VIG - Examen Parcial - Quadrimestre Tardor - 2008/2009 - **Temps 1h 30m**

1. (1 punt) La llibreria Qt té un *widget* anomenat *QTimer* que emet un *signal* cada *n* milisegons. Concretament té el mètode `start(int n)` que inicia el *QTimer* i cada *n* milisegons llença el *signal* `void timeout()`. Suposeu que heu implementat un *widget* que conté tres botons representant un semàfor. La classe que representa el *widget* conté un *slot* `void cambiar()` que modifica el color dels botons. Aqest *slot* s'ha de cridar continuament cada segon per a què vagi canviant el color dels botons i així simular el funcionament del semàfor. Quines serien les intruccions necesàries per a utilitzar un *QTimer* que ens ho permetés? Heu d'incloure tant la declaració com el codi necessari per a que això funcioni.

La declaració de l'objecte QTimer podria ser un atribut del widget i per tant anar al fitxer meuwidget.h:

```
QTimer timer;
```

Al fitxer meuwidget.cpp i en el constructor del meuwidget podriem tenir tant la inicialització com la connexió, i seria el codi següent:

```
timer.start(1000);  // 1000 milisegons són 1 segon
connect (&timer, SIGNAL (timeout()), this, SLOT (canviar()));
```

2. (1.5 punts) Endevinalla: tenim tres objectes de colors purs diferents, però no sabem quins. El que sí sabem és que si il·luminem tots tres amb una font de llum groga, el primer objecte (Obj1) es veu de color groc, el segon (Obj2) es veu de color negre i el tercer (Obj3) de color groc. Si en canvi fem servir una font de llum cyan, el primer es veu cyan, el segon es veu blau i el tercer verd. Per a aquells objectes que sigui possible, digueu quin és el seu color corresponent i la seva codificació en HSB. Justifiqueu la resposta.

Sota llum groga: Obj1 es veu groc ==> Reflecteix R i G i no sabem res de B
Obj2 es veu negre ==> No reflecteix R ni G i no sabem res de B
Obj3 es veu groc ==> Reflecteix R i G i no sabem res de B

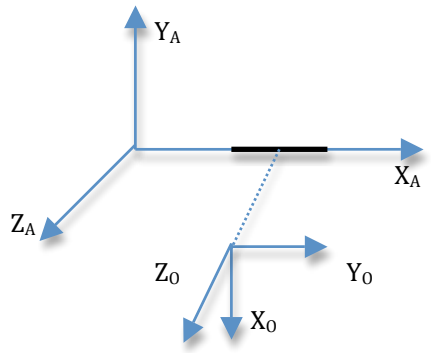
Sota llum cyan: Obj1 es veu cyan ==> Reflecteix G i B ==> Obj1 reflecteix R, G i B: **és blanc**
Obj2 es veu blau ==> Reflecteix B i no G ==> Obj1 només reflecteix B: **és blau**
Obj3 es veu verd ==> Reflecteix G i no B ==> Obj3 només reflecteix R i G: **és groc**

Obj1 és blanc, per tant la saturació és 0, el hue és irrellevant i la brillantor ha de ser 1:
HSB = (?, 0, 1)
Obj2 és blau, per tant la saturació ha de ser 1 (color pur) i la brillantor també, el hue del blau és 240:
HSB = (240, 1, 1)
Obj3 és groc, per tant la saturació també serà 1 igual que la brillantor. En aquest cas el hue és 60:
HSB = (60, 1, 1)

3. (2.5 punts) Inicialitzeu, adequadament, tots els paràmetres d'una càmera perspectiva (definida amb `gluLookAt` i `gluPerpective`) de manera que el segment definit pels vértexs (3,0,0) i (6,0,0) es vegi sencer en el *viewport* com segment vertical que passa pel centre del *viewport*. Indiqueu també les coordenades d'obervador i normalizades dels vèrtexs del segment. Justifiqueu la resposta.

- Per a que es vegi vertical, yobs ha de tenir la direcció del segment: up=(1,0,0)
- Per a que passi pel centre del viewport, zobs ha de passar pel centre del segment; una possibilitat es que VRP estigui en el centre del segment: VRP=(4.5,0,0)
- zobs ha de ser perpendicular a yobs; podria ser qualsevol recta en un pla paral·lel a YZ que passa per VRP; una possibilitat és posar a OBS=(4.5,0,3)
- znear com a molt por valdrà 3; posem znear=3
- zfar ha de ser superior a 3; posem zfar=4
- L'angle d'obertura ha de poder veure tot el segment. Com la distància de l'OBS al centre del segment és 3, i aquest és perpendicular a zobs (mirar figura): tg(alfa)= 1.5/3; alfa=arctg(1.5/3); FOV=2*alfa
- ra en aquest cas, donada la ubicació de la càmera i l'especificació del problema, és arbitrari; posem la ra del viewport.

- Les coordanes de l'observador dels punts (3,0,0) i (6,0,0) d'acord amb la ubicació de la càmera (veure figura) són: (0,-1,5,-3) i (0,1.5,-3)
- En quant a les coordeades normalitzades, com els punts estan situats en znear tindran zN=1 (o -1, segons conveni); com estan sobre una recta para·lela a yobs que passa per l'observador, la seva xN=0; com (6,0,0) està en la part superior del window tindrà yN= 1 i com (3,0,0) està en la inferior tindrà yN=-1.



4. (1 punt) Escriviu un tros de codi (instruccions OpenGL) que utilitzant transformacions geomètriques defineixi la mateixa matriu *ModelView* que la requerida en l'exercici anterior. Justifiqueu la resposta.

Per a ubicar la càmera seria necesari: traslladar-la al VRP; rotació de -90 graus (horaria) respecte z; translació 3 unitats respecte z. El que es el mateix, la TG a efectuar al segment és:
TG= T(0,0,-3)Gy(90º)T(-4.5,0,0).

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0,0,-3);
glRotatef(90,0,0,1);
glTranslatef(-4.5,0,0):
```

5. (1 punt) S'està visualitzant un objecte amb una càmera perspectiva. A l'avançar la càmera cap a l'objecte (només es modifica la posició de l'observador), s'obté un efecte semblant al *zoom*. Passa el mateix amb una càmera axonomètrica? Per què?

No s'obté el mateix efecte. Amb una càmera axonomètrica els punts es projecten en el window en la direcció de zobs; per tant, la seva projecció és independent de la posició de OBS (sempre i quan es mantingui la direcció de zobs). Per a aconseguir un zoom, cal escalar uniformement el window respecte el seu punt central. En la càmera perspectiva, la projecció dels punts es inversament proporcional a la seva distància a OBS (a la seva z en coordenades d'observador), per tant, al modificar OBS es modifica la seva projecció; si ens apropem, les seves coordenades projectades augmenten.

6. (1 punt) Donada la visualització d'una escena, quin és l'efecte que s'observa en la imatge obtinguda quan es fa un *pan*? Quins paràmetres de la càmera cal modificar per a fer un *pan*? Com s'han de modificar i per què?

La imatge es desplaça sobre la pantalla (viewport) cap a la dreta/esquerra, amunt/avall o una composició dels dos moviments; però no canvia la seva orientació respecte la càmera. Per tant, tenim l'efecte que mantenint la direcció de visió, la càmera es desplaça a esquerra/dreta i/o avall/amunt.
Caldrà desplaçar OBS i VRP sobre un pla paral·lel a YX de l'observador. Caldrà desplaçar OBS i VRP una distància proporcional al moviment del ratolí segons els eixos xobs i yobs:
OBS= **OBS** + inc_x* **xobs** + inc_y * **yobs** i **VRP**= **VRP** + inc_x* **xobs** + inc_y * **yobs**

D'aquesta manera, el window es desplaça en el pla de projecció i, per tant, es modifica el que es veu en el viewport però es maté la direcció de visió.

7. (1 punt) La trajectòria d'un cotxe es representa per una poligonal de 20 punts ubicada en el pla x-y. Es disposa de la informació de la caixa mínima contenidora del cotxe i d'una funció `pinta_coche()` que pinta un cotxe centrat a l'origen. Indica l'expressió de les transformacions geomètriques que cal efectuar al cotxe per a generar una animació que ubica successivament el cotxe en els vèrtexs de la trajectòria, amb un escalat uniforme decreixent proporcional a l'índex del vèrtex del polígon (el cotxe es va fent petit). Indiqueu també el codi OpenGL que, suposant una càmera ja inicialitzada, pinti el cotxe en la posició indicada.

Suposant l'índex `i` de la trajectòria, i sabent que `pinta_cotxe()` ja pinta el cotxe a l'origen de coordenades, les transformacions que caldria fer al cotxe són:

1) Escalar-lo de forma homogènia en, per exemple, `(1-i*0.05)`
2) Traslladar-lo a la posició `i` de la trajectòria

Si tenim un vector `trajectoria` on tenim guardats els vèrtexs de la trajectòria, l'expressió de la transformació geomètrica a aplicar al cotxe en l'índex `i` seria:

```
TG = T(trajectoria[i]) * S(1-i*0.05, 1-i*0.05, 1-i*0.05)
```

El codi OpenGL suposant que les matrius ja estan inicialitzades amb els paràmetres de la càmera seria:

```
glMatrixMode (GL_MODELVIEW);
for (int i=0; i<20; ++i) {
   float escalat = 1-i*0.05;
   glPushMatrix () ;
   glTranslatef (trajectoria[i].x, trajectoria[i].y, trajectoria[i].z);
   glScalef (escalat, escalat, escalat);
   pinta_cotxe ();
   glPopMatrix ();
}
```

8. (1 punt) Un arquitecte disposa d'una imatge en pantalla en la que veu la paret d'un edifici que li ocupa des del píxel (150, 100) al píxel (450, 300). Sap que la paret està completament perpendicular a la direcció de visió, i vol saber les mesures en metres d'amplada i alçada de la paret.
Se sap que la vista (*viewport*) és de 600x400 píxels, i que s'ha usat una càmera axonomètrica definida amb uns paràmetres del *window* (en metros) de: *left*=-60, *right*=60, *bottom*=-60, *top*=60.
Pot deduir quines són les dimensions de la paret que està veient? Si la resposta és afirmativa, calculeu aquestes dimenions. Justifiqueu la resposta.

El mapejat del window al viewport és directe, per tant l'amplada passa de 120 metres del window a 600 píxels del viewport, mentre que l'alçada passa de 120 metres del window a 400 píxels del viewport.

Per a saber les mides de la paret, doncs, només ens cal fer una regla de tres entre aquestes dimensions, i sabent que en píxels la paret ocupa en el viewport 300 píxels d'amplada i 200 d'alçada, tindrem que:

Amplada paret = 300*120/600 = 60 metres
Alçada paret = 200*120/400 = 60 metres

Per tant les dimensions de la paret són 60 x 60 metres.