

Laboratori IDI
2014-2015-Q1

Bloc 4: Materials i Llums

IDI 2014-2015 1Q

Introducció. Resum

- 2 sessions. Utilitzar l'aplicació del bloc3
- Entendre els models empírics d'il·luminació i el càlcul de la il·luminació en OpenGL.

Sessió 1:

- Afegir materials als objectes.
 - Entendre el significat de les constants.
- Normal per cara versus normal per vèrtex.
- Llum de defecte: llum de càmera. La vostra llum de càmera.
 - Entendre l'assignació de colors.

Sessió 2:

- Afegir altres llums:
 - Posicionament de les llums => importància de la declaració de la posició de la llum en el codi=> llum de càmera, escena,...
- Llum d'escena, llum en el patricio.
- Crear les funcionalitats demanades

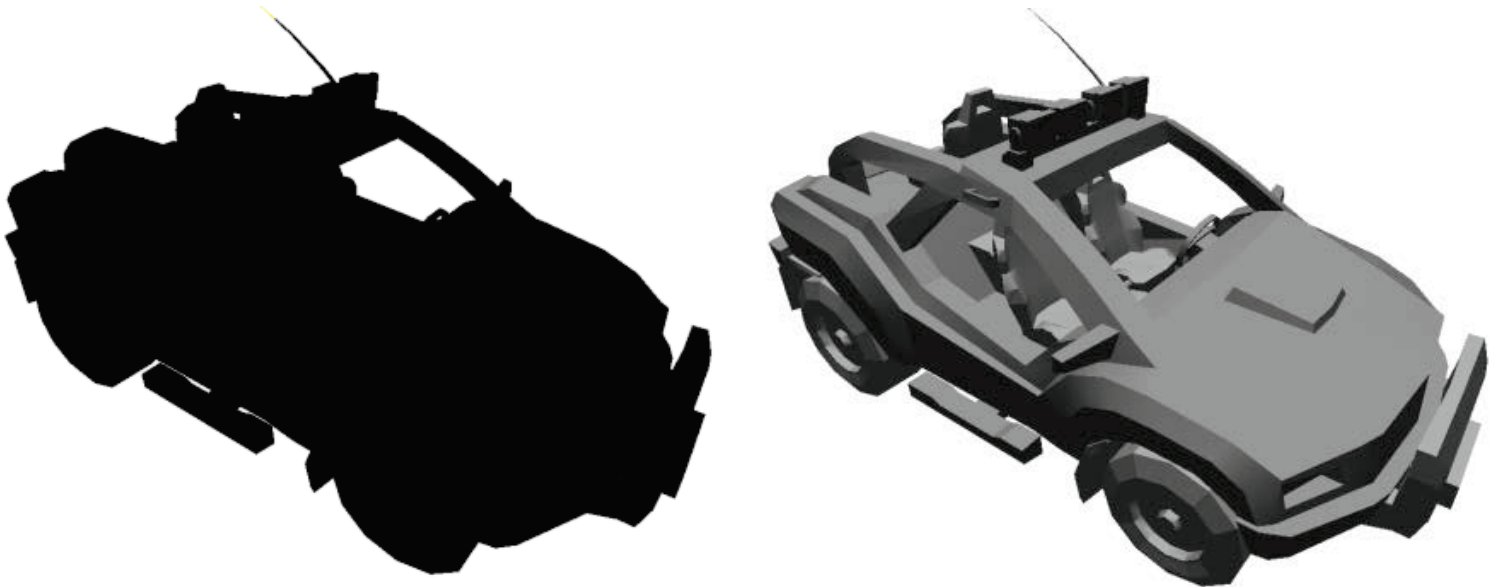
Realisme bàsic

- El·liminació de parts ocultes (suposant cares opaques)
- Il·luminació amb models locals o empírics:
 - Materials Cares
 - Constants empíriques
 - Focus de llum –puntuals-
 - Posicionament
 - Color
 - Activació

Depth test (ho teniu en bloc 2)




- Algoritme de z-buffer
 - **glEnable(GL_DEPTH_TEST);**
 - Esborrar el buffer de profunditat:
 - `glClear(.... | GL_DEPTH_BUFFER_BIT);`
 - En `glutInitDisplayMode` afegir:
 - `| GLUT_DEPTH`
- Pintar les cares en mode “fill” utilitzant:
`void glPolygonMode(GLenum face, GLenum mode)`
 - face: les cares a les que ens referim
 - **GL_FRONT_AND_BACK**, `GL_FRONT`, `GL_BACK`
 - mode: mode de dibuix
 - `GL_POINT`, `GL_LINE`, **GL_FILL**

Il·luminació



OpenGL pot dibuixar polígons il·luminats amb els models empírics d'il·luminació.

Resum Models Empírics

Color d'un punt degut a...	Depèn de la normal?	Depèn de l'observador?	Exemple
Model ambient	No	No	
Model difús	Sí	No	
Model especular	Sí	Sí	

OpenGL:

$$I_{\lambda}(P) = I_{a\lambda}k_{a\lambda} + \sum_i (I_{fa_i\lambda}k_{a\lambda} + I_{fd_i\lambda}k_{d\lambda}\cos(\Phi_i)) + \sum_i (I_{fs_i\lambda}k_{s\lambda}\cos^n(\alpha_i))$$

OpenGL i Il·luminació (1)

Abans d'enviar a pintar una cara cal que estiguin definides en el context gràfic:

- ⇒ Les constants del seu material.
- ⇒ Els focus de llum actius: color i posició.
- ⇒ I que estigui habilitada (enable) la il·luminació i els focus actius.

També cal especificar abans de la crida a `glVertex3dv(...)`·:

- Una normal per cara (mateixa per tots els vèrtex) o una normal per a cada vèrtex.

OpenGL i Il·luminació (2)

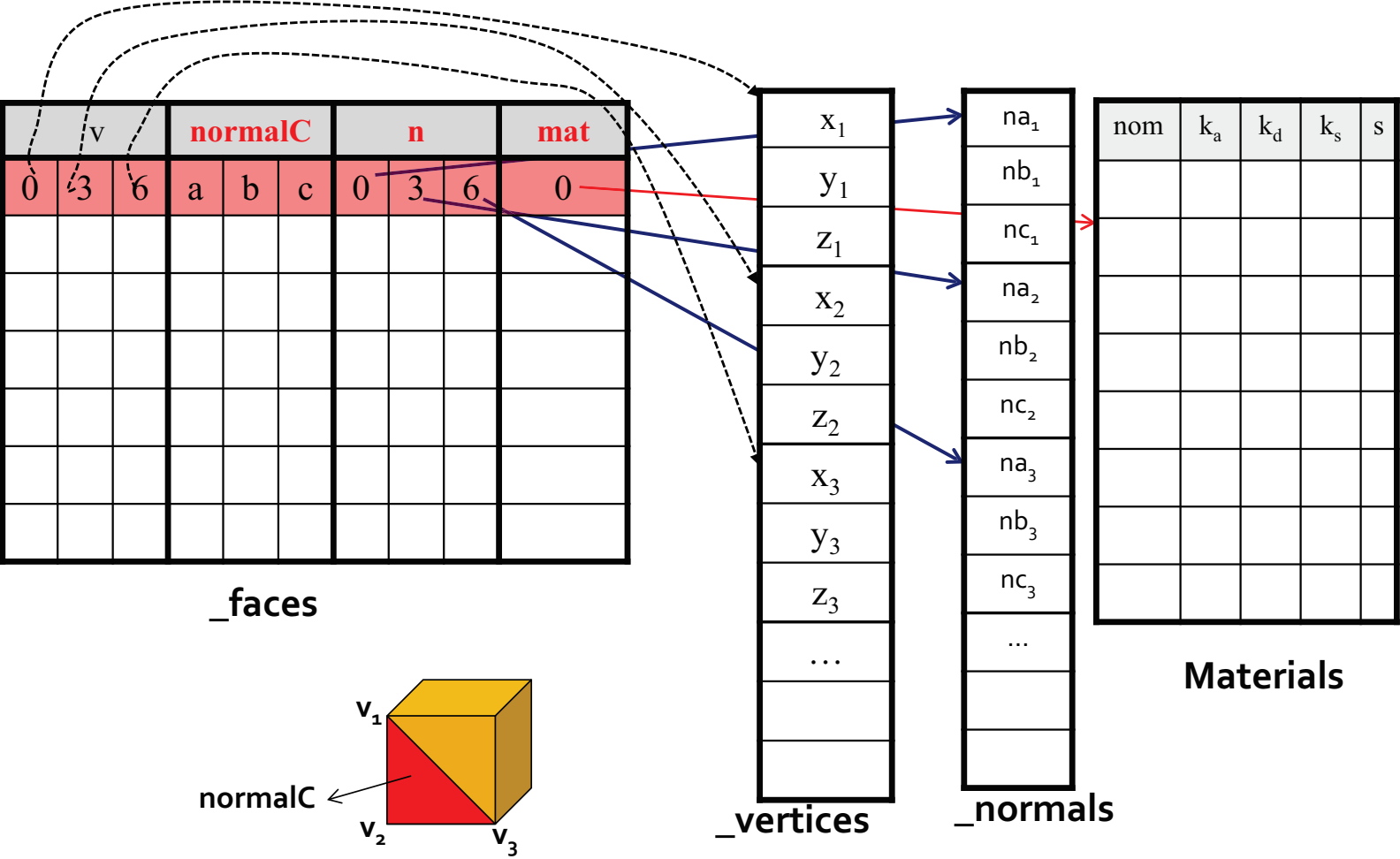
- Activar la il·luminació:
 - ⇒ `glEnable(GL_LIGHTING);`
- Encendre un llum:
 - ⇒ `glEnable(GL_LIGHT#);` De moment **GL_LIGHT0**
 - llum per defecte d'OpenGL, analitzeu els seus colors en el manual-, és un llum de càmera)
- Es poden tenir fins a 8 llums
 - ⇒ `GL_LIGHT0, GL_LIGHT1, ... , GL_LIGHT7`
- El color es calcular per cada vèrtex
 - ⇒ Modes empírics d'il·luminació si `GL_LIGHTING` activat
 - ⇒ Darrer color definit amb `glColor*()` si
 - ⇒ `GL_LIGHTING` desactivat

Open GL i Il·luminació (3)

- `glLight*(light, pname, param)`
 - `light`: la llum a modificar
 - `GL_LIGHT0, GL_LIGHT1, ... , GL_LIGHT7`
 - `pname`: paràmetre a modificar
 - `GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR`
 - `GL_POSITION`
 - Vector de 4 components.
 - Si la 4^a és 1, les tres primeres components s'interpreten com coordenades de la posició de la llum
 - Si la 4^a és 0, s'interpreten com la direcció des de la qual prové la llum

OpenGL i Materials (1)

- `glMaterial*(face, pname, param)`
 - face: tipus de cara que afecta
 - `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK`
 - pname: propietat a modificar
 - `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR`
 - Vector de 4 floats
 - `glMaterialfv()`
 - `GL_SHININESS`: exponent de Phong
 - Enters o floats, entre 0 i 128
 - `glMaterialf()`



Materials: classe model

- El material associat a les cares del model
- **Abans d'enviar els vèrtexs d'una cara** (de glVertex)
 - 4 crides a glMaterial*() per a inicialitzar les constants empíriques.
- Per a cada cara en el vector Faces de la classe model s'emmagatzema : **índex al vector de materials de la cara.**
 - Repasseu la crida aque teniu per accedir al color difús
 - Incloeu les crides només quan el material d'una cara és diferent del de l'anterior => el seu índex és diferent.

OpenGL i Materials i Objectes GLUT

- En el cas dels objectes glut no es possible declarar propietats per cada cara; s'ha de fer a nivell d'objecte.
- S'han de fer les crides a `glMaterial()` abans de cridar a `glutSolidSphere`, `gluSolidCube`...
- Penseu les seves inicialitzacions en funció del material descrit en el guió.
- En el cas d'altres polígons creats per vosaltres (si en teniu), també cal que declareu les seves propietats abans de pintar-los.

Normals

- Els models empírics d'il·luminació fan servir la normal als vèrtexs per a calcular els angles d'incidència i reflexió.

$$I_{\lambda}(P) = I_{a\lambda} k_{a\lambda} + \sum_i (I_{fa_i\lambda} k_{a\lambda} + I_{fd_i\lambda} k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{fs_i\lambda} k_{s\lambda} \cos^n(\alpha_i))$$

- Cal proporcionar aquesta normal:
 - Abans de cada crida a `glVertex*()`, cal afegir una crida a `glNormal*()`
- Sempre les Normals han d'estar normalitzades, per a no preocupar-nos:
 - `glEnable (GL_NORMALIZE);`

Normals. Classe model

- El model OBJ llegit tindrà sempre normals per cara definides (compartides pels 3 vèrtexs de la cara) en el atribut *normalC* del vector de *Faces*
- Alguns models tenen (també) normals per vèrtex. En el vector *Normals*. Per accedir a les 3 components de la normal igual que feu per a accedir a les 3 coordenades d'un vèrtex d'una cara.
 - Abans de pintar cada vèrtex cal especificar la seva normal.
 - Aparença més suau.
 - Mireu en el guió els objectes OBJ que tenen normal per vèrtex.
 - Feu un codi robust, si un model no té normals per vèrtexs e intenteu accedir a elles, l'aplicació hauria de seguir funcionant.

Primeres Proves (1)

- ★ Especifiqueu el material => 4 crides a `glMaterial()`
 - ✦ per les cares de l'objecte OBJ (segons el seu material de la classe model)
 - ✦ pels objectes glut (seguint especificacions del guió)
 - ✦ altres objectes –si en teniu- (seguint les especificacions del guió).
- ★ Especifiqueu la Normal de les cares dels objectes OBJ=> `glNormal()`
- ★ Activeu la il·luminació.
- ★ Activeu el Llum 0 (llum per defecte d'OpenGL)
- ★ Mireu d'entendre el resultat de la ll·luminació:
 - Moure la càmera: canvia el color dels objectes?
 - Proveu moure la càmera en primera persona: canvia el color dels objectes? Igual?
 - Modifiqueu les constants del terra: primer difús, després molt especular. Noteu algun canvi?

Primeres Proves (2)

- Modifiqueu el codi per a permetre Normal per Vèrtex.
 - ✦ Proveu communtar entre Normal per vèrtex i Normal per cara en algun objecte.
- Canvieu els colors de la llum 0.
 - ✦ Enteneu el resultat?.
- Programeu totes les tecles d'activació i desactivació que indica el guió (en seccions 1 i 2).
- Netejeu (si s'escau) el vostre codi per a garantir que no s'estan fent més crides de les necessàries per a definir els colors dels llums