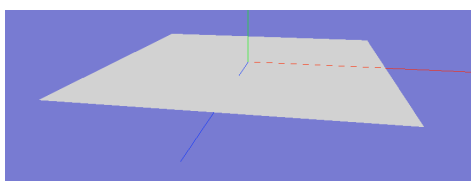


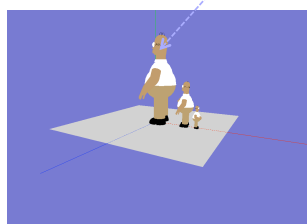
# Exercicis Results

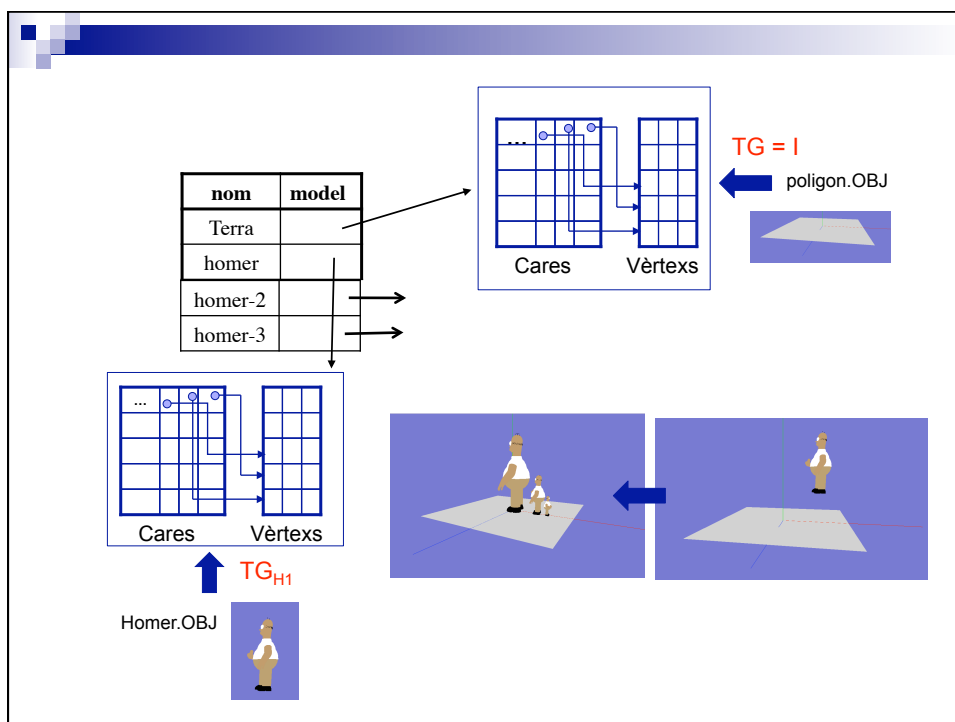
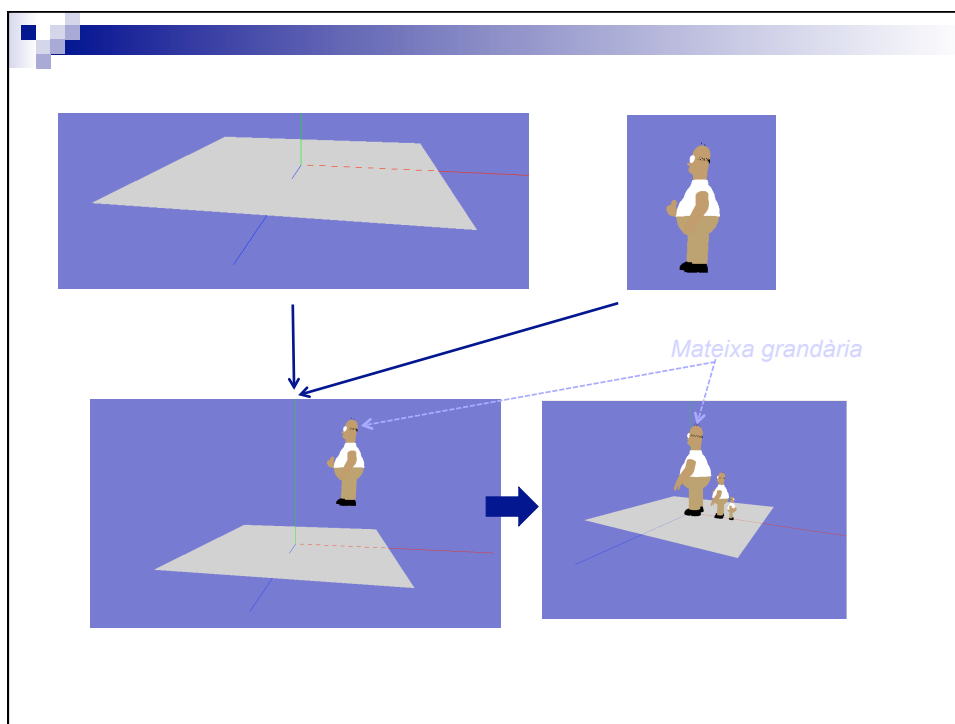
IDI- 2012-2013

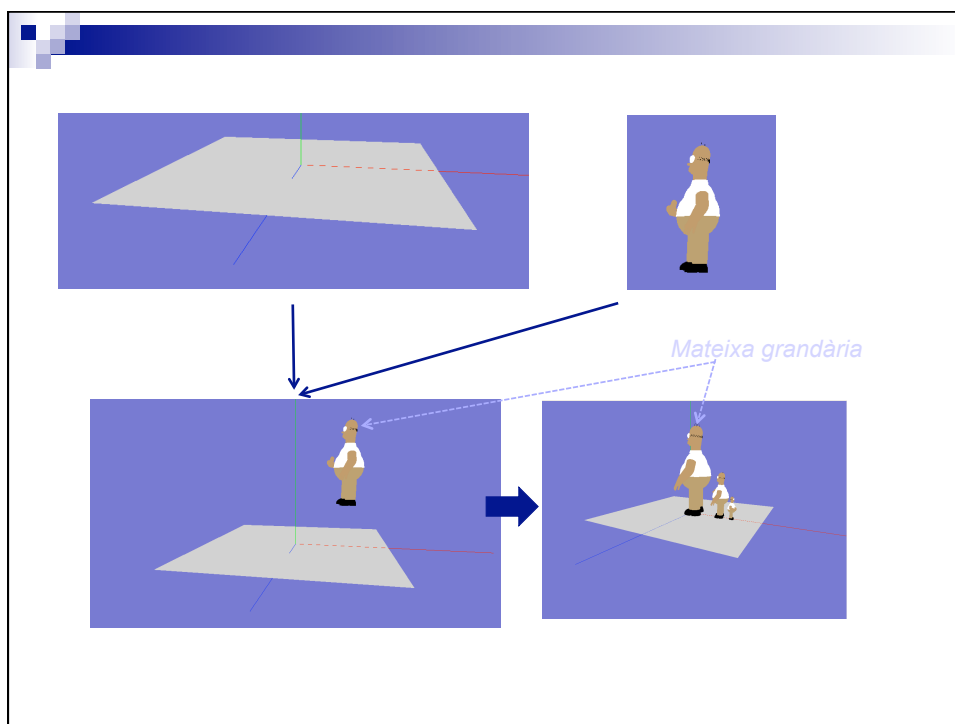
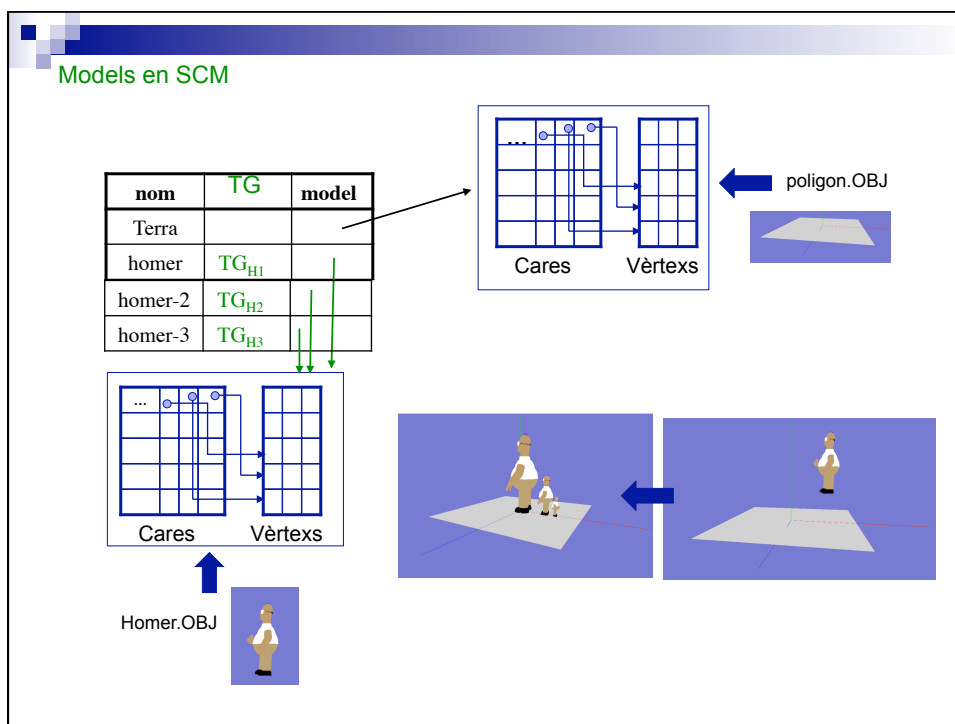
## Exercici 1

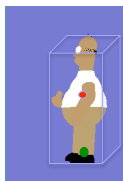


*Mateixa grandària*



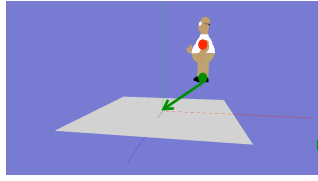




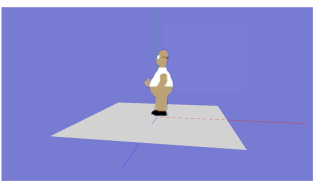


$CapsaMinCont = (xmin, ymin, zmin, xmax, ymax, zmax)$   
 Mides =>  $a = (xmax - xmin)$ ,  $h = (ymax - ymin)$ ,  $f = (zmax - zmin)$   
 $CentBaseCapsa = (cbx, cby, cbz) = (xmin + xmax)/2, ymin, (zmin + zmax)/2$

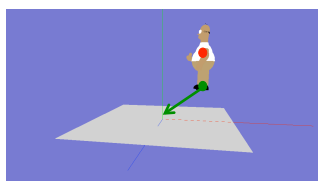
Afegir com atributs al model, si cal calcular sovint



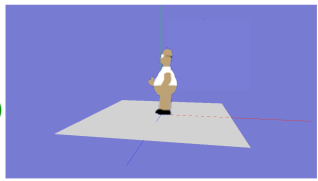
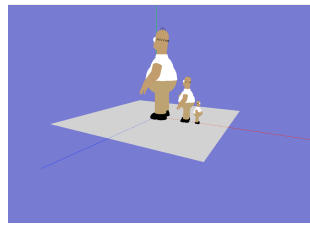
$TG_{H1} = Trans(t)$   
 $t = (-cbx, -cby, -cbz)$



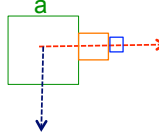
*Noteu que la capsa de Homer mogut és diferent de la del seu model; quina és?*



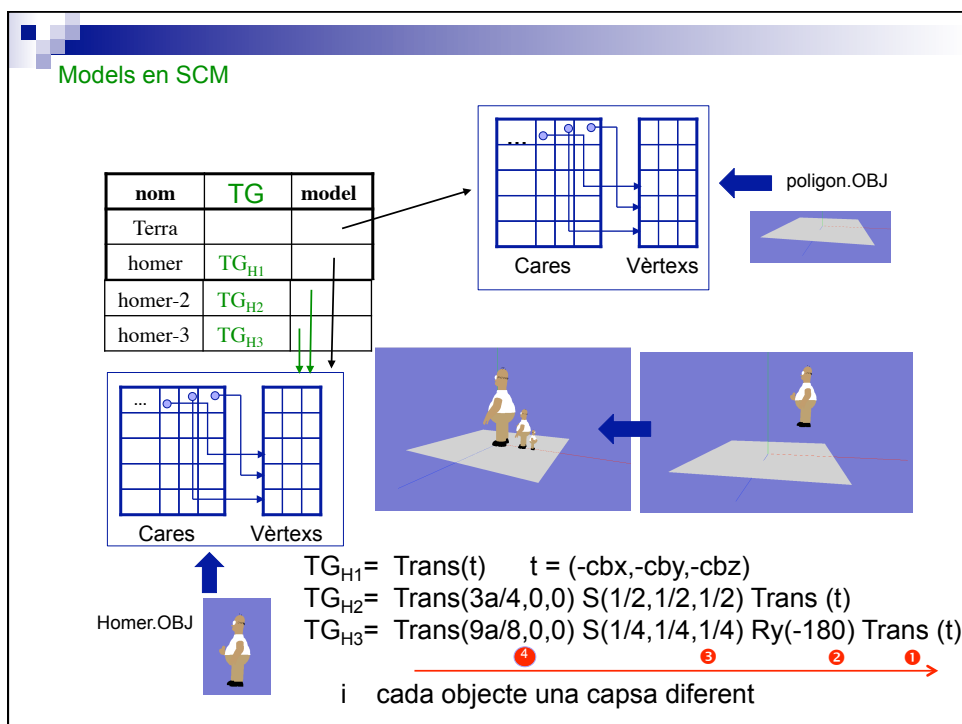
$TG_{H1} = Trans(t)$   
 $t = (-cbx, -cby, -cbz)$

$TG_{H2} = Trans(3a/4, 0, 0) S(1/2, 1/2, 1/2) Trans(t)$



$TG_{H3} = Trans(9a/8, 0, 0) S(1/4, 1/4, 1/4) R_y(-180) Trans(t)$   
 $TG_{H3} = Trans(9a/8, 0, 0) R_y(-180) S(1/4, 1/4, 1/4) Trans(t)$



**Visualització OpenGL: models en SCA (1)**

```
// 1. Netejar finestra
glClear(GL_COLOR_BUFFER_BIT | ...);

// 2a. Definir Viewport
glViewport(0,0,w,h)

// 2b. Model-view Transformation
// Ubicar càmera
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(X,Y,Z, VRPX,VRPY,VRPZ,
uX,uY,uZ);

// 2c. Projection Transformation
// Òptica
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(fov, ratio, near, far);
// o
gluOrtho(left, right, bot, top, near, far);
```

**// 3. Recorregut de l'escena**

```
per cada objecte o fer
per cada triangle t de o
glBegin(GL_TRIANGLES)
glVertex(v1)
glVertex(v2)
glVertex(v3)
glEnd()

fper
fper
```

## Visualització OpenGL: models en SCM i tenim TG (1)

### // 3. Recorregut de l'escena

```

per cada objecte o fer
  TG = o.TG
  per cada triangle t de o
    v1=TG * t.v1; v2 ,...
    glBegin(GL_TRIANGLES)
      glVertex (v1)
      glVertex (v2)
      glVertex (v3)
    glEnd()
  fper
fper

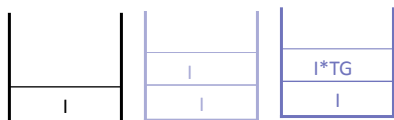
```

## Visualització OpenGL: models en SCM (2)

```

per cada objecte o
  TG = o.TG
  per cada triangle t de o
    glBegin (GL_TRIANGLES)
      v1=TG * t.v1; v2 ,...
      glVertex (v1)
      glVertex (v2)
      glVertex (v3)
    glEnd()
  fper
fper

```



```

glMatrixMode (GL_ModelView)
glLoadIdentity()
per cada objecte o
  TG = o.TG;
  glPushMatrix();
  glMultMatrix(TG)
  per cada triangle t de o
    glBegin (GL_TRIANGLES)
      glVertex (t.v1)
      glVertex (t.v2)
      glVertex (t.v3)
    glEnd()
  fper
  glPopMatrix()
fper

```

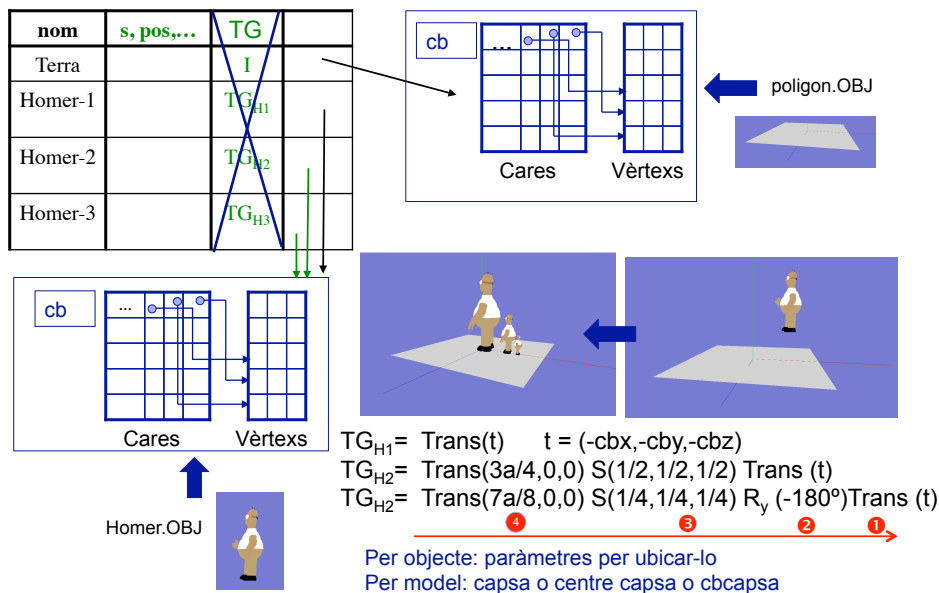
OpenGL projecta el punt **t.v1** (que està en coordenades del model) fent:

$$v1_o = MV * t.v1 = I * TG * t.v1$$

és equivalent a pintar el punt **v1** en coord. de l'aplicació, que és el que volíem!!

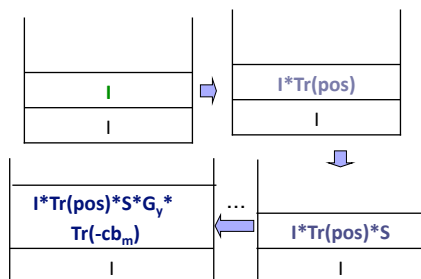
(és a dir, situem objectes al seu lloc emprant OpenGL)

### Visualització OpenGL: models en SCM (3)



### Visualització OpenGL: models en SCM (3)

per cada objecte o  
Càlcul TG a partir o.param  
per cada triangle t de o  
 $v1 = TG * t.v1; v2, \dots$   
glVertex (v1)  
glVertex (v2)  
glVertex (v3)  
fper  
fper

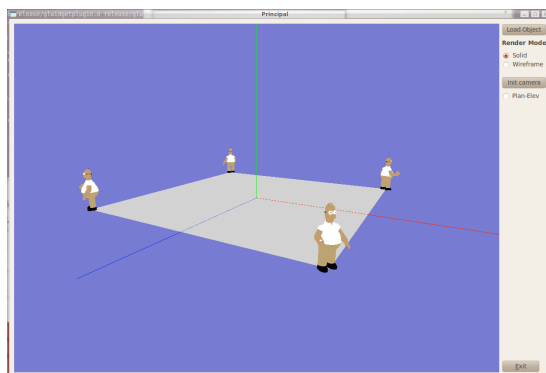


$TG_{H1} = \text{Trans}(t) \quad t = (-cbx, -cby, -cbz)$   
 $TG_{H2} = \text{Trans}(3a/4, 0, 0) S(1/2, 1/2, 1/2) \text{Trans}(t)$   
 $TG_{H2} = \text{Trans}(7a/8, 0, 0) S(1/4, 1/4, 1/4) R_y(-180^\circ) \text{Trans}(t)$

```
//tercer homer
glMatrixMode (GL_ModelView)
glPushMatrix();
glTranslate (posx, posy, poz)
glScale (s,s,s)
glRotate (-180, 0,1,0)
glTranslate (-cb_m_x,-cb_m_y,-cb_m_z)
per cada triangle t de homer_model
glBegin (GL_TRIANGLES)
glVertex (t.v1)
glVertex (t.v2)
glVertex (t.v3)
glEnd()
fper
glPopMatrix()
```

Pinta\_homer

## Exercicis



Mireu la col·lecció de problemes del racó.  
Exercicis 1 i 2 de Bloc 2

20