VIG. EXAMEN PARCIAL – Curs 2007/2008. Quadrimestre de Primavera – TEMPS: 2h

Cognoms: Nom:

1. (1.5 Punts) Volem crear, utilitzant Qt, una interfície, similar a la de la figura que modifiqui el color i el text del *label* cada cop que premem un dels botons.
   Per a implementar l'etiqueta s'ha fet servir una classe derivada de `QLabel` que disposa d'un slot públic `colorFons(const QColor&)`. A més, com sabeu, `QLabel` disposa del slot públic: `setText(const QString&)`.
   Indiqueu si necessiteu altres classes derivades d'aquesta o d'altres classes de Qt, i si és el cas, quins slots i signals implementen. Indiqueu també totes les connexions necessàries per a implementar el comportament descrit.

Podem adoptar dues estratègies diferents. Una consisteix en fer servir una classe derivada del QPushButton que tingui atributs (un color, un text per a emetre en un nou *slot* quan se'l prem) i dos nous *signals* released(const Qcolor&), i released(const Qstring&) que siguin emesos quan es prem el botó. Per a aconseguir-ho, cal afegir un nou *slot* (privat) que el constructor connecta al *signal* released(), i que fa la feina. Els nous *signals*, al seu torn, es poden connectar als *slots* colorFons(const QColor&) i setText(const QString&) de la classe derivada de Qlabel. Els colors de fons dels tres botons i els valors dels atributs (que es fan servir als nous *signals*) s'assignen directament al *designer*.

Si la classe derivada de QLabel que em proporcionen es diu, per exemple miEtiqueta, una alternativa més simple consisteix en derivar una nova classe d'aquesta, diem-ne etiquetaProblema1, que tingui tres nous *slots*: aVermell(), aVerd() i aBlau(). Aquests *slots* tenen al seu codi els corresponents colors i texts de l'etiqueta, i sols cal instanciar tres botons amb el designer, donar-los-hi els texts corresponents i els colors de fons també corresponents, i connectar els seus *signals* pressed() amb el *slot* corresponent d'una única etiquetaProblema1. Aquesta solució requereix menys programació però és més rígida: qualsevol canvi en els texts o els colors desitjats requereixen una nova programació.

2. (1 Punt) Es vol imprimir un dibuix de color RGB = (1,0.5,0.5), en un full blanc usant una impressora que utilitza tintes *Cyan, Magenta* i *Yellow*.
   a) Quines tintes s'han d'usar i en quina quantitat per a obtenir aquest dibuix?
   b) Si la impressora s'ha quedat sense tinta magenta, i imprimeix igualment, de quin color quedarà imprès el dibuix?
   Justitiqueu les respostes.

Per a aconseguir el color RGB=(1, 0.5, 0.5) mitjançant les tintes CMY, cal obtenir el mateix color en el model CMY: CMY = (1-R, 1-G, 1-B) = (0, 0.5, 0.5).

a) Les tintes a usar són el Magenta i el Yellow en una proporció de 0.5 (sobre 1) cadascuna.
b) Si la impressora s'ha quedat sense tinta Magenta, vol dir que no podrà usar la quantitat de Magenta requerida pel color, i per tant, ens quedarà un color en el model CMY=(0, 0, 0.5),que és un groc claret.

3. (1.5 Punts) Es disposa del model geomètric d'una butaca. La seva capsa contenidora està centrada a l'origen de coordenades. La direcció cap amunt de la butaca és la de l'eix Y positiu i el seu davant està orientat cap a la part negativa de l'eix X. Utilizant aquest model, es vol visualitzar la platea d'un teatre que consta de 20 files de 10 butaques cadascuna. Les files són paral·leles a l'eix Z positiu. La primera butaca de la primera fila, *butaca[1,1]*, estarà situada centrada en l'origen de coordenades (la seva capsa contenidora quedarà centrada a l'origen de coordenades). La separació entre els centres de dues butaques d'una mateixa fila serà de +5. La separació entre els centres de la *butaca[i,j]* i la *butaca [i+1,j]* de dues files consecutives serà de –10. El davant de la cadira estarà orientat cap a les X positives.
Escriviu el tros de codi necessari per a pintar les 200 butaques de la platea del teatre.
Suposeu que ja teniu inicialitzades les matrius PROJECTION i MODELVIEW i que disposeu d'una acció `pinta_butaca()` que recorre el model geomètric de la butaca i l'envia a pintar utilitzant les primitives d'OpenGL.

Com que el model geomètric de la butaca es diu que té el davant cap a la part negativa de l'eix X i un cop situada la primera butaca al teatre aquesta ha d'estar mirant (té el seu davant) cap a la part positiva de l'eix X, això vol dir que caldrà girar (rotar) el model de la butaca 180 graus respecte de l'eix Y abans de situar-la a la posició que li toqui.

Pel que fa al posicionament de les butaques, de l'explicació es dedueix que la fila 1 (primera fila) del teatre es troba sobre la part positiva de l'eix Z, i la resta de files, es van situant en línies paral·leles a aquest eix Z desplaçades cap a la part negativa de l'eix X. Així doncs, el codi que seria necessari per a pintar aquestes 200 butaques, suposant que les matrius PROJECTION i MODELVIEW estan inicialitzades correctament amb els paràmetres de la càmera, serà el següent:

```
glMatrixMode (GL_MODELVIEW);
for (int fila=0; fila<20; ++fila) {
  for (int col=0; col<10; ++col) {
    glPushMatrix ();
    glTranslatef (fila*(-10), 0, col*5);
    glRotatef (180, 0, 1, 0);
    pinta_butaca ();
    glPopMatrix();
  }
}
```

4. (1 Punt) Com ja sabeu, quan la finestra GLWidget pateix algun canvi de dimensions, s'han d'actualitzar alguns paràmetres de la càmera. Quins? Per què? Indiqueu les instruccions d'OpenGL que cal utilitzar.

   Quan la finestra GLWidget pateix un canvi de dimensions, el que ha variat són les dimensions del *viewport* (de la vista). Per tant, suposant que no es vol tenir deformació, i que no es vol retallar la visió que es tenia de l'escena, caldrà modificar les dimensions del *window* de manera que tingui la mateixa relació d'aspecte que el *viewport* tot incluint el *window* inicial.

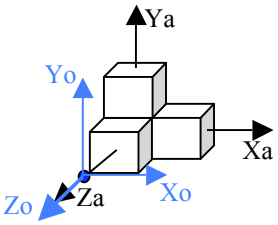
   Si tenim una càmera perspectiva, haurem de canviar la seva relació d'aspecte i, si s'escau, l'angle d'obertura de la càmera. En aquest cas la crida d'OpenGL que cal fer servir és la *gluPerspective*.

   Si tenim una càmera axonomètrica, variar el *window* vol dir variar els paràmetres: *left, right, bottom i top* de la càmera. En aquest cas la crida d'OpenGL que cal fer servir és la *glOrtho*.

   També cal indicar-li a OpenGL que el *viewport* (la vista) ha canviat, i per això s'usa la crida *glViewport*.

5. (1.5 Punts) Tenim una escena formada per tres cubs amb costats de mida 10, centrats en els punts C1=(0,0,10), C2=(0, 10, 0) i C3=(10, 0, 0). Quan es pinta l'escena en filferros volem veure en la vista tres quadrats situats en forma de L, i que el quadrat de la cantonada de la L quedi centrat a la vista. Indiqueu i justifiqueu la inicialització de tots els paràmetres d'una càmera que permeti obtenir la imatge descrita. Els paràmetres de posicionament cal indicar-los:

a. per a inicialitzar la càmera amb gluLookAt()
b. per a definir la MODELVIEW amb transformacions geomètriques.



- *Per a que els 3 cubs es projectin en quadrats, no ha d'haver-hi deformació perspectiva, per tant, la càmera ha de ser de tipus axonomètric i, a més a més, la ra del window ha de ser la del viewport.*
- *Per a que la cantonada de la "L" quedi al mig de la vista, la direcció de visió ha de passar per l'origen de coordenades i el window també ha d'estar centrat en ell.*
- *Hi ha diferents possilibitats de posicionament de la càmera que compleixen els requeriments, escollim la mostrada en la figura.*

Posicionament:
OBS=(0,0,20); VRP=(0,0,0); up = (0,1,0); per tant, gluLookAt (0,0,20,0,0,0,0,1,0)
Per a definir la MODELVIEW amb transformacions geomètriques, només cal fer un glTranslatef (0,0,-20).

Tipus de càmera:
zN=15, zF=25
Per a que es vegi l'escena, necessitarem un window=(-15, -15, 15, 15) i, per tant, raw=1; però volem una raw igual a rav, per tant,
Si raw<rav llavors
aw=30; hw=30; hw_nova=aw/rav; inc_h=hw_nova-hw; window=(-15,-15-inc_h/2,15,15+inc_h/2)
altrament
*aw=30; hw=30; aw_nova=hw*rav; inc_a=aw_nova-aw; window=(-15-inc_a/2,-15,15+inc_a/2,15)*
fisi

6. (1.5 Punts) Imaginem que disposem de dues variables panX i panY inicialitzades en funció dels desplaçaments realitzats per l'usuari per indicar el PAN. Un estudiant, quan ha de fer un PAN, únicament modifica aquestes variables i utilitza la següent funció `setModelView()` per a actualitzar la càmera (tots els signes són correctes)

```
void GLWidget::setModelview()
{
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glTranslatef(panX,panY,-dist);
  glRotated(-angleZ,0,0,1);
  glRotated( angleX,1,0,0);
  glRotated(-angleY,0,1,0);
  glTranslatef(-VRP.x,-VRP.y,-VRP.z);
}
```

El professor indica a l'estudiant que, tot i que l'efecte inicial d'aquest codi és el d'un PAN, en modificar l'orientació de la càmera observarà un comportament diferent a quan no feia PAN (o sigui quan panX=panY=0). Justifiqueu les dues afirmacions anteriors i expliqueu el canvi que s'observarà.

La primera afirmació és certa degut a que el `glTranslatef(panX,panY,-dist)` realitza una translació als objectes en la direcció dels eixos X, Y de l'observador; per tant, equival a moure la càmera segons aquests eixos que és el que produeix el PAN. Però, noteu que no hem modificat el VRP i, per tant, aquest ara NO es projectarà al centre del viewport.

Una modificació de l'orientació de la càmera, comporta modificar els angles d'orientació respecte uns eixos coordenats que passen pel VRP. L'usuari observarà que el canvi d'orientació s'efectua respecte uns eixos que no passen pel centre de la pantalla que és el que succeiria si s'hagués efectuat el PAN actualitzant el VRP (aquest es projectaria al centre de la pantalla).

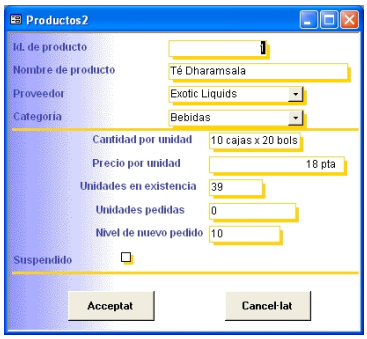
7.(1 Punt) Un estudiant proposa el següent codi per a inicialitzar la càmera utilitzant OpenGL i pintar un cub d'aresta 20 centrat a l'origen. Creieu que amb la seqüència d'instruccions indicada es pot obtenir la imatge requerida? Justifiqueu la resposta.

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
gluPerspective (65, 1, 20, 40);
gluLookAt(0,0,30,0,0,0,0,1,0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glPushMatrix();
glScalef(2,2,2);
PintaCub (0,0,0,10) // pinta un cub centrat a l'origen d'aresta 10
glPopMatrix();
```

Sí que és correcte.
Els paràmetres de la crida gluLookAt defineixen una posició i orientació de la càmera que permeten veure el cub indicat. També els paràmetres de la càmera perspectiva (gluPerspective) són correctes.

La matriu de TC, que calcula la gluLookAt(), és defineix (de manera no habitual) quan es té activa la pila GL_PROJECTION i en el seu "top" hi ha la matriu de projecció TP; per tant, la matriu que quedarà al "top" d'aquesta pila serà: TP=TP*TC.
En el top de la pila GL_MODELVIEW just abans d'enviar a pintar el cub hi haurà la matriu que escala uniformement els punts per 2.
Per tant, quan `PintaCub` enviï a pintar els vèrtexs del cub, aquests es veuran afectats per les matrius de ModelView (l'escalat) i Projection. Aquesta darrera matriu concatena la matriu de projecció amb la de càmera i, per tant, els punts es veuran afectats per les dues transformacions i en l'ordre correcte.
Per tant, malgrat les crides no es troben en l'ordre recomanat, aquest tros de codi pinta correctament el cub en filferros.

8. (1 Punt) Digueu quatre aspectes del disseny d'aquesta interfície que es podrien millorar.



1. Els botons han de tenir comandes enunciades com a verbs.
2. Les separacions s'han de fer amb espais, millor que amb línies.
3. Les etiquetes de dalt haurien d'estar més aprop dels quadres de text.
4. Les etiquetes, quan tenen més d'una paraula haurien d'estar totes aliniades a la dreta, al costat dels quadres de text a qué es refereixen.
5. Els quadres de text que contenen xifres haurien d'estar aliniades a la dreta (o al decimal).
6. Els quadres de text que tenen quantitats de la mateixa magnitud haurien de ser de la mateixa mida (unidades...).
7. S'han de minimitzar les línies verticals, totes les etiquetes i tots els quadres de text haurien d'estar aliniats igual, sempre que pugui ser, i aquí pot ser-ho.
8. La textura del fons dificulta la lectura del text

...