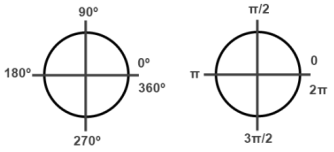
Introducció i conceptes bàsics.

- **Dades :** informació per crear un model (vértexs, colors, etc.).
- **Model :** és l'objecte en qüestió (cub, patricio, terra).
- **Transformacions geomètriques** : fetes per l'usuari, matrius de 4*4 (x, y, z, 1).
  - **Scale :** escala l'objecte de l'escena. TG = scale(TG, vec3(escala)).
  - **Rotate :** gira l'objecte en radians. TG = rotate(TG, float(angle), vec3(eix rotació)). Si girem en sentit horari l'angle serà negatiu mentre que si ho fem en antihorari serà positiu.



  - **Translate :** mou l'objecte x y z posicions. "TG = translate(TG, 3, 4, 5)".

En el cas de escalar i rotar s'ha de fer la transformació desde el centre de l'escena, per tal cal fer un translate primer al centre de l'escena fer la transformació i tornar l'escena a lloc.

- **Visualització :** quan afegim les cameres o llums.
- **Perifèrics :** és el resultat final vist per pantalla.

Dades * Model * Visualització = D * TG * VM(posició) * PM(Óptica)

- **VAO :** VAO = model, és l'objecte. Es crea un únic VAO per model.
- **VBO :** VBO = característiques del model. Tants com característiques hi hagi.

!!! Per cada VAO existeix un VBO amb les coordenades de la posició del model !!!

- **Validació del model :** Procés en el que es comprova si el model que hem creat és correcte o no, es basa en 3 punts :

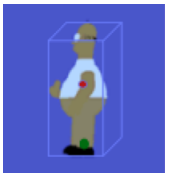
- Que totes les cares estiguin orientades correctament (comprovació de les normals) .
- Que tots els vèrtexs estiguin ordenats de forma coherent.
- Les Arestes, en un model tancat, separen dos triangles o cares.

- **Escena :** escena = perifèrics, resultat final que es veu per la finestra gràfica.

Capses de models : contenen objectes irregulars, les característiques a saber:

- **Límits de la capsa:** Pmin(Xmin, Ymin, Zmin), Pmax(Xmax, Ymax, Zmax)

- **Alçada** = Xmax-Xmin; Altura = Ymax – Ymin; Profunditat = Zmax-Zmin;



- **Centre capsa :** C = ((Xmax-Xmin)/2, (Ymax-Ymin)/2, (Zmax-Zmin)/2);

- **Base capsa :** B = ((Xmax-Xmin)/2, Ymin, (Zmax-Zmin)/2);

PARÀMETRES CÀMERES

- **OBS :** punt origen de la càmera (x, y, z).
- **VRP :** punt en el que apunta la càmera (x, y, z).
- **UP :** Vector vertical que surt de la càmera i que orienta la càmera (x, y, z).
- **zNear :** inici de l'escena, punt en el que es comença a veure.
- **zFar :** fi de l'escena, punt en el que es deixa de veure l'escena.

Càmera perspectiva : focus triangular.

- **FOV** : grau d'apertura de la càmera.
- **Raw** : relació d'aspecte.

#Si Rav > 1 raw = rav

#Si Rav < 1 raw = rav modificar FOV

Perspectiva
FOV = 2 * α
α = arcsin(dOBS/radi)
Raw = aw / hw
hw = 2 * zN * tg(α)

Càmera ortogonal : focus rectangular.

- **l :** left, l'esquerra del rectangle de visió
- **r :** right, la dreta del rectangle de visió.

Ortogonal
Raw = aw / hw
aw = r - l
hw = t - b

- **b :** bottom, la part de sota del rectangle de visió.

- **t :** top, la part de dalt del rectangle de visió.

rav > 1 —> l = -rav * R | r = rav * R | b = -R | t = R

rav < 1 —> l = -R | r = R | b = -R / rav | t = R/rav

Ús de la càmera mitjançant lookAt

1. Definir la posició de la càmera —> **VM = lookAt(OBS, VRP, UP);**
2. Definim la òptica de la càmera —> **PM = perspective(FOV, raw, zNear, zFar);**
   PM = ortogonal(l, r, b, t, zNear, zFar);

Ús de la càmera mitjançant àngles d'Euler

Inicialment la càmera ja està centrada i no es mourà en cap mnomment. Si volem veure-ho de forma diferent, hem de girar l'escena.

VM = VM * Translate(0,0,-d) // d = VRP - OBS

VM = VM * Rotate()

VM = VM * Translate(-VRP)

viewMatrix(VM)

Zoom

Perspectiva	Ortogonal
1. Modificar FOV 2. Modificar zNear zFar 3. Modificar OBS i VRP	1. Modificar window

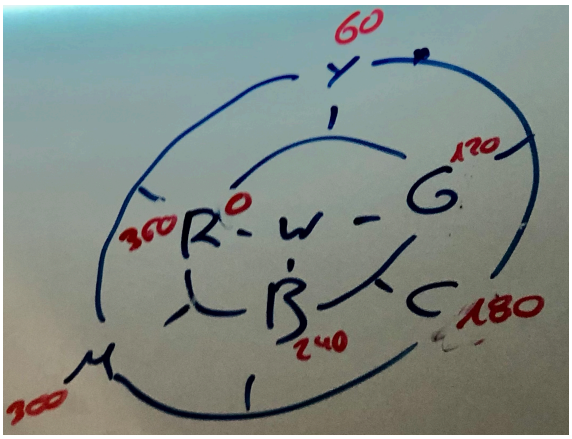
---

Camera 3a persona

1. Buscar Pmin i Pmax de l'escena
2. Definir VRP (centre de la capsa)
3. Buscar R = dist(Pmin,Pmax)/2
4. Definir d = 2 * R
5. Definir OBS = VRP + d * v(direcció en que s'allunya)
6. Definir paràmetre UP
7. Definir tipus òptica
8. Definir zNear(0 < zNear <= d-R) i zFar(zFar >= d+R)
9. Raw = 1
10. Persp. FOV = 2*arcsin(R / d)

Orto. L == b == -R i r == t ==R

Color



Model	RGB	CMYK (k = puresa)	HSV
Mitjà:	Llum	Paper (tinta)	-
Color base:	Negre	Blanc	-
Representació:	(R,G,B)	(C,M,Y)	(H,S,V)
Blanc:	(1,1,1)	(0,0,0)	(x,0,1)
Negre:	(0,0,0)	(1,1,1)	(x,x,0)
Vermell:	(1,0,0)	(0,1,1)	(0,1,1)
Groc:	(1,1,0)	(0,0,1)	(60,1,1)
Verd:	(0,1,0)	(1,0,1)	(120,1,1)
Cian:	(0,1,1)	(1,0,0)	(180,1,1)
Blau:	(0,0,1)	(1,1,0)	(240,1,1)
Magenta:	(1,0,1)	(0,1,0)	(300,1,1)

Preguntes teòriques

---

1. **Processat de Vèrtexs : (**Model, Aplicació, Observació) . Hem mogut l'objecte, la posició o el que sigui del nostre model.
2. **Clipping + Perspective division :** (Clipping,Normalitzades). Borra tots els triangles que es queden fora de l'angle de visió en el que ens trobem com a observador. El clipping borra tot alló que no esta dins del rang de visió.
3. **Device Transform :** (Dispositiu) Es té en compte el viewport I la relació d'aspecte (ra). S'ha de mantenir igual.
4. **Back - Face Culling :** (Back-Face Culling) esborra les parts amagades dels triangles.
5. **Rasterització :** procés en que es reben els triangles i es pinten amb punts o segments. Tot seguït s'envia al fragment shader.
6. **Processat de fragments :** fet en el fragment shader, on es posen les textures.
7. **Z-Buffer :** Al activar-se, elimina parts dels triangles amagades per altres triangles.

SCM -> SCA -> SCO -> SCC **—>** SCN **—>** SCD **—>** SCF

I = dades TG = Transformació Geomètrica VM = View Matrix PM = Project Matrix.

- **SC Model :** conté les coordenades inicials pels models.
- **SC Aplicació :** obtingudes un cop les SCM són posicionades a l'escena (TG * I).
- **SC Observador :** obtingudes un cop les SCA són transformades segons les coordenades de l'observador (VM * TG * I).
- **SC Cippling :** obtingudes un cop les SCO són transformades segons l'òptica de la càmera (V = PM * VM * TG * I)
- **SC Normalitzades :** obtingudes del resultat de clipping (V = V / w)
- **SC Dispositiu :** obtingudes del resultat del Device Transform per ocupar la finestra gràfica correctament (X * V, sent x la transf. per ocupar tota la pantalla).

EXAMEN 2017-2018, Q2.

1. (1 punt) Tenim un dibuix de color CMY = (0.3, 0.3, 1), respon justificant les respostes:

- Quina representació tindria aquest color en HSB?

1. RODONA DE COLORS FOTO MOVIL
2. CMY (0.3, 0.3, 1) = RGB (0.7,0.7,0) (“Restem”
3. HSB = (angle color que predomina, rang (0.7-1), petit del rang) = **HSB(60, 0.7-1,0.7)**

- Tenim un projector al que li falla la llum vermella i al que li posem un filtre magenta davant. De quin color es veurà el dibuix projectat en pantalla usant aquest projector i amb el filtre descrit?

R G B

0.7 0.7 0

+

proj x - -

———————-

Filtre 0. 0.7 0

Si no si el magenta es r + b per això nomes deixem actives r i b

———————

0. 0. 0. No es veu res

2. (2 punts) Disposem d’una funció pintaCub() que pinta un cub de costat 1 orientat amb els eixos i centrat a l’origen de coordenades. També disposem d’una funció pintaPiramide() que pinta una piràmide de base quadrada de costat 1 amb el centre de la seva base a l’origen de coordenades i l’eix de la piràmide en direcció Z+ amb alçada de l’eix 5 (distància del centre de la base al vèrtex compartit per les quatre cares de la piràmide). Es vol construir una escena amb una maneta d’un rellotge. L’escena tindrà:

• Un prisma rectangular de mides 0.5 en X, 4 en Y i 0.5 en Z que tindrà el seu centre al punt (0,1,0).

• Una piràmide de base quadrada de costat 0.5 i alçada 1 que tindrà el centre de la seva base al punt (0,3,0) i el seu eix en direcció Y+.

a) Indica quin seria el pseudocodi d'una funció pintaEscena() que permeti pintar l'escena descrita. Especifica el pseudocodi que permet trobar les TGs requerides per a cada part de l'escena i utilitza els mètodes pintaCub() i pintaPiramide() per a pintar-les.

1 . Recordar que hem de programar de baix a dalt.

2. TG1 = TG1 * translate(0, 1, 0)

TG1 = TG1 * scale(0.5, 4, 0.5)

TG1 = TG1 * translate(-centreCub)

modelMatrix(TG1);

pintaCub();

TG2 = TG1 * translate (0,3,0)

TG2 = TG2 * scale(1,0.2,1)

TG2 = TG2 * rotate(π/2,(1,0,0))

TG2 = TG2 * translate(- centrebase)

modelMatrix(TG2)

pintaPiràmide();

b) Suposant que tenim un angle de rotació en la variable alfa (que algun altre mètode modifica), què caldria modificar del codi de l'apartat anterior per a què l'agulla representada del rellotge (prisma + piràmide) roti alfa graus respecte l'eix de les Z (simulant el que fa la maneta dels minuts d'un rellotge)?

TG1 = TG1 * translate(0, 1, 0)

TG1 = TG1 * rotate(alfa,(0,0,1))

TG1 = TG1 * scale(0.5, 4, 0.5)

TG1 = TG1 * translate(-centreCub)

modelMatrix(TG1);

pintaCub();

TG2 = TG2 * translate (0,3,0)

TG2 = TG2 * rotate(alfa,(0,0,1))

TG2 = TG2 * scale(1,0.2,1)

TG2 = TG2 * rotate(π/2,(1,0,0))

TG2 = TG2 * translate(- centrebase)

modelMatrix(TG2)

pintaPiràmide();

3. (1 punt) Tenim una escena formada per tres cubs alineats 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 c`amera 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 segu¨ents c`ameres aconseguiria aquest objectiu suposant que l'`optica est`a 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)

4. (1 punt) Dos estudiants estan comentant què cal modificar de l'òptica d'una c`amera en tercera persona quan és 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 óptica.

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) Nom´es l'estudiant B ho far`a correctament.

c) Nom´es ho far`a correctament l'estudiant A.

d) Cap dels dos estudiants t´e ra´o.

5. (1 punt) 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(...); és a dir per obtenir la mateixa imatge de l'escena tot suposant que no es modifica l'`optica.

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)

7. (0.5 punts) Per a què el proc´es 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.

8. (0,5 punts) 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.

9. (0.5 punts) Donada una càmera perspectiva en primera persona, un estudiant implementa la funcionalitat d'avançar l'observador en la direcció 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, perquè 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 avançar en una direcció diferent.

d) Pot arribar a mirar en la direcció contrària a la que avança.

10. (0.5 punts) Suposem que tenim una escena en què la seva esfera contenidora està centrada a l'origen i t´e radi 50. Inicialment tenim la següent definició de càmera: FOV=60 (est`a 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.

11. (0.5 punts) 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;

12. (0.5 punts) Un estudiant ha enviat a imprimir a una impressora CMY un dibuix de color verd però quan el recull de la impressora el que hi veu en el paper és un dibuix de color cian. Què ha passat?

a) La impressora no té tinta magenta.

b) El paper en què ha imprès és de color groc.

c) S'ha equivocat en assignar el color al dibuix, és impossible que passi això.

d) Cap de les altres respostes.