Exercicis de Càmera[1]

1. (2004-2005 1Q) Donada la visualització d'una malla de triangles, tenim una interfície que ens permet seleccionar cares a base de moure un cub. Les cares seleccionades seran les interiors al cub. La grandària del cub permet modificar la grandària de la zona a seleccionar. Quan es mou el cub, el seu centre sempre està a sobre d'un dels triangles de la malla i el cub es va orientant de forma automàtica, de manera que quatre de les seves arestes tenen la direcció del vector normal al triangle que conté el centre del cub. Volem que les cares seleccionades es visualitzin en una segona finestra gràfica mentre movem el cub. Si la interfície ens retorna les coordenades del centre del cub, la normal del triangle que conté el centre i la longitud de les arestes del cub, descriviu i justifiqueu els paràmetres de la càmera que hauríem de definir per a garantir que els triangles seleccionats es visualitzin en la vista de la segona finestra gràfica ocupant el màxim d'aquesta.

2. (2004-2005 2Q) Tenim una càmera perspectiva amb el **VRP** a l'origen de coordenades. Volem que un triangle de vèrtexs (2,0,0), (0,2,0), (0,0,2) en el sistema de coordenades de l'aplicació es vegi **també** equilàter a la vista. Quins valors hem de donar als altres paràmetres que defineixen la càmera?.

3. (2005-2006 1Q) Una escena està formada per dues esferes A i B. L'esfera B té radi 3 i està centrada en el punt (5,0,0), i l'esfera A té radi 2 i es troba centrada al punt (12,0,0). Podem definir una càmera perspectiva tal que faci que, si pintem l'escena amb eliminació de parts ocultes, només veiem l'esfera A –sense retallar- donat que l'esfera B queda totalment amagada darrera de l'esfera A? Indica com calcularies tots els paràmetres de la càmera.

4. (2005-2006 2Q)Considera aquest fragment de codi que defineix una càmera perspectiva:

```
1. glMatrixMode(GL_PROJECTION);
2. glLoadIdentity();
3. gluPerspective(60, 1, znear, zfar);
4. glMatrixMode(GL_MODELVIEW);
5. glLoadIdentity();
6. glTranslatef(0, 0, -20);
7. glRotatef(90, 0, 1, 0);
8. glTranslatef(-10, -10, -10);
9. drawModel();
```

   Indica i justifica els valors dels paràmetres **OBS**, **VRP** i up que hauries d'utilitzar per a definir la mateixa càmera amb `gluLookAt(...);`

5. (2005-2006 2Q) Considera aquest fragment de codi:

```
1. glViewport(0, 0, w, h);
2. glMatrixMode(GL_PROJECTION);
3. glLoadIdentity();
4. gluPerspective(60, ...);
5. glMatrixMode(GL_MODELVIEW);
6. glLoadIdentity();
7. glTranslatef(0, 0, -10);
8. drawSphere(5);
```

   on la funció `drawSphere(5)` envia a OpenGL els vèrtexs d'una esfera de radi 5 centrada en l'origen. El viewport és quadrat (w = h). Quins valors donaries a la resta de paràmetres de `gluPerspective` per tal d'obtenir com a resultat una el·lipse amb el doble d'alçada que d'amplada?

---

[1] Basats en exercicis d'examen de les assignatures VIG del pla 2013 d'Enginyeria Informàtica i IDI del Grau en Enginyieria Informàtica (a partir del curs 2011-2012). Els darrers són els més recents. Recordeu que en la DAFIB i en la web de la biblioteca hi ha la solució de molts dels exàmens.

6. (2006-2007 Q1) Suposem que tenim una escena en què la seva esfera contenidora està centrada a l'origen i té com radi 50. Inicialment tenim una definició de càmera donada per les sentències següents d'OpenGL:

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
gluPerspective (60, 1, 50, 150);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
glTranslatef (0, 0, -100);
```

   Quins paràmetres ens caldria canviar d'aquesta definició de càmera si volem realitzar un *zoom* apropant l'observador cap el centre de l'esfera?

7. (2006-2007 Q1) Estem inspeccionant un objecte modelat amb molts triangles i ens volem fixar en una zona quasi plana *P* (els triangles d'aquesta zona estan tots quasi en el mateix pla). La zona té unes dimensions de 4 x 7 mm. La interfície ens dóna el punt mig de la zona *P*, i el vector normal del seu pla aproximat, **n**. La distància màxima entre aquest pla aproximat i els vèrtexs dels triangles de la zona que volem inspeccionar és d'1 mm. Especifica una càmera que ens permeti veure bé la zona que volem inspeccionar.

8. (2006-2007 Q1) Estem inspeccionant una escena i definim els paràmetres de la càmera a partir de la seva esfera contenidora. Quin inconvenient pot tenir fixar el valor de la relació d'aspecte de la càmera a 1?

9. (2006-2007 Q1) Tenim una càmera perspectiva definida per OBS=(0., 50., 0.), VRP=(0., 0., 0.), **VUV**=(0., 0., 1.) i tal que l'angle alfa d'obertura de la càmera és de 45 graus. Indica (i justifica) quines podrien ser les coordenades (en el sistema de coordenades de l'aplicació) d'un punt que veiem al centre de la vora superior de la vista. (Per si ho necessites: sin(45)=0.707; cos(45)=0.707; tangent(45) = 1).

10. (2006-2007 Q2) Una escena consisteix en una esfera de radi R el centre de la qual es mou al llarg d'una trajectòria triangular amb vèrtexs A, B i C de coordenades conegudes. Calcula la posició d'una única càmera perspectiva capaç de mostrar-nos tota l'evolució de l'objecte al llarg de la seva trajectòria a una vista (*viewport*) de 800x600 píxels (amplada x alçada), de forma que:
    - La direcció de visió sigui perpendicular al pla de la trajectòria.
    - L'objecte no resulti retallat en cap instant.
    - No hi hagi deformacions.
    - La vista s'aprofiti raonablement (al màxim possible, però s'accepta una bona aproximació si resulta substancialment més senzilla de calcular sense importants pèrdues d'espai a la pantalla).

11. (2006-2007 Q2P) Diposem d'una càmera axonomètrica amb els següents paràmetres: OBS=(0,0,0), VRP=(0,0,-1), VUV=(0,1,0), Window de (-5, -5) a (5, 5), Zn =5, Zf =10. Indiqueu un altre conjunt de paràmetres que defineixin exactament el mateix volum de visió, però donant a **OBS, VRP, VUV**, Zn i Zf valors diferents. Justifiqueu la resposta.

12. (2006-2007 Q2P) Indiqueu les tres inicialitzacions independents que requereix una càmera per aconseguir les vistes ortogonals (planta, alçada i perfil) d'una escena tal que la seva esfera contenidora està centrada a l'origen i té radi 20. Concretament, cal que indiqueu els paràmetres de les transformacions geomètriques que s'aplicaran als objectes (amb OpenGL) per a veure l'escena en planta, alçat i perfil. Nota: al començar cada inicialització tenim com a ModelView la matriu identitat. No cal que inicialitzeu la `gluPerspective()`.

13. (2006-2007 Q2P) Tenim un rectangle centrat a l'origen de coordenades amb els costats paral·lels als eixos X i Y de l'aplicació i de mides 4cm d'ample i 3cm d'alçada. Tenim una càmera inicialitzada amb OBS=(0,0,5), VRP=(0,0,0), VUV=(0,1,0). El viewport és de 400 píxels

d'amplada i 600 d'alçada. Indiqueu els paràmetres amb què cal inicialitzar gluPerspective() de manera que encara que fem girar el rectangle respecte l'eix Z de l'aplicació sempre es visualitzi sense deformacions en el viewport i sense ser retallat. Nota: la càmera és fixa, és a dir, encara que girem el rectangle NO modifiquem la càmera.

14. (2006-2007 Q2P) Tenim una escena amb moltes esferes, totes elles de radi R. La seva ubicacio és arbitrària, però sabem que la distància mínima entre centres de qualsevol parella d'esferes és no inferior a 4*R. Cada esfera guarda informació de la posició del seu centre i dels tres vectors unitaris vx,vy,vz que defineixen un sistema local de coordenades (diferent per cada una d'elles). La superfície de l'esfera conté un text en relleu, que es llegeix bé quan es projecta en la direcció Z del seu sistema local; en aquest cas, el text segueix la direcció de l'eix X d'aquest sistema de coordenades de l'esfera. Especifiqueu els paràmetres d'una càmera que visualitzi una de les esferes de manera que es llegeixi bé el seu text. Heu d'evitar que el text sigui tapat totalment o parcialment per altres esferes de l'escena. La vista (viewport) té una mida de 600 píxels d'ample i 400 d'alçada.

15. (2007-2008 Q1) Dos estudiants volen obtenir una visualització en planta d'una escena que té el seu centre en l'origen de coordenades. Un d'ells utilitza el tros de codi codi-1 i l'altre el codi-2 per a ubicar la càmera. Creus que obtindran la mateixa imatge? Per què? Justifiqueu la resposta.

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho (-20,20,-20,20,5,15);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,10,0,0,0,0,0,0,-1);
```

codi-1 (càmera_1)

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho (-20,20,-20,20,5,15);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0,0,-10);
glRotatef(90,1,0,0);
```

codi-2 (càmera_2)

16. (2007-2008 Q1) Disposem del model geomètric de les cares i vèrtexs d'un cub amb longitud d'aresta l i ubicat en una posició i orientació arbitrària en l'espai. L'usuari selecciona una de les seves cares (cara_i). Descriviu el procediment per a calcular totes les inicialitzacions d'una càmera (posició, orientació, tipus,...) que permeti visualitzar la cara_i en pantalla en verdadera magnitud; és a dir, que multiplicant per un mateix factor d'escala les seves mides en píxels és corresponguin a les reals. El viewport és de 1024x1024. Suposeu que teniu accés a tota la informació geomètrica de la cara_i.

17. (2007-2008 Q1P) Imaginem que tenim definida una càmera perspectiva que permet veure tota l'escena. Com sabeu, per tal d'actualitzar adequadament la visualització de l'escena quan la finestra GLWidget pateix algun canvi de dimensions, s'ha d'actualitzar, si s'escau, l'angle d'obertura de la càmera (fovy). Completeu el següent mètode `setProjection()` per a implementar aquesta actualització si no volem deformacions i volem continuar veient tota l'escena en la vista.

```
void GLWidget::setProjection()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    double ar = (double) width()/height();
    double fovy;
    if (ar>=1) fovy=anglecam;
    else {
    ...
    }
    gluPerspective(fovy,ar,znear,zfar);
}
```

18. (2007-2008 Q1P) Una escena està formada per dos cubs, un de costat 20 centrat al punt (0,0,0), i l'altre de costat 10 centrat al punt (15,0,0). Indiqueu TOTS els paràmetres d'una càmera que permeti veure a la vista dos quadrats, un damunt de l'altre (el més gran a sota), de manera que ocupin el màxim de la vista (viewport). Cal que indiqueu la posició i orientació de la càmera especificant **VRP**, **OBS** i **up**.

19. (2007-2008 Q2) Disposem d'una càmera definida amb OBS=(10.0,0,0), VRP=(0,0,0), up=(0,1,0), ZN=3, ZF=6, FOV=60, ra=1 i un viewport de (1024,1024) píxels. Escriu el tros de codi OpenGL requerit per a definir la mateixa càmera (indicant el valor de tots els seus paràmetres). Per a la definició de la posició i orientació de la càmera utlitzeu transformacions geomètriques (i no la gluLookAt()).

20. (2007-2008 Q2) Tenim una escena amb dos cubs amb arestes alineades amb els eixos. Un d'ells està centrat a l'origen i és de costat 2, l'altre està centrat al punt C=(4,0,0) i és de costat 4. La posició i orientació de la càmera es defineix com: OBS=(10,0,0), VRP=(0,0,0) i up=(0,1,0). Tenim una càmera ortogonal amb window: left=-2, right=2, bottom=-2, top=2 i ZNear=6 i ZFar=12 i un viewport de 512 x 512. Sabem que el cub petit és de color (1,0,0) i el gran és de color (0,1,0), i que usem z-buffer, backface culling i mode "fill" per a la visualització de les cares. Indiqueu què es veurà en el viewport i de quin color.

21. (2007-2008 Q2P) Com ja sabeu, quan la finestra GLWidget pateix algun canvi de dimensions, s'han d'actualitzar alguns paràmetres de la càmera. Quins? Per què? Indiqueu les instruccions d'OpenGL que cal utilitzar.

22. (2007-2008 Q2P) Tenim una escena formada per tres cubs amb costats de mida 10, centrats en els punts C1=(0,0,10), C2=(0, 10, 0) i C3=(10, 0, 0). Quan es pinta l'escena en filferros volem veure en la vista tres quadrats situats en forma de L, i que el quadrat de la cantonada de la L quedi centrat a la vista. Indiqueu i justifiqueu la inicialització de tots els paràmetres d'una càmera que permeti obtenir la imatge descrita. Els paràmetres de posicionament cal indicar-los:
    - per a inicialitzar la càmera amb gluLookAt()
    - per a definir la MODELVIEW amb transformacions geomètriques.

23. (2007-2008 Q2P) Un estudiant proposa el següent codi per a inicialitzar la càmera utilitzant OpenGL i pintar un cub d'aresta 20 centrat a l'origen. Creieu que amb la seqüència d'instruccions indicada es pot obtenir la imatge requerida?

```
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
gluPerspective (65, 1, 20, 40);
gluLookAt(0,0,30,0,0,0,0,1,0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glPushMatrix();
glScalef(2,2,2);
PintaCub (0,0,0,l0) // pinta cub centrat a l'origen d'aresta 10
glPopMatrix();
```

24. (2008-2009 Q1) 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.

25. (2008-2009 Q1) 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.

26. (2008-2009 Q1P) Inicialitzeu, adequadament, tots els paràmetres d'una càmera perspectiva (definida amb gluLookAt i gluPerpective) de manera que el segment definit pels vértexs (3,0,0) i (6,0,0) es vegi sencer en el viewport com segment vertical que passa pel centre del viewport.

27. (2008-2009 Q1P) Escriviu un tros de codi (instruccions OpenGL) que utilitzant transformacions geomètriques defineixi la mateixa matriu ModelView que la requerida en l'exercici anterior. Justifiqueu la resposta.

28. (2008-2009 Q1P) S'està visualitzant un objecte amb una càmera perspectiva. A l'avançar la càmera cap a l'objecte (només es modifica la posició de l'observador), s'obté un efecte semblant al zoom. Passa el mateix amb una càmera axonomètrica? Per què?.

29. (2008-2009 Q1P) Donada la visualització d'una escena, quin és l'efecte que s'observa en la imatge obtinguda quan es fa un pan? Quins paràmetres de la càmera cal modificar per a fer un pan? Com s'han de modificar i per què?

30. (2008-2009 Q1P) Un arquitecte disposa d'una imatge en pantalla en la que veu la paret d'un edifici que li ocupa des del píxel (150, 100) al píxel (450, 300). Sap que la paret està completament perpendicular a la direcció de visió, i vol saber les mesures en metres d'amplada i alçada de la paret. Se sap que la vista (viewport) és de 600x400 píxels, i que s'ha usat una càmera axonomètrica definida amb uns paràmetres del window (en metros) de: left=-60, right=60, bottom=-60, top=60. Pot deduir quines són les dimensions de la paret que està veient? Si la resposta és afirmativa, calculeu aquestes dimenions. Justifiqueu la resposta.

31. (2008-2009 Q2) En les inicialitzacions prèvies al pintat d'una escena, suposa que tenim la següent seqüència d'instruccions OpenGL que defineix una càmera amb un window quadrat i un viewport també quadrat:

```
gluPerspective(myFovy, 1.0, myNear, myFar);
glViewport(0, 0, 400, 400);
```

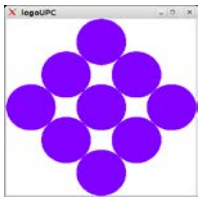
Quina diferència s'observaria en la visualització de l'escena si les canviem per:

```
gluPerspective(myFovy, 2.0, myNear, myFar);
glViewport (0, 0, 400, 400);
```

Com s'hauria de redefinir el viewport per tal que les imatges obtingudes siguin similars en ambdues visualitzacions? Raona la teva resposta.

32. (2008-2009 Q2P) Tenim una escena formada per un sol segment definit entre els punts (0,0,3) i (0,0,-3). Volem visualitzar la totalitat d'aquest segment en un viewport de 300x300 píxels usant una càmera perspectiva de manera que el segment travessi el viewport diagonalment des de la cantonada superior esquerra fins a la cantonada inferior dreta. Es demana: el tros de codi d'OpenGL requerit per a definir tots els paràmetres d'una càmera que produeixi la visualització indicada. La ubicació de la càmera s'ha de definir mitjançant transformacions geomètriques. Justifiqueu els procediment per decidir els valors escollits.

33. 2008-2009 2Q) La Degana de la FIB ens ha demanat que fem un logo de la UPC per computador, però amb el conjunt d'esferes girat 45 graus (veure figura). Totes les esferes tenen el seu centre ubicat en el pla XZ (és a dir, la seva y=0). Disposem d'una acció pinta_esfera(R) que pinta una esfera de radi R centrada en l'origen de coordenades. El viewport és quadrat i ocupa tota la finestra gràfica i, per tant, es defineix amb glViewport(0, 0, w, h). Defineix TOTS els paràmetres d'una càmera axonomètrica que permet generar la figura adjunta i indica el codi requerit per a generar l'imatge (definició de la càmera i pintat de la geometria). Dóna el codi per a definir la ModelView tant amb transformacions geomètriques com amb gluLookAt.



34. (2009-2010 Q1) Considereu la piràmide de la figura. Els seus vèrtexs es troben a les coordenades (0; 0; 0), (10; 0; 0), (0; 10; 0), (0; 0; 10).
    - Implementa una funció `pintaPiramide()` que faci les crides a OpenGL necessàries per a pintar aquesta piràmide. No cal fixar les propietats de material de l'objecte, però cal tenir en compte que la piràmide s'ha de veure correctament il·luminada si la il·luminació està activada.
    - Defineix TOTS els paràmetres d'una càmera ortogonal que quan es pinti la piràmide permeti veure a la vista un triangle equilàter amb la base horitzontal i de manera que es vegi centrat en un window de dimensions 20x16. Escriu el tros de codi OpenGL que defineix aquesta càmera usant transformacions geom_etriques per a definir la matriu GL_MODELVIEW
    - Quin podria ser un viewport que permeti veure aquest triangle sense deformació? Quina serà la seva relació d'aspecte?

35. (2009-2010 Q2) Un professor demana a dos estudiants que li escriguin el codi OpenGL necessari per a definir una càmera que mostra una certa escena en una vista en planta. Els estudiants responen amb els dos codis següents respectivament:

    Codi 1:

    ```
    1 glMatrixMode ( GL_PROJECTION );
    2 glLoadIdentity ();
    3 glOrtho ( -100 , 100 , -100, 100 , 10, 150);
    4 glMatrixMode ( GL_MODELVIEW );
    5 glLoadIdentity ();
    6 gluLookAt (0, 80, 0, 0, 50, 0, 1, 0, 0);
    ```

    Codi 2:

    ```
    1 glMatrixMode ( GL_PROJECTION );
    2 glLoadIdentity ();
    3 glOrtho ( -100 , 100 , -100, 100 , 10, 150);
    4 glMatrixMode ( GL_MODELVIEW );
    5 glLoadIdentity ();
    6 glTranslatef (0, 0, -80);
    7 glRotatef (90 , 0, 0, 1);
    8 glRotatef (90 , 1, 0, 0);
    9 glRotatef (-90, 0, 1, 0);
    ```

    Es demana:
    - Compleixen l'objectiu demanat cadascun d'aquests dos codis?
    - Defineixen tots dos la mateixa càmera?
    - Creus que les transformacions geomètriques del Codi 2 es poden optimitzar?

36. (2009-2010 2Q) Tenim una càmera ortogonal definida de la següent manera:

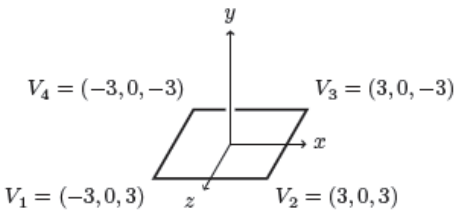
    ```
    glOrtho ( -1.0 , 1.0 , -1.0, 1.0 , 1.0 , 10.0)
    ```

    Asumint la mateixa posició de l'observador, quin és el mínim angle d'obertura vertical amb el qual s'ha de definir la següent càmera perspectiva, per tal que el volum de visió de la càmera ortogonal estigui totalment inclòs al seu volum de visió?

    ```
    gluPerspective (fovy ,1.0 , 1.0 , 10.0)
    ```

37. (2009-2010 2Q) Donat el tetraedre definit pels vèrtexs: V1=(0; 0; 0), V2=(10; 0; 0), V3=(0; 10; 0) i V4=(0; 0; 10), i pintant amb omplert de polígons, volem veure a la vista una imatge que ens mostra tres triangles iguals compartint un vèrtex que queda al mig de la vista. Defineix tots els paràmetres d'una càmera que ens permeti veure justament aquesta imatge a la vista, i escriu el tros de codi OpenGL que defineixi aquesta càmera.

38. (2009-2010 Q2P) Volem dibuixar el quadrat que es mostra a la figura, en una vista que té l'origen a (0; 0), i és de 800x600 píxels.
    - Dóna una elecció possible (concreta) de la posició de l'observador **OBS**, del view reference point **VRP**, i view up vector **VUP**, tals que en dibuixar aquest quadrat el punt V1 sigui sobre la mateixa recta vertical a la vista que passi per V3.
    - Dóna codi OpenGL per a posicionar i orientar la càmera definida a l'apartat anterior, usant transformacions geomètriques. No facis servir gluLookAt().
    - Dóna codi OpenGL que defineixi una matriu GL_PROJECTION que es pugui fer servir en conjunció amb la càmera que hagis definit als apartats anteriors de tal manera que el quadrat es dibuixi sense deformacions, i els vèrtexs V1 i V3 apareguin just a la vora de la vista.



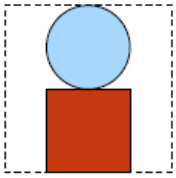
39. (2010-2011 Q1P) Una escena té *n* objectes; per a cada objecte es disposa tant de la seva capsa mínima contenidora com de la seva geometria (cares i vèrtexs) en coordenades de l'aplicació. Especifiqueu i justifiqueu els valors de TOTS els paràmetres (posició, orientació i tipus) que defineixen una càmera perspectiva que ha de permetre visualitzar el tercer objecte de l'escena, centrat i maximitzant la seva imatge en el *viewport*, sense retallar i sense tenir deformacions per la relació d'aspecte del *viewport* actiu. La posició i l'orientació de la càmera especifiqueu-la mitjançant transformacions geomètriques i també donant els valors d'**OBS**, **VPR** i vector **up** que permeten definir la mateixa càmera.

40. (2010-2011 Q1P) Considera el següent fragment de codi que defineix una càmera perspectiva:
    1. glMatrixMode(GL_PROJECTION);
    2. glLoadIdentity();
    3. gluPerspective(60, 1, zNear, zFar);
    4. glMatrixMode(GL_MODELVIEW);
    5. glLoadIdentity();
    6. glRotate(45, 1, 0, 0);
    7. glTranslate(0, 0, -2);
    8. glRotate(-90, 0, 1, 0);

    Indica i justifica els valors dels paràmetres **OBS**, **VRP** i **up** que hauries d'utilitzar per a definir la mateixa càmera amb gluLookAt(...).

41. (2010-2011 Q1) Desitjem inspeccionar un objecte situat en una posició qualsevol de l'espai; a tal efecte, volem obtenir "n" imatges seves ubicant una càmera ortogonal en l'equador d'una esfera contenidora de l'objecte (cada 360/n graus). Disposeu de les dades de la capsa contenidora de l'objecte i de l'acció PintaObjecte() que realitza les crides de pintat d'OpenGL. Indiqueu l'algorisme que permetria obtenir les imatges desitjades. Cal indicar els valors de TOTS els paràmetres de la càmera (o el codi OpenGL corresponent). El viewport és quadrat. Justiqueu els valors escollits.

42. (2010-2011 Q2) Tenim una escena formada per un cub de costat 4 centrat a l'origen de coordenades, i una esfera de radi 2 centrada al punt (-4, 2, 0). Tenint en compte que no usem il·luminació i els objectes es pinten amb color constant, indica TOTS els paràmetres d'una càmera axonomètrica per a què en la visualització de l'escena es vegi a la vista un quadrat amb un cercle al damunt centrats tots dos sobre la mateixa línia vertical, tal i com indica la figura. La vista és de 512x512 píxels.



43. (2010-2011 Q2) Donats els següents mètodes `pintaTriangle()` i `paintGL()`:

```
void GLWidget::pintaTriangle ()
{
glColor (1,0,0);
glBegin (GL_TRIANGLES);
glVertex3f (-1, -1, -dist);
glVertex3f (1, -1, -dist);
glVertex3f (0, 1, -dist);
glEnd();
}
void GLWidget::paintGL (void)
{
// Esborrem els buffers
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// Par_ametres de c_amera
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
glOrtho (-1, 1, -1, 1, dist-1, dist+1);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity ();
// (A)
glTranslatef (0, 0, -dist);
glRotatef (90, 0.0, 1.0, 0.0);
glTranslatef (0, 0, dist);
// (B)
}
```

Indica (o dibuixa) raonadament què es veuria en una vista de 600x600 píxels en els següents casos:

- Es fa una crida a `pintaTriangle()` en el punt (A) del codi del `paintGL()`
- Es fa una crida a `pintaTriangle()` en el punt (B) del codi del `paintGL()`

44. (2011-2012 1Q) L'esfera contenidora d'una escena està centrada al punt C=(5,5,0) i té radi R=3. Defineix TOTS els paràmetres d'una càmera axonomàtrica que permeti veure l'escena en planta, sencera, sense deformacions i ocupant el màxim d'una vista de 800x400 píxels.

45. (2011-2012 1Q) Tenim un triangle rectangle amb els vèrtexs V1=(2,0,-2), V2=(0,0,0) i V3=(4,0,4). Escriu el codi OpenGL que defineix una càmera perspectiva de manera que els vèrtex en Sistema de Coordenades de Dispositiu (SCD) en un viewport de 800x400 píxels siguin V1scd=(0,400), V2scd=(0,0) i V3scd=(800,0). Defineix els paràmetres de posició i orientació mitjançant transformacions geomètriques (no usis gluLookAt).

46. (2011-2012 1Q) Tenim el model geomètric d'un cub de costat 4, centrat en el punt (2,2,2) amb les cares paral·leles als plans de coordenades. Tenint en compte que no usem il·luminació i els objectes es pinten totes amb el mateix color (glColor(...)), indica TOTS els paràmetres d'una càmera axonomètrica que genera com imatge del cub un hexàgon regular. La vista és de 512x512 píxels. Justifica la resposta. Doneu dues inicialitzacions alternatives de la ModelView: una amb transformacions geomètriques i altre usant gluLookAt(..).

47. (2011-2012 2QP) Considera el següent fragment de codi que defineix una càmera perspectiva:
    1. glMatrixMode(GL_PROJECTION);

2. glLoadIdentity();
3. gluPerspective(60, 1, zNear, zFar);
4. glMatrixMode(GL_MODELVIEW);
5. glLoadIdentity();
6. glTranslate(0, 0, -10);
7. glRotate(90,0,0,1);
8. glRotate(-90, 0, 1, 0);

Indica i justifica els valors dels paràmetres **OBS**, **VRP** i **up** que hauries d'utilitzar per a definir la mateixa càmera amb gluLookAt(...).

48. (2011-2012 2QP) En la visualització d'una escena la posició i orientació d'una càmera s'ha definit mitjançant la instrucció gluLookAt(...);essent OBS=(10,0,0), VRP=(2,0,0) i up=(0,0,-1). Indica i justifica el fragment de codi OpenGL que es requereix per a obtenir la mateixa visualització de l'escena tot definint la transformació de càmera mitjançant transformacions geomètriques.

49. (2011-2012 2QP) En la una sessió extra de la pràctica d'IDI és demana inicialitzar una càmera perspectiva per aconseguir la vista en planta d'una escena tal que la seva esfera contenidora està centrada en (10,0,4) i té radi 20. L'escena no ha de resultar retallada, no ha d'haver deformacions i el *viewport* s'ha d'aprofitar raonablement (el màxim possible). Indica i justifica la inicialització de tots els paràmetres d'aquesta càmera, suposant que el *viewport* és de 600x400.

50. (2011-2012 2QP) En la pràctica d'IDI s'ha d'inicialitzar una càmera perspectiva que permet visualitzar tota l'escena maximitzant la seva grandària en el *viewport*. A tal efecte, els seus paràmetres és calculen a partir de l'esfera contenidora de l'escena. Suposant que aquesta es coneguda, indica i justifica la inicialització de tots els paràmetres d'aquesta càmera, suposant que el *viewport* és de 600x400.

51. (2011-2012 2Q) Donada una càmera en primera persona perspectiva i un *viewport* quadrat, quins dels seus paràmetres caldrà modificar en els següents casos:
    - S'avança cap al davant de la càmera (en la direcció de visió).
    - Es modifica la relació d'aspecte de la pantalla (és fa un resize).
    - Es vol realitzar un zoom.
    - S'inclina la càmera, mantenint la mateixa direcció de visió.

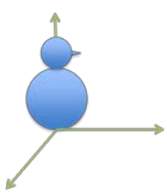
    Raona la resposta en base a la mateixa definició de la càmera en tots els casos, usant `gluPerspective()` i bé `gluLookAt()` o bé transformacions geomètriques.

52. (2011-2012 2Q) El següent tros de codi defineix la posició i orientació d'una càmera; quina seria la definició de la mateixa càmera utilitzant *gluLookAt(...)*?

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslate(0, 0, -2);
glRotate(90, 0, 1, 0);
glRotate(90, 0, 0, 1);
```

53. (2012-2013 1QPa) Donada l'escena de la figura, formada per: una esfera de radi 10 i centre (0,10,0), altre esfera de radi 5 i centre (0,25,0), i un con de base r=2 i llargada 5 orientat segons l'eix X, un estudiant ha inicialitzat la càmera per veure l'escena amb el codi següent:

```
glViewport (0,0,600,400)
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho(-15,15,-10,10,15,45);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0.,0.,-30.)
glRotatef (90.,0.,0.,1.)
glRotatef (-90.,0.,1.,0.)
```

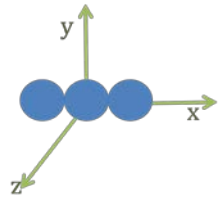


glTranslatef (0.,-15.,0.)

Indica i justifica:
a) La imatge que es veu a la vista.
b) La posició de la càmera amb gluLookAt(...).
c) Què passaria si és modifica la grandària de la finestres i el nou viewport queda definit com glViewport(0,0,600,600)? Com ho solucionaries?

54. (2012-2013 1Qb) Mostra la ubicació i orientació de la càmera respecte l'escena i dibuixa acuradament en quins llocs de la vista apareixen els Homers si dibuixem l'escena de la figura amb les inicialitzacions de càmera indicades en el següent codi. Observació: per a simplificar el dibuix, s'han dibuixat només les esferes mínimes contenidores dels Homers; totes tenen radi=1 i els seus centres estan a (0,1,0),(-2,1,0), (2,1,0)

```
glViewport (0,0,100,100);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
gluPerspective (60, 1., 3., 9.);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,7,0,0,0,0,1,0,1);
```

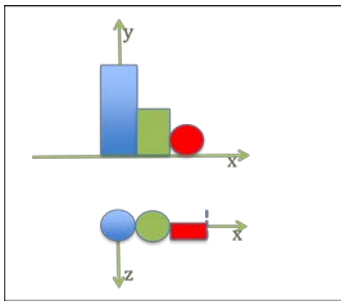


Quin efecte tindria canviar la relació d'aspecte de la càmera a 2? Com ho solucionaries sense modificar cap paràmetre de la càmera?

Indica i justifica TOTS els paràmetres d'una càmera perspectiva que, utilitzant també el codi de pintat de l'exercici anterior, genera una imatge en què el Homer ubicat a (2,0,0) queda centrat en el viewport, vertical i orientat cap a l'observador, maximitzant la seva ocupació i sense deformacions. Només ha de sortir a la imatge aquest Homer. El viewport és quadrat. Posiciona la càmera amb VRP, d i angles d'Euler.

55. (2012-2013 1Qc) En unes eleccions es presenten 3 partits que obtenen, respectivament, 50, 25, i 0 escons. 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. L'escena queda com a la figura adjunta. Si s'utilitza la càmera indicada en el següent codi per a pintar l'escena, què és mostraria en el viewport (vista)? Dibuixa i justifica el resultat per a aquesta càmera.

```
glViewport (0,0,100,100);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glOrtho (-1, +1., 0., 5.,2.,12.);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();
glTranslatef (0.,0.,-7.);
glRotatef (-90.,0.,1.,0.);
```



Modifica el codi anterior de manera que només es vegi el cilindre més alt, vertical en la pantalla i no hi hagi deformacions. No s'ha de modificar l'orientació ni la posició de la càmera.

Indica i justifica la posició i orientació d'una càmera amb OBS, VRP i UP que permetria veure vertical i centrat en el viewport el cilindre corresponent al partit que no ha obtingut cap escó. Suposem que l'òptica de la càmera està correctament definida.

56. (2012-2013 1Q) Tenim un rectangle centrat a l'origen de coordenades amb els costats paral·lels als eixos X i Y de l'aplicació i de mides 4cm d'ample i 3cm d'alçada. Tenim una càmera inicialitzada amb OBS=(0,0,5), VRP=(0,0,0), VUV=(0,1,0). El *viewport* és de 400 píxels

d'amplada i 600 d'alçada. Indiqueu els paràmetres amb què cal inicialitzar *gluPerspective()* de manera que encara que fem girar el rectangle respecte l'eix Z de l'aplicació sempre es visualitzi sense deformacions en el *viewport* i sense ser retallat.
Nota: la càmera estarà fixa, és a dir, encara que girem el rectangle NO modifiquem la càmera.

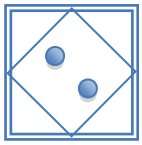
57. (2012-2013 2Q) Imaginem una escena formada per: un terra representat per un quadrat de 10x10 situat en y=0 i centrat en (0,0,0), i 2 arbres que tenen el centre de la base dels seus troncs en els punts (-2.5,0,0) i (2.5,0,0) respectivament, i una alçada de 5. Un gegant camina sobre el terra mirant 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:
    1. Cada cop que el gegant gira "fita" graus cap a la seva esquerra per modificar la seva direcció d'avançament.
    2. Cada cop que el gegant es mou "lambda" unitats en la seva direcció d'avançament.

58. (2012-2013 2Q) 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 57). Observació: ra_v és la relació d'aspecte del *viewport*.

    1. FOV=60º, ra=ra_v, zNear=0.1, zFar= 20
    2. FOV=60º, ra=ra_v, zNear=R, zFar= 3R; essent R el radi de l'esfera contenidora de l'escena.
    3. 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.**
    4. 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.**

59. (2012-2013 2Q) 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 57, una visualització similar a l'esquematitzada en la figura?

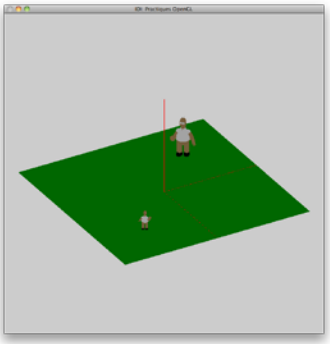
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0,0,-d);
glRotatef(fi,0,0,1);
glRotatef(fita,1,0,0);
glRotatef(psi,0,1,0);
glTranslatef(-VRPx, -VRPy, -VRPz);
pintaEscena();
```



    6. VRP=(0,0,0), d=6, psi=+45, fita=+90, fi=-90
    7. VRP=(0,0,0), d=6, psi=0, fita=+90, fi=+45
    8. VRP=(0,0,0), d=6, psi=+90, fita=0, fi= -45
    9. VRP=(0,0,0), d=6, psi=+45, fita=90, fi=45

62. (2013-2014 1Q). Es disposa d'una funció *pinta_escena* que envia a pintar la geometria que permet configurar una escena (veure figura) formada per:
    - un terra ubicat en el pla Y=0 amb centre (0,0,0) i de mides 10x10.
    - un homer d'alçada 1 amb el seu nas mirant cap l'eix Y i amb el centre de la base de la seva capsa en (-2.5,0,2.5). Altre homer amb el centre de la capsa de la seva base en (2.5,0.,-2.5), alçada 2 també mirant cap l'eix Y.

    S'observa l'escena amb una càmera posicionada amb VRP=(0,1,0), OBS=(20,1,20) i up=(0,1,0), i una òptica axonomètrica amb un window=(-7,5,7.5,-7.5,7.5), Znear=1 i Zfar=50. Contesta i raona les respostes següents:
    a) Si el viewport és de 600x600, dibuixa la imatge que es veuria (pots aproximar els "ninots" per cilindres i indicar l'orientació del seu nas amb una fletxa).
    b) Si en comptes d'una òptica axonomètrica, utilitzes una de perspectiva amb ra=1, els mateixos Znear i Zfar, i un angle FOV que permet veure el mateix tros d'escena, observaries alguna diferència en la image final?



62. (2013-2014 1Q). Una esfera de radi 1 es visualitza en un *viewport* quadrat de 400 per 400, amb una càmera posicionada correctament amb angles d'Euler, i on el mètode per a definir la projecció de la càmera utilitza la següent crida:

```
gluPerspective(60.0, 1.0, 1.0, 10.0);
```

L'usuari ha redimensionat la finestra a 500 d'amplada per 400 d'alçada. Digues què cal canviar de la càmera per tal que es vegi l'esfera correctament (sense retallar-la ni deformar-la).

a. Incrementar l'angle d'obertura vertical (FOV) i la relació d'aspecte del *window*.
b. Augmentar la relació d'aspecte del *window* i la distància al ZNear.
c. Només augmentar la relació d'aspecte del *window*.
d. Només canviar l'angle d'obertura vertical (FOV).

62. (2013-2014 1Q) Tenim una esfera de radi 1 centrada al punt (0.0, 0.0, 0.0) i una càmera amb una òptica definida com

```
glMatrixMode/(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, 0.0, 3.0);
      // left, right, bottom, top, znear, zfar
```

Si el mètode de pintat és:

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glColor3f(1.0, 0.0, 0.2);
glTranslatef(0.0, 0.0, -2.0);
glutSolidSphere(1.0, 20., 20.);
```

Què es veurà?

a. **La part superior dreta de l'esfera.**
b. La part superior de l'esfera.
c. La part inferior esquerra de l'esfera.
d. La part dreta de l'esfera.

63. (2013-2014 1Q) Quin dels següents codis creus que permetria definir la transformació de la càmera VRP=(0,1,0), OBS=(20,1,20) i up=(0,1,0) amb transformacions geomètriques?

a.
```
glLoadIdentity();
glTranslatef(0, 0, -20*sqrt(2.));
glRotated(45, 0, 0, 1);
glTranslatef(0, -1, 0);
```

b.
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0, 0, 20);
glRotated(-45, 0, 1, 0);
glTranslatef(0, -1, 0);
```

c.
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0, 0, -20);
glRotated(45, 0, 1, 0);
glTranslatef(0, -1, 0);
```

d.
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(0, 0, -20*sqrt(2.));
glRotated(-45, 0, 1, 0);
```

64. 2013-2014 1Q) Quan s'inicialitza la càmera, en quin ordre cal indicar les transformacions de càmera i el viewport a OpenGL?

    a. No importa l'ordre en què s'indiquen.
    b. Transformació de posició+orientació, transformació de projecció, viewport.
    c. La transformació de projecció, transformació de posició+orientació, viewport.
    d. Viewport, tansformació de projecció, transformació de posició+orientació.

65. (2013-2014 1Q) Una escena representa un mini sistema solar format pel sol, un planeta i un satèl·lit. Tots es representen 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. Suposant que es disposa d'un mètode pinta_escena(). 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.

66. (2013-2014 1Q) 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);

67. (2013-2014 1Q) 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

68. (2013-2014 1Q) 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.

69. (2013-2014 1Q) 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.

70. (2013-2014 1Q) Una escena està formada per dos avions. Un avió tindrà el centre de la seva capsa mínima contenidora en el (0,0,0) i estarà orientat cap l'eix X+; l'altre avió tindrà en centre de la seva capsa mínima contenidora en (0,0,120) i orientat cap l'eix Z^+. La llargada dels avions és 100 i de punta a punta de les ales 100.
Indica els paràmetres d'una càmera ortogonal (posició + orientació amb transformacions geomètriques, i òptica) i codi OpenGL que permeti veure només el primer dels dos avions en una vista que permeti veure tant el seu cilindre central com les seves "ales". Dibuixa la imatge resultant i justifica l'elecció de tots els paràmetres. El viewport és quadrat.

71. (2013-2014 1Q) Disposem d'una càmera axonomètrica amb els següents paràmetres: OBS=(0.,0.,0.), VRP=(-1.,0.,0.), up=(0.,1.,0.), window de (-5,-5) a (5,5), zn=5, zf=10. Indiqueu quin altre conjunt de paràmetres de càmera defineix exactament el mateix volum de visió (és a dir, garanteix generar exactament la mateixa imatge de l'escena):

a. **OBS= (1,0,0), VRP= (0,0,0), up=(0,2,0), zn= 6, zf=11**
b. OBS= (0,1,0), VRP=(0,0,0), up= (0,1,0), zn=5, zf=10
c. OBS= (0,0,0), VRP=(-2,0,0), up=(0,1,0), zn=6, zf=11
d. OBS= (-1,0,0), VRP=(0,0,0), up=(0,1,0), zn=-1, zf=9

72. (2013-2014 1Q) Tenim el següent codi que pinta una escena:

```
glViewport (0,0,600,600);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60., 1.0, 1.0, 100.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0., 0.,10,0,0,0,0,1,0);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glEnable (GL_DEPTH_TEST);
glColor3f(1,0,0);
glScaled(1.0, 5.0, 1.0);
glRotatef(-90, 1.0, 0.0, 0.0);
glRotatef(-45, 0.0, 1.0, 0.0);
glScaled(1.0, 5.0, 1.0);
glutSolidCube(1.0);
glRotatef(30., 0, 0, 1.);
```

Digues què es veu:

a. Un triangle vermell amb la seva base horitzontal al centre de la pantalla i la punta superior cap amunt.
b. Cap de les altres
c. Un rombe més ample que alt amb el seu centre situat al centre de la pantalla.
d. Un rombe més alt que ample amb el seu centre situat al centre de la pantalla.

73. (2013-2014 1Q) Quan es realitza la crida a glVertex3f(x,y,z), OpenGL realitza una sèrie de transformacions per a obtenir el píxel en què cal pintar-lo. Quina d'aquestes seqüències es correspon amb les transformacions que es fan?

a. Transformació de modelview, transformació de projecció, transformació window-viewport.
b. Transformació de projecció, transformació window-viewport, transformació de modelview.
c. Transformació de projecció, transformació de modelview, transformació de window-viewport.
d. Transformació de window-viewport, transformació de projecció, transformació de modelview.

74. (2013-2014 1Q) En les inicialitzacions prèvies al pintat d'una escena, tenim la següent seqüència d'instruccions OpenGL que defineix una càmera amb un window quadrat i un viewport també quadrat:

```
gluPerspective(myFovy, 1.0, myNear, myFar);
glViewport (0, 0, 400, 400);
```

quina diferència s'observaria en la visualització de l'escena si les canviem per:

```
gluPerspective (myFovy, 2.0, myNear, myFar);
```

```
glViewport (0, 0, 400, 400);
```

a. Cap perquè la relació d'aspecte de la càmera és superior a la del *viewport*, per això no cal modificar res més.
b. L'escena es veurà deformada amb el doble de llargada que amplada.
c. L'escena es veurà deformada amb el doble d'amplada que llargada.
d. Com no hem modificat FOV, es veurà retallada l'escena respecte l'inicial.

75. (2013-2014 2Q) 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 X^+. 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 l'altra (objecte 2) amb el centre de la base de la capsa al punt (5,0,5). Les capses dels dos objectes han de tenir alçada=2, amplada=1, fondària=1, i els seus davants s'han d'estar mirant.
Disposem també d'un mètode `pinta_escena()` que utilitzant `pinta_model()` permet generar l'escena.
Indiqueu i justifiqueu tots els paràmetres d'una càmera perspectiva ubicada en el centre de l'objecte 2 que mira cap al centre de l'objecte 1 i que permet veure l'objecte 1 optimitzant l'espai d'un viewport de 600x500 sense deformacions. La posició de la càmera l'heu de definir amb VRP, OBS i up. Observació: quan es visualitza l'escena utilitzant aquesta càmera no s'envia a pintar l'objecte 2. Podeu deixar indicats (però explicats) els càlculs que requeriu.

76. (2013-2014 2Q) Si pintem en una *viewport* de 400 per 400 amb el següent codi:

```
void codiPinta(void)
{
  glClearColor(0.0, 0.0, 0.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glDisable(GL_DEPTH_TEST);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-6.0, 6.0, -6.0, 6.0, 1.0, 10.);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glTranslatef(0.0, 0.0, -10.0);
  glRotatef(180., 0.0, 1.0, 0.0);
  glPushMatrix();
  glRotatef(-90.0, 1.0, 0.0, 0.0);
  glColor3f(0.0, 0.5, 1.0);
  glutSolidCone(3.0, 5.0, 20, 20); //radi,alçada, eix segons Z+
  glPopMatrix();
  glPushMatrix();
  glRotatef(90.0, 1.0, 0.0, 0.0);
  glColor3f(0.0, 0.5, 1.0);
  glutSolidCone(3.0, 5.0, 20, 20);
  glPopMatrix();
  glutSwapBuffers();
}
```

Què veurem?

a. Dos triangles un al costat de l'altre amb la punta cap amunt.
b. Dos triangles, un al costat de l'altre amb la base cap amunt.
c. Dos triangles, un mirant cap amunt i l'altre apuntant cap a la dreta.
d. Un rombe amb la seva diagonal més llarga paral·lela a l'eix vertical.

77. (2013-2014 2Q) Tenim una càmera orthogonal amb *window*=(-10, 10, -10, 10), zN=1 i ZF=30, correctament definits per veure tota una escena en un viewport de 400 per 400. L'usuari realitza un *resize* i el nou *viewport* és de 400 d'amplada per 600 d'alçada. Quins dels següents codis et sembla correcte per a no tenir deformacions i continuar veient tota l'escena?

`glMatrixMode (GL_PROJECTION);`
`glLoadIdentity();`
`float ra=400.0/600.0;`
`glOrtho (-10,10,-10/ra,10/ra,zN,ZF);`
	`glMatrixMode (GL_PROJECTION);`
	`glLoadIdentity();`
	`float ra=400.0/600.0;`
	`glOrtho (-10/ra,10/ra,10, 10, zN,ZF);`

`glMatrixMode (GL_PROJECTION);`
`glLoadIdentity();`
`float ra=400.0/600.0;`
`glOrtho(-10,10,-10*ra, 10*ra,zN,ZF);`
	`glMatrixMode (GL_PROJECTION);`
	`glLoadIdentity();`
	`float ra=400.0/600.0;`
	`glOrtho (-10*ra,10*ra,-10,10,zN,ZF);`

a. El codi de dalt a l'esquerra.
b. El codi de sota a l'esquerra.
c. El codi de dalt a la dreta.
d. El codi de sota a la dreta.

78. (2013-2014 2Q) Al final del codi del *resfresh*, ens hem oblidat de posar el `glutSwapBuffers()`. Si tenim activat pintar en doble *buffer*, el color de fons de pantalla està inicialitzat a vermell, i pintem una esfera de color verd que d'acord amb la càmera s'hauria de veure centrada en el *viewport*, quina imatge obtindrem?

a. Un cercle de color verd.
b. Un cercle de color groc.
c. No es veurà cap cercle.
d. Aniran alternant-se un cercle verd i el fons vermell, a mesura que girem l'esfera.

79. (2013-2014 2Q) En les inicialitzacions prèvies al pintat d'una escena, tenim la següent seqüència d'instruccions que defineixen una càmera i un *viewport*:

```
gluPerspective (myFovy, 2.0, myNear, myFar);
glViewport (0,0, 400,400);
```

Si la mida de la pantalla és de 800x400 i no es modifica, quina diferència s'observaria en la imatge que es visualitzarà si modifiquem el codi a:

```
gluPerspective (myFovy, 1.0, myNear, myFar);
glViewport (0,0, 200,200);
```

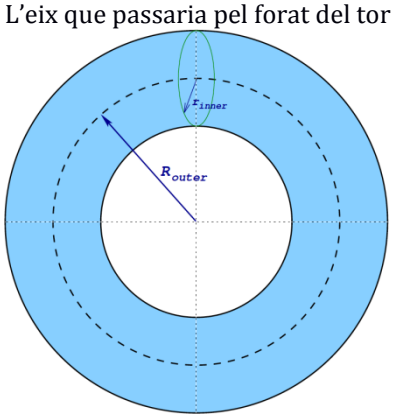
a. Com reduïm tant el *window* com el *viewport*, es veurà la mateixa imatge però ocupant un quart de la pantalla.
b. En la nova imatge es veurà l'escena sense deformacions però potser retallada en amplada respecte a la imatge anterior.
c. En la nova imatge es veurà l'escena sense deformacions però potser retallada en alçada respecte a la imatge anterior.
d. Com la pantalla té relació d'aspecte 2, en la imatge inicial es veia l'escena sense deformacions i ara es veurà deformada.

80. (2013-2014 2Q) Pintem una escena utilitzant el següent codi:

```
void pintaEscena(void)
{
  glClearColor(0.0, 1.0, 0.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-4.0, 4.0, -4.0, 4.0, -40.0, 40.0);
  glMatrixMode(GL_MODELVIEW);
```

```
    glLoadIdentity();
    glRotatef(angleY, 0.0, 1.0, 0.0);
    glRotatef(180.0, 0.0, 1.0, 0.0);
    glTranslatef(0, 0, -10);
    glColor3f(0.0, 0.5, 1.0);
    glutSolidTorus(2.0, 4.0, 20, 20);
    glutSwapBuffers();
}
```

On la crida *glutSolidTorus(GLdouble rInner, GLdouble rOuter, GLint nsides, GLint rings)* genera un tor (dònut) centrat a l'origen de coordenades amb el radi interior *rInne*r i el radi exterior *rOuter.* L'eix que passaria pel forat del tor és l'eix Z.



Què es veu?

a. Un cercle verd al centre de la pantalla rodejat de color blau.
b. Una circumferència centrada al centre de la pantalla i que és pràcticament tangent als quatre costats de la finestra.
c. Un cercle verd gairebé tangent als quatre costats de la finestra.
d. Dues circumferències concèntriques verdes centrades al centre de la pantalla.

81. (2013-2014 2Q) Disposem de la funció `pinta_model()` que encapsula les primitives gràfiques requerides per a pintar les cares d'un *legoman*. Els vèrtexs extrems de capsa mínima contenidora del model tenen coordenades (xmin,ymin,zmin) i (xmax,ymax,zmax), i el seu "davant" està orientat segons les Z^+.
Disposem de la funció `pinta_escena()`que, utilitzant el mètode `pinta_model()`, permet visualitzar una escena formada per dues instàncies del *legoman*: una (*legoman 1*) amb alçada=2, i l'amplada i la fondària igual a 1; el centre de la base de la seva capsa ha d'estar a l'origen de coordenades i el seu davant en la direcció de les Z^+. L'altra instància (*legoman 2*) tindrá una capsa el doble de gran que la del *legoman 1* i ubicada a sobre seu, però amb el seu "davant" mirant cap a Y^+, l'eix Y (de l'escena) passant pel seu centre i essent l'eix que travessa el *legoman 2* de cap a peus paral·lel a l'eix X (de l'escena).
Indiqueu i justifiqueu tots els paràmetres d'una càmera axonomètrica que permet veure a la pantalla només el *legoman 2*, centrat en la pantalla, vertical/dret, mirant cap a la càmera i optimitzant l'espai d'un viewport de 600x600 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.

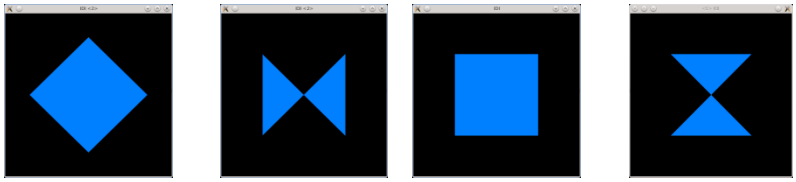
82. (2013-2014 2Q) Pintem una escena amb el següent codi:

```
glClearColor(0.0, 0.0, 0.0, 1.0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-2.0, 2.0, -2.0, 2.0, -10., 15.);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

```
glPushMatrix();
glTranslatef(0, -1, -10.);
glRotatef(-90.0, 1.0, 0.0, 0.0);
glColor3f(0.0, 0.5, 1.0);
glutSolidCone(1.0, 1.0, 20, 20);
glPopMatrix();

glPushMatrix();
glTranslatef(0, 1, -10.);
glRotatef(90.0, 1.0, 0.0, 0.0);
glutSolidCone(1.0, 1.0, 20, 20);
glPopMatrix();
```



Digues quina de les imatges següent es veu, essent el *viewport* quadrat.
Nota: glutSolidCone() pintaria un con amb radi de la base=1, alçada=1 i orientat segons l'eix Z+

a. La primera de la fila
b. La segona de la fila
c. La tercera de la fila
d. La quarta de la fila

83. (2013-2014 2Q) Tenim una càmera perspectiva amb els seus paràmetres correctament definits per veure tota una escena en un viewport de 400 per 600 sense deformació. Es vol realizar un *zoom-in* per a veure l'escena ampliada. Quins paràmetres de la càmera caldrà modificar?

a. Amb la configuració que tenim, únicament caldrà reduir el FOV, i no caldrà tocar cap paràmetre més.
b. Com la relació d'aspecte és menor que 1, caldrà incrementar FOV, i no caldrà tocar cap paràmetre més.
c. Com la relació d'aspecte és menor que 1, no és possible fer un *zoom-in* modificant FOV, es requereix també apropar l'observador a l'escena.
d. Serà necessari reduir el FOV i, segons la distància de l'observador al VRP, potser caldrà modificar zN.

84. (2013-2014 2Q) Dos estudiants inicialitzen la matriu de càmera de manera independent utilitzant cadascun un dels codis adjunts. La definició de la transformació de projecció la fan igual, tenen definit el mateix *Viewport* i l'escena que envien a pintar és la mateixa. (i ho fan correctament). Quina de les següents afirmacions és la correcta?

`glMatrixMode (GL_MODELVIEW);`
`glLoadIdentity();`
`//OBS,VRP,up`
`gluLookAt(10, 0, 0, 0, 0, 0, 0, 1, 0);`
 	
codi_1
	`glMatrixMode (GL_MODELVIEW);`
	`glLoadIdentity();`
	`glTranslated(0,0,-9);`
	`glRotated(180,0,0,1);`
	`glRotated(-90,0,1,0);`
	`glTranslated(-1,0,0);`
	codi_2

a. Per a obtenir la mateixa imatge, les transformacions de projecció (òptica) haurien de ser diferents pel que respecta a la definició de zN i ZF.
b. La imatge de l'estudiant del codi_1 és veurà capgirada en pantalla respecte a la del codi_2.

c. Obtindran la mateixa imatge. Amb els dos codis s'obté la mateixa posició relativa dels objectes respecte de la càmera.
d. No obtindran la mateixa imatge perquè un estudiant té el VRP en (0,0,0) i l'altre en (1,0,0).

85. (2013-2014 2Q) Un estudiant planteja el següent pseudocodi.

```
void refresh(){
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glEnable (GL_DEPTH_TEST);
  Set_càmera();
  glViewport (0,0,500,500);
  pinta_escena();
  Set_càmera();
  glViewport (500,500,500,500);
  pinta_escena();
  glutSwapBuffers();
}
```

Si la pantalla (finestra gràfica) és de 1000x1000, Quina afirmació és correcta?

a) **Es veuran dues imatges de l'escena a la pantalla, una abaix a l'esquerra i l'altre dalt a la dreta.**
b) Com no es pot modificar el *viewport* mentre s'està pintant l'escena, el resultat és indeterminat.
c) El segon *viewport* no està definit correctament. La segona imatge es projectarà en un píxel.
d) Si es vol pintar dues vegades l'escena, només es pot fer modificant les transformacions geomètriques que se li apliquen i pintant en un mateix *viewport*.

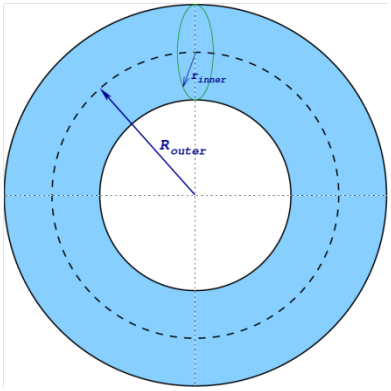
86. (2013-2014 2Q) Pintem en un *viewport* quadrat utilitzant el següent codi:

```
void pintaCodi(void)
{
  glClearColor(0.0, 0.0, 0.0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glViewport(0,0,1000,1000);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-2.0, 2.0, -1.0, 1.0, -10.0, 10.0);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  glPushMatrix();
  glRotatef(90.0, 1.0, 0.0, 0.0);
  glRotatef(90.0, 0.0, 1.0, 0.0);
  glPopMatrix();

  glColor3f(0.0, 0.5, 1.0);
  glutSolidTorus(0.25, 0.5, 20, 20);
  glutSwapBuffers();
}
```

On la crida *glutSolidTorus(GLdouble rInner, GLdouble rOuter, GLint nsides, GLint rings)* genera un tor (dònut) centrat a l'origen de coordenades amb el radi interior *rInne*r i el radi exterior *rOuter.* L'eix que passaria pel forat del tor és l'eix Z.



Què es veu?

a. Un cercle situat al centre de la pantalla.
b. Dues circumferències concèntriques centrades a la pantalla.
c. Un tor (dònut) al centre de la pantalla.
d**. Una espècie de dònut ovalat en vertical.**

87. (2013-2014 2Q) 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^+.
També disposem del mètode pinta_esceba() que permet 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 és un cub d'aresta 2; i l'altra (objecte 2) queda just damunt del primer, de la meitat de la mides i cap per abaix.
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.

88. (2013-2014 2Q) 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)

89. (2013-2014 2Q) Suposant que tenim una càmera axonomètrica amb la seva posició definida amb VRP, OBS i up. Quina de les següents afirmacions NO es correcta?

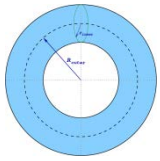
a. Si movem l'observador (OBS) avançant en la direcció de visió, la grandaria dels objectes no s'incrementa.
b. Com zN pot ser negatiu, no ens hem de preocupar d'actualitzar el VRP quan l'observador (OBS) avança en la direcció de visió.
c. Si l'observador gira per avançar en altre direcció, cal que modifiquem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

90 (2013-2014 2Q) 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).

91 (2013-2014 2Q) 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 òval ni circumferència.
d. Un rectangle vertical de color verd que atravessa la pantalla de dalt a baix.

Solucions

4. Serien: VRP = (10, 10, 10), OBS = (-10, 10, 10) i VUV = (0, 1, 0).
   Si ens fixem la TC que apliquem al punt (Matriu de ModelView) és: TC=T(0,0,-20)Gy(90)T(-10,-10,-10). Això comporta que per a ubicar la càmera s'haurien de composar les següens transformacions –locals a la càmera- : la traslladem al (10,10,10) (VRP) i després l'orientem fent un gir de –90 graus (cap a l'oest) respecte a un eix vertical (paral·lel a eix vertical de l'aplicació) que passa per VRP. Per tant, l'eix zobs quedarà amb l'orientació de –xapl. Posteriorment, allunyem la càmera 20 unitats del VRP en direcció de zobs (que com hem dit coincidirà amb –xapl); per tant, les coordenades de l'observador respecte el SC de l'aplicació seran
   OBS= VRP+(-20,0,0)=(-10,10,10).
   El VUV és (0,1,0) per que l'únic gir que fem a la càmera inicial que té l'eix vertical orientat segons (0,1,0) és respecte un eix vertical que no el modificarà.

7. El VRP el situarem al centre de la zona per a que aquesta quedi centrada en el viewport: **VRP** = **P.** Per a veure la zona més acuradament, situem la direcció de visió perpendicular a ella:
   OBS:= **VRP**+d***n**; podem posar una d=8 que estarà fora de la zona a inspeccionar (de gruix 2mm). Com a **VUV** podem agafar un vector que tingui la direccio d'una de les 4 fronteres de la zona. Aquella que volguem que surti vertical en pantalla. De fet serveix qualsevol vector que indiqui una direcció diferent a **n**.

   Suposant una càmera perspectiva:
   Zprop = d – 1; Zlluny = d + 1; per a no tallar la zona i que el frustum estigui ajustat a ella.
   Relació d'aspecte: ar= arv per a no tenir deformació (arv és la relació d'aspecte de la vista)
   Si volem veure vertical el costat de 4mm, requeriríem una window amb relació d'aspecte
   ar=7. /4. i un angle mínim d'obertura de la càmera de valor alfa=arc tg (2./d) (FOV=2*alfa).
   Com que la relació d'aspecte ar ha de ser la de la vista, hem de modificar, si s'escau, alfa, per a poder continuar veient tota la zona – sense deformació-,
   Si ar > arz llavors alfa := arc tg (2./d);
   sino altura := 7./(2.*ar); alfa := arc tg (altura/d);
   fisi

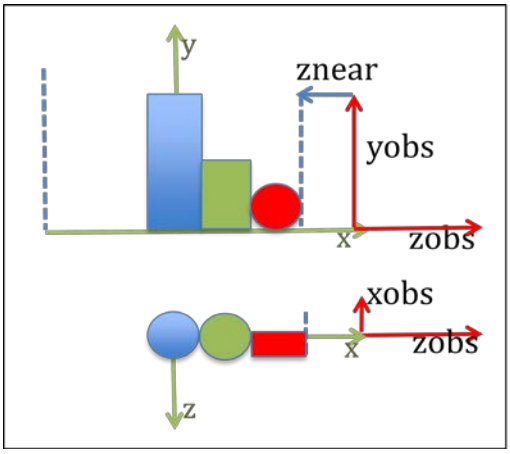
15. La matriu de projecció (TP) es construeix de la mateixa manera en tots dos codis i per tant és la mateixa.
   El posicionament de la càmera (matriu TC) es construeix de forma diferent en els dos codis:
   a) En el codi-1, l'observador es situa a la posició (0,10,0), el VRP a l'origen de coordenades i el vector up en la direcció negativa de l'eix Z, per tant estem definint un sistema de coordenades d'observador amb l'origen en el punt (0,10,0) i eixos amb la següent orientació: XO igual que XA ; YO la de -ZA; ZO la de YA .
   b) En el codi-2, les transformacions a fer al sistema de coordenades (a la càmera de referència) són: un gir sobre l'eix X de -90 graus seguit d'una translació de 10 unitats sobre l'eix Z resultant. El gir ens deixa el sistema de coordenades orientat tal que: X' = XA; Y' = -ZA; Z' = YA . Després la translació, símplement modifica la posició de la càmera sobre l'eix Z' (YA) aplicant la translació (0,0,10); de manera que l' origen del sistema de coordenades resultant estarà en (0,10,0). El resultat final de les dues transformacions ens defineix el mateix sistema de coordenades d'observador que el codi-1.

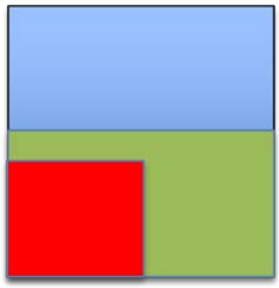
   Com que els dos codis ens situen la càmera al mateix lloc i orientada de la mateixa manera i amb la mateixa projecció, és evident que tots dos codis obtindran la mateixa imatge.

16. Usarem una càmera axonomètrica situada en la recta que passa pel punt central de la cara_i en direcció de la normal de la cara. La definició del window (paràmetres de la càmera axonomètrica) haurà de permetre que la cara quedi dins del window. Si es vol veure la cara en verdadera magnitud no pot haver deformació, per tant la relació d'aspecte ha de ser la mateixa que la del viewport, és a dir 1.
   Podem definir una càmera amb els paràmetres següents:
   OBS = C + dist * N,
   on C és el centre de la cara_i que es pot calcular com el punt mig de la seva diagonal i N és el vector normal a la cara_i.
   VRP = C,

VUV (vector up) = qualsevol vector contingut en el pla de la cara_i (per exemple en la direcció d'una aresta de la cara),
Left = -l/2, Right = l/2, Bottom = -l/2, Top = l/2
0 < Znear < dist, Zfar > dist (perquè només cal que garantim que es vegi la cara).

55. D'acord amb les TG indicades en el codi, per tal d'ubicar la càmera, a partir de la seva posició de defecte, cal efectuar les següents transformacions locals (sempre respecte al SC de la càmera): primer una rotació respecte de l'eix y de 90º i a continuació una translació segons l'eix z de la càmera de -7unitats. Per tant, s'està mirant l'escena des del punt (7,0,0) cap el (0,0,0) amb un up=(0,1,0). En la figura següent es mostra la posició de la càmera respecte a l'escena i la seva òptica. Com el zn=2 i zf=12, la profunditat de visió abarca des de X= 5 a X=-5; per tant abarca tots els cilindres. El window te una grandaria i posicionament (respecte la càmera) que emmarca el cilindre corresponent a 5 esccons (ocupa tot el window). Per tant, tots els cilindres estan en el camp de visió. La raw= 2/5=0.4 i la relació d'aspecte del viewport rav=100/100=1, per tant, hi haurà deformació al pintar en el viewport; com rav>raw, escalarem més les amplades que les llargades dels cilindres.





Imatge en el viewport

Només podem canviar l'optica de la càmera, per tant, podem modificar la profunditat de visió (znear i zfar) de manera que es retallin els altres cilindres. Si no volem que hi hagi deformació, hauren de fer que la relació d'aspect del window raw sigui la del viewport , o sigui raw=1. Si volem que es vegi tot el cilindre, haurem de fer el window de la mateixa amplada que la seva alçada, el quadrat més que englobi el window anterior.
Per tant, la nova càmera seria: glOrtho (-2.5, +2.5, 0, 5, 6, 8)

Per a veure el cilindre petit vertical en la pantalla, el vector up ha de tenir l'orientació de l'eix del cilindre: up=(0,0,1).
Per a veure el cilindre centrat, el VRP ha de'estar en el seu centre (4,1,0.5).
Tenim diferents possibilitats per ubicar l'observador. Una posibilitat és (4, 7, 0.5), és a dir, amb una direcció de visió segons (0,-1,0) –mirant el cilindre des d'alt-.