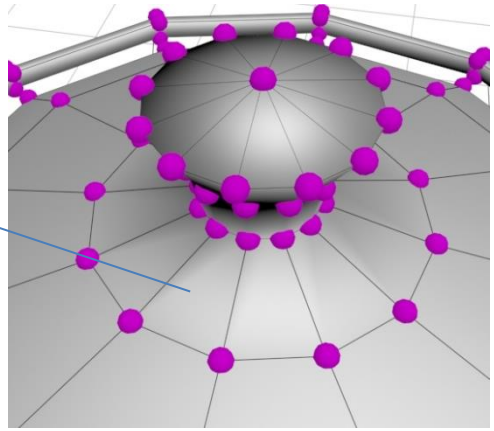
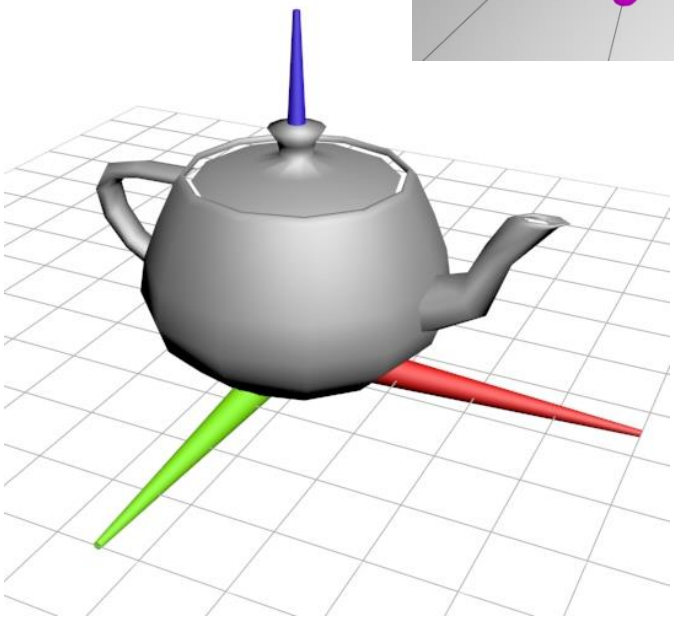
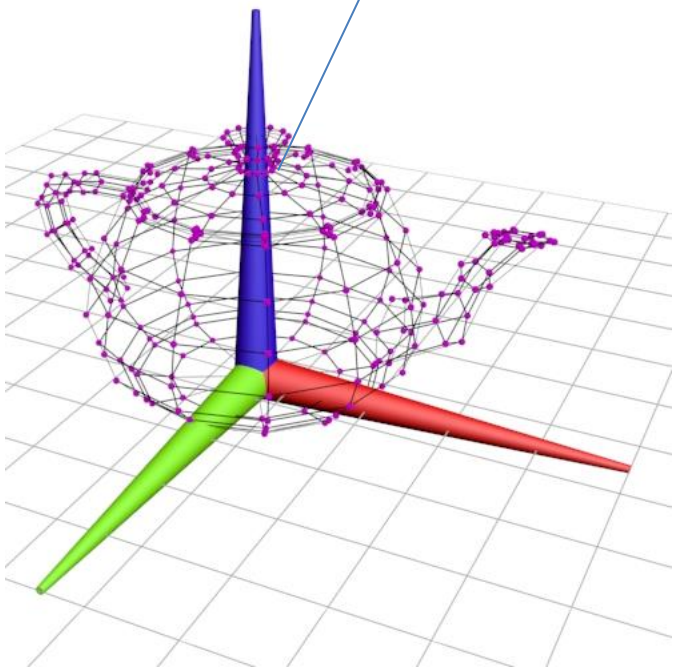
Modelos 3D – Vértices / triángulos

Triángulos



Vértice X,Y,Z





Matríces

- Hablamos de transformaciones lineales representables con matrices de 4 x4

$$\begin{bmatrix} Vx' \\ Vy' \\ Vz' \\ Vw' \end{bmatrix} = \begin{bmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \\ m41 & m42 & m43 & m44 \end{bmatrix} * \begin{bmatrix} Vx \\ Vy \\ Vz \\ 1 \end{bmatrix}$$

Transformaciones típicas

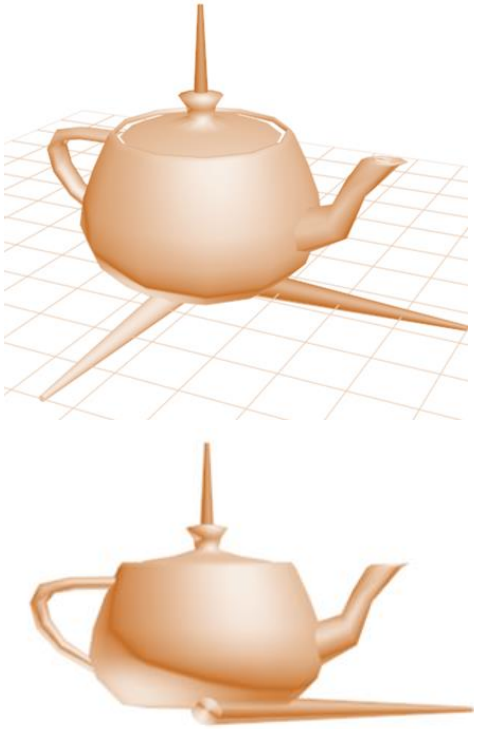
- Modelado
  - Trasladar (X, Y, Z) Mt
  - Rotar(ángulo, EjeX, EjeY, EjeZ) Mr
  - Escalar(X, Y, Z) Me
- Vista
  - Es una combinación de traslaciones y rotaciones
    Mv= T * R * T
- Proyección
  - Perspectiva, Ortográfica Mp

Sistemas de Coordenadas

- Matríz de Modelado
  - De Coord. Modelado -> Coord. de Mundo
- Matríz de Vista
  - De Coord. de Mundo -> Coord. De Camara
- Matríz de Proyección
  - De Coord de Cámara -> Coord de Pantalla (clipping)

Matríces de Modelado

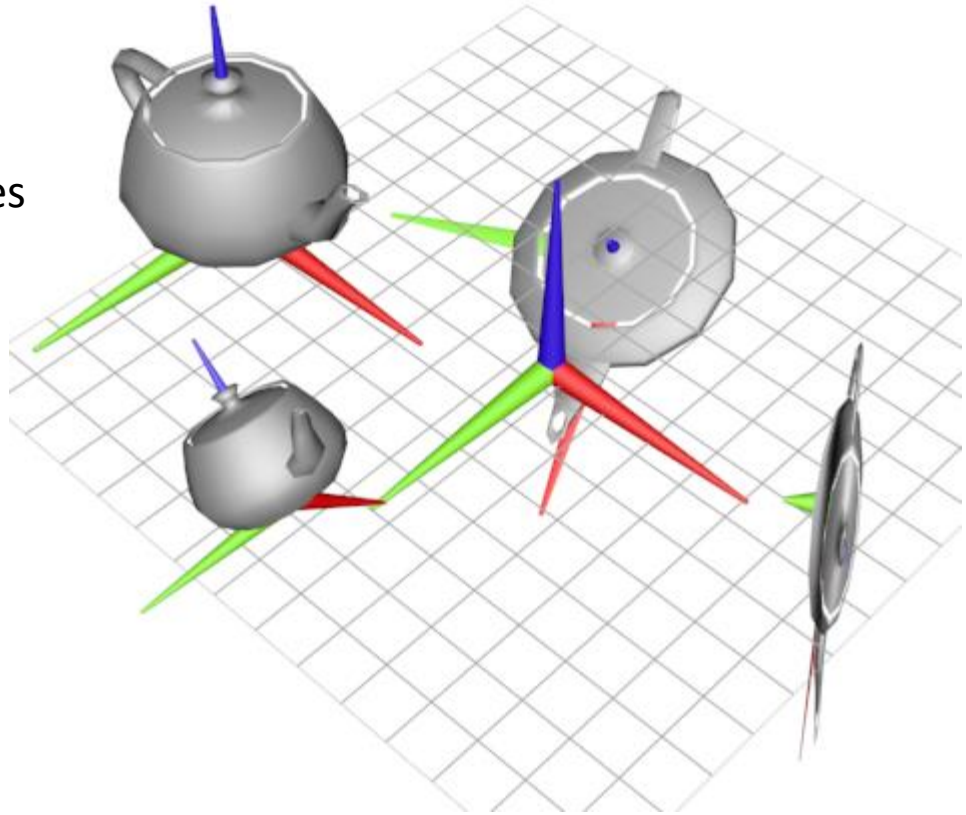
Sistema de coordenadas
de modelado



Modelo

Transformaciones
De Modelado

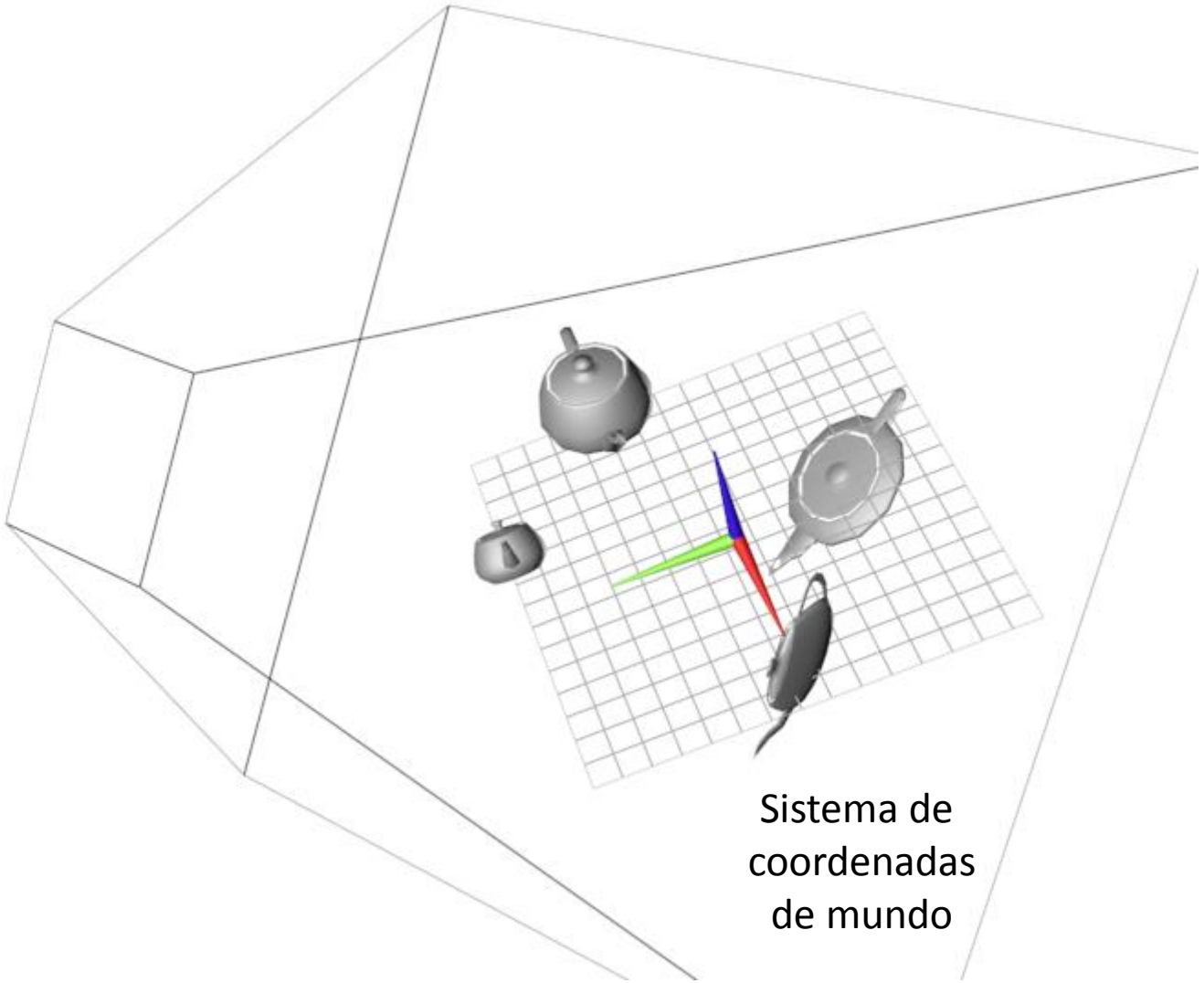
Sistema de coordenadas
de mundo



IMPORTANTE: Hay 4 instancias del modelo “tetera”
cada instancia requiere una matríz de modelado distinta

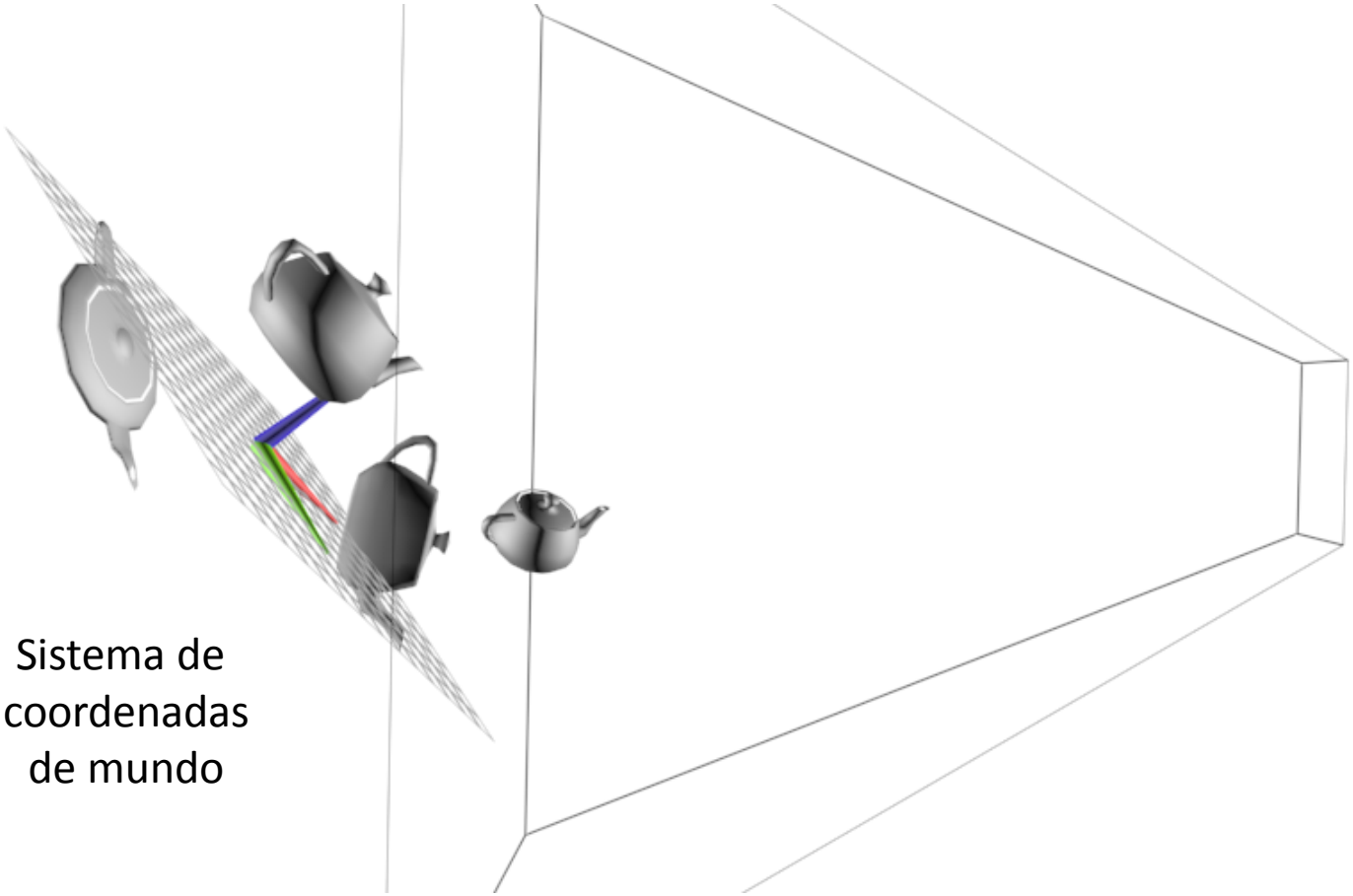
Matríz de Vista

Sistema de
coordenadas
de vista
(origen de cámara)

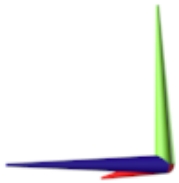


Sistema de
coordenadas
de mundo

Matríz de Vista



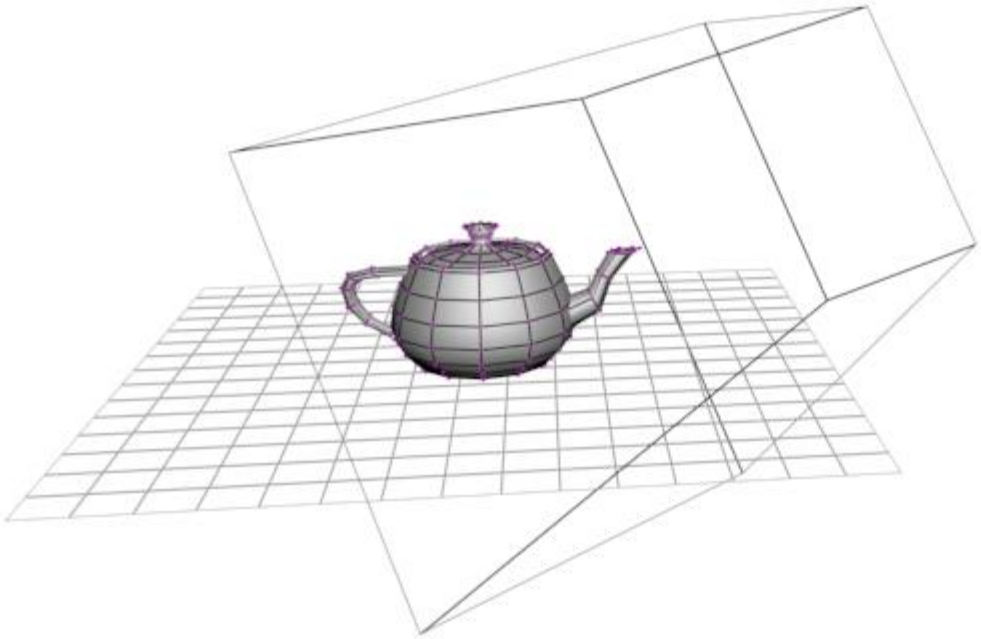
Sistema de coordenadas de mundo



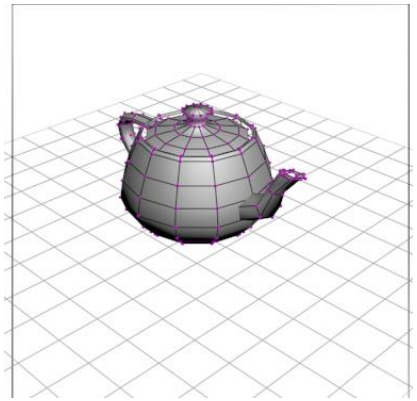
Sistema de coordenadas de vista (origen de cámara)

Cálculo de Matríz de Vista

```
mat4.lookAt(outputMat,
    cameraPosition, // posición de la cámara en el mundo
    cameraTarget,    // punto al cual mira la cámara en el mundo
    upVector         // si Z es arriba, comunmente glm::vec3(0,0,1)
);
```



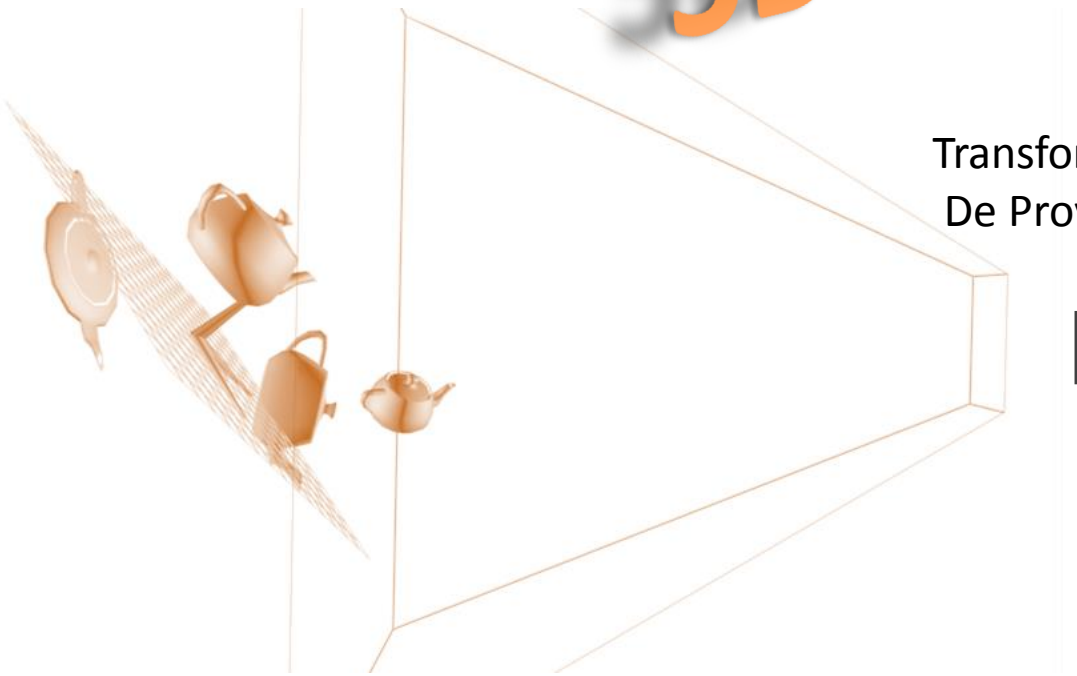
Vista en Pantalla



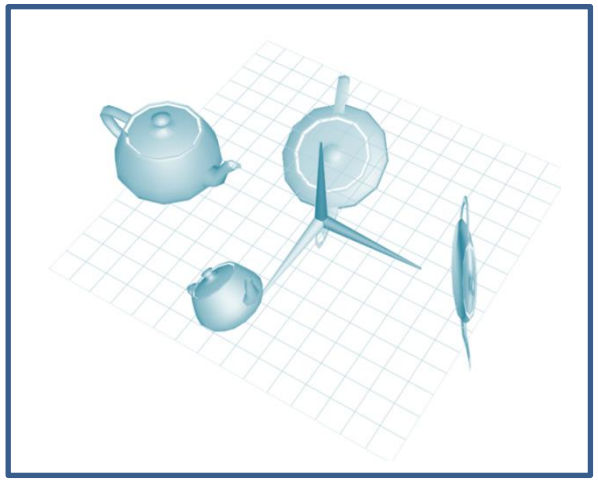
Matríz de Proyección

3D

2D



Transformación
De Proyección



X,Y,Z

X,Y

Librería gl-matrix.js **(glmatrix.net)**

Funciones matemáticas para el manejo de Matrices y Vectores. Hay dos versiones disponibles 1.x y 2.x

Formato general de las funciones sobre matrices

mat4.<operación>(output, input ,parámetros)

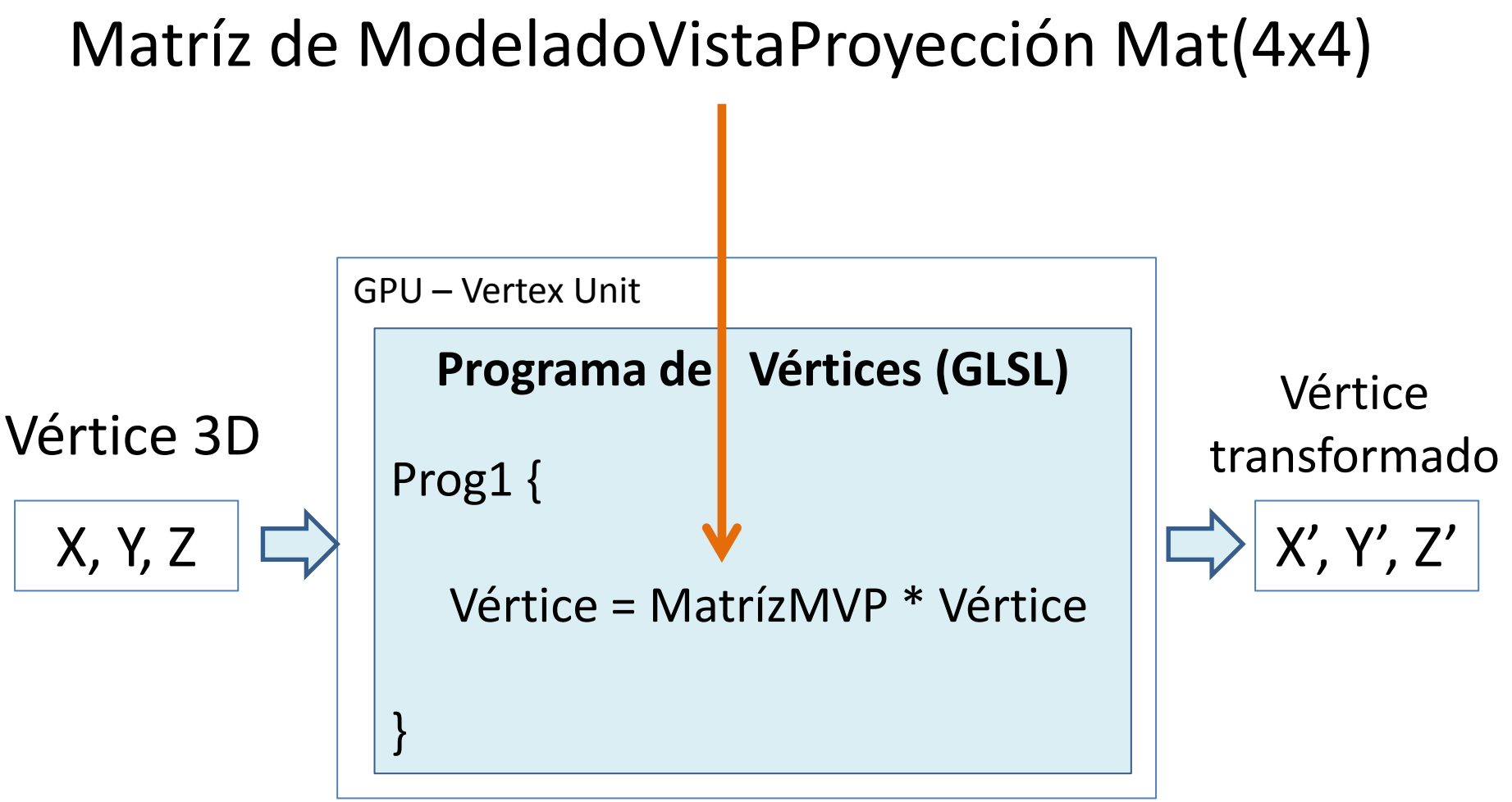
[output] = [input] x operación (multiplica a derecha)

mat2,mat3,mat4,vec2,vec3,vec4

Librería gl-matrix.js **(glmatrix.net)**

```
var m1 = mat4.create();                          // m1=identidad

mat4.translate(m1, m1, [0, 0, 10]);              // trasladar -10 en Z
mat4.rotate(m1, m1, Math.PI/2, [0, 1, 0]);       // rotar 90 grados
mat4.scale(m1, m1, [2, 2, 2]);                   // escalar 200%

//        m1 = MtrasZ10 x Mrot90 x Mesc2

var m2 = mat4.create();                          // m2=identidad
mat4.scale=(m2,m1,[5,5,5]);                      // escalar 500%

//        m2 = m1 x Mesc5
//        m2 = MtrasZ10 x Mrot90 x Mesc2 x Mesc5
```

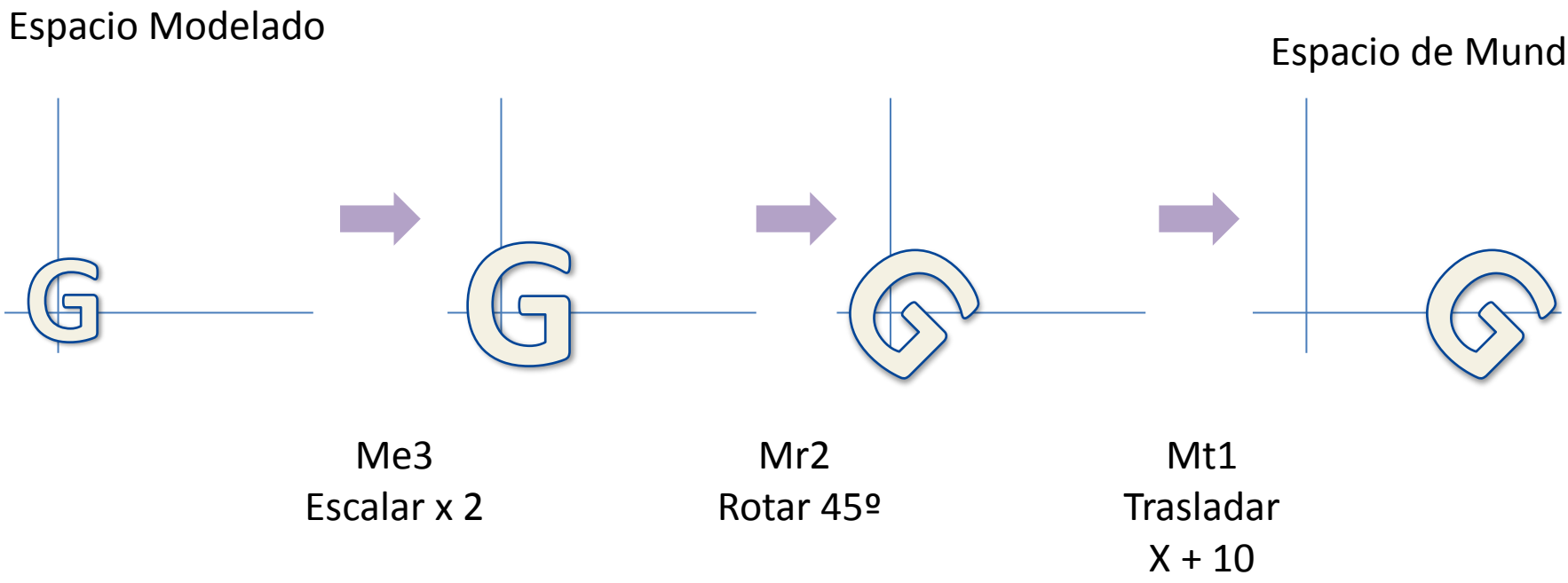
El procesador de Vértices



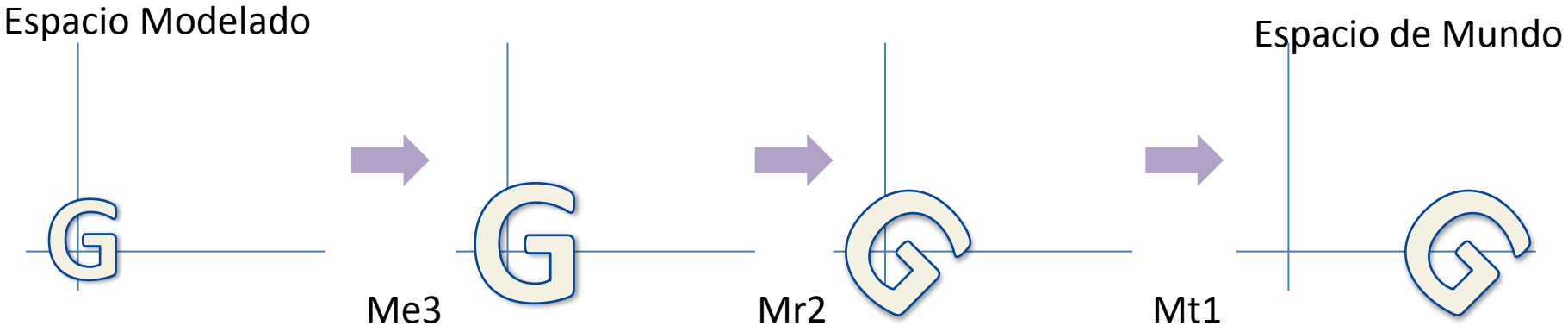
Concatenación de transformaciones

$$Mt = Mt1 * Mr2 * Me3$$

$$V' = Mt * V$$



Concatenación de transformaciones



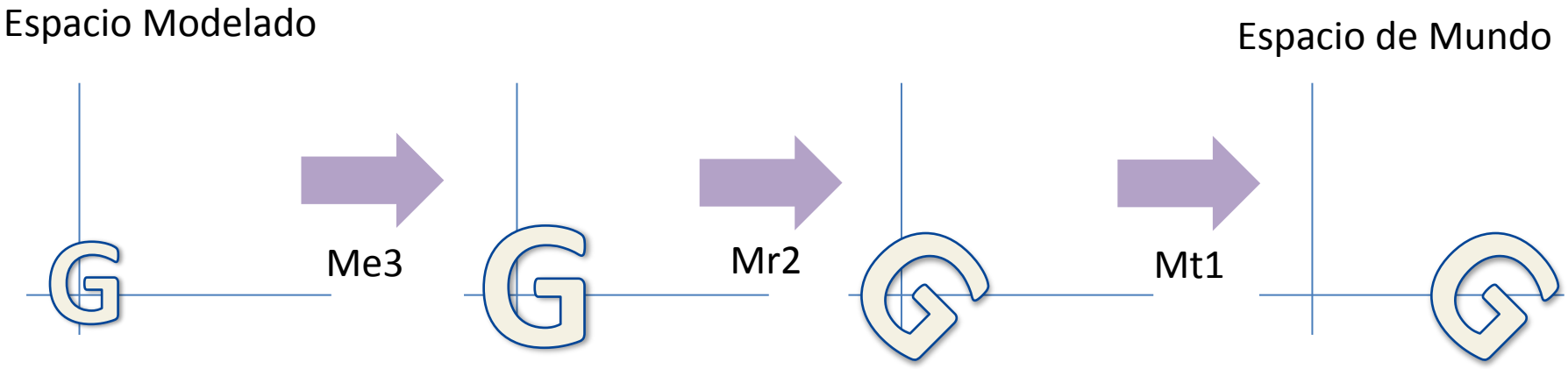
Secuencia de transformaciones

Paso 1) Escalar (2,2,2) duplicar escala

Paso 2) Rotar (45,0,0,1) 45º respecto de Z

Paso 3) Trasladar (10,0,0) trasladar 10 en X

Concatenación de transformaciones



Código JS (atención!!! El último paso se aplica primero)

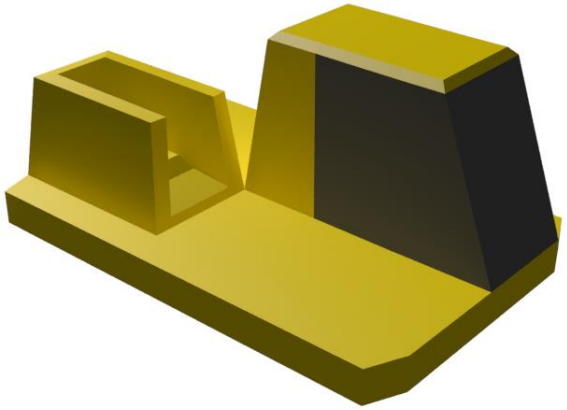
```
m1=mat4.create()
mat4.translate(m1,m1, [trX,trY,trZ]])
mat4.rotate(m1,m1,ang, (ejeX,ejeY,ejeZ))
mat4.scale(m1, m1,(escX,escY,escZ))
dibujarModelo(m1);
```

m1=Identidad
m1=Mt1
m1=Mt1 x Mr2
m1=Mt1 x Mr2 x Me3

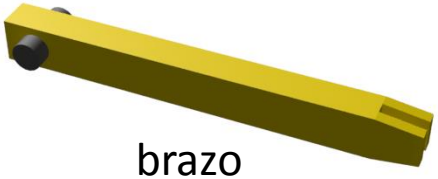
Caso práctico – Escena 3D

tuerca

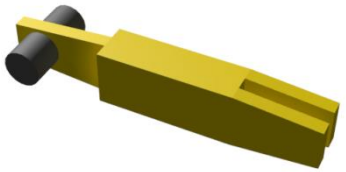
eje



cabina



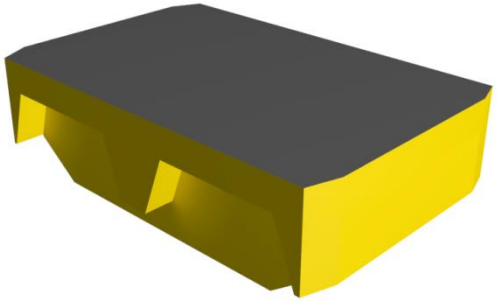
brazo



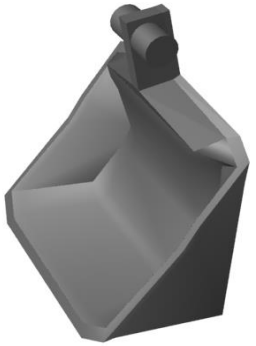
antebrazo

cubierta

llanta



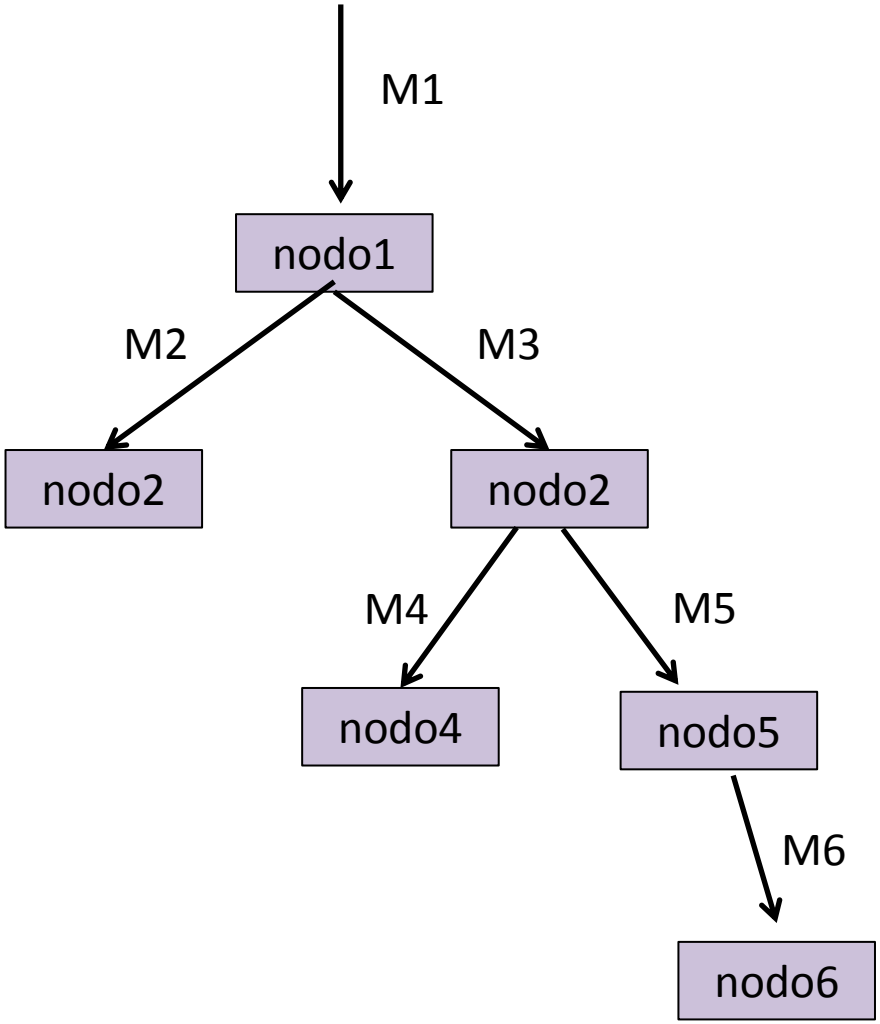
chasis



pala

Modelos de la escena

Árbol de la escena



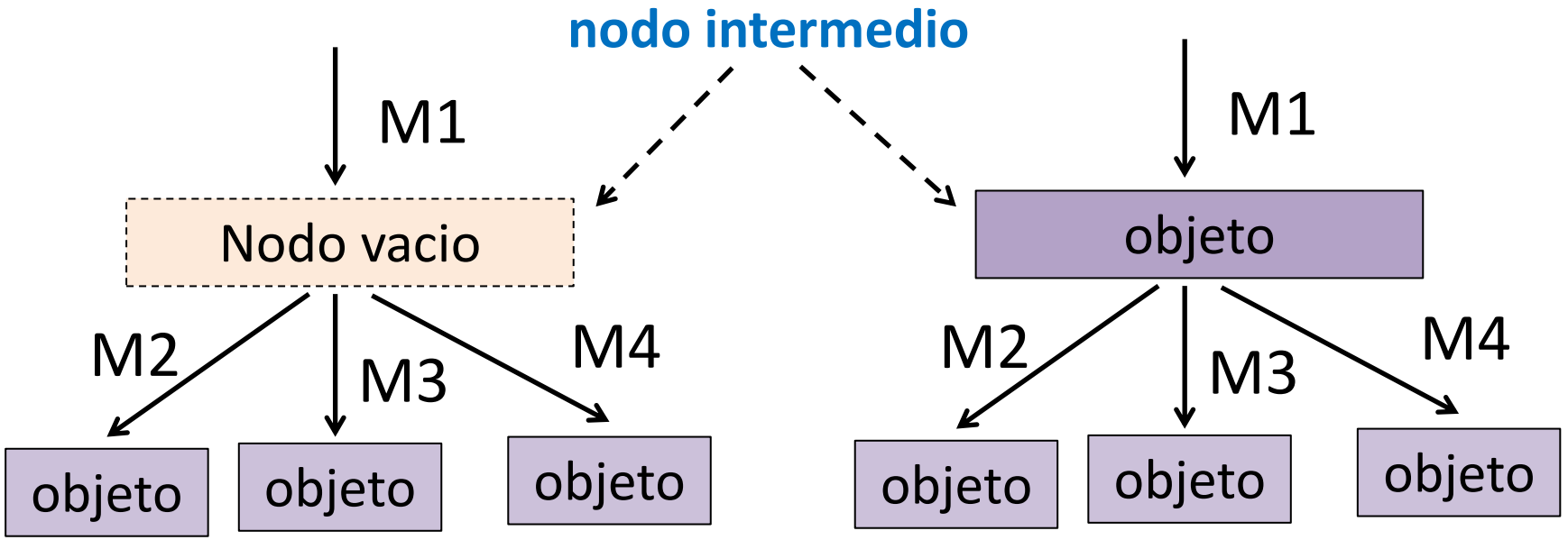
Matriz final del nodo 6

M = M1 x M3 x M5 x M6

Matriz final del nodo 4

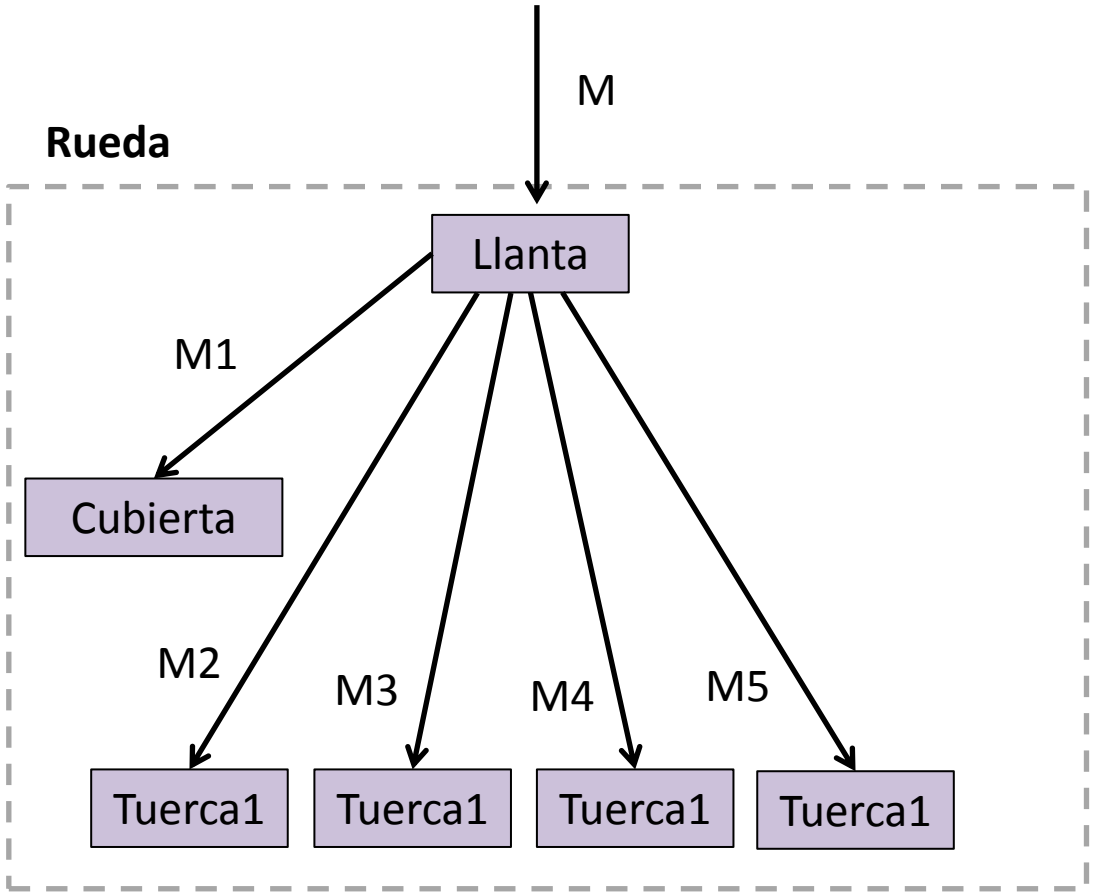
M = M1 x M3 x M4

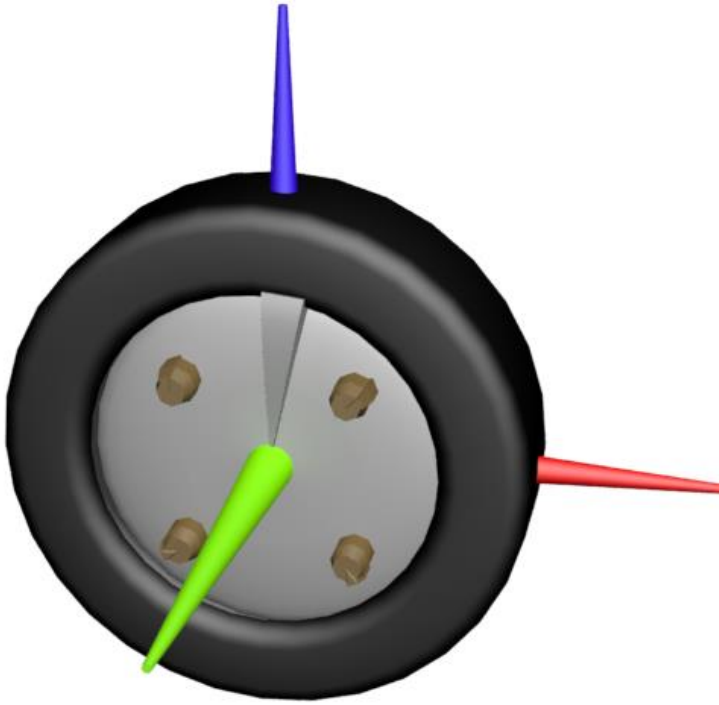
Subárboles y Objetos Compuestos

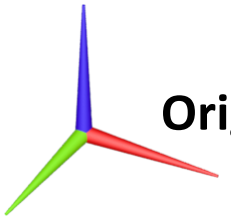


El nodo intermedio puede tener o no una representación visual

Subárbol - Rueda



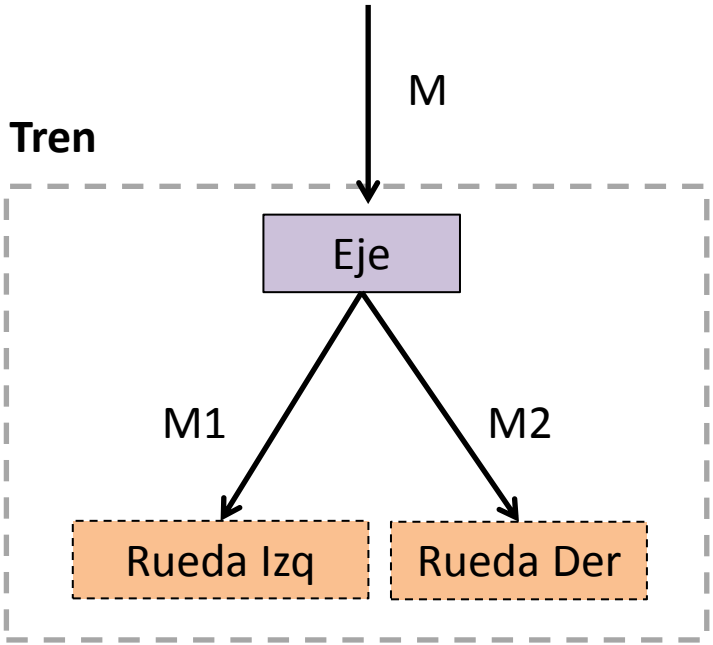


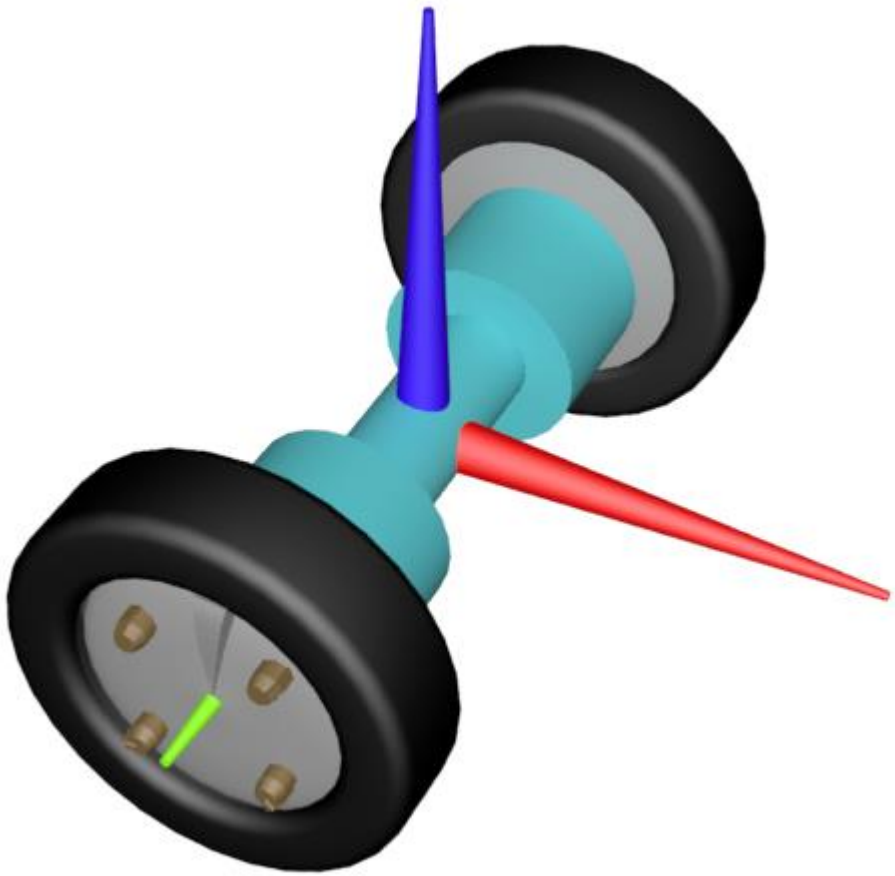


Origen de coord. de mundo

M2 = MrotY(45) x Mtras(-d,0,0) d=distancia al centro

Subárbol - Tren



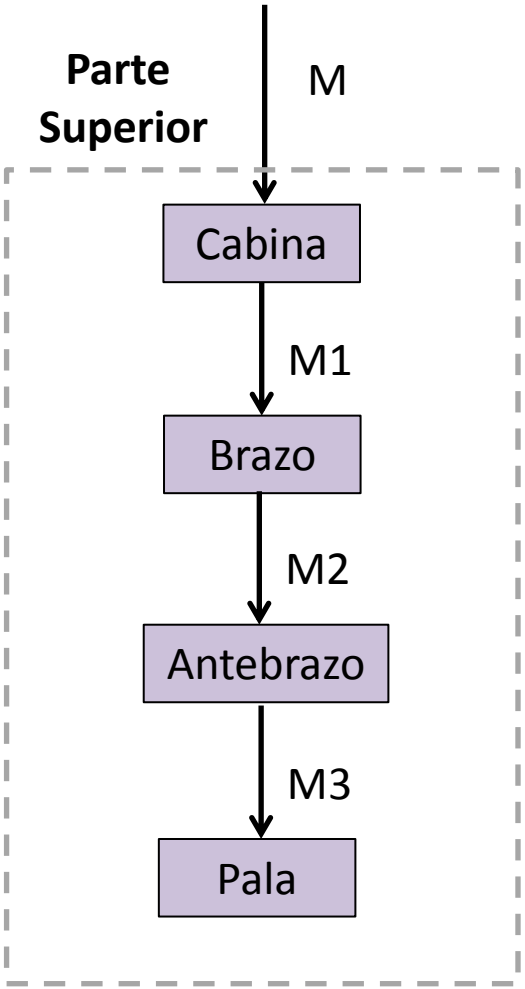


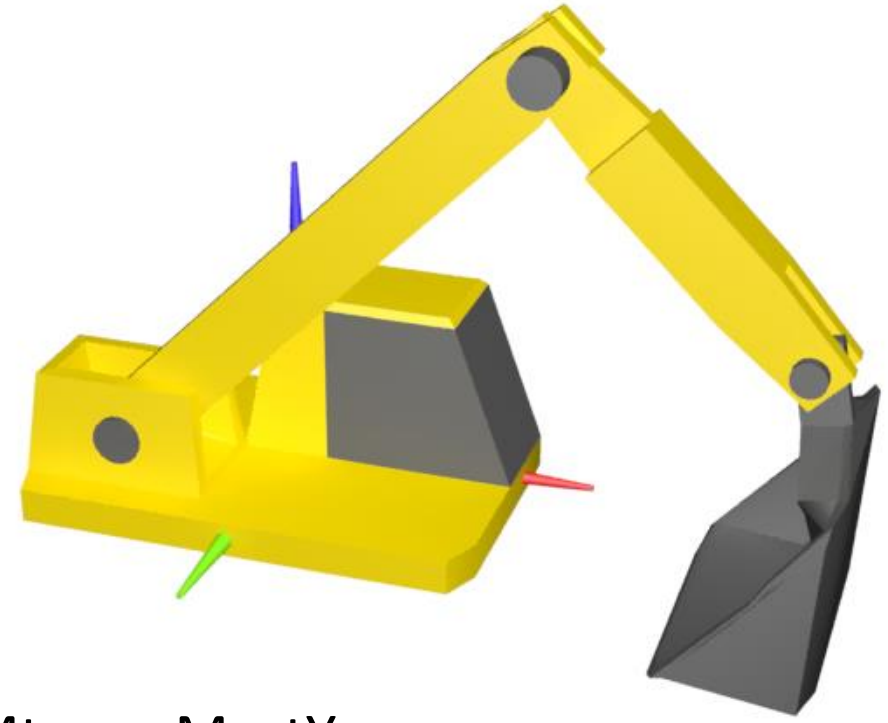
M1= Mtras(-d1,0,0)

M2= Mtras(-d1,0,0) x MRotZ (180º)

d1= distancia desde el centro del eje
a la posición de la rueda

Subárbol - Parte Superior





M1= Mtras x MrotY

M2= Mtras x MrotY

M3= Mtras x MrotY

Dibujar recursivamente

```
Objeto3D.dibujar(matriz);

Function dibujar(matriz){

   // comandos webgl para dibujar el buffer asociado al objeto
   // con la matriz provista por parámetro

   for (i=0; i<CantNodosHijos; i++){
       hijos[i].dibujar(matriz)
   }

}
```

Dibujar Vehículo

```
chasis.dibujar()
    cabina.dibujar()
        brazo.dibujar()
            antebrazo.dibujar()
                pala.dibujar()
    eje.dibujar()                   // eje delantero
        llanta.dibujar()            // rueda delantera izquierda
            cubierta.dibujar()
            tuerca.dibujar()
            tuerca.dibujar()
            tuerca.dibujar()
            tuerca.dibujar()
        llanta.dibujar()            // rueda delantera derecha
            tuerca.dibujar()
            ...
```