8. Gyakorlat - 01_Lights

Forrás: http://cg.elte.hu/~bsc_cg/gyak/08/01_Normals.zip

Bővebb projekt mesh-sel, textúrával:http://cg.elte.hu/~bsc_cg/gyak/07/03_LightsMesh.zip

Megjegyzés: A dokumentáció a tananyag egy régebbi változatához készült, ekkor még a mesh-es, textúrás projekten voltak bemutatva a fények.

Kapcsolódó előadás-anyag: http://cg.elte.hu/~hajder/2018osz/BSc_EA_06.pdf
http://cg.elte.hu/~hajder/2018osz/BSc_EA_08.pdf (3. rész)

Elmélet

Azért, hogy a színterünk realisztikus hatást keltsen, szeretnénk fényhatásokat megjeleníteni. A fények kezelése kritikus kérdés a grafikában, mivel az általános és teljes megoldása normális idő alatt nem kiszámítható. Számos egyszerűsítési megközelítés létezik azonban, hiszen a valós idejű alkalmazások esetén a képeket a másodperc törtrésze alatt kell előállítani.

Megjegyzés: Fényhatások nélkül az elkészült képek minősége gyakran nem elfogadható, például egy homogén egyszínű geometria valahogy így néz ki:

Szükségünk van tehát fényforrás- és anyagmodellekre, melyek közül most néhánnyal fogunk megismerkedni:

- **Ambiens fénymodell**
  Nincsen pozícióval rendelkező fényforrás, a felületre egy konstans fénymennyiség vetül.
  Például ködös időjárás szimulálására alkalmas.
- **Diffúz fénymodell**
  A fényenergia nagysága függ a fényforrás helyétől, de a kamera pozíciótól nem. (Minden irányba azonos intenzitással veri vissza a beeső fényt) Például meszelt fal.
- **Spekuláris fénymodell**
  A fényenergia a fényforrás és a kamera pozíciójától is függ, ugyanis a fény legnagyobb részét az ideális visszaverődés irányába veri vissza.
  Például fényben megcsillanó pénzérme.
- **Átlátszó, áttetsző felületek**
  Nem tananyag, érdeklődőknek: (Ctrl-F: fénykard)

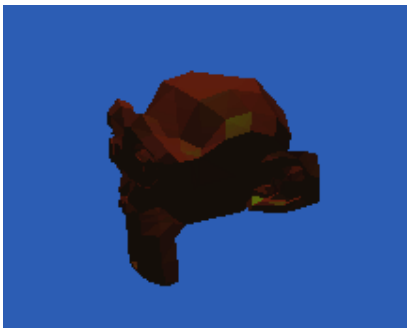
Az első három modell megfelelő paraméterekkel történő kombinációival elég jó közelítéseket kaphatunk számos fény-anyag kölcsönhatásra.

A modellben szereplő fényforrások lehetnek pont fényforrások (ekkor egy adott pozíciójuk van), illetve irány fényforrások, amikor az összes onnan érkező fénysugár egy irányból, párhuzamosan érkezik be. (Irány fényforrással szokás modellezni olyan fényforrásokat is, ami “nagyon” messze van. Így például a Napból származó fénysugarakat egy jelenetben tekinthetjük úgy, mintha mindenhol ugyanabból az irányból jönne.)
Most pedig vizsgáljuk meg, hogyan tudjuk a háromszög modelleinken meghatározni a fényintenzitást.

Korábbi gyakorlatok projektjei és a most megismert modellek alapján már ismerjük:

- A háromszögek csúcsainak pozícióját (világ koordináta-rendszerben is)
- A háromszögek mindhárom csúcsában a felületi normálvektort
- A háromszöglap normálvektorát (ez bármely két él vektoriális szorzata – több háromszög esetén érdemes figyelni arra, hogy a különböző lapokra kapott normálisok orientációja legyen konzisztens, azaz ne váltakozzanak fel és le)
- A felület anyagára jellemző paramétereket (ambiens/diffúz/spekuláris modell szerint)

Első megközelítés: ***Flat shading.*** Számítsuk ki a háromszögre, mint síklapra vonatkozó fényintenzitást, majd használjuk ezt az egész háromszögön homogén módon. Az eredmény egy, a lapokat élesen elhatároló megvilágítás. A módszer hátránya, hogy ívelt felületeket töredezetté tesz.



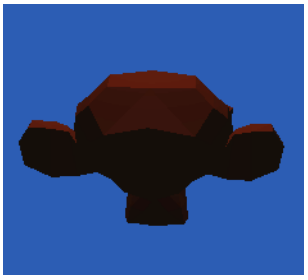
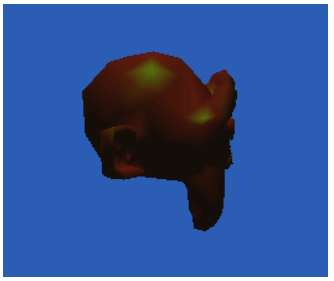
Ha ehelyett a felületi normálvektorokat csúcsonként tekintjük (mert például egy parametrikus felületből tudjuk számolni őket), akkor tehetjük az alábbiakat:

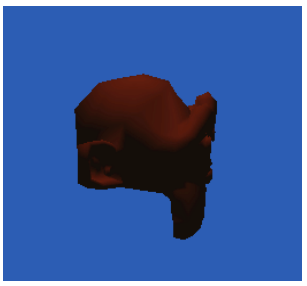
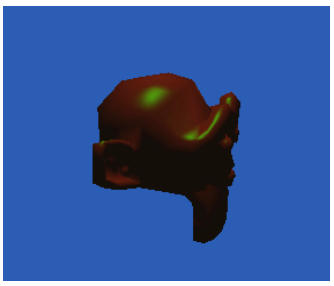
- Kiszámítjuk a fényintenzitást a háromszögek csúcsaiban.
- A háromszögön belül a színeket az eddigiek alapján interpoláljuk.

Ez a módszer a ***Gouraud-árnyalás***. Realisztikusabb képet ad, mint a flat shading, de itt is elveszíthetjük a spekuláris “csillanásokat”, ha azok éppen háromszöglapok közepén vannak, és a csúcspontokban az intenzitás épp gyengébb.

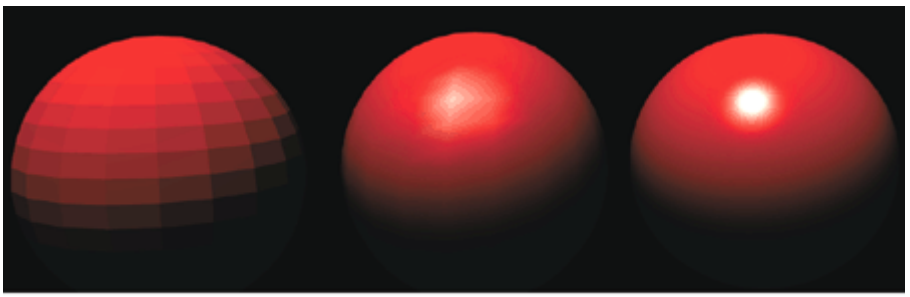
Gondoljuk eggyel tovább a Gouraud módszert, és a kiszámolt színek helyett magukat a normálvektorokat interpoláljuk még a színek kiszámítása előtt. Ezzel egy új módszert, a ***Phong-árnyalás*** módszerét kapjuk. Ezzel a módszerrel már a csillanások is bárhol meg tudnak jelenni.

Az alábbi táblázatban egy összehasonlítás látható, ahol ugyanarra az objektumra alkalmazunk különböző fénymodelleker, illetve fénytípusokat. A szimulációban itt most a diffúz fény színe barnás, míg a spekuláris fény zöldes színnel jelenik meg.

	Ambiens	Ambiens + diffúz	Ambiens + diffúz + spekuláris
Flat			
Gouraud			

Phong			

Az alábbi képen egy másfajta objektumon (gömbön) látható a modellek összevetése:



Természetesen a modellek számításigénye és képminősége is a táblázatban fentről lefelé nő. Mi a gyakorlaton a Phong-modellt fogjuk használni, ugyanis a fragment shaderek már eleve támogatják az ehhez szükséges számításokat (konkrétan a minden pixelre párhuzamos számításokat.) Apró történelmi érdekesség, hogy a fragment shaderek bevezetése előtt egyébként sokáig eleve egyedül csak a Gouraud-árnyalás volt hatékonyan megvalósítható (illetve a grafikus API-k úgynevezett fix-funkciós részei (FFP) ezt valósították csak meg).

Példaprogram

A példaprogramban még nincsen megvalósítva fénymodell, csak az ehhez szükséges paraméterek (pipeline csatornák, uniform változók) beállítása történik meg.
Egy feltextúrázott síkot látunk, ezen tudjuk majd a modelleket tesztelni.

C++ program

A headerben jó néhány, a fentebb említett paraméterekhez kapcsolódó változót, illetve uniform változó-azonosítót veszünk fel.
Át kell adni ugyanis a shadernek – a transzformációk szorzatán kívül – a világ transzformációt, illetve annak inverz transzponáltját is (hiszen a normálvektorokat ezzel kell a vertex shaderben transzformálni). Így elő tudjuk majd állítani a világ koordináta-rendszerbeli adatokat.
Továbbá mostantól átadjuk a shader-nek a kamerapozíciót is - ez majd a spekuláris modellhez fog kelleni.

Az Update függvényben előállítjuk a kamera pozíciót, ezt használjuk fel a nézeti transzformáció megállapításához.

A Render függvényben kiszámítjuk az inverz transzponáltat (a megfelelő glm metódusok segítségével), majd az összes szükséges mátrixot átadjuk a vertex shadernek.

Vertex shader

A vertex shaderrel 4 dolgot kell előállítanunk:

- A gl_Position-t továbbra is a három transzformációs mátrix szorzatával szorozzuk
- A textúra koordinátákat nem kell transzformálni
- A világkoordinátákat a world mátrixszal szorzott pozícióból kapjuk
- A normálvektorokat a world inverz transzponáltjával kell szorozni

Figyelem: itt a homogén koordinátát 0-ra kell állítani!

Az utóbbi három adatot átadjuk a fragment shadernek.

Fragment shader

Phong modellt fogunk megvalósítani, azaz minden fragment-ben már rendelkezésünkre áll az interpolált normálvektor. A fényintenzitásokat tehát a fragment shaderben számítjuk. (Gouraud-modell esetén lehetne a vertex shaderben is.)

A stratégiánk az lesz, hogy egymás után előállítjuk az ambiens, diffúz és spekuláris modellből származó fényintenzitást, az RGB színmodell mindhárom komponensében, majd ezeket összegezzük.

A képletekben mindenhol $L \in \mathbb{R}^3$ jelenti a visszavert fény intenzitását (RGB kódolás szerint), $l \in \mathbb{R}^3$ a fény erősségét (fényforrásra jellemző), $k \in \mathbb{R}^3$ az anyag fényre való reagálási tulajdonságát (felületre jellemző), $d \in \mathbb{R}^3$ a visszavert fény irányát (egység hosszú vektor) és $n \in \mathbb{R}^3$ a felületi normálvektort (szintén egységvektor).

Ambiens fény számítása

Mivel az ambiens fény független a pozícióktól és a normálisoktól, ezért csak összeszorozzuk az anyagra jellemző értéket a színtérben jelen levő ambiens fényre jellemző értékkel. Ezeket vec3 változókban tudjuk tárolni (RGB), és komponensenként kell őket összeszorozni. (a * operátor ezt is teszi)

$$L_{ambient} = l_{ambient} * k_{ambient}$$

Kódban:

```
vec4 ambient = La * Ka;
```

Diffúz fény számítása

A diffúz fény intenzitása (a fényforrás erősségén, és az anyagtulajdonságokon kívül) attól függ, hogy a fénysugarak beesési iránya mennyire esik egybe a felületi normálissal. Ezt az

“egybeesést” legjobban a két irányvektor skaláris szorzatával tudjuk mérni. A beesési irány vektorát megfordítjuk, így a skaláris szorzat akkor lesz 1, ha a két irány egybeesik, és akkor 0, ha merőlegesek egymásra. A skaláris szorzat lehet negatív is, de negatív fényerőnek nincs értelme, ezért a negatív értékeket 0-nak vesszük (erre alkalmas a *clamp* függvény).

Az így kapott értéket (valós szám) hozzászorozzuk a már előbbiekben látott módon összeszorzott anyag- és fénytulajdonság-vektorral. Természetesen ezek a vektorok az előbbiektől jellemzően különbözni fognak.

$$L_{diffuse} = \langle d,\ n \rangle\ *\ l_{diffuse}\ *\ k_{diffuse}$$

Kódban:

```
float di = clamp( dot( toLight, normal), 0.0f, 1.0f );
vec4 diffuse = Ld*Kd*di;
```

Spekuláris fény számítása

A spekuláris fénynek az a jellemzője, hogy az ideális visszaverődéshez közel nagyon intenzív, de ettől távolodva gyorsan elhal. Egy olyan függvényre van tehát a modellezéséhez szükségünk, amely az ideális visszaverődési irány és a fragment pozícióból a kamera pozícióba mutató vektor által bezárt szögtől függ, a 0-ban 1, a $\pi/2$-ben 0, és a 0-tól távolodva gyorsan lecsökken az értéke. Egy alkalmas ilyen függvény lesz a $\cos^k(\varphi)$, ahol *k* egy tetszőlegesen megválasztott pozitív szám (mondjuk 10, de lehet vele kísérletezni). A $\cos(\varphi)$ az előző részben már látott skaláris szorzattal számítható.

Tehát képeznünk kell az ideális visszaverődési irány és a fragmentpozícióból a kamerapozícióba mutató vektor skaláris szorzatát, majd ezt felemelni az n-edik hatványra.

$$L_{specular} = \langle e,\ r \rangle^{n}\ *\ l_{specular}\ *\ k_{specular}$$

Kódban:

```
float si = pow( clamp( dot(e, r), 0.0f, 1.0f ), specular_power );
vec4 specular = Ls*Ks*si;
```

Vigyázat: Ha n páros, akkor az n-edik hatványra egy negatív skaláris szorzatból pozitív értéket csinál. Ez azt jelenti, hogy a fényforrás a felület *mögött* van. A legkönnyebben ez úgy kiszűrhető, hogy ha a diffúz fényintenzitás negatív, akkor 0-nak vesszük és nem számolunk spekuláris modellt sem.

Végső intenzitás számítása

A három modellből kapott értékeket összegezzük, és az így kapott RGBA értéket összeszorozzuk a textúra mintavételezéséből származó színértékkel (elemenként), ezzel elérve a fényhatás “keveredését” a textúrából nyert színnel.

```
fs_out_col = (ambient + diffuse + specular ) * texture(texImage, vs_out_tex0.st);
```