Laboratori IDI: OpenGL, bloc 1

Professors d'IDI, 2013-14.Q1

9 de setembre de 2013

En aquest laboratori us proposem un seguit d'experiments que us aniran mostrant de forma incremental aspectes de les APIs d'OpenGL i de `glut`, una llibreria lleugera que farem servir en la construcció d'aplicacions OpenGL. Per a què aquest laboratori us sigui de profit, no l'heu de recórrer a vol d'ocell, seguint mecànicament les passes que us enumerem, sinó que us heu d'aturar en cada apartat; a cada modificació proposada, construïu un nou executable i compareu el resultat d'executar-lo amb les passes anteriors. Experimenteu modificant valors de paràmetres, ordres de crides, i qualsevol altre aspecte possible, mirant d'entendre què passa, fent conjectures sobre el funcionament del programa, i dissenyant els vostres propis experiments per a confirmar-les. Si ho feu així, hauríeu d'adquirir ràpidament una comprensió ferma del funcionament de la llibreria que us permetrà abordar reptes més complexes.

Al llarg dels diferents guions us anirem indicant les instruccions de glut i d'OpenGL bàsiques per al seu desenvolupament, tanmateix la seva especificació completa l'haureu de cercar als seus manuals que teniu accessibles on-line. Aprofitem per recomanar-vos que utilitzeu, sempre que us sigui possible, els tipus pre-definits d'OpenGL i glut, us facilitaran la portabilitat de les aplicacions.

En aquest primer bloc, presentem els primers experiments que us proposem, que us portaran a construir una aplicació molt simple des de zero, i aquesta aplicació, donada la seva senzillesa, serà un bon banc de proves per a ulteriors experiments. És molt important que entengueu l'estructura d'una aplicació glut i el tractament d'events. Hem intercalat el signe '►' per a assenyalar punts específics en què es planteja un exercici o quelcom que necessita experimentació per part vostra.

1 Primeres passes

Per a realitzar aquesta pràctica, ►el primer que us caldrà és entrar el següent codi, a partir del qual fareu els diferents experiments que us proposarem:

```
1  #if defined(__APPLE__)
2    #include <OpenGL/OpenGL.h>
3    #include <GLUT/GLUT.h>
4  #else
5    #include <GL/gl.h>
6    #include <GL/freeglut.h>
7  #endif
8  void refresh(void)
9  {
10
11 }
12
13 int main(int argc, const char * argv[])
14 {
15     glutInit(&argc, (char **)argv);
16     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
17     glutInitWindowSize (600, 600);
18     glutCreateWindow("IDI: Practiques OpenGL");
19     glutDisplayFunc (refresh);
20     glutMainLoop();
21     return 0;
22 }
```

Aprofiteu per a considerar el contingut del que esteu introduint. Fixeu-vos que les línies 1–7 no són estrictament necessàries, a menys que vulgueu que el vostre codi compili tant a MacOSX com a linux. Tanmateix, si us plau, poseu-les totes per què ens facilita la correcció.

Als laboratoris de la fib i en una instal·lació *standard* de linux heu de posar a la cua de la vostra comanda de compilació/muntatge `-lglut -lGL`. Us recomanem que feu un `Makefile` per a facilitar la recompilació del vostre codi, que haureu de fer nombroses vegades.

Mireu ara la funció principal `main()`. Consisteix d'una sèrie d'inicialitzacions de la llibreria `glut`; especialment fixeu-vos en `glutDisplayFunc`, que exemplifica la forma en què en `glut` es declaren *callbacks*, en aquest cas un que serà executat cada vegada que calgui refrescar el contingut de la finestra. Experimenteu per a veure quines crides són indispensables, i si l'ordre importa.

Depenent de la màquina en què ho feu, pot ser que us mostri una finestra negra, o que us mostri una finestra amb soroll (els continguts que hi hagués a la memòria de vídeo assignada a la finestra). Podeu triar un color de fons diferent del negre fent servir `glClearColor()`. ►Creeu una funció `initGL()` per a encapsular les inicialitzacions d'OpenGL, i crideu-la des del `main()`. `glClearColor` declara el color a fer servir per a inicialitzar el *frame buffer*. Es fa servir quan hom crida `glClear` amb el bit de `GL_COLOR_BUFFER_BIT` activat. ►Definiu ara el cos de la funció `refresh()` així:

```
1 void refresh(void)
2 {
3   glClear(GL_COLOR_BUFFER_BIT);
4   glutSwapBuffers();
5 }
```

La segona crida que hem afegit intercanvia els *front* i *back buffers*.

2 Dibuixant...

Ara estem a punt per a començar a dibuixar. ►Afegiu a `refresh()` el codi OpenGL necessari per a dibuixar un triangle amb els vèrtexs a $(-\frac{1}{2}, -\frac{1}{3}, 0)$, $(\frac{1}{2}, -\frac{1}{3}, 0)$ i a $(0, \frac{2}{3}, 0)$. Podeu utilitzar com a mostra l'exemple de les transparències.

2.1 Redimensionaments i deformacions

Un cop aconseguiu veure el triangle, proveu de redimensionar la finestra. Què succeeix? Si no us n'adoneu, proveu de redimensionar-la radicalment!

Per a aquestes circumstàncies, `glut` disposa d'un altre *callback* que podeu definir cridant `glutReshapeFunc()`. Té un únic paràmetre que ha de ser un apuntador a una funció amb dos paràmetres enters i que no retorna res, la qual serà cridada cada vegada que canviï la mida de la finestra. Proveu què passa si poseu una funció de *callback* que no fa res. Els dos paràmetres del *callback* us informen en cada moment de les mides de la finestra. Proveu de definir el cos de la funció cridant `glViewport(0, 0, ample, alt)`. ►Després, mireu de redefinir la funció de *callback* de forma que el triangle aparegui centrat a la finestra, i sense deformacions en un viewport que ocupi el màxim possible de la finestra.

2.2 Retallat

Un cop hagueu aconseguit dibuixar el triangle sense deformació i centrat, aprofitant tant bé com heu pogut la superfície de la finestra, passeu a experimentar amb les coordenades dels vèrtexs del triangle. ►Canvieu els valors de les z dels vèrtexs; proveu en cada cas quin efecte té el canvi sobre la imatge. Què passa si poseu una $z = 0.9$ al vèrtex central, i $z = 1.1$ als vèrtexs inferiors? Què passa si intercanvieu aquests valors de z? Què passa si totes les z són 0.9? I si totes són 1.1?

3 Esdeveniments de ratolí

Per a què les nostres aplicacions puguin tenir interès, ben segur ens caldrà poder-hi interactuar. `glut` proporciona *callbacks* per a rebre els esdeveniments del ratolí. Podeu afegir, a sota de la crida a `glutDisplayFunc` una altra a `glutMouseFunc` per a registrar un nou *callback*, que rebrà quatre paràmetres: un enter que identifica el botó que s'ha premut, un segon enter que indica si l'esdeveniment consisteix en que s'ha premut el botó, o en que se l'ha deixat anar, i finalment dos enters més que donen les coordenades en píxels on es trobava el ratolí en el moment de produir-se l'esdeveniment. També hi ha `glutMotionFunc` per a subscriure'ns als esdeveniments d'arrossegament del ratolí (moure'l amb un botó premut). El corresponent *callback* rebrà cada cop dos enters, indicant una nova posició del ratolí respecte un sistema de coordenades amb origen a la cantonada superior esquerra de la finestra.

Fent servir aquests nous *callbacks*, ►modifiqueu el programa anterior per a què fent click amb el ratolí i arrossegant es pugui modificar el color de fons (per exemple, dins d'un mateix color, triar que sigui més clar o més fosc arrossegant el ratolí). Des del propi *callback*, necessitareu fer que es refresqui la finestra. No crideu directament la vostra funció

`refresh()`; la llibreria proporciona una crida `glutPostRedisplay()` que marca la finestra actual com "necessita refrescar-se". La següent iteració del bucle d'events desencadenarà una crida a `refresh()`, però tant sols es farà una vegada, encara que diversos *callbacks* hagin demanat un refresc (i per aquest motiu és la manera correcta de fer-ho).

4 Esdeveniments de teclat

`glut` també proporciona *callbacks* per a rebre els esdeveniments de teclat. Podeu afegir, a sota de la crida a `glutMouseFunc` una altra a `glutKeyboardFunc` per a registrar un nou *callback*, que rebrà tres paràmetres: un `unsigned char` amb el caràcter corresponent a la tecla en qüestió, i dos enters indicant la posició del ratolí a la finestra quan s'ha premut la tecla.

►Feu servir aquest *callback* per a proporcionar una ajuda quan hom prem la tecla 'h', que s'escriu per la terminal des de la que heu endegat el procés, i que prement la tecla 'ESC' tanqui l'aplicació. Més endavant afegirem altres tecles per a controlar diferents comportaments de l'aplicació.

5 Interaccions més elaborades

►Amb tots aquests ingredients, ara podem abordar interaccions més complexes. En aquest apartat, us demanem que afegiu la gestió d'estats, de tal manera que prement tecles puguem canviar d'un mode d'interacció a un altre (assigneu un mode al canvi de color de fons que ja heu programat, de manera que, per exemple, prement 'f' aconseguim que els següents esdeveniments de ratolí afectin el color de fons). Creeu un nou estat, per exemple amb la lletra 't' (per triangle) tal que després de prémer-la, els següents tres *clics* del ratolí defineixin les coordenades dels tres vèrtexs del triangle a pintar. El ratolí us donarà sols dues coordenades (en coordenades de la finestra, i ... ull que origen en la part superior esquerra d'aquesta), que haureu de convertir en coordenades de la vostra aplicació i enviar a pintar de forma que el vèrtex resulti dibuixat precisament al píxel en que s'ha fet el *click*. Podeu assignar arbitràriament la z de tots els punts entrats a zero. Què caldria per a tenir més triangles a l'aplicació?

A més de `G_TRIANGLES`, `glBegin()` admet altres primitives: `GL_POINTS`, `GL_LINES`, `GL_LINE_STRIP`, `GL_QUADS` i `GL_POLYGON`, entre d'altres. Si et queda temps, t'animem a afegir codi per a veure almenys algunes d'aquestes primitives (prement una altra tecla per a preludiar l'entrada dels punts corresponents amb el ratolí, i una tecla o botó especial per a l'últim). Amplieu a més el `help()` documentant cada tecla que afegiu.

6 Més color. Opcional però maco...

En la funció `refresh()`, proveu d'afegir al començament una crida a `glColor`. Després, proveu de fer-ho abans de cada crida a `glVertex`, amb colors diferents, primer diferents intensitats del mateix color, després amb colors de tonalitats completament diferents. Podreu veure com funciona la interpolació de colors a una primitiva (a OpenGL els colors estan associats als vèrtexs). Mireu d'entendre realment el que apareix a pantalla.

7 Resum

Al final de fer totes les proves, hauríeu de netejar el codi per a tenir una aplicació que:

- En iniciar-se pinta el triangle indicat en la secció 2 sense deformacions i ocupant el màxim de la finestra gràfica. Comporta haver definit els *callbacks* associats a `glutRefreshFunc()` i `glutReshapeFunc()`
- No deforma el triangle encara que es realitzi un resize de la finestra.
- Permet sortir de l'aplicació al picar 'ESC' i escriu per la terminal una ajuda en relació a les seves funcionalitats en picar una 'h'.
- Permet modificar l'estat de l'aplicació, de manera que en picar una 't' es poden introduir unes noves coordenades pels 3 vèrtexs del triangle amb el ratolí; i en picar una 'f' s'activa/desactiva la funcionalitat que permet modificar el color del fons en moure el ratolí.