Examen Final Interacció i Disseny d'Interfícies Juny 2013 2 Hores

Pregunta 1 (1.5 punts) Es vol crear una escena formada per: un terra representat per un quadrat de 10x10 situat en y=0 i centrat en (0,0,0), i 2 arbres que tenen el centre de la base dels seus troncs en els punts (-2.5,0,0) i (2.5,0,0) respectivament. Es demana:

a. El codi de la funció *pintaArbre()* que pintaria un arbre amb un tronc d'alçada 3 representat per un cilindre de radi 0.25 amb centre de la base en (0,0,0), i una copa representada per una esfera amb centre (0,3.5,0) i radi 1. Utilitzeu les funcions glutSolidSphere(..) i glutSolidCilindre(..).

b. El codi que, utilitzant la funció pintaArbre(), permet enviar a pintar l'escena desitjada. Es suposa que la càmera està correctament inicialitzada.

Recordeu que:
void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
draws a sphere centered at the coordinates origin of the specified radius.
void glutSolidCylinder(GLdouble radius, GLdouble hight, GLint slices, GLint stacks);
draws a cylinder, the center of whose base is at the origen and whose axis is along the positive Z axis.

a) Utilitzant les funcions de glut, per a què es pintin el tronc i la copa de l'arbre, cal aplicar al cilindre i a l'esfera, respectivament, les següents transformacions:
- $TG_{TRONC} = G_x(-90º)$
- TG_{COPA} = Trans(0,3.5,0)

```
void pintaArbre(void)
{
glMatrixMode (GL_MODELVIEW);
glPushMatrix();
glTranslated (0.,3.5,0);
glutSolidSphere(1.,20,20);
glPopMatrix();
glPushMatrix();
glRotated (-90,1,0,0);
glutSolidCylinder(0.25,3.0,20,20);
glPopMatrix();
}
```

b) Per a crear l'escena, cal pintar el terra i dos arbres. Aquests darrers utilitzant la funció anterior tenint en compta que cal aplicar una tanslació als arbres.

```
void pintaEscena (void)
{
glMatrixMode (GL_MODELVIEW);
glPushMatrix();
glTranslated (-2.5,0,0);
pintaArbre();
glPopMatrix();
glPushMatrix();
glTranslated (+2.5,0,0);
pintaArbre();
glPopMatrix();
glBegin (GL_POLYGON);
 glVertex3f (-5,0,-5);
 glVertex3f (-5,0,5);
 glVertex3f (5,0,5);
 glVertex3f (5,0,-5);
glEnd();
}
```

Pregunta 2 (1.5 punts) Un gegant camina sobre el terra mirant l'escena. Inicialment té ubicada la càmera en la posició (0,3.5,5) mirant cap el punt (0,3.5,0) cap al qual avança, i amb un ***up*** inicial igual a (0,1,0). El gegant pot fer dues accions: girar o avançar; però sempre enfoca la càmera en la direcció d'avançament. Indiqueu com actualitzar els paràmetres de la càmera ***OBS, VRP*** i ***up*** en els següents casos:

a. Cada cop que el gegant gira "fita" graus cap a la seva esquerra per modificar la seva direcció d'avançament.
b. Cada cop que el gegant es mou "lambda" unitats en la seva direcció d'avançament.

a) En aquest primer cas, el gegant gira per a modificar la seva orientació d'avançament. Per tant, la càmera gira sobre un eix paral·lel a l'eix Y de l'aplicació passant per **OBS**, motiu pel que **OBS** no és modifica. **Up** tampoc es modifica ja que, únicament, fem un gir sobre l'eix Y. En canvi **VRP** és modificarà donat que modifiquem la direcció d'enfoc tot mantenint la distància de **VRP** a **OBS**, formalment, girem **VRP** sobre un eix que passa per **OBS** i té vector director (0,1,0).

Si **v** és el vector que indica la direcció d'avançament, alfa és l'angle acumulat que el gegant ha girat cap a la seva esquerra, i **OBS** i **VRP** són les posicions actuals de l'Observador i d'enfoc, i *d* és la distància entre ells, per a calcular la nova posició de VRP cal:

```
alfa= alfa+fita
//nova direcció: gir respecte Y de la direcció d'avançament inicial  (0,0,-1)
v= (-sin(alfa), 0, cos(alfa)))
// VRP estarà a distància d de OBS en la nova direcció d'avançament
VPR= OBS + d*v
```

b) En aquest cas el gegant simplement es desplaça en la direcció d'avançament sense modificar-la, per tant, up no es modificarà donat que fem una translació en un pla perpendicular a ell. **VRP** caldrà modificar-lo (avançar-lo) igual que **OBS** perquè altrament podriem arribar a modificar el sentit del vector d'avançament i, a més a més, perquè no es modifica la distància d'enfoc. Per tant,

```
// la direcció d'avançament es pot calcular com:
    v= VRP-OBS;
// cal vector unitari per a què lambda sigui la quantitat avançada
    v=normalitza(v)
    OBS= OBS + lambda* v
    VRP= VRP+ lambda* v
```

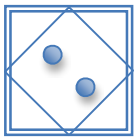
Pregunta 3 (1 punt) Indica quina de les inicialitzacions de l'òptica perspectiva és més apropiada per a una càmera que porta un observador que camina per una escena fent fotos (com en la pregunta 2). Observació: ra_v és la relació d'aspecte del *viewport*.

a. FOV=60º, ra=ra_v, zNear=0.1, zFar= 20
b. FOV=60º, ra=ra_v, zNear=R, zFar= 3R; essent R el radi de l'esfera contenidora de l'escena.
c. FOV=2*(arcsin(R/d)*180/PI); ra=ra_v, zNear=R, zFar= 3R; essent R el radi de l'esfera contenidora de l'escena i d la distància d'**OBS** a **VRP.**
d. FOV=2*(arcsin(R/d)*180/PI), ra=ra_v, zNear=0, zFar= 20; essent R el radi de l'esfera contenidora de l'escena i d la distància d'**OBS** a **VRP.**

Quan és vol simular un "walk" per una escena (com en la pregunta 2), l'observador vol veure tot el que té davant seu, per tant, el camp de visió ha de començar just en **OBS**; com zNear en una càmera perspectiva no pot ser zero, el correcte és zNear=0.1 per no retallar l'escena i poder veure tot el que està davant de la càmera. Aquest raonament, descarta 3 de les 4 respostes.

Pregunta 4 (1 punt) Si inicialitzem la MODELVIEW amb transformacions geomètriques, donat el codi adjunt, quina de les següents inicialitzacions dels angles permetria (si l'optica és correcta) obtenir, per l'escena de la pregunta 1, una visualització similar a l'esquematitzada en la figura?

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0,0,-d);
glRotatef(fi,0,0,1);
glRotatef(fita,1,0,0);
glRotatef(psi,0,1,0);
glTranslatef(-VRPx, -VRPy, -VRPz);
pintaEscena();
```



a. **VRP=(0,0,0), d=6, psi=+45, fita=+90, fi=-90**
b. VRP=(0,0,0), d=6, psi=0, fita=+90, fi=+45
c. VRP=(0,0,0), d=6, psi=+90, fita=0, fi= -45
d. VRP=(0,0,0), d=6, psi=+45, fita=90, fi=45

La transformació que s'aplica quan es pinta l'escena és:
T=Trans(0,0,-d)*G_z(fi)*G_x(fita)*Gy(psi)*Trans(-VRPx, -VRPy,-VRPz).
Recordeu que el terra estava en y=0 i el seu centre en (0,0,0).
Si provem les diferents opcions, i recordem que els girs positius són en sentit anti-horari, la resposta correcta és la a).

Pregunta 5 (1 punt) Es vol pintar una escena amb un focus situat a la posició (10,20,0) en coordenades de l'aplicació, indica en quina posició del codi següent posaries la instrucció `glLightfv(GL_LIGHT0, GL_POSITION, posfocus)` i amb quins valors s'ha d'inicialitzar la posició del focus (posfocus),

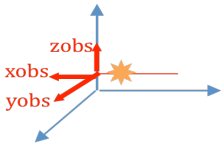
```
1. glMatrixMode(GL_PROJECTION);
2. glLoadIdentity();
3. gluPerspective(alfa, ra, znear, zfar);
4. glMatrixMode(GL_MODELVIEW);
5. glLoadIdentity();
6. gluLookAt (0,20,0,0,0,0,0,0,1)
7. drawScene();
```

a. **Entre les instruccions 5 i 6, i posfocus=(-10,0,0,1)**
b. Entre les instruccions 6 i 7, i posfocus=(-10,-20,0,1)
c. Després de la instrucció 7, i posfocus=(10,20,0,1)
d. Entre les instruccions 6 i 7, i posfocus=(-10,0,0,1)

- La quarta coordenada de la posició és per a tots els casos 1, per tant, es tracta d'un focus puntual del que indiquem la seva posició.
- La posició de la llum s'ha de declarar ABANS de pintar, per tant abans de la instrucció 7 (descartem la resposta *c*).
- Entre la les instruccions 6 i 7 ja tenim definida la càmera (gluLookAt), per tant, *posfocus* hauria de ser la seva posició del focus en l'escena (com un punt qualsevol), o sigui (10,20,0); això descarta les respostes *b* i *d*.

Podem comprobar que la *a* és correcte:

- Quan es realitza la crida a glLightfv(), la matriu de modelview (MV) és la identitat. OpenGl emmagatxema les coordenades del focus després de multiplicar-les per la MV; per tant, considerarà el focus a la posició (-10,0,0) respecte el sistema de coordenades de l'observador.
- En el moment de pintar la càmera està ubicada com indica gluLookAt(); per tant, el sistema de coordenades de l'observador té el seu origen en OBS=(0,20,0) i el seus eixos tenen les següents direccions: $zobs^+$ la direcció y^+ de l'aplicació, $yobs^+$ la direcció de z^+, i **$xobs^+$ la direcció de x^-**.



- Com el focus per OpenGl es troba a (-10,0,0) respecte el SCO, d'acord amb els comentaris anteriors, es trobarà a (10,20,0), com voliem.

Pregunta 6 (1 punt) Una escena està formada per dos cubs amb les cares paral·leles als plans de coordenades. El CUB1 té aresta 20, el centre de la seva base en (0,0,0) i és de color verd i mate; el CUB2 té aresta 20, centre de la seva base en (30,0,0) i és del mateix color verd però brillant. Il·luminem l'escena amb un focus groc situat en (50,10,0). L'obsevador es troba en una posició que pot veure les cares dels cubs ubicades en x=10 i x=40. Si es pinta l'escena amb OpenGL utilitzant model d'il·luminació de Phong i Smooth shading (Gouraud Shading), de quin color es veuran aquestes cares? No hi ha llum ambient.

a. **La cara en x=10 és veurà de color verd constant, la cara en x=40 també és veurà de color constant però d'un verd més fosc.**
b. La cara en x=10 és veurà de color verd constant, la cara en x=40 també és veurà de color constant però d'un verd més clar.
c. La cara en x=10 és veurà de color verd constant, la cara en x=40 també és veurà de color constant però d'un verd més clar i amb una taca especular groga en mig de la cara.
d. La cara en x=10 és veurà amb diferents tonalitats de verd, la cara en x=40 també és veurà amb diferents tonalitats de verd però més clars i amb una taca especular groga en mig de la cara.

- Donat que el model d'Il·luminació és Phong, es calcularà el color en els vèrtexs de les cares tenint en compte la reflexió difusa i l'especular.

  $I(V) = I_f\, kd \cos(\text{fita}) + I_f\, k_s \cos(\text{fi})$

  Fita= angle entre la normal en un vèrtex V i la direcció de la llum incident en ell.
  Fi= angle entre el raig reflectit en el vèrtex V i la recta que uneix el vèrtex amb l'observador.
- El color als punts interiors de les cares es calcularà interpolant el color dels seus vèrtexs (smooth shading).
- El focus està situat sobre la recta que passa just pel mig de les cares x=10 i x=40; per tant, pels 4 vèrtexs d'una mateixa cara, l'angle entre la direcció d'incidència de la llum i la seva normal és el mateix i, per tant, tots els vèrtexs tindran el mateix color difus => color de la cara constant. Podem desacartar la resposta *d*.

- Com les dues cares tenen el mateix color verd, són il·luminades pel mateix focus groc (per exemple, If=(1,1,0)); però l'angle fita és més gran pels vèrtexs de la cara x=40 que per la cara x=10 => la cara x=40 és veurà més fosca (cos(fita) serà més petit pels seus vèrtexs). Per tant, podem descartar les respostes *b* i *c.*
- Comprovem que la responsta *a* és correcta. Noteu que és impossible que l'observador vegi una taca especular al mig de la cara x=40 per molt especular que sigui, perquè el color es calcula en els vèrtexs. Com a molt podria veure la taca especular en un vèrtex (i aquesta resposta no hi és).

Pregunta 7 (1 punt) Quina d'aquestes tasques és responsabilitat del *briefer* en un estudi d'usabilitat.

a. Explicar als usuaris l'objectiu de l'estudi.
b. Fer l'anàlisi estadística de les dades.
c. Gravar per vídeo el que fan els usuaris.
d. Dirigir la confecció de l'informe final.

Pregunta 8 (1 punt) Una animació de progrés externa...

a. És necessària per totes les tasques que requereixin coordinació ull-mà.
b. Cal posar-la quan la tasca s'acaba en menys d'un segon sense bloquejar l'usuari encara que tinguem barra de progrés interna.
c. Es pot posar quan la tasca s'acaba en menys d'una dècima de segon.
d. És una finestra externa que pot distreure a l'usuari i que normalment s'acompanya d'informació extra que no hi cap en una animació interna.

Pregunta 9 (1 punt) En l'Alert Box de Jakob Nielsen "Mental Models" s'explica que

a. Un model mental es basa en fets, no creences.
b. Un model mental es basa en el que els usuaris coneixen d'un sistema (com una pàgina web).
c. Els models mentals són col·lectius, no individuals.
d. Els models mentals dels dissenyadors i dels usuaris són iguals.