Nom i Cognoms: _____Grup: ____DNI: _____

1 (3 punts) Volem crear una escena que representa un mini sistema solar format pel sol, un planeta i un satèl·lit. Tots es representaran per esferes: el sol per una de radi 5 amb centre a l'origen de coordenades, el planeta per una de radi 3 que gira entorn del sol sobre una circumferència de radi 20, i el satèl·lit per una de radi 1 que gira entorn del planeta en una òrbita circular de radi 5. Per a pintar les esferes s'utilitza la instrucció *glutSolidSphere(R,20,20)* que pinta una esfera de radi *R* centrada en l'origen de coordenades. Si α i β indiquen els angles que han girat, respectivament, el planeta i el satèl·lit en les seves òrbites:

- indica l'expressió de les transformacions geomètriques que cal aplicar a l'esfera de radi R centrada a l'origen per a ubicar cada astre en l'escena,
- escriu el tros de codi OpenGL que pinta l'escena.

- *El sol ha de tenir el centre en la mateixa posició en què el pinta glutSolidSphere(), per tant, no cal aplicar-li cap transformació.*
- *El planeta estarà ubicat a una distància 20 del sol i girarà entorn de l'eix Y de l'escena (el que passa pel sol); per tant la transformació que cal aplicar-li és* $TG= G_Y(\alpha) * T(20,0,0)$
- *EL satel·lit, d'una banda gira entorn de l'eix paral·lel a l'eix Y que passa pel centre del planeta i està a una distància 5 d'aquest. Per tant, imaginant que el planeta estigues a l'origen li faríem* $TG_p= G_Y(\beta) * T(5,0,0)$*, a més a més, li hem de fer girar el mateix que gira el planeta respecte el sol. Per tant, la seva:* $TG= G_Y(\alpha) * T(20,0,0)* G_Y(\beta) * T(5,0,0)$

```
glMatrixMode(GL_MODELVIEW);
glutSolidSphere(5,20,20);
glPushMatrix();
glRotatef(α, 0,1,0);
glTranslatef(20,0,0);
glutSolidSphere(3,20,20);
glRotatef(β,0,1,0);
glTranslatef(5,0,0);
glutSolidSphere(1,20,20);
glPopMatrix();
```

Nom i Cognoms: _____Grup: ____DNI: _____

2 (3 punts) Indica TOTS els paràmetres d'una càmera axonomètrica per veure una vista en planta de l'escena anterior, sense retallar, sense deformació, i optimitzant l'espai en un *viewport* quadrat. Defineix la posició i orientació de la càmera amb transformacions geomètriques. Dibuixa també la imatge que obtindries. Justifica l'elecció de cadascun dels paràmetres.

Donat que les circumferències (òrbites) estan paral·leles al pla XZ i el centre del sol està a l'origen, per a tenir una vista en planta, l'observador ha d'estar ubicat sobre l'eix Y mirant cap a l'origen (si volem vista en planta) i a una distància del pla que permeti veure l'esfera més gran => d> 5; per tant, una alternativa és:

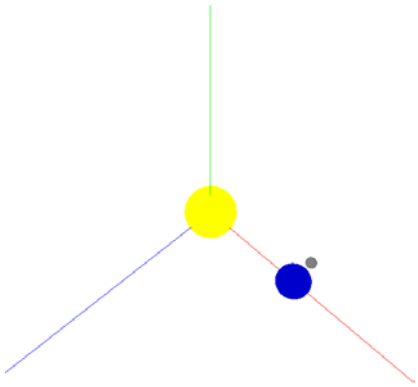
- *OBS=(0,10,0), VRP=(0,0,0) i up=(0,0,-1)*

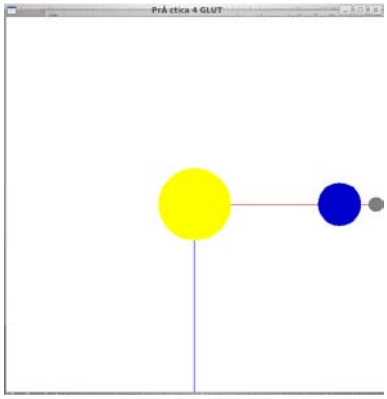
En transformacions geomètriques

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0,0,-10);
glRotatef(90,1,0,0);
```

L'òptica és axonomètrica, si cal veure tota l'escena sense retallar, cal estar segurs que el window (que podem centrar respecte la càmera) conté la posició del satel·lit més allunyada del sol, o sigui,
D= 20+5+1=26 (radi òrbita planeta+radi òrbita satel·lit+ radi satèl·lit). Com el viewport és quadrat, el window també per a què no hi hagi deformacions. El window més ajustat serà (-26,26,-26,26).
En quant al camp de visió, per assegurar no retallar: Znear <15, posem 10 i Zfar>25 posem 30.

Una imatge general i altre en planta s'observen en les figures següents:





Nom i Cognoms: _____Grup: ____DNI: _____

3 (1 punt) Tenim una càmera axonomètrica que permet veure tota una escena sense retallar ni deformar, des de qualsevol punt de vista, amb l'òptica definida amb:

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho(-10, 10, -2, 2, 1, 30);
```

Modifiquem la relació d'aspecte del *viewport* a 1, quina seria la crida a glOrtho correcta per a continuar veient tota l'escena sense deformacions?

a. **glOrtho(-10, 10, -10, 10, 1, 30);**
b. glOrtho(-10, 10, 1, -1, 1, 30);
c. glOrtho(-2, 2, -2, 2, 1, 30);
d. glOrtho(-5, 5, -5, 5, -5, 5);

4 (1 punt) Disposem d'una càmera ortogonal amb els següents paràmetres: OBS=(0.,0.,0.), VRP=(-1.,0.,0.), up=(0.,1.,0.), window de (-5,5) a (5,5), ra=1, zn=5, zf=10.
Indiqueu quin conjunt de paràmetres d'una càmera perspectiva defineix un volum de visió que conté l'anterior (és a dir, garanteix que es veurà, coma mínim, el mateix que amb la càmera axonomètrica):

a. **FOV= 90, ra=1, zn= 5, zf=10**
b. FOV= 60, ra=1, zn=5, zf=10
c. FOV= 60, ra= 2, zn=6, zf=11
d. FOV= 90, ra= 0.5, zn=5, zf=10

5 (1 punt) Tenim una escena que es pinta de la següent forma:

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glEnable (GL_DEPTH_TEST);
glMatrixMode(GL_PROJECTION);
glLoadIndentity();
gluPerspective(60.0, 1.0, 1.0, 100.);
glMatrixMode(GL_MODELVIEW);
glTranslatef(0.0, 0.0, -20.0);
glRotated(-90.0, 0.0, 1.0, 0.0);

glColor3f(1.0,0.0,0.0);
glBegin(GL_QUADS);
glVertex3f(0.0, -2.0, -1.0);
glVertex3f(0.0, -2.0, 1.0);
glVertex3f(0.0, 2.0, 1.0);
glVertex3f(0.0, 2.0, -1.0);
glVertex3f(0.0, 2.0, -2.0);
glVertex3f(0.0, 2.0, 2.0);
glVertex3f(0.0, 4.0, 2.0);
glVertex3f(0.0, 4.0, -2.0);
glEnd();
```

Digues què es veu:

a. Una espècie de *T* invertida formada per dos rectangles de color vermell amb el seu centre una mica per sobre del centre de la pantalla.
b. Dos rectangles, un vertical i un d'horitzontal que formen una espècie de *L* centrada en la pantalla una mica cap a l'esquerra.
c. Una línia vermella que va des del centre de la pantalla fins una mica més amunt sense arribar al límit superior del *viewport.*
d. **Una espècie de *T* formada per dos rectangles de color vermell amb el seu centre una mica per sobre del centre de la pantalla.**

6 (1 punt) Per a posicionar una càmera perspectiva a una determinada distància del VRP i en una determinada orientació, el codi OpenGL ha de realitzar una sèrie de crides de transformacions geomètriques. Digues quina de les combinacions següents de crides a rotacions i translacions conté les crides necessàries i està en l'ordre correcte:

a. Rotació respecte X, rotació respecte Z, rotació respecte Y, translació –VRP.
b. Rotació respecte Z, rotació respecte Y, rotació respecte X, translació –VRP i translació en Z –distància.
c. **Translació en Z –distància, rotació respecte Z, rotació respecte X, rotació respecte Y, translació –VRP.**
d. Rotació respecte Z, rotació respecte Y, rotació respecte X, translació –VRP.