VIG - Examen Final - 15-1-2009 - TEMPS 2h. 1Punt per pregunta

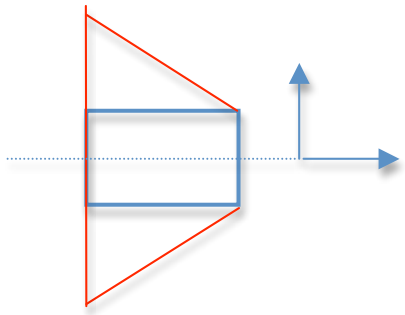
==========

COGNOMS: NOM:

==========

1. S'està visualitzant una escena utilitzant una càmera axonomètrica definida amb ***Obs***, ***VRP***, ***up***, *window*, *znear* i *zfar* i un cert *viewport*. Indiqueu tots els paràmetres d'una càmera perspectiva que permeti veure, com a mínim, el mateix volum de visió que amb la càmera axonomètrica. Justifiqueu la resposta.

   VRP, ***OBS*** *i* ***up*** *serveixen per definir la posició i orientació del SCO que és independent del tipus de càmera; per tant, tindran els mateixos valors.*
   zN i ZF defineixen la distància de l'observador al pla anterior i posterior del volum de visió. Com l'observador està en la mateixa posició, no es modifiquen.
   L'angle obertura de la càmera FOV s'ha d'inicialitzar de manera que la piràmide de visió inclogui el window de la càmera axonomètrica (veure figura). D'acord amb la resta de paràmetres ja fixats:
   $FOV = 2\alpha_v = 2\ arctg\ ((window.up-window.bottom)/2*zN)$.
   (cal recordar que el window està situat en el pla anterior de retallat, és a dir, a una distància zN de l'observador)
   La relació d'aspecte de la càmera perspectiva ha de ser la mateixa que la del window de l'axonomètrica per a garantir que aquest es veu tot: ra= ((window.left- window.right)/window.up-window.bottom)).
   Dit d'altra manera, el window d'ambdues càmeres ha de ser el mateix.



2. Volem fer una aplicació informàtica per veure i analitzar parts de models d'estàtues clàssiques. En concret, volem poder visualitzar cada un dels dits de les mans. El model de cada estàtua és una malla de triangles, i per cada dit de cada mà tenim la llista de triangles que el formen i dos punts 3D: un a la base del dit (punt ***B***) i un altre a la punta (punt ***P***). També tenim el vector ***n*** promig dels vectors normals dels triangles de la seva ungla. Definiu tots els paràmetres d'una càmera axonomètrica que permeti veure un dit determinat de la estàtua, de manera que aquest sempre quedi horitzontal dins la vista i es vegi amb la normal promig de l'ungla mirant cap a la càmera. Justifiqueu la resposta.

   Si volem veure tot el dit i, aproximadament, centrat en el viewport, podem definir el ***VRP*** *al centre del segment* ***BP****; per tant,* ***VPR****= (****B****+****P****)/2*
   *La direcció de visió ha de permetre veure la part superior de l'ungla; per tant, l'eix z del SCO ha de tenir la direcció d'****n****; la distància de l'observador al* ***VRP*** *ha de ser superior al gruix del dit. Si ens serveix un càlcul aproximat i fem la hipòtesi que el dit és més llarg que ample, l'esfera englobant del dit tindrà com a radi*
   *R= dist(****B****,****P****)/2 i l'observador haurà d'estar fora d'aquesta esfera, per exemple, a una distància 2R del* ***VRP:***
   OBS *=* ***VRP****+ 2*R*****n***
   L'eix y del SCO ha de ser perpendicular a ***BP*** *i a zobs (****n****), per tant,* ***up****=****BP*** *x* ***n***

   El window ha de permetre veure tot el dit. Per tant, la seva amplada ha de ser, com a mínim, ***BP*** *i la seva alçada el gruix del dit. Si no sabem el gruix del dit, d'acord amb la hipòtesi anterior, podem fixar l'alçada igual que l'amplada. Per tant, win.up=R; win.bottom=-R; win.right=R; win.left=-R. Tanmateix, caldrà ajustar la seva relació d'aspecte d'acord a la del viewport si no volem deformacions.*
   En quant a zN i ZF, han de poder garantir veure tot el gruix del dit. D'acord amb la posició de l'observador, possibles valors són zN=R dist, zF= 3R

3. Tenim una càmera axonomètrica definida amb els paràmetres: ***OBS***=(50,0,0), ***VRP***= (0,0,0), ***up***=(0,1,1), window={-20, 20, -20, 20}, ZNear=1, ZFar=70 i pintem en una vista de 512x512.
   - Quines transformacions geomètriques caldria aplicar a un cub de costat 20, centrat a l'origen i amb costats orientats segons els eixos de coordenades per a què a la vista es vegi un quadrat amb costats paral·lels al viewport?
   - Escriu el tros de codi OpenGL requerit per a inicialitzar la càmera utilitzant transformacions geomètriques i per a pintar el cub. Suposeu que teniu un procediment *pinta_cub(L)* que genera les crides a OpenGL requerides per a pintar un cub d'aresta *L* centrat a l'origen i amb cares orientades segons els plans coordenats.

   El cub, que està centrat a l'origen i és de costat 20, ja cau dins del volum de visió, i amb una cara perpendicular a la direcció de visió, per tant només caldria girar el cub respecte a l'eix de la direcció de visió per a què quedi aliniat amb la finestra (window), perquè el vector ***up*** *ens indica que tenim l'eix Y de l'observador en la direcció (0,1,1) de l'aplicació. La rotació a fer al cub ha de ser de 45 graus respecte de l'eix de les X de l'aplicació:* G_x ***(45º)***

   El codi OpenGL per a situar la càmera i pintar correctament el cub serà el següent:

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
glOrtho (-20, 20, -20, 20, 1, 70);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
// Situem la càmera on ens indica i amb transformacions
glTranslatef (0, 0, -50);
glRotatef (-45, 0, 0, 1);
glRotatef (-90, 0, 1, 0);
// Fem la rotació necessària per a pintar el cub
glRotatef (45, 1, 0, 0);
// Pintem el cub
pinta_cub (20);
```

4. Estem visualitzant una escena amb bastants polígons. Realitzem una ampliació del viewport i observem que en modificar la càmera (girar-la), l'actualització de la imatge és molt més lenta. Justifiqueu aquest efecte.

   Amb una ampliació del viewport estem incrementant el nombre de píxels a processar, per tant el procés de rasterització i de z-buffer es veuen afectats. Com que l'ampliació del viewport (i no del window) provoca que el que es veia originalment en un viewport més petit és exactament el mateix que es veu ara en un viewport més gran, tots els polígons que cauen en el volum de visió ocupen ara més píxels i per tant es carrega molt més el procés de rasterització perquè s'han de processar molts més fragments. De la mateixa manera, el z-buffer també s'ha de calcular (actualitzar) per a més píxels.

5. Una escena amb dues esferes *A* i *B* (representades per un poliedre convex) es mira des d'una posició on l'esfera *B* queda davant de la *A* i la tapa parcialment. Usant per a l'eliminació de parts amagades l'algorisme de *culling*, ens trobem que la visualització obtinguda no és correcta perquè el tros de l'esfera A que ha de ser tapat per l'esfera *B* es veu en la imatge.
   - Perquè no està funcionant correctament l'eliminació de parts amagades?
   - Com ho solucionaries?

   L'algorisme de culling actua en espai objecte i elimina totes aquelles cares que queden "d'esquena" a l'observador. Però en el procés de comprovació de si la cara queda de cara o d'esquena no es veuen implicats altres objectes ni altres cares de l'objecte, per tant no té en compte si una cara inicialment visible perquè està de cara a l'observador està tapada per alguna altra cara d'un altre objecte (o del mateix).

   Es pot resoldre combinant el culling amb el z-buffer i per tant usant el culling només com a preprocés per a tractar menys cares del model, o bé mantenint un ordre de pintat de les esferes assegurant que sempre es pinta primer la que està més lluny de l'observador.

6. Tenim una esfera el material de la qual està definit com: Ka= (0.5,0.5,0), Kd=(0,1,1), Ks=(1,1,1). Suposant que està il·luminada per un únic focus de llum, indica els colors del focus de llum i de la llum ambient per a què a l'esfera es vegin els següents efectes: color verd fosc allà on no il·lumina el focus de llum i, en la zona il·luminada pel focus, una gradació de blaus i una taca clarament més lluminosa de color magenta (tot amb un fons verd fosc). Justifica la resposta.

   En les zones de l'esfera on no il·lumina el focus de llum, només tenim llum ambient i per tant només la component ambient del càlcul d'il·luminació hi tindrà efecte. Com que la Ka és de color groc, per a què es vegi verd fosc la llum ambient no ha de tenir component vermella, per a què només es reflecteixi la component verda del material. L llum ambient ha de ser baixa per a què es vegi un verd fosc: ***Llum ambient: (0, 0.5, 0)***

   A la zona il·luminada pel focus, el fons verd fosc ens ve donat per la component ambient del càlcul d'il·luminació, ja que aquesta component no és nul·la. La resta ens indica que la component difusa ha de resultar blava (0, 0, 1) i la component especular ha de resultar magenta (1, 0, 1). Com que la Kd és de color cyan i la Ks és blanca, això ens indica que el focus de llum ha de tenir components vermella i blava però no verda, per tant un possible color per al focus de llum és el magenta: ***Focus llum: (1, 0, 1).***

7. Una escena està formada per diverses esferes de diferents grandàries però totes de mateix material. Observem en cadascuna d'elles 3 taques especulars blanques i ubicades, pràcticament, en la mateixa posició relativa en cada esfera.
   - Creus que és posible? En cas afirmatiu, què podries dir de l'entorn d'il·luminació (nombre de focus, colors, ubicació,...) i de la posició de l'observador?
   - Les taques tenen forma poligonal, per què?

   Si es veuen tres taques especulars en totes les esferes, com a mínim ha d'haver-hi 3 focus de llum (un per taca). Si les taques són blanques, els focus són de llum blanca, ja que les taques especulars tenen el color del focus.

   Si les taques es produeixen en les mateixes posicions en totes les esferes, vol dir que es produeixen en punts que tenen normals semblants. Per a què l'observador observi la taca en aquests punts ha d'estar situat en la direcció de la llum reflectida especularment en ells. Aquesta direcció és funció de la direcció de la llum incident i de la normal en el punt de reflexió. Una possibilitat és que en la llum incident tingui, pràcticament, la mateixa direcció d'incidència en tots els punts en que s'observa la reflexió especular, és a dir, que els focus siguin direccionals (focus ubicats molt lluny de les esferes). Per a què la direcció de visió dels punts en què es veu la taca sigui la mateixa per totes les esferes, l'única possibilitat és que l'observador es trobi en una posició molt allunyada.

   Si les taques es veuen poligonals és que representem les esferes mitjançant un poliedre de relativament poques cares. Donat que el càlcul del color s'efectua en els vèrtexs, que la direcció de la reflexió especular es funció de la normal en el vèrtex i de la direcció de la llum, i que la seva visió és funció de la posició de l'observador, encara que fem suavitzat d'arestes (utilitzem una única normal per vèrtex i smooth shading) es pot observar la taca poligonal (si tenim poques cares) perquè el color especular dels vèrtexs d'un polígon pot ser molt diferent. Òbviament, si utilitzem normal per cara i/o flat shading, encara es notarà més l'efecte poligonal de les taques.

8. Indiqueu com poden afectar al resultat de la selecció d'objectes d'una escena qualsevol tenir activats/desactivats cadascun dels següents estats d'OpenGL: *culling*, il·luminació i pintat en filferros. Cal tractar cada estat indepentdent als altres.

   El culling no afecta ni a l'eficiència ni al resultat de la selecció (nombre de "hits") perquè, en OpenGL, s'efectua després del procés de transformació a coordenades de dispositiu, per tant, després d'efectuar-se la selecció (retallat).

   La il·luminació no afecta al resultat. Cal recordar que la selecció OpenGL detecta si una primitiva és interior o no al volum de selecció. Tanmateix, donat que el càlcul del color en els vèrtexs s'efectua prèviament al retallat i que en mode de selecció no es pinten les primitives, si tenim la il·luminació desactivada, s'incrementa l'eficiència de la selecció.

   Si pintem en filferrors (arestes), un objecte/cara serà seleccionada (produirà "hit") si alguna de les seves arestes talla o és interior al volum de selecció. Per a seleccionar un objecte/cara, l'usuari haurà de clicar a sobre –a prop- d'una de les seves arestes. Altrament no serà detectada la primitiva.

9. Tenim una escena formada per dos cubs i l'estem visualitzant amb un càmera perspectiva que està definida (matrius GL_MODELVIEW i GL_PROJECTION inicialitzades). La volem il·luminar per tres focus de llum: un ha d'estar situat al centre del segment que uneix el centre dels cubs (punts ***C1*** i ***C2***); altre sempre estarà ubicat 10 unitats per sobre de l'observador i el tercer el volem ubicar 5 unitats a la "dreta visual" del centre del segon cub (punt ***C2***). Un cop ubicat aquest tercer focus, ha de quedar en una posició fixa respecte de l'escena, encara que es modifiqui la càmera. Indiqueu el codi OpenGL per a ubicar els tres focus. Observació: entenem per "dreta visual" que el focus estigui desplaçat de la projecció de *C2* en el viewport sobre una recta que es veu paral·lela al costat horitzontal del viewport.

```
glMatrixMode(GL_MODELVIEW);
pos1=(C1+C2)/2;
glLigtfv(GL_LIGHT0, GL_POSITION, pos1); //llum fix respecte l'escena
GLfloat mv[4][4];
glGetFloatv (GL_MODELVIEW_MATRIX, &mv[0][0]);
Vector xobs={mv[0][0],mv[0][1],mv[0][2])};
float pos3[1]={C2.x+xobs.x*5,C2.y+xobs.y*5,C2.z+xobs.z*5,1};
glLigtfv(GL_LIGHT2, GL_POSITION, pos3);
glPushMatrix();
glLoadIdentity();
float pos2[4]={0,10,0,1};
glLigtfv(GL_LIGHT2, GL_POSITION, pos2);
glPopMatrix();
```

10. Suposem que hem implementat un widget per a modificar les constants Ka_λ, Kd_λ, Ks_λ del material d'un objecte d'una escena. Aquest widget està format bàsicament per botons que permeten seleccionar el color RGB de cada constant. Expliqueu, breument, quin seria el procediment necessari per a aconseguir que un canvi en qualsevol component del color d'una de les constants modifiqui el material de l'objecte de l'escena sense haver de tancar el widget.
    Indica el codi bàsic de Qt per a què aquest procediment que has descrit funcioni: declaració de signals, connexions,...

    El widget de materials haurà de tenir un signal que s'emeti cada cop que es modifica el color d'algun dels botons. Aquest signal tindrà com a paràmetre una variable que contingui totes les característiques del material que està modificant. Aquest signal serà capturat per un slot del widget de GL (el widget de l'aplicació) que prendrà aquest paràmetre i aplicarà el nou material a l'objecte.

A la classe del widget de materials cal afegir la declaració del signal:

```
signals:
        void materialModificat (Material *);
```

Als slots dels botons que permeten canviar les característiques de la component corresponent del material cal afegir l'emit del signal anterior :

```
emit materialModificat (m); // on m és la variable que conté el material
```

Al widget de GL caldrà afegir la declaració del slot i la corresponent connexió:

```
public slots:
        void aplicaMaterial (Material *);
```

Considerant que la variable que conté el widget de materials és wmat:

```
connect (&wmat, SIGNAL(materialModificat(Material *)),
            this, SLOT(aplicaMaterial(Material *)));
```