Laboratorul 08

Video Laborator 8: https://youtu.be/QuhUGAhrXUQ
Autori: Philip Dumitru, Andrei Lăpușteanu

Modelarea reflexiei luminii

Va reamintim formulele pentru calculul culorii intr-un punct al unei suprafețe:

$culoarea = c_e + c_a + c_d + c_s$

Emisiva: $c_e = K_e$

Ambientala: $c_a = I_a \cdot K_a$

Difuza: $c_d = K_d \cdot I_{sursă} \cdot max(\vec{N} \cdot \vec{L}, 0)$

Speculara: $c_s = K_s \cdot I_{sursă} \cdot lum \cdot (max(\vec{N} \cdot \vec{H}, 0))^n$, unde $lum = (\vec{N} \cdot \vec{L} > 0)?1 : 0$

Dacă introducem mai multe lumini în scenă și ținem cont și de factorul de atenuare, atunci culoarea intr-un punct al unei suprafețe este:

$culoarea = K_e + I_a \cdot K_a + \sum f_{at_i} \cdot I_{sursă_i}(K_d \cdot max(\vec{N} \cdot \vec{L_i}, 0) + K_s \cdot lum_i \cdot (max(\vec{N} \cdot \vec{H_i}, 0))^n)$

La laboratorul de saptamana trecuta, pentru usurința implementării, am considerat mai multe simplificări:

- am considerat că toate constantele de material K_e, K_a, K_d, K_s, sunt variabile de tip float (un singur canal)
- deoarece constantele de material au fost considerate pe un singur canal, s-a introdus variabila uniformă object_color, o variabilă de tip vec3 care a modelat culoarea obiectului
- am considerat că intensitatea sursei de lumină este o constantă float (cu valoarea 1)
- am ignorat culoarea emisă
- am înlocuit constanta de material K_a cu constanta K_d (pentru a trimite mai putine uniforme)
- am considerat intensitatea luminii ambientale o constantă float (cu valoarea 0.25)

Totuși, trebuie să menționăm că modelul complet urmărește formula de mai sus, unde constantele de material K_e, K_a, K_d, K_s sunt diferite și au 3 canale (R, G, B), iar intensitatea luminii ambientale și intensitatea sursei de lumină au de asemenea 3 canale. Expresia luminii se evaluează separat pentru cele trei canale.

Iluminare Phong in Fragment Shader

Modelul de iluminare aplicat in cazul implementarii in fragment shader este acelasi cu cel studiat in Laboratorul 07, din punct de vedere matematic. Totusi, exista o diferenta majora intre cele doua implementari prin faptul ca iluminarea nu se mai aplica la nivelul fiecarui vertex ci la nivel de fragment. Rezultatul final este superior calitativ intrucat iluminarea fiecarui fragment nu se va mai calcula pe baza interpolarii luminii calculate la nivel de vertex ci pe baza normalei si pozitiei in spatiu a fiecarui fragment.

Valorile de intrare primite de fragment shader sunt interpolate linar intre valorile vertexilor ce compun primitiva utilizata la desenare.

Imaginea de mai sus este obtinuta prin desenarea unui triunghi avand cele 3 varfuri de culori diferite: **rosu**, **verde**, **albastru**

Prin **transmiterea culorii de la Vertex Shader la Fragment Shader** culoarea fiecarui fragment de pe suprafata triunghiului este calculata ca o **interpolare liniara intre culorile** vertexilor ce compun primitiva specificata (in acest caz, un triunghi).

Acelasi procedeu se aplica pentru orice alta proprietate, cum ar fi:

- pozitia in spatiul lume a unui fragment (daca trimitem pozitiile vertexilor)
- normala in spatiul lume a unui fragment (daca trimitem normalele vertexilor)
- orice alta valoarea transmisa de la vertex shader la fragment shader
- etc

Modelul de interpolarea implicit utilizat (**smooth**) calculeaza interpolarea tinand cont si de perspectiva (se face o interpolare perspectiva).
API-ul OpenGL permite specificarea modelului de interpolare prin utilizarea unor termeni specifici in cadrul Fragment Shaderului:

- **flat** - valoarea nu va fi interpolata
- **smooth** - interpolare perspectiva (implicita)
- **noperspective** - interpolare liniara in spatiu fereastra

Mai multe detalii despre modelele de interpolare se pot gasi accesind urmatoarele resurse:

- OpenGL Type Qualifier
- OpenGL Interpolation Qualifiers Tutorial

Astfel, utilizand valorile interpolate de pozitie si normala (in spatiul lume) putem sa calculam modeul de iluminare Phong pentru fiecare fragment al unei primitive rasterizate, rezultatul final fiind mult superior intrucat prin **interpolarea normalelor** se obtine o trecere lina intre suprafete adiacente (sunt interpolate normalele de pe muchii), deci si iluminarea finala va oferi impresia unei suprafete netede. Astfel poligoanele componente ale obiectelor nu vor mai aparea vizibil in imagine.

Detalii de implementare

1. Se calculeaza **world_position** si **world_normal** in Vertex Shader ca in Laboratorul 07
2. Se transmit cele 2 valori catre Fragment Shader
3. Se aplica calculul luminii (componenta ambientala, difuza, speculara) in Fragment Shader

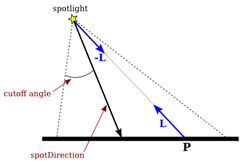
Pentru a primi valoarea unei variabile de tip `uniform` este suficient sa declarati respectiva variabila in shaderul in care este necesara. **Deci, NU trimiteti valoarea unei variabile de la Vertex Shader la Fragment Shader**

```
// Vertex Shader
uniform vec3 light_position;
```

```
// Fragment Shader
uniform vec3 light_position;
```

Iluminare Spot-light

Nu toate surselе de lumina sunt punctiforme. Daca dorim sa implementam iluminarea folosind o sursa de lumina de tip spot trebuie sa tinem cont de o serie de constrangeri



Asa cum se poate vedea si in poza pentru a implementa o sursa de lumina de tip spot avem nevoie de urmatorii parametri aditionali:

- orientarea spotului (directia luminii)
- unghiul de cut-off al spotului ce controleaza deschiderea conului de lumina
- un model de atenuare unghiular al luminii ce tine cont valoarea de cut-off a spot-ului

Astfel, punctul **P** se afla in conul de lumina (primeste lumina) daca conditia urmatoare este indeplilita:

```
float cut_off = radians(30);
float spot_light = dot(-L, light_direction);
if (spot_light > cos(cut_off))
{
        // fragmentul este iluminat de spot, deci se calculeaza valoarea luminii conform  modelului Phon
        // se calculeaza atenuarea luminii
}
```

Pentru a simula corect iluminarea de tip spot este nevoie sa tratam si atenuarea luminii corespunzatoare apropierii unghiului de cut-off. Putem astfel sa utilizam un model de atenuare patratica ce ofera un rezultat convingator.

```
float cut_off = radians(30);
float spot_light = dot(-L, light_direction);
float spot_light_limit = cos(cut_off);

// Quadratic attenuation
float linear_att = (spot_light - spot_light_limit) / (1.0f - spot_light_limit);
float light_att_factor = pow(linear_att, 2);
```

Cerinte laborator

tasta **F5** - reincarca shaderele in timpul rularii aplicatiei.

In cazul in care ati modificat doar sursele shader nu este nevoie sa opriti aplicatia intrucat shaderele sunt compilate si rulate de catre placa video si nu au legatura cu codul sursa C++ propriu zis, iar framework-ul ofera suport pentru reincarcarea acestora la runtime.

1. Descarcati framework-ul de laborator
2. Sa se implementeze iluminarea de tip Phong in Fragment Shader
3. Atunci cand se apasa tasta F sa se treaca in modul de iluminare Spot-light
   - Directia de iluminare este transmisa ca `uniform vec3 light_direction`
   - Nu uitati sa aplicati un model de atenuare al luminii in functie de apropierea fragmentelor de unghiul de cut-off

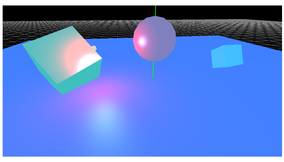
[Bonus]

1. Sa se modifice directia si unghiul de cut-off al luminii spotlight de la tastatura
   - logica in `OnInputUpdate`
   - rotirea spotului: **sus, jos, stanga, dreapta**
   - 2 taste pentru a creste/micsora unghiul de iluminare al spot-ului
2. Modelul de iluminare Phong, folosind 3 canale de culoare:

Atât în implementearea laboratorului de săptămâna trecută, cât și în implementarea laboratorului din această săptămână, am folosit un model simplificat. Din acest motiv, propunem ca bonus să modificați fragment shader și vertex shader astfel încât să vedeți cum se face de fapt implementarea iluminării. Astfel, va trebui sa implementați următoarele:

- Constantele de material Ke, Ka, Kd, Ks vor fi variabile de tip vec3, cu valori diferite pe toate cele cele 3 canale de culoare (de data aceasta va fi nevoie să trimiteți toate aceste constante către shader).
- Veți renunța la variabila object_color prin care ați modelat culoarea obiectului. În această implementare, culoarea obiectului va fi calculată folosind cele 4 constante de material.
- Intensitatea sursei de lumină și intensitatea luminii ambientale vor fi tot variabile de tip vec3, care pot avea și alte valori, nu numai glm::vec3(1, 1, 1). Ce zici vectori pot fi diferiți. 😃

Exemplu de rezultat (a se observa cum componenta difuză a luminii este mult mai roșiatică față de componenta speculară):



Info curs
- Elemente de Grafică pe Calculator
- Infographie

Cataloage EGC
- TBA

Laboratoare
- Laboratorul 01
- Laboratorul 02
- Laboratorul 03
- Laboratorul 04
- Laboratorul 05
- Laboratorul 06
- Laboratorul 07
- Prezentare Tema 1
- Laboratorul 08
- Laboratorul 09
- Vacanță
- Prezentare Tema 2
- Recuperări laborator
- Prezentare Tema 3
- Resurse: Redare text

Teme
- Regulament General
- Tema 1 - Bow and Arrow
- Tema 2 - Skyroads
- Tema 3 - Stylised Runner

Resurse
- Resurse Utile
- Notare

Table of Contents