Barva, světlo, materiály v počítačové grafice

Petr Felkel

Katedra počítačové grafiky a interakce, ČVUT FEL

místnost KN:E-413 (Karlovo náměstí, budova E)

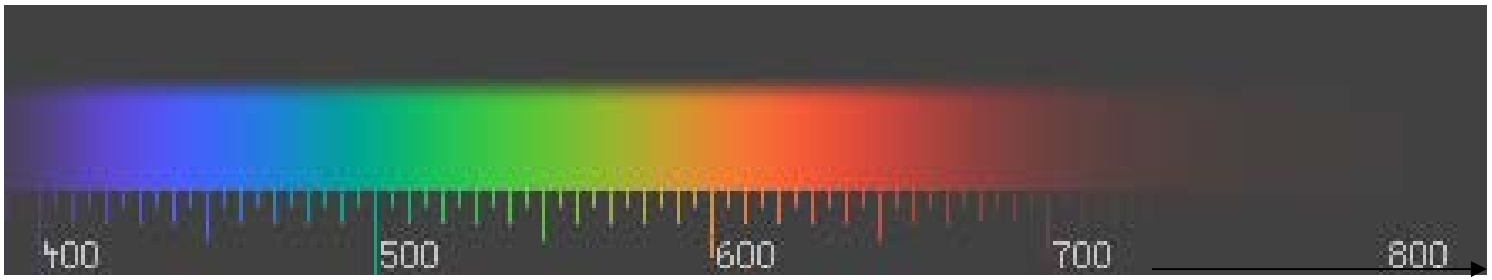
E-mail: felkel@fel.cvut.cz

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

Poslední změna: 16.6.2016

Viditelné světlo

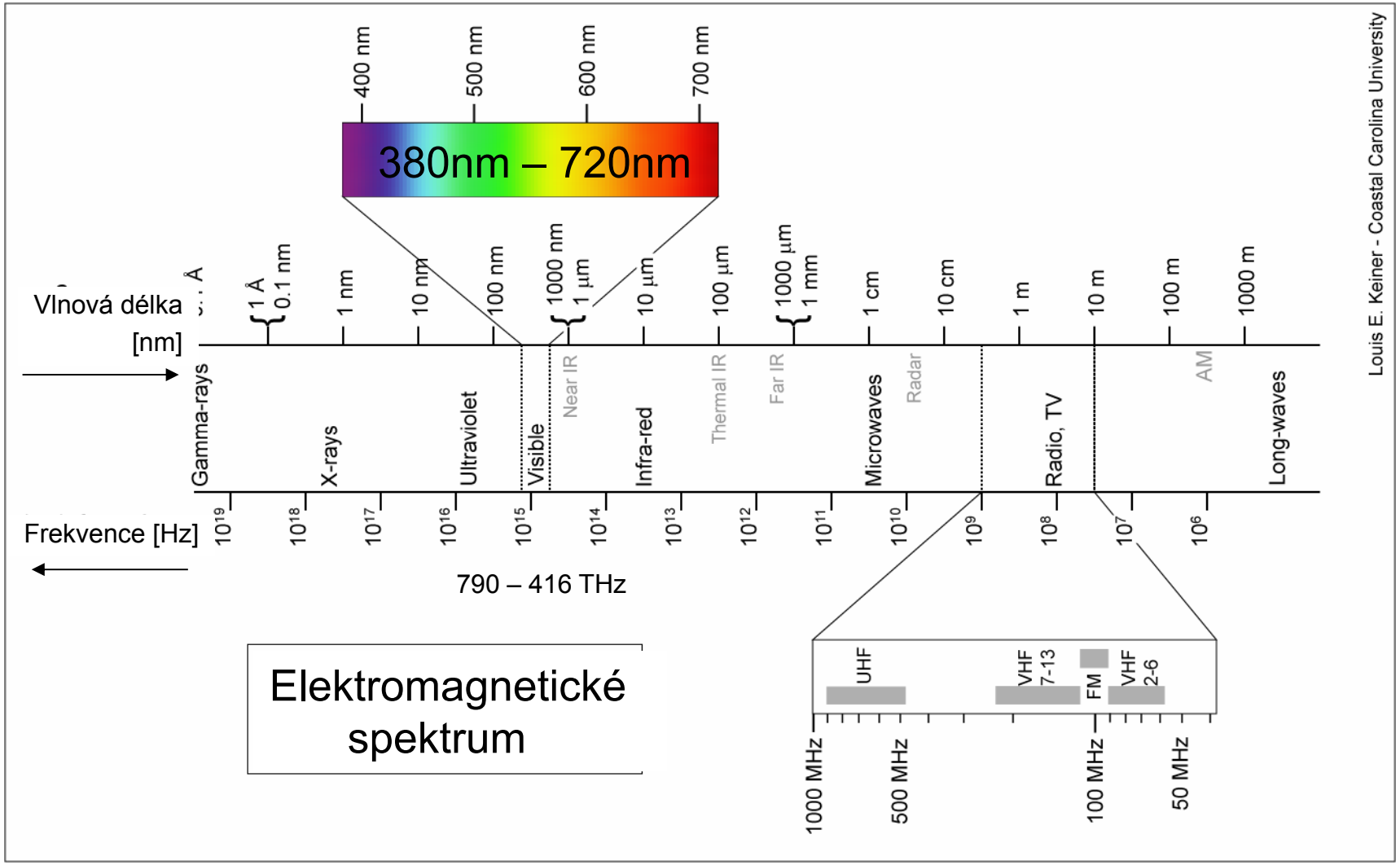
- Elektromagnetické záření na něž jsou citlivé buňky sítnice
  cca 380nm – 720nm (790 – 416 THz)



Vlnová délka

- Světlo je *fyzikální jev*
- Pro popis světla lze použít fyziku
- „Toto světlo je směsí těchto vlnových délek…“

Viditelné světlo

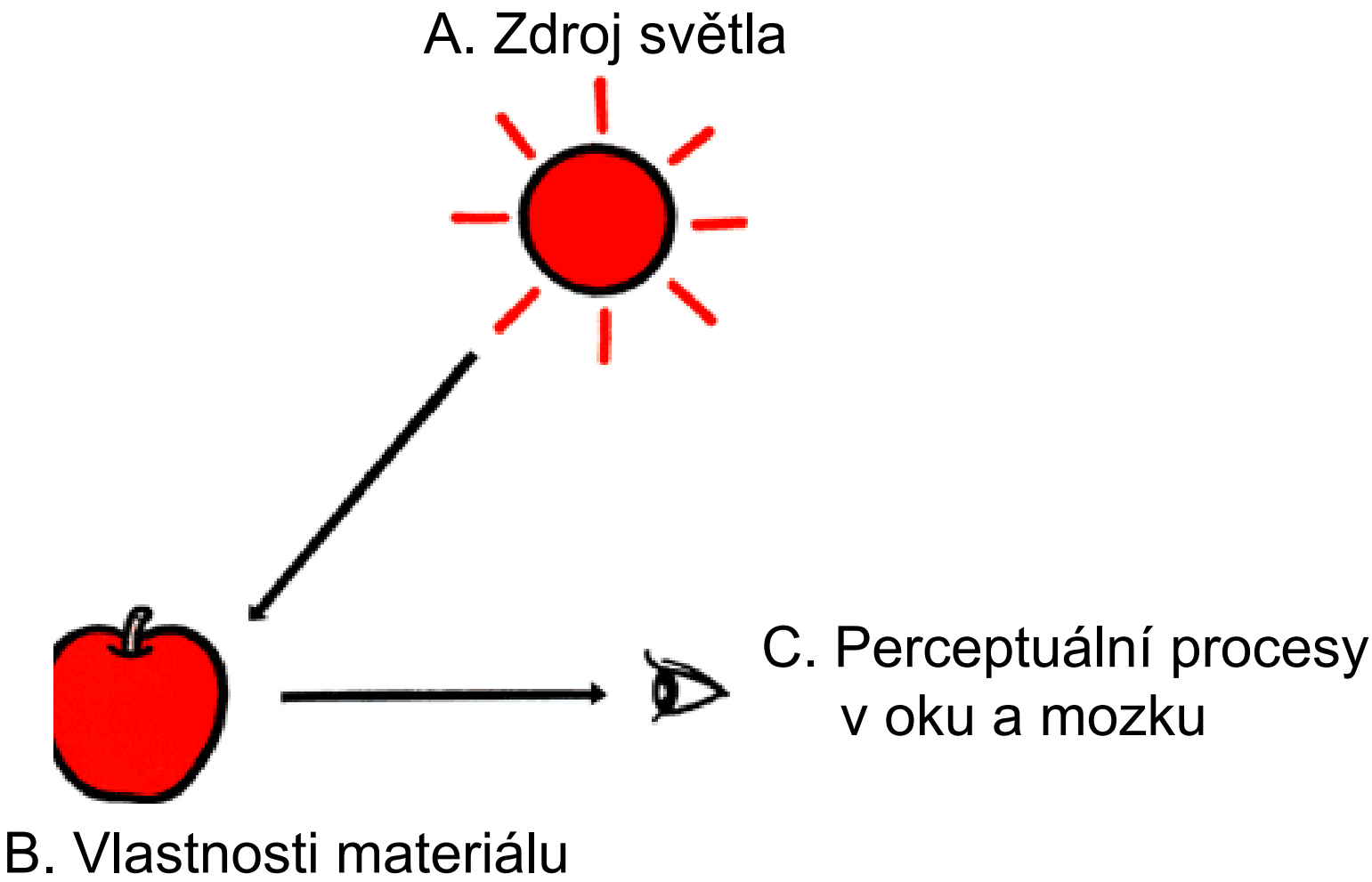


Barva

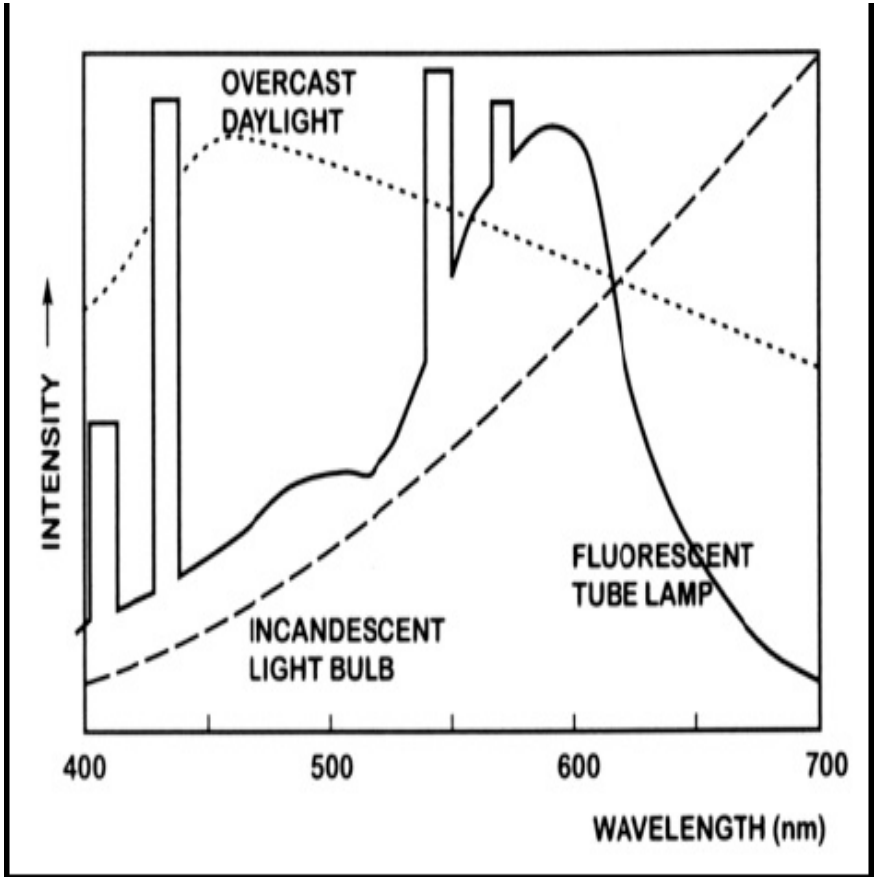
= Vjem

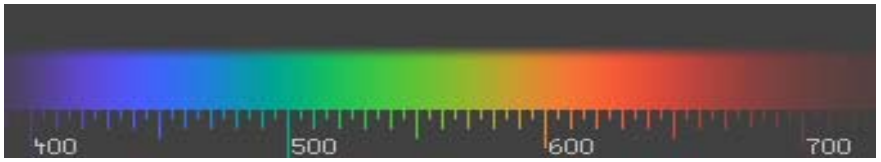
- Nekoresponduje přímo (1 : 1) s fyzikální skutečností
- Pro popis barvy je třeba vycházet z psychologie (experimentální psychologie, psychofyziky)
- Zahrnuje vlastnosti oka a vědomé zpracování mozkem
- Odezva oka je logaritmická
- Člověk vnímá i značně odlišná spektra kmitočtů jako stejnou barvu
- „Toto světlo vnímám jako světle zelené“ -> Metamerismus
- Vliv mají okolní barvy, kontext

Čím je určen barevný vjem



Čím je určen barevný vjem



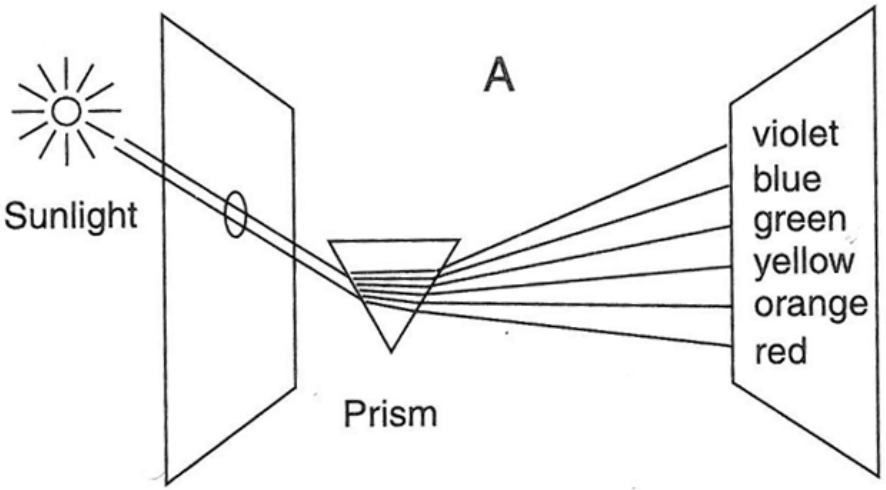


A. Zdroj světla

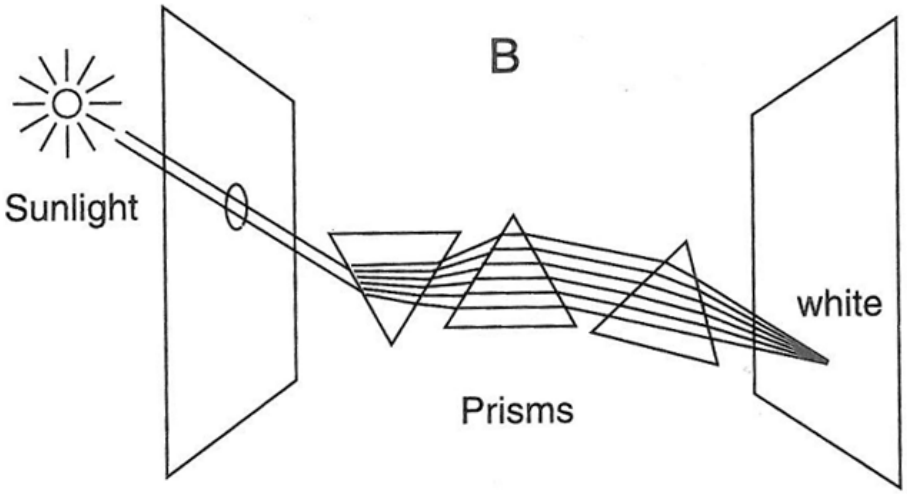
- Hustotu světelného toku v závislosti na vlnové délce λ vyjadřuje ***spektrální radiance***, ***L*(λ)**
- Pozn.: Oficiální, normou přijatý, český název radiance, používaný v osvětlovacím inženýrství, je „*zář*“, v grafice se však používá slovo radiance

www.konicaminolta.com

Rozklad denního světla (Newton 1666)



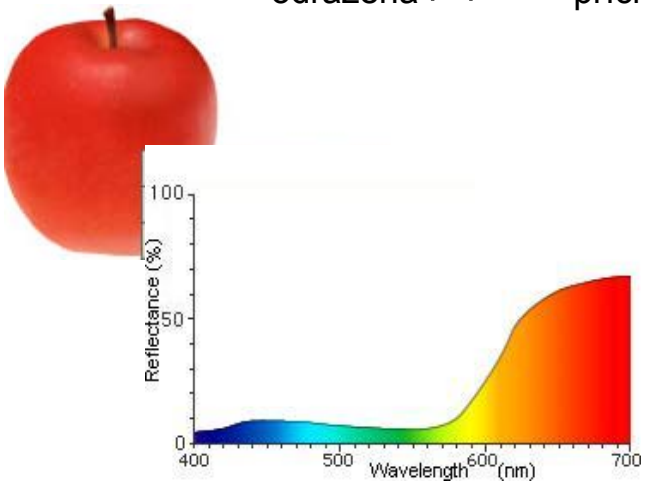
Neexistuje samostatné bílé světlo, ale je to směs frekvencí

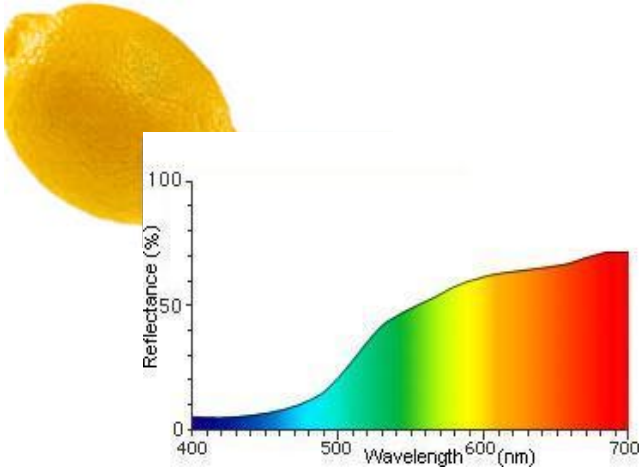


Čím je určen barevný vjem

B. Vlastnosti materiálu pozorovaného objektu

- ***Spektrální odrazivost*** $\rho(\lambda)$
  Radiance $L_{\text{odražená}}(\lambda) = L_{\text{příchozí}}(\lambda)\ \rho(\lambda)$





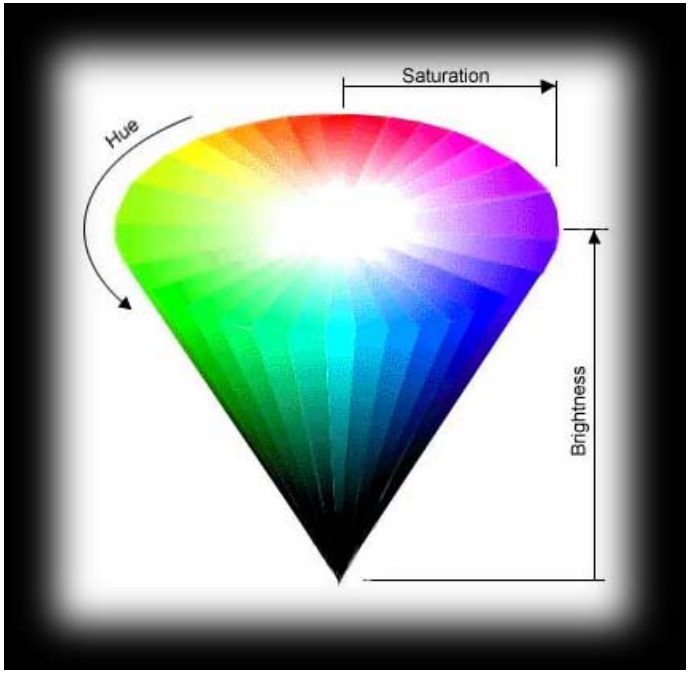
www.konicaminolta.com

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

Čím je určen barevný vjem

C. Perceptuální procesy v oku a mozku

- Radiance $L(\lambda)$ vstupující do oka (tj. spojitá funkce vlnové délky) je „dekódována“ (okem a mozkem) na **vědomý vjem**, jež popisujeme v dimenzích:



- **Barevný odstín** (Hue) – čistá barva bez příměsi bílého či černého pigmentu
- **Sytost** barvy (Saturation) – jak je barva čirá (užší spektrum)
- **Světlost, jas** (Brightness, Lightness, Intensity) – jak je barva světlá
- Viz barevné modely v (MGA)

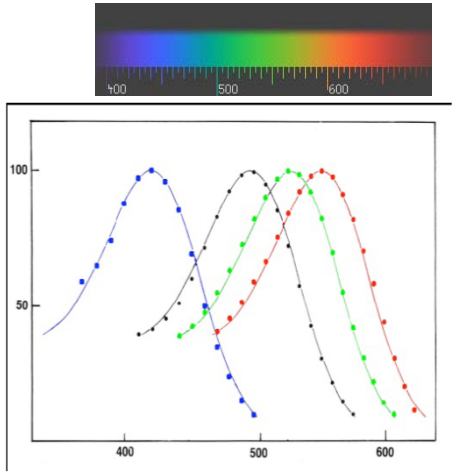
Percepční funkce lidského OKA

Na sítnici jsou dva druhy světlocitlivých buněk:

- Tyčinky (*rods*)
  - Jsou aktivovány při nízkých intenzitách světla – noční vidění
  - Při denním světle jsou saturovány – nedodávají žádný signál
  - Pouze jeden druh tyčinek – černobílé vidění

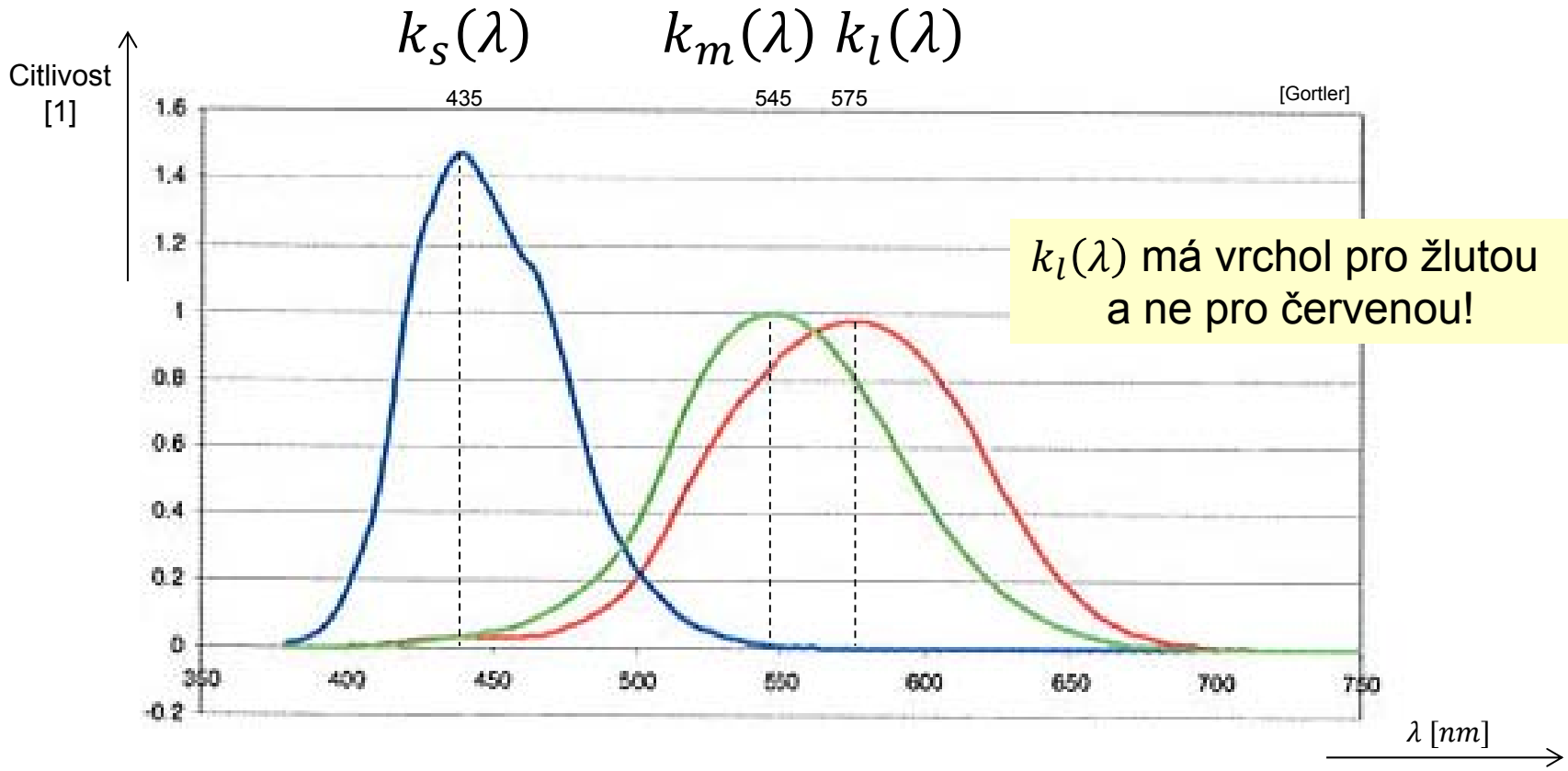
    => Za tmy je obtížné až nemožné rozeznávat barvy
    => Důležité pro návrh aplikací používaných za šera
- Čípky (*cones*)
  - 3 druhy reagující různě citlivě na světlo o různých vlnových délkách

    S / **M** / **L** … short / medium / long wavelength (krátké, střední a dlouhé vlnové délky)
  - Zprostředkovávají barevné vidění



Relativní citlivost čípků na pro různé frekvence monochromatického světla

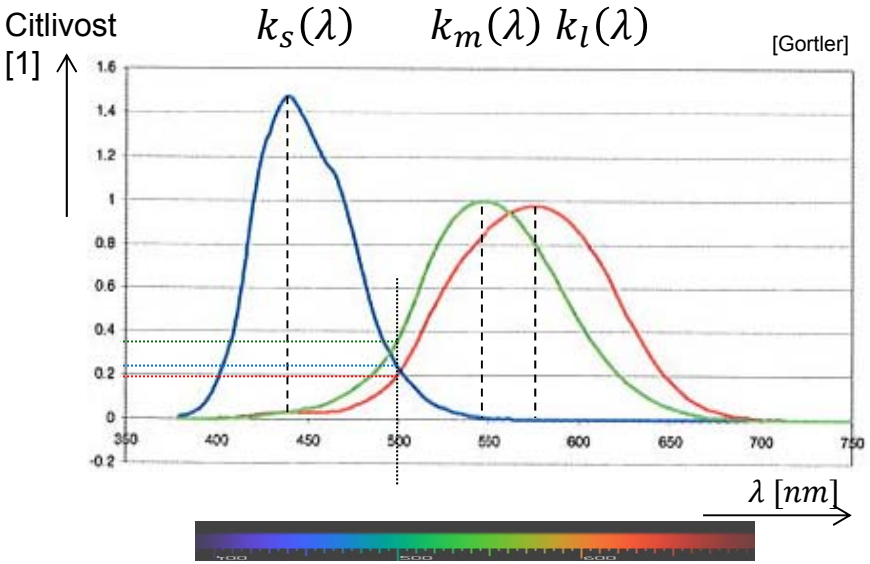
Luminosity



$k_l(\lambda)$ má vrchol pro žlutou a ne pro červenou!

2% 32% 64%

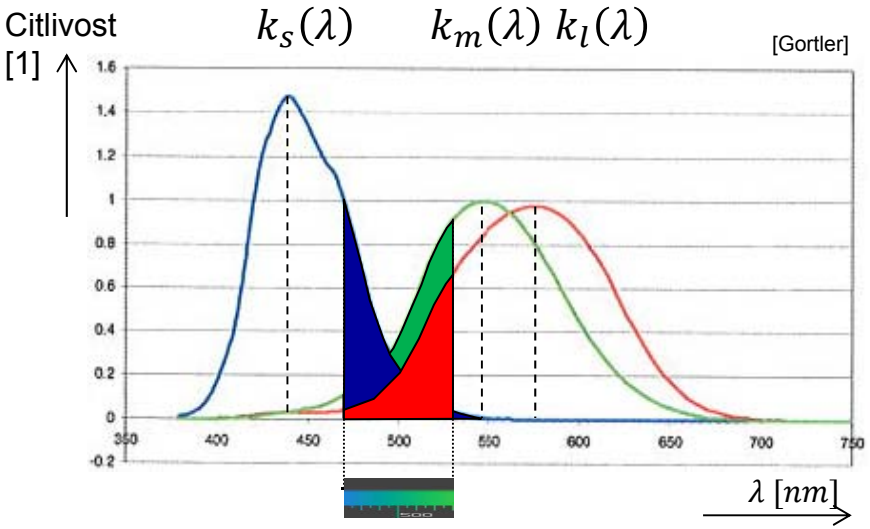
Odezva čípků na monochromatické světlo



- Monochromatické světlo I_λ
  - Má jednu vlnovou délku λ
  - Je vnímáno jako jednotkách iradiance (dopadající radiance)
  - Reagují na něj tyčinky i čípky
  - $k_s(\lambda)$ je příklad odezvy čípku **S** na monochromatické světlo o vlnové délce λ
  - Př.: Odezva čípků na světlo o vlnové délce $\lambda = 500\text{nm}$:

    $k_s = 0.35$
    $k_m = 0.25$
    $k_l = 0.19$

Odezva čípků na smíšené světlo



Zjednodušený příklad pro
$I(\lambda) = const.$
Pro různá $I(\lambda)$ se integrují součiny $I(\lambda)k_m(\lambda)$

- Smíšené světlo $I(\lambda)$
  - Obsahuje spektrum vlnových délek
  - V omezeném rozsahu intenzit reagují čípky lineárně na světlo
  - Odezva čípků je integrací odezev na jednotlivé vlnové délky

$$S = \int_{380}^{770} I(\lambda)k_s(\lambda)\, d\lambda$$

$$M = \int_{380}^{770} I(\lambda)k_m(\lambda)\, d\lambda$$

$$L = \int_{380}^{770} I(\lambda)k_l(\lambda)\, d\lambda$$

  - Odezvy **L, M, S** lze reprezentovat jako bod v prostoru LMS

Metamerismus

= jev, kdy se **barva** dvou vzorků pozorovaných

- pod určitým světlem jeví stejná, přičemž
- pod světlem s jinou spektrální charakteristikou se jeví rozdílná

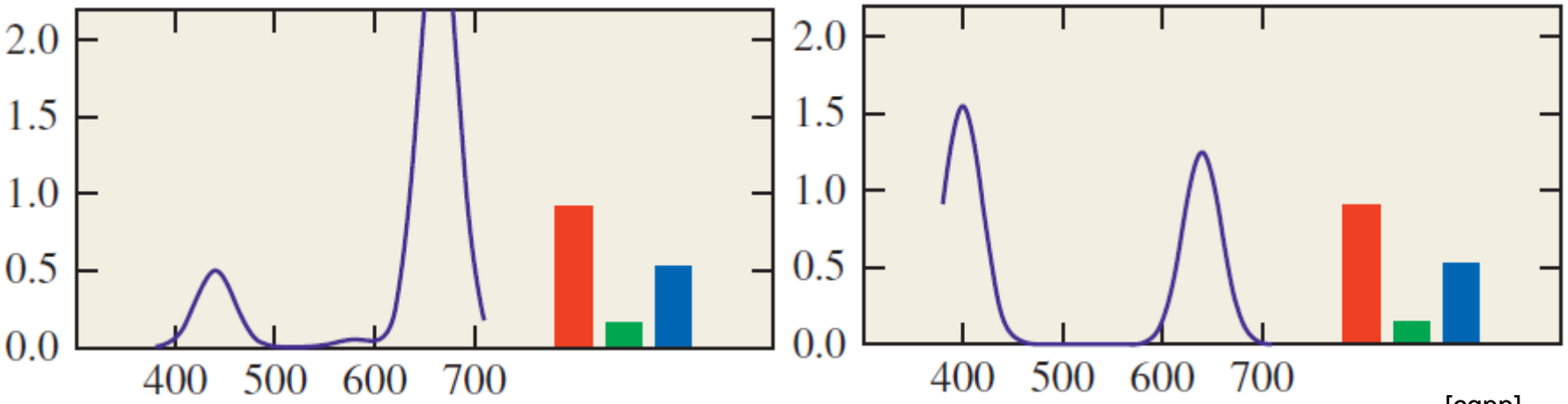
- Různé materiály vnímány jako stejně barevné
  - díky metamerismu lze namíchat „stejné“ barvy s jiným rozpouštědlem (vodové, acetonové, …), tisknout barevně,…
  - vjem barvy záleží na spektru světla, které objekt osvětluje!

Metamerismus

- Standardizované iluminanty
  - Referenční podmínky pro porovnávání „barev“ materiálů
  - D50 – světlo 2 hod. po východu slunce v Evropě, 5003K (DTP a tisk)
  - D65 – denní světlo v Evropě v zimě v poledne, 6500K
  - A – žárovka 2 854K
  - B - střední denní sv. 4800K

Metamerismus světel

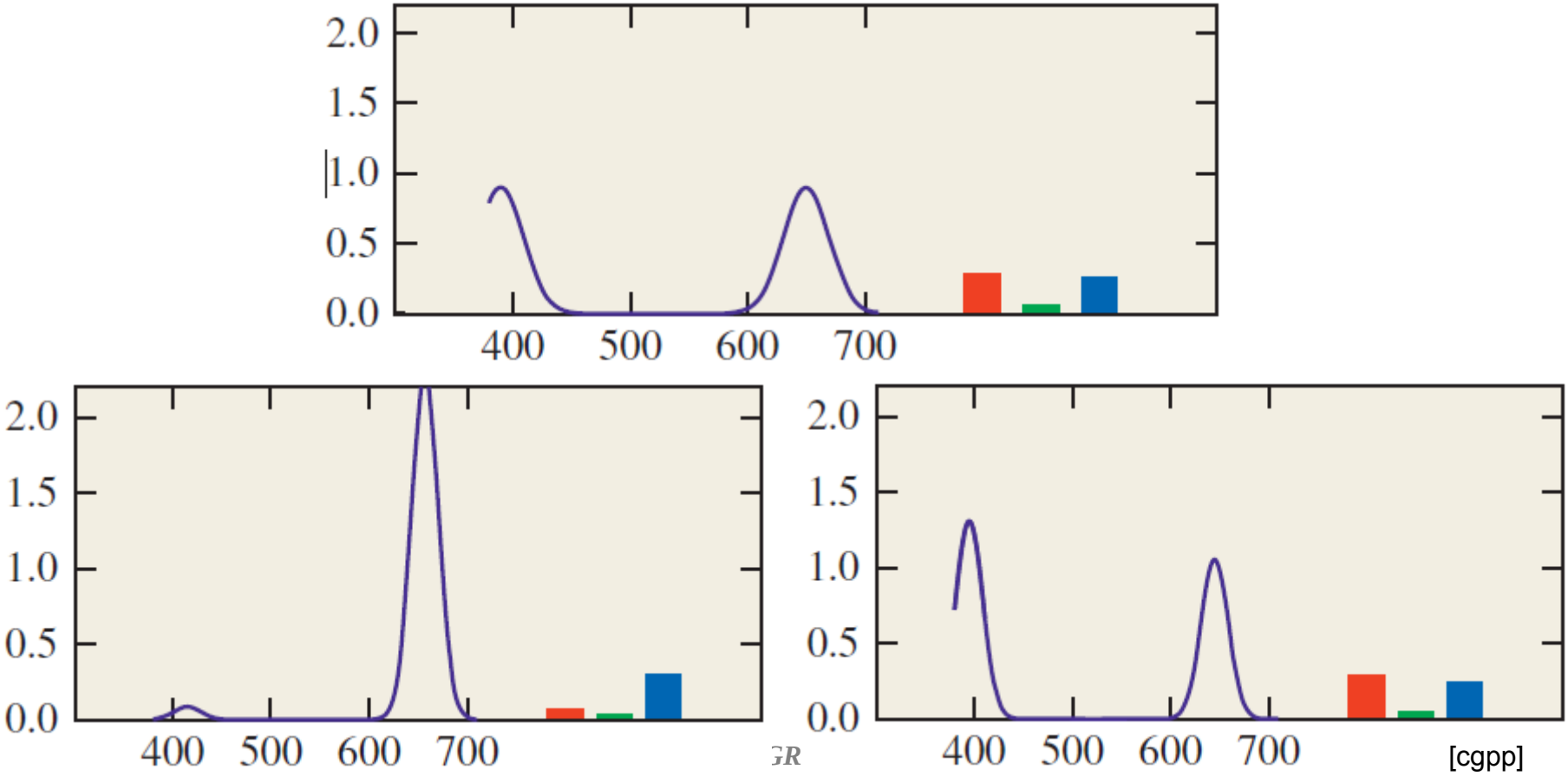
- Světla o různém spektrálním složení vnímáme jako stejná (metamery)
  - Mají stejnou odezvu SML
  - Díky metamerismu lze sestrojit televizi, …
  - V real-time grafice – místo celého spektra jen kanály R,G a B



[cgpp]

Metamerismus světel

- Svítíme-li na stejný materiál metamerickými světly, dostaneme značně odlišnou spektrální odezvu

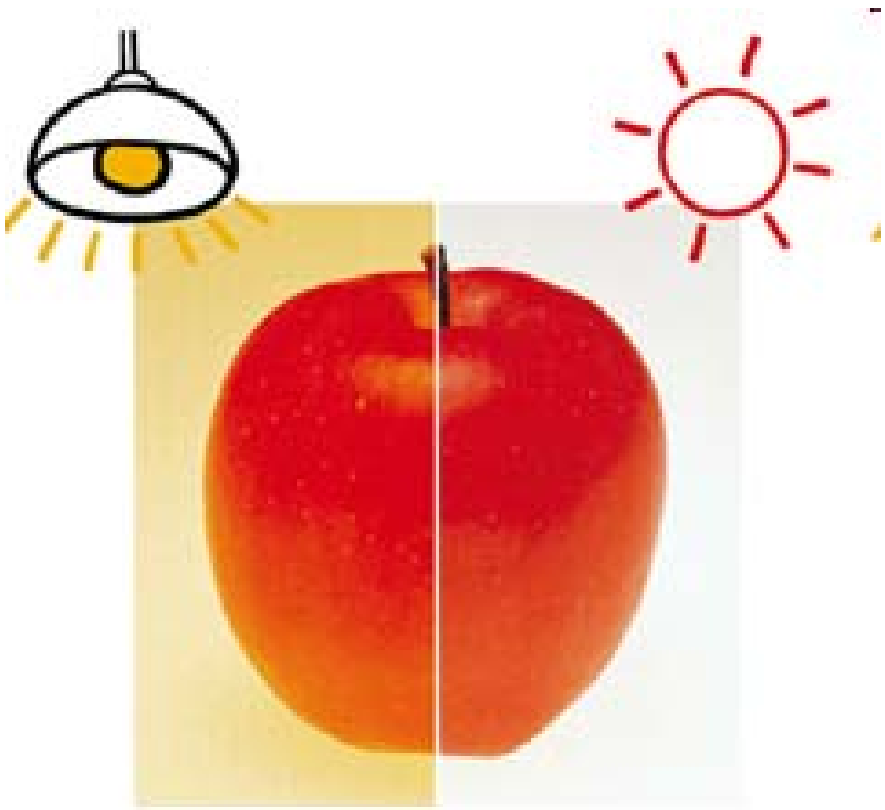


[cgpp]

Chromatická adaptace

= Tendence posouvat rozsah oka tak, aby byla vnímána *bílá*

- Na fotografii bez korekce bílé vidíme, že večer do modra, žárovka do žluta,…



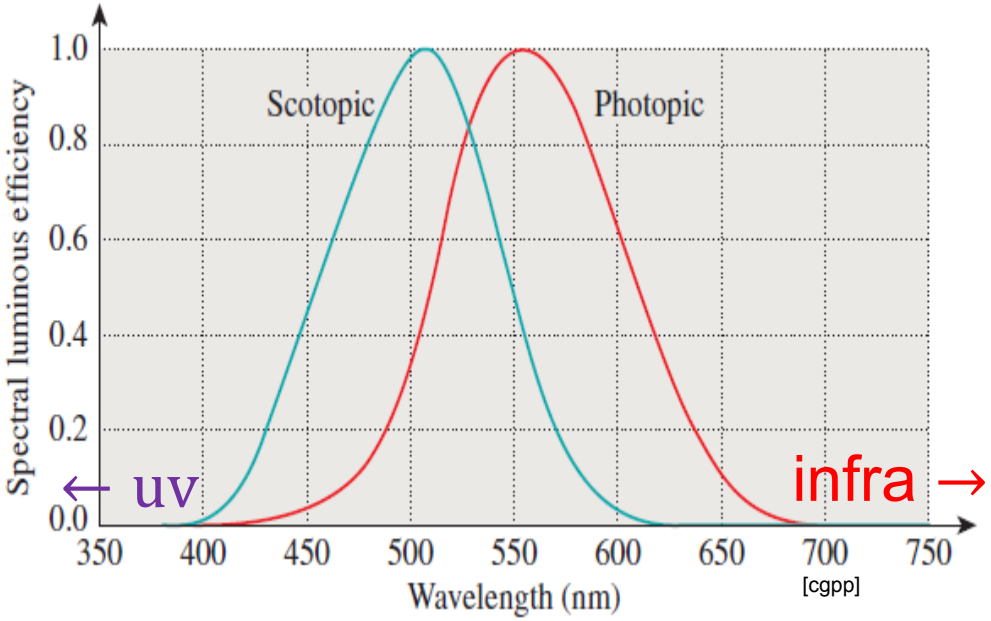
www.konicaminolta.com

Purkyňův jev [disertace1818]

= posun relativní citlivosti k modré v noci

- Denní vidění (fotopické)
  – čípky maximum pro zelenou
- Noční vidění (skotopické)
  – tyčinky maximum pro modrou barvu, nevidí červenou

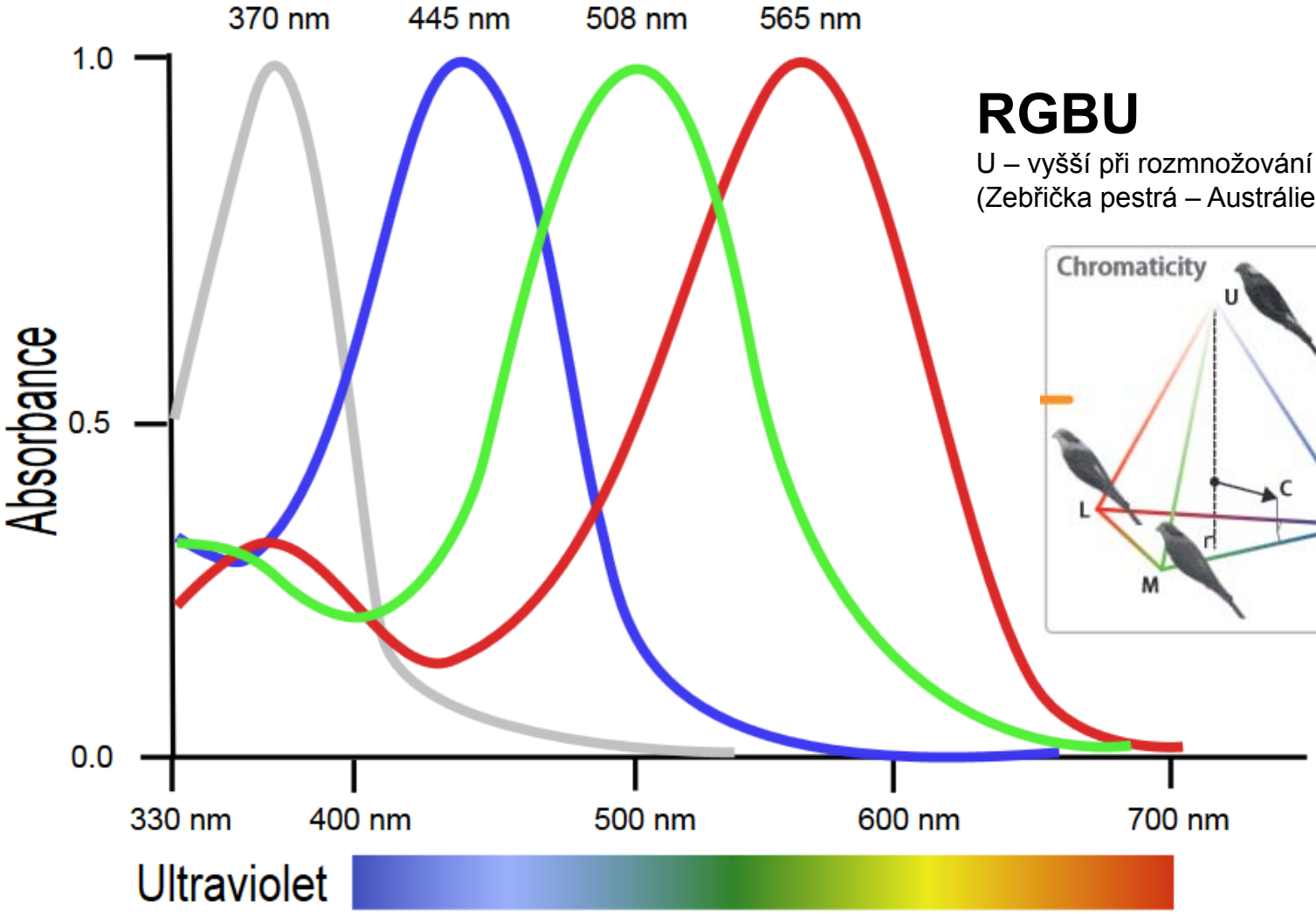
[Grygar]



[cgpp]

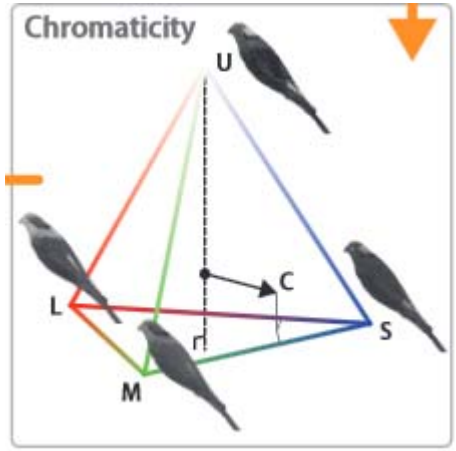
- Modrý papír v noci světlejší než červený a naopak
- Večer červené květy tmavnou, žluté blednou, modré vidíme jasněji

Zajímavost – tetrachromáti vidí 4 barvy



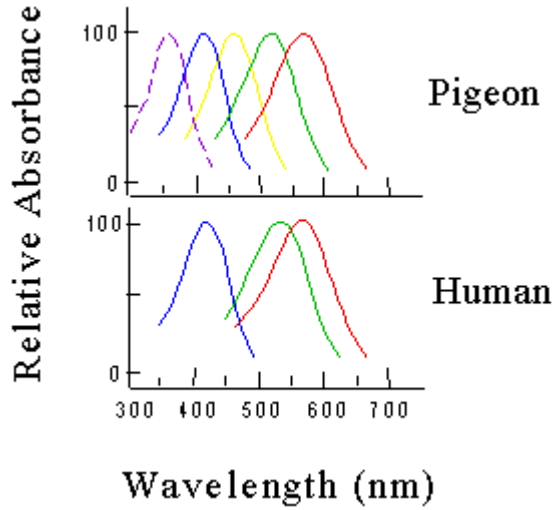
RGBU

U – vyšší při rozmnožování u ptáků
(Zebřička pestrá – Austrálie)



[http://en.wikipedia.org/wiki/File:BirdVisualPigmentSensitivity.svg]

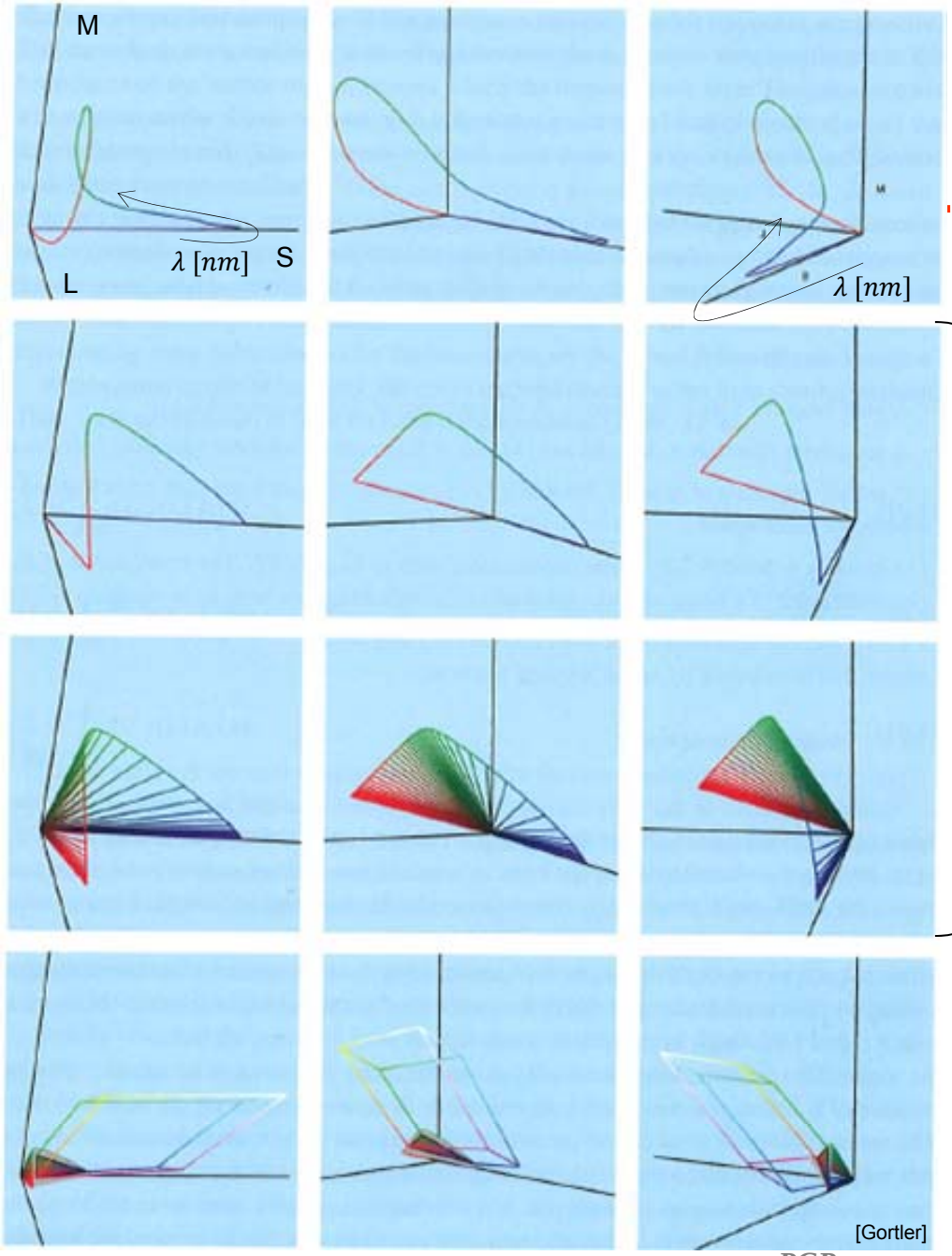
Zajímavosti



[http://people.eku.edu/ritchisong/pigment.gif]

- Holubi, motýli
  - pentachromáti - vidí 5 barev
- Včely
  - trichromáti, ale nižší vlnové délky
  - místo červené vidí ultrafialovou
- Savci a opice (jiné než primáti příbuzní s člověkem)
  - Dichromáti
  - Rozlišují barvy i v noci
- Mořští savci
  - Monochromáti
- Dodecachromáti (12)
  - Krevety – 16 fotoreceptorů – rozlišují polarizaci světla

Proč má chromatický diagram CIE xyY tvar podkovy (horseshoe)?



Prostor LMS(citlivost čípků oka

= Odezva I_λ na monochromatické světlo (o délce λ) lze zobrazit jako bod v prostoru LMS

Zelená osa je imaginární

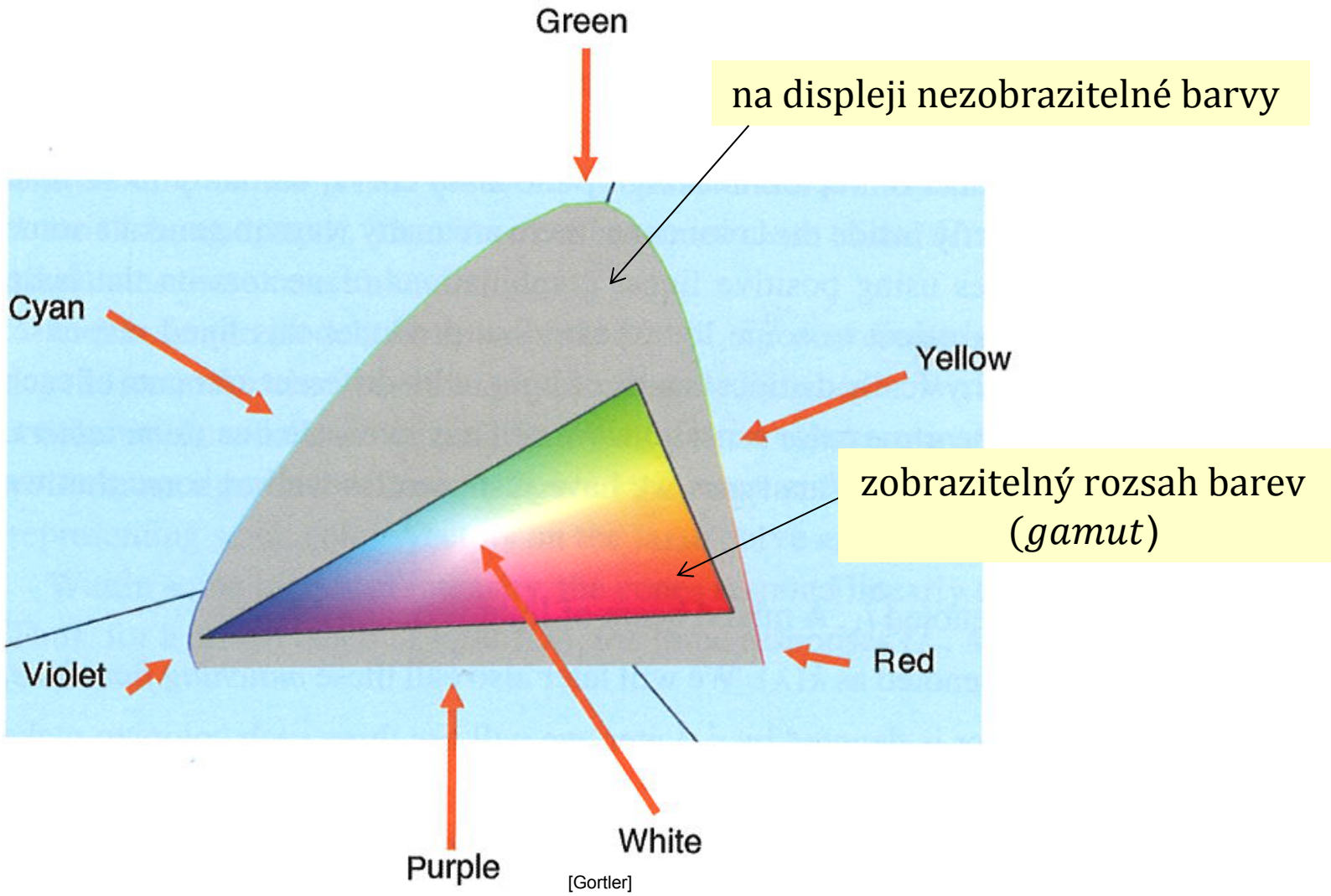
- nelze izolovaně vybudit jen čípky typu M
- vždy se vybudí i S, a L

Po normalizaci vektorů $L + M + S = k\ , k = konst.$

Přidán řez a RGB prostor

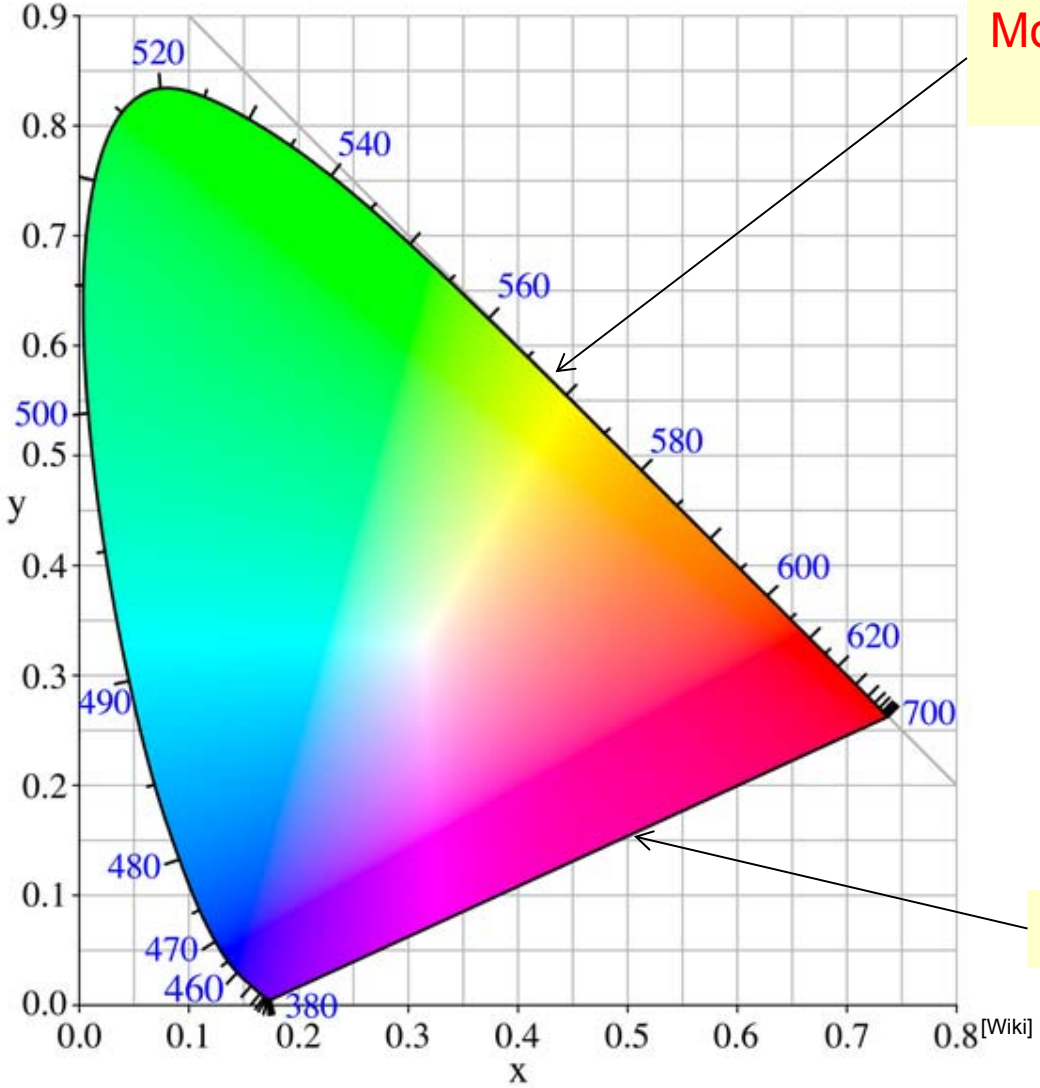
(Bodem $[1,0,0]^t_{RGB}$)

2D diagram barev



[Gortler]

Chromatický diagram CIE 1931 xyY



[Wiki]

Monochromatické barvy spektra
(po obvodu)

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z}$$

Protože $x + y + z = 1$,
stačí ukládat xyY
(x,y a jas Y)

Červeně purpurové odstíny

Barevné modely

- **Barevný model** = abstraktní mat. model, v němž jsou barvy reprezentovány n-ticemi čísel (n = 3 nebo 4)
  => výsledná sada barev tvoří **barevný prostor**

- **Aditivní skládání barev –** barva je lineární kombinací primárních barev (*primaries*)

  $$C = \alpha_1 . C_1 + \alpha_2 . C_3 + \alpha_3 . C_3 , \qquad 0 \leq \alpha_i \leq 1$$

  $\Rightarrow$ popisuje, která světla nutno vyzářit k vytvoření barvy
  $\Rightarrow$ intenzita roste s rostoucími koeficienty α_i
  $\Rightarrow$ maximální intenzita $\alpha_i \Rightarrow$ bílá barva

- **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

Barva a světlo podrobněji

[cgcc] Hughes et al.: Computer Graphics – Principles and practice, 3rd edition, Addison-Wesley, 2014

[MGA] Předmět Multimediální a grafické aplikace

[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

Osvětlování - *lighting*

- Osvětlení přidá na realističnosti objektů ve scéně

Neosvětlené objekty

Osvětlené objekty

Osvětlení objektu

Pro každý pixel obrázku je nutno určit barvu

Osvětlovací model

- Spočítá barvu v bodě scény (vrchol, pixel,…)
- Aproximace reality

Osvětlovací model

- Postup, jak vypočítat osvětlení v konkrétním bodě scény (vrchol, pixel,…)
- Fyzikálně přesný
  - Spektrum světelných zdrojů, šíření světla prostředím, interakce s materiálem (hrubost povrchu, rozptyl uvnitř materiálu, …)
  - Př.: Lambertův model pro difůní odraz
- Empirický
  - Nesnaží se popsat realitu
  - Vzorce tak, aby „dobře“ vypadal
  - Př.: Phongův osvětlovací model

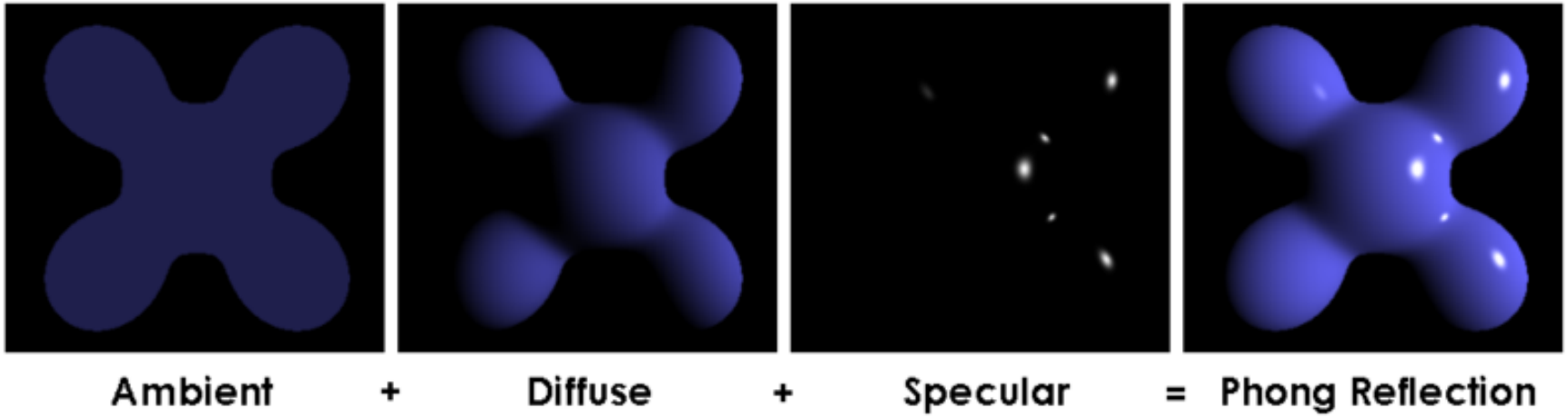
Osvětlovací model

- Postup, jak vypočítat osvětlení v konkrétním bodě scény (konkrétním pixelu obrázku)

- Typickými vstupy jsou
  - Normála povrchu
  - Směr ke světlu
  - Směr k pozorovateli
  - Materiál

Phongův osvětlovací model

- Empirický osvětlovací model
- Počítá barvu v bodě po složkách
  - Ambientní (globální od prostředí a od světel)
  - Difúzní – Lamberian (od světel)
  - Zrcadlově odražené (od světel)
  - Vyzářené (tělesem)



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

Difúzní světlo (diffuse, Lambertian reflection)

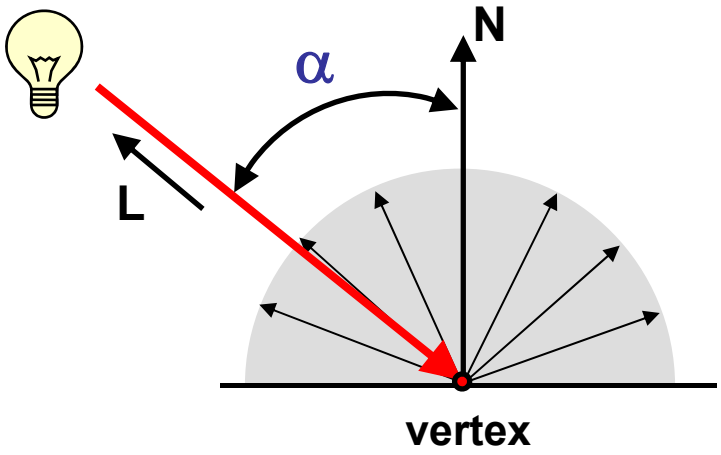
- **Difúzní světlo** je směrové světlo vysílané světelným zdrojem
- Po dopadu je odraženo do všech směrů, proto je stejně jasné, nezávisle na poloze kamery
- Množství rozpáleného světla je úměrné úhlu mezi směrem ke světlu a normálou povrchu $\Rightarrow$ **mění se s cosinem úhlu dopadu**

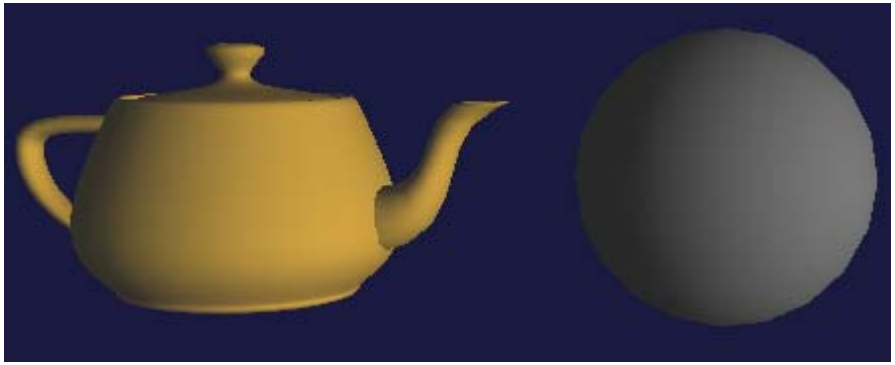
Rovnice odraženého difúzního světla

$$\mathbf{diffuse_{reflected} = (max[\cos \alpha, 0]) * diffuse_{light} * diffuse_{material}}$$

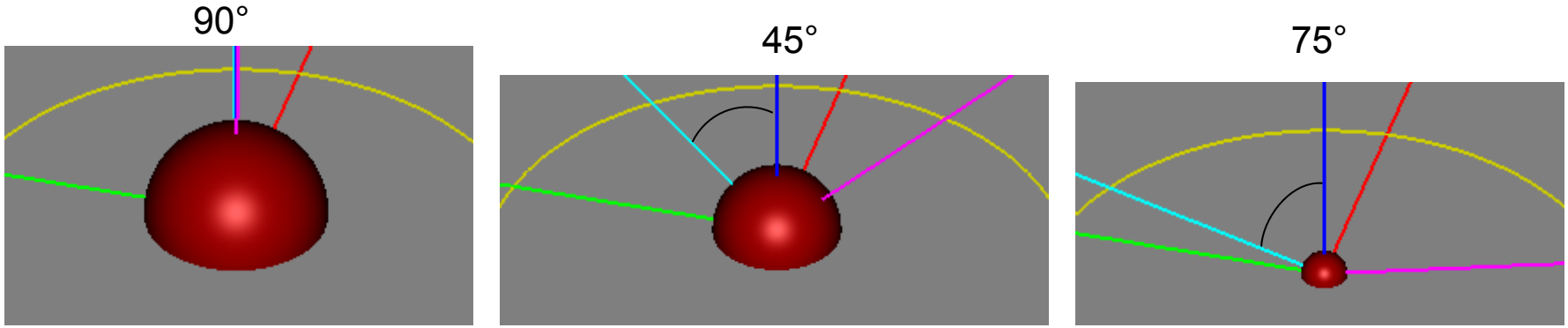
$\cos \alpha$ = L.N skalární součin L, N jsou jednotkové vektory

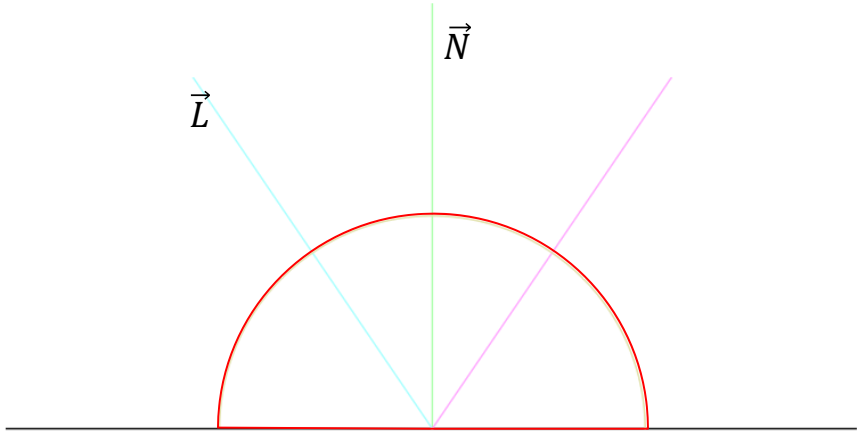
$\cos \alpha$ je maximální při dopadu kolmo na plochu





Lambertian diffuse model



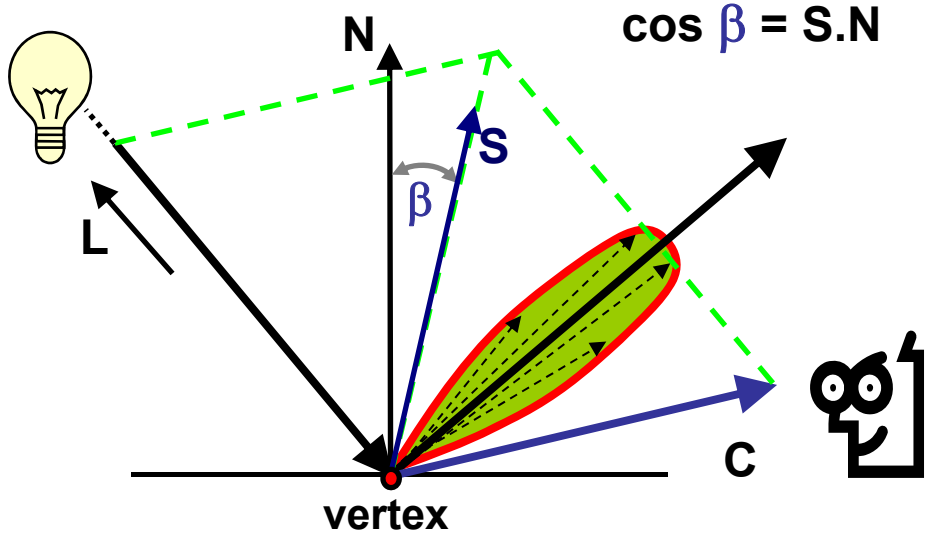


Zrcadlově odražené světlo (specular)

- 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

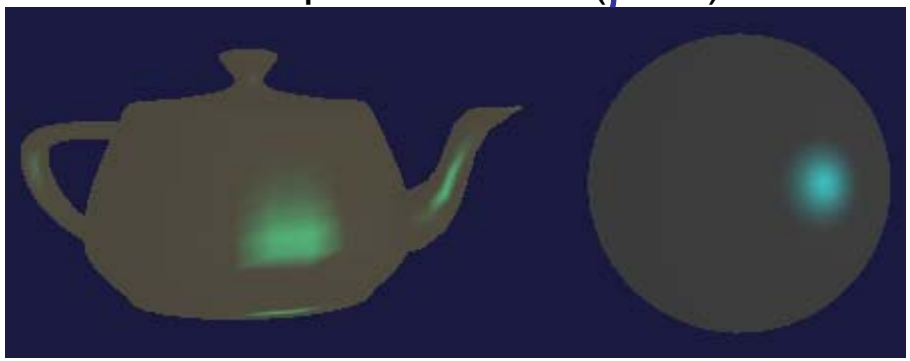
$$\mathbf{specular_{reflected} = (max[\cos \beta, 0])^{shininess\_material} * specular_{light} * specular_{material}}$$



$\cos \beta = S.N$

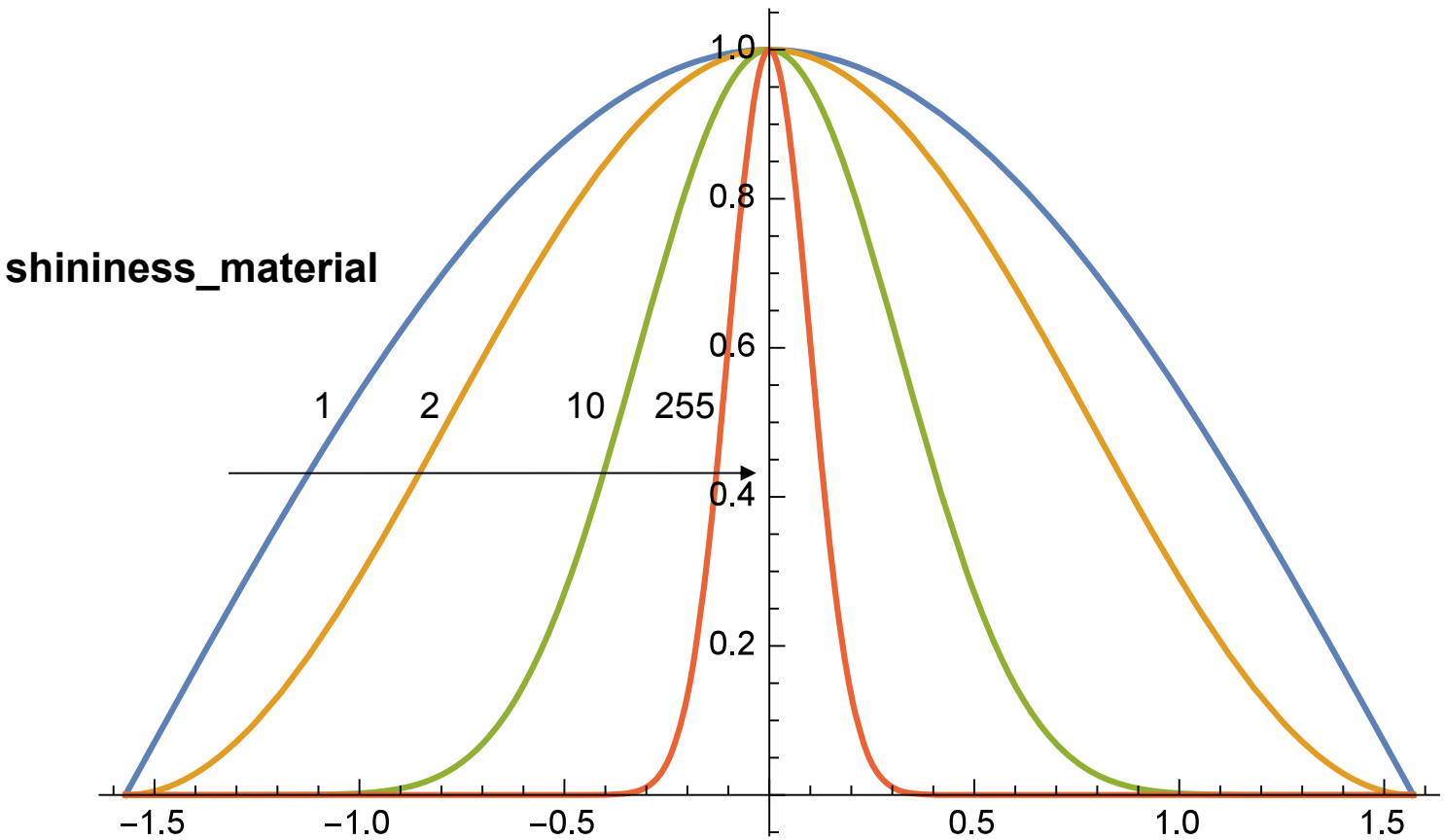
S, N jsou jednotkové vektory

$\cos \beta$ je maximální když se světlo odráží do oka pozorovatele (β =0).

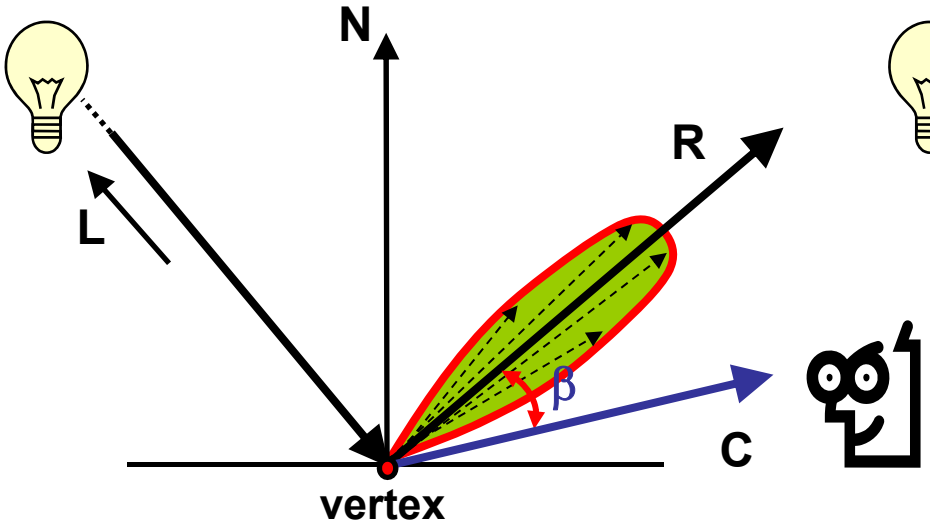


S = (C+L) / |C+L|
half angle

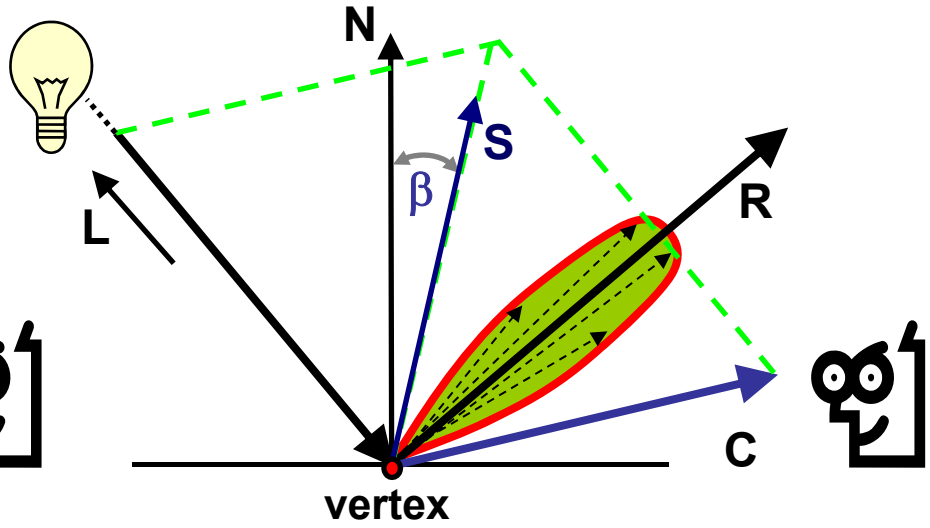
Different values of shininess



Varianty pro výpočet zrcadlové složky



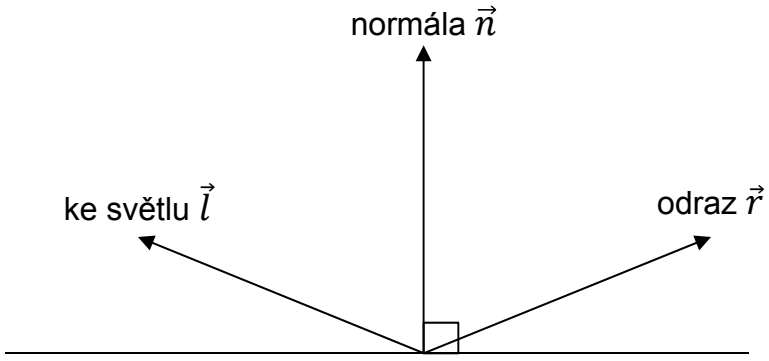
Phongův

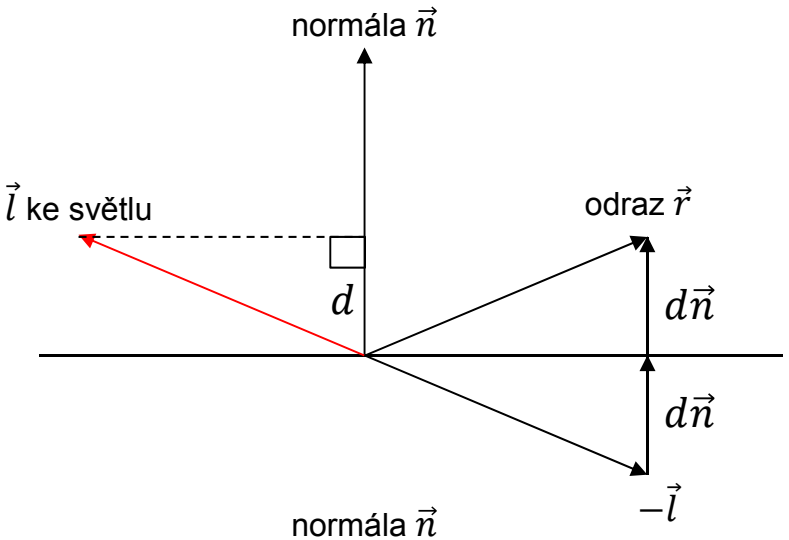


S = (C+L) / |C+L|
half angle

Blinn-Phongův

Vzorec pro odražený vektor - odvození



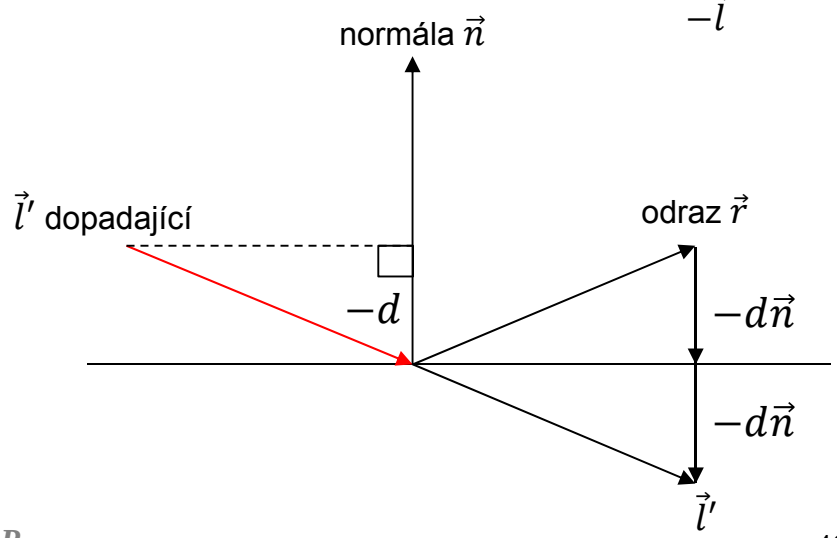


$$\vec{r} = -\vec{l} + 2d\vec{n}$$

$$\vec{r} = -\vec{l} + 2\cos(\alpha)\,\vec{n}$$

$$\vec{r} = -\vec{l} + 2\left(\vec{l}.\vec{n}\right)\vec{n}$$

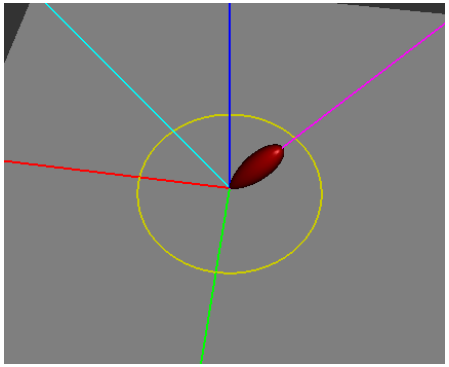
$$\vec{r} = -\vec{l} + 2\,\mathrm{dot}\left(\vec{l},\vec{n}\right)\vec{n}$$

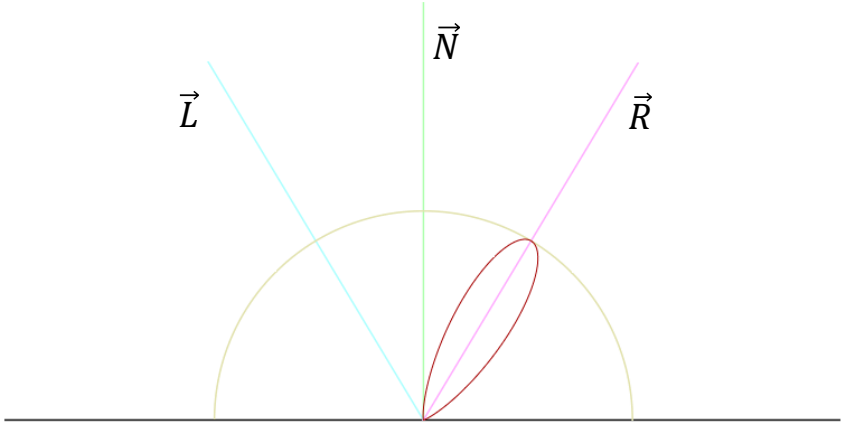


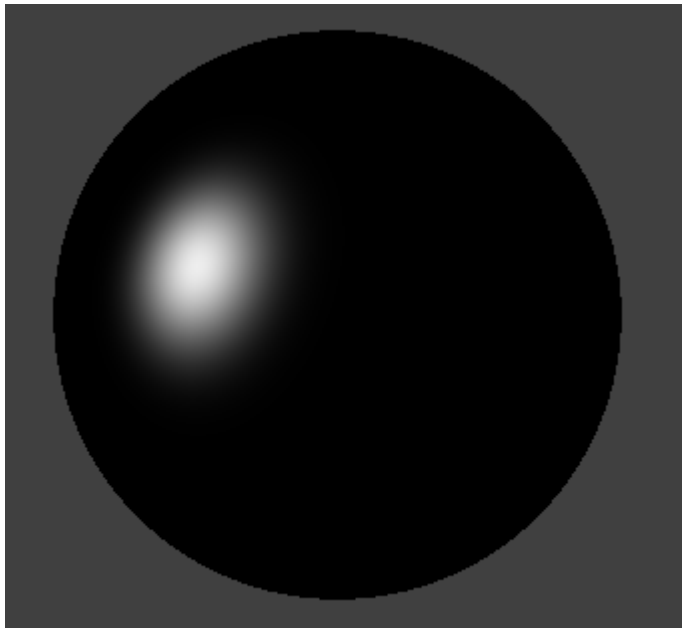
V GLSL je funkce *reflect*()
počítá s dopadajícím vektorem $\vec{l}'$

$$\vec{r} = \vec{l}' - 2\,\mathrm{dot}\left(\vec{l}',\vec{n}\right)\vec{n}$$

Phong model – reflectance

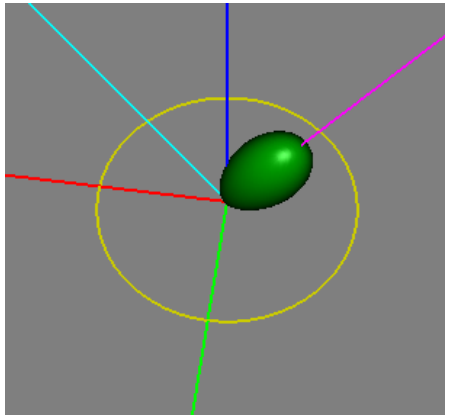


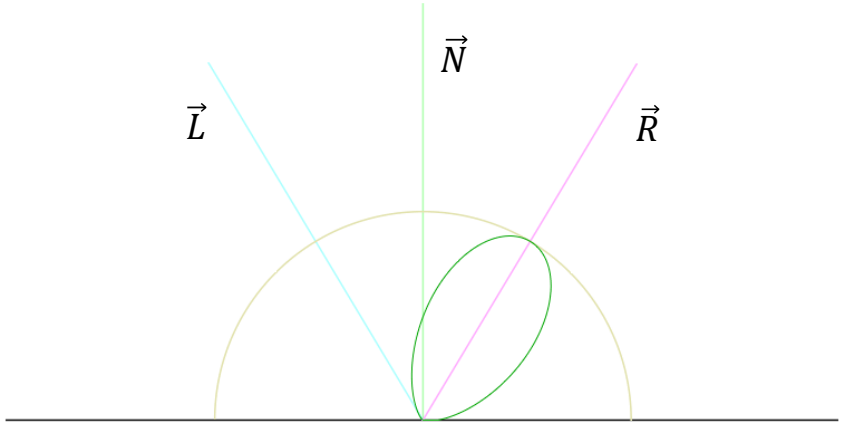


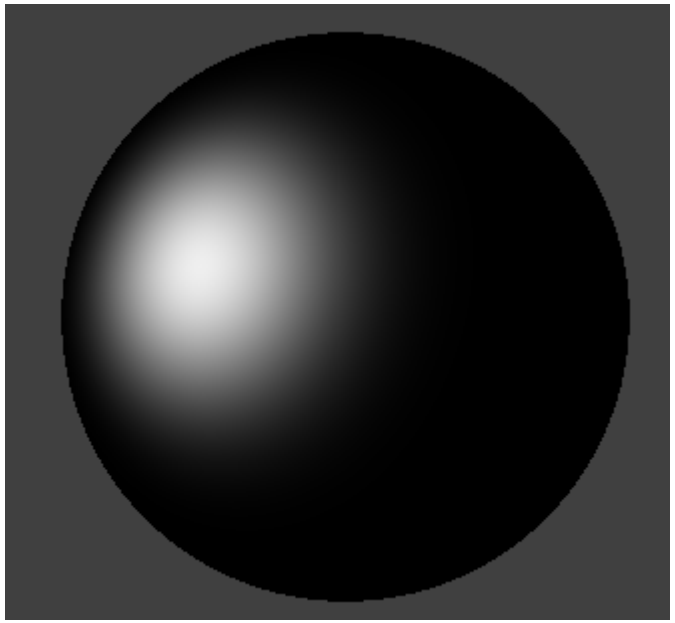


[Images made by BRDF Explorer, n=18]

Blinn-Phong model

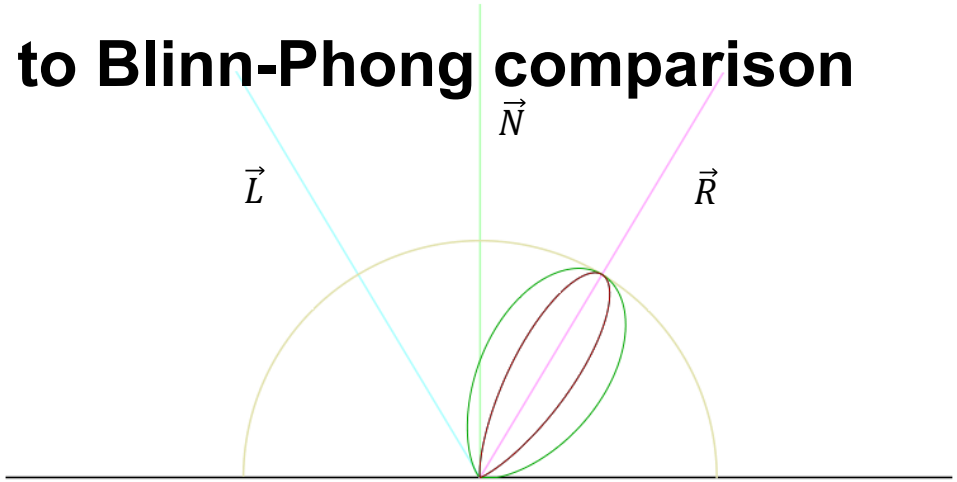


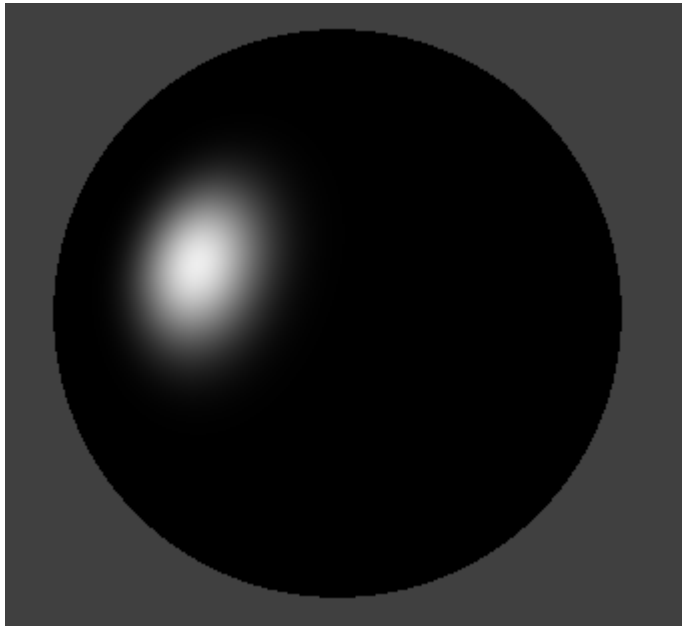


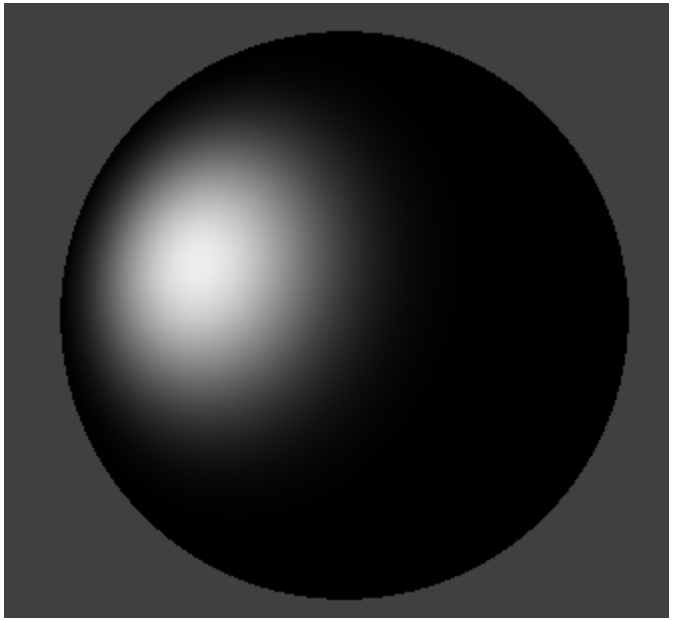


[Images made by BRDF Explorer, n=18]

Phong to Blinn-Phong comparison

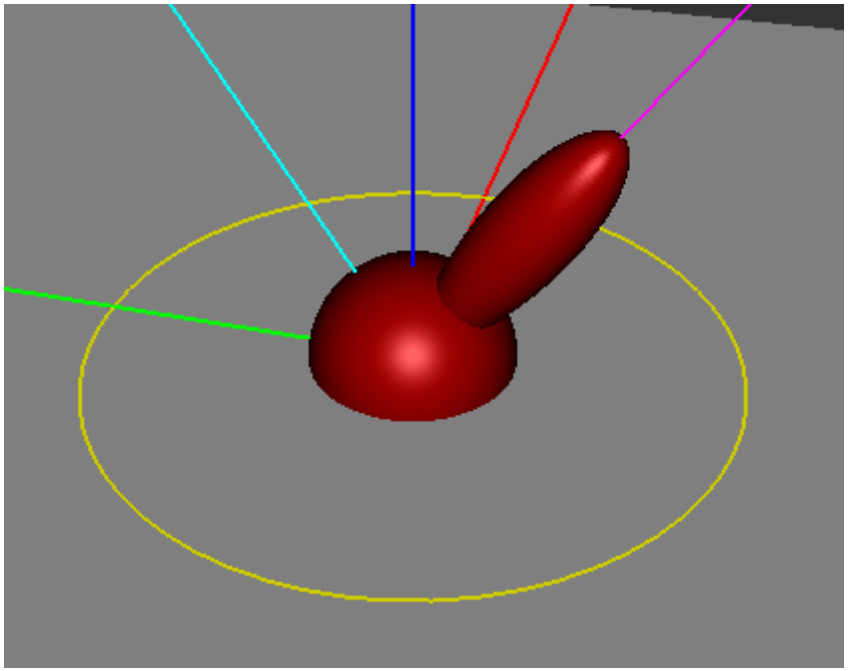






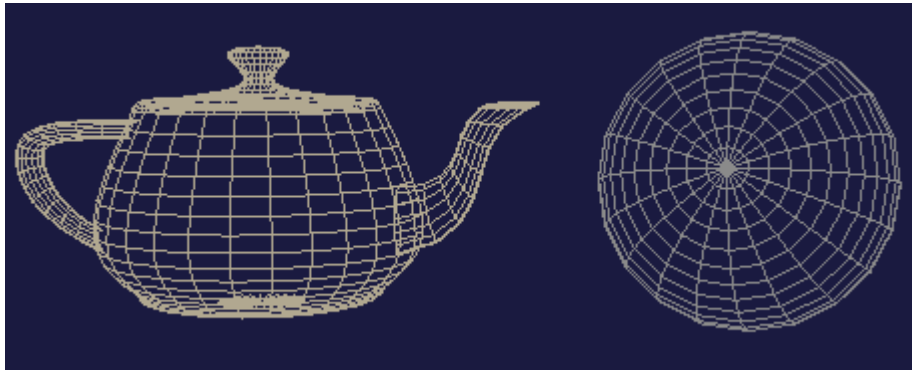
[Images made by BRDF Explorer, n=18]

Lambert + Phong reflectance



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 aproximace reality**.



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

Ambientní světlo (ambient)

- **Ambientní světlo od světel**

  Rovnice pro odražené ambientní světlo od jednoho sv. zdroje

  $$\mathbf{ambient_{reflected} = ambient_{light} * ambient_{material}}$$

  (nezávisí na poloze světla)

- **Ambientní světlo od prostředí**

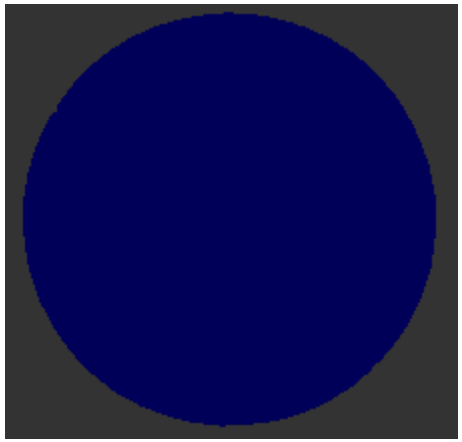
  Rovnice pro odražené ambientní světlo od prostředí

  $$\mathbf{global\_ambient_{reflected} = global\_ambient_{light\_model} * ambient_{material}}$$

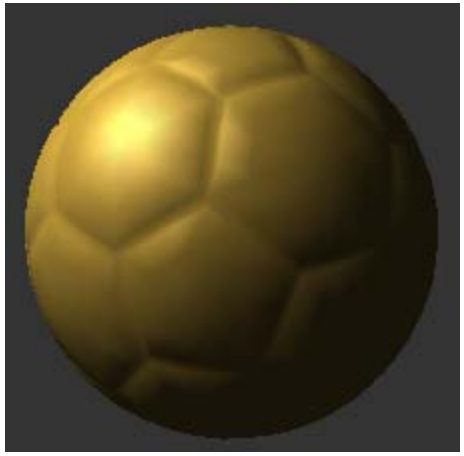
  (nezávisí na světlech)

Vyzářené světlo (emissive)

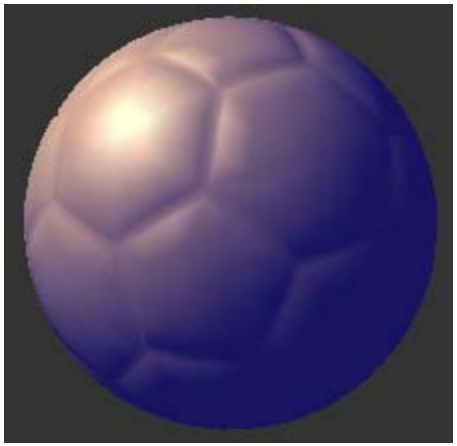
- Simuluje světlo vyzářené objektem, ale
  - nedodává světlo jiným objektům ve scéně
  - není ovlivněno světelnými zdroji
- Objekt osvětlený pouze vyzářeným světlem (a žádným sv. zdrojem) vypadá jako objekt osvětlený ambientním osvětlením



jen vyzářené



zrcadlové + difúzní + ambientní



vše dohromady

Kompletní Phongův osvětlovací model

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

Osvětlení v bodě scény (Phongův osvětlovací model)

$$\mathbf{color = emission + global\_ambient_{reflected} + \sum light_i}$$

$$\mathbf{light_i = spotlightEffect * attenuationFactor * (ambient_{reflected} + diffuse_{reflected} + specular_{reflected})}$$

Vlastnosti světel a útlum se vzdáleností – viz dále

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)

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

- 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 při interpolaci a některými transformacemi
- 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 **4x4** maticí **M**
- transformace se provede vynásobením matice **M** a vektorem **v**.
- na vektory nemá vliv posunutí (mají w=0)

Lineární transformace
Koeficienty
rotace (zkosení)

Koeficienty změny měřítka

Koeficienty posunutí

$$v' = [x', y', z', w']^T = M.v = \begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

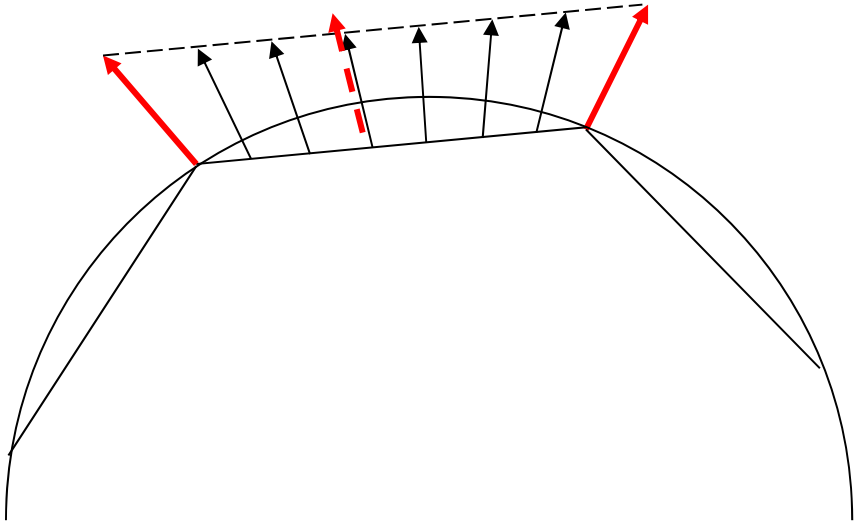
Transformo-vaný vrchol

Projekční koeficienty

Vrchol, který je transformován

Zkrácení normál při interpolaci

- Při interpolaci mezi vrcholy při rasterizaci
- Interpolované vektory na vstupu fragment shaderu jsou kratší



⇒ **normály nutno před použitím normalizovat**

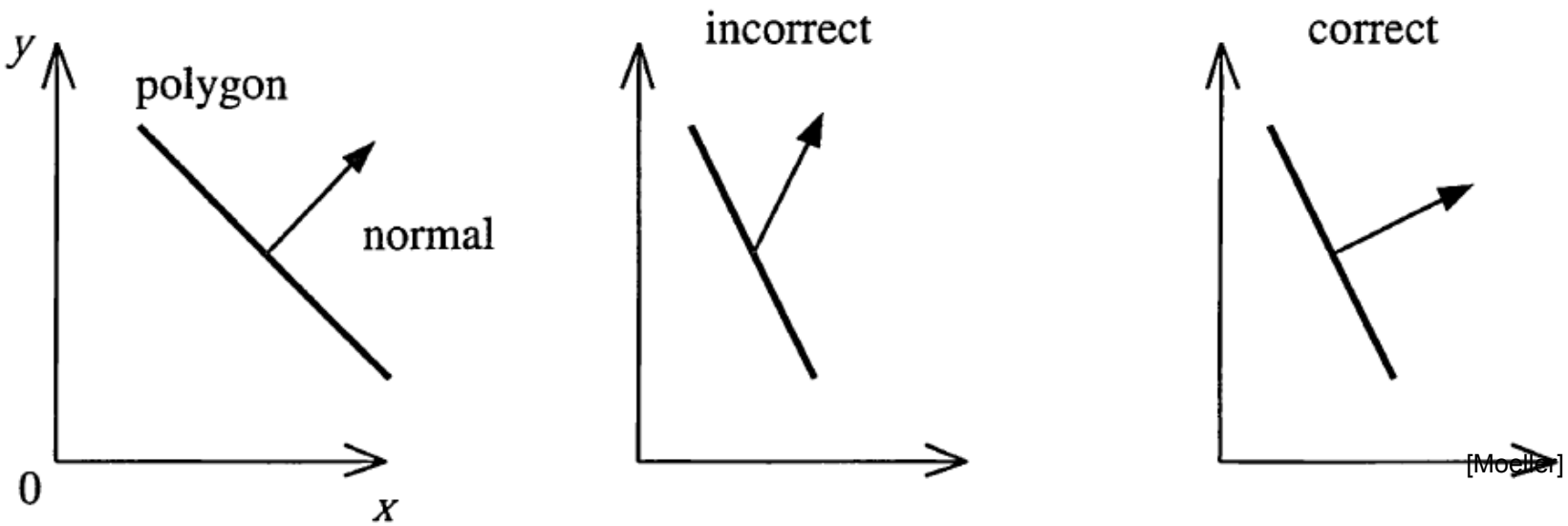
Ve FS: N = **normalize** (interpolatedNormal);
N = **normalize(**Matrix * interpolatedNormal**);**

Změna délky normály při změně měřítka

- Při změně měřítka ve všech směrech stejné *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 afinní modelovací transformace M
- Transformovaná normála přestane být kolmá k povrchu (vektor $M\boldsymbol{n}$, o kterém si myslíme, že je to normála, už normálou není) => pro normály je matice $N = (M^{-1})^T$



[Moeller]

scaled by 0.5 along the x dimension

Odvození transformace pro normály 1

- Vrchol $\boldsymbol{x}$ je transformován maticí M násobením touto maticí:

  $\boldsymbol{x}' = M\boldsymbol{x}$ matici M známe – je to modelovací matice

- Stejně je transformována tečna $\boldsymbol{t}$ (rozdíl dvou povrchových vrcholů)

  $\boldsymbol{t}' = \bar{M}\boldsymbol{t}$ $\bar{M}$ je lineární část matice M

- Normála $\boldsymbol{n}$ je transformována maticí N, matici N hledáme

  $\boldsymbol{n}' = N\boldsymbol{n}$

- Předpoklad: normála $\boldsymbol{n}$ je kolmá na tečnu $\boldsymbol{t}$ před transformací i po transformaci geometrického objektu maticí M:

  před $\boldsymbol{n}\,.\,\boldsymbol{t} = 0$ po $\boldsymbol{n}'\,.\,\boldsymbol{t}' = 0$

- Silnější tvrzení – $\boldsymbol{n}$ a $\boldsymbol{t}$ svírají před a po transformaci stejný úhel

$$\boldsymbol{n}\,.\,\boldsymbol{t} = \boldsymbol{n}'\,.\,\boldsymbol{t}'$$
$$\boldsymbol{n}\,.\,\boldsymbol{t} = \boldsymbol{n}'.\bar{M}\boldsymbol{t}$$

Odvození transformace pro normály 2

Vektory $\boldsymbol{n}$ a $\boldsymbol{t}$ svírají před a po transformaci stejný úhel

$\boldsymbol{n}\,.\,\boldsymbol{t} = \boldsymbol{n}'.\bar{M}\boldsymbol{t}$ vyměníme strany rovnice

$\boldsymbol{n}'.\bar{M}\boldsymbol{t} = \boldsymbol{n}\,.\,\boldsymbol{t}$ skalární součin -> násobení matic

$\boldsymbol{n}'^{\boldsymbol{T}}\bar{M}\boldsymbol{t} = \boldsymbol{n}^T\boldsymbol{t}$ pro nenulové $\boldsymbol{n}, \boldsymbol{t}$ a regulární $\bar{M}$

$\boldsymbol{n}'^{\boldsymbol{T}}\bar{M} = \boldsymbol{n}^T$ $/\ \bar{M}^{-1}$ zprava

$\boldsymbol{n}'^{\boldsymbol{T}} = \boldsymbol{n}^T\bar{M}^{-1}$ $/\ ^T$ $(AB)^T = B^TA^T$

$\boldsymbol{n}' = (\bar{M}^{-1})^T\boldsymbol{n}$ $(\bar{M}^{-1})^T$ matice pro transformaci normál

Normály transformujeme maticí $N = (\bar{M}^{-1})^T$

Pro rigidní transformace je $\bar{M}^{-1} = M^T$ a tedy $N = M$

Inverse matrix computation

- For sequence of simple transformations
  - Use inverse transformations in reverse order
  - Example: $M = T(\boldsymbol{t})R(\alpha)$
    invert as $M^{-1} = R(-\alpha)T(-\boldsymbol{t})$
- For orthogonal M (rigid transf., such as rotations)
  - $M^{-1} = M^T$, i.e.
    $N = (\bar{M}^{-1})^T = \bar{M}$
- For unknown transformations, compute the inverse
  - 3x3 matrix $\bar{M}$ is enough for vectors
  - Instead of division by the determinant use normalization of transformed normals

Inverse matrix computation by means of cofactor matrices

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

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

Transposed:

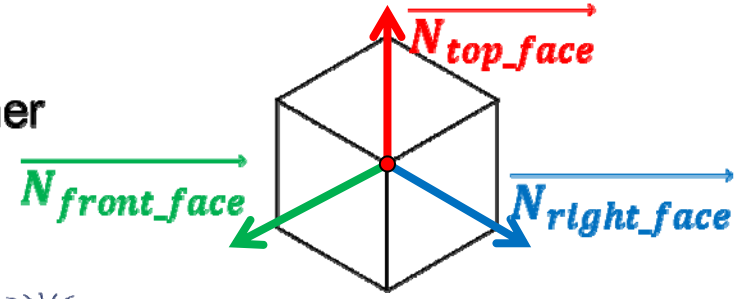
$$\mathbf{N} = \begin{bmatrix} m_{22}m_{11} - m_{12}m_{21} & m_{12}m_{20} - m_{10}m_{22} & m_{10}m_{21} - m_{20}m_{11} \\ m_{02}m_{21} - m_{22}m_{01} & m_{22}m_{00} - m_{02}m_{20} & m_{20}m_{01} - m_{00}m_{21} \\ m_{12}m_{01} - m_{02}m_{11} & \boxed{m_{10}m_{02} - m_{12}m_{00}} & m_{00}m_{11} - m_{10}m_{01}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

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

How to calculate normal vectors

- Cube

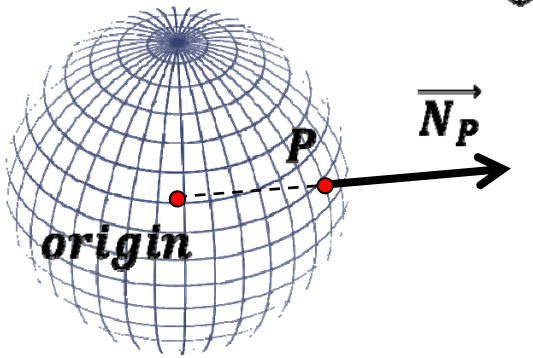
  3 different normal vectors per each corner



- Sphere

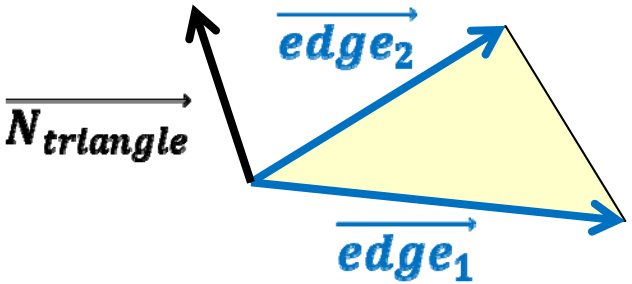
  $\overrightarrow{N_P}$ = normalize(P – origin)

  P ... any point on a sphere



- Triangle

  $\overrightarrow{N_{triagle}}$ = normalize(crossProduct($\overrightarrow{edge_1}$, $\overrightarrow{edge_2}$))



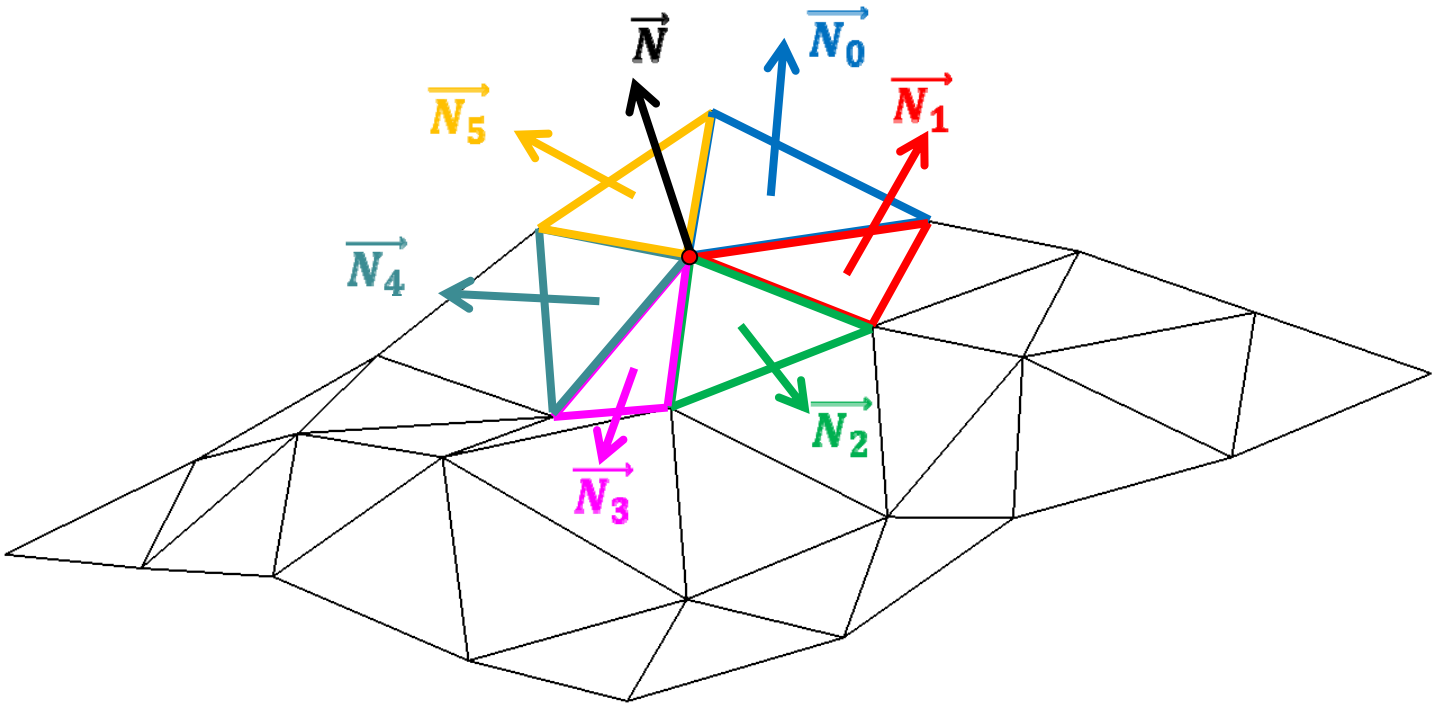
How to calculate normal vectors

- Mesh (polygonal surface)

$$\vec{N} = \text{normalize}\left(\sum_{i=0}^{pcount} \vec{N_i}\right)$$

pcount … number of surrounding polygons

$\vec{N_i}$ … normal vector of polygon i

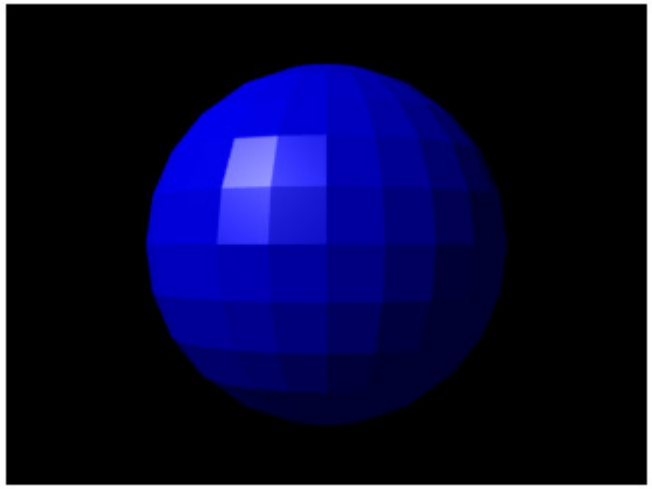


Stínování – výpočet osvětlení plošek

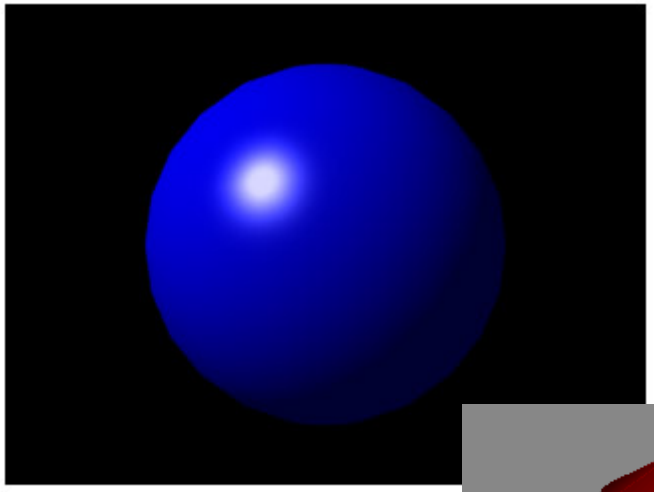
Stínování (shading) = postup vybarvení plošek

- **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

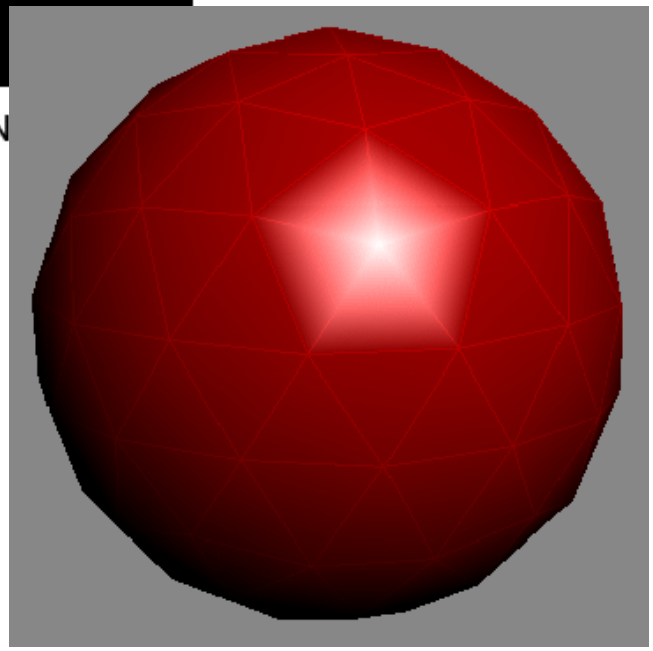


FLAT SHADING



PHONG SHADIN

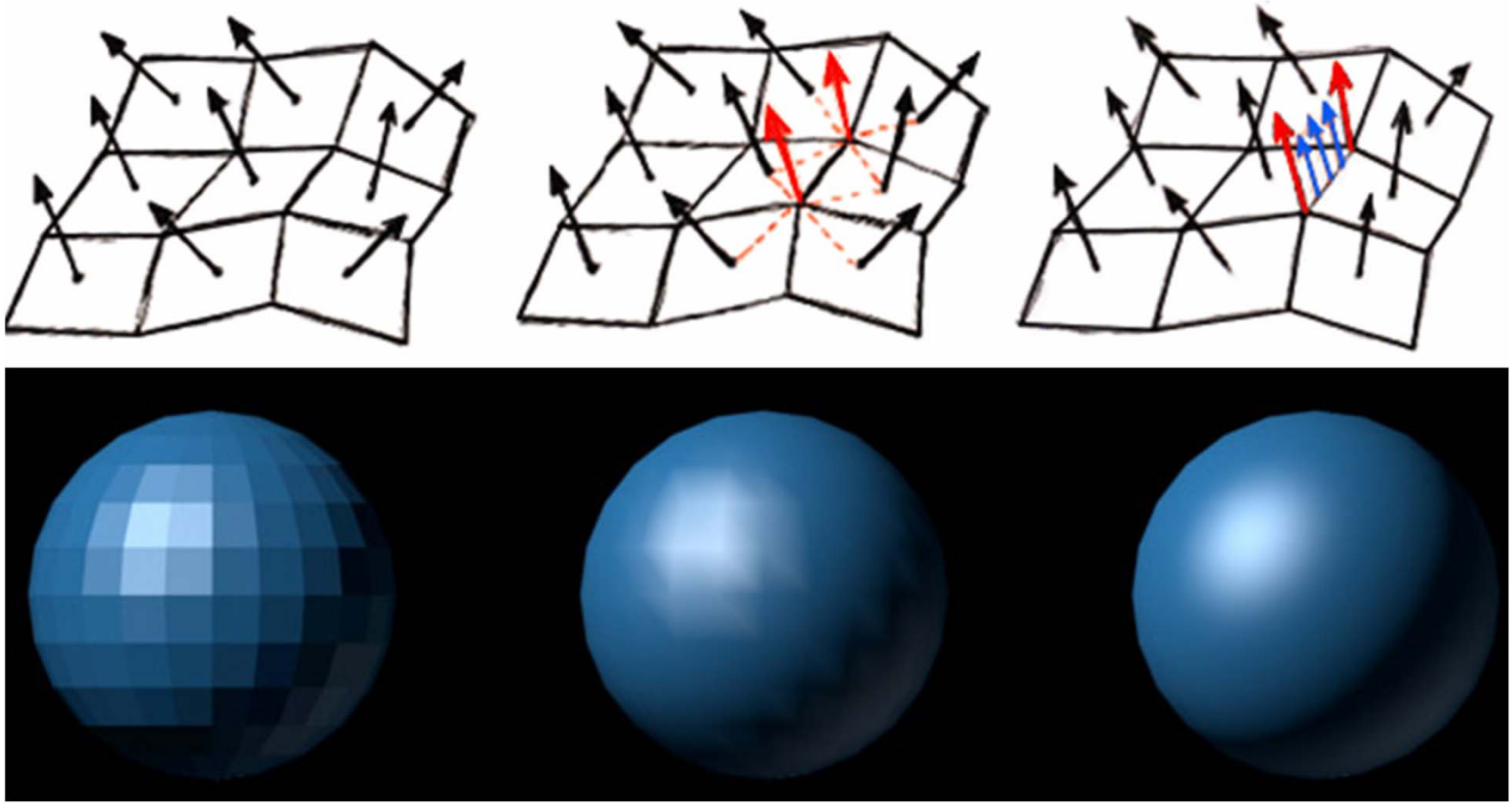
GOURAUD SHADING



Obrázky převzaty z [Wikipedia]

Gouraud_low_anim.gif

Normals Used for Flat, Gouraud and Phong Shading



[Introduction to Programming for Image Analysis with VTK, http://bioimagesuite.org/vtkbook/index.aspx]

Vliv rozlišení modelu a stínování

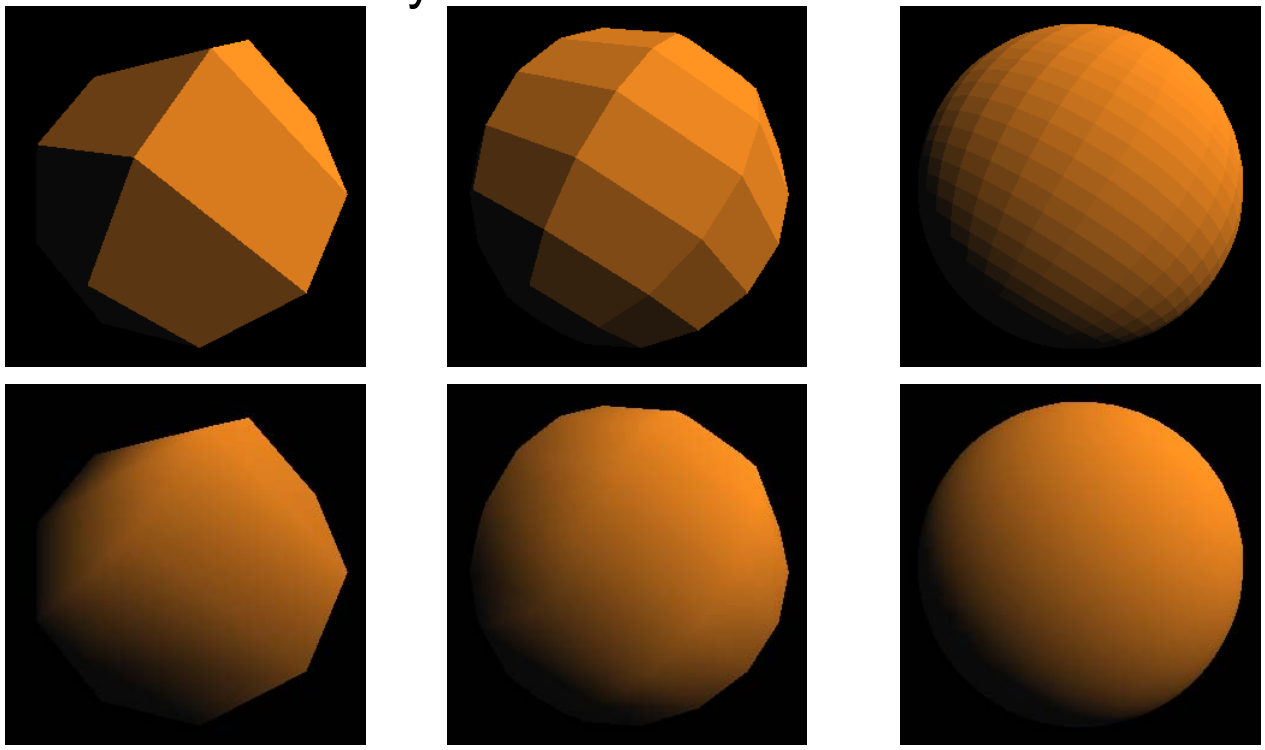
- Pokud se barva (osvětlení) počítá ve vrcholech je třeba
- dopočítat barvu mezi vrcholy - dle *stínovacího modelu*

FLAT

konstantní stínování

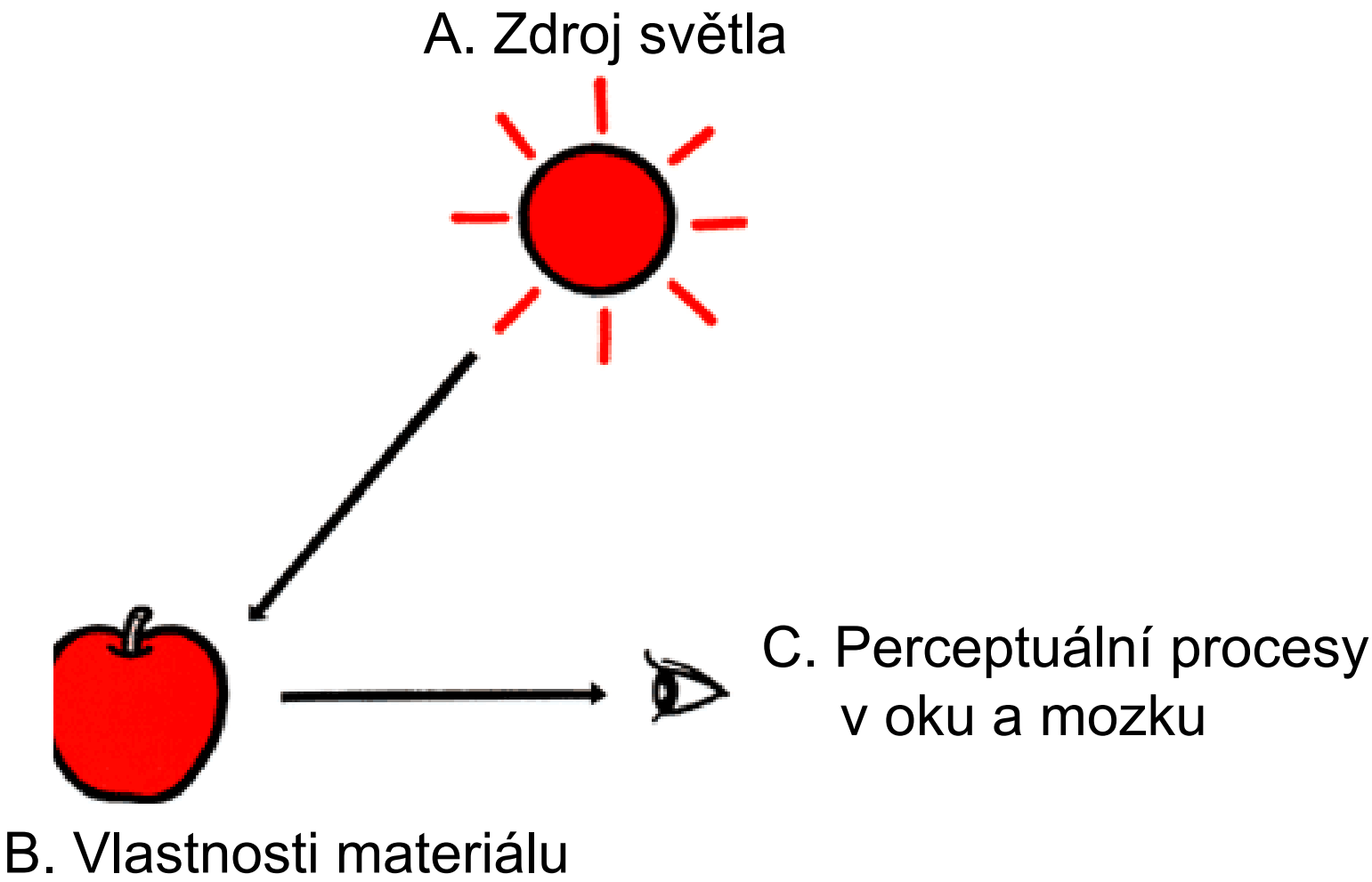
SMOOTH

Barva interpolována mezi vrcholy (Gouraudovo stínování)



- Při nízkém rozlišení vznikají artefakty na obrysových hranách
  $\Rightarrow$ vyšší rozlišení modelu vede na kvalitnější osvětlení

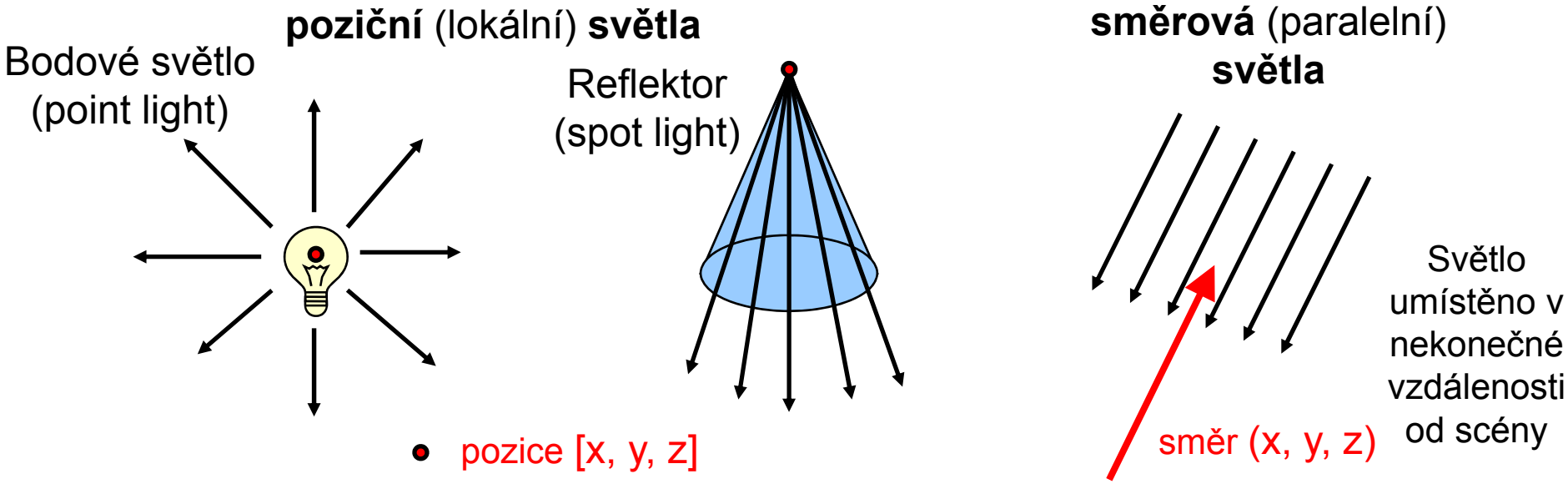
A. Zdroj světla



Světelné zdroje

- **Nezobrazují se**, jen osvětlují okolní objekty !!!
- Každý bod scény osvětlen všemi zapnutými světly
  $\Rightarrow$ OpenGL **nepočítá zastínění okolními objekty**
- Barva světla určena množstvím červené, zelené a modré
  $\Rightarrow$ RGB intenzita

Typické světelné zdroje v OpenGL



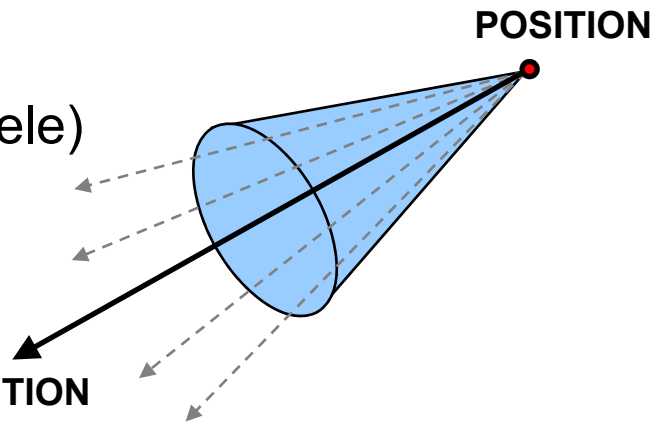
typ světelného zdroje se určí parametrem **POSITION** = (x, y, z, w)

- *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)

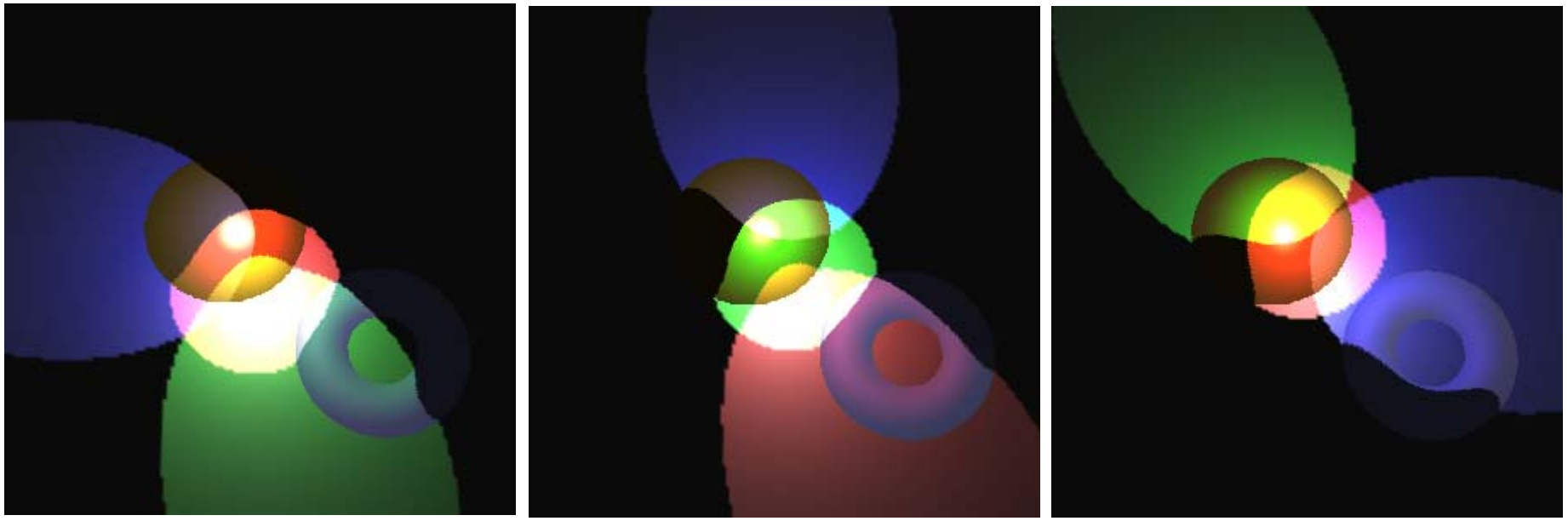
Světelné zdroje

SPOT_DIRECTION

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



red, green, and blue spot lights



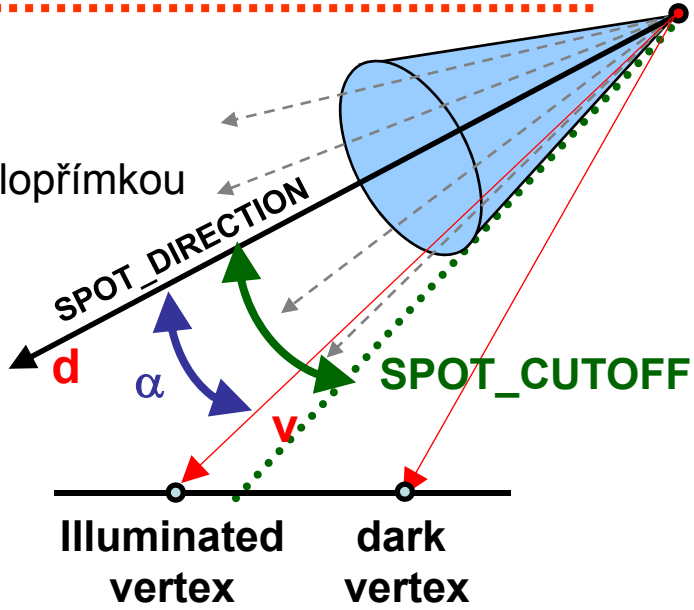
Světelné zdroje

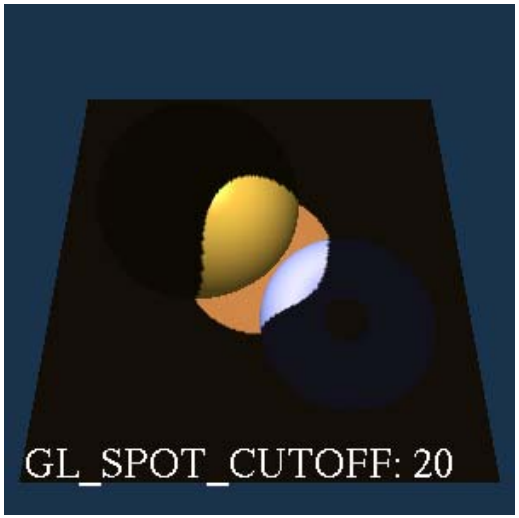
SPOT_CUTOFF

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

Povolené hodnoty pro **SPOT_CUTOFF**

- **180.0** *bodové světlo*
- ⟨**0.0, 90.0**⟩ *reflektor*
- $\cos\alpha \geq \cos(\text{SPOT\_CUTOFF})$

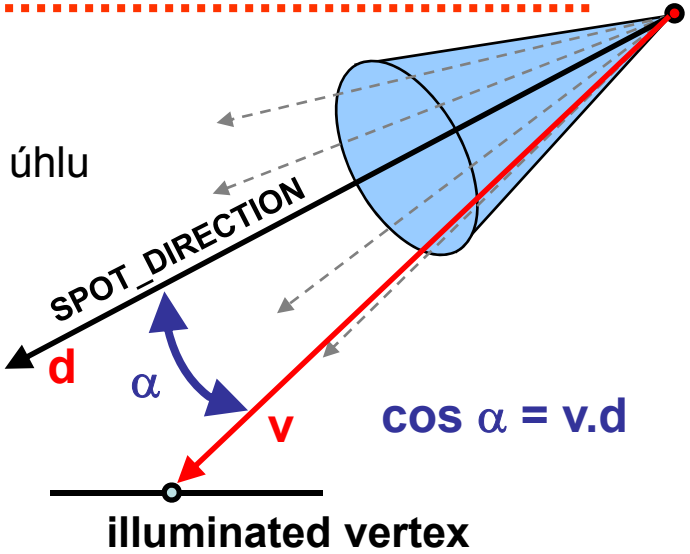




Světelné zdroje

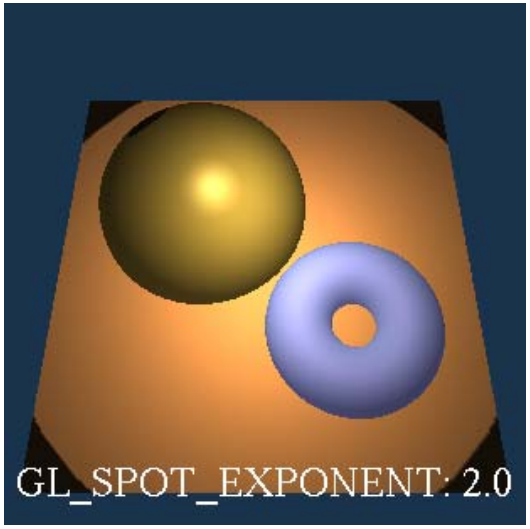
SPOT_EXPONENT

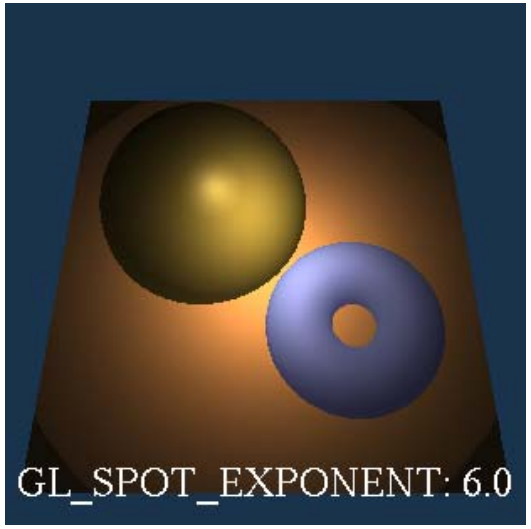
- Popisuje rozložení světla v rámci kužele v závislosti na úhlu od osy kužele (zaostření světla)
- Nejvyšší intenzita je ve středu kužele a klesá směrem ke stěnám

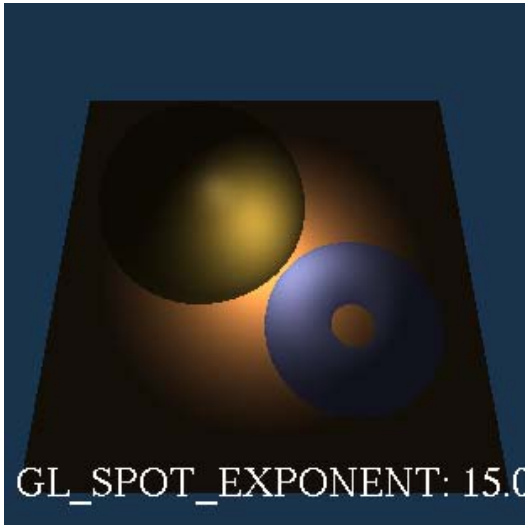


Efekt reflektoru se projeví následujícími hodnotami:

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







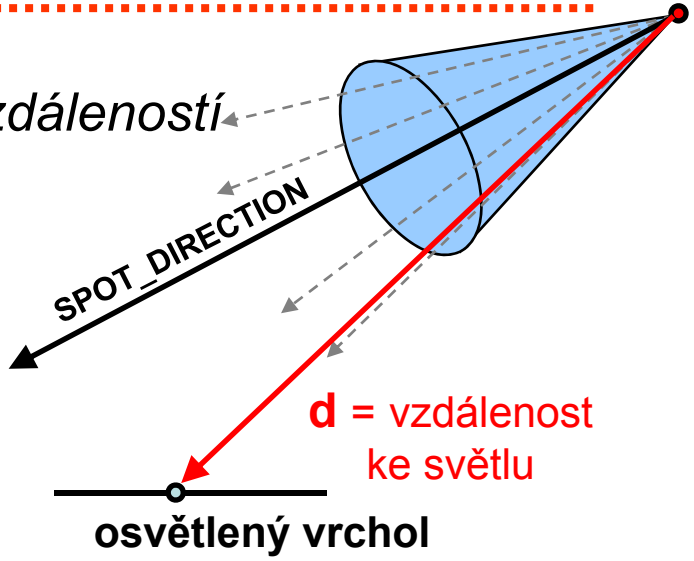
Světelné zdroje

V reálném prostředí klesá intensita světla se vzdáleností

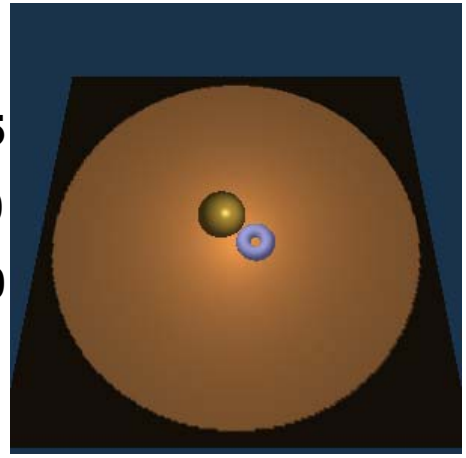
Zeslabení je modelováno faktorem:

attenuationFactor = 1.0 / (k_C+k_L*d + k_Q*d^2)

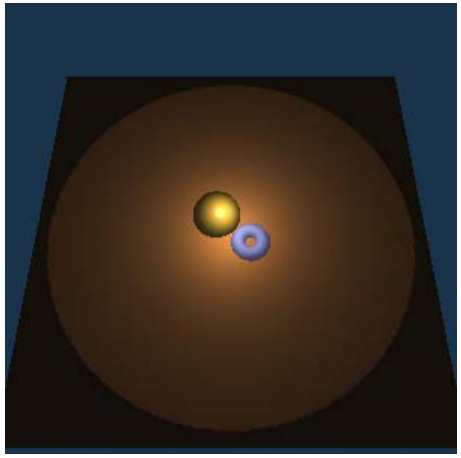
- **d** = vzdálenost mezi světlem a vrcholem
- **k_C** = constant attenuation
- **k_L** = linear attenuation
- **k_Q** = quadratic attenuation



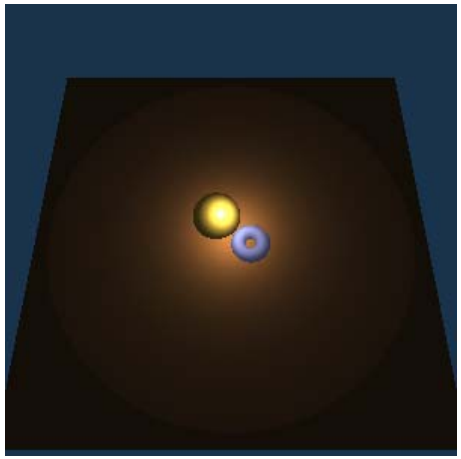
k_C = 1.5
k_L = 0.0
k_Q = 0.0



k_C = 0.0
k_L = 1.5
k_Q = 0.0



k_C = 0.0
k_L = 0.0
k_Q = 1.5

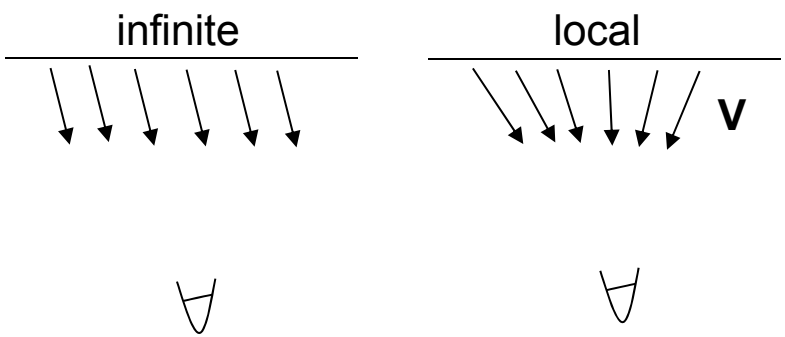


Výpočet směru ke světlu

Viewpoint („poloha pozorovatele“) při výpočtu vektoru V

– poloha pozorovatele ovlivňuje rychlost výpočtů zrcadlových odlesků

- **infinite viewpoint** $(0, 0, 1)_{eye}$ – směr mezi pozorovatelem a libovolným vrcholem scény zůstává konstantní
- **local viewpoint** vede k realističtějším výsledkům, směr k pozorovateli se počítá pro každý vrchol
  - výpočet (pozice_světla – pozice_bodu)
  - Optimální v souřadnicích kamery (*eye space) V = - pozice_bodu*

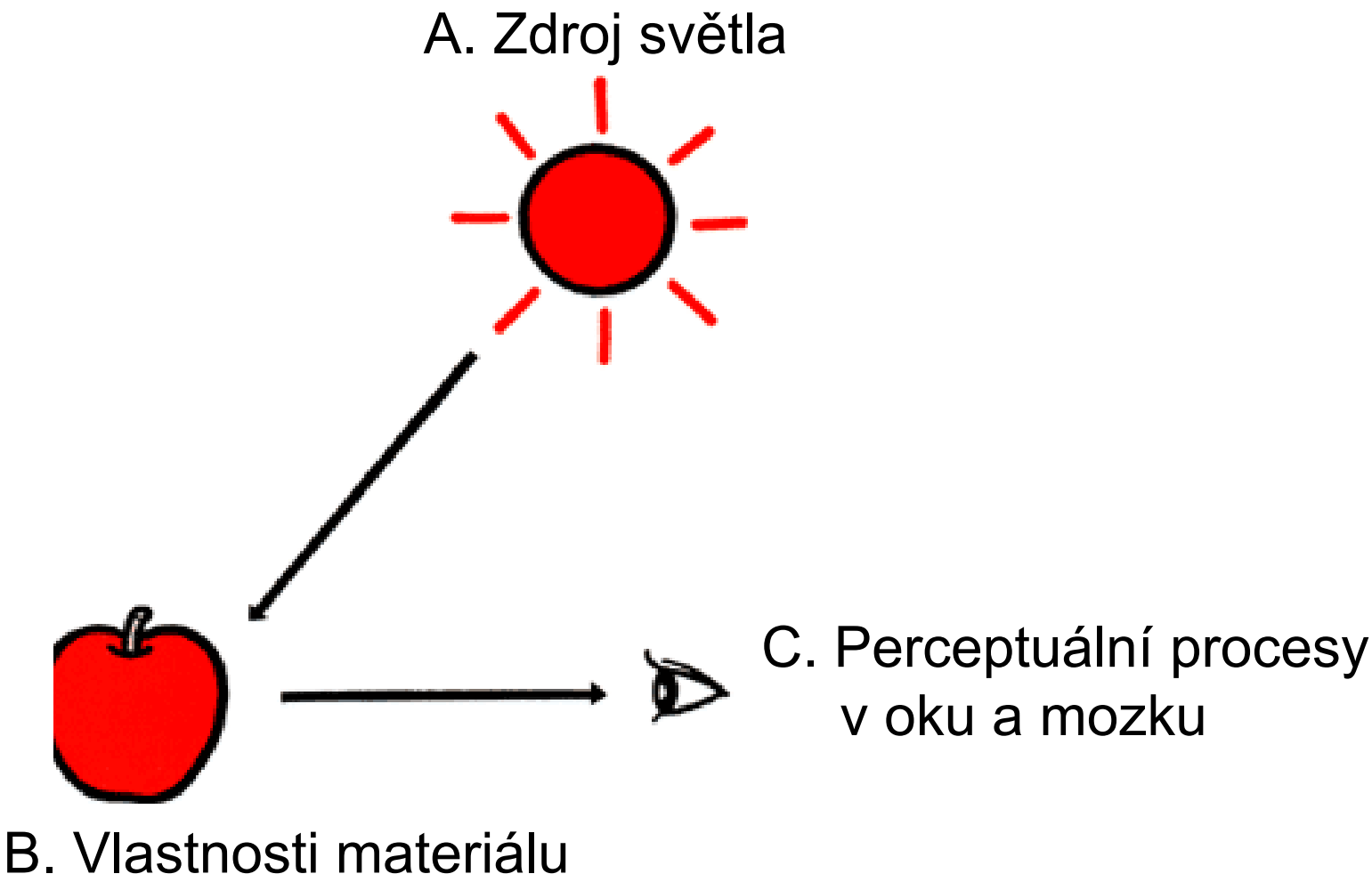


Definice světel, standardní notace a názvosloví

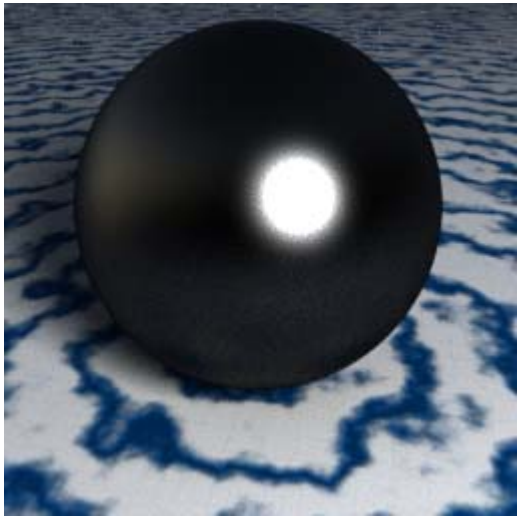
Jméno parametru	Iniciální hodnota	Význam
AMBIENT	(0.0,0.0,0.0,1.0)	ambientní složka barvy světla
DIFFUSE	(1.0,1.0,1.0,1.0)	difúzní složka barvy světla
SPECULAR	(1.0,1.0,1.0,1.0)	zrcadlová složka barvy světla
POSITION	(0.0,0.0,1.0,0.0)	(x, y, z, w) poloha světla
SPOT_DIRECTION	(0.0,0.0,-1.0)	(x, y, z) směr reflektoru
SPOT_EXPONENT	0.0	exponent reflektoru
SPOT_CUTOFF	180.0	hraniční úhel výřezu reflektoru
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.

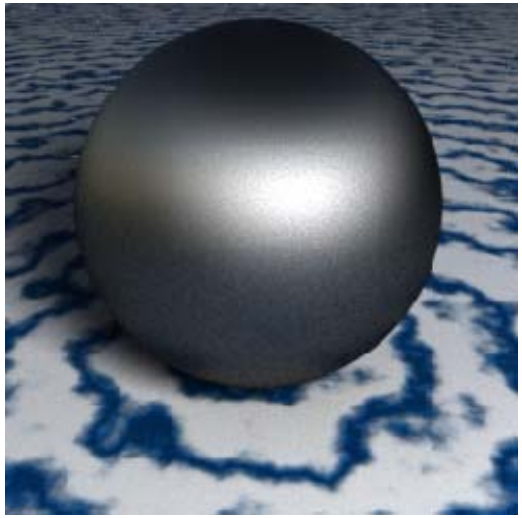
B. Vlastnosti materiálu

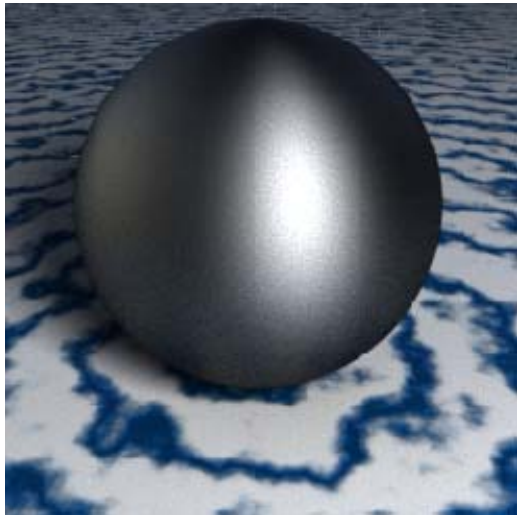


Izotropní a anizotropní odrazy



Izotropní





Anizotropní
- kartáčovaný kov, vlasy

[http://www.sidefx.com/docs/houdini12.1/vex/pbr]

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“)

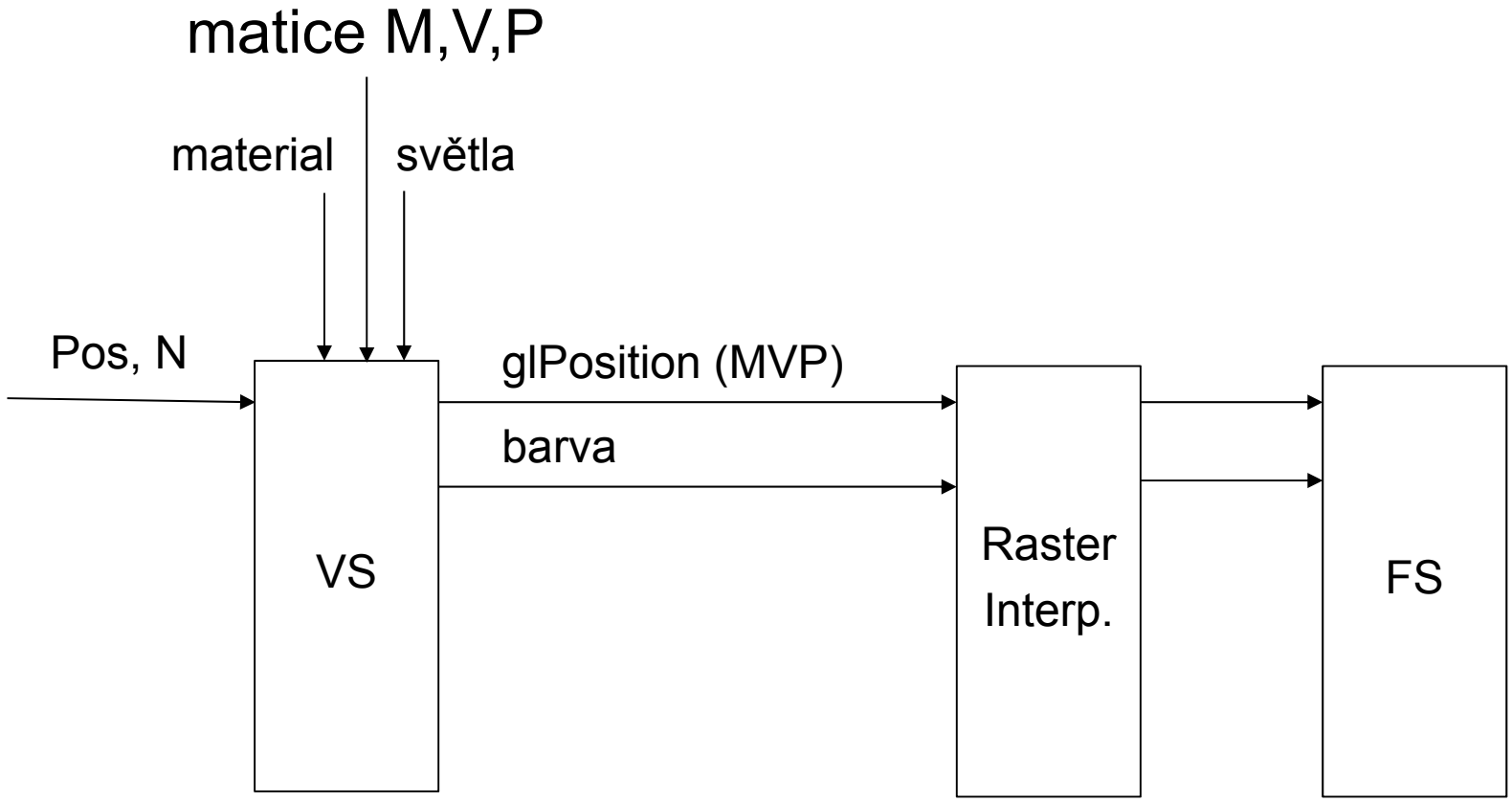
- Materiál povrchu = procento odražené pro jednotlivé složky R,G,B

Material examples

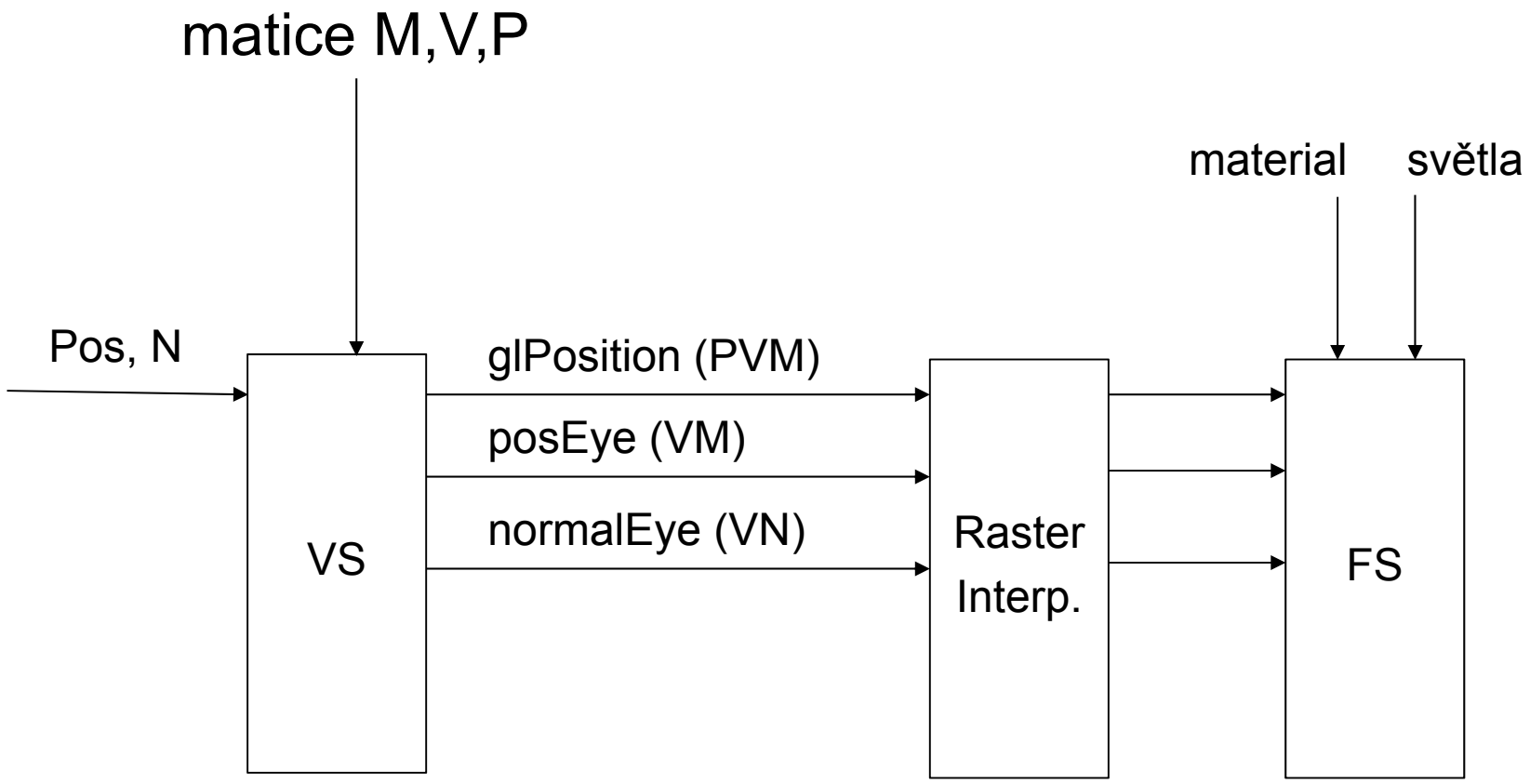
Material examples

```
//Zlato
GLfloat goldSpecular [] = {0.628281, 0.555802, 0.366065, 1.0};
GLfloat goldDiffuse [] = {0.75164, 0.60648, 0.22648, 1.0};
GLfloat goldAmbient [] = {0.24725, 0.1995, 0.0745, 1.0};
GLfloat goldShininess [] = {51.2};

//Cín
GLfloat pewterSpecular [] = {0.1059, 0.0588, 0.1137, 1.0};
GLfloat pewterDiffuse [] = {0.4275, 0.4706, 0.5412, 1.0};
GLfloat pewterAmbient [] = {0.3333, 0.3333, 0.5216, 1.0};
GLfloat pewterShininess [] = {9.85};

//mosaz
GLfloat brassSpecular [] = {0.3294, 0.2235, 0.0275, 1.0};
GLfloat brassDiffuse [] = {0.7804, 0.5686, 0.1137, 1.0};
GLfloat brassAmbient [] = {0.9922, 0.9412, 0.8079, 1.0};
GLfloat brassShininess [] = {27.89};
```

Osvětlení per vertex



Osvětlení per fragment



Příklad v OpenGL – část na CPU 1/2

```
// Příprava bufferů - navíc buffer s normálami
GLint m_norLoc = -1;
m_normalLoc = glGetAttribLocation(m_program, "normal");
glBindVertexArray( vao );    // inicializace
  // position
   ...
  // každý vrchol má navíc navíc atribut 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 );
```

Příklad v OpenGL – část na CPU 2/2

```
// Přidána uniformíní proměnná s inverzní transponovanou M
glm::mat4 VMmatrix = Vmatrix * Mmatrix;
glm::mat4 VNmatrix = Vmatrix * glm::transpose(
                                  glm::inverse(Mmatrix))
glUniformMatrix4fv(
  glGetUniformLocation(m_program, "VMmatrix"),
  1, false, glm::value_ptr(VMmatrix) );

glUniformMatrix4fv(
  glGetUniformLocation(m_program, "VNmatrix");
  1, false, glm::value_ptr(VNmatrix) );
```

```
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;  // pro rigidní transformace   V*M
uniform mat4 VNmatrix;  // pro nerigidní transformace V*(M^(-1))^T
uniform mat4 Pmatrix;
in vec3 position;       // object space
in vec3 normal;
smooth out vec3 thePosition; // camera space coordinates
smooth out vec3 theNormal;   // camera space normal

void main()
{
  thePosition = (VMmatrix * vec4(position, 1)).xyz;

  theNormal = (VMmatrix * vec4(normal, 0)).xyz;//pro rigidní transf.
  theNormal = (VNmatrix * vec4(normal, 0)).xyz;//pro nerigidní
  gl_Position = Pmatrix * thePosition; // clip space position
}
```

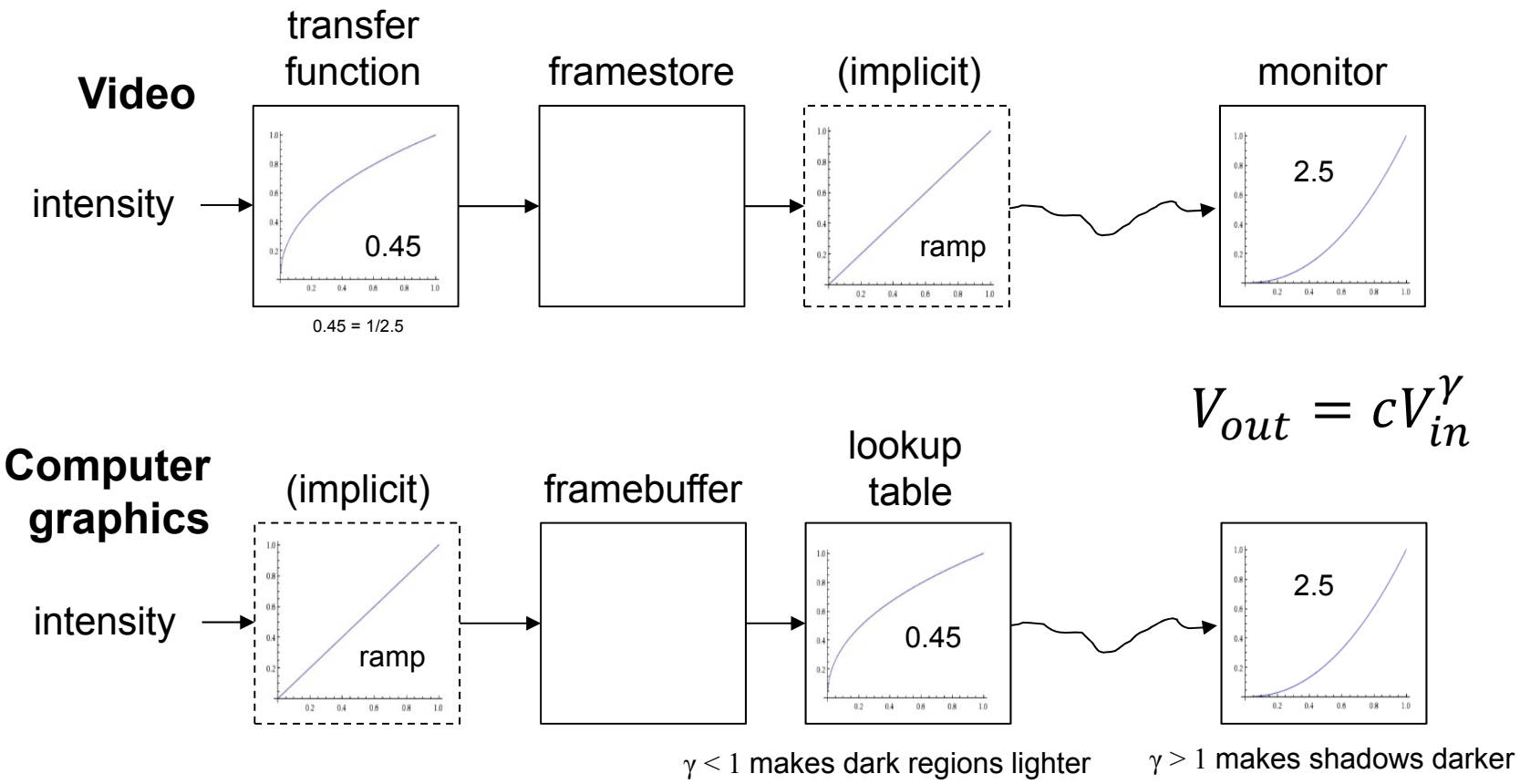
Příklad v OpenGL – Fragment shader

```
// Fragment shader for meshes of class MeshNode
#version 130
smooth in vec3 thePosition;
smooth in vec3 theNormal;
uniform vec3 pointlightPos;
uniform vec4 color;

out vec4 outputColor;

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

Gama korekce

- Počítačová grafika
  - počítá interakci světel a objektů
  - tyto interakce jsou ve fyzikální oblasti
  - počítají se lineárně
- Monitory zobrazují nelineárně
  - CRT televize jsou nelineární (nelinearita elektronového děla)
  - vstupní hodnota 0,5 se zobrazí jako 0,25 intenzita bodu (pro rozsah 0..1)
  - Moderní monitory tento standard zachovávají
  - Prvotním důvodem je zlepšení vizuální kvality video signálu z z kamery (odstup signál/šum)

Gama korekce

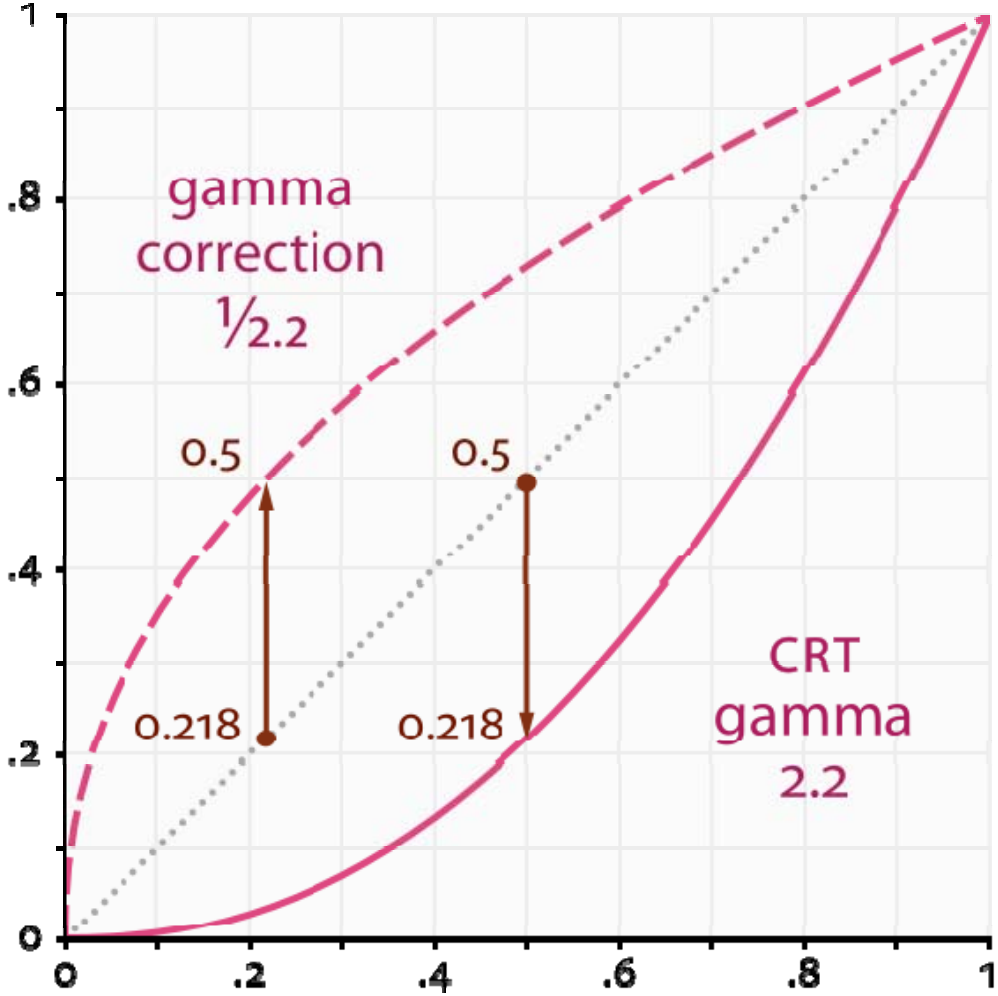


$$V_{out} = cV_{in}^{\gamma}$$

$\gamma < 1$ makes dark regions lighter

$\gamma > 1$ makes shadows darker

Gama korekce - příklad uložení hod. 0.218



$$V_{out} = cV_{in}^{\gamma}$$

[http://en.wikipedia.org/wiki/File:GammaFunctionGraph.svg]