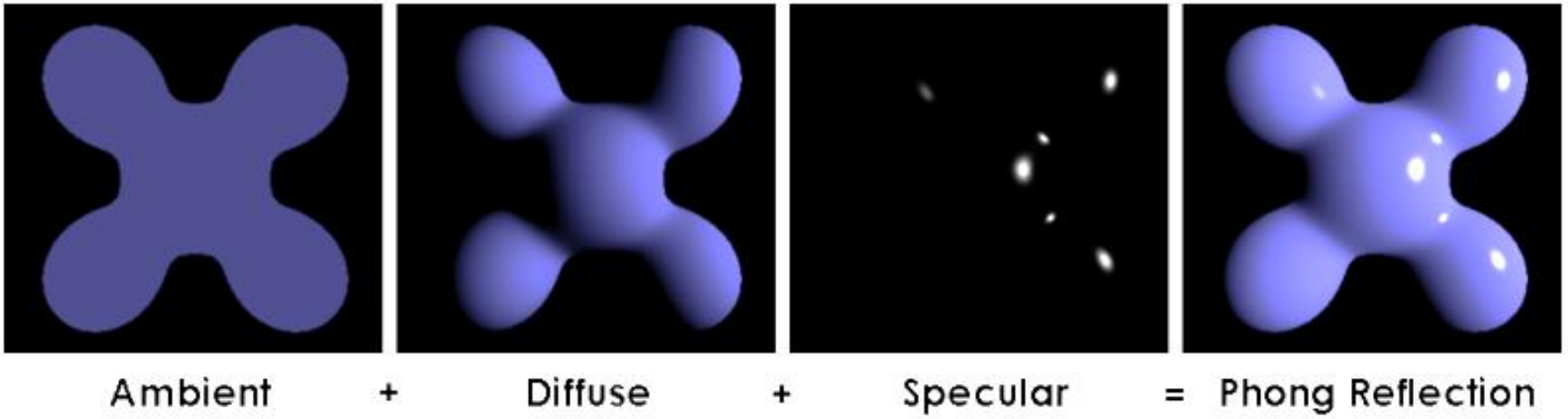
DCGI

Cvičení III

Implementace Phongova osvětlovacího modelu

Phongův osvětlovací model



- intensita a barva světla
  - **ambient**$_{light}$ *ambient intensity of light (RGB color)*
  - **diffuse**$_{light}$ *diffuse intensity of light (RGB color)*
  - **specular**$_{light}$ *specular intensity of light (RGB color)*
- parametry materiálu
  - **ambient**$_{material}$ *ambient color of material (RGB color)*
  - **diffuse**$_{material}$ *diffuse color of material (RGB color)*
  - **specular**$_{material}$ *specular color of material (RGB color)*
  - **shininess**$_{material}$ *specular exponent (float)*

Co potřebujeme pro výpočet?

- parametry materiálu
- parametry světel
- další důležité vstupní parametry:

$\vec{n}$ normálový vektor

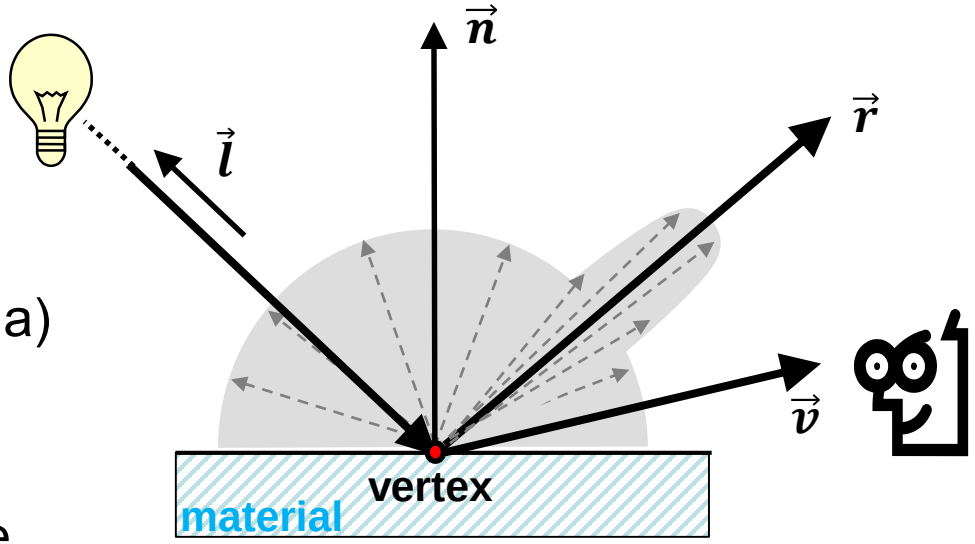
$\vec{l}$ směr ke světlu

$\vec{v}$ směr ke kameře

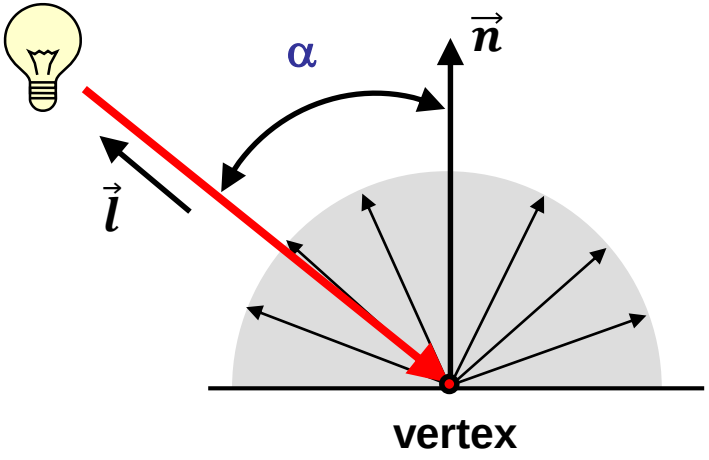
($\vec{r}$ směr odraženého světla)

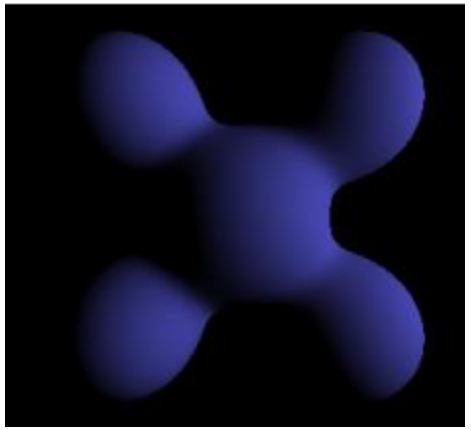
jednotkové vektory

→ jednodušší implementace



Difúzní složka

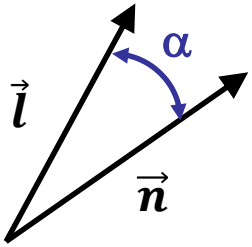




osvětlení nemůže být záporné
(může nastat pro odvrácené plošky)
→ max(, 0)

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

jak určit $\cos\alpha$?



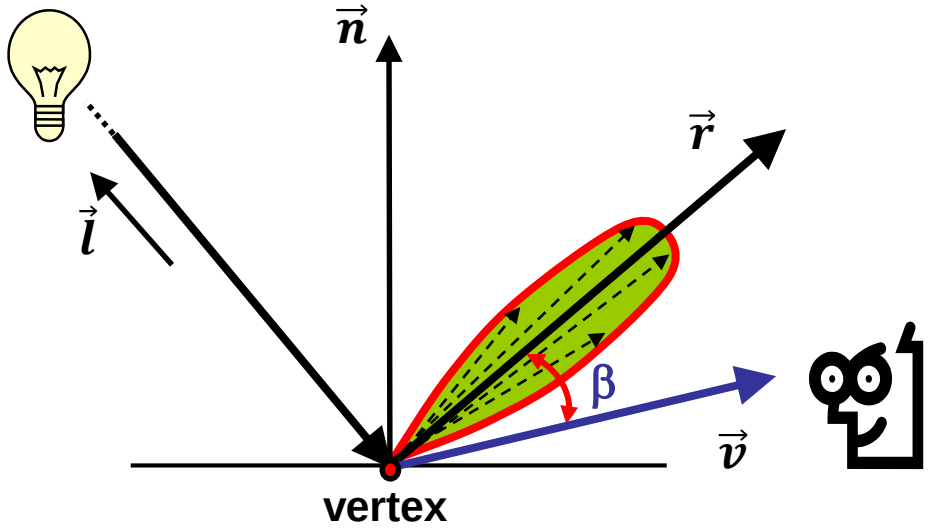
$$\cos\alpha = \frac{\vec{l} \,.\, \vec{n}}{\|\vec{l}\|\|\vec{n}\|}$$

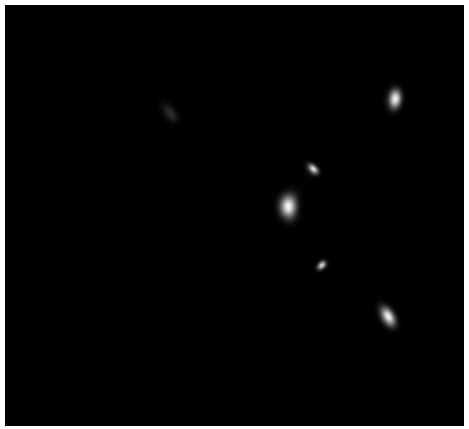
jednotkové vektory

$$\|\vec{l}\| = 1 \text{ a } \|\vec{n}\| = 1$$

$$\cos\alpha = \vec{l} \,.\, \vec{n}$$

Zrcadlová složka

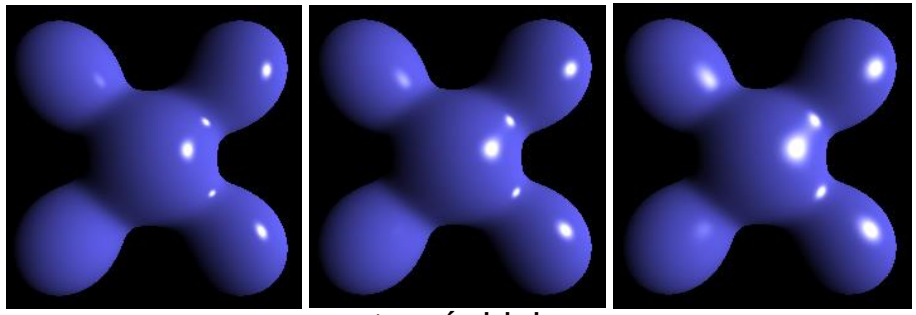




$$specular_{reflected} = (max[\cos \beta, 0])^{shininess\_material} * specular_{light} * specular_{material}$$

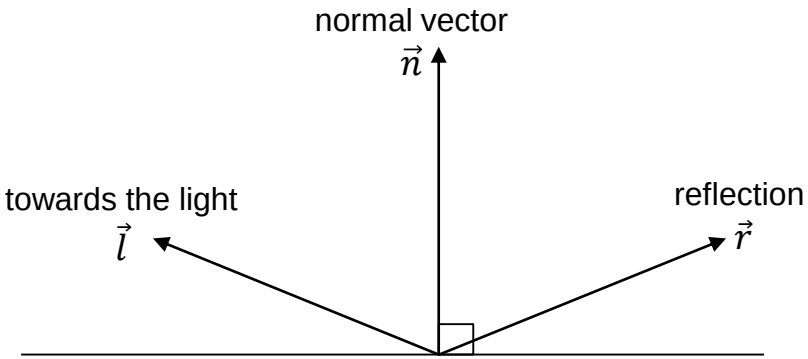
jednotkové vektory

$$\cos \beta = \vec{r} \,.\, \vec{v}$$



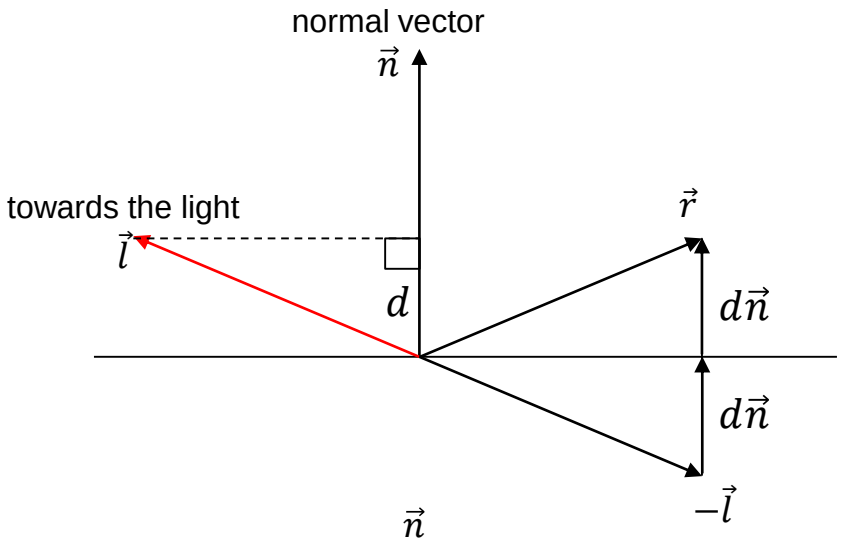
rostoucí $shininess_{material}$

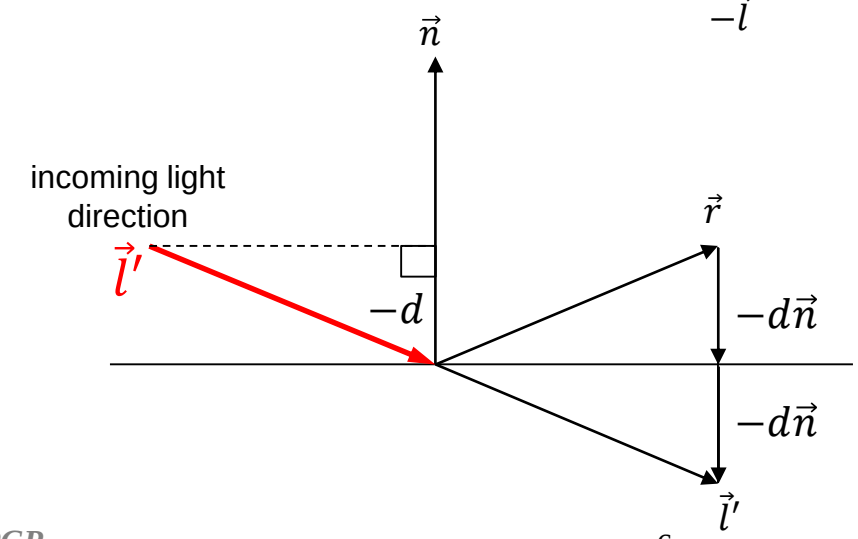
Jak určit směr odraženého světla $\vec{r}$?



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

vestavěná funkce GLSL
reflect($\vec{l}', \vec{n}$) používá směr dopadajícího paprsku světla $\vec{l}'$
$$\vec{r} = \vec{l}' - 2\,\mathrm{dot}\left(\vec{l}', \vec{n}\right)\vec{n}$$





Složení složek osvětlení

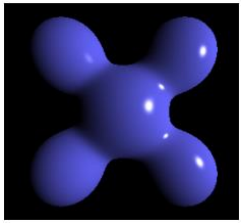
- ambientní složka

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

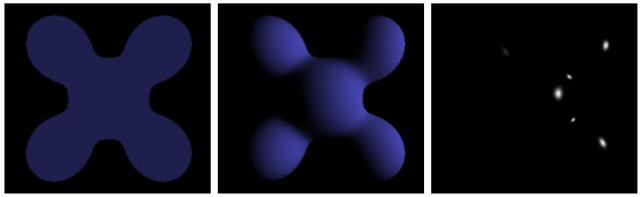
osvětlení vrcholu

$$barva\ vrcholu = \sum light_i$$

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



ovlivňuje intensitu světla v závislosti na typu zdroje



Jak předat parametry do shaderů?

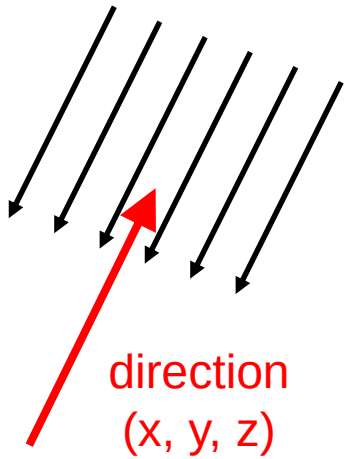
```
struct Material {
  vec3  ambient;          // ambient component
  vec3  diffuse;          // diffuse component
  vec3  specular;         // specular component
  float shininess;        // sharpness of specular reflection
};
uniform Material material;    // current material
```

```
struct Light {
  vec3  ambient;          // intensity & color of the ambient component
  vec3  diffuse;          // intensity & color of the diffuse component
  vec3  specular;         // intensity & color of the specular component
  vec3  position;         // light position for the point light or direction to directional light
  vec3  spotDirection;    // spotlight direction
  float   spotCosCutOff; // cosine of the spotlight's half angle
  float   spotExponent;  // distribution of the light energy within the reflector's cone
};
```

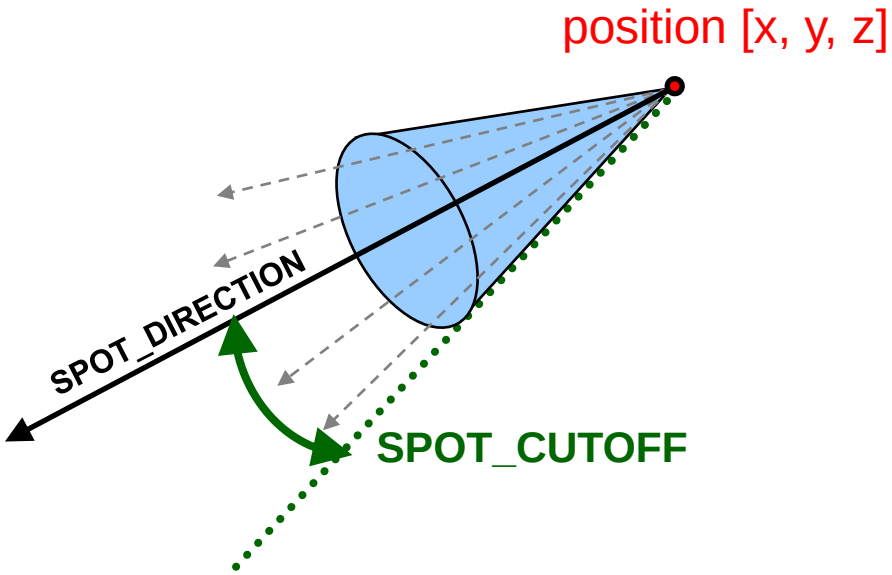
```
uniform float time;                  // time used for simulation of moving lights
in vec3 position;                    // vertex position in world space -> VS:vertexPosition in camera sp.
in vec3 normal;                      // vertex normal                  -> VS: vertexNormal
uniform vec3 reflectorPosition;      // reflector position (world coordinates)
uniform vec3 reflectorDirection;     // reflector direction (world coordinates)
```

Parametry světel

směrové světlo
(např. Slunce)



spot light
(***reflektor***)



POSITION
[x, y, z, 0]

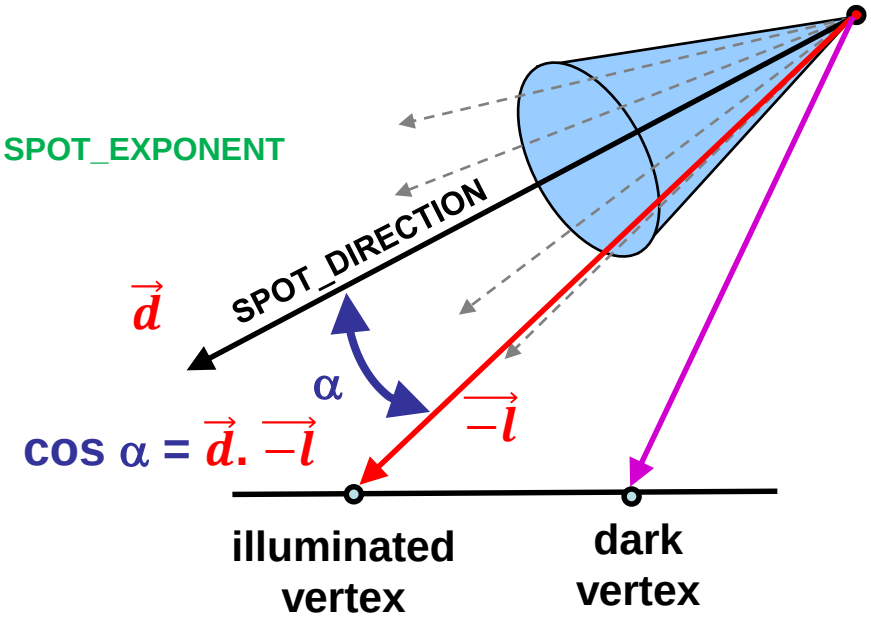
POSITION
[x, y, z, 1]

pozice světla v homogenních souřadnicích

Osvětlení reflektorem

hodnota spotlightEffect

- *paprsek mimo kužel* **0.0**
- *paprsek uvnitř* **(max {cos α, 0})**



jak rozlišit paprsky?

- uvnitř světelného kuželu

$$cos\ \alpha \geq cos(\text{SPOT\_CUTOFF})$$

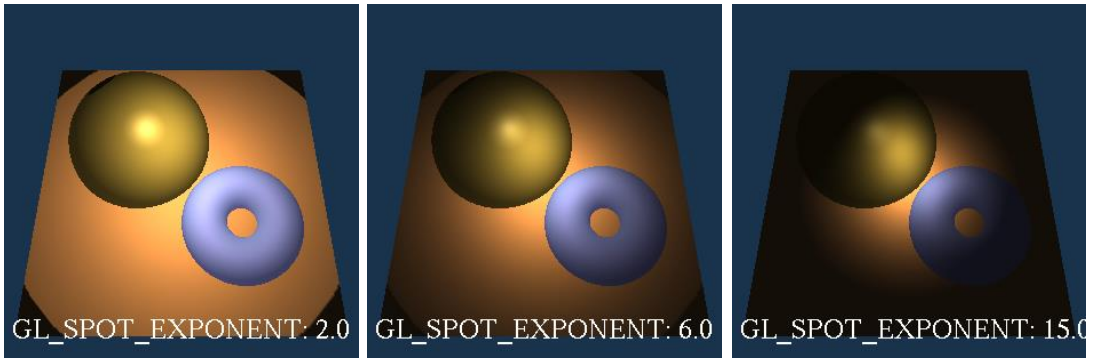
parametr **SPOT_EXPONENT**

→

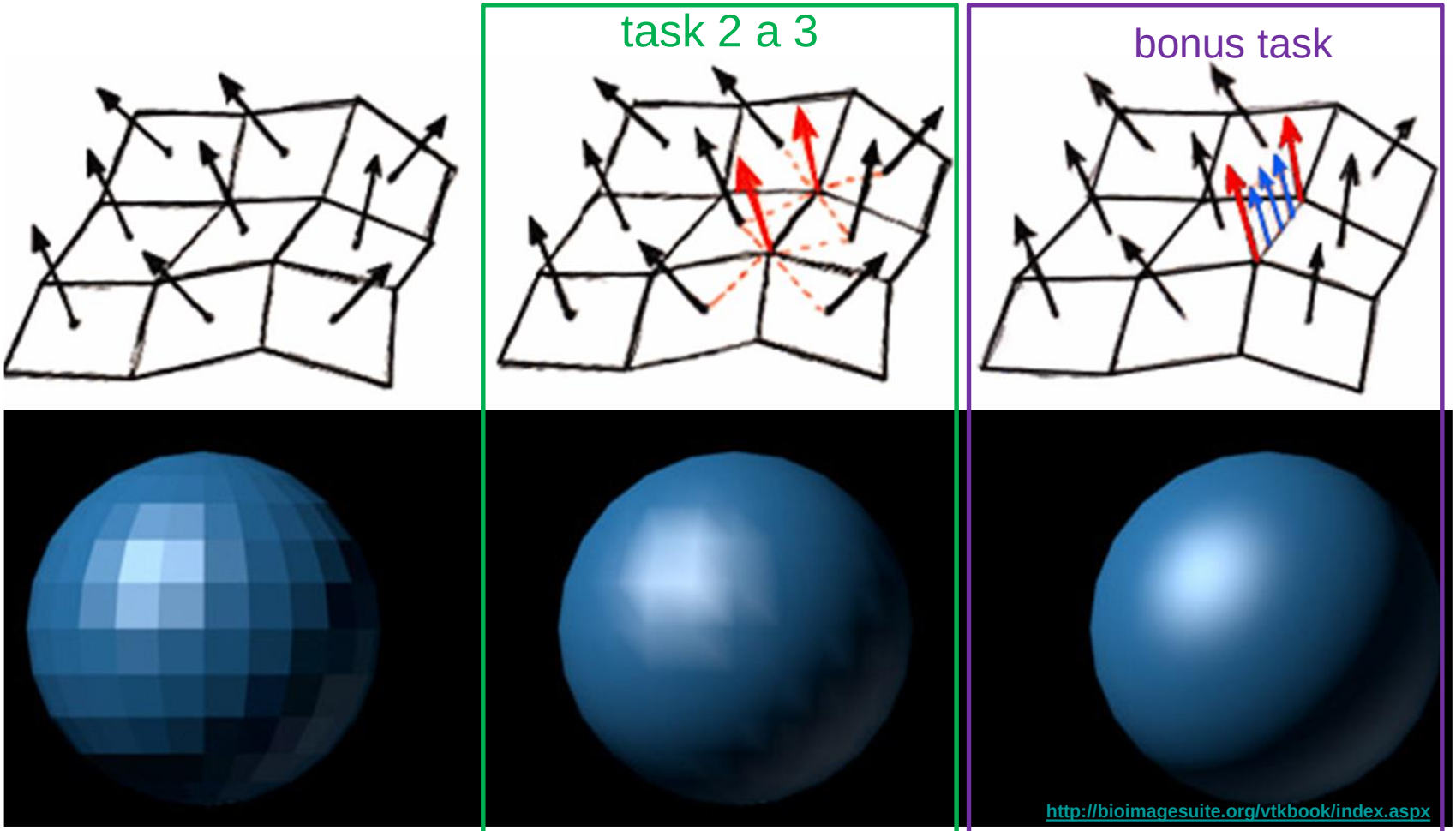
určuje rozložení intensity světla uvnitř reflektoru

→

pokles od středu ke kraji



Stínovací modely



$\vec{n}$ per face

$\vec{n}$ per vertex

$\vec{n}$ per fragment

Úlohy

- osvětlení počítáno v prostoru kamery
  - kamera je v počátku → $\vec{v} = -vertexPosition$

- osvětlení počítáno v každém vrcholu → barva vrcholu
- barva fragmentu určena interpolací barev vrcholů

per vertex lighting

Úloha 1: seznámení s kódem

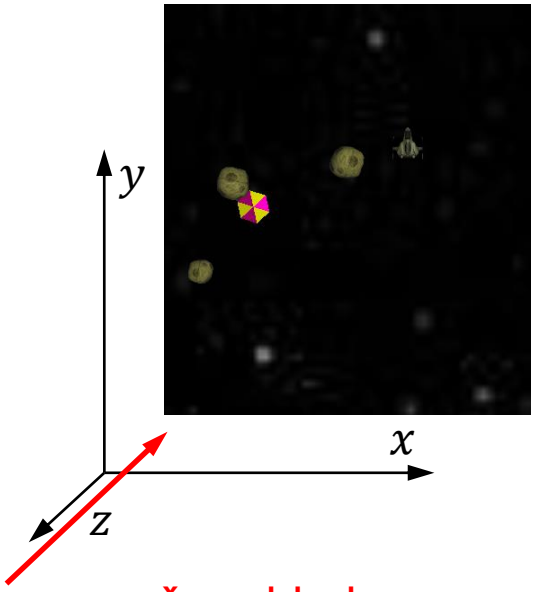
Úloha 2: výpočet osvětlení směrovým světlem

Úloha 3: výpočet osvětlení reflektorem umístěným na raketě

Hra Asteroidy

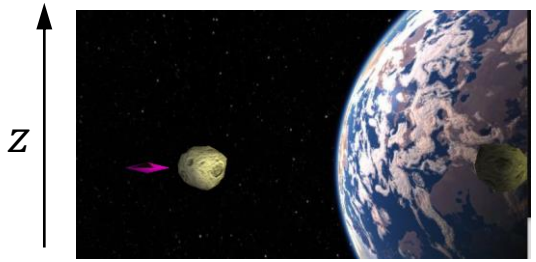
- všechny objekty scény se pohybují v rovině xy světa

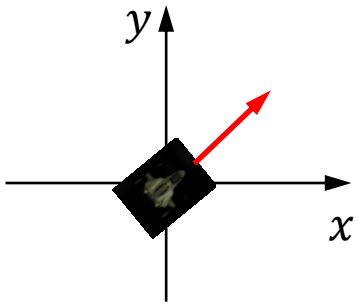
statický pohled na celou scénu



směr pohledu: osa $-z$
up-vektor: osa $+y$

dynamický pohled z rakety (budeme dělat příště)





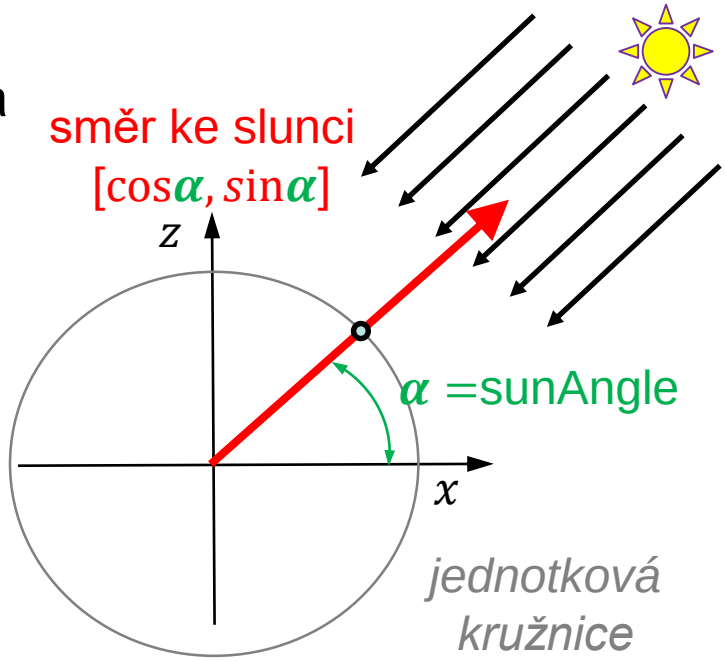
směr pohledu
=
směr natočení rakety v rovině xy $(z = 0)$

Úloha 2 – osvětlení od Slunce

- Slunce – směrové světlo
  - pohyb v rovině xz po kružnici se středem v počátku → úhel natočení sunAngle dán časem a rychlostí pohybu (vstupní uniform time a proměnná sunSpeed)
  - transformace pozice (= směru ke slunci) do prostoru kamery
  - spotlightEffect = 1
- výpočet osvětlení od směrového světla

  → funkce directionalLight()
- inicializace struktury světla pro Slunce

  → funkce setupLights()

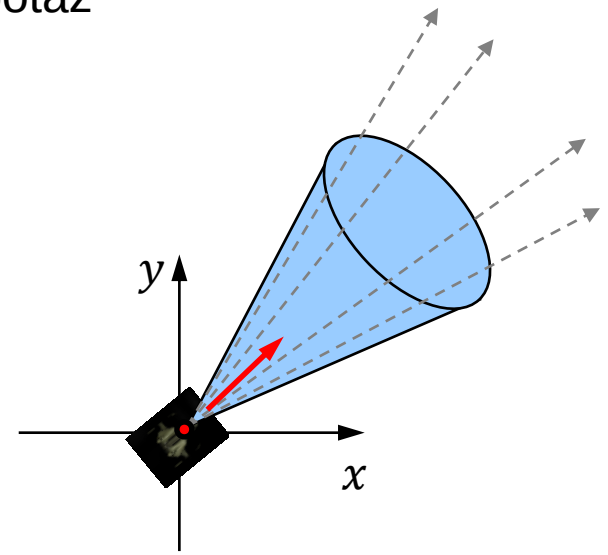


Úloha 3 – reflektor na raketě

- reflektor
  - pozice → poloha rakety → předání přes uniform reflectorPosition
  - směr → směr pohybu rakety → uniform reflectorDirection
  - pozor: oba uniformy je nutné nastavit v C/C++ části aplikace
  - směr a pozici nutno přetransformovat do prostoru kamery
  - pro výpočet spotlightEffect nutno vzít v potaz
    - **SPOT_DIRECTION**
    - **SPOT_CUTOFF**
    - **SPOT_EXPONENT**
- výpočet osvětlení od reflektoru

  → funkce spotLight()
- inicializace struktury světla pro reflektor

  → funkce setupLights()



Úloha 3 – per fragment lighting

- výpočet osvětlení ve fragment shaderu
  - přesun funkcí pro inicializaci světel a výpočet osvětlení z vertex shaderu do fragment shaderu
- interpolace normály
  - transformace normály do prostoru kamery zůstane ve vertex shaderu
  - interpolace může změnit délku normály → znovu normalizovat normálu ve fragment shaderu
- interpolace pozice vrcholu
  - transformace pozice vrcholu do prostoru kamery zůstane ve vertex shaderu

Užitečné rady

- vektory lze celé najednou násobit po složkách pomocí operátoru *
- skalární součin vektorů v GLSL → funkce dot()
- normalizace vektoru v GLSL → funkce normalize()
- pohyb asteroidů lze zastavit zakomentováním řádku ve funkci updateObject() v souboru asteroids.cpp:

```
asteroid->position += timeDelta * asteroid->speed * asteroid->direction;
```