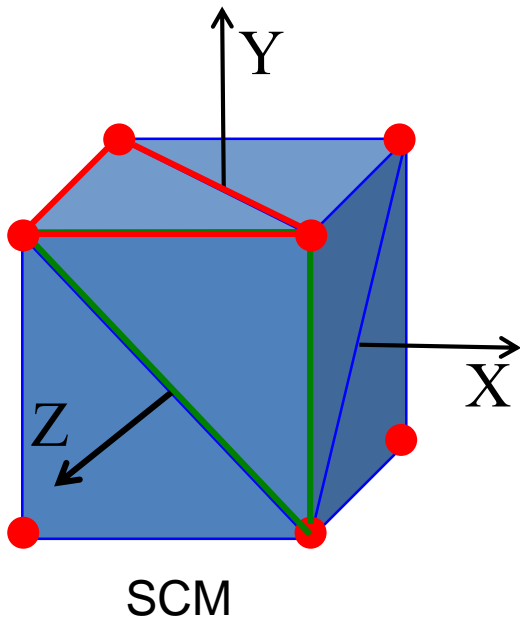


Classe 2: Contingut

- Introducció al procés de visualització
- **Transformacions geomètriques**
- Exercicis

Model Fronteres: conjunt de triangles



Vèrtexs

x	y	z
-1	1	-1
-1	1	1
1	1	1
-1	1	1
1	-1	1
1	1	1
.	.	.
.	.	.

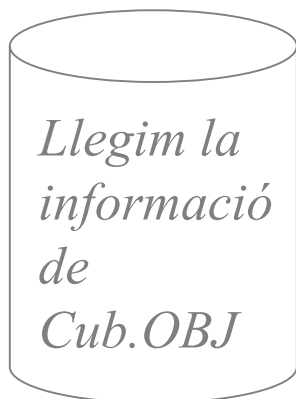
Topologia implícita

Cares

normal	Id Vèrtexs
a_1, b_1, c_1	1,2,3
a_2, b_2, c_2	2,4,3
...	...

Vèrtexs

x	y	z
-1	1	-1
-1	1	1
1	1	1
1	-1	1
1	-1	-1
1	1	-1
-1	-1	-1
-1	-1	1



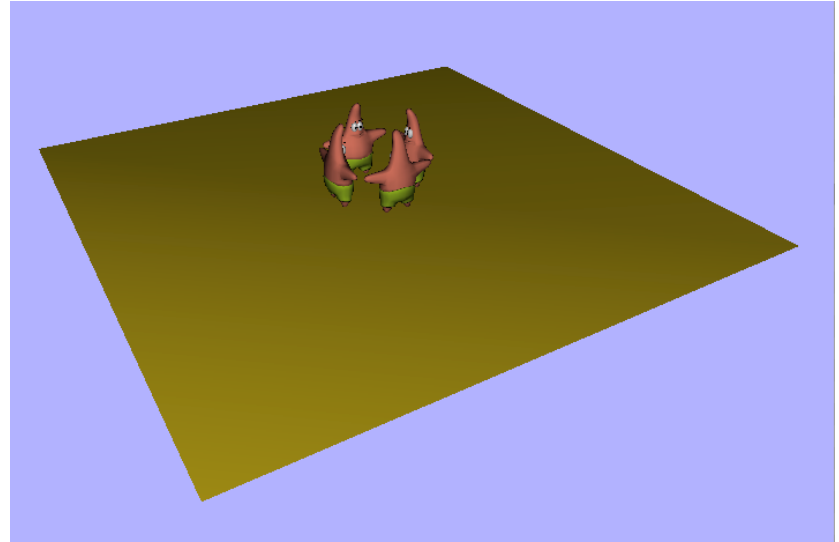
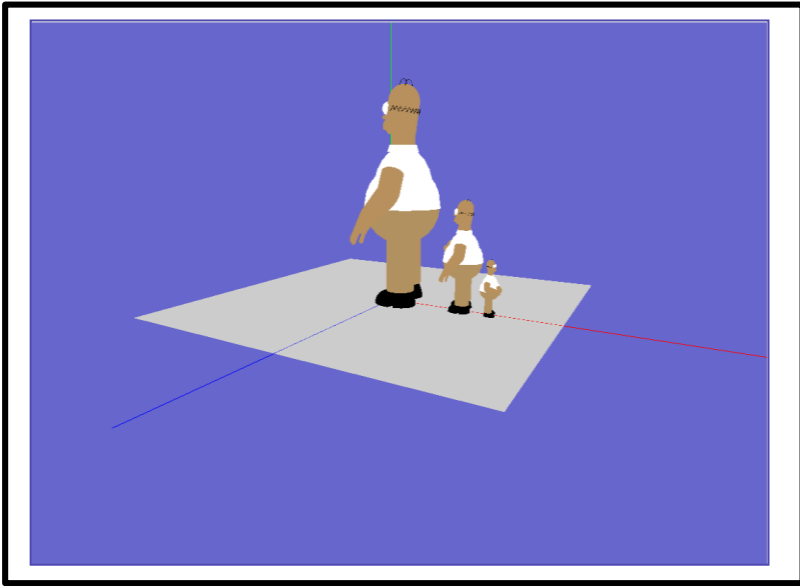
`glm::vec3 Vertices[36]`



Assignem valors en el codi a estructures C++

Topologia explícita

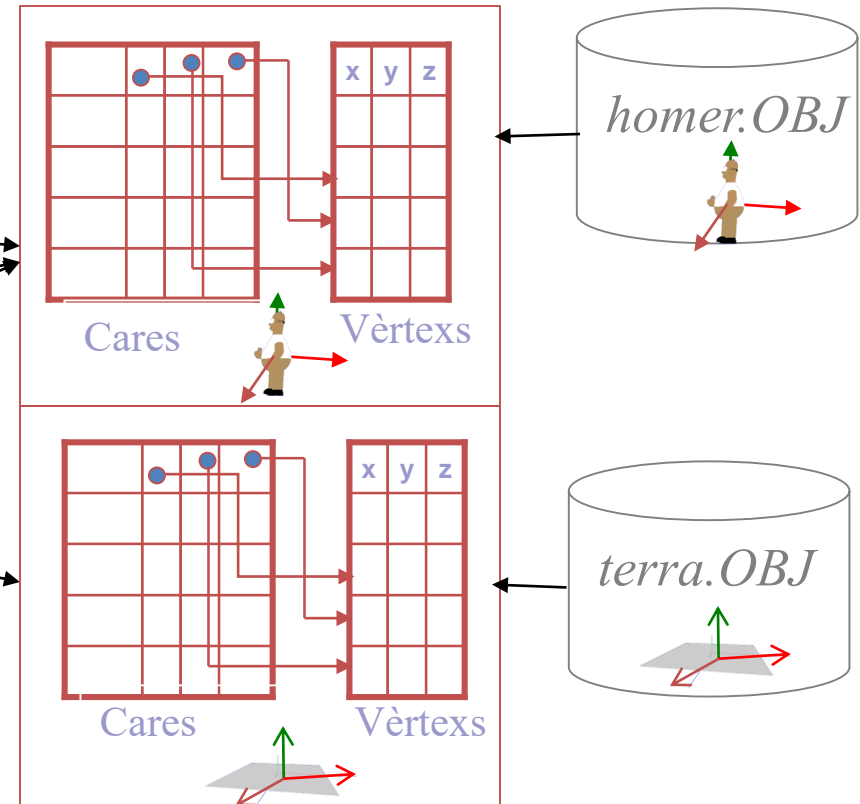
Escenes: Conjunt d'objectes



Escenes: Objectes en SCM.

Objectes

nom			model
h1			
h2			
h3			
terra			

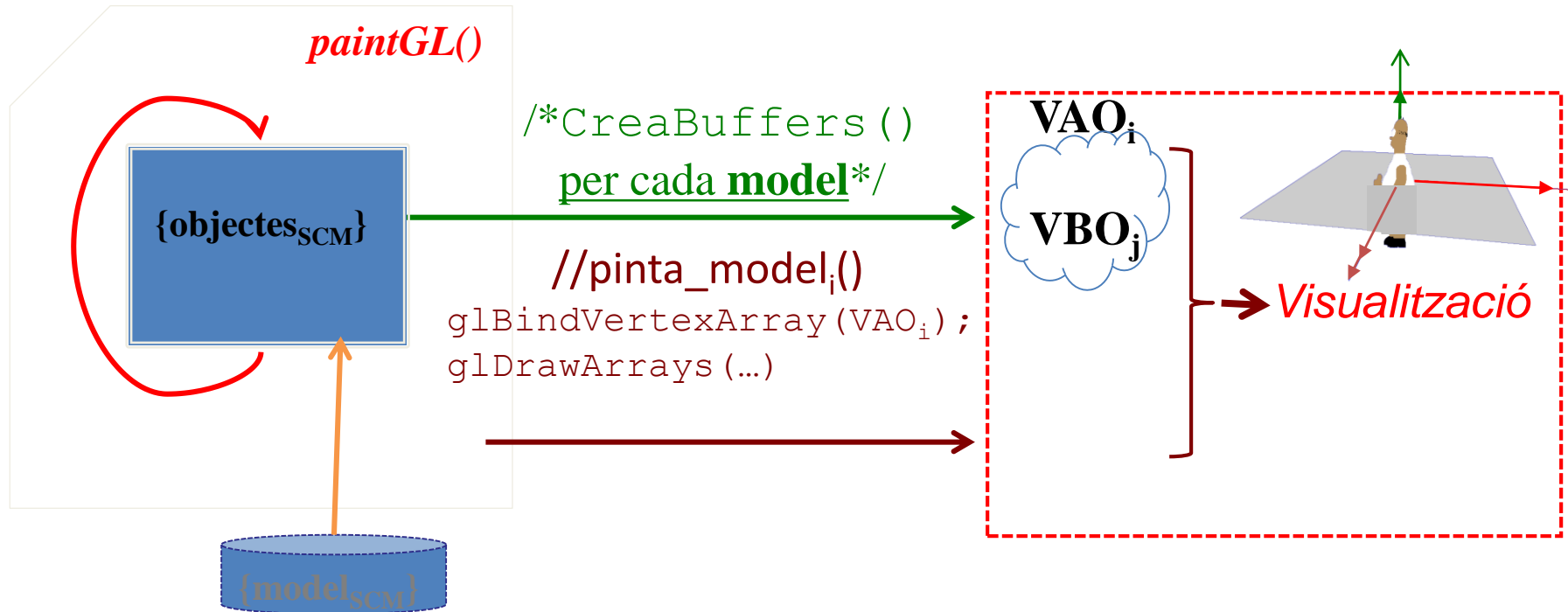


Models en SCM

Escenes: Objectes en SCM. Com fem per visualitzar? (1)

```
/*crear un únic VAOi i VBOj per cada  
model, en CreaBuffers()*/  
per cada model  
    llegir_Model ();  
    crear i omplir VAOi, VBOj  
fper
```

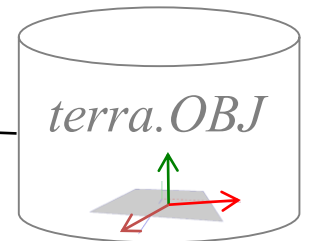
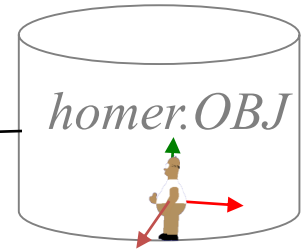
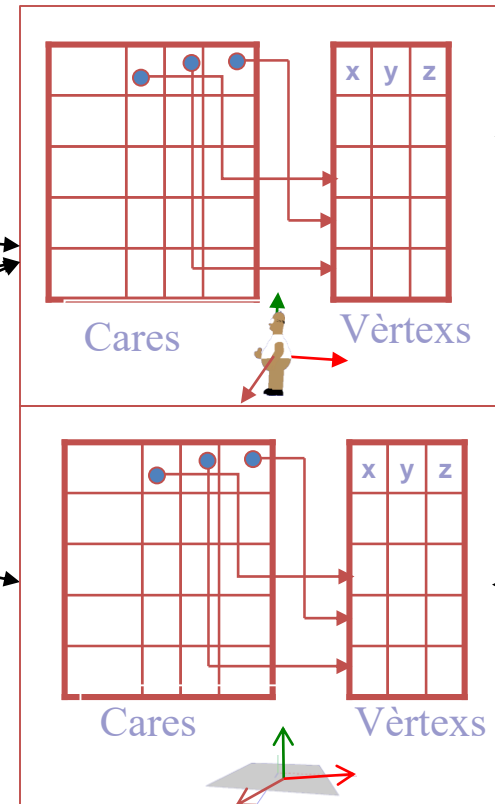
```
//paintGL ();  
per cada objectei  
    // activa VAOi i pinta  
    pinta_modeli ();  
fper
```



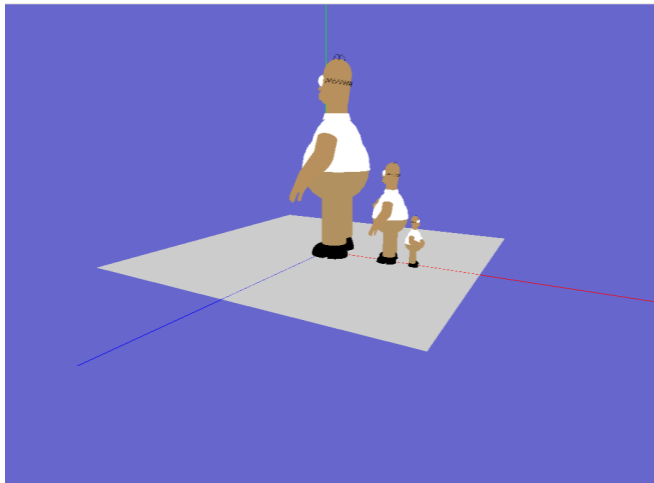
Escenes: Objectes en SCM.

Objectes

nom	...	TG	model
h1			
h2			
h3			
terra			



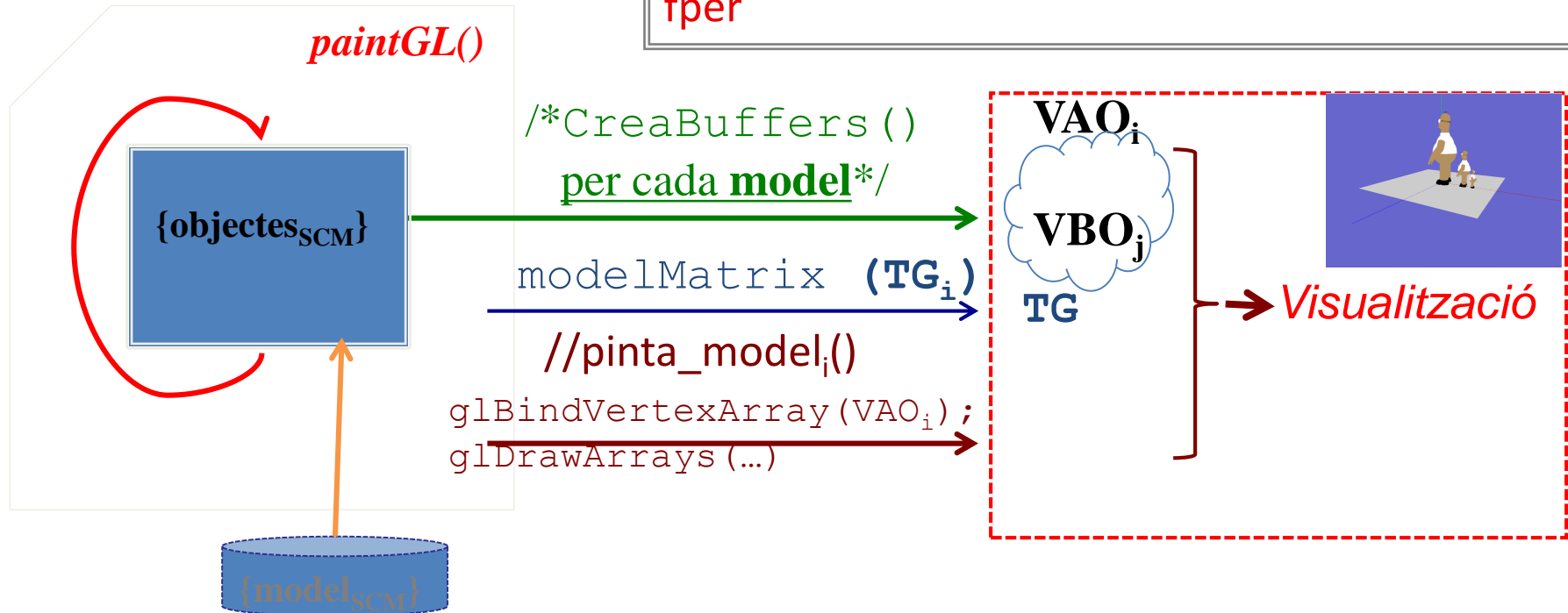
Models



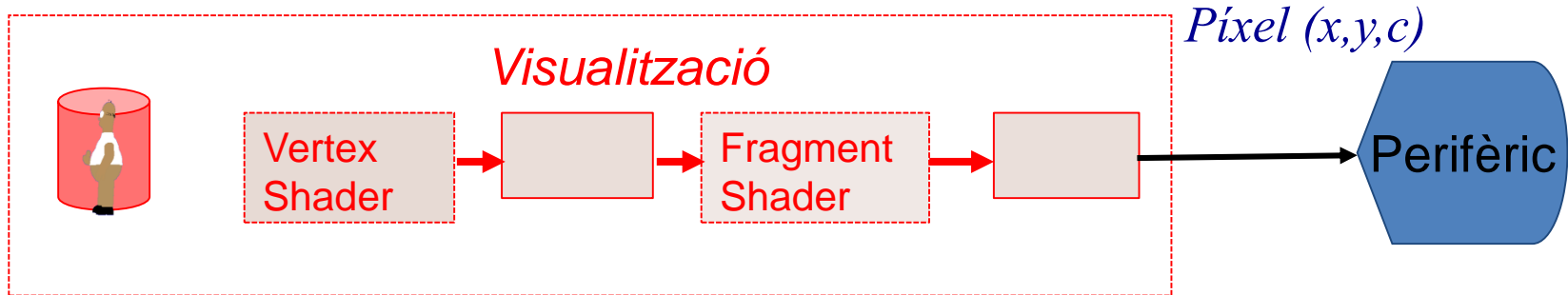
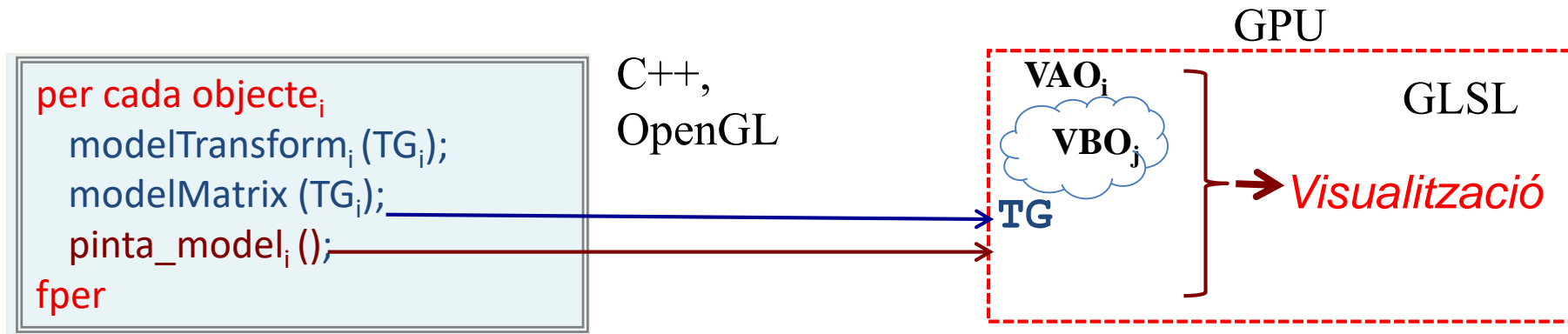
Escenes: Objectes en SCM. Com fem per visualitzar? (2)

```
/*crear un únic VAOi i VBOj per cada  
model, en CreaBuffers()*/  
per cada model  
    llegir_Model ();  
    crear i omplir VAOi,VBOj  
fper
```

```
//paintGL ();  
per cada objectei  
    //Obtenir/calcular la TGi a aplicar a model  
    modelTransformi(TGi);  
    //indicar a OpenGL la TGi enviant “uniform”  
    modelMatrix (TGi);  
    pinta_modeli (); //el VAOi del seu model  
fper
```



Pintar en OpenGL 3.3: “core” mode



```
#version 330 core
in vec3 vertex;
uniform mat4 TG;

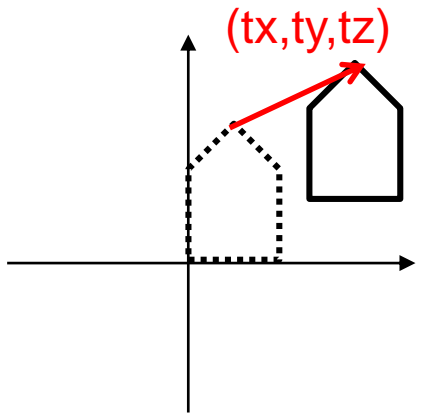
void main() {
    gl_Position = TG * vec4(vertex,1.0);
}
```


Transformacions Geomètriques

Transformació
geomètrica



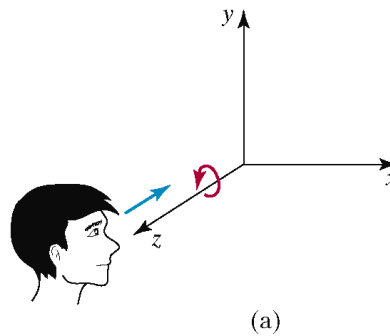
Matriu 4x4
TG



$$x' = x + tx; y' = y + ty; z' = z + tz$$

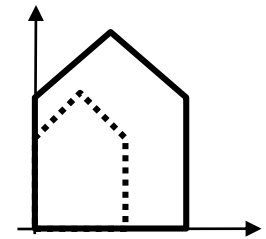
$T(tx, ty, tz)$

$$\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$R_z(\text{angle})$

$$\begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

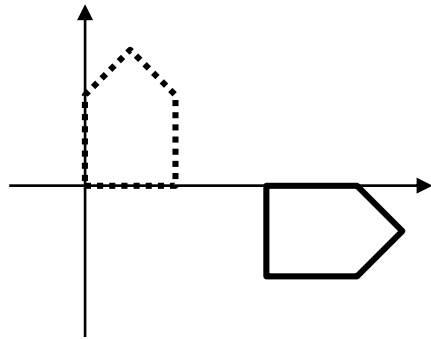


$S(sx, sy, sz)$

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

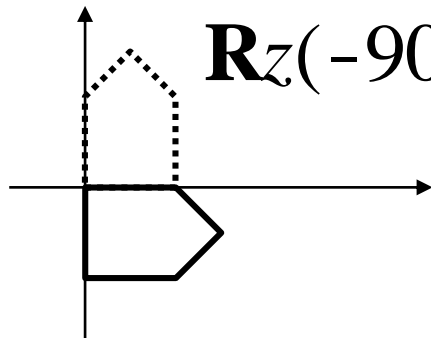
Composició de Transformacions

- Imaginem que volem

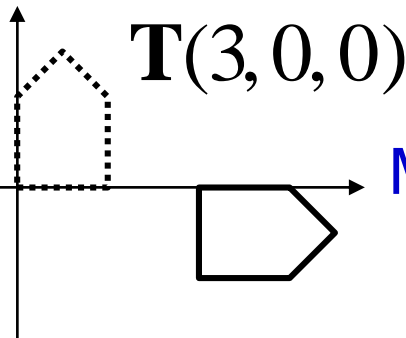


No es pot fer amb cap de les matrius anteriors

- Cal composar/efectuar dues transformacions



$R_z(-90^\circ)$



$T(3, 0, 0)$

&

$$M = T(3, 0, 0) \cdot R_z(-90^\circ)$$

$$P' = R_z(-90^\circ) \cdot P$$

$$P'' = T(3, 0, 0) \cdot P'$$

$$P'' = T(3, 0, 0) \cdot (R_z(-90^\circ) \cdot P) = (T(3, 0, 0) \cdot R_z(-90^\circ)) \cdot P = M \cdot P$$

Composició de Transformacions

$$\mathbf{T}(3,0) \cdot \mathbf{R}(-90^\circ) \neq \mathbf{R}(-90^\circ) \cdot \mathbf{T}(3,0)$$

2

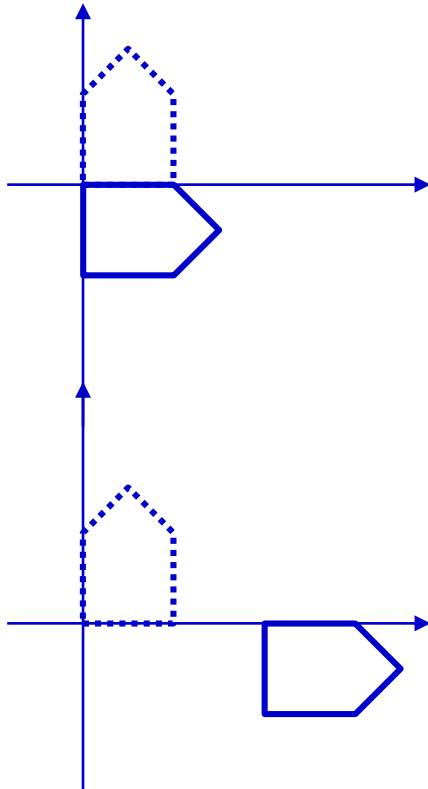
1

2

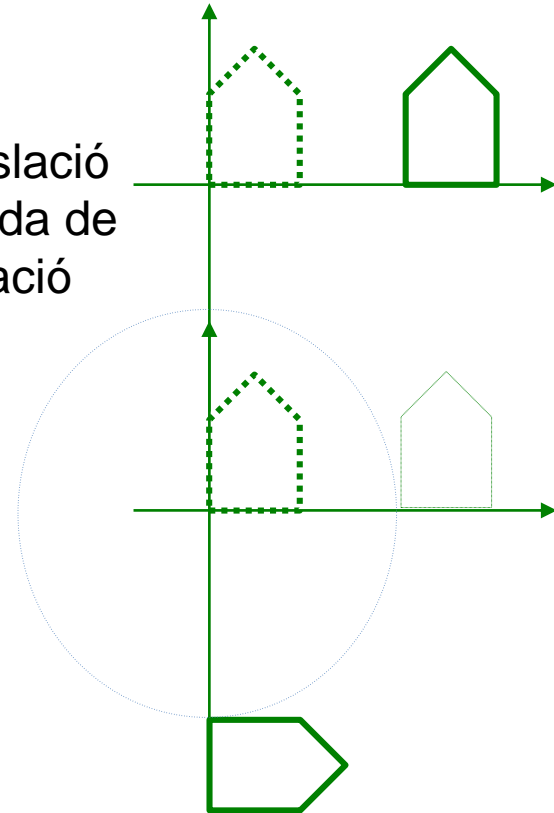
1

- Multiplicació de matrius no és commutativa

Rotació
seguida de
translació



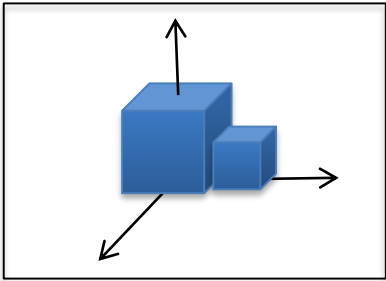
Translació
seguida de
rotació



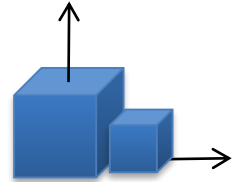
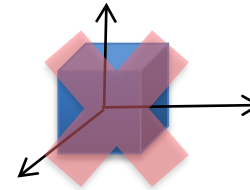
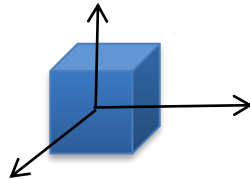
Classe 2: Contingut

- Introducció al procés de visualització
- Transformacions geomètriques
- **Exercicis**

Exemple simple de TG (1)



- Visualitzar una escena com la de la figura de l'esquerra
- Utilitzant el mètode `pinta_cub()` que indica a OpenGL que ha de visualitzar/pintar el VAO que conté el model de triangles d'aquest cub:



```
...  
CreaBuffers();
```

```
...  
modelMatrix_cub1();  
pinta_cub();
```

```
modelMatrix_cub2();  
pinta_cub();
```

TG1

TG2

VAO

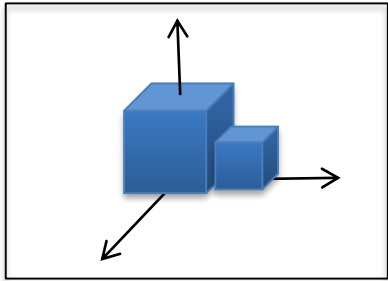
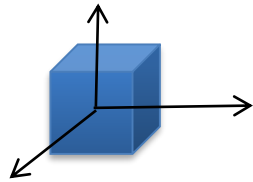
VBO

TG

OpenGL 3.3

Visualització

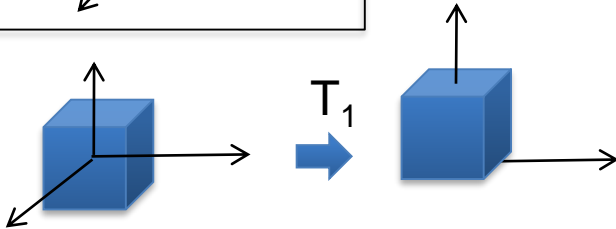
Exemple simple de TG (1)



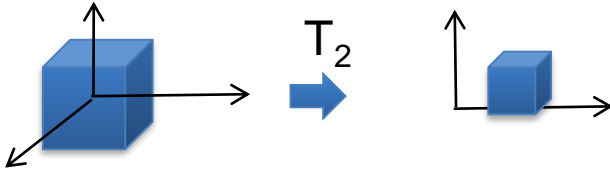
Escena a visualitzar

Pseudo-codi

```
TG = Translate (0, 0.5, 0);  
modelMatrix (TG);  
pinta_cub ();  
TG = Translate (0.75, 0.25, 0);  
TG = TG*Scale (0.5, 0.5, 0.5);  
modelMatrix (TG);  
pinta_cub();
```



$$T_1 = \text{Trans}(0, 0.5, 0)$$



$$T_2 = \text{Trans}(0.75, 0.25, 0) * S(0.5, 0.5, 0.5)$$

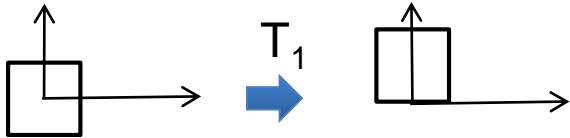
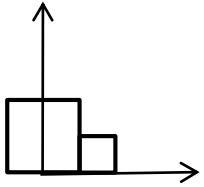
```
glm::mat4 TG = glm::mat4(1.f);  
TG = glm::translate (TG, glm::vec3(0,0.5,0));  
glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);  
pinta_cub ();
```

```
TG = glm::mat4(1.f);  
TG = glm::translate (TG, glm::vec3(0.75,0.25,0));  
TG = glm::scale(TG, glm::vec3(0.5,0.5,0.5));  
glUniformMatrix4fv (transLoc, 1, GL_FALSE, &TG[0][0]);  
pinta_cub();
```

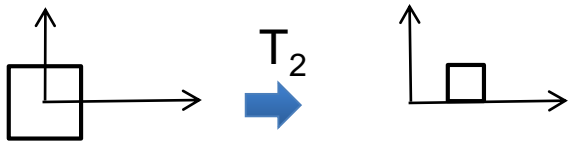
Exemple codi per pintar en CPU

Com faríeu per girar tota l'escena alfa graus respecte l'eix x?

Exemple simple (2): Girar escena repecte eix X alfa graus



$$T_1 = R_x(\text{alfa}) * \text{Trans}(0, 0.5, 0)$$



$$T_2 = R_x(\text{alfa}) * \text{Trans}(0.75, 0.25, 0) * S(0.5, 0.5, 0.5)$$

```
AUX = Rotate (alfa, (1,0,0));
```

```
TG = AUX * Translate(0, 0.5, 0);
```

```
modelMatrix (TG);
```

```
pinta_cub ();
```

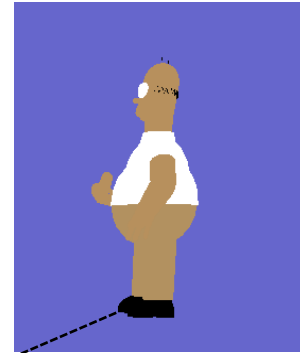
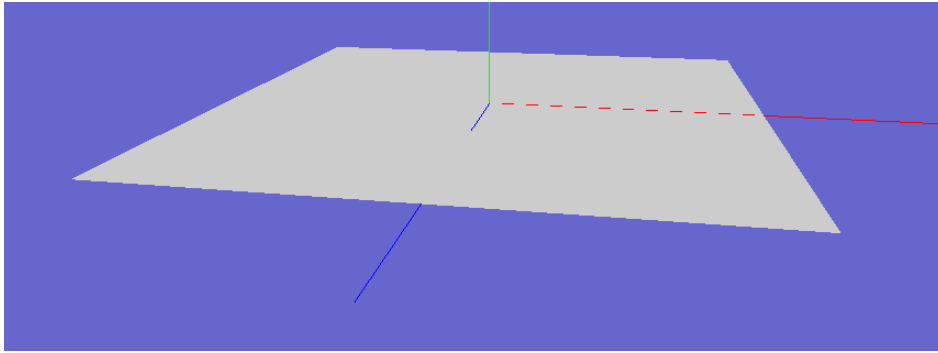
```
TG = AUX * Translate(0.75, 0.25, 0);
```

```
TG = TG * Scale(0.5,0.5,0.5);
```

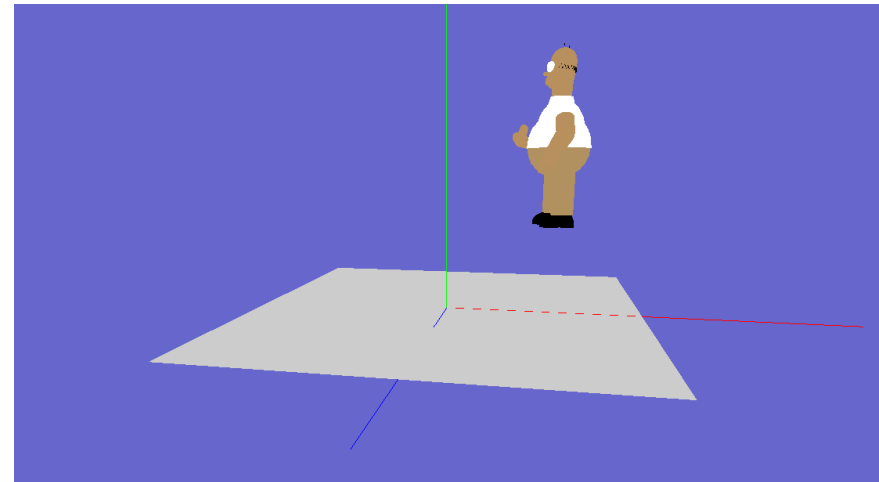
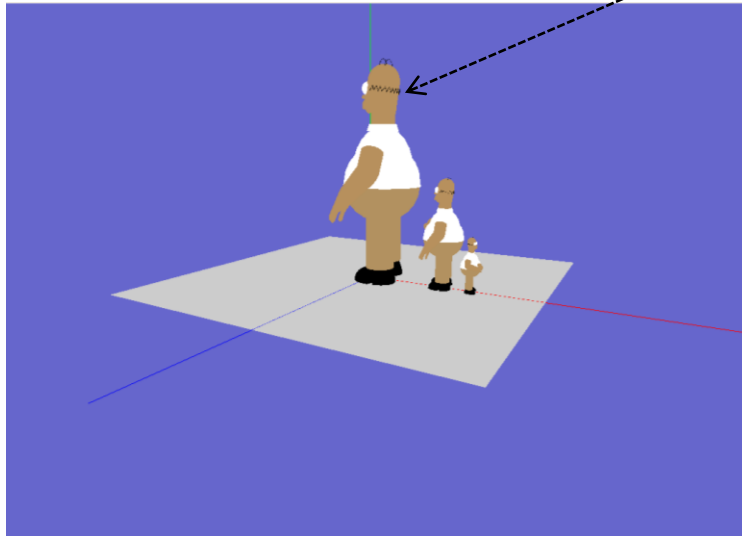
```
modelMatrix (TG);
```

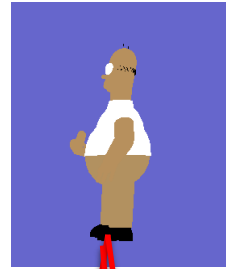
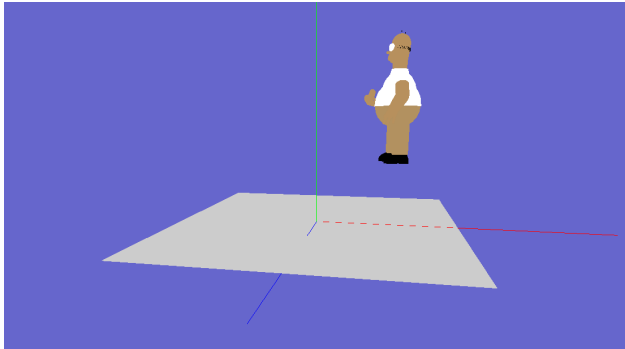
```
pinta_cub();
```

Exemple 3

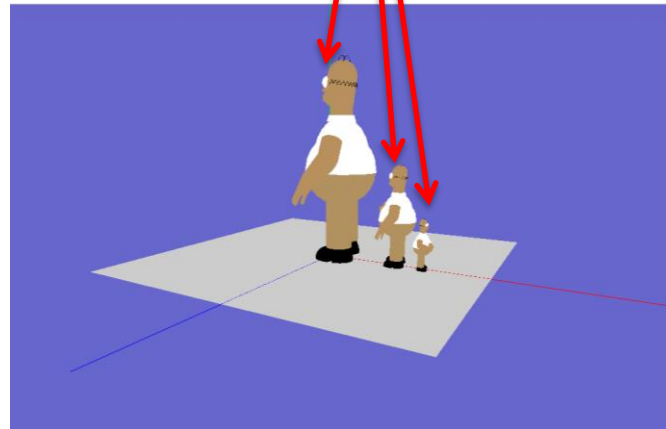


Mateixa grandària





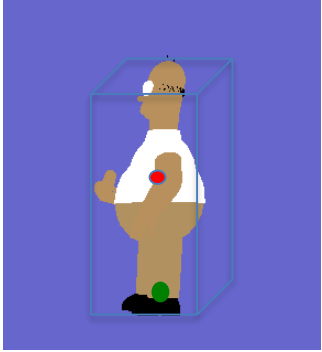
Només càlcul de TG



Transformació
geomètrica



Matriu 4x4
TG

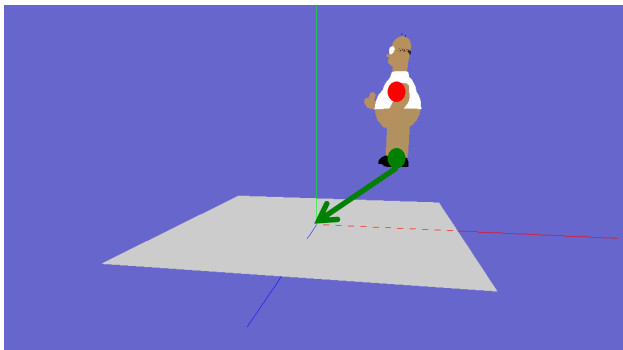


$CapsaMinCont = (xmin, ymin, zmin, xmax, ymax, zmax)$

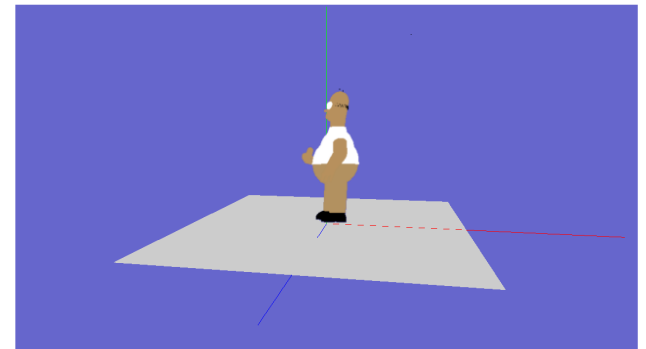
Mides $\Rightarrow a = (xmax - xmin)$, $h = (ymax - ymin)$, $f = (zmax - zmin)$

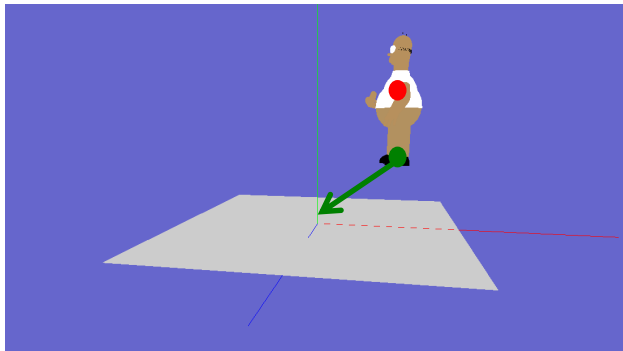
$CentBaseCapsa = (cbx, cby, cbz) = (xmin + xmax)/2, ymin, (zmin + zmax)/2)$

Els podem afegir com atributs al model geomètric



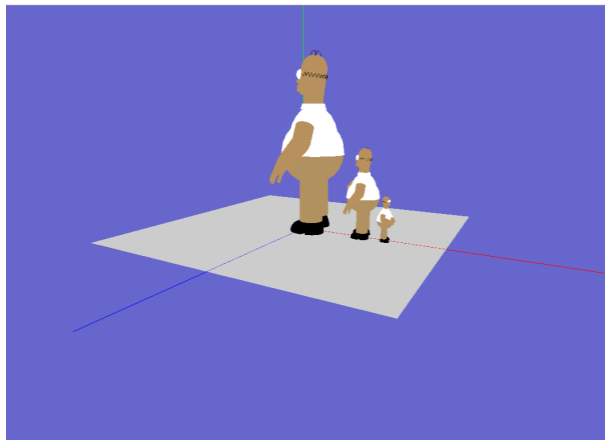
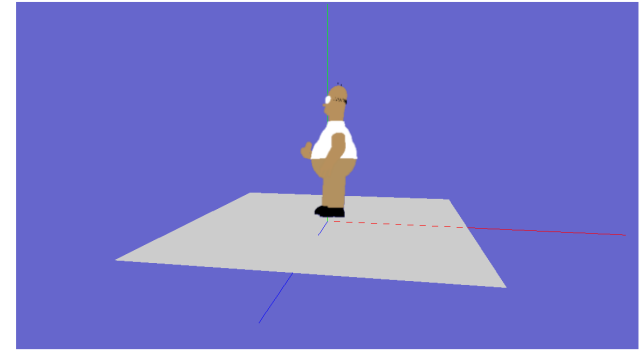
$t = (-cbx, -cby, -cbz)$
 $TG_{H1} = Trans(t)$



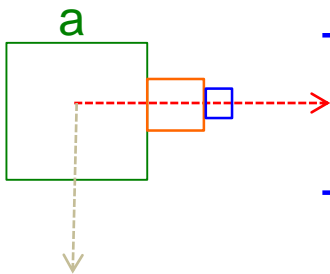


$$t = (-cbx, -cby, -cbz)$$

$$TG_{H1} = \text{Trans}(t)$$



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



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

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

Exemple pseudo-codi

per cada model

llegir_Model ();

crear_buffer_model ();

fper

//paintGL ();

per cada objecte_i

modelTransform_i (TG_i); //calcular TG

modelMatrix (TG_i); //envia "uniform"

pinta_model_i (); //visualitza model

fper

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

4

3

2

1

modelTransformHomer3()

//tercer homer

{

TG = I;

TG = TG * Translate (posx, posy, posz));

TG = TG * Scale (s, s, s);

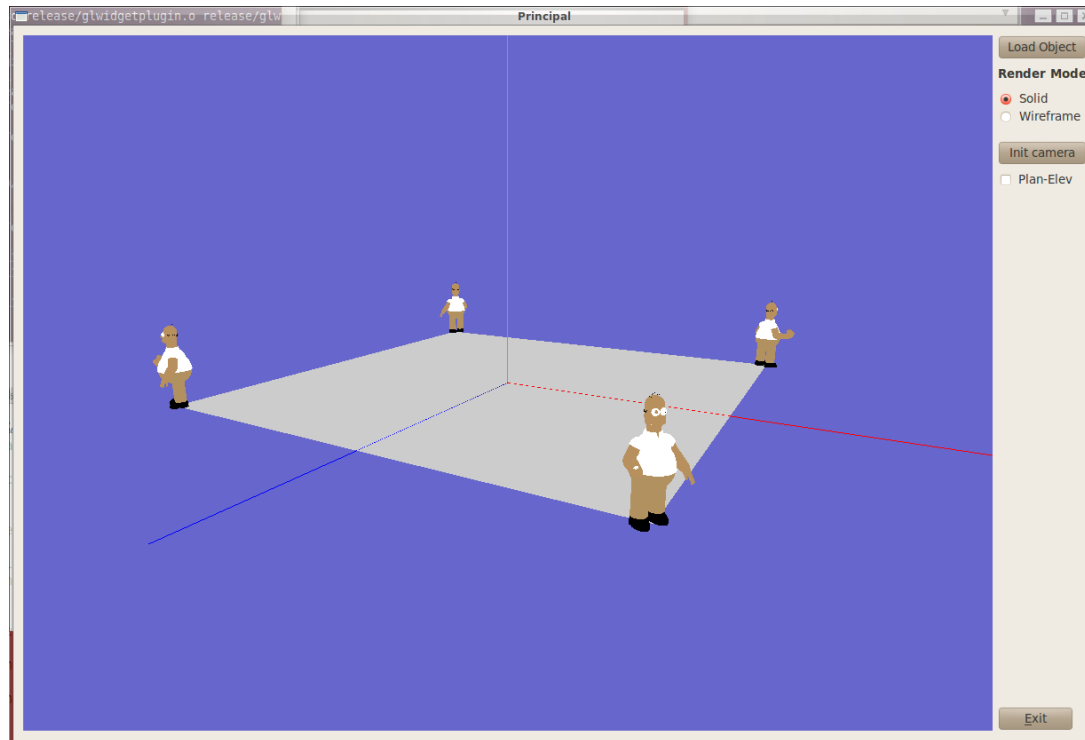
TG = TG * Rotate (-180, (0, 1, 0));

TG = TG * Translate (-cb.x, -cb.y, -cb.z);

modelMatrix (TG); //enviar uniform

}

Exercicis



Mireu la col·lecció de problemes del racó.
Proposta de mínims: 16, 19, 24, 25 de la col·lecció

Classe 2: Conceptes

- Quins processos bàsics formen part del procés de visualització projectiu
- Perquè cal definir una càmera i els seus paràmetres
- Vertex Shader i Fragment Shader
- Transformacions geomètriques bàsiques: entendre i conèixer-les
- Composició transformacions: entendre i importància de l'ordre de multiplicació.
- Saber calcular TG i especificar en codi/pseudo-codi
- VAO, VBO i aplicació de TG en Vertex Shader
- Orde d'enviament de TG a la GPU