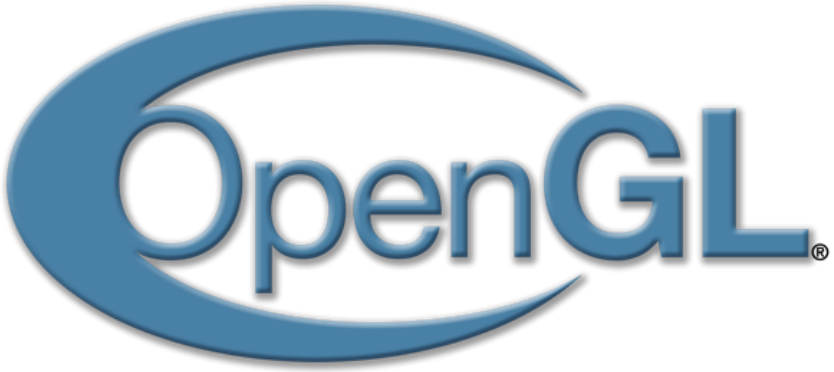


PV112 – Programování grafických aplikací

5. přednáška – Osvětlení a materiály

Osvětlení

- OpenGL již nepodporuje osvětlení pomocí vestavěných funkcí, je třeba vše nadefinovat
- Pozor na to, ve kterém souřadném systému budete osvětlení počítat
- Zásadní roli hrají normály

Normálové vektory

- Zásadní pro práci s osvětlením
- Pro trojúhelník lze spočítat pomocí vektorového součinu dvou jeho hran a následné normalizace

```
triangle (v1, v2, v3);
edge1 = v2 - v1;
edge2 = v3 - v1;
triangle.normal =
   cross(edge1, edge2).normalize();
```

Normálové vektory vrcholu

- Lze spočítat z normál okolních trojúhelníků

```
vertex v1, v2, v3, ....;
triangle tr1, tr2, tr3;
   // všechny sdílí vrchol v1
v1.normal = normalize(tr1.normal
        + tr2.normal + tr3.normal);
```

Normály v OpenGL

- Zachází se s nimi stejně jako s dalšími parametry vrcholů (pozice, barva, …)

```
GLuint normalbuffer;
glGenBuffers(1, &normalbuffer);
glBindBuffer(GL_ARRAY_BUFFER, normalbuffer);
glBufferData(GL_ARRAY_BUFFER, normals.size()
     * sizeof(glm::vec3), &normals[0],
     GL_STATIC_DRAW);
```

Normály v OpenGL

```
glEnableVertexAttribArray(2);
glBindBuffer(GL_ARRAY_BUFFER, normalbuffer);
glVertexAttribPointer(
     2,          // attribute
     3,          // size
     GL_FLOAT, // type
     GL_FALSE, // normalized?
     0,          // stride
     (void*)0  // array buffer offset
);
```

Normálová matice

- Matice 3x3 reprezentující inverzní transponovanou matici transformující vrcholy

- Jak je to možné?
  - Trocha matematiky…

Normálová matice

- Normálové vektory si představíme jako roviny kolmé k těmto vektorům – převedeme transformace normálových vektorů na transformace těchto kolmých rovin
- Homogenní rovina = řádkový vektor *(a, b, c, d)*, alespoň jedna z proměnných je nenulová
- *q* != 0, pak *(qa, qb, qc, qd)* reprezentuje stejnou rovinu

Normálová matice

- Bod $(x, y, z, w)^T$ leží v rovině (a, b, c, d), pokud $ax+by+cz+dw = 0$
  - $w = 1$, jde o rovnici Euklidovské roviny (jedna z hodnot a, b, c musí být nenulová)
  - $(0, 0, 0, d)$ reprezentuje nekonečnou rovinu obsahující všechny body v nekonečnu

Normálová matice

- Je-li **p** homogenní rovina a **v** homogenní vrchol, pak **v** leží na **p** zapíšeme jako

$$\boldsymbol{pv} = 0$$

- Vezměme transformaci $\boldsymbol{M}$ (matice 4x4 mající inverzní matici $\boldsymbol{M^{-1}}$)
- $\boldsymbol{pv} = 0$ je ekvivalentní k $\boldsymbol{pM^{-1}Mv} = 0$
  - tedy $\boldsymbol{Mv}$ leží v rovině $\boldsymbol{pM^{-1}}$
- $\boldsymbol{pM^{-1}}$ je tedy obrazem roviny $\boldsymbol{p}$ transformované pomocí $\boldsymbol{M}$

Normálová matice

- Reprezentujeme-li normálový vektor pomocí vektoru, pak uvažujme vzájemně kolmé vektory $\boldsymbol{v}$ a $\boldsymbol{n}$, pak

$$\boldsymbol{n}^T\boldsymbol{v} = 0$$

- Pro libovolnou transformaci $\boldsymbol{M}$ máme

$$\boldsymbol{n}^T\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{v} = 0$$

  - $\boldsymbol{n}^T\boldsymbol{M}^{-1}$ je transponovaná matice transformovaného normálového vektoru $(\boldsymbol{M}^{-1})^T\boldsymbol{n}$
- Normálové vektory jsou tedy transformovány inverzní transponovanou maticí transformace aplikované na vrcholy

Osvětlení a materiály

- Phongův osvětlovací model
  - Rychlý výpočet osvětlení
  - Výsledný obrázek se snaží co nejvíce přiblížit realitě
- 3 světelné složky:
  - Ambientní
  - Difúzní
  - Spekulární

Ambientní složka

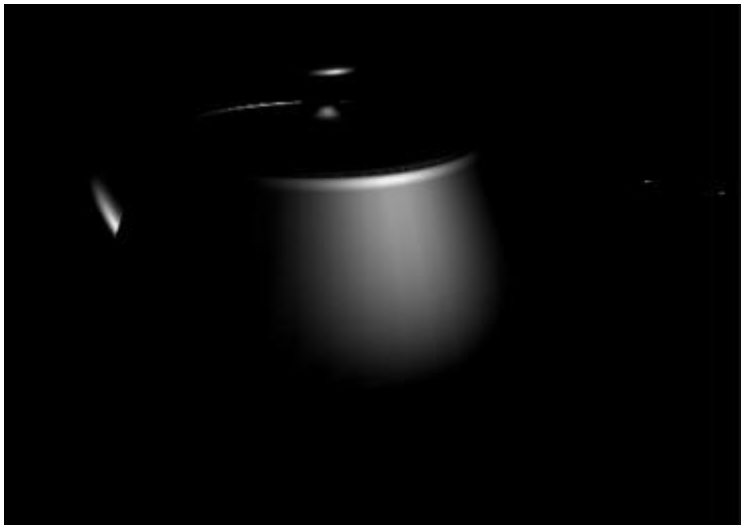
- Všesměrové světlo – nezávislé na poloze a orientaci zdrojů světla
- Objekt se jeví jako plošný – ztrácíme prostorový vjem

Difúzní složka

- Světlo dopadající na povrch objektu z konkrétního světelného zdroje
- Od povrchu objektu se odráží do všech směrů se stejnou intenzitou – matné materiály
- Poloha kamery nemá vliv na barvu ani intenzitu difúzní složky

Spekulární složka

- Paprsek světla se od povrchu tělesa odráží podle zákona odrazu a lomu
- Paprsek se neodráží ideálně – bereme v úvahu koeficient změny intenzity odraženého světla



Kombinace světelných složek

Materiály v OpenGL

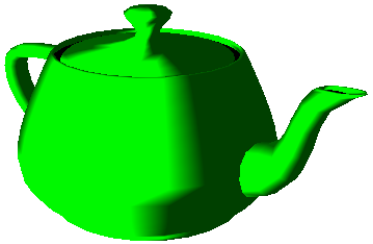
- Používají se často v kombinaci se světly
- Pro výpočet difúzní a spekulární složky světla nutné znát normály ve vrcholech
- Vlastnost materiálu se může měnit pro přední a zadní stranu vykreslovaných polygonů (musíme si to ale nadefinovat sami)

Materiály

- Objekt může mít různý materiál uvnitř a vně (musíme si nadefinovat sami)
- Difúzní složka je pro rozpoznávání barev nejdůležitější
- Difúzní odrazivost (reflectance) nezávisí na pozici pozorovatele
- Spekulární odrazivost závisí na pozici pozorovatele
- V reálném světě je difuzní a ambientní světlo ekvivalentní

Typy světel v OpenGL

- Defaultně nejsou podporována žádná světla, mezi nejjednodušší typy patří:
  - Směrová
  - Bodová
  - Reflektorová



Směrová světla

- Nemají žádný pevně daný světelný zdroj
- Šíří se z jednoho směru nezávisle na vzájemné poloze a vzdálenosti objektů ve scéně
- Např. slunce – paprsky téměř paralelní

Ambientní složka směrového světla

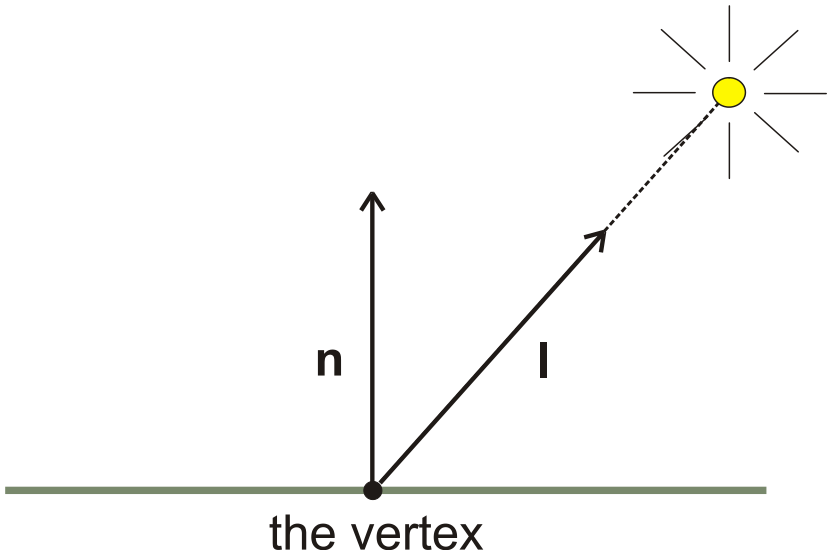
```
ambient = light_ambient_color *
material_ambient_color
```

Jak je počítána barva?

$$ambient = ambient_{light} * ambient_{material}$$

$$diffuse = (max\{\mathbf{l}.\mathbf{n},0\})\ diffuse_{light} * diffuse_{material}$$

- difúze nezáleží na pozici pozorovatele
- difúze je největší, když světlo dopadá kolmo na plochu

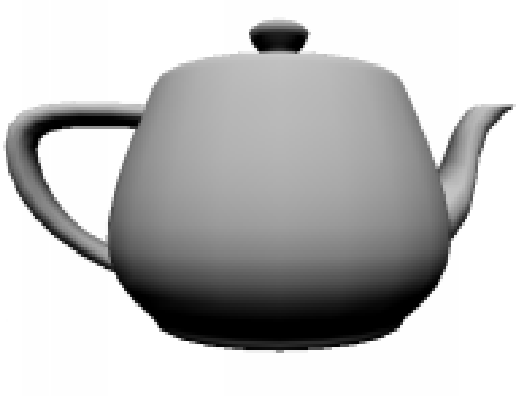


Vertex shader

```
in vec4 position;
in vec3 normal;

out vec3 N;
out vec3 v;

uniform mat4 myModelViewMatrix;
uniform mat4 myModelViewProjectionMatrix;
uniform mat3 myNormalMatrix;

void main()
{
	v = vec3(myModelViewMatrix * position);
	N = normalize(myNormalMatrix * normal);

	gl_Position = myModelViewProjectionMatrix * position;
}
```

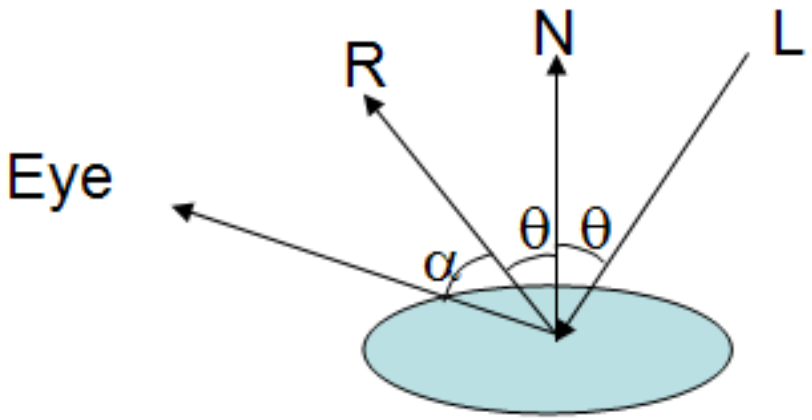
Fragment shader

```
in vec3 N;
in vec3 v;
out vec4 final_color;

uniform vec3 material_diffuse_color;
uniform vec4 light_position;
uniform vec3 light_diffuse_color;

void main()
{
vec3 L = normalize(light_position.xyz); // Toto platí pro
             směrové světlo, tedy za předpokladu, že
             light_position je místo v nekonečnu, ODKUD světlo
             svítí
      vec3 normN = normalize(N);
      vec3 Idiff = light_diffuse_color * material_diffuse_color
             * max(dot(normN, L), 0.0);
      Idiff = clamp(Idiff, 0.0, 1.0);

      final_color = vec4(Idiff, 1.0);
}
```



Spekulární složka směrového světla

- Phongův model



Spekulární složka směrového světla

- L = vektor směřující od zdroje světla do vrcholu na povrchu objektu
- N = normálový vektor
- Eye = vektor směřující od vrcholu do oka pozorovatele (nebo kamery)
- R = vektor odrazu (světla ze směru vektoru L)
- Spekulární složka je úměrná kosinu úhlu α

Spekulární složka směrového světla

- Pokud má Eye vektor stejný směr jako R, je intenzita spekulárního odrazu největší
- Úbytek intenzity (útlum) je úměrný vektoru mezi Eye a R
- Útlum je řízen parametrem *shininess*

Shininess = 8

Shininess = 64

Shininess = 128

Spekulární složka směrového světla

- Rovnice pro vektor odrazu

$$R = -2N(L \cdot N) + L$$

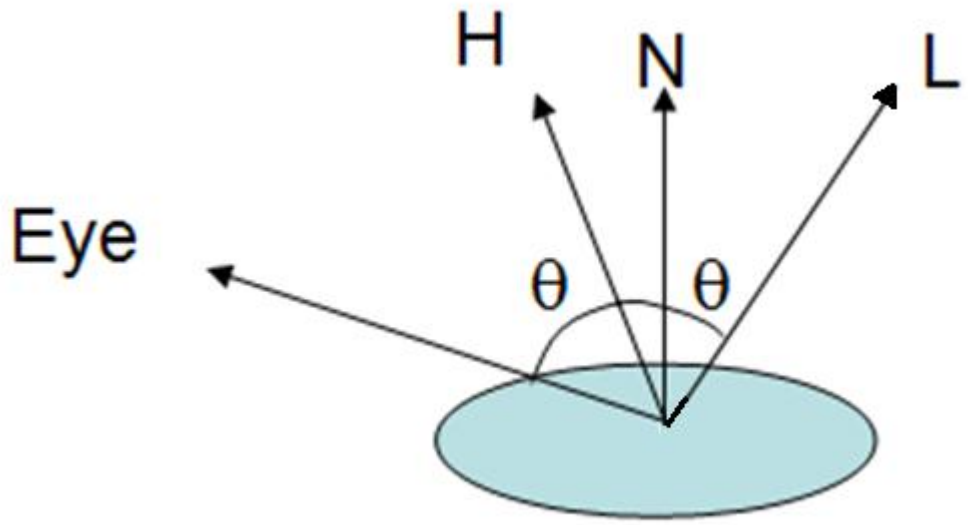
- Spekulární složka Phongova modelu:

$$Spec = (R \cdot Eye)^s * L_s * M_s$$

- Exponent *s* = hodnota shininess
- L_s = intenzita spekulárního světla
- M_s = spekulární koeficient materiálu

Blinn-Phongův model

- Zjednodušený Phongův model
- Využívá tzv. half-vektor = vektor se směrem mezi Eye vektorem a vektorem světla



Blinn-Phongův model

- Intenzita spekulární složky = *cos* úhlu mezi half-vektorem a normálou povrchu

$$H = Eye + L$$

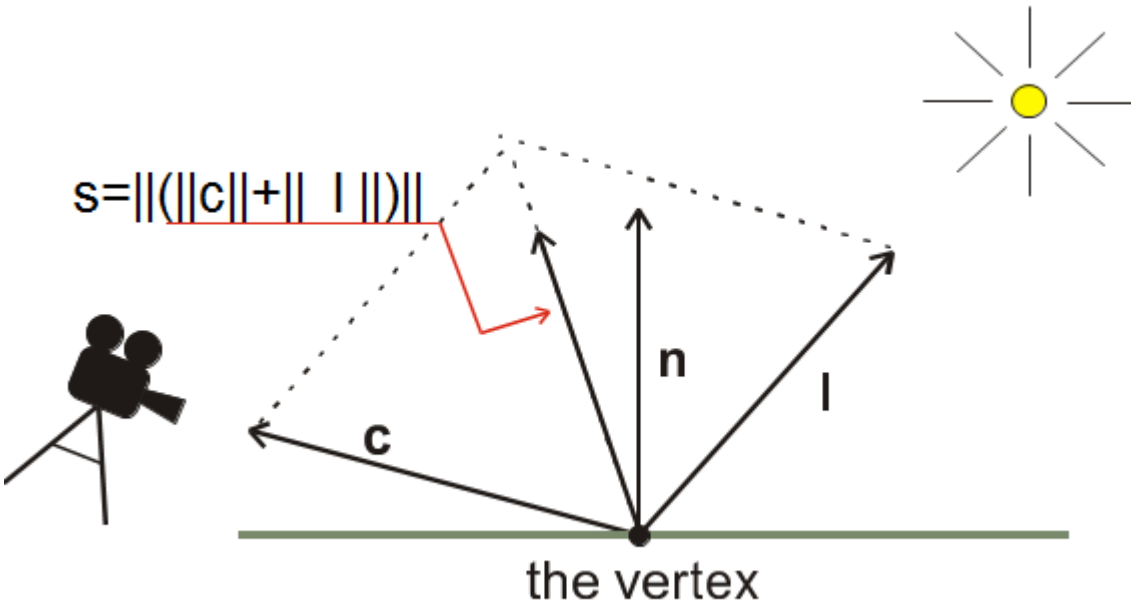
- Vektory musí být normalizovány
- Spekulární složka Blinn-Phongova modelu:

$$Spec = (N \cdot H)^s * L_s * M_s$$

Osvětlení – jak je počítána barva?

specular = (max{**s.n**,0}) $^{\text{shininess}}$ specular $_{\text{light}}$ * specular $_{\text{material}}$

- spekulární odraz záleží na pozici pozorovatele



Bodová světla

- Point light – žárovka v prostoru, která vyzařuje světlo na všechny strany se stejnou intenzitou
- Intenzita klesá se vzdáleností
- Dva parametry:
  - *w* = pozice světla (hodnota 0 odpovídá směrovému světlu)
  - *attenuation* = útlum světla (konstatní, lineární, kvadratický)

Útlum

- Snižování intenzity světla se vzrůstající vzdáleností od světelného zdroje
- Jednoduchá simulace kombinuje konstantní c_1, lineární c_2 a kvadratický c_3 útlum
- Tyto koeficienty jsou kombinovány v rovnici:

$$attenuation = \frac{1}{\left(c_1 + c_2 d + c_3 d^2\right)}$$

- d = vzdálenost světla od vrcholu

Jak je počítána barva?

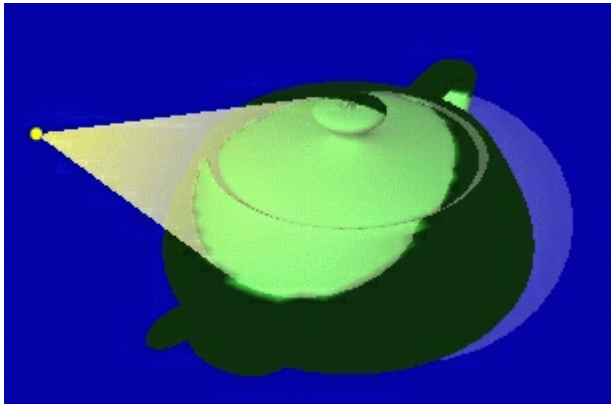
- Pro červenou, zelenou, a modrou komponentu odděleně (Toto je zjednodušení! V reálném světě je spektrum spojité.)
- Ambientní, difuzní, a spekulární části jsou sečteny.
- Poté jsou sevřeny (clamped) do rozsahu [0,1].

$$\text{color} = \text{global ambient}_{\text{light}} * \text{ambient}_{\text{material}} + \Sigma \text{ light}$$

$$\text{light} = \text{attenuation} * (\text{ambient} + \text{diffuse} + \text{specular})$$

$$\text{attenuation} = 1/(\text{const} + d * \text{linear} + d^2 * \text{quadratic})$$

Reflektorová světla

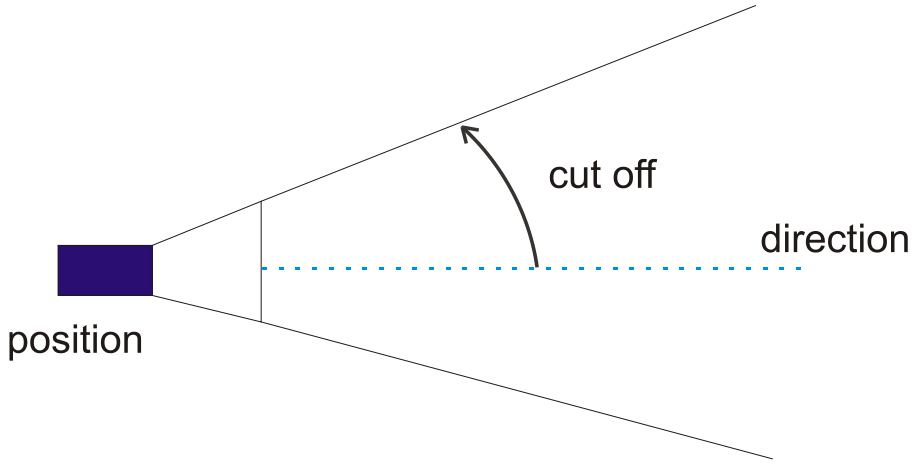


- Nejsložitější světlo
- Světlo se šíří z jednoho bodu pouze v předem zadaném kuželu
- Musíme zadat více parametrů:
  - Pozice světla
  - Směr šíření světelných paprsků
  - Úhel šíření světla, kde intenzita světla postupně klesá směrem k nule

Reflektorová světla

`spotDirection` směr reflektor. světla (0,0,-1)

`spotCosCutoff` cos úhlu ořezání refl. světla



`spotExponent` jak se intenzita světla snižuje od středu světelného kužele k jeho stěnám

Reflektorová světla

- Útlum se musí vynásobit následujícím efektem:

$$spotEffect = (spotDirection \cdot lightDir)^{spotExp}$$

- *spotDirection* = definovaný parametr světla
- *lightDir* = vektor od zdroje světla k vrcholu
- *spotExp* = útlum v rámci kužele světla
  (parametr *spotExponent* světla)

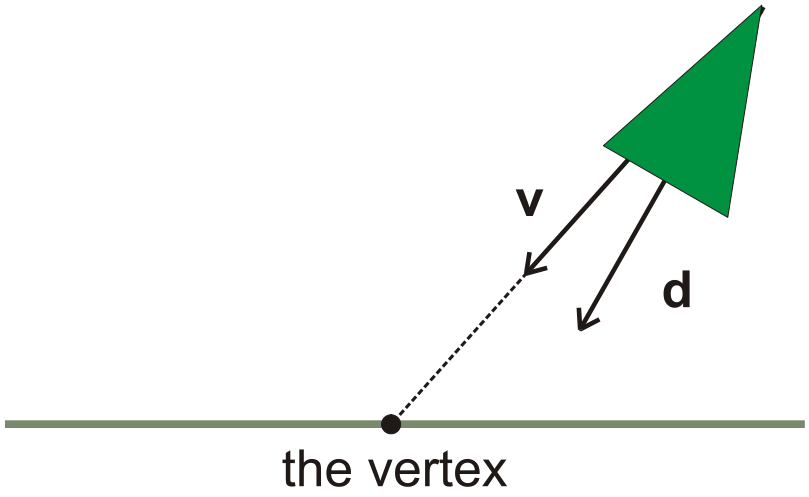
Jak je počítána barva?

spot = 1 jestliže světlo není reflektorové (CUTOFF je 180^o)

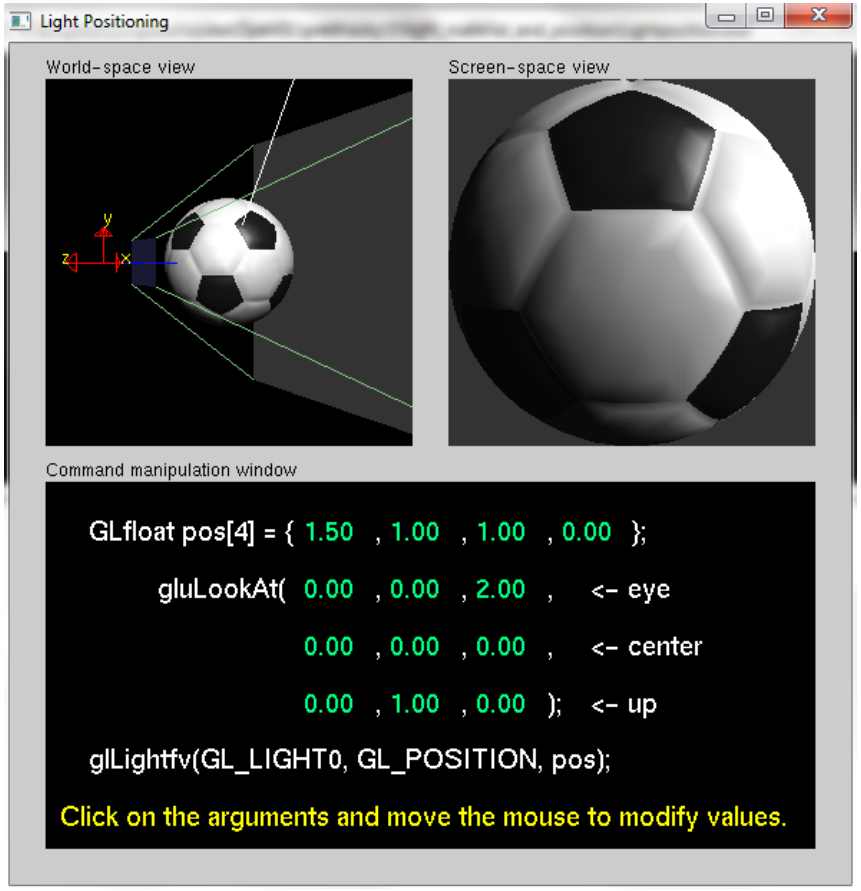
0 jestliže je bodové a vrchol je mimo kužel

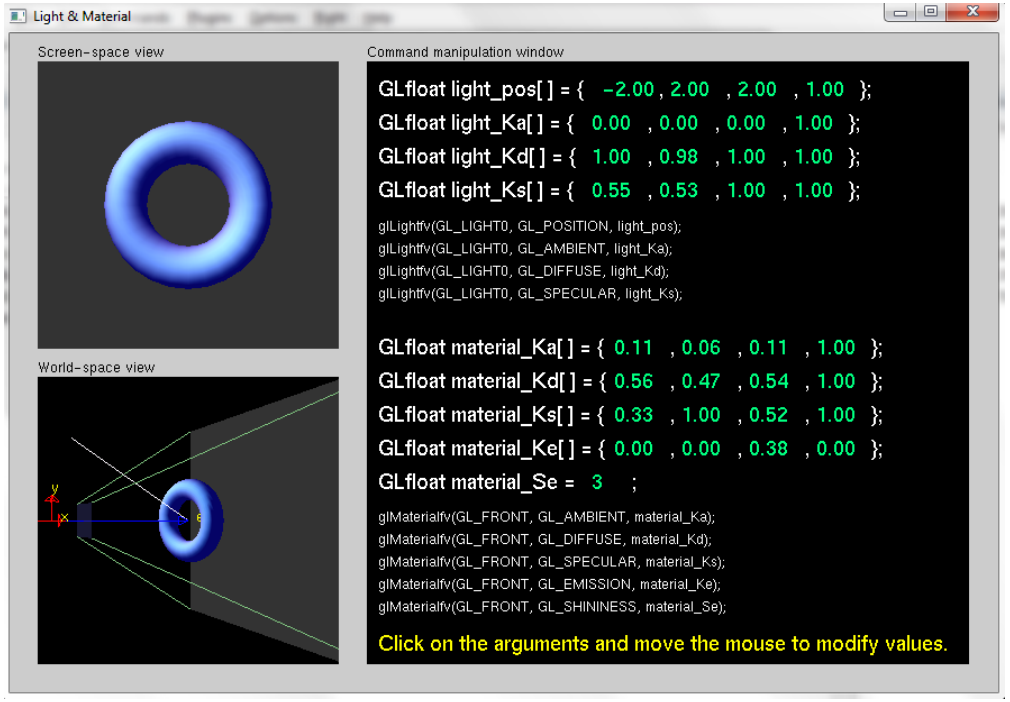
$\max\{\mathbf{v}.\mathbf{d},0\}^{GL\_SPOT\_EXPONENT}$

- ve skutečnosti se provádí kontrola, jestli je vrchol mimo kužel a provádí skalární součin dvou vektorů



Interaktivní ukázka





Toon shading

- Nefotorealistické zobrazení využívající často pouze tóny jedné barvy

- Výpočet kosinu úhlu mezi směrem světla a normálou povrchu

Toon shading

- Ve vertex shaderu:
  - Směr světla

```
uniform vec3 lightDir;
```

- Uložení výpočtu kosinu úhlu do proměnné *intensity*:

```
intensity = dot(lightDir, gl_Normal);
```

Toon shading

- Fragment shader – spočtení barvy fragmentu s ohledem na intenzitu světla

```
vec4 color;

if (intensity > 0.95)
        color = vec4(1.0,0.5,0.5,1.0);
else if (intensity > 0.5)
        color = vec4(0.6,0.3,0.3,1.0);
else if (intensity > 0.25)
        color = vec4(0.4,0.2,0.2,1.0);
else
        color = vec4(0.2,0.1,0.1,1.0);
```

Výsledek

Materiály

Material	AMBIENT	DIFFUSE	SPECULAR	SHININESS
Brass	0.329412 0.223529 0.027451 1.0	0.780392 0.568627 0.113725 1.0	0.992157 0.941176 0.807843 1.0	27.8974
Bronze	0.2125 0.1275 0.054 1.0	0.714 0.4284 0.18144 1.0	0.393548 0.271906 0.166721 1.0	25.6
Polished Bronze	0.25 0.148 0.06475 1.0	0.4 0.2368 0.1036 1.0	0.774597 0.458561 0.200621 1.0	76.8
Chrome	0.25 0.25 0.25 1.0	0.4 0.4 0.4 1.0	0.774597 0.774597 0.774597 1.0	76.8
Copper	0.19125 0.0735 0.0225 1.0	0.7038 0.27048 0.0828 1.0	0.256777 0.137622 0.086014 1.0	12.8

Material	AMBIENT	DIFFUSE	SPECULAR	SHININESS
Polished Copper	0.2295 0.08825 0.0275 1.0	0.5508 0.2118 0.066 1.0	0.580594 0.223257 0.0695701 1.0	51.2
Gold	0.24725 0.1995 0.0745 1.0	0.75164 0.60648 0.22648 1.0	0.628281 0.555802 0.366065 1.0	51
Polished Gold	0.24725 0.2245 0.0645 1.0	0.34615 0.3143 0.0903 1.0	0.797357 0.723991 0.208006 1.0	83.2
Pewter	0.105882 0.058824 0.113725 1.0	0.427451 0.470588 0.541176 1.0	0.333333 0.333333 0.521569 1.0	9.84615