Laboratorul 07

Video Laborator 7: https://youtu.be/y1st9QxXbn8
Autor: Cristian Lambru

Iluminare folosind GLSL

Lumina este un factor foarte important în redarea cât mai realistă a unei scene 3D. Împreună cu proprietățile de material ale unui obiect, lumina determină modalitatea în care obiectul este afișat în scena 3D.

Există mai multe modele empirice pentru calculul reflexiei luminii într-un punct al unei suprafețe: Phong (1975), Blinn (1977), Oren-Nayar (1994), Cook-Torrance (1981), Lambert (1760), etc (la curs veți discuta despre modelul Lambert, Phong și Blinn).

Modelul Phong pentru calculul reflexiei luminii

Intensitatea luminii reflectată într-un punct de pe suprafață către observator este normalizată în intervalul $[0, 1]$, unde 0 reprezintă situația în care lumina care ajunge în acel punct nu este reflectată deloc către observator și 1 este situația în care tot fasciculul de lumină care ajunge in punctul respectiv este reflectat către observator. Pentru a calcula această intensitate în punctul ales vom folosi un model de reflexie care extinde modelul Phong și care conține un total de 4 componente ale intensității luminii pentru a descrie intensitatea finală in punctul de pe suprafață:

- Componenta emisivă
- Componenta ambientală
- Componenta difuză
- Componenta speculară

Contribuția fiecărei componente este calculată ca o combinație dintre proprietățile de material ale obiectului (factorul de strălucire și de difuzie al materialului) și proprietățile sursei de lumină (intensitatea sursei de lumină, poziția sursei de lumină).

Astfel, intensitatea finală a luminii într-un punct aparținând unei suprafețe este:

```
float intensitate = emisiva + ambientala + difuza + speculara; # GLSL
```

În cele ce urmează prezentăm pe scurt ce reprezintă cele 4 componente și cum pot fi calculate.

Componenta emisivă

Aceasta reprezintă lumina emisă de un obiect și nu ține cont de nicio sursă de lumină. O utilizare des întâlnită pentru componenta emisivă este aceea de a simula obiectele care au strălucire proprie (de ex: sursele de lumina precum neonul sau televizorul).

Avem astfel:

```
float emisiva = Ke; # GLSL
```

- Ke – intensitatea emisivă a materialului

Componenta ambientală

Aceasta reprezintă lumina reflectată de către obiectele din scenă de atât de multe ori încât pare să vină de peste tot.

Astfel, lumina ambientală nu vine dintr-o direcție anume, apărând ca și cum ar veni din toate direcțiile. Din această cauză, componenta ambientală este independentă de poziția sursei de lumină.

Componenta ambientală depinde de intensitatea de material ambientală a suprafeței obiectului și de intensitatea luminii.

Similar componentei emisive, componenta ambientală este o constantă (se poate extinde modelul atribuind fiecărei lumini din scenă o intensitate ambientală).

Avem astfel:

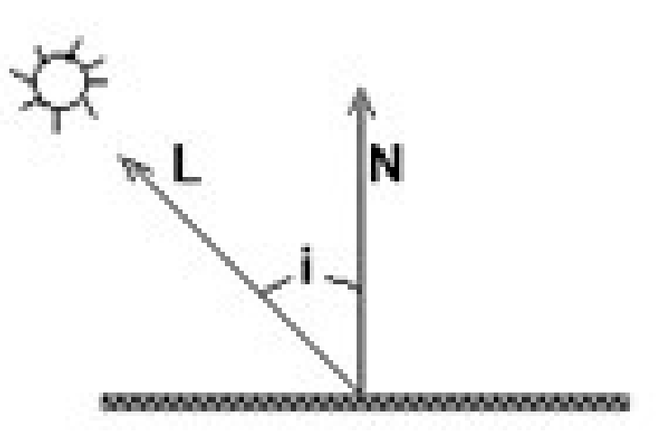
```
float ambientala = Ka * intensitateAmbientalaGlobala; # GLSL
```

- Ka – constanta de reflexie ambientală a materialului
- intensitateAmbientalaGlobala – intensitatea ambientală a luminii

Componenta difuză

Aceasta reprezintă lumina reflectată de suprafața obiectului în mod egal în toate direcțiile.

Cantitatea de lumină reflectată este proporțională cu unghiul de incidență al razei de lumină cu suprafața obiectului.



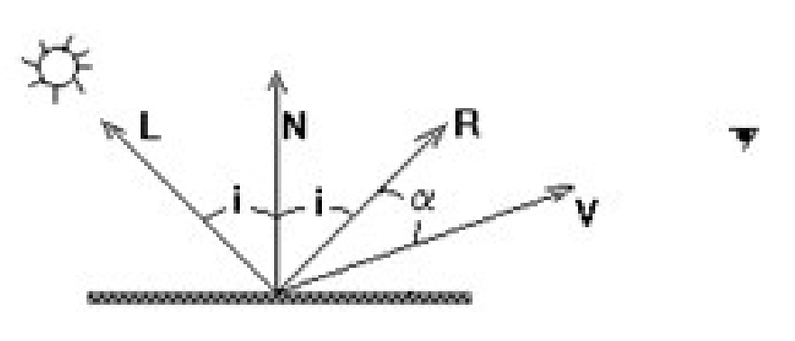
Avem astfel: $difuza = K_d \cdot intensitateLumina \cdot max(\vec{N} \cdot \vec{L}, 0)$

```
float difuza = Kd * intensitateLumina * max (dot(N,L), 0); # GLSL
```

- Kd - constanta de reflexie difuză a materialului
- intensitateLumina – intensitatea luminii
- N – normala la suprafață (normalizată)
- L – vectorul direcției luminii incidente (normalizat)
- $max(\vec{N} \cdot \vec{L}, 0)$ – produsul scalar $\vec{N} \cdot \vec{L}$ reprezintă măsura unghiului dintre acești 2 vectori; astfel, dacă i este mai mare decât $\pi/2$ valoarea produsului scalar va fi mai mică decât 0, acest lucru însemnând că suprafața nu primește lumină (sursa de lumină se află în spatele suprafeței) și de aici și formula care asigură că în acest caz suprafața nu primește lumină difuză

Componenta speculară

Un reflector perfect, de exemplu o oglindă, reflectă lumina numai într-o singură direcție $\vec{R}$, care este simetrică cu $\vec{L}$ față de normala la suprafață. Prin urmare, doar un observator situat exact pe direcția respectivă va percepe raza reflectată.



Componenta speculară reprezintă lumina reflectată de suprafața obiectului numai în jurul acestei direcții, $\vec{R}$. Acest vector se obține prin:

```
vec3 R = reflect (-L, N) # GLSL
```

- Este necesar să se utilizeze -L deoarece `reflect()` are primul parametru vectorul incident care intră în suprafață, nu cel care iese din ea așa cum este reprezentat în figură

În modelul Phong se aproximează scăderea rapidă a intensității luminii reflectate atunci când α crește prin $(cos\alpha)^n$, unde n este exponentul de reflexie speculară al materialului (shininess).

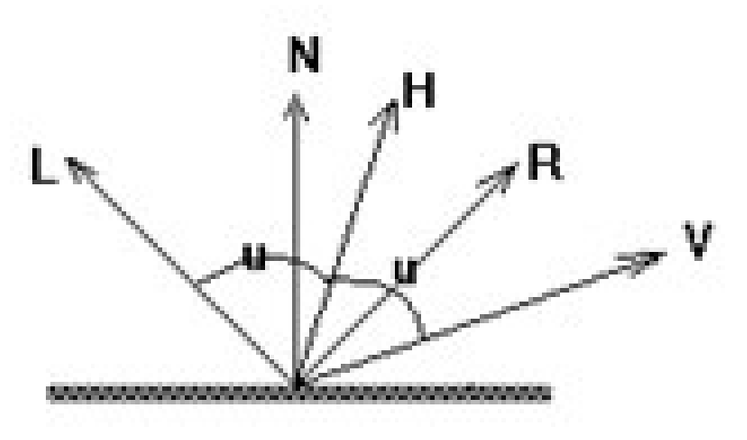
După cum se observă, față de celelalte 3 componente, componenta speculară depinde și de poziția observatorului. Dacă observatorul nu se află într-o poziție unde poate vedea razele reflectate, atunci nu va vedea reflexie speculară pentru zona respectivă. De asemenea, nu va vedea reflexie speculară dacă lumina se află în spatele suprafeței.

Astfel avem: $speculara = K_s \cdot intensitateLumina \cdot primesteLumina \cdot (max(\vec{V} \cdot \vec{R}, 0))^n$

```
float speculara = Ks * intensitateLumina * primesteLumina * pow(max(dot(V, R), 0), n) # GLSL
```

- Ks - constanta speculară de reflexie a materialului
- V – vectorul direcției de vizualizare (normalizat)
- R – vectorul direcției luminii reflectate (normalizat)
- n – coeficientul de strălucire (shininess) al materialului
- primesteLumina – 1 dacă $\vec{N} \cdot \vec{L}$ este mai mare decât 0; sau 0 în caz contrar

Un alt model de iluminare (Blinn (1977)) pentru componenta speculară se bazează pe vectorul median, notat cu $\vec{H}$. El face unghiuri egale cu $\vec{L}$ și cu $\vec{V}$. Dacă suprafața ar fi orientată astfel încât normala sa să aibă direcția lui $\vec{H}$, atunci observatorul ar percepe lumina speculară maximă (deoarece ar fi pe direcția razei reflectate specular).



Termenul care exprimă reflexia speculară este în acest caz: $(\vec{N} \cdot \vec{H})^n$

```
pow(dot(N, H), n) # GLSL
```

- $\vec{H} = (\vec{L} + \vec{V})$ (normalizat)

Atunci când sursa de lumină și observatorul sunt la infinit, utilizarea termenului $\vec{N} \cdot \vec{H}$ este avantajoasă deoarece $\vec{H}$ este constant.

Ținând cont de toate acestea, avem pentru componenta speculară următoarea formulă: $speculara = K_s \cdot intensitateLumina \cdot primesteLumina \cdot (max(\vec{N} \cdot \vec{H}, 0)^n$

```
float speculara = Ks * intensitateLumina * primesteLumina * pow(max(dot(N, H), 0), n) # GLSL
```

Atenuarea intensității luminii

Atunci când sursa de lumină punctiformă este suficient de îndepărtată de obiectele scenei vizualizate, vectorul $\vec{L}$ este același în orice punct. Sursa de lumină este numită în acest caz direcțională. Aplicând modelul pentru vizualizarea a două suprafețe paralele construite din același material, se va obține o aceeași intensitate (unghiul dintre $\vec{L}$ și normală este același pentru cele două suprafețe). Dacă proiecțiile suprafețelor se suprapun în imagine, atunci ele nu se vor distinge. Această situație apare deoarece în model nu se ține cont de faptul că intensitatea luminii descrește proporțional cu inversul pătratului distanței de la sursa de lumină la obiect. Deci, obiectele mai îndepărtate de sursă sunt mai slab luminate. O posibilă corecție a modelului, care poate fi aplicată pentru surse poziționale (la distanță finită de scenă) este:

```
float intensitate = emisiva + ambientala + factorAtenuare * ( difuza + speculara ); # GLSL
```

- factorAtenuare = $1/d^2$ este o funcție de atenuare
- d este distanța de la sursă la punctul de pe suprafață considerat

Corecția de mai sus nu satisface cazurile în care sursa este foarte îndepărtată. De asemenea, dacă sursa este la distanță foarte mică de scenă, intensitățile obținute pentru două suprafețe cu același unghi i, între $\vec{L}$ și $\vec{N}$, vor fi mult diferite.

O aproximare mai bună este următoarea: factorAtenuare = $1/(K_c + K_l \cdot d + K_q \cdot d^2)$

- K_c - factorul de atenuare constant
- K_l - factorul de atenuare liniar
- K_q - factorul de atenuare patratic

Pentru a include în final culoarea de material a obiectului și culoarea luminii (care alternativ pot fi incluse și în formulele de mai sus) se folosește:

```
vec3 culoare = culoareObiect * (emisiva + culoareLumina * (ambientala + factorAtenuare * ( difuza +
```

Modele de shading

De asemenea, există mai multe modele de shading, care specifică metoda de implementare a modelului de calcul al reflexiei luminii. Mai exact, modelul de shading specifică unde se evaluează modelul de reflexie. Dacă vrem să calculăm iluminarea pentru o suprafață poligonală:

- în modelul de shading Lambert, se calculează o singură culoare pentru un poligon al suprafeței
- în modelul de shading Gouraud (1971), se calculează câte o culoare pentru fiecare vârf al unui poligon. Apoi, culorile fragmentelor poligonului se calculează prin interpolare între vârfuri (interpolarea liniară a culorilor vârfurilor, pentru fragmentele de pe laturi și interpolare liniară între culorile capetelor fiecărui segment interior, pentru fragmentele interioare poligonului). Calcularea culorilor vârfurilor se poate efectua în vertex shader.
- în modelul de shading Phong (1975), se calculează câte o normală pentru fiecare vârf al unui poligon. Apoi, pentru fiecare fragment se determină o normală prin interpolare între normalele din vârfuri. Astfel, se calculează o culoare pentru fiecare fragment al unui poligon (în fragment shader)

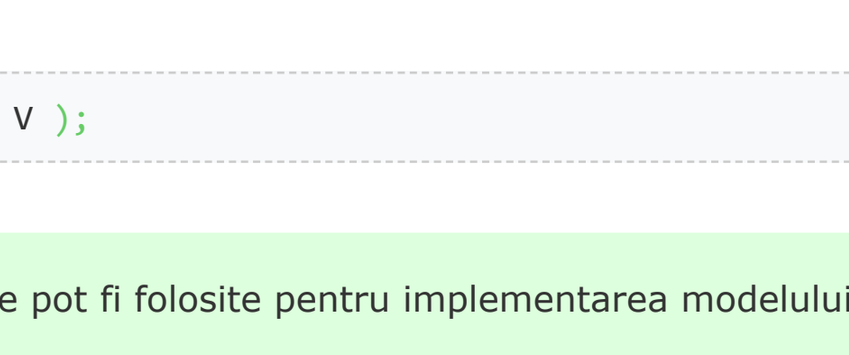


Figura 1. Diferite modele de shading: Lambert (o culoare per primitivă), Gouraud (o culoare per vârf), Phong (o culoare per fragment)

În acest laborator se va discuta modelul de shading Gouraud.

Detalii de implementare

Pentru simplitate, în cadrul laboratorului vom implementa modelul de shading Gouraud (în vertex shader):

- Se vor calcula practic doar componentele difuze și speculare așa cum au fost prezentate anterior; componenta emisivă nu va fi folosită iar calculul componentei ambientale va fi simplificat astfel încât să nu mai trebuiască trimis nimic din program către shader (mai multe detalii la punctul 3).
- Vom folosi ca proprietăți de material pentru obiecte doar intensitatea de material difuză și speculară (transmise din program către shader) : Ks și Kd.
- În shader vom aproxima lumina ambientală cu o `intensitateAmbientalaGlobala` care va fi o constantă în shader, iar în loc de Ka (constanta de material ambientală a obiectului) vom folosi Kd (constanta de material difuză a obiectului).
- Intensitatea luminii va fi 1 și nu va mai fi necesar să fie folosită la înmulțirile din formulele de calcul pentru componentele difuză și speculară.
- Calculele de iluminare se vor face în world space, deci înainte de a fi folosite, poziția și normala vor trebui aduse din object space în world space. Acest lucru se poate face astfel:
  - pentru poziție:
    ```
    vec3 world_pos = (model_matrix * vec4(v_position,1)).xyz;
    ```
  - pentru normală:
    ```
    vec3 world_normal = normalize( mat3(model_matrix) * v_normal );
    ```
- Vectorul direcției luminii L:
  ```
  vec3 L = normalize( light_position - world_pos );
  ```
- Vectorul direcției din care priveste observatorul V:
  ```
  vec3 V = normalize( eye_position - world_pos );
  ```
- Vectorul median H:
  ```
  vec3 H = normalize( L + V );
  ```

Funcții GLSL utile care pot fi folosite pentru implementarea modelului de iluminare

- normalize(V) – normalizează vectorul V
- normalize(V1+V2) – normalizează vectorul obținut prin V1+V2
- normalize(P1-P2) - returnează un vector de direcție normalizat între punctele P1 și P2
- dot(V1,V2) – calculează produsul scalar dintre V1 și V2
- pow(a, shininess) – calculează a la puterea shininess
- max(a,b) – returnează maximul dintre a și b
- distance(P1,P2) – returnează distanța euclidiană dintre punctele P1 și P2
- reflect(V,N) - calculează vectorul de reflexie pornind de la incidenta V și normala N

Cerințe laborator

Tasta **F5** - reîncarcă programele shader în timpul execuției aplicației. Nu este nevoie să opriți aplicația întrucât un program shader este compilat și executat de către procesorul grafic și nu are legătură cu codul sursă C++ propriu-zis.

1. Descărcați framework-ul de laborator
2. Completați funcția `RenderSimpleMesh` astfel încât să trimiteți corect valorile uniforme către Shader:
   - poziția luminii
   - poziția camerei
   - proprietățile de material (Kd, Ks, shininess, culoare obiect)
3. Implementați iluminarea în Vertex Shader
   - Vectorii N, V, L și poziția în spațiul global
   - Componenta ambientală
   - Componenta difuză
   - Componenta speculară (atât în varianta de bază cât și folosind vectorul median)
   - Factor de atenuare
   - Culoarea finală
4. Completați fragment shader-ul astfel încât să aplicați iluminarea calculată în Vertex Shader
5. Colorați sfera și planul din scenă (de ex: sfera - albastru, planul - gri)

Table of Contents

BY-SA CHIMERIC DE CSS DOKUWIKI GET FIREFOX RSS XML FEED XHTML 1.0