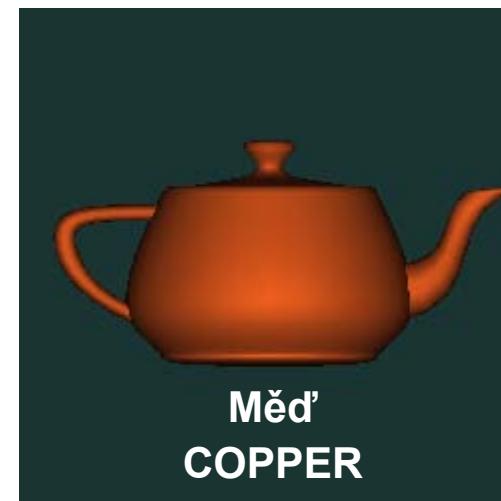
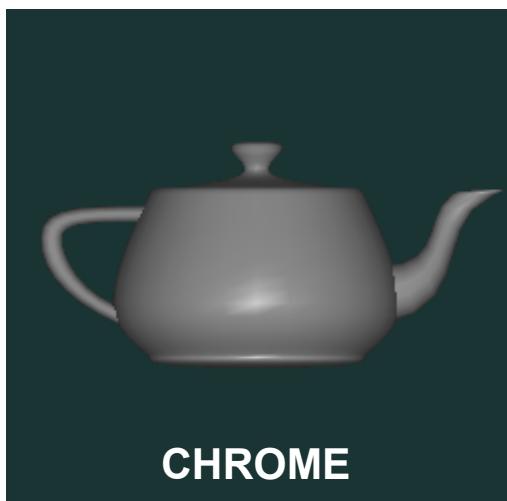
- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

Definice materiálů

Jméno parametru	Iniciální hodnota	Význam
AMBIENT	(0.2,0.2,0.2,1.0)	ambientní barva materiálu
DIFFUSE	(0.8,0.8,0.8,1.0)	difúzní barva materiálu
SPECULAR	(1.0,1.0,1.0,1.0)	zrcadlová složka materiálu
EMISSION	(0.0,0.0,0.0,1.0)	vyzařovací barva materiálu
SHININESS	0.0	lesklost materiálu, exponent pro zrcadlovou složku (čím vyšší hodnota, tím jasnější, ale menší (více zaostřený) „flek“)

- Materiál povrchu = procento odražené pro jednotlivé složky R,G,B

Material examples



Osnova

- Oko a vnímání barev (percepční funkce lidského oka)
- Generování barev na monitoru
- Výpočet osvětlení v bodě scény
- Stínování ploch (konstantní, Gouraud, Phong)
- Osvětlování v OpenGL
 - Světla (poloha, směrové, bodové, reflektor,...)
 - Materiály
 - Normály

Normálové vektory

- slouží ke stanovení orientace objektů (plošek) vůči světlům
- jsou to vektory kolmé k povrchu o délce 1 (musí se normalizovat)
- normalizace normály (normálového vektoru)

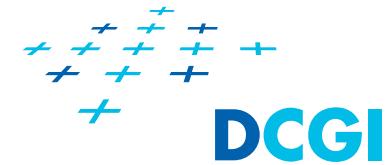
$$\|\vec{n}\| = \frac{\vec{n}}{|\vec{n}|}$$

- normálu je třeba přiřadit každému **vrcholu**
- rovný povrch má normálu všude stejnou,
zakřivený povrch obecně různou (normála se naklání)
- POZOR!
**Normály se nijak automaticky nepočítají (zdržovalo by to)
Proto se musejí „dodávat“ s vrcholy**

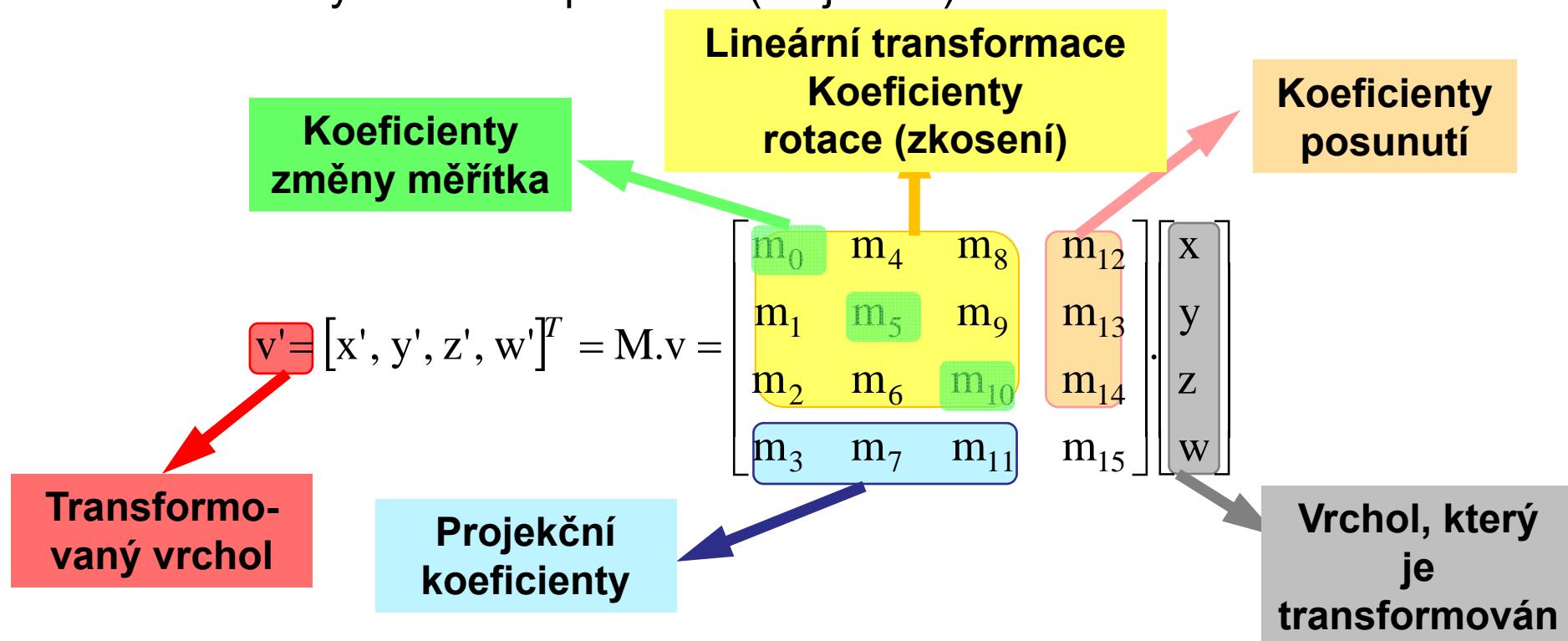
Normálové vektory musí být normalizované

- Proč musejí mít normály délku 1 ?
 - Používají se ve skalárních součinech nahrazujících cos (muselo by se stejně dělit jejich délkou)
- Proč nestačí zadávat jednotkové vektory?
 - Protože se mění jejich délka **transformacemi** a při **interpolaci**
- Které transformace **nemění délku normál**?
 - Tzv. **rigidní transformace**, tj. ty, které mají ortogonální matici (řádky tvoří na sebe kolmé jednotkové vektory)
 - Typickým příkladem jsou **rotace**
 - Normály jsou vektory – proto na ně nemá vliv **posunutí**

Součásti transformační matice

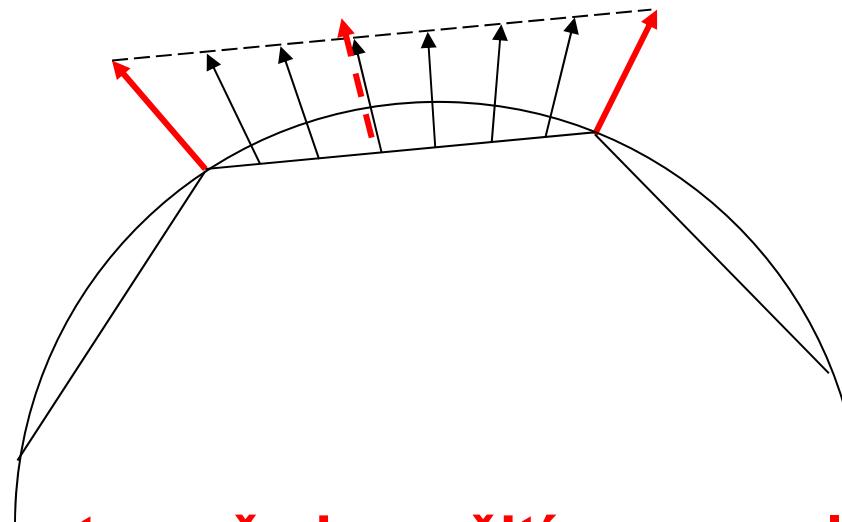


- vrcholy jsou reprezentovány **sloupcovými vektory** $v = [x, y, z, w]^T$
- **transformace** jsou zadány 4×4 maticí M
- transformace se provede vynásobením matice M a vektorem v .
- na vektory nemá vliv posunutí (mají $w=0$)



Zkrácení normál při interpolaci

- Při interpolaci mezi vrcholy při rasterizaci
- Interpolované vektory jsou kratší

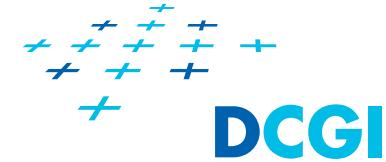


⇒ normály nutno před použitím **normalizovat**

V shaderu:

```
N = normalize (interpolatedNormal);  
N = normalize(Matrix * interpolatedNormal);
```

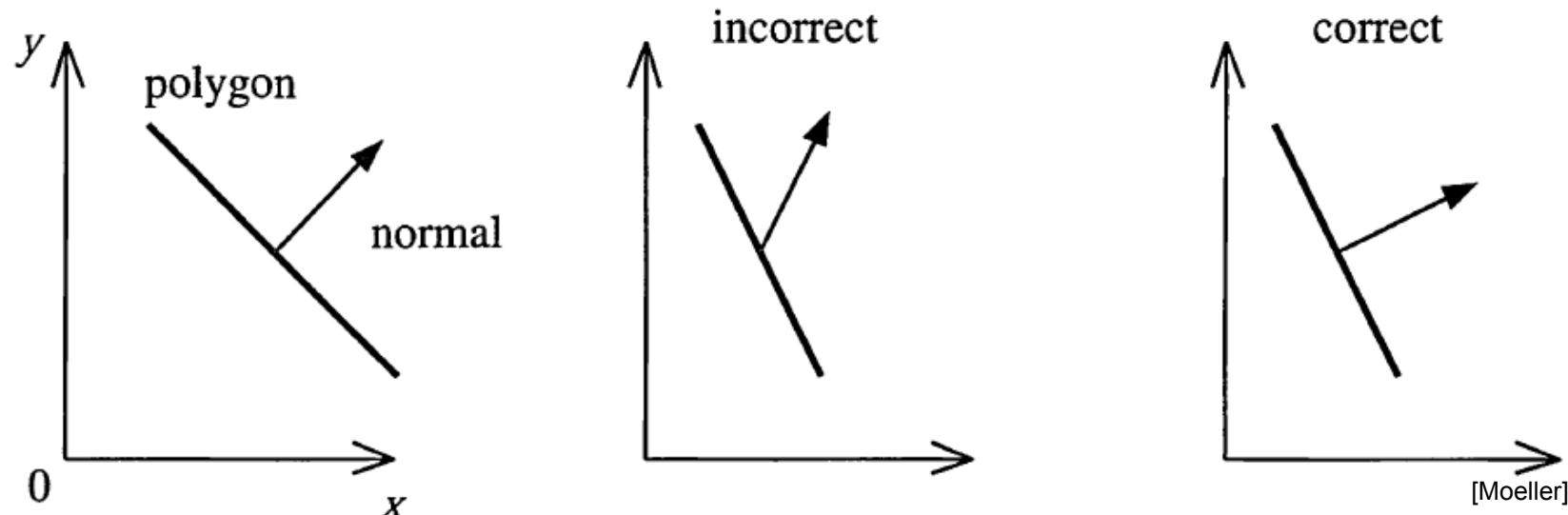
Změna délky normály při změně měřítka



- Při změně měřítka ve všech směrech stejně k -krát se k -krát prodlouží i normála
 - Stačí vydělit složky normály hodnotou k
 - Není třeba počítat celý vzorec pro normalizaci

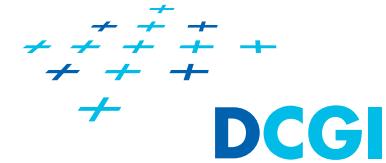
Změna sklonu normály

- Nesymetrická změna měřítka, či jiná **nerigidní affinní modelovací transformace M**
- Transformovaná normála **přestane být kolmá k povrchu** (vektor **Mn**, o kterém si myslíme, že je to normála, už normálou není) => pro normály je matice **N = (M⁻¹)^T**



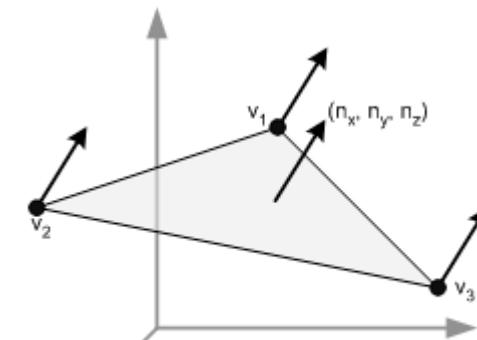
scaled by 0.5 along the x dimension
PGR

Odvození transformace pro normály 1



- Vrchol x je transformován maticí M násobením touto maticí:
$$x' = M x \quad \text{matici } M \text{ známe} - \text{je to modelovací matice}$$
- Stejně je transformována tečna t (rozdíl dvou povrchových vrcholů)
$$t' = M t$$
- Normála n je transformována maticí N tak:
$$n' = N n \quad \text{matici } N \text{ hledáme}$$
- Předpoklad: normála n je kolmá na tečnu t před transformací i po transformaci geometrického objektu maticí M :
před $n \cdot t = 0$ - zapsáno jako skalární součin
$$n^T t = 0 \quad \text{- dílto jako vektor } n \text{ krát } t$$

po $n' \cdot t' = 0$



[http://www.songho.ca/opengl/gl_normaltransform.html]

Odvození transformace pro normály 2

Hledáme neznámou matici \mathbf{N}

- Vyjdeme z rovnice: $\mathbf{n}^T \mathbf{t} = 0$ Trik: $\mathbf{M}^{-1} \mathbf{M} = \mathbf{I}$

$$\Rightarrow \mathbf{n}^T \mathbf{M}^{-1} \mathbf{M} \mathbf{t} = 0$$

$$\Rightarrow (\mathbf{n}^T \mathbf{M}^{-1}) (\mathbf{M} \mathbf{t}) = 0$$

- Transformace vektoru \mathbf{t} maticí \mathbf{M} je tangenta

$$\mathbf{t}' = \mathbf{M} \mathbf{t}$$

$$\Rightarrow (\mathbf{n}^T \mathbf{M}^{-1}) \mathbf{t}' = 0$$

- Transformovaná normála a tangenta jsou na sebe kolmé.

$$\mathbf{n}'^T \mathbf{t}' = 0$$

$$\Rightarrow \mathbf{n}'^T = (\mathbf{n}^T \mathbf{M}^{-1})$$

\Rightarrow Algebraickou úpravou $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$ dostaneme

$$\mathbf{n}' = (\mathbf{n}^T \mathbf{M}^{-1})^T = (\mathbf{M}^{-1})^T \mathbf{n} = \mathbf{N} \mathbf{n}$$

\Rightarrow transformační matice \mathbf{N} pro normálu je rovna

$$\mathbf{N} = (\mathbf{M}^{-1})^T$$

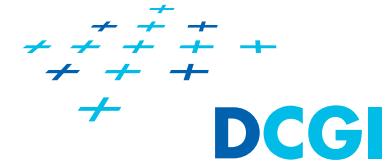
$$(n_x \ n_y \ n_z \ n_w) \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = 0$$

$$(n_x \ n_y \ n_z \ n_w) \underbrace{M^{-1} M}_{\text{normal}^T} \underbrace{\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}}_{\text{vertex}} = 0$$

$$\begin{pmatrix} nx_{eye} \\ ny_{eye} \\ nz_{eye} \\ nw_{eye} \end{pmatrix} = (nx_{obj} \ ny_{obj} \ nz_{obj} \ nw_{obj}) M^{-1}$$

$$\begin{pmatrix} nx_{eye} \\ ny_{eye} \\ nz_{eye} \\ nw_{eye} \end{pmatrix} = (M^{-1})^T \begin{pmatrix} nx_{obj} \\ ny_{obj} \\ nz_{obj} \\ nw_{obj} \end{pmatrix}$$

Výpočet inverzní matice



- Pro sekvenci jednoduchých transformací
 - Invertujeme dosazením invertovaných tr. a opačným pořadím např. pro $\mathbf{M} = \mathbf{T}(t)\mathbf{R}(\alpha)$ je $\mathbf{M}^{-1} = \mathbf{R}(-\alpha)\mathbf{T}(-t)$
- Pokud je \mathbf{M} ortogonální – např. sada rotací
 - je $\mathbf{M}^{-1} = \mathbf{M}^T$
- Je-li transformace neznámá, vypočteme inverzní matici
 - Pro vektory stačí 3x3
 - Není třeba dělit determinantem původní matice
 - Stačí matice algebraických doplňků (*adjoint method*)
 - A tu následně normalizovat vypočtenou normálu

Inverzní matice pomocí algebraických doplňků



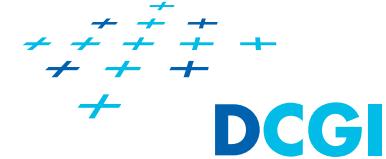
$$\mathbf{M}_1 = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix}$$

$$\text{adj } (\mathbf{M}_1) = \begin{bmatrix} m_{22}m_{11} - m_{12}m_{21} & m_{02}m_{21} - m_{22}m_{01} & m_{12}m_{01} - m_{02}m_{11} \\ m_{12}m_{20} - m_{10}m_{22} & m_{22}m_{00} - m_{02}m_{20} & m_{10}m_{02} - m_{12}m_{00} \\ m_{10}m_{21} - m_{20}m_{11} & m_{20}m_{01} - m_{00}m_{21} & m_{00}m_{11} - m_{10}m_{01} \end{bmatrix}$$

Transponovaná

$$\mathbf{N} = \begin{bmatrix} m_{22}m_{11} - m_{12}m_{21} & m_{12}m_{20} - m_{10}m_{22} & m_{10}m_{21} - m_{20}m_{11} \\ m_{02}m_{21} - m_{22}m_{01} & m_{22}m_{00} - m_{02}m_{20} & m_{20}m_{01} - m_{00}m_{21} \\ m_{12}m_{01} - m_{02}m_{11} & \boxed{m_{10}m_{02} - m_{12}m_{00}} & m_{00}m_{11} - m_{10}m_{01} \end{bmatrix}$$

[Turkowski - <http://203.252.22.28/Tutor%28KyungKi%29/NormalTransformations.pdf>]



Příklad v OpenGL – část na CPU

```
GLint MeshNode::m_norLoc = -1;  
m_normalLoc = glGetUniformLocation(m_program, "normal");  
glBindVertexArray( vao );      // inicializace  
// position, color  
...  
// normal  
glEnableVertexAttribArray( m_normalLoc );  
glBindBuffer(GL_ARRAY_BUFFER, normalBufferID);  
glVertexAttribPointer(m_normalLoc, 3, GL_FLOAT, GL_FALSE,  
                     0, 0);  
glBindBuffer(GL_ARRAY_BUFFER, 0 );  
glBindVertexArray( 0 );  
  
glBindVertexArray( vao ); // kreslení  
glDrawArrays(GL_TRIANGLES, 0, m_nVertices);  
glBindVertexArray( 0 );
```

Příklad v OpenGL – Vertex Shader

```
// Vertex shader for meshes of class MeshNode
#version 130
uniform mat4 VMmatrix;
uniform mat4 Pmatrix;
in vec3 position;
in vec4 color;
in vec3 normal;
smooth out vec3 thePosition; // camera space coordinates
smooth out vec4 theColor;
smooth out vec3 theNormal; // camera space normal

void main()
{
    vec4 pos_v = VMmatrix * vec4(position, 1);
    theColor = color;
    theNormal = (VMmatrix * vec4(normal, 0)).xyz; // pokud není nutno
                                                    // počítat (M^(-1))^T
    gl_Position = Pmatrix * pos_v;
}
```



Příklad v OpenGL – Fragment shader

```
// Fragment shader for meshes of class MeshNode
#version 130

smooth in vec3 thePosition;
smooth in vec4 theColor;
smooth in vec3 theNormal;
uniform vec3 pointlightPos;
out vec4 outputColor;

void main()
{
    vec3 L = normalize(pointlightPos - thePosition);
    outputColor = theColor * dot(L, normalize(theNormal));
}
```