Pregunta 1 (2 Punts) Disposem del model geomètric d'un objecte i de la funció `pinta_model()` que encapsula les primitives gràfiques requerides per a pintar les seves cares. Els vèrtexs extrems de capsa mínima contenidora del model tenen coordenades (xmin,ymin,zmin) i (xmax,ymax,zmax). El "davant" del model està orientat segons les Z^+. Es vol visualitzar una escena formada per dos instàncies del model: una d'elles (objecte 1) amb el centre de la base de la seva capsa a l'origen de coordenades i escalat de manera que la seva capsa sigui un cub d'aresta 2; i l'altra (objecte 2) que quedi just damunt del primer, de la meitat de la mides i cap per baix. Es demana: el codi del mètode `pinta_escena()` que utilitzant `pinta_model()` permet generar l'escena. Cal justificar les transformacions geomètriques utilitzades.

D'acord amb les condicions de l'enunciat, les transformacions a efectuar a cada objecte són

TG_ Objecte1= Escalat (sx,sy,sz)*Trans(-CentreBaseCapsa)

TG_ Objecte2= Trans (0,3,0)*Gir$_Z$(180)* Escalat (sx',sy',sz')*Trans(-CentreBaseCapsa)

on:

- CentreBaseCapsa són les coordenades de la base de la capsa del model geomètric; les utilitzem per a portar aquest punt de referència a l'origen de coordenades.
- sx, sy, sz indiquen l'escalat a efectuar respecte l'origen de coordenades a l'objecte 1. sx=2/(xmax-xmin) essent (xmx-xmin) l'amplada de la capsa i 2 la seva mida final desitjada. Anàlogament es calculen sy=2/(ymax-ymin) i sz=2/(zmax-zmin).
- sx', sy', sz' indiquen l'escalat a efectuar respecte l'origen de coordenades a l'objecte 2. sx'=1/(xmax-xmin) essent (xmx-xmin) l'amplada de la capsa i 1 la seva mida final desitjada. Anàlogament es calculen sy=1/(ymax-ymin) i sz=1/(zmax-zmin).
- Gir$_Z$(180)* és el gir per a posar l'objecte 2 cap per avall.
- Trans (0,3,0) és la translació per a posar l'objecte 2 a sobre del objecte 1.

```
glMatrixMode (GL_MODELVIEW);
glPushMatrix();
glScaled(2/(xmax-xmin),2/(ymax-ymin), 2/(zmax-zmin));
glTranslated((xmax+xmin)/2, ymin, (zmax+zmin)/2));
pinta_model();
glPopMatrix();
glPushMatrix();
glTranslated (0,3,0);
glRotated(180,0,0,1);
glScaled(1/(xmax-xmin),1/(ymax-ymin), 1/(zmax-zmin));
glTranslated((xmax+xmin)/2, ymin, (zmax+zmin)/2));
pinta_model();
glPopMatrix();
```

Pregunta 2 (2 Punts) Indiqueu i justifiqueu tots els paràmetres d'una càmera axonomètrica que permet veure a la pantalla només l'objecte 2 cap per amunt i optimitzant l'espai d'un *viewport* de 600x500 sense deformacions. La posició de la càmera l'heu de definir amb VRP, OBS i up. Podeu deixar indicats (però explicats) els càlculs que requeriu.

Si només cal veure l'objecte 2 i volem optimitzar l'espai, una possible solució és:

- VRP (punt d'enfoc) ubicat al seu centre: (0,2.5,0)
- Per veure l'objecte 2 cap per amunt, up=(0,-1,0)
- Si volem veure la cara de l'objecte 2, OBS mirant cap a VRP en direcció de Z-: (0,2.5,10)
- Per a que quedi dins del camp de visió: Znear=9.5, Zfar=10.5
- Per a que maximitzi espai en pantalla, el *window* mínim seria (d'acord amb les mides de l'objecte): (-0.5,0.5,-0.5,0.5) però com ra_viewport=600x500=1.2 hem d'augmentar l'amplada del window perquè tingui aquesta relació d'aspecte i no hi hagi deformacions => window=(-0.6,0.6,-0.5,0.5)

Pregunta 3. (1 Punt) Es vol simular que l'objecte 2 de l'escena de l'exercici 1 es desplaça cap amunt (eix Y+) mentre gira sobre sí mateix. Es proposa el codi adjunt, creieu que s'obtindrà l'efecte desitjat?
Observació: podeu suposar que tant les transformacions (TG1 i TG2) per a ubicar les dues instàncies en la posició que indica l'exercici 1 com la càmera estan correctament definides. L'angle "alfa" indica la rotació de l'objecte 2 i "distY" el seu desplaçament respecte la posició inicial en un moment donat.

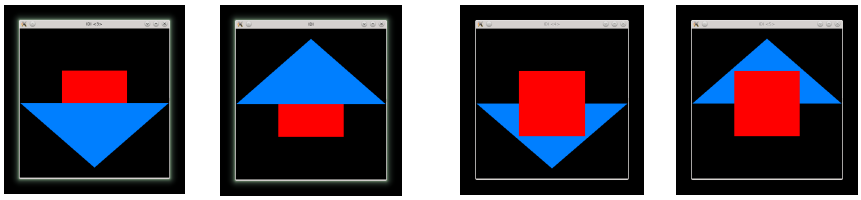
```
1.  ...
2.  glPushMatrix();
3.  // aquí el codi TG1 per ubicar l'objecte 1
4.  pinta_model();
5.  glPopMatrix();
6.  glPushMatrix();
7.  glTranslated (0,distY,0);
8.  glRotated(alfa,0,1,0);
9.  //aquí el codi TG per ubicar l'objecte 2
10. pinta_model()
11. glPopMatrix();
12. ...
```

a) No s'obté l'efecte desitjat, les instruccions 7 i 8 haurien d'anar després de les indicades en 9.
b) **S'obté l'efecte desitjat.**
c) No s'obté l'efecte desitjat, caldria permutar les instruccions 7 i 8.
d) No es pot contestar sense saber la relació d'aspecte del *viewport*.

Pregunta 4. (1 Punt) Pintem una escena amb el següent codi:

```
void renderScene(void)
{
  glClearColor(0.0, 0.0, 0.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glDisable(GL_DEPTH_TEST);
  // aquí es defineix l'òptica de manera correcta
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glTranslatef(0.0, 0.0, -10.0);
  glRotatef(-90.0, 1.0, 0.0, 0.0);
  glColor3f(0.0, 0.5, 1.0);
  glutSolidCone(5.0, 5.0, 20, 20); //radi, alçada, orientació Z+
  glColor3f(1.0, 0.0, 0.0);
  glutSolidCube(4.0);
  glutSwapBuffers();
}
```

Digues quina de les imatges següent es veu.



a. La Primera imatge
b. La segona imatge
c. La primera imatge
d. La quarta imatge

Pregunta 5. (1 Punt). La posició i orientació d'una càmera s'ha definit amb el codi següent:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslated(0,0,-10);
glRotated (90,1,0,0);
glRotated(-90,0,1,0);
```

Quins valors de OBS, VRP i up hauries d'utilitzar per a obtenir la mateixa imatge utilitzant gluLookAt(...)?

a. OBS=(0,10,0), VRP=(0,0,0), up=(1,0,0)
b. OBS=(10,0,0), VRP=(0,0,0), up= (0,1,0)
c. OBS= (0,10,0), VRP= (0,0,0), up= (-1,0,0)
d. OBS=(0,-10,0), VRP=(0,0,0), up= (1,0,0)

Pregunta 6. (1 Punt). Suposant que tenim una càmera axonomètrica amb la seva posició definida amb VRP, OBS i up. Quina de les següents afirmacions NO es correcta?

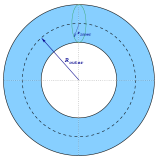
a. Si movem l'observador (OBS) avançant en la direcció de visió, la grandària dels objectes no s'incrementa.
b. Com zN pot ser negatiu, no ens hem de preocupar d'actualitzar el VRP quan l'observador (OBS) avança en la direcció de visió.
c. Si l'observador gira per avançar en altre direcció, cal que modifiquem la posició de VRP.
d. Com zN pot ser negatiu, podem veure objectes situats darrera de l'observador.

Pregunta 7. (1 Punt). La finestra gràfica (pantalla) és de 500x1000 píxels i es defineix una càmera axonomètrica amb `glOrtho (-5,5,-10,10,10,20)` i un *viewport* amb `glViewport(0,0,500,1000)`. L'usuari realitza un *resize* i la finestra gràfica passa a ser de 1000x1000, es torna a enviar a pintar l'escena sense modificar el *viewport* ni els paràmetres de la `glOrtho()`. Quina de les següents afirmacions és correcta?

a. Es veurà la mateixa imatge però, si abans quedava ajustada a la pantalla, ara quedarà espai buit als dos costats.
b. Hauria de modificar el *window* a quadrat altrament hi haurà deformacions.
c. Veurà la mateixa imatge però ocupant només la meitat esquerra de la pantalla.
d. Veurà més part de l'escena que abans (pels costats).

Pregunta 8 (1 punt). Pintem un Torus utilitzant el següent codi:

```
void renderScene(void) {
  glClearColor(0.0, 0.0, 0.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-2., 2., -2., 2., -2.0, 12.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glTranslatef(0.0, 0.0, -5.0);
  glRotatef(90.0, 0.0, 1.0, 0.0);
  glColor3f(0.1, 0.8, 0.1);
  glutSolidTorus(1.0, 2.0, 20, 20);
  glutSwapBuffers();
}
```



Què es veu?

a. Dues circumferències de color verd, de la mateixa mida, sobre fons negre una a la dreta de l'altra.
b. Dues circumferències de color verd, de la mateixa mida, sobre fons negre, una a sobre de l'altra.
c. Es veu tota la pantalla negra, sense cap cercle ni oval ni circumferència.
d. Un rectangle vertical de color verd que travessa la pantalla de dalt a baix.