Transformaciones con WebGL

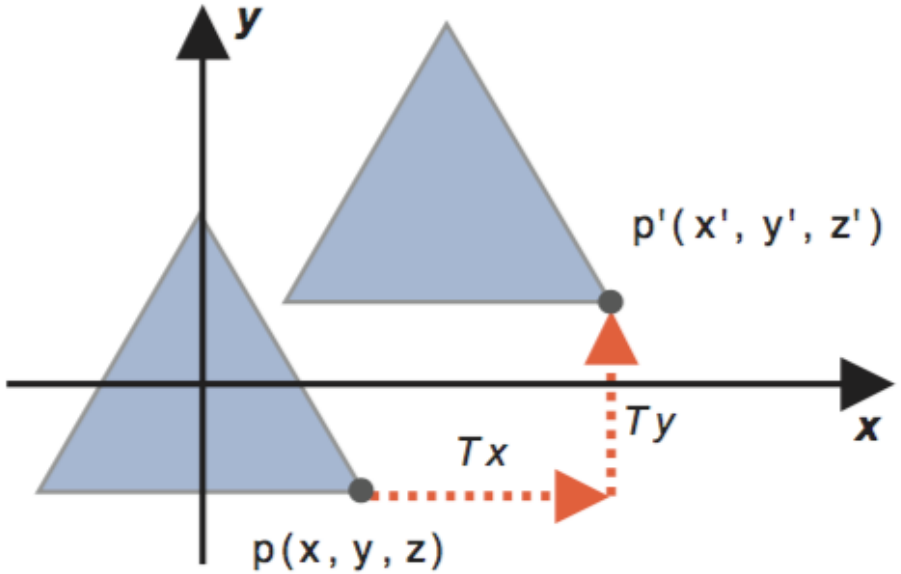
Katia Leal Algara
Web: http://gsyc.urjc.es/~katia/
Email: katia.leal@urjc.es

Dept. Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación (GSyC)
Escuela Superior De Ingeniería De Telecomunicación (ETSIT)
Universidad Rey Juan Carlos (URJC)

Translation

- Las coordenadas del nuevo punto **p'** se calculan por medio de:

$$x' = x + Tx$$
$$y' = y + Ty$$
$$z' = z + Tz$$



Translation

- ¿Dónde debemos implementar esta operación?
- … en el vertex shader. Se deben pasar las distancias de movimiento Tx, Ty y Tz al vertex shader, se debe aplicar la ***ecuación de movimiento*** y asignar el resultado a `gl_Position`.

Introducción a Matrices

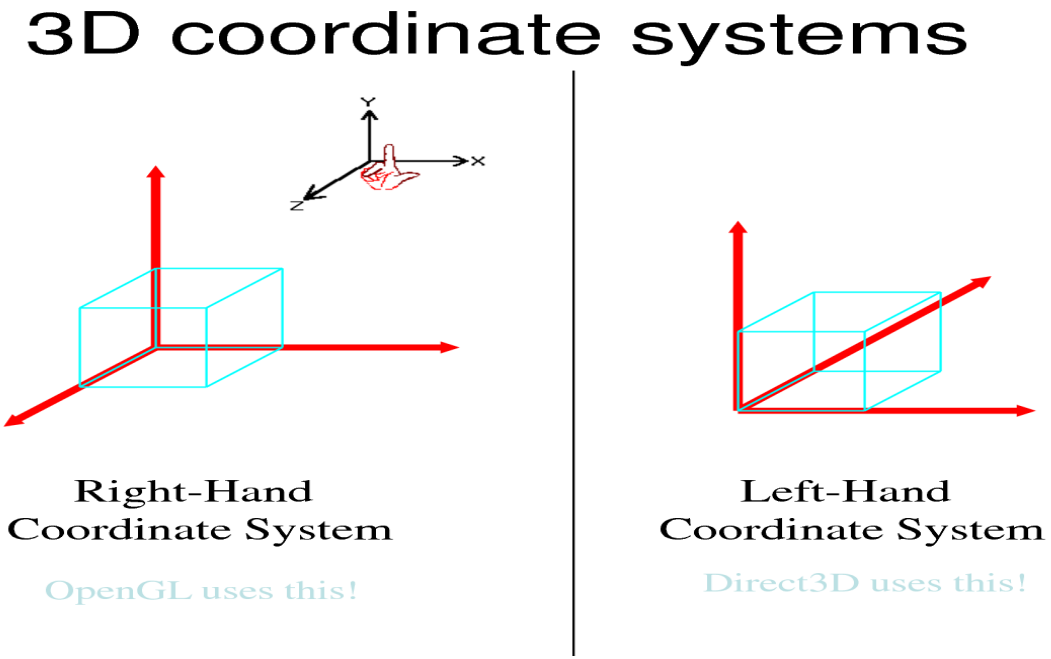
WebGL utiliza matrices (Matrix).

- Las matrices definen las transformaciones del objeto (desplazamiento, escalado, etc.).
- Las matrices describen el tipo de cámaras.
- Las matrices describen la configuración actual del espacio 3D.

Introducción a Matrices

Sistema de coordenadas en OpenGL

- De “mano derecha”:



Transformaciones básicas en 2D

Las transformaciones se pueden combinar utilizando álgebra básica.

- **Translación:**

  $x' = x + tx$

  $y' = y + ty$

- **Escalado:**

  $x' = x * sx$

  $y' = y * sy$

- **Cizalla (Shear):**

  $x' = x + hx*y$

  $y' = y + hy*x$

- **Rotación:**

  $x' = x*\cos\Theta - y*\sin\Theta$

  $y' = x*\sin\Theta + y*\cos\Theta$

Representación matricial

- Una transformación se puede representar por una matriz:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Multiplicando la matriz por una vector columna, estamos aplicando una transformada a un vértice.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{aligned} x' &= ax + by \\ y' &= cx + dy \end{aligned}$$

Uso de matrices

- Ejemplo con escalado:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$$

- En forma matricial:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Representación matricial

Por lo tanto:

- X-axis = pulgar = 1, 0, 0
- Y-axis = índice = 0, 1, 0
- Z-axis = medio = 0, 0, 1

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

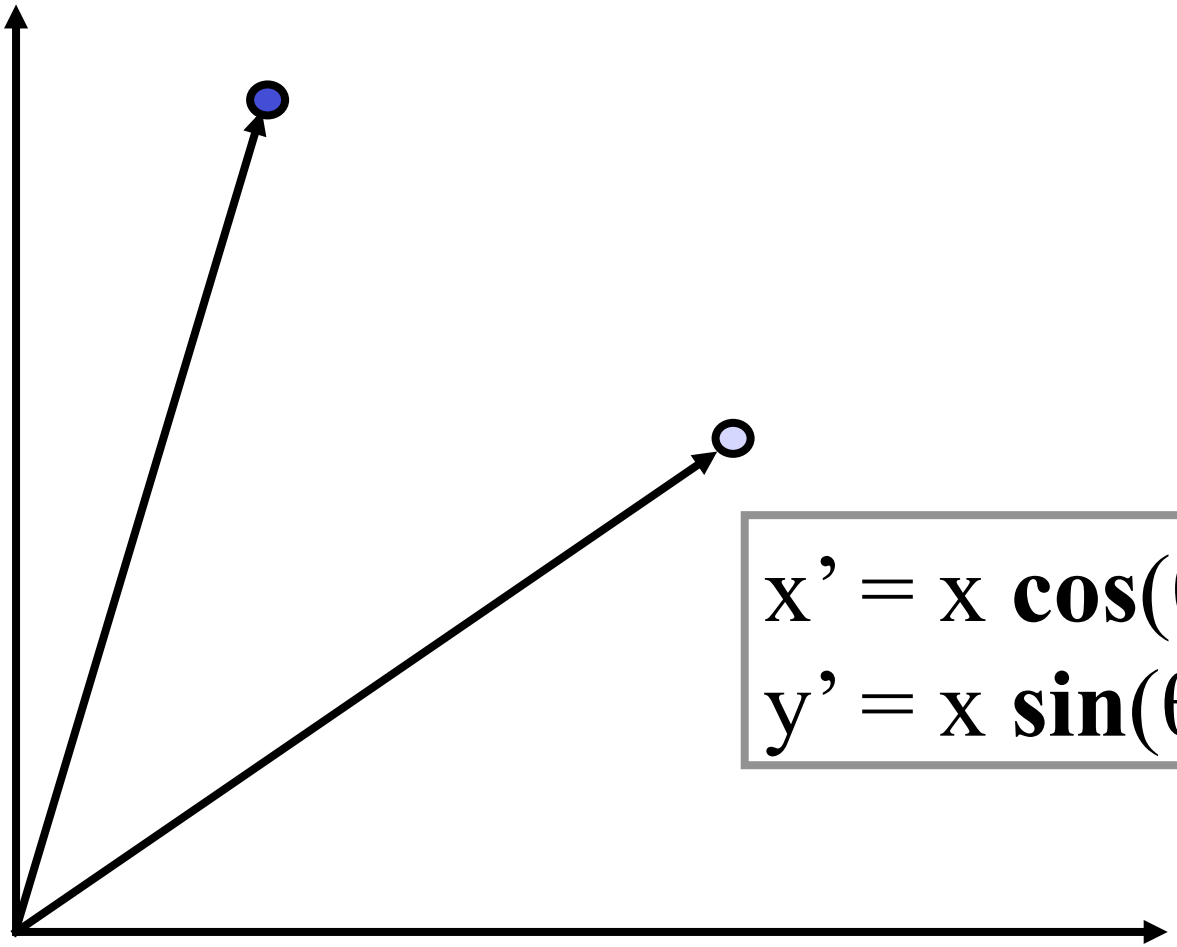
Representación matricial

- Las transformaciones se concatenan al multiplicar matrices:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Las matrices son muy útiles para representar transformaciones sucesivas.

Rotación en 2-D



$$x' = x\ \mathbf{cos}(\theta) - y\ \mathbf{sin}(\theta)$$
$$y' = x\ \mathbf{sin}(\theta) + y\ \mathbf{cos}(\theta)$$

Rotación 2-D

- Esto lo podemos expresar en una matriz:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Incluso si sin(θ) y cos(θ) no son funciones lineales de θ,

  x' es una combinación lineal de x e y

  y' es una combinación lineal de x e y

- Una matriz es un operador lineal.

Matriz de transformación

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

- ¿Qué opináis? ¿Guardamos la matriz de transformación o el resultado final de aplicarla?

Matrices de 2x2

- ¿Qué podemos representar con matrices de 2x2?

- **Identidad** en 2D:

$$x' = x \qquad y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- **Escalado** en 2D:

$$x' = s_x * x \qquad y' = s_y * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrices de 2x2

- ¿Qué podemos representar con matrices de 2x2?

- **Rotaciones** en 2D:

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- **Cizalla** en 2D:

$$x' = x + sh_x * y$$
$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrices de 2x2

- ¿Qué podemos representar con matrices de 2x2?

- **Espejo** sobre el eje Y:

$$x' = -x \qquad y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- **Espejo** sobre el origen:

$$x' = -x \qquad y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Transformaciones lineales

- Las transformaciones lineales son una combinación de:
  - Escalado,
  - Rotación,
  - Cizalla, y
  - Espejo.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

¿Y qué hacemos con la translación?

- ¿Se puede representar una translación con una matriz 2x2?

$$x' = x + t_x$$
$$y' = y + t_y$$

NO!

Las matrices 2x2 únicamente se pueden utilizar para representar transformaciones lineales.

Coordenadas homogéneas

- Coordenadas homogéneas:
  - Representan coordenadas en 2D pero utilizan un vector de 3D.
  - Fueron introducidas por el matemático alemán August Ferdinand Möbius en el año 1837.

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Aunque no son nada intuitivas son de gran utilidad para los sistemas gráficos!!

Coordenadas homogéneas

- ¿Y de qué nos sirve para la translación?

$$x' = x + t_x$$
$$y' = y + t_y$$

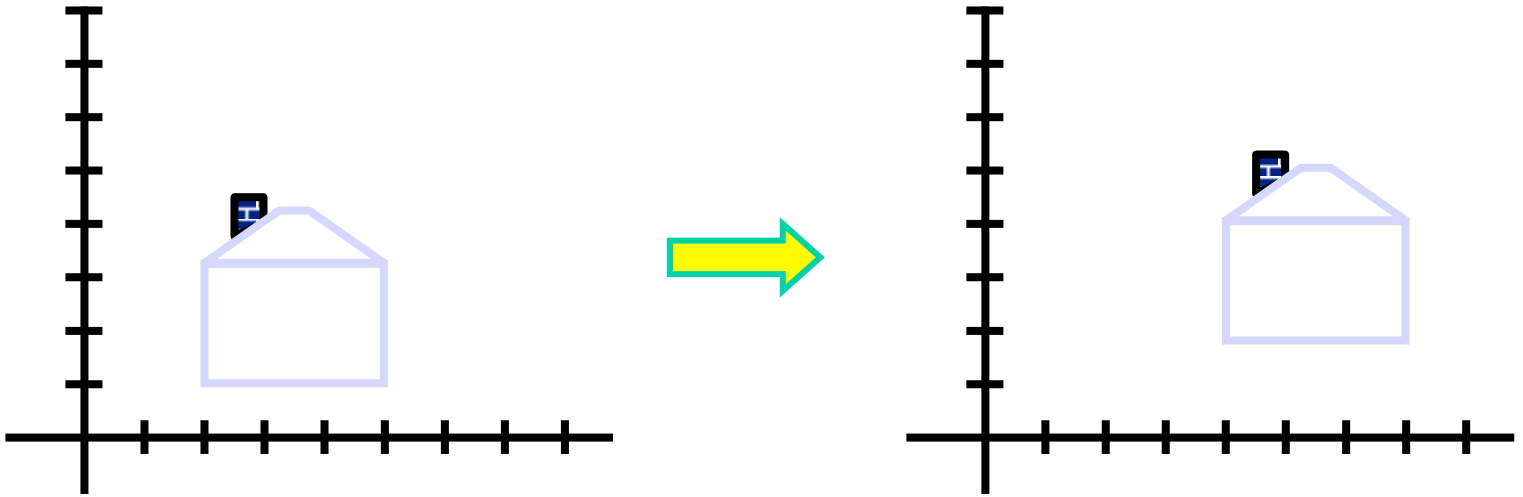
- De mucho!!

$$\mathbf{T}ranslation = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translación

- Ejemplo:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



Transformaciones básicas en 2D

- Las transformaciones básicas en 2D se realizan utilizando matrices de 3x3

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translación

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Escalado

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotación

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Shear

Transformaciones en 3D

- Utilizamos la misma idea que en 2D
  - Coordenadas homogéneas: (x, y, z, w).
  - La matriz de transformación es de 4x4.

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Transformaciones en 3D

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Identidad

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ \boldsymbol{z'} \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 & 0 \\ 0 & 0 & \boldsymbol{s}_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Escalado

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ \boldsymbol{z'} \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{t}_x \\ 0 & 1 & 0 & \boldsymbol{t}_y \\ 0 & 0 & 1 & \boldsymbol{t}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Translación

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Espejo

Transformaciones en 3D

Giro sobre el eje Z:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Giro sobre el eje Y:

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ \boldsymbol{z}' \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ \boldsymbol{z} \\ \boldsymbol{w} \end{bmatrix}$$

Giro sobre el eje X:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta & 0 \\ 0 & \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Example 4-1 a 4-4

- Al comienzo del fichero incorporan la librería:

```
<script src="gl-matrix-min.js"></script>
```

- Un ejemplo en Javascript sería:

```
var mvMatrix = mat4.create();
mat4.identity(mvMatrix);
mat4.translate(mvMatrix, [newPos , 0.0, 0.0]);
mat4.scale(mvMatrix, [scale, scale, 0.0]);
mat4.rotate(mvMatrix, angle, [0.0, 1.0, 0.0]);
```

Example 4-1 a 4-4

- En los shaders se utiliza como:

```
uniform mat4 uMVMatrix;

void main(void) {

  gl_Position = uMVMatrix * vec4(aVertexPosition, 1.0);

}
```

- Y desde Javascript se asigna un valor:

```
gl.uniformMatrix4fv(glProgram.uMVMatrix, false,
  mvMatrix);
```

gl.**uniformMatrix4fv**

gl.uniformMatrix4fv(location, transpose, array)

Assign the 4×4 matrix specified by *array* to the uniform variable specified by *location*.

Parameters	location	Specifies the storage location of the uniform variable.
	Transpose	Must be `false` in WebGL.[3]
	array	Specifies an array containing a 4×4 matrix in column major order (typed array).
Return value	None	
Errors	INVALID_OPERATION	There is no current program object.
	INVALID_VALUE	*transpose* is not `false`, or the length of *array* is less than 16.