GiVD - curs 2014-2015 – Pràctica 3

Pràctica número 3

Tema: Il·luminació de l'escena: Implementació de la il·luminació d'una escena amb diferents models locals de *shading*: flat, *Gouraud*, *Phong* i *toon shading*, tenint en compte les textures.

Objectiu: Visualització de la taula de billar amb diferents *shaders* que permetin els diferents tipus d'il·luminació. Aprendre a utilitzar els shaders per a programar els models d'il·luminació.

La pràctica té com objectiu aprendre a il·luminar l'escena amb diferents models de *shading*. Per això s'aprendrà a passar diferents valors al *vertex shade*r i al *fragment shader.* S'inclouran dues noves classes: la **classe llum**, que representa tots els atributs d'una llum i la **classe material**, que codificarà els atributs bàsics que codifiquen un material (component difusa, component especular i component ambient).

Fins a la pràctica 2, heu inclòs nous objectes en una escena i heu codificat la càmera per a poder visualitzar la taula de billar des de qualsevol posició i amb diferents opcions de zoom. En aquesta tercera pràctica s'inclou en l'escena una llum i a cada bola de billar i al pla base, el seu material associat. Per a començar a desenvolupar el codi d'aquesta tercera pràctica, es pot utilitzar el codi realitzat a la pràctica 2 fins a la primera visualització usant la càmera general.

La pràctica 3 es composa dels següents exercicis:

- Implementació d'una nova classe anomenada **llum**, que permeti emmagatzemar tots els atributs necessaris per a codificar una llum puntual, una llum direccional, una llum de tipus spot-light i amb possible atenuació en profunditat.
- Implementació d'una nova classe **material** que permeti representar les característiques del material d'un objecte.
- Implementació dels càlculs d'il·luminació segons les següents tècniques: *flat, gouraud, phong-shading* i *toon-shading* tenint en compte les textures.

1. Etapes de desenvolupament de la pràctica:

PAS 1. Creació d'una nova classe llum

Creació d'una nova classe Llum

Descripció:

Aquesta classe haurà de permetre codificar una llum i totes les seves propietats, segons el tipus de llum que codifiqui. En principi, en aquesta classe heu de definir els atributs que permetin representar:

- llums puntuals (posició)
- llums direccionals (direcció)
- llums spot-light (direcció, angle d'obertura)

Addicionalment, cal que cada llum codifiqui la intensitat en la què emet (ambient, difusa i especular) i els coeficients d'atenuació en profunditat (constant, lineal i quadràtica).

En la classe llum cal afegir un mètode anomenat **toGPU()** que permeti passar la informació d'una llum a la GPU, segons la següent capçalera:

```
void Llum::toGPU(QGLShaderProgram *program)
```

Com fer-ho?

Per estructurar la informació en els *shaders*, s'utilitzarà un "*struct*" per una llum. Recordeu que per a comunicar la CPU amb la GPU s'han de fer diferents passos. A continuació es detallen els passos concrets per a passar una llum a la GPU, si s'utilitza un "*struct*" en els *shaders*.

a. Com es defineix i s'utilitza un struct en els shaders?

```
struct tipusLlum
{
    vec4 LightPosition;
    vec3 Ld;
    float coef_a;     // Aqui s'han de definir tots els atributs d'una llum
    ...
};

uniform tipusLlum light;
```

Fixeu-vos que es la variable **light** és de tipus "uniform" ja que serà constant a tots els vèrtexs de l'objecte.

b. Com es fa el lligam entre les variables de la CPU i la GPU?

En la CPU, farem el lligam de les variables de la GPU de la següent forma:

```
  // 1. Per a passar els diferents atributs del shader declarem una estructura amb els identificadors de la
GPU
    struct {
        GLuint posicio;
        GLuint ld;
        GLuint a;
        ....
    } gl_ldLlum;

   // 2. obtencio dels identificadors de la GPU
    gl_ldLlum.posicio = program->uniformLocation("light.LightPosition");
    gl_ldLlum.ld = program->uniformLocation("light.Ld");
    gl_ldLlum.a = program->uniformLocation("light.a");
    ...

// 3. Bind de les zones de memoria que corresponen
   glUniform4fv(gl_ldLlum.posicio, 1, posicioLlum);  //posicioLlum és un vec4
   glUniform3fv(gl_ldLlum.ld, 1, difusa );                 // difusa és un vec3
   glUniform1f(gl_ldLlum.a, coef_a);                       // coef_a és un Glfloat
            ...
```

c. On es declara la llum?

Finalment, per a que es pugui utilitzar una llum en l'escena, cal que afegiu un atribut de llum a la classe escena, juntament amb la intensitat d'ambient global a l'escena. Aquesta intensitat global, caldrà passar-la també a la GPU per a ser utilitzada pels *shaders*. Per això cal que implementeu un mètode a la classe escena anomenat setAmbientToGPU, amb la següent capçalera:

```
void escena::setAmbientToGPU(QGLShaderProgram *program)
```

Com comprovo que es passen bé els valors?

Per a validar que passeu bé les dades a la GPU, com no es pot imprimir per pantalla des del *shader*, utilitzeu la variable color del *vertex shader*. Si per exemple, li assigneu la intensitat difusa de la llum, hauríeu de visualitzar els objectes del color que heu posat a la intensitat difusa des de la CPU.

Comenceu definint només una llum puntual i comproveu que tots els seus atributs es passen correctament a la GPU.

Possible ampliació: Afegir diferents llums enlloc de només una

En l'escena es poden tenir diferents llums. Per això cal implementar una classe que contingui el conjunt de llums de l'escena. Per a passar-les al vèrtex shader, no es poden passar directament com una taula de structs, sinó que cal passar-les d'una a una. Fes un màxim de tres llums.

Finalment, per a que es pugui utilitzar una o varies llums en l'escena, cal que afegiu un atribut conjunt de llums a la classe escena,

PAS 2. Creació d'una nova classe material

Creació d'una nova classe Material

Descripció:

Implementeu una nova classe per a representar el **material** d'un objecte. Aquesta classe definirà els atributs de les propietats òptiques d'un objecte (component ambient, component difús, component especular i coeficient de reflexió especular). Cada objecte tindrà associat un material.

Implementeu també el mètode per a passar els seus valors a la GPU:

```
void Material::toGPU(QGLShaderProgram *program)
```

Utilitzeu també "*structs*" per a estructurar la informació tant a la CPU com a la GPU. Aquest mètode es cridarà des de cadascun dels objectes, dins dels mètodes **toGPU** corresponents.

On es declaren els materials?

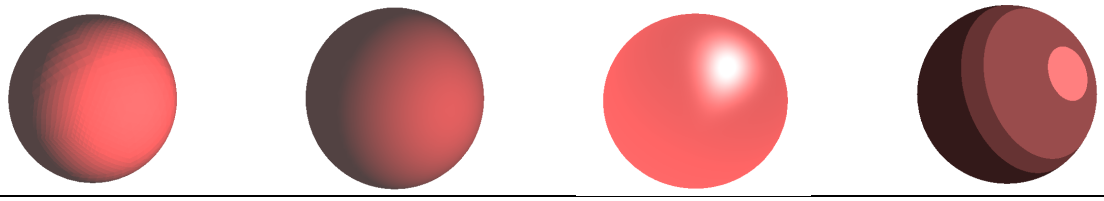
Cada objecte té associat el seu material. Quan afegiu el material a l'objecte, observeu que el vector de colors associats a cada vèrtexs ja no té sentit i per tant ja no és necessari passar-lo a la GPU. En canvi, pel càlcul de la il·luminació és necessari passar les normals associades a cada vèrtex de l'objecte.

Modifiqueu la classe objecte per a tenir en compte aquests canvis en el pas de dades a la GPU.

Codifica el *vèrtex shader* per a que el pugui rebre convenientment i comprova que les dades estiguin ben passades. Posa un material gris al pla base per a provar-ho.

PAS 3: Implementació dels diferents tipus de *shading*

Creació de diferents tipus de shading



Descripció:

En aquest apartat es tracta d'implementar els següents diferents tipus de *shading*:

- *Flat shading* (Color constant a cada triangle)
- *Gouraud* (suavització del color)
- *Phong shading* (suavització de les normals)
- *Toon shading* (shading discret segons el producte del vector de la llum direccional amb la normal al punt)

Aquests tipus d'il·luminació estan definits conceptualment a les transparències de teoria. Suposarem que per aquests *shadings* poligonals usarem el model de **Blinn-Phong.**

Teniu en compte que les variables de tipus **out** del *vertex shader* són les que es rebran com a **in** en el fragment shader, ja interpolades en el píxel corresponent.

Es vol activar cada shader amb una tecla numèrica: la tecla 1 activarà el *Flat shading* a tots els objectes de l'escena, la 2 el *Gouraud*, la 3 el *Phong shading* i la 4 el *Toon-shading*. Si no es té activada la tecla Majúscula, es visualitzarà el color del Material de l'objecte. Si l'usuari té premuda la tecla Majúscula alhora que la tecla 1, es visualitzarà la textura associada a l'objecte combinada amb el material base de l'objecte. *Per exemple, un 75% del color final serà de la textura i un 25% el color del material base.*

Breu explicació de cada shader:

Flat shading (Color constant a cada triangle): En el *Flat shading* es calculen les normals a cada vèrtex de cada objecte, tenint en compte que tots els vèrtexs d'una mateixa cara tenen la mateixa normal.

D'aquesta forma cada vèrtex d'un triangle tindrà el mateix color, ja que té la mateixa normal i només en el *fragment shader* només es farà la interpolació del color.

Gouraud (suavització del color a partir de les normals calculades a cada vèrtex): En el *Gouraud shading* es calculen les normals a cada vèrtex i s'interpola el color a nivell de píxels. En aquest tipus de *shading* heu de calcular les normals "reals" a cada vèrtex dels objectes.

Phong shading (suavització de les normals a nivell de píxels): En el *Phong shading* les normals s'interpolen a nivell de píxels. Raoneu on és calcula la il·luminació i modifiqueu convenient els fitxers de la pràctica.

Toon shading (shading discret segons el producte del vector de la llum direccional amb la normal al punt): Calculeu a nivell de píxel quatre intervals de valors on donareu diferents tonalitats d'un mateix color a l'esfera.

Si vols utilitzar diferents *shaders* en temps d'execució raona on s'inicialitzaran els *shaders* i com controlar quin *shader* s'usa? Cal tornar a passar l'escena a la GPU quan es canvia de shader?

Com fer-ho? Passos a seguir:

1. Implementa inicialment el *flat shading* amb una llum puntual per aplicar-lo a tots els objectes de l'escena. Per implementar el *flat shading* de cada objecte cal tenir definides a cada vèrtex de cada triangle la mateixa normal. Per això, a cada vèrtex cal calcular la seva normal, segons la cara que estigui representant. Cal passar aquestes normals a la GPU, modificant els mètodes **toGPU** i **draw** dels objectes i el *vertex shader*. Cal implementar el model de *Phong-Blinn* en el *vertex shader*.
2. Prova amb diferents materials canviant les diferents components de les propietats òptiques.
3. Implementa la possibilitat de fer *flat shading* amb i sense textura segons es premi la tecla 1 o Majúscules+Tecla 1.
4. Implementa *Gouraud-shading* quan es premi la tecla 2, calculant les normals reals a cada vèrtex. Com canviaràs el càlcul de les normals de cada vèrtex? Prova la diferència de visualització amb el *flat-shading*. Necessites uns nous *vertex-shader* i *fragment-shader*? Raona on és calcula la il·luminació i modifica convenientment els fitxers de la pràctica. Implementa la possibilitat de fer *flat shading* amb i sense textura segons es premi la tecla 2 o Majúscules+Tecla 2.
5. Implementa el *Phong-shading* quan es premi la tecla 3. Quina diferència hi ha amb el *Gouraud-shading*? On l'has de codificar? Necessites uns nous *vertex-shader* i *fragment-shader*? Raona on és calcula la il·luminació i modifica convenientment els fitxers de la pràctica. Implementa la possibilitat de fer *flat shading* amb i sense textura segons es premi la tecla 3 o Majúscules+Tecla 3.
6. Implementa el *Toon-shading* quan es premi la tecla 4. Cal que implementis la llum direccional per a poder realitzar el *toon-shading.* Quina diferència hi ha amb el *Gouraud-shading*? On l'has de codificar? Necessites uns nous *vertex-shader* i *fragment-shader*? Raona on és calcula la il·luminació i modifica convenientment els fitxers de la pràctica. Implementa la possibilitat de fer *flat shading* amb i sense textura segons es premi la tecla 4 o Majúscules+Tecla 4.
7. Després implementa l'atenuació amb profunditat a cadascun dels mètodes.

2. **Planificació del desenvolupament de la pràctica 3**

Les dates límit de lliurament de la pràctica 1 és els dies 18 i 19 de maig en el teu grup de pràctiques. A continuació es detalla la planificació aconsellada per a desenvolupar-la.

Abril	**27**	**28**	**29**	**30**	**1**	**Enunciat de la pràctica 2**
Maig	4	5	6	7	8	Punts 1, 2, 3.1, 3.2, 3.3 implementats
	11	12	13	14	15	Punts 3.4, 3.5 implementats
	18	19				Lliurement de la Pràctica 3