EXAMEN PARCIAL de VIG -- 16 de maig del 2007 -- Temps: 1.30 h

1. (1 Punt) Tenim una finestra de 1024x1024 píxels configurada de forma que cada component de color R, G, B es representa amb un enter entre 0 i 255. Indiqueu la quantitat de memòria necessària pel framebuffer, si fem servir doble-buffering. Justifiqueu la resposta.

   *Tenim:*
   *- 2 buffers de color (front buffer i back buffer)*
   *- 1024*1024 pixels a cada buffer de color*
   *- 3 components de color per pixel (per R, G, B)*
   *- 1 byte per component (1 byte permet codificar un enter entre 0 i 255)*
   *Per tant, la memòria mínima = 2*1024*1024*3 bytes = 6Mbytes*

2. (1.5 Punts) Suposem que tenim el model d'un personatge que, en el seu sistema de coordenades local, té el davant en la direcció positiva de l'eix Z i la seva capsa contenidora té dimensions x=50, y=100 i z=65 i està centrada en el punt C=(0,50,50). Quines transformacions de model caldrà aplicar a aquest personatge per a situar-lo, convenientment escalat, en el nostre laberint en la casella [3,5]? Cada casella del laberint té una dimensió de 1x1, les caselles es numeren començant per la [0,0] que té de coordenades mínimes el punt (0,0,0), i el terra està a 0.25 d'alçada i és paral·lel al pla X-Z del sistema de coordenades de l'aplicació.

   *Una possible solució:*
   - *Escalar el personatge per a que (per exemple) la base de la capsa tingui com a costat de longitud màxima la unitat. A tal efecte, cal:*
     - *Traslladar(per exemple) el centre de la base de la capsa a l'origen de coordenades: T(0,0,-50)*
     - *Realitzar un escalat uniforme de valor (1/65,1/65,1/65).*
   - *Moure el centre de la base al centre de la casella [3,5] i a l'alçada del terra: T (3.5,0.25,5.5)*

   *El OpenGL:*

```
glTranslate (3.5,0.25,5.5);
glScale (1/65,1/65,1/65);
glTranslate (0,0,-50);
```

3. (1.5 Punts) Tenim la següent classe que implementa un llum de semàfor:

```
class QButtonSemafor: public QPushButton
{
   Q_OBJECT
public:
   QButtonSemafor(QWidget *parent, const char *name=0);
   enum Color {Vermell=0, Groc=1, Verd=2;}
   Color getColor() const {return mColor;}
   void  setColor(Color _color) {mColor = _color; repaint();}
protected:
   void drawButtonLabel(QPainter *);
private:
   Color mColor;
};
```

   Volem que el comportament de la classe `QButtonSemafor` sigui el següent: Quan se'l premi, ha de canviar de color (verd a groc, groc a vermell i vermell a verd). La funció `setColor` ja s'encarrega de repintar-lo utilitzant la funció `drawButtonLabel` que utilitza el color actual (mColor) de forma convenient.

Completeu la classe amb els signals, slots i/o connexions necessaris per a definir aquest comportament. Implementeu les funcions que considereu oportunes i indiqueu els canvis (poseu el codi) que s'han de produir a les ja existents si és necessari.

- *Cal afegir un slot a QButtonSemafor per gestionar el canvi de color:*

```
void QButtonSemafor::changeColor()
{
  switch(getColor())
  {
   case Verd:    setColor(Groc); break;
   case Groc:    setColor(Vermell); break;
   case Vermell: setColor(Verd); break;
  }
}
```

- *Cal connectar el signal clicked() amb aquest slot. Això ho farem al constructor:*

```
QButtonSemafor::QButtonSemafor(QWidget *parent, const char* name)
 : QPushButton(parent, name), mColor(Vermell)
{
  connect(this, SIGNAL(clicked()), this, SLOT(changeColor()));
}
```

4. (1 Punt) Diposem d'una càmera axonomètrica amb els següents paràmetres:OBS=(0,0,0),VRP=(0,0,-1), VUV=(0,1,0), Window de (-5, -5) a (5, 5), Zn =5, Zf =10. Indiqueu un altre conjunt de paràmetres que defineixin exactament el mateix volum de visió, però donant a OBS, VRP, VUV, Zn i Zf valors diferents. Justifiqueu la resposta.

*Una possible solució és:*

*OBS = (0,0,1) (l'hem mogut una unitat cap a les Z+)*
*VRP = (0,0,-2) (qualsevol (0,0,z) amb z<Obsz fara que el vector de visió coincideixi amb l'eix Z negatiu)*
*VUV = (0,2,0) (qualsevol vector que projectat sobre el pla XY sigui de la forma (0, y, 0) amb y>0 produirà el mateix YOBS.).*
*Window = (-5, -5) - (5, 5)*
*Zn = 6 (ajustem Zn i Zf per contrarrestar l'efecte de moure l'observador una unitat)*
*Zf =11*

5. (1 Punt) Indiqueu les tres inicialitzacions independents que requereix una càmera per aconseguir les vistes ortogonals (planta, alçada i perfil) d'una escena tal que la seva esfera contenidora està centrada a l'origen i té radi 20. Concretament, cal que indiqueu els paràmetres de les transformacions geomètriques que s'aplicaran als objectes (amb OpenGL) per a veure l'escena en planta, alçat i perfil. Nota: al començar cada inicialització tenim com a ModelView la matriu identitat. No cal que inicialitzeu la `gluPerspective()`.

*Visió en alçat:*

```
glTranslate(0,0,-40);  // només cal desplaçar la càmera en l'eix z
```

*Visió en planta:*

```
glTranslate(0,0,-40);
glRotate(90,1,0,0);   // zobs coincident amb yAplicació
```

*Visió en pèrfil:*

```
glTranslate(0,0,-40);
glRotate(-90,0,1,0); // zobs coincident amb xAplicació
```

6. (1 Punt) Tenim un rectangle centrat a l'origen de coordenades amb els costats paral·lels als eixos X i Y de l'aplicació i de mides 4cm d'ample i 3cm d'alçada. Tenim una càmera inicialitzada amb Obs=(0,0,5), VRP=(0,0,0), VUV=(0,1,0). El viewport és de 400 píxels d'amplada i 600 d'alçada. Indiqueu els paràmetres amb què cal inicialitzar `gluPerspective()` de manera que encara que fem girar el rectangle respecte l'eix Z de l'aplicació sempre es visualitzi sense deformacions en el viewport i sense ser retallat. Nota: la càmera és fixa, és a dir, encara que girem el rectangle NO modifiquem la càmera.

   - *Cal que no hi hagi deformació, per tant, la relació d'aspecte serà la de la vista ra=400/600=2/3*

   - *Cal que no és retalli al girar. Per tant l'alçada (i amplada) del window que haurien de ser, com a mínim, de la mida de la diagonal del rectangle: sqrt(16+9)= 5. Per tant, donat que la relació d'aspecte és menor que la unitat, és requereix un angle d'obertura que garateixi una amplada de window de 5. Per tant, l'alçada del window serà:*

     *h= 5 /ra = 5*3/ 2 = 7.5*

     *i l'angle d'obertura serà:*

     *alfa= arctg(h/2*D) on D=distancia de l'observador al plà del rectangle=5*

     *FOV= 2*alfa*

   - *Znear pot ser qualsevol valor ]0,5] i Zfar ha de ser superior a 5, per a no retallar el rectangle.*

7. (1 Punt) En una aplicació, definim la matriu ModelView com:

$$M=\begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

   i fem que la matriu de projecció sigui la identitat. Podem saber les coordenades en el Sistema de Coordenades (SC) d'observador, SC normalitzades i SC de dispositiu del punt que té coordenades (-1,0,0) en el SC de l'aplicació? Justifiqueu la resposta i, en cas que sigui afirmativa, doneu el valor de les coordenades.

   *- Les coordenades del punt en SCO són (0,0,-1)*
   *- Les coordenades del punt en SCN són (0,0,-1) perque la matriu de Projecció és la identitat*
   *- No podem saber les coordenades en SC de dispositiu, perque no tenim informació sobre la vista (viewport)*

8. (2 Punts) Tenim una escena amb moltes esferes, totes elles de radi R. La seva ubicacio és arbitrària, però sabem que la distància mínima entre centres de qualsevol parella d'esferes és no inferior a 4*R. Cada esfera guarda informació de la posició del seu centre i dels tres vectors unitaris vx,vy,vz que defineixen un sistema local de coordenades (diferent per cada una d'elles). La superfície de l'esfera conté un text en relleu, que es llegeix bé quan es projecta en la direcció Z del seu sistema local; en aquest cas, el text segueix la direcció de l'eix X d'aquest sistema de coordenades de l'esfera. Especifiqueu els paràmetres d'una càmera que visualitzi una de les esferes de manera que es llegeixi bé el seu text. Heu d'evitar que el text sigui tapat totalment o parcialment per altres esferes de l'escena. La vista (viewport) té una mida de 600 píxels d'ample i 400 d'alçada.

*La càmera pot ser axonomètrica*
*Si el centre de l'esfera és C=(cx,cy,cz) i el seu sistema de coordenades queda definit pels tres vectors ortonormals Vx, Vy, Vz,*

- *VRP = C*
- *OBS = C + 2*R*Vz (per exemple)*
- *VUV = Vy (view up vector)*
- *Znear = 0.8*R (per exemple)*
- *Zfar = 3.2*R (id)*
- *relació d'aspecte: la de la vista*
- *Coodenades xmin,xmax,ymin,ymax de la finestra (en SCO): tals que el minim (xmax-xmin, ymax-ymin) sigui una mica mes gran que 2*R, i que respectin la relació d'aspecte de la vista.*