# Laboratori IDI: OpenGL, bloc 2

Professors d'IDI, 2016-17.Q1

10 de març de 2017

---

En aquest segon bloc, partirem d'un codi que pinta un polígon amb forma de caseta (amb un color diferent per a cada vèrtex). Aquest codi el pots trobar a `/assig/idi/blocs/bloc-2`. La idea és que primer de tot et familiaritzis amb el codi que et passem i entenguis què fa i perquè cal cada línia del codi.

**La durada d'aquest bloc és de 4 sessions de laboratori**.

## 1 Sessió 2.1: Transformacions de càmera i càrrega d'OBJs

Un cop hem vist a classe la sessió 4 de laboratori, per a començar amb els exercicis, ► el primer que et caldrà fer és compilar i executar l'exemple que et passem per a veure què fa, i estudiar amb detall el codi a partir de l'explicació que s'ha fet a classe.

Quan ja hagis mirat i entès l'exemple d'aquest bloc, fes **tots** els següents exercicis en l'ordre en què estan, perquè defineixen una guia a seguir per a poder entendre tots els passos.

### 1.1 Exercicis:

1. ► Afegeix al codi de l'exemple un mètode `projectTransform ()` que implementi la crida a perspective amb paràmetres FOV = M_PI/2, ra = 1, znear = 0.4, zfar = 3, i enviï el uniform corresponent de la matriu de projecció al vertex shader, que farà el corresponent amb ella.

   És normal el que es veu considerant que la posició de la càmera és el punt (0,0,0) (matriu identitat en view transform)?

2. ► Afegeix al codi de l'exemple un mètode `viewTransform ()` que implementi la crida a lookAt amb paràmetres OBS = (0,0,1), VRP = (0,0,0), UP = (0,1,0), i enviï el uniform corresponent de la matriu view al vertex shader, que farà el corresponent amb ella.

   Un cop tenim tota la càmera perspectiva definida podem convertir els paràmetres (OBS, VRP, up, FOV, ra, znear i zfar) en variables que s'inicialitzen en un mètode `ini_camera ()`. Aquest mètode a més d'inicialitzar aquestes variables farà les crides necessàries per a inicialitzar les matrius (`projectTransform ()` i `viewTransform ()`).

3. ► Modifica el paràmetre del vector UP i mira quin efecte té en la imatge que veus. Pots provar, per exemple, de posar-li valors (1,0,0), (-1,0,0) o (1,1,0).

4. ► Modifica l'exercici traient la caseta i pintant en el seu lloc l'objecte definit pel fitxer `HomerProves.obj`. Carrega l'objecte (load) abans de construir els buffers i construeix dos VBOs per a aquest objecte, un a partir de les dades dels seus vèrtexs que el retorna el mètode `VBO_vertices()` de

la classe Model i l'altre a partir de la informació del seu material que ens retorna el mètode `VBO_matdiff()` i que podem usar com a informació de color per a passar-la al vertex shader.

Recorda que per a veure correctament qualsevol objecte cal activar l'algorisme d'eliminació de parts amagades. Activa el Z-buffer com s'ha explicat a classe.

5. ► Afegeix una interacció de teclat de manera que amb la tecla R el Homer roti cada vegada 45 graus (M_PI/4 radians) respecte l'eix vertical Y.

6. ► Afegeix a l'escena un terra quadrat de mides 2 en X i en Z i situat a una alçada Y = -1.

   Assegura't que el terra no rota, és a dir, hauràs de tenir una transformació de model diferent per al terra i per al Homer.

## 2 Sessió 2.2: Euler i objecte qualsevol

Un cop hem vist a classe la sessió 2.2 de laboratori, podeu continuar amb la llista d'exercicis següents. Us aconsellem que us guardeu el resultat de l'exercici de la sessió anterior.

### 2.1 Exercicis:

1. ► Afegeix a l'exercici que mostra el terra i el HomerProves la implementació necessària per a que, quan l'usuari fa una redimensió de la finestra (*resize*), no hi hagi deformació de l'escena ni es retalli el que s'estava veient.

2. ► Fes un mètode que a partir de dos punts (punt-mínim i punt-màxim) d'una capsa contenidora d'una escena, calculi i guardi en variables globals de la classe (o en variables de sortida del mètode) el centre de la capsa contenidora i el radi de l'esfera que conté la capsa.

   Utilitza aquest mètode per a calcular el centre i radi de l'esfera que conté l'escena que estem visualitzant, en aquest cas, els punts mínim i màxim de l'escena es poden posar a mà perquè són coneguts sabent l'escena que es visualitza (punts extrems del terra + alçada del HomerProves, que és 2).

3. ► Modifica els paràmetres de les crides a lookAt del mètode `viewTransform ()` i a perspective del mètode `projectTransform ()` per a què usin les dades de l'esfera contenidora de l'escena calculada en l'exercici anterior. Amb això aconseguim una càmera en tercera persona que ens permet veure tota l'escena sencera i ocupant el màxim del viewport.

4. ► Ara volem visualitzar una instància del model `Patricio.obj`, però aquest model no està creat per a que els seus vèrtexs estiguin tots en el volum de visió que tenim definit fins ara. Has d'afegir al teu codi el càlcul de la capsa contenidora del model (a partir dels seus vèrtexs) i pintar el Patricio centrat a l'origen (sense escalar). Com que no es vol que el Patricio s'escali per a què hi càpiga al volum de visió que tenim, caldrà que calculis també els paràmetres d'una càmera perspectiva que et permeti veure aquest objecte centrat a l'origen, sencer, i ocupant el màxim del viewport.

   Què ha passat amb el terra que teníem? Elimina (o comenta) el seu pintat al mètode paintGL.

5. ► Modifica el mètode `viewTransform ()` per a afegir el càlcul de la matriu *view* a partir de les transformacions mitjançant angles d'Euler. Inicialitza aquests paràmetres per a veure exactament el que es veia a l'exercici anterior.

6. ► Afegeix la possibilitat que l'usuari pugui interactuar amb el ratolí per a modificar els angles que giren la càmera respecte els eixos Y i X (en les transformacions d'Euler). Quan l'usuari mou el ratolí en horitzontal d'esquerra a dreta la càmera s'ha de moure sobre l'esfera de visió

en direcció a la dreta. Quan l'usuari mou el ratolí en vertical de baix a dalt la càmera s'ha de moure sobre l'esfera de visió en direcció cap a dalt.

# 3 Sessió 2.3: Escena completa i càmera axonomètrica (ortogonal)

En aquesta tercera sessió del bloc 2, continueu amb la llista d'exercicis següents. Us aconsellem que us guardeu el resultat de l'exercici de la sessió anterior.

## 3.1 Exercicis:

1. ► Afegeix a la implementació del teu programa (del que tens fins ara) la possibilitat de fer zoom-in (amb la tecla Z) i zoom-out (amb la tecla X) de manera que es modifiqui l'angle FOV de la càmera perspectiva.

2. ► Ara, per a fer una escena completa, modifica el teu codi (pots fer una còpia i guardar el que tenies) de manera que la teva aplicació pinti una escena que té un terra centrat a l'origen de mida 4x4, amb 2 patricios que estan escalats de manera que la seva alçada és 1 i posicionats sobre el terra de manera que un té el centre de la base de la seva capsa contenidora al punt (1,0,1) i l'altre el té al punt (-1,0,-1). El primer Patricio estarà mirant cap a les Z positives (sense cap rotació) i l'altre estarà mirant cap a les Z negatives. Cal tenir definida una càmera perspectiva que vegi l'escena sencera, centrada, sense deformació i ocupant el màxim del viewport (fixa't que les mides de l'escena sencera són conegudes, per tant són coneguts els punts de la seva capsa contenidora).

3. ► Modifica el mètode `projectTransform ()` per a modificar el tipus d'òptica i fer una càmera axonomètrica (ortogonal). Fes que els paràmetres també siguin els adients per a que l'esfera contenidora de l'escena estigui completament dins del volum de visió.

4. ► Fes que el redimensionament de la finestra (*resize*) no deformi ni retalli tampoc quan s'usa aquest tiput de càmera (axonomètrica).

# 4 Sessió 2.4: Afegim elements d'interfície amb Qt

En aquesta última sessió del bloc el que farem és afegir elements d'interfície que permetin fer algunes de les coses que ja hem fet mitjançant interacció directa i d'altres. Cal haver vist primer les explicacions de classe d'aquesta sessió (2.4). Continueu amb els següents exercicis guardant-vos els que teníeu com a resultat de la sessió anterior.

***Nota:*** *D'aquesta sessió considerem important que feu com a mínim els exercicis 1, 2, 5 i 6. La resta també però els podeu considerar opcionals.*

## 4.1 Exercicis:

1. ► Afegeix a la interfície del teu programa un *Slider* i un *Spinbox* que estiguin sincronitzats entre ells i que permetin controlar l'angle d'obertura de la càmera (FOV) per a fer el Zoom. Caldrà que els dos elements de la interfície estiguin límitats en els seus valors de forma adient.

2. ► Afegeix a la interfície del teu programa un *Radio Button* que permeti canviar el model entre el Patricio i el Legoman. Fixa't que hauràs de reconstruir els buffers (VAO i VBOs) cada cop que hi hagi aquest canvi. Creus que seria millor tenir-los els dos carregats i decidir quin pintes en el moment de pintat?

3. ► Afegeix a la interfície del teu programa (del que tens fins ara) un *Radio Button* que permeti decidir entre usar l'òptica perspectiva o l'axonomètrica.

4. ► Afegeix a la interficie del teu programa un *Spinbox* que permeti modificar el factor d'escala del model. Fes que l'ús d'aquest element de la interfície sigui compatible amb l'ús de les tecles 'Z' i 'X' que ja tenies implementades per a fer el mateix.

5. ► Afegeix a la interfície del teu programa dos *Dials* que permetin controlar i modificar els angles d'Euler ($\Psi$ i $\Theta$) de la càmera. Afegeix també la possibilitat que si els angles es modifiquen de manera directa amb el ratolí es vegin afectats també els valors dels *Dials*.

6. ► Implementa una classe derivada de *QLabel* (MyLabel) com a classe pròpia que permeti mostrar mitjançant el color del seu *background* el color representat per 3 *Spinbox* que defineixen els valors de R, G i B del color mostrat.

   Afegeix aquesta MyLabel i els tres *Spinbox* a la teva interfície per a permetre decidir amb ells el color del terra.

7. ► Afegeix a la interfície del teu programa dos *Sliders*, un d'horitzontal col·locat sota de la finestra gràfica i ocupant tot l'espai que aquesta ocupa, i un de vertical, col·locat al costat de la finestra gràfica i ocupant tot l'espai que aquesta ocupa. Aquests dos *Sliders* volem que controlin les dimensions del viewport en el que pintem (paràmetres amplada i alçada de la crida a glViewport), de manera que el *Slider* horitzontal tindrà valors entre 0 i l'amplada de la finestra gràfica (width) i el *Slider* vertical tindrà valors entre 0 i l'alçada de la finestra gràfica (height).

   Quan l'usuari mou un d'aquests *Sliders* s'ha de modificar la dimensió corresponent del viewport en què es pinta la nostra imatge final, sense modificar l'origen del viewport que continuarà sent el píxel de la cantonada inferior esquerra.

   No cal que tingueu en compte el redimensionat de la finestra gràfica en aquest exercici, és a dir podeu considerar que no es modifica la finestra gràfica.