VIG- EXAMEN FINAL- Quadrimestre de Primavera- Curs 2007-2008 **TEMPS: 2h**

==========

COGNOMS: NOM:

==========

1) (1 punt) Un tècnic, veient que en una impressora de color s'ha acabat la tinta magenta, i veient que no en té, la substitueix per tinta vermella, pensant que s'hi assembla. Però naturalment els colors surten canviats. De quina manera canviaran els colors resultants? Per què? Hi ha colors que ara no podrem imprimir? Si és així, quins?

La tinta de color vermell s'obté amb magenta més groc. Quan pintem un color (c.m.y) i utilitzem tinta vermella en comptes de magenta és com si pintéssim amb magenta més groc (c,m,m+y) i, per tant, obtindrem colors més groguencs del que esperaríem.

Els colors (r,g,b) que és veuran diferents un cop impressos seran els que requereixen magenta; és a dir, aquells que $g \neq 1$. Els colors que no es podran imprimir (que mai podrem obtenir) són aquells que requereixen una quantitat $0<m<y$. Noteu que hi haurà una gamma menor de colors blavosos.

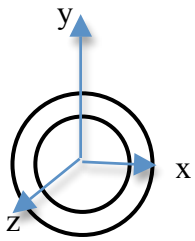
2) (1 punt) Tal i com us vàrem indicar en la solució de l'examen parcial, el codi següent pinta correctament en filferros una escena formada per un cub d'aresta 20 centrat a l'origen. Tanmateix, quan afegim il·luminació i un únic "focus de càmera" (GL_LIGHT1) no obtenim l'efecte desitjat. Per què? De quin color veurà l'observador les cares del cub? Justifiqueu la resposta.
Observació: anomenem "focus de càmera" a aquell que sempre es troba a la mateixa ubicació relativa a la càmera. Podeu considerar que el focus és de llum blanca i el cub és d'un material mat amb constant de reflexió difusa (1,0,0).

```
float pos[] = { 0., 0., 0., 1.}; //instrucció nova
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
gluPerspective (65, 1, 20, 40);
gluLookAt(0,0,30,0,0,0,0,1,0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glLightfv(GL_LIGHT1, GL_POSITION, pos) //instrucció nova
glPushMatrix();
glScalef(2,2,2);
PintaCub (0,0,0,10) // pinta un cub centrat a l'origen d'aresta 10
glPopMatrix();
```

En OpenGL es calcula el color en els vèrtexs en coordenades de l'observador. Quan definim la posició del focus, la matriu de MODELVIEW és la identitat; per tant la posició del focus en el SCO serà també (0,0,0). Quan enviem a pintar el cub, la matriu de MODELVIEW codifica un escalat uniforme de valor 2; per tant, els vèrtexs del cub en el SCO tindran com a coordenades les que li correspodrien al cub que volem visualitzar en el SCA. En aquesta situació, quan es calcula el color, el focus es troba just al centre del cub (origen de coordenades) i no en la posició de l'observador com esperaríem. Les cares del cub quedaran il·luminades per la banda interior; per tant, l'angle fita entre la normal als vèrtexs i la direcció d'incidència de la llum en ells serà fita$>|90^{o}|$ i, per tant, només es tindrà en compte per al càlcul del color la llum ambient i la constant de reflexió ambient del material. El cub es veurà com una taca de color vermell fosc (suposant que la llum ambient del focus és blanca i que la constant de reflexió ambient és com la difusa), o negre si no hi ha llum ambient.

3) (1.5 punt) Tenim el següent pseudo codi OpenGL per pintar un tor que té una capsa contenidora de longitud de costat màxim igual a 1, centrat a l'origen i que està orientat com s'indica esquemàticament en la figura.

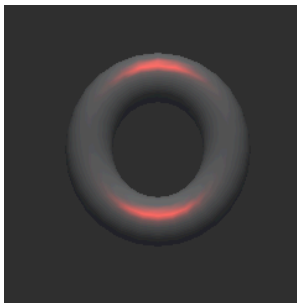
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glLightfv(GL_LIGHT0, GL_POSITION,{0.,2.,0.,1.});
glLightfv(GL_LIGHT0, GL_AMBIENT, {0.25,0.25,0.25,1.});
glLightfv(GL_LIGHT0, GL_DIFFUSE, {0.5,0.5,0.5,1.});
glLightfv(GL_LIGHT0, GL_SPECULAR,{0.5,0.5,0.5,1.});

gluLookAt(0.,0.,2.,0.,0.,0.,0.,1.,0.); //Obs, VRP, up
glMaterialfv(GL_FRONT, GL_AMBIENT, {0.25, 0.25, 0.25, 1.});
glMaterialfv(GL_FRONT, GL_DIFFUSE, {0.5,0.5, 0.5, 1.});
glMaterialfv(GL_FRONT, GL_SPECULAR, {1., 0., 0., 1.});
glMaterialfv(GL_FRONT, GL_SHININESS, 60);
drawModel();
```



Indiqueu de quin(s) color(s) és veurà el tor i com quedaran distribuït(s) en ell. Justifiqueu la resposta.
Observació: podeu considerar que el nombre de triangles del torus és molt elevat i que es realitza *smooth shading* i suavitzat d'arestes.

És veurà com es veu en la figura adjunta. Noteu que és un "focus de càmera", i, per tant, donada la càmera que tenim definida, està situat en la posició (0,2,2) de l'escena.
Hi podrem distinguir:

a) La zona no il·luminada pel focus, que serà aquella en què la normal en els vèrtexs forma un angle fita>|90º| amb la direció d'incidència de la llum en ells (cares "d'esquena a la llum"). Aquesta zona correspon a les cares del tor que la seva normal té una direcció propera a la de l'eix -y del SCA, i és veurà d'un color gris fosc constant (0.0625,0.0625,0.0625), pràcticament negre.
b) La zona il·luminada pel focus, que serà aquella en què les cares del tor estan "de cara" a la llum, angle fita<|90º|. Aquesta zona es veurà, a més del color ambient, amb una gradació de gris (0.25,0.25,0.25)*cos(fita) com aportació de la reflexió difusa, gris que serà més clar quan fita=0º.
c) Dins de la zona il·luminada, observarem dues taques especulars vermelles que es corrponen a les zones en que l'observador capta la reflexió especular de la llum en el tor. Es trobarà en la zona en què l'angle entre la direcció de reflexió especular i la direcció de visió en el vèrtexs sigui petit (perque el shininess és moderat); degut a què la reflexió especular es calcula com: $(0.5,0.5,0.5)(1,0,0)\cos^{60}(fi)= (0.5,0,0)\cos^{60}(fi)$, que és vermell.



4) (1.5 punt) Disposem d'una càmera definida amb OBS=(10.0,0,0), VRP=(0,0,0), up=(0,1,0), ZN=3, ZF=6, FOV=60, ra=1 i un viewport de (1024,1024) píxels, quines serien les coordenades d'ull (observador), normalitzades i de dispositiu (només les coordenades xd, yd) del punt (7,0,0)? I quines serien les del VRP?

Amb la definició de la càmera donada, tenim que el Sistema de Coordenades d'Observador (SCO) té el seu origen al punt (10, 0, 0) de l'aplicació, amb l'eix XObs en la direcció negativa de l'eix Z de l'aplicació, l'eix YObs en la direcció de l'eix Y de l'aplicació i l'eix ZObs en la direcció de l'eix X de l'aplicació.

El punt P = (7, 0, 0) en el SCO té com a coordenades Po = (0, 0, -3). Com que Po està sobre el pla del ZNear, i en l'eix -ZObs, les seves coordenades en el SCN seran Pn = (0, 0, -1), considerant que ZNear = -1 en SCN

(o bé Pn = (0, 0, 1) si considerem ZNear = 1). I com que està sobre ZObs i dins del volum de visió, el punt es projectarà sobre el centre del viewport, així que les seves coordenades de dispositiu seran Pd = (512, 512).

El VRP = (0, 0, 0) en el SCO té com a coordenades VRPo = (0, 0, -10). Aquest punt cau més lluny de l'observador que el ZFar, i en l'eix -ZObs, per tant, les seves coordenades en SCN seran VRPn = (0, 0, zn) on zn > 1 (més enllà del ZFar). Però com que el punt no es troba dins del volum de visió, aquest no passarà el procés de retallat i, per tant, no serà projectat en el viewport, així que no es calcularan les seves coordenades de dispositiu.

5) (1 punt) Escriu el tros de codi OpenGL requerit per a definir la càmera de l'exercici anterior (indicant el valor de tots els seus paràmetres). Per a la definició de la posició i orientació de la càmera utlitzeu transformacions geomètriques (i no la `gluLookAt()`).

Les transformacions geomètriques necessàries per a passar del SCA al SCO són primer una rotació Ry(90) i després una translació T(0, 0, 10), per tant, la TG a fer és TG = T(0,0,-10)*Ry(-90) i el codi OpenGL necessari per a definir aquesta càmera serà el següent:

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
gluPerspective (60, 1, 3, 6);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glTranslatef (0, 0, -10);
glRotatef (-90, 0, 1, 0);
```

6) (1.5 punt) Si recordeu, a la pràctica 2, la selecció dels fanals a encendre/apagar es fa mitjançant el clic del ratolí. Això implica haver de canviar el *frustum* per tal de concentrar la selecció a una petita àrea al voltant del píxel on es prem. Quina matriu es veurà afectada per aquesta modificació? Quina instrucció d´OpenGL ens permet implementar aquesta modificació? Escriu un tros de codi OpenGL que mostri les crides requerides per a obtenir la funcionalitat indicada.

La matriu afectada per la modificació del frustum per a la selecció és la matriu de projecció TP o GL_PROJECTION

La crida d'OpenGL que ens facilita la implementació d'aquesta modificació és la crida:

`gluPickMatrix(xclick,yclick,ample,alt,vp)`

on (xclick, yclick) defineixen el pixel on s'ha clicat amb el ratolí (considerant un viewport amb l'origen a la cantonada esquerra abaix), *ample* i *alt* defineixen les dimensions, en píxels, de la finestra al voltant del píxel que es vol considerar en la selecció i *vp* és el *viewport* . Un possible codi correcte seria:

```
GLfloat proj[16];
GLint vp[4];
glGetFloatv (GL_PROJECTION_MATRIX, proj);
glGetIntegerv (GL_VIEWPORT, vp);
glMatrixMode (GL_PROJECTION);
glPushMatrix ();
glLoadIdentity ();
gluPickMatrix (xclick, yclick, 3, 3, vp);
glMultMatrix (proj);
```

7) (1 punt) Tenim una escena amb dos cubs amb arestes alineades amb els eixos. Un d'ells està centrat a l'origen i és de costat 2, l'altre està centrat al punt C=(4,0,0) i és de costat 4. La posició i orientació de la càmera es defineix com: OBS=(10,0,0), VRP=(0,0,0) i up=(0,1,0). Tenim una càmera ortogonal amb *window*: left=-2, right=2, bottom=-2, top=2 i ZNear=6 i ZFar=12 i un *viewport* de 512 x 512. Sabem que el cub petit és de color (1,0,0) i el gran és de color (0,1,0), i que usem z-buffer, *backface culling* i mode *"fill"* per a la visualització de les cares. Indiqueu què es veurà en el *viewport* i de quin color. Justifiqueu la resposta.

Donada la posició i orientació de la càmera i la definició del window en la càmera axonomètrica, tindriem que el cub gran es pintaria en la totalitat del *window*/*viewport*, però com que el pla de retallat definit pel ZNear queda just a la meitat d'aquest cub, resulta que només la meitat del darrera del cub queda dins del volum de visió. Tenint en compte que es fa *backface culling*, que vol dir que les cares que queden d'esquena a l'observador no s'envien a pintar, resulta que del cub gran no s'envia a pintar cap de les seves cares. Com que el cub petit sí que queda completament dins del volum de visió, aquest serà l'únic que es veurà pintat en el *window/viewport*. Així doncs, el que es veurà en el *viewport* és un quadrat vermell de la meitat de la mida del viewport i centrat en ell, és a dir un quadrat que anirà del pixel (128, 128) al pixel (348, 348).

8) (1.5 punt) En una aplicació trobem sovint ocasions en les quals una tasca demanada per l'usuari triga un temps relativament llarg. Com a dissenyadors de la interfície, hem de fer quelcom especial? Per què? Quines opcions tenim i quins criteris fonamentals hem de seguir per a escollir l'opció més vàlida? Justifiqueu la resposta.

Quan una tasca triga més d'un segon, aproximadament, hem de proporcionar realimentació visual a l'usuari perquè sàpiga que el sistema està fent el que se li demana i que no ha perdut el control. En aquest sentit, a més, cal proporcionar, sempre que es pugui, una aproximació al temps que queda per a què la tasca s'hagi completat i s'ha d'evitar bloquejar l'usuari per a què pugui fer altres coses.

Hi ha tres formes d'indicar a l'usuari que una tasca s'està completant: modificació del punter, animació interna i animació externa. La **modificació del punter** serveix per a indicar que l'usuari està bloquejat però s'ha d'acompanyar d'algun tipus d'animació si el temps que resta es pot calcular. L'**animació interna** és l'activació de la barra de progrés que porten algunes aplicacions. L'**animació externa** és una finestra que indica el progrés. A diferència de l'anterior, pot proporcionar més informació i permetre l'existència de botons de cancel·lació.

Si l'aplicació bloqueja l'usuari, utilitzarem la modificació del punter i si podem calcular el temps que queda, l'acompanyarem d'animació interna o externa. Utilitzarem una barra de progrés en cas que la nostra aplicació la tingui; altrament una finestra de progrés, tot i que l'obertura d'aquesta pot arribar a distreure. Si el procés és molt llarg i ens interessa poder cancel·lar-lo, cal tenir una finestra de progrés amb botons per a cancel·lació.