Laboratori IDI: OpenGL, bloc 4

Professors d'IDI, 2012-13.Q2

13 de maig de 2013

Igual que els blocs 1, 2, i 3 aquesta part està pensada per a que la feu pausadament, experimentant amb l'efecte de les diferents crides i paràmetres. Disposareu d'una sessió de laboratori per a completar-la. També com en laboratoris anteriors, hem intercalat el signe '►' per a senyalar punts específics en què es planteja un exercici o quelcom que necessita d'experimentació o proves per part vostra. No us estigueu de fer altres experiments, naturalment!

Insistim també que cal que mireu de produir codi net, que més endavant pugueu aprofitar. Mireu de deixar una aplicació "neta" al final del bloc, en què es puguin activar les diferents funcionalitats (o almenys una versió de cadascuna) via tecles o botons del ratolí (amb modificadors).

1 Introducció

En aquest bloc anem a intentar veure les escenes amb cert realisme. Això comporta pintar l'escena amb parts amagades, amb i·luminació (focus de llums) i considerar els materials dels objectes.

Feu servir l'escena i l'aplicació del bloc 3, amb totes les funcionalitats que tingueu implementades. Per entendre els conceptes d'aquest bloc, com a mínim cal poder carregar l'escena, definir una càmera arbitraria que veu tota l'escena i que es pot moure interactivament.

Reviseu les instruccions del bloc 3 per a activar, si no ho heu fet encara, l'el·limació de parts amagades i per pintar les cares amb omplert de polígons.

2 Materials

Per a poder dibuixar objectes amb més realisme que fins ara, caldrà que descrivim més en detall les propietats dels materials de què estan fets. OpenGL pot dibuixar polígons il·luminats amb els models empírics d'il·luminació que heu estudiat a classe, però per a què ho faci haurem d'afegir la descripció d'aquests materials.

Tingueu present que quan dibuixem, la il·luminació d'OpenGL pot estar en un de dos estats: activada o no. Aquests estats es canvien mitjançant les funcions `glEnable(GL_LIGHTING)` i `glDisable(GL_LIGHTING)`).[1] ►Activeu la il·luminació, per exemple, a la inicialització. A més caldrà que encengueu alguna llum. ►Afegiu la crida `glEnable(GL_LIGHT0)` a la inicialització (tindreu la llum de defecte encesa, mireu en el manual les seves caraterístiques). No cal que feu el corresponent `glDisable()` en cap cas, llevat que vulgueu apagar aquest llum.

Quan OpenGL té activat el mode `GL_LIGHTING`, fa servir els models empírics d'il·luminació per a calcular els colors dels vèrtexs; en canvi, si està desactivat, fa servir el darrer color que haguem definit amb `glColor*()`. Si activeu la il·luminació, per tant, les crides a `glColor*()` deixaran de tenir efecte (fins que la torneu a desactivar).

Per a fer servir la il·luminació, i aprofitar els materials descrits als arxius que carreguem, haurem d'afegir a la nostra aplicació les crides convenients dins del bucle de dibuix. La manera en que això funciona és com sol ser en OpenGL: disposem d'unes funcions (`glMaterial*()`) que modifiquen l'estat d'OpenGL. Quan enviem pel *pipeline* un vèrtex, aquest s'il·luminarà d'acord amb els materials que hi hagi en aquell moment al context gràfic. Per tant, abans d'enviar cada vèrtex, hem d'assegurar-nos que el material actiu és el correcte. Per a assignar un nou material actiu, fem un cert nombre de crides a `glMaterial*()`, una per cada propietat que volem modificar. Els paràmetres d'aquesta crida són el tipus de cares que afecten (igual que en el cas de `glPolygonMode()`, poden ser `GL_FRONT`, `GL_BACK` o `GL_FRONT_AND_BACK`), la propietat del material que volem assignar (que pot ser `GL_AMBIENT`, `GL_DIFFUSE`, `GL_SPECULAR` o `GL_SHININESS`, entre d'altres), i els nous valors. En el cas de les primeres tres propietats, que corresponen al color ambient, difús i especular, el valor que fareu servir serà un vector de quatre `float`s (i fareu servir `glMaterialfv()`). Per a assignar el quart paràmetre, `GL_SHININESS`, que dóna l'exponent de

[1] `glEnable` i `glDisable()` admeten una multitud d'altres valors del paràmetre, que activen i desactiven diferents opcions i processos que pot realitzar el servidor d'OpenGL. Consulteu la pàgina del manual d'aquestes funcions per a saber més, però tingueu present que moltes d'aquelles opcions no us seran significatives en aquest punt.

Phong a fer servir en el càlcul de la il·luminació, podeu fer servir enters o `floats`; el valor, però, ha d'estar entre 0 i 128.

Tal com s'emmagatzemen els models a la classe `Model` que us proporcionem, els materials estan associats a les cares del model. Així, ►abans d'enviar pel pipeline els vèrtexs d'una cara, haureu de fer les quatre crides a `glMaterial*()` per a assignar el material corresponent. Com que el que emmagatzemem és en realitat sols un enter que indica la posició al vector materials de les dades, podeu fer una petita optimització incloent aquestes crides sols quan el material d'una cara és diferent del de l'anterior, amb el simple mecanisme de guardar un enter que indica l'índex del material anterior, i comparar-lo amb l'actual. ►El terra haurà de ser d'un material brillant blau (ha de tenir reflexió especular).

Com que els models empírics d'il·luminació fan servir la normal a la superfície en que incideix la llum, l'haurem de proporcionar també. ►Per tant, haureu d'afegir, abans de cada crida a `glVertex*()`, una crida a `glNormal*()` per a emmagatzemar la normal actual al context gràfic. El model que heu llegit de disc, tal com hem indicat, tindrà sempre normals per cara definides (i compartides pels tres vèrtexs de la cara, pel que no cal tornar a enviar la normal per cada vèrtex; però si el model les conté, també tindreu definides normals per vèrtex, que podeu fer servir per a pintar el model amb una aparença més suau). ►carregueu en l'aplicació desenvolupada en el bloc 3 un model que contingui informació de normals (com per exemple `porsche.obj`, `f-16.obj` o `cow.obj`), i feu que amb una tecla es pugui commutar entre fer servir normals per cara i normals per vèrtex, i observeu les diferències resultants.

Per a que tot el nostre esforç doni fruits, però, recordeu que caldrà que estigui activat el *flag* `GL_LIGHTING` (cosa que podeu fer en la inicialització, fins que implementeu alguna tecla per a canviar d'un mode a l'altre).

Després d'aquesta feinada, hauríeu de veure els models que carregueu amb els colors que els seus dissenyadors hagin assignat (si ho han fet).

3 Llums

Per a què tot l'anterior funcioni, cal que hi hagi llums a l'escena. Si us ha funcionat és perquè per defecte OpenGL s'inicialitza amb una llum (`GL_LIGHT0`) col·locada a la càmera i inicialitzada amb uns valors per defecte. Però podem modificar aquests valors, posicionar-la en altres llocs, o fins i tot encendre i apagar independentment fins a 8 llums, anomenades `GL_LIGHT0` fins a `GL_LIGHT7`[2]. El mecanisme és semblant al dels materials, però aquest cop fem servir `glLight*()` (que també existeix en les variants `f`, `i`, `fv` i `fi`). Aquestes crides tenen tres paràmetres, que designen la llum a modificar, el paràmetre específic que volem assignar, i el nou valor. Els paràmetres específics poden ser, entre d'altres, `GL_AMBIENT`, `GL_DIFFUSE` i `GL_SPECULAR` (que com en els materials designen els corresponents colors, però aquest cop de la llum), o `GL_POSITION` (per a donar una nova posició de la llum). Hi ha d'altres paràmetres per a definir atenuacions i *spotlights* que no veurem en aquest laboratori (però podeu explorar si voleu).

Aquestes crides funcionen, pel que fa als colors de la llum, de manera semblant als materials. Si la llum no ha de canviar de color o intensitat, ►podeu assignar-los durant la inicialització i oblidar-los (recordeu fer servir vectors de quatre floats, i la variant `glLightfv()`).

El cas de la posició és una mica diferent. Quan executeu la crida `glLightfv(GL_LIGHTx, GL_POSITION, pos)`, **cal** també que `pos` contingui una posició en coordenades homogènies (amb quatre components). Si la quarta és 1, les tres primeres components s'interpretaran com coordenades de la posició de la llum. Si és zero, s'interpretaran com components d'un vector que indica la direcció des de la qual prové la llum (per a modelar llums molt distants, com la del sol).

Altra diferència important és que **en el moment de fer la crida** OpenGL multiplicarà aquesta posició per la matriu de ModelView actual (la que estigui activa en aquell moment) per a convertir les coordenades a coordenades d'ull. **Si volem posicionar una llum en coordenades de món** (per exemple, una farola en una escena), haurem de repetir aquesta crida cada vegada que la ModelView canviï. **Si en canvi volem que una llum tingui una posició donada en coordenades d'ull** (per exemple per a simular llums solidàries amb la càmera) haurem d'assegurar-nos que la matriu de ModelView conté la identitat en el moment de cridar a `glLight*()` (per exemple fent un `glPushMatrix()` i un `glLoadIdentity()` just abans de cridar `glLight*`, i un `glPopMatrix()` just després per a restablir la transformació que hi havia).

►La llum 0, definida per defecte per OpenGL, és una llum de càmera (per què?). Programeu una tecla que permeti activar-la i desactivar-la. Analitzeu els seus colors i, per entendre-ho tot, és convenient que els inicialitzeu amb valors coneguts per vosaltres. ►Si en comptes de voler fer servir la llum 0 com llum de càmera, feu servir altre llum (per exemple, la llum 1), on indicaríeu la seva posició?

[2] El standard d'OpenGL indica que tota implementació d'OpenGL ha de suportar almenys vuit llums, però com cada llum addicional representa un esforç computacional no menystenible, tots els fabricants opten per suportar exactament vuit; així compleixen el standard sense degradar el rendiment.

►Col·loqueu una llum a una alçada fixa i petita per sobre **d'una de les cantonades** del terra de la vostra aplicació (en coordenades de món, pel que s'ha explicat més amunt, haureu de repetir la crida a `glLightfv(..., GL_POSITION,...)` cada vegada que es modifiqui la matriu de ModelView). Analitzeu el que veieu; sabeu explicar-ho?

►Programeu una tecla que us permeti modificar la posició de la llum, per a què estigui -successivament- sobre el següent vèrtex (en el sentit cíclic dels vèrtexs del rectangle) a la mateixa alçada; a més, abans de tornar al vèrtex inicial, és situarà -a la mateixa alçada- sobre el centre del terra. El que veieu és consistent amb la vostra explicació?

►Se us acut com millorar la qualitat d'aquestes imatges, quant a la il·luminació?

4 Lliurament

Cal que, com a molt tard, el dia abans de l'examen de laboratori lliureu l'aplicació resultant del bloc 3 ampliada amb les següents funcionalitats:

- A l'engegar l'aplicació mostri el terra i un objecte OBJ amb els materials que tenen definits i iluminats per una llum de càmera.
- Prement la tecla 'c' es pugui activar/desactivar teclat el llum de càmera.
- Prement la tecla 'f' es pugui encendre i apagar un llum d'escena ubicat sobre una cantonada del terra.
- Prement la tecla 's' la posició del focus d'escena passi a estar sobre el vèrtex següent del terra.