Model Geomètric (2)

Dades

Càmera Il·luminació

Model *Geomètric*

Visualització

OpenGL

Píxel(x,y,c)

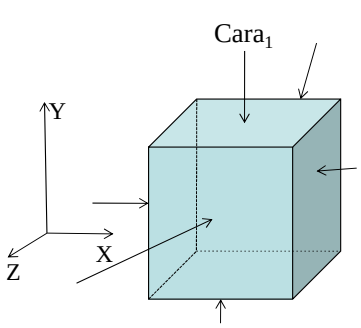
FrameBuffer

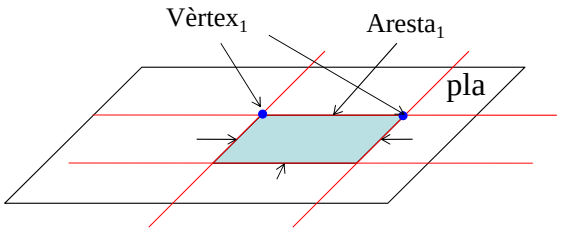
- Sòlids (3D)
- Determinats per la seva superfície
- Model de Fronteres
- Cares Planes

taula [res_x][res_y] de color
color = (r,g,b)
r,g,b en [0…$n_{r,g,b}$-1]

- *Material Docent: CD tema 5*

Model Fronteres: Cub

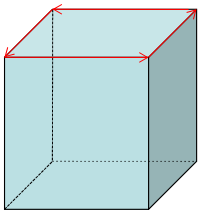




- 6 cares (C_1...C_6)
- 24 arestes (A_1...A_{24})
- 48 vèrtexs (V_1...V_{48})

- Pla: ax+by+cz+d=0;
- 4 arestes/cara ➔ aresta: recta
- 2 vèrtexs/aresta ➔ (x,y,z)

Model Fronteres: Cub



Cares

Eqüació (a,b,c,d)

Arestes

Eqüació (P_o, v)
…

Vèrtexs

x	y	z
…	…	…

- 6 cares, 24 arestes, 48 vèrtexs

Problemes:
- Repetició informació geomètrica
- Manca relació entre elements geomètrics => topologia/adjacència

Preguntes
- Què és requereix per pintar?
- Cóm sabem que tenim correctament representat l'objecte?

Model Fronteres

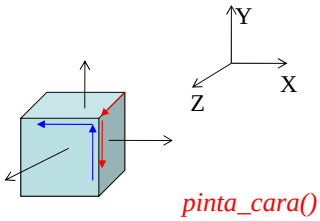
- Guardar explícitament informació superfície: cares, arestes, vèrtexs

⇨ Geometria: cares (equació), arestes (equació) i vèrtexs (coordenades)

Sistema de Coordenades (del Model)

⇨ Topologia: relació d'adjacència entre elements

C: {C}	C: {A}	C: {V}
A: {C}	A: {A}	A: {V}
V: {C}	V: {A}	V: {V}



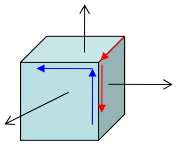
- Vàlid:
  - cares orientades
  - cada aresta separa 2 cares
  - **C: {A} + A: {V}** => polígons frontera **C: {$V_{ordenats}$}**
    - tancats i simples
    - ordenació vèrtexs coherent amb l'orientació de les cares
  - intentar que no hi hagi informació geomètrica redundant

pinta_cara()

```
glBegin (GL_POLYGON)
  glVertex (v1.x,v1.y,v1.z)
  glVertex (v2.x,v2.y,v2.z)
  glVertex (v3.x.v3.y,v3.z)
  glVertex (v4.x,v4.y,v4.z)
glEnd ()
```

- Per a visualitzar: **cares Planes** (facetització, triangulació,…) => C: {$V_{ordenats}$}

Exemple: pintar cub aresta 1 centrat en origen



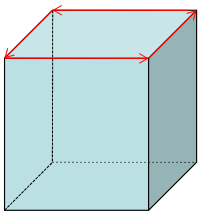
```
void refresh (void) {
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glColor3d (1,0,0);
 glPolygonMode (GL_FRONT,GL_FILL);
 glBegin(GL_QUADS);
    glVertex3f(0.5,0.5,0.5);
    glVertex3f(0.5, -0.5,0.5);
    glVertex3f(0.5,-0.5,-0.5);
    glVertex3f(0.5, 0.5,-0.5);
 glEnd();
 glBegin(GL_QUADS);
    glVertex3f(0.5,0.5,0.5);
    glVertex3f(-0.5, 0.5,0.5);
    glVertex3f(-0.5,-0.5,0.5);
    glVertex3f(0.5, -0.5,0.5);
 glEnd();
 ….
 glutSwapBuffers();
}
```

Pinta_Cara

Pinta_Cub (1);

glutWireCube (1);

Exemples



Cares

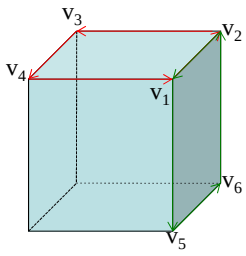
normal	nº V	*Id* 1er V
a, b, c	4	1

Vèrtexs

x	y	z
…		

Vèrtexs repetits ☹

Exemples



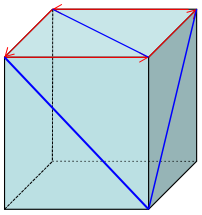
Cares

normal	*Id* [0..nºV-1]
a_1, b_1, c_1	1,2,3,4
a_2, b_2, c_2	1,5,6,2

Vèrtexs

x	y	z

Exemple: No forats, 3 Vèrtexs



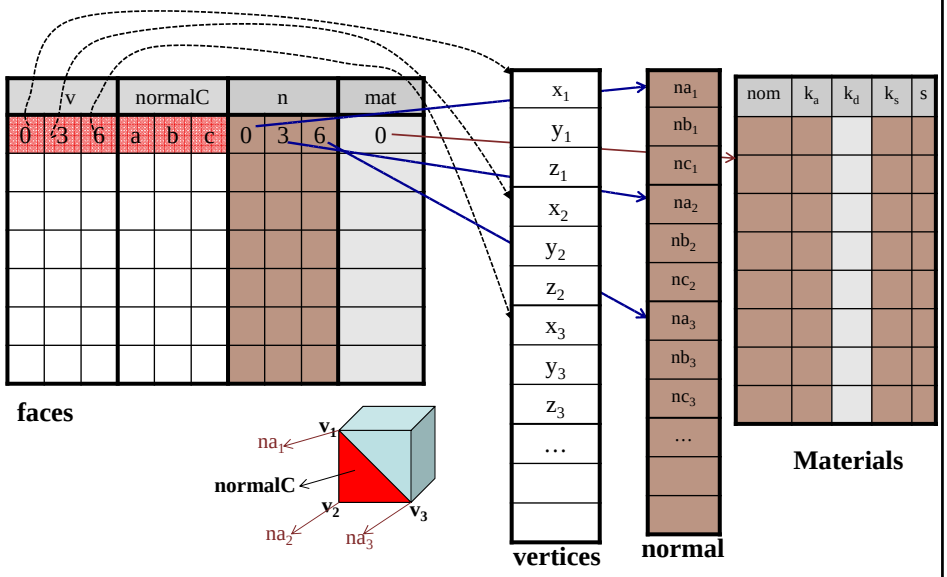
Cares

normal	*Id* [0..2]
a_1, b_1, c_1	1,2,9

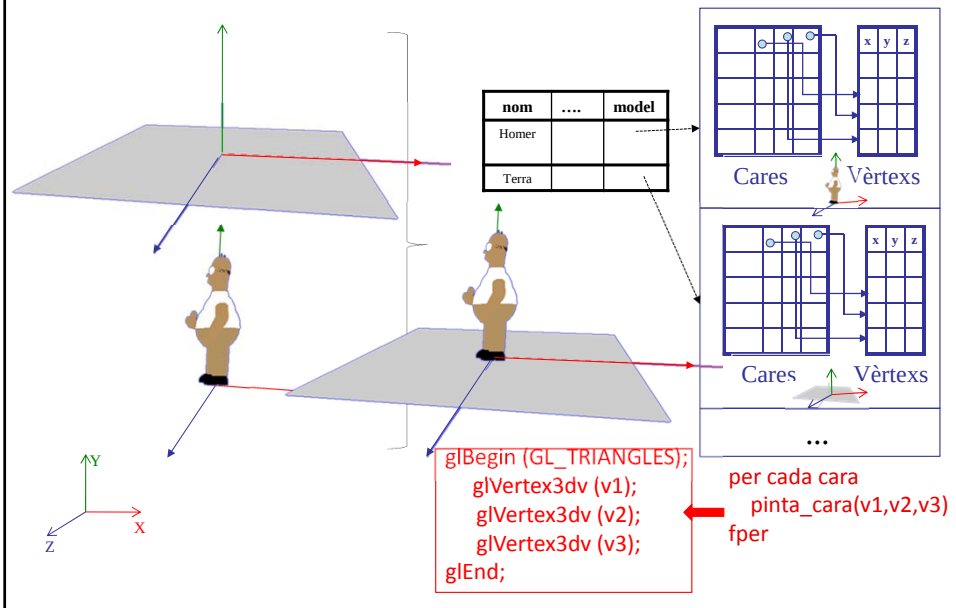
Vèrtexs

x	y	z
…		

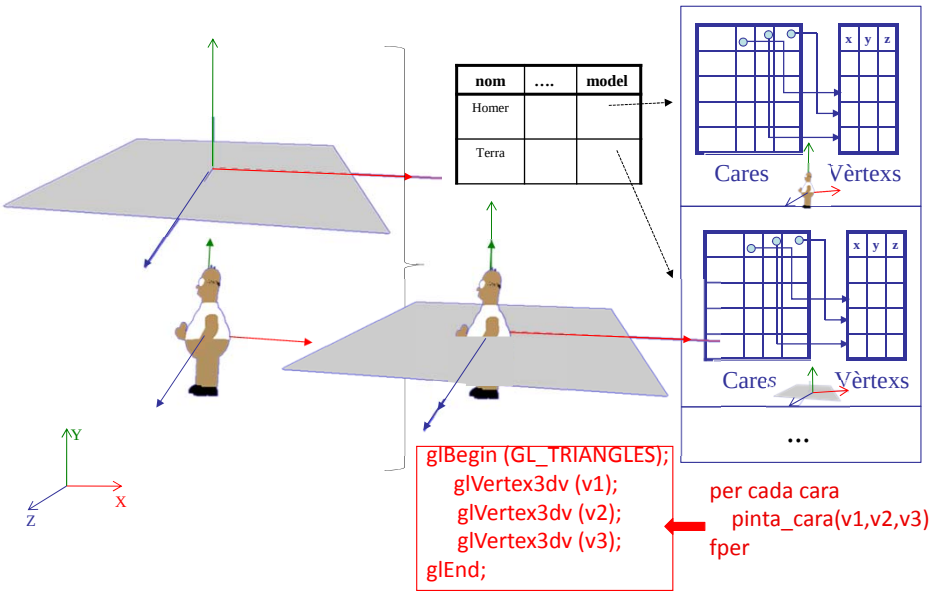
Exemple Laboratori

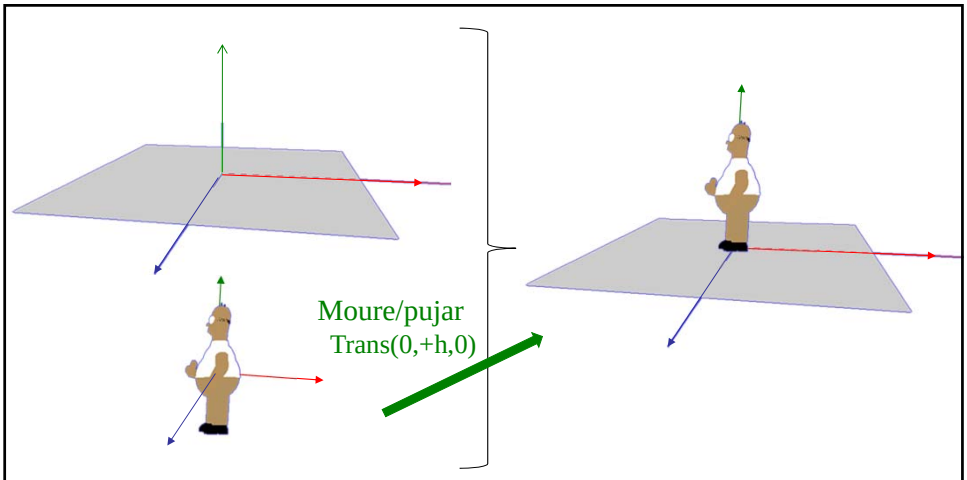


Escena : {objectes}



Models d'escenes : {models}

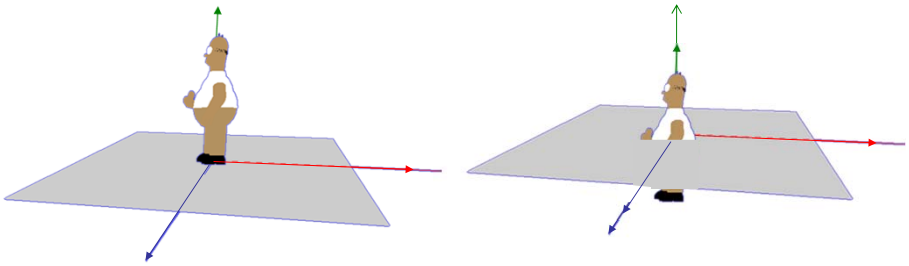




- Modificar coordenades vèrtexs (i eqüació de les cares)

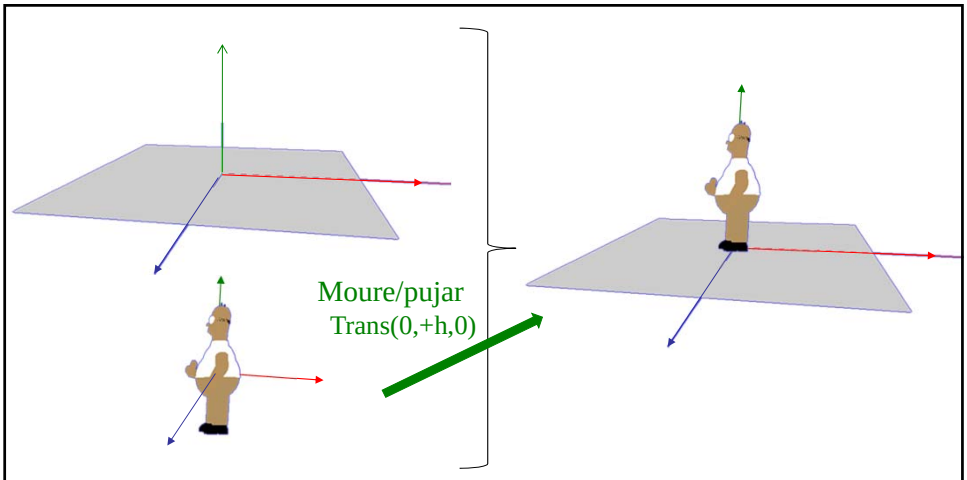
Repasar Àlgebra i TG

Models d'escenes : {models}



SC Aplicació versus SC Model:

- Global versus Local
- SCA => elements directament visualitzables
- SCModel => cal posicionar objectes abans de visualitzar-los.

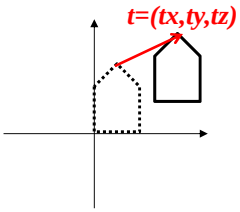


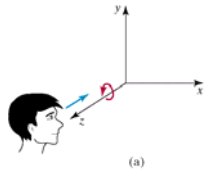
- Cal aplicar TG que modifica coordenades vèrtexs (i equació de les cares)
- TG queda definida per matriu 4x4:

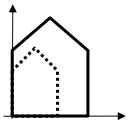
$\mathbf{V_A} = TG\ \mathbf{V_m} = T(0,+h,0)\ \mathbf{V_m}$

$$T(t_x,t_y,t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformació geomètrica → [ ] → Matriu 4x4 TG



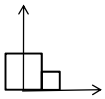




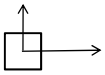
T(tx,ty,tz) G_z(angle) S(sx,sy,sz)

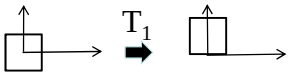
Altres: composició de transformacions => producte de matrius

TG : exemple 3 en laboratori

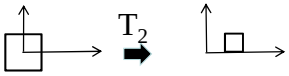


Escena a pintar utilitzant instàncies de pinta_cub(1) (*glutWireCube(1)*)





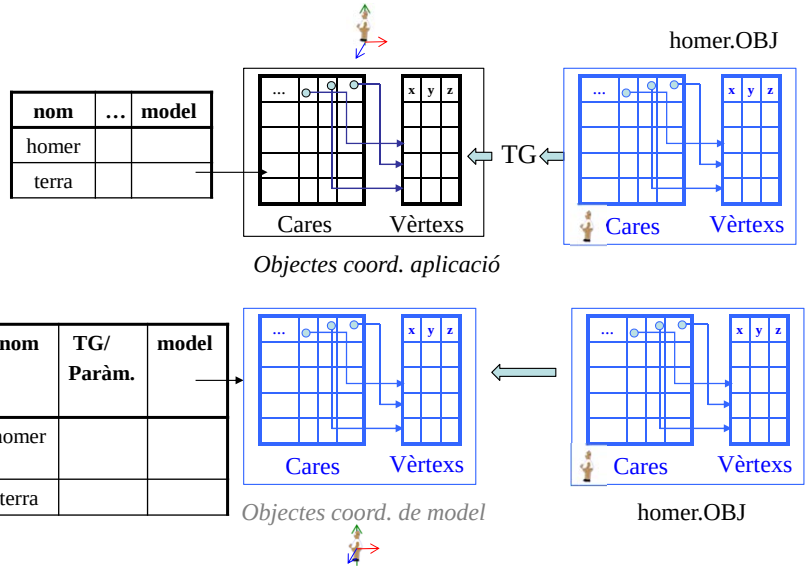
T_1= Trans(0,0.5,0)



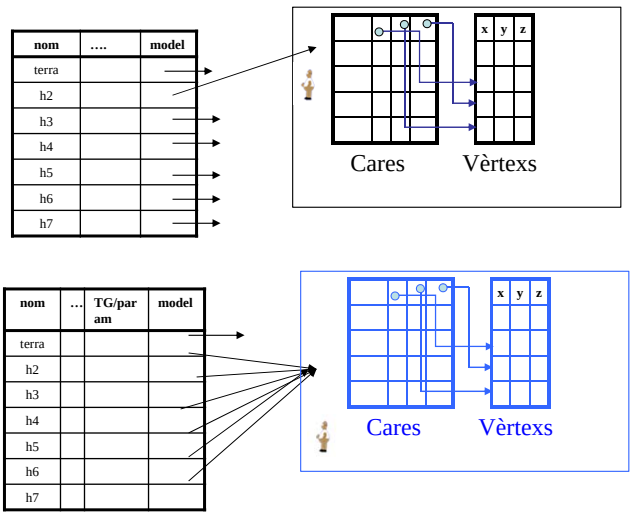
T_2= Trans(0.75,0.25,0)*S(0.5,0.5,0.5)

En laboratori veurem com OpenGL pot ajudar a fer la composició (generar matrius i fer el producte de matrius)

Models escenes



Models escenes



Model Geomètric

Dades

Càmera Il·luminació

Model *Geomètric*

Visualització

OpenGL

Píxel(x,y,c)

FrameBuffer

taula [res_x][res_y] de color
color = (r,g,b)
r,g,b en [0…$n_{r,g,b}$-1]

- Sòlids (3D)
- Determinats per la seva superfície
- Cares Planes
- Model de Fronteres
- Representació escenes, TG

- *Material Docent: CD tema 5*