Laboratori IDI: OpenGL, bloc 1

Professors d'IDI, 2023-24.Q2

12 de setembre de 2024

En els guions de laboratori et proposarem un seguit d'experiments i exercicis que t'aniran mostrant de forma incremental aspectes de les APIs d'OpenGL i de Qt. Per a què aquest laboratori et sigui de profit, no l'has de recórrer a vol d'ocell, seguint mecànicament les passes que t'enumerem, sinó que t'has d'aturar en cada apartat; a cada modificació proposada, construeix un nou executable i compara el resultat d'executar-lo amb les passes anteriors. Experimenta mirant d'entendre què passa. Si ho fas així, hauries d'adquirir ràpidament una comprensió ferma del funcionament de la llibreria que et permetrà abordar reptes més complexos.

Per a cada guió que et proporcionarem, tindràs un codi d'aplicació inicial que hauràs d'anar ampliant i modificant per resoldre els exercicis proposats. Cal que assisteixis a les classes de laboratori per tenir tota la informació necessària per poder resoldre els exercicis satisfactòriament.

En aquest primer bloc, començarem per un codi molt simple que només pinta un triangle, però que ens servirà per introduir les funcionalitats més bàsiques i necessàries de la llibreria d'OpenGL. Aquest codi el pots trobar a `~/assig/idi/blocs/bloc-1`

La durada d'aquest bloc és de 3 sessions de laboratori.

Hem intercalat el signe '►' que indica punts específics en què es planteja un exercici que és important que aconsegueixis implementar.

La idea d'aquests documents (en tindreu un per a cada bloc de laboratori) és que sigui una ***ampliació*** del que s'explica a classe, per tant és estrictament necessari que vingueu a classe (o que us preocupeu d'enterar-vos del que es fa a classe) perquè aquests guions no són en absolut autocontinguts, bàsicament només tenen els enunciats dels exercicis a resoldre.

Per a cada sessió, al costat tindreu el nombre d'hores de dedicació que cada una requereix. Fixeu-vos que no n'hi ha prou amb la dedicació de les hores de classe de laboratori per assolir els objectius, sinó que es requereix un conjunt d'hores de dedicació pròpia també (en aquestes hores es compta també l'estona d'explicació del professor a classe).

Sessió 1.1: QOpenGLWidget i estructura VBO — 3 hores

Un cop hem vist a classe la primera sessió del bloc 1, per començar amb els exercicis, ► el primer que et caldrà fer és compilar i executar l'exemple que et passem per veure què fa, i estudiar amb detall el codi a partir de l'explicació que s'ha fet a classe.

1.1 Exercicis:

1. ► Prova de modificar les coordenades dels vèrtexs del triangle per veure què passa, tenint en compte que el món que estem veient és aquell en que x, y i z pertanyen a [-1,1].

2. ► Modifica el triangle per fer un quadrat. Recorda que el que et cal pintar són triangles.

3. ► Modifica el VBO per fer que et pinti una caseta (tres triangles).

4. ► Afegeix al teu codi un nou objecte independent del primer (pot ser un triangle, un quadrat, el que vulguis). Fes que aquest nou objecte estigui definit usant un nou VAO i amb les dades dins d'un nou VBO. A l'hora de pintar has de pintar tots dos VAOs.

5. ► Juga amb els paràmetres de la crida `glViewport(...)`. Primer fes que el viewport en el que es pinta la teva escena sigui només la meitat de la finestra que tens disponible (amplada del viewport ample/2 i alçada alt/2). Després fes que la teva escena es pinti en dos viewports diferents, tots dos de mida (ample/2,alt/2), un que comenci en el pixel (0,0) i l'altra que comenci en el pixel (ample/2,alt/2).

Sessió 1.2: Shaders (Vertex + Fragment) 3 hores

Un cop hem vist a classe la segona sessió del bloc 1, abans de començar amb els exercicis d'aquesta sessió, ► primer cal que et miris el codi de l'esquelet i entenguis tot allò que tens afegit per poder usar un Vertex Shader i un Fragment Shader, i el propi codi dels dos shaders (vertex i fragment) que tens ja implementat.

1.2 Exercicis:

1. ► Implementa i usa un Fragment Shader que pinti els fragments de colors diferents depenent del quadrant del *viewport* en el que es pintin. Per exemple, tots els fragments del quadrant superior esquerre, vermells, els del quadrant superior dret, blaus, els del quadrant inferior esquerre grocs i els de l'inferior dret, verds.

2. ► Implementa i usa un Fragment Shader que pinti a franges horitzontals. És a dir, els fragments que queden en les franges que no volem pintar hauran de quedar descartats.

3. ► Implementa i usa un Vertex Shader que modifiqui les coordenades dels vèrtexs de manera que aquestes quedin multiplicades per 0.5. Quin efecte està fent?

4. ► Volem pintar cada vèrtex del nostre objecte (triangle, quadrat, etc.) d'un color diferent. Per fer això, caldrà construir un nou VBO amb informació dels colors per a cada un dels vèrtexs de l'objecte, i definir un atribut per al color que es passarà al Vertex Shader. El Vertex Shader per la seva banda haurà de rebre aquest atribut de color per a cada vèrtex i enviar de sortida el color per a que el rebi el Fragment Shader. El Fragment Shader rebrà el color interpolat i ha de pintar el fragment d'aquest color que rep.

Sessió 1.3: Transformacions Geomètriques + Interacció teclat 4 h

En la tercera sessió del bloc 1, abans de començar amb els exercicis d'aquesta sessió, primer cal que introdueixis al teu codi tot allò que s'ha explicat a la sessió de laboratori i que cal per poder passar un *uniform* al Vertex Shader. ► Afegeix un *uniform* de tipus float al Vertex Shader que el faràs servir per multiplicar les coordenades del vertex i que inicialment tindrà valor 0.5. Afegeix al teu programa la possibilitat de modificar aquest valor del *uniform* mitjançant les tecles 's' i 'd'.

1.3 Exercicis:

1. ► Modifica el teu programa per a que construeixi una matriu de translació a partir d'un vector de translació *(tx, ty, tz)* i la passi com a uniform al Vertex Shader, fent que aquest la utilitzi per multiplicar als vèrtexs. Pots fer que els valors *tx* i *ty* es puguin modificar mitjançant les fletxes del teclat (tecles Key_Left, Key_Right, Key_Up i Key_Down).

2. ► Modifica la transformació de l'exercici anterior de manera que als vèrtexs, després de la translació, se'ls apliqui una rotació de 45 graus (M_PI/4 radians) respecte l'eix Z. Utilitza color per vèrtex per veure millor l'efecte.

3. ► Ara fes que la transformació aplicada als vèrtexs sigui del revés, primer s'aplica al vèrtex una rotació de 45 graus i després una translació respecte al vector *(tx, ty, tz)*.

4. ► Afegeix a la transformació geomètrica dels vèrtexs un escalat uniforme amb un factor d'escala variable mitjançant teclat (com abans amb 's' i 'd'). Juga amb l'ordre de les transformacions.

5. ► Modifica l'escalat de l'exercici anterior per a què no sigui uniforme, és a dir, el factor d'escala serà diferent per a les components x i y (la component z de moment la podem deixar igual).

6. ► Recupera de la sessió 1 el tros de codi que permetia pintar dos objectes (obj1 i obj2) -si ja els tens no cal que facis res.
   Fes que, en prémer la tecla 'p', s'apliqui als dos objectes una rotació sobre el seu centre (cadascun el seu) però aquesta rotació ha de ser en sentit contrari l'un de l'altre. És a dir per a l'objecte obj1, l'angle de rotació s'incrementarà cada cop en 30 graus (M_PI/6 radians), mentre que per a l'objecte obj2, l'angle de rotació es decrementarà cada cop en 30 graus. Fixa't que això vol dir que cada objecte ha de disposar d'una matriu de transformació diferent quan el pintes. Com ho pots fer?