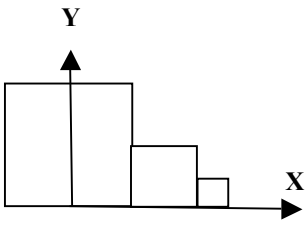
Estructura de dades i transformacions geomètriques[1]

1. (2005-2006 2Q) Es disposa d'una acció `pinta_esfera (c, r)` que, a partir de les coordenades del centre i del valor del radi d'una esfera, genera i utilitza primitives d'OpenGL per a enviar a pintar els triangles que la modelen. Es desitja visualitzar un sistema solar molt simple: el sol, els planetes i els satèl·lits es consideren esferes, i les òrbites dels planetes i dels satèl·lits es consideren circulars essent els eixos de gir paral·lels a l'eix Y del sistema de coordenades de l'escena i passant pel centre del sol (en el cas dels planetes) o dels planetes (en el cas dels seus satèl·lits). Es demana:
   - Dissenya una estructura de dades que permeti emmagatzemar la informació requerida per a programar un tros de codi que, utilitzant `pinta_esfera`, permeti obtenir imatges del sistema solar en diferents instants de temps.
   - Suposant que teniu definida una camera, dissenyeu el tros de codi que recorre l'estructura de dades dissenyada i utilitzant `pinta_esfera(c,r)` i comandes OpenGL genera la imatge del sistema solar en un instant de temps determinat.

2. (2006-2007 1Q) Disposem del model d'un personatge que té com a capsa contenidora la capsa definida pels punts MinCapsa=(10,10,10) i MaxCapsa=(20,20,12). Ens diuen que la part del davant d'aquest personatge és la que mira cap a la part positiva de l'eix Z. Considerem que tenim una habitació dividida en 10x10 cel·les quadrades de costat c=1 situades en el pla XZ (la cel·la x=1, z=1 té coordenades mínimes xmin=0 i zmin=0). Quines transformacions li haurem de fer al personatge si el volem situar centrat i escalat adientment en la cel·la x=5, z=3 de la nostra habitació i mirant cap a la cel·la x=5, z=4? Escriu el tros de codi amb les transformacions en OpenGL que li aplicaries. L'alçada del terra de l'habitació (sobre el qual cal posar el personatge) és y=0.

3. (2006-2007 1Q) Suposa definida una funció `dibuixaCub()` que dibuixa un cub d'aresta unitat centrat a l'origen. Fent servir aquesta funció i les transformacions geomètriques d'OpenGL, has d'obtenir una successió de tres cubs similar a la indicada a la figura. Tots els cubs estan recolzats sobre el pla XZ. El primer cub té aresta unitat i el centre de la seva base està ubicat a l'origen de coordenades. El segon està a la dreta del primer, adossat a ell, la seva aresta és la meitat que la d'ell i l'eix X passa pel centre de la seva base. Anàlogament el tercer cub respecte del segon.



4. (2006-2007 2Q) 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.

[1] Corresponen a exercicis d'examen de les assignatures VIG (pla 2013 d'Enginyeria Informàtica) i IDI del Grau en Enginyieria Informàtica (a partir del curs 2011-2012).

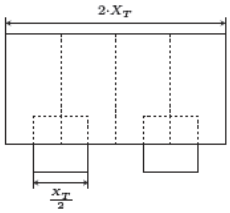
5. (2007-2008 1Q) Tenim la informació geomètrica (cares i vèrtexs) d'un objecte bàsic i un mètode `PintaObjecte()` que implementa el seu recorregut enviant a pintar les primitives gràfiques corresponents. Coneixem els punts de coordenades mínimes (MinX,MinY,MinZ) i màximes (MaxX,MaxY,MaxZ) de la seva capsa mínima contenidora (amb les seves arestes orientades segons els eixos de coordenades). Volem visualitzar una instància d'aquest objecte tal que l'amplada (mida segons l'eix X) de la seva capsa sigui AmplaX (escalat uniforme) i el centre de la base d'aquesta estigui ubicat en el punt (10,0,10). Indiqueu:
   - L'expressió de la composició de les transformacions geomètriques que cal efectuar a l'objecte bàsic per tal de visualitzar la instància utilitzant el mètode `PintaObjecte()`.
   - El codi OpenGL que hauríeu d'implementar. Ha de quedar clarament indicat quin és el contingut de la matriu de ModelView en el moment que comença el codi que proposeu.

6. (2007-2008 2Q) 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.

7. (2008-2009 1Q) La trajectòria d'un cotxe es representa per una poligonal de 20 punts ubicada en el pla x-y. Es disposa de la informació de la caixa mínima contenidora del cotxe i d'una funció `pinta_coche()` que pinta un cotxe centrat a l'origen. Indica l'expressió de les transformacions geomètriques que cal efectuar al cotxe per a generar una animació que ubica successivament el cotxe en els vèrtexs de la trajectòria, amb un escalat uniforme decreixent proporcional a l'índex del vèrtex del polígon (el cotxe es va fent petit). Indiqueu també el codi OpenGL que, suposant una càmera ja inicialitzada, pinti el cotxe en la posició indicada.

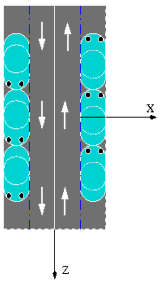
8. (2008-2009 2Q) Un joc de computadors es basa en la recreació d'una ciutat romana. Es disposa del model de fronteres de 5 romans amb diferent indumentària, els anomerarem "romans patró". En l'escena poden haver-hi fins a 100 romans de cada tipus (és a dir, fins a 100 de cada "patró"). La ubicació de cada romà queda determinada pel centre de la seva capsa mínima contenidora, la seva orientació per un vector i la seva grandària serà la del romà patró corresponent escalat un cert %. Es demana el disseny d'una estructura de dades compacta que permeti emmagatzemar les dades requerides per a la visualització de la població dels romans de la ciutat. El model de fronteres dels romans "patró" està constituït per una malla de triangles on cada triangle pot ser d'un color diferent.

9. (2009-2010 2Q) La facultat esta redissenyant les aules que vol usar pels propers cursos i, apart de fer-les una mica 20 més petites, volen també canviar les actuals cadires de braç per files de taules i cadires. Demanen que els hi mostrem com quedaria una d'aquestes aules, considerant que a cada taula hi volen posar dues cadires de manera que quedin com al dibuix, i volen fer 5 files de 5 taules cadascuna; les files estan separades entre sí per un espai igual al gruix d'una taula. Els alumnes han de quedar mirant cap a les X positives del sistema de coordenades de l'aplicació, i el terra ser_a el pla XZ (y=0). Disposem de métodes `pintaTaula()` i `pintaCadira()` que dibuixen els corresponents models de forma que ambdós tenen la direcció vertical paral·lela a l'eix Y , i la taula és més gran en X que en Z (i les vores estan alineades amb els eixos). La cadira es dibuixa amb el davant cap a la direcció de les X negatives, i totes dues peces es dibuixen centrades a l'origen. La taula que dibuixa pintaTaula() té una capsa mínima contenidora alineada amb els eixos que va des de (-XT;-YT;-ZT) a (XT;YT;ZT), i la cadira en té una que va des de (-XC;-YC;-ZC) a (XC;YC;ZC). Escriu una porció de codi OpenGL necessari per a pintar totes les taules i cadires de l'aula disposades d'acord amb aquestes especicacions, i considerant que la càmera ja ha estat definida.



10. (2010-2011 2Q) Tenim un cilindre d'altura 1 i radi 1 amb una de les seves bases centrada al (0, 0, 0) i l'altra al (0, 0, 1). Suposant que la funci´o `dibuixaCilindre()` ja fa les crides necessaries a OpenGL per a pintar aquest cilindre, escriu el codi OpenGL que es necessitaria per a obtenir en la nostra escena un cilindre de radi 2 i amb les seves bases situades als punts (2,-1,-1) i (6,-1,-1).

11. (2011-2012 1Q) Disposem d'un model de cotxe que la seva part del davant mira cap a la part positiva de l'eix X i la seva capsa contenidora ve definida pels punts Min=(3,2,0) i Max=(5,3,1). Tenim la funció `PintaCotxe()` que fa les crides a OpenGL per a pintar aquest cotxe. Disposem també d'un model de carrer, d'amplada 4, centrat a l'origen, que el seu eix central coincideix amb l'eix Z i pel qual circulen els cotxes en totes dues direccions (és a dir el carrer és de doble sentit). Volem disposar d'aquests cotxes aparcats a les 2 vores del carrer de manera que estiguin aparcats orientats en la direcci´o correcta de circulació del carrer. Suposant que els cotxes no s'han d'escalar i que han d'estar tocant-se amb el de davant/darrera (veure esquema de la figura), indica el tros de codi OpenGL que usaries per a pintar aquests 6 cotxes suposant una càmera ja inicialitzada.



12. (2011-2012 2QP) Disposem del model geomètric d'un cub de costat 1 amb cares paral·leles als plans coordenats amb el centre de la seva base a (0,0,0). Suposant que la funció `dibuixaCub()` ja fa les crides necessaries a OpenGL per a pintar aquest cub, escriu (i justifica tot indicant la transformació geomètrica requerida) el codi OpenGL "específic"que es necessitaria per a obtenir una escena formada per dos cubs: un de costat 2 amb el centre de la seva base a l'origen de coordenades i l'altre amb els vèrtexs de la seva cara inferior al centre de les arestes de la cara superior del primer cub. Considereu la càmera ja inicialitzada, només cal el codi per a pintar els dos cubs de l'escena.

13. (2011-2012 2QP) Suposem que tenim el model geomètric d'un prisma de base hexagonal amb una capsa contenidora de dimensions a=40, h=100 i z=40, centrada en el punt C=(50,0,0). Es vol obtenir la visualització d'una escena formada per dos prismes orientats segons l'eix Z positiu; el primer ha de tenir alçada 100 i estar centrat a l'origen de coordenades, el segon d'alçada 50 ha d'estar ubicat just darrera del primer. Si disposem d'una funció `dibuixaPrisma()` que fa les crides a OpenGL per pintar el model del prisma. Escriu (i justifica) el codi OpenGL "específic" que es necessari per a generar l'escena desitjada.

14. (2012-2013 1QP) Es vol crear un ninot de neu mitjançant dues esferes i un con. L'esfera inferior ha de tenir el seu centre a (0, 10, 0) i un radi de 10, i la superior, que simula el cap, cal posar-la amb el seu centre a (0, 25, 0) i amb un radi de 5. El con té un radi de 2, llargada de 5, i el centre de la seva base a (2.5, 25, 0) i està orientat segons l'eix X.
    Es disposa de les següents funcions:
    void glutSolidSphere(GLdouble R, GLint slices, GLint stacks);
    que pinta una esfera centrada en l'origen de radi R, i
    void glutSolidCone(GLdouble R, GLdouble h, GLint slices, GLint stacks);
    que pinta un con orientat segons l'eix Z amb la seva base en el pla Z=0, essent R el radi de la seva base i la seva llargada h.
    Es demana: Dóna el codi necessari en OpenGL per a dibuixar els tres objectes indicats en les posicions i mides donades, usant *glutSolidSphere* i *glutSolidCone*. Suposeu que la càmera i el viewport estan correctament inicialitzats.

15. (2012-2013 1QP) En unes eleccions es presenten 3 partits que obtenen, respectivament, 50, 25, i 0 escons. Es vol fer una visualització 3D que il·lustri els resultats. A tal efecte, es representen els partits amb escons amb un cilindre vertical (eix orientat segons +Y) de radi R=1 amb una alçada igual a la dècima part del seu nombre d'escons i amb el centre de les seves bases als punts (0,0,0) i (2,0,0), respectivament. Per a fer una mica de gràcia, el partit que no ha obtingut cap escó es vol representar con un cilindre tombat: centre de la seva base en (4,1,0), eix en direcció +Z i alçada 1. Es disposa de la funció:

    void drawSolidCylinder (GLdouble R, GLint Slices, GLint stacks);

    que pinta un cilindre de radi R i alçada 1, amb centre de la base a (0,0,0) i eix en la direcció +Z.
    Es demana: el codi necessari en OpenGL, usant *drawSolidCylinder(...),* per a dibuixar els cilindres en les posicions indicades. Suposeu que la càmera i el viewport (vista) estan correctament inicialitzats.

16. (2012-2013 1QP) Com visualització il·lustrativa del joc de tres en ratlla, es vol mostrar una imatge formada per tres Homers de la mateixa alçada (h=2) tals que el seus centres estan ubicats a: (0,1,0), (-2,1,0), (2,1,0), i els seu davant orientats, respectivament, en la direcció dels eixos +Z, -X i +X. Es disposa de la funció:

    void drawHomer (GLdouble h);

    que pinta un Homer d'alçada "h" amb el seu davant orientat cap a l'eix +Z i centre en (0,1,0). Observació: podeu considerar que l'esfera contenidora del Homer té radi=1.
    Es demana: usant *drawHomer(),* el codi necessari en OpenGL per a dibuixar els Homers de l'alçada i orientació indicades i en les posicions donades. Suposeu que la càmera i el viewport (vista) estan correctament inicialitzats.

Solucions

3.
```
glMatrixMode(GL_MODELVIEW);
glPushMatrix();
glTranslatef(0,0.5,0);
dibuixaCub();
glPopMatrix();
glPushMatrix();
glTranslatef(0.75, 0.25,0);
glScalef(0.5,0.5,0.5);
dibuixaCub();
glPopMatrix();
glPushMatrix();
glTranslatef(1.125,0.125,0);
glScalef(0.25,0.25,0.25):
dibuixaCub();
glPopMatrix();
```

5. Per a ubicar i escalar l'objecte adequadament, traslladarem el centre de la base de la capsa contenidora a l'origen de coordenades, seguidament realitzarem un escalat uniforme respecte l'origen de valor AmpleX/(MaxX-MinX) i, per últim, traslladarem l'objecte a la posició final (com que tenim el centre de la base de la capsa a l'origen i aquest és el punt que volem situar al punt donat, directament farem la translació al punt donat).

a) Si considerem escalat = AmplaX/(MaxX-MinX) tenim que la composició de transformacions és:
TG = T(10, 0, 10) * S(escalat, escalat, escalat) * T(-(MaxX+MinX)/2, -MinY, -(MaxZ+MinZ)/2)

b) Suposem que el contingut de la matriu ModelView en aquest punt és la matriu que permet portar els vèrtexs del sistema de coordenades de l'aplicació al sistema de coordenades de l'observador, és a dir, en aquest punt del codi la matriu ModelView conté la transformació de coordenades:

```
escalat = AmplaX/(MaxX-MinX);
glPushMatrix ();
glTranslatef (10, 0, 10);
glScalef (escalat, escalat, escalat);
glTranslatef (-(MaxX+MinX)/2, -MinY, -(MaxZ+MinZ)/2);
PintaObjecte ();
glPopMatrix ();
```

7. Suposant l'índex i de la trajectòria, i sabent que pinta_cotxe() ja pinta el cotxe a l'origen de coordenades, les transformacions que caldria fer al cotxe són:
   1) Escalar-lo de forma homogènia en, per exemple, (1-i*0.05)
   2) Traslladar-lo a la posició i de la trajectòria

   Si tenim un vector trajectoria on tenim guardats els vèrtexs de la trajectòria, l'expressió de la transformació geomètrica a aplicar al cotxe en l'índex i seria:

   TG = T(trajectoria[i]) * S(1-i*0.05, 1-i*0.05, 1-i*0.05)

   El codi OpenGL suposant que les matrius ja estan inicialitzades amb els paràmetres de la càmera seria:

```
glMatrixMode (GL_MODELVIEW);
for (int i=0; i<20; ++i) {
    float escalat = 1-i*0.05;
    glPushMatrix () ;
    glTranslatef(trajectoria[i].x,trajectoria[i].y,trajectoria[i].z);
    glScalef (escalat, escalat, escalat);
    pinta_cotxe ();
    glPopMatrix ();
    }
```

15. Les transformacions geomètriques a aplicar a cada objecte, d'acord amb l'enunciat i les funcions de pintat de l'encunciat, són:

    TG1= Escalat(1,5,1)*Girx (-90)
    TG2= Tr(2,0,0)*Escalat(1,2.5,1))*Girx (-90)
    TG3= Tr(4,1,0)

    El codi OpenGL seria:

```
glMatrixMode(GL_MODELVIEW);
// pintat cilindre del partit amb 50 escons
glPushMatrix();
glScalef(1.,5.,1.)
glRotatef (-90.,1,0.,0);
drawSolidCylinder(1.,20,20);
glPopMatrix();
//pintat cilindre del partit amb 25 escons
glPushMatrix();
glTranslatef (2.,0.0,0.);
glScalef(1.,2.5,1.)
glRotatef (-90.,1,0.,0);
drawSolidCylinder(1.,20,20);
glPopMatrix();
//pintat cilindre del partit amb 0escons
glPushMatrix();
glTranslatef (4.,1.0,0.);
drawSolidCylinder(1.,20,20);
glPopMatrix();
```