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. (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.

5. (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?

6. (2006-2007 Q1) Tenim una càmera perspectiva definida per OBS=(0., 50., 0.), VRP=(0., 0., 0.), **up**=(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).

7. (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.

---

[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.

- 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).

8. (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, up**, Zn i Zf valors diferents. Justifiqueu la resposta.

9. (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 viewMatrix la matriu identitat. No cal que inicialitzeu la `projectMatrix`.

10. (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 l'òptica perspectiva 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.

11. (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.

12. (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.

13. (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**.

14. (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 lookAt()).

15. (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.

16. (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.

17. (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 lookAt()
    - per a definir la viewMatrix amb transformacions geomètriques.

18. (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.

19. (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.

20. (2008-2009 Q1P) Inicialitzeu, adequadament, tots els paràmetres d'una càmera perspectiva (definida amb lookAt i perspective) 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.

21. (2008-2009 Q1P) Escriviu un tros de codi (instruccions OpenGL) que utilitzant transformacions geomètriques defineixi la mateixa matriu viewMatrix que la requerida en l'exercici anterior. Justifiqueu la resposta.

22. (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?.

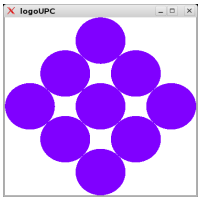
23. (2008-2009 Q1P) Donada la visualització d'una escena, quin és l'efecte que s'observa en la imatge obtinguda quan es fa un pan? Quins paràmetres de la càmera cal modificar per a fer un pan? Com s'han de modificar i per què?

24. (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.

25. (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.

26. 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 la imatge (definició de la càmera i pintat de la geometria). Dóna el codi per a definir la viewMatrix tant amb transformacions geomètriques com amb lookAt.

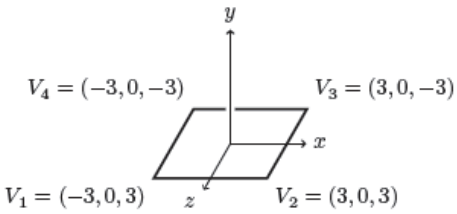


27. (2009-2010 Q1) Considereu una piràmide amb els seus vèrtexs a les coordenades (0; 0; 0), (10; 0; 0), (0; 10; 0), (0; 0; 10).
    - Implementa una funció `pintaPiramide()` que faci les crides a OpenGL necessàries per a pintar aquesta piràmide. No cal fixar les propietats de material de l'objecte, 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 geomètriques per a definir la matriu viewMatrix
    - Quin podria ser un viewport que permeti veure aquest triangle sense deformació? Quina serà la seva relació d'aspecte?

28. (2009-2010 2Q) Tenim una càmera ortogonal definida de la següent manera:
    Ortho (-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? perspective (fovy ,1.0 , 1.0 , 10.0)

29. (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; 0), 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.

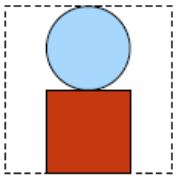
30. (2009-2010 Q2P) Volem dibuixar el quadrat que es mostra a la figura, en una vista que té l'origen a (0; 0), i és de 800x600 píxels.
    - Dóna una elecció possible (concreta) de la posició de l'observador **OBS**, del view reference point **VRP**, i view up vector **VUP**, tals que en dibuixar aquest 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 lookAt().
    - Dóna codi OpenGL que defineixi una matriu projectMatrix 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.



31. (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.

32. (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.

33. (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.



34. (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.

35. (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 lookAt).

36. (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 les cares es pinten totes amb el mateix color, 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 viewMatrix: una amb transformacions geomètriques i altre usant lookAt(..).

37. (2011-2012 2QP) En la visualització d'una escena la posició i orientació d'una càmera s'ha definit mitjançant la instrucció lookAt(...);essent OBS=(10,0,0), VRP=(2,0,0) i up=(0,0,-1). Indica i justifica el fragment de codi que es requereix per a obtenir la mateixa visualització de l'escena tot definint la transformació de càmera mitjançant transformacions geomètriques.

38. (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.

39. (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.

40. (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 `perspective()` i `lookAt()` o bé transformacions geomètriques.

41. (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 *perspective()* 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.

42. (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.

43. (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.**

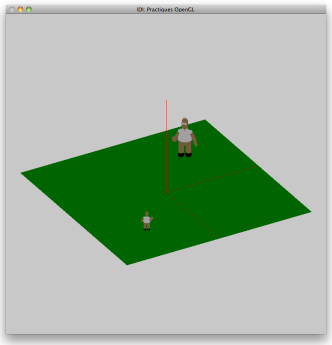
44 (2013-2014 1Q). Es disposa d'una funció *pinta_escena* que envia a pintar la geometria que permet configurar una escena (veure figura) formada per:

- 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?



45 (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:

```
perspective(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).

46 (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.

47 (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.

48 (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

49 (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.

50 (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.

51 (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

52 (2013-2014 1Q) Quan s'envia un vèrtex al procés de visualització, 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 viewmatrix, transformació de projecció, transformació window-viewport.
b. Transformació de projecció, transformació window-viewport, transformació de viewmatrix.
c. Transformació de projecció, transformació de viewmatrix , transformació de window-viewport.
d. Transformació de window-viewport, transformació de projecció, transformació de viewmatrix.

53 (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:

```
PM=perspective(myFovy, 1.0, myNear, myFar); projectMatrix(PM);
glViewport (0, 0, 400, 400);
```

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

```
PM=perspective (myFovy, 2.0, myNear, myFar); projectMatrix (PM);
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.

54 (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.

55. (2013-2014 2Q) En les inicialitzacions prèvies al pintat d'una escena, tenim la següent seqüència

```
PM=perspective (myFovy, 2.0, myNear, myFar); projectMatrix(PM);
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:

```
PM=perspective (myFovy, 1.0, myNear, myFar); projectMatrix(PM);
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.

56. (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.

57. (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.

58. (2013-2014 2Q) Un estudiant planteja el següent pseudocodi.

```
void paintGL(){
   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();
}
```

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*.

59. (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_escena() 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.

60. (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?

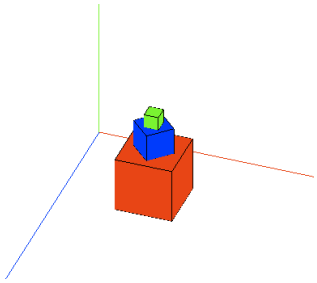
a. Si movem l'observador (OBS) avançant en la direcció de visió, la grandaria dels objectes no s'incrementa.
b. Com zN pot ser negatiu, no ens hem de preocupar d'actualitzar el VRP quan l'observador (OBS) avança en la direcció de visió.
c. Si l'observador gira per avançar en altre direcció, cal que modifiquem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

61 (2013-2014 2Q) La finestra gràfica (pantalla) és de 500x1000 píxels i es defineix una càmera axonomètrica amb `Ortho(-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).

62. (2014-2015 1Q) Una escena està formada per múltiples objectes i l'observador es mou per dins d'ella. L'òptica de la càmera pot ser perspectiva o ortogonal. Indica quina de les respostes següents és certa si es pinta l'escena amb el màxim realisme que sabem (mètodes empírics d'il·luminació i *depth-buffer*):
a. En les dues òptiques, en càmera amb primera persona, cal modificar el zNear de l'òptica a mesura que avancem si volem veure només els objectes que tenim davant.
b. Només cal modificar algun paràmetre de l'òptica si es fa un resize.
c. Si es fa un resize en l'òptica axonomètrica cal modificar zNear i window, mentre que en perspectiva només FOV.
d. En càmera perspectiva modificant el zNear i el zFar podem fer un efecte de zoom.

63 (2014-2015 1Q, P) Indica TOTS els parámetres d'una cámera axonométrica que en pintar l'escena que us indiquem cridant a la funció pinta_escena() en un viewport (vista) de 800x400, es vegin a la vista només les cares superiors de tots tres cubs, centrades, ocupant el màxim del viewport i sense deformació. Utilitza com a parámetres de posició i orientació OBS, VRP i Vup. Dibuixa la imatge que et quedarà a la vista i justica les respostes.
L'escena és un pilar de tres cubs de costats 2, 1 i 0.5 respectivament (no importa el color). El cub de baix de tot té el centre de la seva base al punt (3, 0, 3), i les seves cares estan orientades segons els plans coordenats. Tots tres cubs fan pila, és a dir, les cares corresponents es toquen i comparteixen el mateix eix vertical que passa pel seu centre.



64 (2014-2015 1Q P) Tenim una escena com la de la pregunta 93 però amb la base (de la pila de cubs) centrada en el punt (X, 0, Z). Volem una càmera que miri els cubs en una vista en planta (des de dalt) i els vegi centrats a la vista. Suposant l'òptica de la càmera ben definida, indica quin dels seguents trossos de codi et permetria definir amb transformacions geomètriques la posició i orientació de la càmera.

a) MV=Translate (0, 0, -5);
MV= MV*Rotate (-90, 0, 0, 1);
MV= MV*Rotate (90, 1, 0, 0);
MV=MV*Rotate (-90, 0, 1, 0);
MV=MV*Translate (X, 0, Z);
projectMatrix(MV);
	b) MV=Translate (0, 0, -5);
	MV=MV* Rotate (90, 1, 0, 0);
	MV=MV* Rotate (-90, 0, 1, 0);
	viewMatrix(MV);

c) MV=Translate (0, 0, -5);
MV= MV*Rotate (90, 0, 0, 1);
MV=MV*Rotate (-90, 1, 0, 0);
MV=MV*Rotate (-90, 0, 1, 0);
MV= MV*Translate (-X, 0, -Z);
viewMatrix(MV);
	d) MV=Translate(0, 0, -3);
	MV= MV*Rotate (90, 0, 0, 1);
	MV= MV*Rotate (90, 1, 0, 0);
	MV= MV*Translate (-X, -2, -Z);
	viewMatrix(MV);

65 (2014-2015 1QP) Tenim definida una càmera axonomètrica amb paràmetres: OBS = (0, -1, 0), VRP = (0, 0,0); Vup = (0, 0, 1), Window = (-2, 2, -2, 2), ZNear = 1, ZFar = 4, i una relació d'aspecte de la vista de 1 (rav = 1). Indica quina de les següents càmeres perspectiva seria adient per a definir un volum de visió que inclogui completament l'anterior:

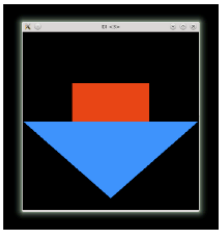
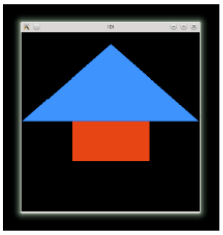
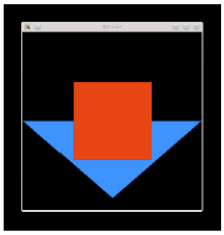
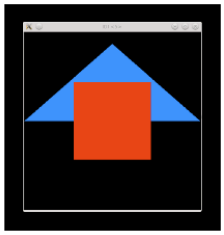
a) OBS=(0, -2, 0), VRP=(0, 2, 0), Vup=(0, 0, 1), FOV=90, ra=1, ZNear=2, ZFar=5.
b) OBS=(0, 2, 0), VRP=(0, 0, 0), Vup=(0, 1, 0), FOV=90, ra=1, ZNear=1, ZFar=4.
c) OBS=(0, -1, 0), VRP=(0, 0, 0), Vup=(0, 0, 1), FOV=90, ra=1, ZNear=1, ZFar=4.
d) OBS=(0, -2, 0), VRP=(0, 0, 0), Vup=(0, 0, 1), FOV=60, ra=1, ZNear=1, ZFar=4.

66 (2014-2015 1QP) Tenim definida correctament una càmera perspectiva que ens permet veure una escena completa sense retallar i sense deformació. Si a aquesta càmera li modifiquem el FOV, indica què caldrà modificar també si no volem que hi hagi deformació:
a) La relació d'aspecte de la vista (rav).
b) No cal modificar res més.
c) La relació d'aspecte del window (raw).
d) Depèn de si incrementem o decrementem el FOV.

67 (2014-2015 1QP) Pintem una escena amb el següent codi:

```
glViewport (0, 0, 800, 800);
glEnable (GL_DEPTH_TEST);
...
// Inici definició de càmera
PM= glm::Ortho (-5, 5, -5, 5, 5, 15);
projectMatrix(PM);
MV=Translate (0, 0, -10);
MV=MV*Rotate (180, 0, 0, 1);
MV=MV*Rotate(90, 1, 0, 0);
viewMatrix(MV);
// Fí definició de càmera
//color cyan
TG=I ; //I=matriu identitat
modelMatrix(TG);
SolidCone (5,5,20,20); // radi, alçada, orientació Z+, base centrada en (0,0,0)
TG=Translate (0, -2, 0);
 modelMatrix(TG);
 SolidCube (4); // costat 4, centrat a l'origen
```

Digues quina de les imatges següents es veu:

a) La primera imatge
b) La segona imatge
c) La tercera imatge
d) La quarta imatge

68. Una escena està formada per múltiples objectes i l'observador es mou per dins d'ella. L'òptica de la càmera pot ser perspectiva o ortogonal. Indica quina de les respostes següents és certa si es pinta l'escena amb el màxim realisme que sabem (mètodes empírics d'il·luminació i depth-buffer):

a. En les dues òptiques, en càmera amb primera persona, cal modificar el zNear de l'òptica a mesura que avancem si volem veure només els objectes que tenim davant.
b. Només cal modificar algun paràmetre de l'òptica si es fa un resize.
c. Si es fa un resize en l'òptica axonomètrica cal modificar zNear i window, mentre que en perspectiva només FOV.
d. En càmera perspectiva modificant el zNear i el zFar podem fer un efecte de zoom.

69 (2014-2015P 2Q). Es vol veure l'escena de l'exercici 30 de la col·lecció de TG de manera que la vaca amb el Patricio mirin cap a l'observador, quedin centrats en el viewport, sense retallar i optimitzant l'espai que ocupen en el viewport. Indica TOTS els paràmetres d'una càmera perspectiva (posició+orientació i òptica) que ho faria possible. El viewport _es de 600x400. Per facilitar els càlculs, en aquest exercici podeu considerar que les noves mides de les capses de la vaca i del Patricio un cop ubicats com toca són:

Patricio: dimensió en X = 1; dimensió en Y = 1; dimensió en Z = 0.3.
Vaca: dimensió en X = 1; dimensió en Y = 2; dimensió en Z = 4.

70 (2014-2015P 2Q) Per a visualitzar l'escena de l'exercici 1, un estudiant proposa la següent càmera: OBS=(20,0,0), VRP=(-9,0,0), up=(0,0,1) , ZN=15, ZF=40, FOV=90o i ra=1. Quan pinta l'escena no veu res en pantalla (malgrat que la rutina pintaEscena() és correcta). Quin paràmetre de la càmera creus que no és el correcte?
Considerem les mides de les capses de la vaca i del Patricio, un cop ubicats, iguals que en l'exercici anterior.
a) vector up
b) ZN
c) ra
d) VRP

71 (2014-2015P 2Q) Cal definir una càmera a OpenGL; quin dels següents pseudocodis és correcte? Noteu que tant sols canvia l'ordre en què es fan les crides.

1) VM=lookAt(OBS, VRP, up) viewMatrix (VM) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) glViewport(...) modelMatrix(TG) pintaescena()	2) modelMatrix(TG) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) VM=lookAt(OBS, VRP, up) viewMatrix (VM) glViewport(...) pintaescena()

3) VM=lookAt(OBS, VRP, up) viewMatrix (VM) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) modelMatrix(TG) glViewport(...) pintaescena()	4) glViewport(...) VM=lookAt(OBS, VRP, up) viewMatrix (VM) PM=perspective (FOV, ra, zn,zf) projectMatrix(PM) modelMatrix(TG) pintaescena()

a) nom_es 1) i 4) són correctes
b) només 4) és correcte
c) tots són correctes
d) tots són correctes menys 2)

72 (2014-2015P 2Q) Tenim una piràmide de base quadrada de costat 5, amb la base centrada al punt (0,0,2.5) i alçada de la piràmide 5 amb l'eix en direcció Z+. A l'escena tenim també un cub de costat 5 centrat a l'origen. El viewport està definit amb glViewport (0,0,400,800). Si a la vista es veu la imatge que teniu al dibuix (caseta), quines inicialitzacions d'una càmera axonomètrica (posició+orientació i òptica) permetrien veure aquesta imatge? Tots els angles estan en graus.

PM=perspective (90, 1, 5, 10); projectMatrix (PM) VM=translate (0,0,-10); VM=VM*rotate (90,1,0,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena ();	PM=ortho (-2.5, 2.5, -5, 5, 5, 10); projectMatrixPM) VM=translate (0,0,-7.5); VM=VM*rotate (-90,0,0,1); VM=VM*rotate (90,0,1,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena ();
PM=ortho (-2.5, 2.5, -5, 5, 5, 10); **projectMatrix (PM)** **VM=translate (0,0,-7.5);** **VM=VM*rotate (90,0,0,1);** **VM=VM*rotate (90,0,1,0);** **VM=VM*translate (0,0,-2.5);** **viewMatrix (VM);** **pinta_escena ();**	PM=ortho (-5, 5, -5, 5, 5, 10); projecMatrix (PM) VM=translate (0,0,-7.5); VM=VM*rotate (90,0,0,1); VM=VM*rotate (90,0,1,0); VM=VM*translate (0,0,-2.5); viewMatrix (VM); pinta_escena ();

73. (2014-2015P 2Q) Es vol realitzar una vista en planta (visió des de dalt) d'una escena/objecte que està centrat a l'origen amb una capsa contenidora de mides 10x10x10. Quina de les següents definicions et sembla correcta per definir la posició + orientació de la càmera (per a calcular la viewMatrix)? Sabem que la càmera és perspectiva i els angles de les rotacions estan en graus.

a) OBS = (0,10,0); VRP = (0,0,0); up = (0,1,0);
VM = lookAt (OBS, VRP, up);
viewMatrix(VM);
b) OBS = (0,0,0); VRP = (0,10,0); up = (0,0,-1);
VM = lookAt (OBS, VRP, up);
viewMatrix(VM);
c) VM = translate (0,0,-10);
VM = VM * rotate (90, 1,0,0);
viewMatrix(VM);
d) VM = translate (0,0,-10);

```
VM = VM * rotate (-90, 0,1,0);
viewMatrix(VM);
```

74. (2014-2015P 2Q) Una aplicació permet, prement la tecla 'o', permutar entre una càmera perspectiva i una axonomètrica, ambdues amb el mateix ZNear i ZFar. Totes dues càmeres veuen l'escena completa i sense deformacions. Un estudiant dubte de quina càmera és la que està activa en un cert moment, quin dels seguents experiments li aconsellaries fer per a deduir-ho?
a) Movent l'observador en la direcció del VRP, la perspectiva retallarà per culpa del ZNear i l'axonomètrica no.
b) Fent un resize de la finestra, la perspectiva deformarà i l'axonomètrica no.
c**) Movent l'observador en la direcció del VRP, la grandària de l'objecte no canviarà en l'axonomètrica, sí en la perspectiva.**
d) Girant la càmera en tercera persona (mitjançant angles d'Euler), l'axonomètrica retallarà i la perspectiva no.

75. (2014-2015 2Q) Volem simular una càmera (en primera persona) fixa en una moto de carreres per mostrar el que veu el pilot; el viewport sempre ocupa tota la pantalla. Indica quins són els paràmetres de la càmera que es poden mantenir fixes en tota la simulació (suposant que inicialment estaven correctament calculats):

a) FOV, ra i up

b) Up, znear, zfar, FOV i ra

c) znear, zfar, FOV i ra

d) tots els paràmetres poden requerir esser calculats en cada nova visualització

76. (2014-2015 2Q) En el procés de visualització projectiu, si implementem un vèrtex i fragment shader, quina de les següents respostes indica l'ordre correcte en què s'executen certes accions quan s'envia a pintar un triangle?
a) Pas a coordenades clipping, pas a coordenades de dispositiu, rasterització, z-buffer.

b) Pas a coordenades de dispositiu, retallat, rasterització, back-face culling.

c) Pas a coordenades de clipping, retallat, z-buffer, back-face culling.

d) Pas a coordenades observador, retallat, z-buffer, càlcul color per cada fragment.

77. (2015-2016P Q1) Tenim una escena amb un cub de costat 2 orientat amb els eixos i de manera que el seu vèrtex mínim està situat a l'origen de coordenades. La cara del cub que queda sobre el pla x=2 és de color vermell, la cara que queda sobre el pla z=2 és de color verd i la resta de cares són blaves.
a) Indica TOTS els paràmetres d'una càmera axonomètrica que permeti veure a la vista un rectangle centrat amb la meitat esquerra de color verd i la meitat dreta de color vermell. La relació d'aspecte del viewport (vista) és 2.
b) Quin efecte tindria en la imatge final modificar l'òptica a perspectiva? Pots indicar-ho utilitzant un dibuix si vols.

78. (2015-2016P Q1) Ordena de forma correcta els processos del pipeline de visualització projectiu d'OpenGL, és a dir, en quin ordre afecten aquests processos a la primitiva que s'envia a pintar:

a) 1) ProjectTransform; 2) ViewTransform; 3) ModelTransform; 4) Retallat;
b) 1) ModelTransform; 2) ViewTransform; 3) ProjectTransform; 4) Retallat;
c) 1) ModelTransform; 2) ViewTransform; 3) Retallat; 4) ProjectTransform;

d) 1) ViewTransform; 2) ModelTransform; 3) ProjectTransform; 4) Retallat;

79. (2015-2016P Q1) Tenim una càmera axonomètrica definida amb els paràmetres: OBS = (5,0,0), VRP = (0,0,0), up = (0,1,0), Window = (-2,2,-2,2), Znear = 2, Zfar = 8. Indica quins paràmetres definirien el mateix volum de visió considerant que l'observador passa a estar en OBS = (0,5,0). La visió de la imatge final no té perquè ser la mateixa i la relació d'aspecte del viewport no és rellevant.

a) VRP = (0,1,0), up = (0,0,1), Window = (-3,3,-2,2), Znear = 2, Zfar = 8.
b) VRP = (0,0,0), up = (1,0,0), Window = (-2,2,-2,2), Znear = 2, Zfar = 8.
c) VRP = (0,2,0), up = (0,0,1), Window = (-3,3,-2,2), Znear = 3, Zfar = 7.
d) VRP = (0,0,0), up = (0,0,-1), Window = (-2,2,-2,2), Znear = 3, Zfar = 7.

80. (2015-2016P Q1) Volem ubicar un model en una posició concreta d'una escena que es visualitza amb una càmera correctament definida. Tal i com indica el codi següent, hem passat al vèrtex shader com uniforms les matrius següents: TG que permet ubicar el model, View Matrix (VM) i Project Matrix (PM). Completa la instrucció que permet calcular les coordenades d'un vèrtex de l'objecte respecte el sistema de coordenades de l'observador (SCO).

```
in vec3 vertex;
uniform mat4 TG, VM, PM;
void main(){
vec4 vobs;
vobs =
...
}
```

a) vobs = TG*VM*PM*vec4(vertex,1.0);
b) vobs = VM*TG*vec4(vertex,1.0);
c) vobs = TG*VM*vec4(vertex,1.0);
d) vobs = VM*vec4(vertex,1.0);

81. (2015-2016P Q1) Tenim una càmera en primera persona correctament definida en posicionament i en òptica. El viewport és de 500x500. Quina de les següents afirmacions és correcta respecte a la relació d'aspecte (ra)?

a) S'ha de modificar ra sempre que és modifiqui el viewport sigui quin sigui el tipus de l'òptica.
b) S'ha de modificar ra només si es modifica el viewport i l'òptica és perspectiva.
c) S'ha de modificar ra si es modifica la finestra gràfica encara que el viewport no es modifiqui.
d) En la càmera en primera persona mai s'ha de modificar la relació d'aspecte de la càmera.

82. (2015-2016P Q1) Tenim una escena en la que utilitzem el codi següent per ubicar la càmera. Quins serien els paràmetres OBS, VRP i up que permetrien definir la mateixa càmera? (no modifiquem l'òptica).

```
VM = Translació(0,0,-10);
VM = VM*Rotacio_z (90);
VM = VM*Rotacio_y (-90);
VM = VM*Translació (10,-10,0)
ViewMatrix (VM);
```

a) OBS = (0,10,0), VRP = (10,10,0), up = (0,0,1)

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

83. (2015-2016P Q1) Quina de les següents afirmacions és incorrecta?
a) Si tenim una càmera axonomètrica i reduim el seu window (respectant la seva relació d'aspecte), estem fent un zoom-in.
b) Si incrementem el FOV de la càmera perspectiva, haurem d'incrementar la relació d'aspecte, per a què el window mantigui la seva proporció.
c) L'algorisme de retallat (clipping) és el mateix sigui la càmera perspectiva o axonomètrica.
d) L'eix Y del sistema de coordenades de l'observador (SCO) sempre es projecta vertical (direcció Y) en el sistema de coordenades de dispositiu (SCD).

84. (2015-2016 Q1) Tenim una escena formada per un "terra" de dimensions 10x10 ubicat en el pla Y=0 i centrat en el (0,0,0), i un objecte amb capsa contenidora de punt mínim (-2.5,0,-2.5) i punt màxim (2.5,10,2.5). Volem definir una càmera en tercera persona que mostri l'escena centrada en el viewport, quina de les següents afirmacions és correcta?

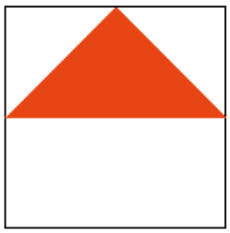
a) Es pot calcular la viewMatrix usant lookAt() amb el següent pseudocodi:
```
glm::vec3 OBS (0, 0, 20);
glm::vec3 VRP (0, 0, 0);
glm::vec3 up (0, 1, 0);
VM = lookAt (OBS, VRP, up);
```
b) No es pot definir el posicionament de la càmera si no sabem el tipus d'òptica.
c) Un possible càlcul de la viewMatrix (VM) seria amb el següent pseudocodi:
```
VM = Translate (0, 0, -20);
VM = VM * Rotate (30, 1, 0, 0);
VM = VM * Rotate (30, 0, 1, 0);
```
d) Cap de les altres respostes és correcta.

85 (2015-2016P Q2) Tenim una escena amb un triangle vermell amb vèrtexs V1=(2,0,0), V2 = (2, 0, 0) i V3= (0, 1, 0). Suposant que tenim un viewport quadrat de 600x600 píxels, indica TOTS els paràmetres d'una càmera ortogonal (axonomètrica) que permeti veure a la vista un triangle de manera que els seus vèrtexs en coordenades de dispositiu siguin: V1= (0, 0), V2=(600,0) i V3=(300, 600). Justifica la resposta.

86. (2015-2016P Q2) Donada la descripció de l'escena de l'exercici anterior i havent inicialitzat les matrius de càmera (view) i projecció (proj) a la matriu identitat, indica quina de les següents imatges és la que sortirà en un viewport de 600x600 (sabem que el Vertex Shader i el Fragment Shader estan correctament implementats):

a)

b)



c)

d)

87. (2015-2016P Q2) Suposant que tenim la matriu de transformació de model (TG), la matriu de canvi de punt de vista (view) i la matriu de projecció (proj), quina és la multiplicació correcta que transforma el vèrtex (vertex) a coordenades de clipping?

a) proj * TG * view * vertex.
b) vertex * proj * view * TG.
c) vertex * TG * view * proj.
d) proj * view * TG * vertex.

88. (2015-2016P Q2) Un estudiant defineix la seva càmera amb OBS, VRP i up. L'estudiant defineix d com la distància entre OBS i VRP i v com el vector normalitzat que va d'OBS a VRP, és a dir ha definit: VRP = OBS + d * v. En un cert moment, l'estudiant incrementa d i actualitza la view matrix però no la projection matrix, quin efecte tindrà en la visualització de l'escena?

a) Si no actualitza l'òptica, retallarà l'escena per Znear.
b) Veurà l'escena més petita, el punt d'enfoc està més lluny.
c) Veurà exactament el mateix.
d) Afectarà en la deformació perspectiva que observarà.

89. (2015-2016P Q2) Imaginem que tenim la escena formada per:

- Un prisma recte de base quadrada de costat 2 i alçada 5, situat amb el centre de la seva base al punt (0,0,0).
- Un Patricio d'alçada 2 situat al damunt de l'objecte anterior, és a dir, la base de la capsa del Patricio està sobre la cara superior del paral.lelepípede i mirant cap a les X+.

Volem pintar l'escena de manera que es vegi en 3a persona, sencera i centrada al viewport. Quina d'aquestes inicialitzacions dels paràmetres de càmera seria correcta?

a) VRP = (0, 3.5, 0), OBS = VRP + 2*radi esfera*v, amb v=(0,1,0)), up = (0, 1, 0).
b) VRP = ((Patxmin+Patxmax)/2, (Patymin+Patymax)/2, (Patzmin+Patzmax)/2), OBS = (VRP.x + 10, VRP.y, VRP.z), up = (0, 1, 0).
c) VRP = (1, 3.5, 0), OBS = (15, 3.5, 0), up = (0, 1, 0).
d) VRP = ((Patxmin+Patxmax)/2, Patymin, (Patzmin+Patzmax)/2)), OBS = (VRP.x, VRP.y, VRP.z + 10), up = (0, 1, 0).

90. (2015-2016P Q2) Tenim una escena en la que utilitzem una càmera amb OBS=(-5,0,3), VRP = (5,0,3) i up= (0,0,1). Quin conjunt de transformacions geomètriques permetrien definir la mateixa càmera, és a dir, generar la mateixa viewMatrix? (no modifiquem l'òptica).

a) VM = Translació(0,0,-10);
VM = VM*Rotacio_z (90);
VM = VM*Rotacio_y (-90);
VM = VM*Translació (-5,0,-3)
ViewMatrix (VM);

b) VM = Translació(0,0,-10);
VM = VM*Rotacio_z (90);
VM = VM*Rotacio_y (90);
VM = VM*Translació (-5,0,-3)
ViewMatrix (VM);

c) VM = Translació (-5,0,-3)
VM = VM*Rotacio_y (90);
VM = VM*Rotacio_z (90);
VM = VM*Translació(0,0,-10);
ViewMatrix (VM);

d) VM = Translació (-5,0,-3)

VM = VM*Rotacio_y (-90);
VM = VM*Rotacio_z (90);
VM = VM*Translació(0,0,-10);
ViewMatrix (VM);

91. (2015-2016P Q2) Respecte a la inspecció d'una escena amb càmera en primera i en tercera persona (tal i com s'han definit en l'assignatura), indica quina de les següents afirmacions és correcta:

a) Sempre han de ser amb òptica perspectiva, mai axonomètrica/ortogonal.

b) En la càmera en tercera persona, els paràmetres de l'òptica depenen dels de posició.

c) En ambdues càmeres, l'angle d'obertura és funció de la ra (relació d'aspecte) del viewport.

d) La View Matrix (view) en una càmera en 1a persona només es pot especificar mitjançant lookAt i en la de 3a persona mitjançant les transformacions geomètriques basades en els angles d'Euler.

92 (2016-2017P Q1) Es vol pintar una escena formada per dos Patricios situats formant un pilar. El primer Patricio (que li diem Pat1) ha de fer d'alçada 3 (un cop escalat uniformement) i es troba amb el centre de la base de la capsa contenidora al punt (5,0,0) i mirant cap a les Z+. I el segon Patricio (que li diem Pat2) ha de fer d'alçada 2 (també un cop escalat uniformement) i es troba amb el centre de la base de la seva capsa coincidint amb el centre de la cara del damunt de la capsadel primer Patricio (és a dir el petit "damunt" del gran) i mirant cap a les Z-. Pensant en aquesta escena, quins dels següents paràmetres d'una càmera axonomètrica serien adients per a poder veure en un viewport quadrat, únicament el Patricio de dalt del pilar (Pat2) i de manera que aquest estigui dret i de cara a la càmera?

a) OBS = (5,4,5); VRP = (5,4,0); up = (0,1,0);
left = -2; right = 2; bottom = -2; top = 2; Znear = 2; Zfar = 8;

b) OBS = (5,4,-5); VRP = (5,4,0); up = (0,1,0);
left = -2; right = 2; bottom = -2; top = 2; Znear = 2; Zfar = 8;

c) OBS = (5,4,-5); VRP = (5,4,1); up = (0,1,0);
left = -1; right = 1; bottom = -1; top = 1; Znear = 2; Zfar = 8;

d) OBS = (5,4,-5); VRP = (5,4,0); up = (1,0,0);
left = -1; right = 1; bottom = -1; top = 1; Znear = 3; Zfar = 7;

93 (2016-2017P Q1) Tenint en compte l'escena dels dos Patricios de l'exercici anterior i una càmera posicionada i orientada amb el VRP al centre de l'escena i l'observador a distància d = 6.0 del VRP en una certa direcció, indica quins paràmetres d'una òptica perspectiva serien adients per a veure l'escena (és a dir el pilar dels dos Patricios) sencera, sense deformar i optimitzant el viewport amb una càmera en tercera persona (per inspecció). Considereu que el radi de l'esfera contenidora és R = 3:0 i que el viewport és de 600x400 p__xels.

a) raw = 1.5; FOV = atan(tan(R/2)/raw); ZNear = 3; ZFar = 9;

b) FOV = 2*asin(R/6); raw = 1.5; ZNear = 3; ZFar = 9;

c) FOV = 2*asin(R/6); raw = 1; ZNear = 3; ZFar = 9;

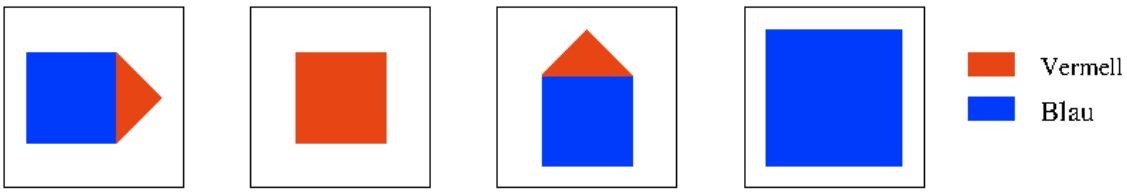
d) FOV = 2*atan(R/6); raw = 1.5; ZNear = 6; ZFar = 9;

94 (2016-2017P Q1) Tenim una escena amb una caseta formada per:

- un cub de color blau de costat 20 amb les cares paral_leles als plans coordenats i amb el centre de la seva cara inferior situat en el punt (10,0,0).
- una piràmide de color vermell de base quadrada d'aresta 20 i alçada 10, ubicada just a sobre del cub, amb la base de la piràmide coincidint amb la cara superior del cub.

Al pintar la caseta en un viewport quadrat amb els paràmetres de càmera:

OBS = (10,40,0); VRP=(10,-30,0); up=(1,0,0); FOV = 90º; ra = 1.0; Znear = 10; Zfar = 45;
Quina de les següents figures representa la imatge resultant?



a) b) c) d)

95. (2016-2017P Q1) Quin dels següents codis per a inicialitzar la viewMatrix generaria la mateixa matriu que la càmera descrita a la pregunta anterior?

a) VM = Translació(0,0,-40);
VM = VM*Rotacio_z (90);
VM = VM*Rotacio_x (90);
VM = VM*Translació (-10,0,0)
viewMatrix (VM);

b) VM = Translació(0,0,-40);
VM = VM*Rotacio_z (90);
VM = VM*Rotacio_y (90);
VM = VM*Translació (-10,0,0)
viewMatrix (VM);

c) VM = Translació (-10,30,0)
VM = VM*Rotacio_x (90);
VM = VM*Rotacio_z (90);
VM = VM*Translació(0,0,-70);
viewMatrix (VM);

d) VM = Translació (-10,0,0)
VM = VM*Rotacio_y (-90);
VM = VM*Rotacio_z (90);
VM = VM*Translació(0,0,-40);
viewMatrix (VM);

96. (2016-2017P Q1) Quin efecte tindrà en la imatge resultant si en la càmera descrita en l'exercici 94 canviéssim el vector up i poséssim up = (0,0,1)?

a) Sempre que modifiquem el vector up cal també modificar OBS.
b) Per poder veure la caseta hauríem de modificar també l'òptica de la càmera.
c) Veuríem una cara lateral del cub.
d) La imatge resultant seria idèntica a la de la pregunta 94.

97. (2016-2017P Q1) En l'escena de la caseta de la pregunta 94, en enviar-la a pintar amb una altra càmera, es veu la caseta allargada, és a dir, el cub, per exemple, es veu com un prisma amb la base el doble d'amplada que l'alçada. Què és el que ho provoca?

a) La ra del viewport (rav) és < 1 i no s'ha modificat el FOV.
b) La ra del window (raw) i la del viewport (rav) no són iguals.
c) La ra del window (raw) és > 1 i la del viewport (rav) és 1.
d) La ra del window (raw) és > 1 i la del viewport (rav) és > 1.

98. (2016-2017P Q1) Un estudiant implementa el Vertex Shader oblidant-se de multiplicar els vèrtexs per la viewMatrix, és a dir, el càlcul del gl Position el fa fent:

gl_Position = proj * TG * vec4(vertex, 1.0);

on TG és la modelMatrix i proj és la projectMatrix d'una òptica perspectiva. Quina afirmació és la correcta?

a) El volum de visió queda definit pels punts (-1,-1,-1) i (1,1,1) en coordenades de l'aplicació (SCA).
b) Té el mateix efecte que tenir OBS = (0,0,0), VRP = (0,-10,0) i up = (0,1,0).
c) No podem conèixer la posició de la càmera.
d) Cap de les altres respostes és correcta.

99. (2016-2017P Q1) Per a què el procés de visualització funcioni correctament pel que respecta a la projecció de la geometria de l'escena, hem de programar obligatòriament els VS (Vertex Shader) i FS (Fragment Shader) de manera que:
a) En el VS es calculin les coordenades de clipping del vèrtex i en el FS les de dipositiu.
b) En el VS es calculin les coordenades d'observador del vèrtex i en el FS les de clipping.
c) En el VS es calculin les coordenades de clipping del vèrtex.
d) En el VS es calculin les coordenades de clipping i en el FS les d'observador.

100. (2016-2017P Q1) Dos estudiants discuteixen respecte a la implementació del zoom amb òptica axonomètrica (ortogonal) i perspectiva. Quina de les seves afirmacions és certa?
a) En òptica ortogonal només es pot obtenir un efecte de zoom modificant OBS i VRP en la direcció de visió.
b) En òptica perspectiva cal modificar FOV, Znear i Zfar.
c) En les dues òptiques es pot fer zoom modificant el window de la càmera.
d) En òptica perspectica si avancem OBS i VRP en la direcció de visió cal anar amb compte amb la ra.

101. (2016-2017 Q1) Suposem que tenim una escena centrada en el punt (10,15,0) i que la seva capsa contenidora té mides 20, 30, 20 en X, Y i Z respectivament. L'escena està pintada en un viewport quadrat usant una càmera inicialitzada amb les matrius que descriu el següent codi:

```
PM = ortho (-15, 15, -15, 15, 10, 30);
projectMatrix (PM);
VM = Translació (0,0,-20)
VM = VM*Rotacio_z (-90);
VM = VM*Rotacio_y (90);
VM = VM*Translació(-10,-15,0);
viewMatrix (VM);
```

Quins paràmetres OBS, VRP i up definirien exactament la mateixa View Matrix?
a) OBS = (-10,15,0); VRP = (10,15,0); up = (0,0,1);
b) OBS = (-10,15,0); VRP = (10,15,0); up = (0,0,-1);
c) OBS = (10,15,0); VRP = (0,15,0); up = (0,0,-1);
d) OBS = (10,15,20); VRP = (10,15,0); up = (1,0,0);

102. (2017-2018P Q2) Tenim una escena formada per tres cubs aliniats amb els eixos coordenats, 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 amb una càmera ortogonal 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. Quina de les següents càmeres aconseguiria aquest objectiu suposant que l'òptica està correctament definida?
a) VM=I; VM=VM*Translate (0,0,-20); VM= VM*Rx(90); VM=VM*Translate (-5,-5,-5)
b) VM=I; VM=VM*Translate (0,0,-30); VM= VM*Rz(90);
c) OBS= (0,0,20), VRP=(0,0,0), up=(0,1,0)
d) OBS=(0,50,0), VRP= (0,0,0), up=(0,0,1)

103. (2017-2018P Q2) Dos estudiants estan comentant què cal modificar de l'òptica d'una càmera en tercera persona quan es fa el resize en òptica ortogonal i en perspectiva per a què no hi hagi deformació ni retallat de l'escena. El que diuen respectivament és:
Estudiant A: en les dues òptiques cal modificar el window modificant els paràmetres adequats de cada òptica.
Estudiant B: en l'òptica ortogonal cal modificar el window i en la perspectiva raw i segons el valor de rav també el FOV.
a) Els dos estudiants tenen raó.
b) Només l'estudiant B ho farà correctament.
c) Només ho farà correctament l'estudiant A.
d) Cap dels dos estudiants té raó.

104. (2017-2018P Q2) Considera el següent fragment de codi que defineix una càmera perspectiva (els angles
estan en graus):

```
1. PM = perspective (60, 1, zNear, zFar);
2. projectMatrix (PM);
3. VM = I;
4. VM = VM * Translate (0, 0, -2);
5. VM = VM * Rotate (-90, 0, 1, 0);
6. VM = VM * Translate (-1, -1, -1);
7. viewMatrix (VM);
```

Indica i justifica els valors dels paràmetres OBS, VRP i up que hauries d'utilitzar per a definir la mateixa càmera amb LookAt(...); _es a dir per obtenir la mateixa imatge de l'escena tot suposant que no es modifica l'òptica.
a) OBS=(0,0,0); VRP= (0,0,2), up=(0,1,0)
b) OBS= (3,1,0), VRP= (1,1,1), up=(0,2,0)
c) OBS= (1,1,2), VRP=(1,1,1), up =(1,0,0)
d) OBS=(3, 1,1), VRP=(1,1,1), up=(0,2,0)

105. (2017-2018P Q2) Per a què el procés de visualització funcioni correctament en el Vertex Shader cal assignar a la variable gl position:
a) Les coordenades del vèrtex en el VBO.
b) Les coordenades de clipping del vèrtex.
c) La posició del vèrtex en el sistema de coordenades de l'escena.
d) Les tres coordenades del vèrtex en el VBO ampliades amb una quarta que val 1.

106. (2017-2018P Q2) Quina afirmació és certa respecte l'òptica ortogonal?
a) L'algorisme de clipping funciona igual que amb càmera perspectiva.
b) Per a realitzar un zoom cal modificar znear i zfar.
c) No es pot utilitzar si es té activat el back-face culling.
d) No es pot utilitzar en càmera en primera persona.

107. (2017-2018P Q2) Donada una càmera perspectiva en primera persona, un estudiant implementa la funcionalitat d'avançar l'observador en la dirección de visió però no modifica VRP, ni cap altre paràmetre de l'òptica perspectiva.
a) Si no modifica VRP hauria de controlar el valor de znear, perqu_e en algun moment l'ha de canviar de signe per seguir avançant correctament.
b) Si no modifica VRP, hauria de modificar també FOV o la deformació perspectiva que tindrà serà molt rara.
c) Si en algun moment modifica el valor d'up, pot arribar a avançaar en una direcció diferent.
d) Pot arribar a mirar en la direcció contrària a la que avaça.

108. (2017-2018P Q2) Suposem que tenim una escena en què la seva esfera contenidora està centrada a l'origen i té radi 50. Inicialment tenim la següent definició de càmera: FOV=60 (està en graus), ra=1, znear=50, zfar=150 i que la viewMatrix s'inicialitza:

```
VM = I;
VM = VM * Translate (0,0,-100);
viewMatrix (VM);
```

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

a) FOV = 30.
b) VM = VM * Translate (0,0,-75) i zNear = 25.
c) La inicialització de la view matrix és incorrecta.
d) zNear = 25.

109. (2017-2018P Q2) 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 (Euler). Digues quina de les combinacions següents de crides a rotacions i translacions conté les crides necessàries i està en l'ordre correcte en el codi:

a) Rotació respecte X; rotació respecte Z; rotació respecte Y; translació -VRP;
b) Rotació respecte Z; rotació respecte Y; rotació respecte X; translació -VRP; 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;

110. (2017-2018 Q2) Tenim una escena formada per un cub vermell de costat 2 orientat amb els eixos i amb el seu vèrtex de coordenades mínimes situat a l'origen de coordenades. Es visualitza sense il·luminació i amb una càmera ortogonal situada al punt OBS = (4, 1, 4), mirant cap al punt VRP = (0, 1, 0) i amb vector up = (1, 0, -1). Els paràmetres de l'òptica són: left = -2, right = 2, bottom = -2, top = 2, znear = 2 i zfar = 10. Si estem veient l'escena en un viewport de 600 x 600, què veurem en la imatge final?

A. Veurem un rectangle més ample que alt.
B. Veurem un hexàgon més alt que ample.
C. Veurem un rectangle més alt que ample.
D. Veurem un quadrat perquè el window és quadrat.

111. (2017-2018 Q2) Tenint en compte el procés de visualització d'OpenGL, indica quin dels següents és l'ordre correcte en què es realitzen els processos indicats:

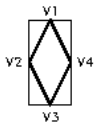
A. View Transform – Project Transform – Z-buffer – Back-face culling
B. Project Transform – View Transfom – Z-buffer – Rasterització
C. Project Transform – View Transform – Clipping – Rasterització
D. View Transform – Project Transform – Rasterització – Z-buffer

112. (2018-2019P Q1) En la següent llista teniu etapes/tasques del procés de visualització d'OpenGL ordenades per ordre alfabètic. Escriu-les de nou refent l'ordre per a què sigui l'ordre en què es realitzen en el procés de visualització d'OpenGL.

Clipping
Fragment shader
Pas de coordenades del vèrtex a SCO
Rasterització

113. (2018-2019P Q1) Pintem un quadrilàter amb vèrtexs V1=(3,0,0), V2=(0,0,-2), V3=(-3,0,0) i V4=(0,0,2).

a) Acaba d'omplir els paràmetres que falten d'una càmera ortogonal que en un viewport de 400x800 mostra la imatge següent:



VM = lookAt ((0,10,0), (0,0,0),);
PM = ortho (, , , , 8, 12);

Solució:
VM = lookAt ((0,10,0), (0,0,0), (1,0,0);
PM = ortho (-2, 2, -3, 3, 8, 12);

b) Quina és la relació d'aspecte del window (raw) de la càmera?
Solució: raw = 2/3

114. (2018-2019P Q1) Una escena formada per tres objectes es pinta usant el mètode següent:

```
1- pinta_Escena () {
2- modelMatrix (TG1);
3- pinta_obj1 ();
4- modelMatrix (TG2);
5- pinta_obj2 ();
6- modelMatrix (TG3);
7- pinta_obj3 ();
8- }
```

Suposant que TG1, TG2 i TG3 són variables globals que contenen les matrius de transformació que necessita cada objecte respectivament i que la crida modelMatrix(TG) només envia la matriu a la tarja gràfica, què afegiries i on en aquesta rutina pinta Escena() per a que, donat un cert vector de translació (tx,ty,tz), en coordenades de l'aplicació, tota l'escena es traslladi segons aquest vector de translació?

115. (2018-2019P Q1) Donada una càmera perspectiva en primera persona, un estudiant implementa la funcionalitat. d'avançar l'observador en la direcció de visió i observa que després d'haver avançat uns quants cops, l'observador mira en sentit contrari al qual avança, la qual cosa és clarament un error.
a) Això no pot passar, deu tenir un error en la definicó de l'escena.
b) Deu haver fet un resize i no ha modificat la relació d'aspecte (ra) del window apropiadament i li retalla.
c) Deu haver modificat la inclinació (up) de la càmera (tot mantenint la direcció de visió).
d) Deu haver modificat OBS per avançar però no VRP.

116. (2018-2019P Q1) Indica quina de les següents inicialitzacions permet obtenir una visualització en planta (projecció en el pla X-Z de l'aplicació) d'una escena de la qual sabem que la seva esfera contenidora està centrada a l'origen i té radi 20.
a) VM=I ; VM= VM*Gx(90); VM=VM*Translació (0,0,-30)
b) VM=I ; VM=VM*Translació (0,0,-30); VM= VM*Gy(90)
c) OBS= (0,50,0), VRP=(0,10,0), up=(0,0,1)
d) OBS=(0,50,0), VRP= (0,0,0), up=(0,1,0)

117. (2018-2019P Q1) Definint la viewMatrix utilitzant transformacions geomètriques a partir dels angles d'Euler:
a) No es pot reproduir l'efecte de modificar el vector up.
b) L'òptica obligatòriament ha de ser perspectiva.
c) Aquesta matriu no podrà ser la identitat.
d) Podem obtenir la mateixa visualització de l'escena que si s'ha obtingut a partir de lookAt.

118. (2018-2019P Q1) En el procés de visualització, l'algorisme de retallat implementat per OpenGL (clipping) per a triangles que es pintaran omplerts de color...
    a) és el mateix algorisme tant si la càmera és perspectiva com si és ortogonal.
    b) és funcio de si el tipus de càmera és perspectiva o és ortogonal.
    c) és funció de si el volum de visió és un tros de piràmide o un paral_lelepípede.
    d) és funció de si les coordenades dels vèrtexs a la sortida del vèrtex shader (gl Position) están en coordenades de clipping o no.

119. (2018-2019P Q1) Tenim una funció pinta cub que envia a pintar un cub d'aresta 1 centrat a l'origen de coordenades i una càmera ortogonal que permet visualitzar-ho correctament. Volem que l'usuari, mitjançant una tecla, pugui decidir si vol que el cub es vegi escalat al doble de gran o no. Com ho implementem?
    a) Podem enviar un uniform al FS (amb valors 1 o 2) i, en cas que tingui valor 2, desplaçar els fragments en el viewport per ocupar més espai.
    b) Sense modificar la posició de la càmera no es pot aconseguir cap escalat.
    c) Podem enviar un uniform al VS (amb valors 1 o 2) i que es multipliquin les coordenades dels vèrtexs pel uniform.
    d) Es pot aconseguir aquest efecte reprogramant tant el VS com el FS indistintament.

120. (2018-2019P Q1) Es vol modificar una aplicació per a què cada crida a paintGL pinti la mateixa escena en dos viewports diferents. L'escena es pinta mitjançant una crida pinta escena(). Els dos viewports defineixen respectivament el quadrant inferior esquerre i el quadrant superior dret de la finestra gràfica, que és de 800x600 píxels, tots dos tenen la mateixa relació d'aspecte. La posició i orientació de la càmera no es volen modificar. Per a fer això caldrà fer canvis a paintGL:
    a) Hem d'afegir una segona crida a pinta escena(), precedint cada crida a pinta escena() de la definició del viewport corresponent.
    b) Tant sols caldrà modificar la definició del viewport, la raw i la projectMatrix abans d'una segona crida a pinta escena() que afegirem.
    c) Tant sols caldrà fer dos crides a glViewport, sense necessitat d'afegir res més.
    d) Afegirem una segona crida a pinta escena(), però redefinirem la projectMatrix abans de cadascuna de les dues crides a aquesta funció.

121. (2018-2019 Q1) Una escena està formada per un cub de costat 2 centrat a l'origen i una esfera de radi 1 amb el seu centre al punt (0, 0, -2). Tenim un viewport de 400x800 (amplada x alçada) i volem que en el viewport es vegi un quadrat amb un cercle al damunt. Quina de les següents definicions de càmera usaries?

    A. OBS = (0, -3, -1); VRP = (0, -2, -1); up = (1, 0, 0); left = -2; right = 2; bottom = -2; top = 2;
    B. OBS = (0, 3, -1); VRP = (0, 0, -1); up = (0, 0, 1); left = -1; right = 1; bottom = -2; top = 2;
    C. OBS = (3, 0, -1); VRP = (0, 0, -1); up = (0, 0, -1); left = -1; right = 1; bottom = -2; top = 2;
    D. OBS = (-3, 0, -1); VRP = (-2, 0, -1); up = (0, 0, -1); left = -2; right = 2; bottom = -2; top = 2;

122. (2018-2019 Q1) En OpenGL es poden habilitar dos mètodes d'eliminació de parts amagades: Depth-buffer i back-face culling. Tenim una escena formada per dos cubs d'aresta 2 amb cares paral·leles als plans de coordenades i centres en (0,0,0) i (3,0,0). Un observador inspecciona l'escena amb una càmera en 3ra persona correctament definida.
    A. Les cares visibles que observarà seran les mateixes estiguin activats els dos o només depthbuffer.
    B. Com només hi ha dos objectes convexos i la càmera és en 3ra persona, podem no activar cap mètode i és veurà bé.
    C. Les cares visibles que observarà seran les mateixes estiguin activats els dos o només back-face culling.
    D. No poden estar activats els dos mètodes a la vegada.

123. (2018-2019 P Q2) Una escena està formada per un terra quadrat de 20x20 ubicat en el pla XZ, centrat a l'origen de coordenades i amb costats paral·lels als eixos de coordenades, 3 Patricios d'al_cada 4 ubicats amb el centre de la base de la seva capsa mínima contenidora en els punts (4,0,-6), (-4,0,-6) (0,0,-6) respectivament, mirant en direcció X+ i un legoman d'alçada 4 amb el centre de la base de la seva capsa en (0,0,0) i mirant en direcció Z-. Tant el legoman com els Patricios estan escalats uniformement.

- Tenint en compte l'escena 1, indica el pseudo-codi necessari per a realitzar el càlcul de la matriu VM mitjançant angles d'Euler per a què la imatge que s'obtingui sigui la mateixa que amb una VM construïda fent: VM = lookAt ((15,2,-6), (0,2,-6), (0,1,0)). Nota: L'òptica (PM) és perspectiva i no la modifiquem.

  Solució:
  VM = Translate (0,0,-15);
  VM = VM* G_y (-90);
  VM = VM * Translate (0,-2,6);
- Tenint en compte la descripció de l'escena 1 feta en el full 1 de l'examen i considerant com a posicionament de càmera: OBS=(15,2,-6); VRP=(0,2,-6); i up=(0,1,0) (ja vista en un exercici anterior). Què es veuria en el viewport si tenim una òptica ortogonal amb znear=1, zfar=30 i window=(-3,3,-3,3)? El viewport és quadrat.

  Veuríem els tres Patricios en profunditat i el terra.
  Veuríem un únic Patricio i no veuríem el terra.
  Veuríem el legoman davant dels tres Patricios i el terra.
  Veuríem un únic Patricio i un legoman, no veuríem el terra.

124. (2018-2019 P Q2) El procés que obté les coordenades de dispositiu d'un vèrtex dins del procés de visualització:
  a) Només requereix el viewport i les coordenades normalitzades del vèrtex.
  b) Requereix el vèrtex en coordenades de clipping, el window de la càmera i el viewport.
  c) Les dades que requereix depenen de si es realitza abans o després del Fragment Shader.
  d) Podem triar si es realitza abans o després del clipping.

125. (2018-2019 P Q2) Quan definim la posició i orientació de la càmera (viewMatrix) mitjançant angles d'Euler:
  a) No es pot emular l'efecte del vector up que tenim quan la definim amb lookAt.
  b) Sempre és equivalent a tenir un up=(0,1,0).
  c) Cal utilitzar lookAt per ubicar la càmera i lookAt seguit de transformacions d'Euler quan tenim interacció.
  d) Modificant l'angle d'Euler de gir respecte Z podem obtenir l'efecte del vector up.

126. (2018-2019 P Q2) Tenim una escena formada per un cup de costat 1 amb el vèrtex de coordenades mínimes a l'origen de coordenades i totes les cares del mateix color. Completa els paràmetres d'una càmera ortogonal que permeti veure centrat al viewport un hexàgon regular (polígon de 6 costats iguals). L'única restricció al window és que no deformi i permeti veure tot el polígon. El viewport és quadrat.

  VM = lookAt (**(2,2,2)** , (0.5,0.5,0.5), (0,1,0));
  PM = ortho (**-1 , 1 , -1 , 1 , sqrt(3) , 2*sqrt(3)**);

127. (2018-2019 P Q2) Completa el codi del Vertex Shader següent per a què gl Position tingui les coordenades que s'esperen a la sortida del Vertex Shader:

```
in vec3 vertex;
uniform mat4 VM, PM, TG;
void main()
{
gl Position = PM * VM * TG * vec4(vertex, 1) ;}
```

128. (2018-2019 P Q2) Ordena els següents processos del procés de visualització d'OpenGL:

i. Obtenció de coordenades d'observador	1) ...
ii. Rasterització	2) ...
iii. Obtenció de coordenades de dispositiu	3) ...
iv. Obtenció de coordenades normalitzades	4) ...

Solució: 1) i. 2) iv. 3) iii. 4) ii.

129. (2018-2019 P Q2) Dibuixem un quadrat mitjançant dos triangles amb vèrtexs V1=(-1,-1,0), V2=(1,-1,0), V3=(-1,1,0), V4=(-1,1,0), V5=(1,-1,0) i V6=(1,1,0). Suposant que el viewport és quadrat, dibuixa què es veurà en el viewport si pintem aquest quadrat amb els següents vertex i fragment shader (els colors els pots deixar indicats).

Vertex Shader:

```
in vec3 vertex;
uniform mat4 VM, PM, TG;

void main()
{
gl_Position = vec4(vertex,1);
}
```

Fragment Shader:

```
void main ()
{
        if (gl_FragCoord.x < gl_FragCoord.y)
          gl_FragColor = vec4 (1, 0, 0, 1);
        else
          gl_FragColor = vec4 (0, 0, 1, 1);
}
```

131. (2018-2019 Q2) Tenim una escena formada per dues parets. La primera paret medeix 20x6x2 (en X, Y i Z respectivament) i té el centre de la seva base al punt (0,0,1) i la segona paret medeix 2x6x18 (en X, Y i Z respectivament) i té el centre de la seva base al punt (0,0,11). Indica quin és el càlcul de la viewMatrix d'una càmera ortogonal que permet veure una imatge en forma de T centrada en el viewport. Suposa que el càlcul de la projectMatrix és correcte..

A. VM = Translate (0,0,-12);
VM = VM * Rotate_Y (90);
VM = VM * Translate (0,-3,-10);
viewMatrix (VM);

B. VM = Translate (0,0,-12);
VM = VM * Rotate_Z (90);
VM = VM * Translate (0,0,-10);
viewMatrix (VM);

C. VM = lookAt (0,12,10, 0,0,10, 0,0,-1);
viewMatrix (VM);

D. VM = lookAt (0,12,10, 0,3,10, 1,0,0);
viewMatrix (VM);

131. (2018-2019 Q2) Indica quina de les següents llistes de processos del Procés de Visualització d'OpenGL està en l'ordre correcte:

A. Vertex Shader - Rasterització - Fragment Shader - Transformació a coordenades de dispositiu

B. Vertex Shader - Divisió perspectiva - Transformació a coordenades de dispositiu - depth-buffer

C. Transformació a coordenades de dispositiu - Rasterització - depth-buffer - Fragment Shader

D. Clipping - Rasterització - Transformació a coordenades de dispositiu - Fragment Shader

132. (2018-2019 Q2) Indica quina de les següents afirmacions és la correcta:

A. Si l'escena té un sol objecte no cal tenir activat el depth-buffer per eliminar les caresno visibles per l'observador, perquè res el taparà.

B. El depth-buffer és un algorisme d'elimació de cares ocultes que només cal activar si la il·luminació es calcula en el Fragment Shader.

C. Sempre cal tenir activat el depth-buffer per assegurar que es visualitza la geometria visible per l'observador.
D. Si es realitza el clipping no cal activar el depth-buffer perquè ja s'hauran eliminat les cares no visibles per l'observador.

Solucions

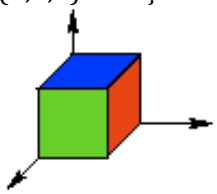
4. El VRP el situarem al centre de la zona per a que aquesta quedi centrada en el viewport: **VRP** = **P.** Per a veure la zona més acuradament, situem la direcció de visió perpendicular a ella:
OBS:= **VRP**+d***n**; podem posar una d=8 que estarà fora de la zona a inspeccionar (de gruix 2mm). Com 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

12. 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).

77.

a) El cub descrit és com es veu a la imatge. Així doncs, l'única manera de veure les dues cares vermella i verda al viewport és mirant-les des d'una direcció paral_lela al vector (1,0,1) i a alçada 1 per a què les cares quedin centrades.



Llavors, posem el VRP per exemple a VRP=(2,1,2) que _es el centre de l'aresta que uneix aquestes dues cares, i la càmera la posem per exemple a OBS=(3,1,3) (podria ser qualsevol punt (x,1,x) on x>=2. Com que a la imatge final hem de veure la meitat esquerra del triangle de color verd i la meitat dreta de color vermell, això vol dir que la vertical de la càmera ha de ser la mateixa que la de l'escena, i per tant up=(0,1,0).

Els paràmetres d'una òptica axonomètrica són el window (left, right, bottom, top), el Znear i el Zfar. El window, com que l'alçada de les cares que volem veure _es de 2 unitats i la relació d'aspecte del viewport és de 2, podria ser left=-2, right=2, bottom=-1, top=1. El Znear ha d'estar per davant del cub, així que com que la distància de l'observador a l'aresta més propera és d'arrel(2), podem posar un Znear=1, per exemple, i el Zfar el podem posar darrera del tot del cub i per tant hauria de ser més

gran que 3*arrel(2), per exemple Zfar=5.

b) Amb una càmera perspectiva, ben definida per a què es vegi el mateix window, el que observarem és que les arestes que estan més lluny de l'observador es veuen més petites, i per tant, les arestes de la vora del que es veu del cub es veuran més petites que l'aresta que es veu al centre. L'efecte seria el mostrat a la figura.