

Laboratori Gràfics

Shaders - Sessió 3

Sistemes de Coordenades, Transformacions
geomètriques, animacions.

Sistemes de coordenades i matrius

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

*Viewport transform &
Depth transform*

Window space

uniform mat4 modelMatrix;

uniform mat4 viewMatrix;

uniform mat4 projectionMatrix;

uniform mat4 modelViewMatrix;

uniform mat4 modelViewProjectionMatrix;

uniform mat4 modelMatrixInverse;

uniform mat4 viewMatrixInverse;

uniform mat4 projectionMatrixInverse;

uniform mat4 modelViewMatrixInverse;

uniform mat4 modelViewProjectionMatrixInverse;

uniform mat3 normalMatrix;

Transformacions bàsiques

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

**Viewport transform &
Depth transform**

Window space

Modeling transforms

translate(t_x, t_y, t_z)

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

scale(s_x, s_y, s_z)

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

rotate(a, x, y, z)

$$T = \begin{bmatrix} x^2 d + c & xyd - zs & xzd + ys & 0 \\ yxd + zs & y^2 d + c & yzd - xs & 0 \\ xzd - ys & yzd + xs & z^2 d + c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$c = \cos(a)$, $s = \sin(a)$, $d = 1 - \cos(a)$

Transformacions bàsiques

Object space

Modeling transform

World space

Viewing transform

Eye space

Projection transform

Clip space

Perspective division

Normalized Device space

**Viewport transform &
Depth transform**

Window space

Modeling transforms

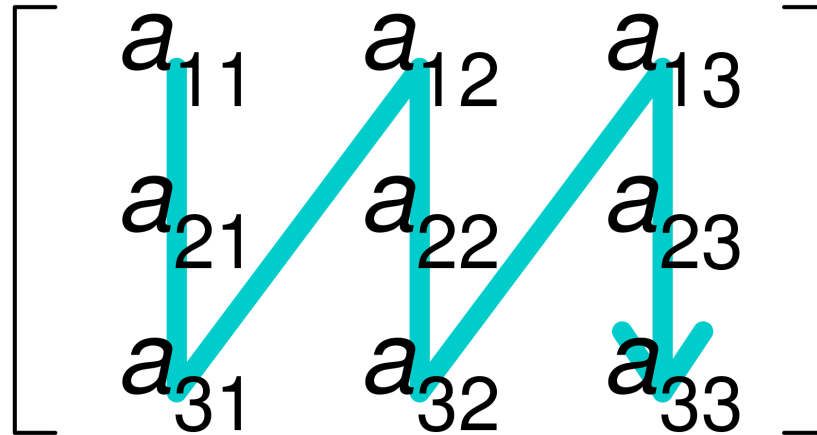
$$\text{glRotate}*(a, 1, 0, 0): \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & -\sin a & 0 \\ 0 & \sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{glRotate}*(a, 0, 1, 0): \begin{bmatrix} \cos a & 0 & \sin a & 0 \\ 0 & 1 & 0 & 0 \\ -\sin a & 0 & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{glRotate}*(a, 0, 0, 1): \begin{bmatrix} \cos a & -\sin a & 0 & 0 \\ \sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrius en GLSL

```
mat3 m = mat3(vec3(1,0,0), vec3(0,1,0), vec3(0,0,1));  
// els tres vectors són les columnes de la matriu
```

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$
A 3x3 matrix is shown with elements labeled a_{ij}. Cyan arrows point from each element to the column it belongs to: a₁₁, a₂₁, and a₃₁ point to the first column; a₁₂, a₂₂, and a₃₂ point to the second column; and a₁₃, a₂₃, and a₃₃ point to the third column.

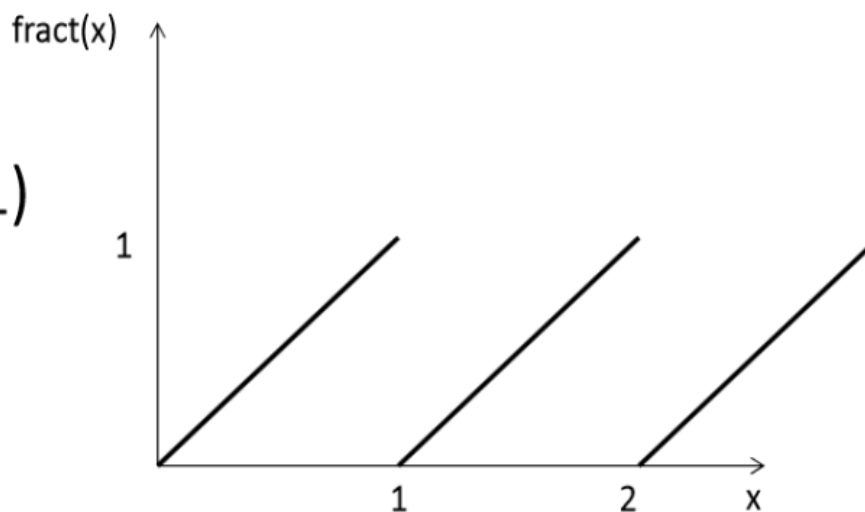
Funcions GLSL

Funcions GLSL – $\text{fract}(x)$

Retorna la part fraccionària de x , calculada com

$$x - \text{floor}(x)$$

- Domini: \mathbb{R}^n
- Recorregut: $[0, 1)$
- Període: 1

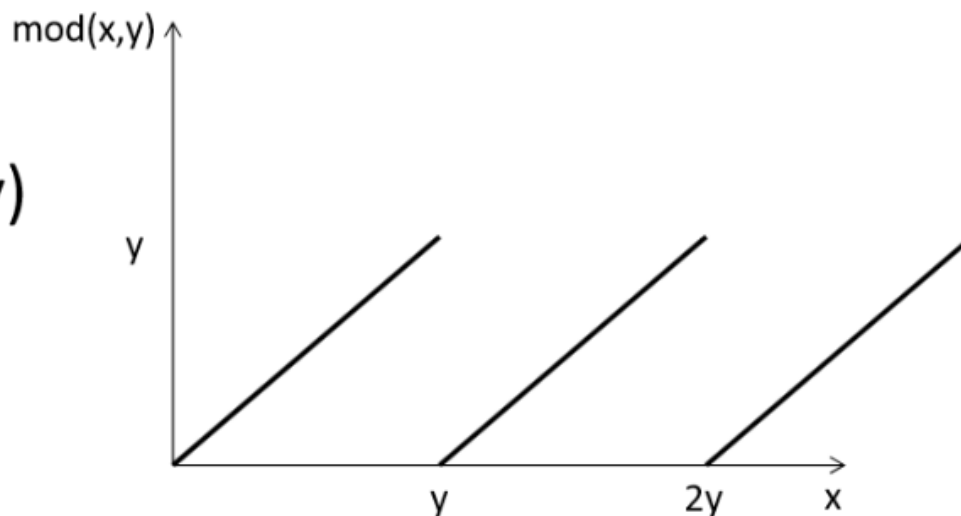


Funcions GLSL – $\text{mod}(x,y)$

Retorna **x mòdul y** , calculat com

$$x - y * \text{floor}(x/y)$$

- Domini: \mathbb{R}^n
- Recorregut: $[0, y)$
- Període: y



Funcions GLSL – `mix(a,b,t)`

Retorna la interpolació lineal entre `a` i `b` ponderada per `t`:

$$a(1-t)+bt = a + t(b-a)$$

- Habitualment `t` és un escalar en `[0,1]`.
- Els paràmetres `a`, `b` poden ser vectorials (en aquest cas la interpolació es fa per components):

`mix(vec3(1,0,0), vec3(0,1,0), 0.5) → retorna vec3(0.5,0.5,0)`

Funcions GLSL – $\sin(x)$

Retorna el sinus de x (en radians).

És freqüent usar sinusoidals de la forma:

$$A * \sin(2\pi * f * t + \Theta)$$

A = amplitud

f = freqüència; el factor 2π apareix només si volem que freq estigui en Hz

t = temps (en segons)

Θ = fase; si per exemple $\Theta = \{0, \pi/2, 3\pi/2\}$, llavors per $t=0$ la sinusoidal serà $\{0, A, -A\}$

Animacions als shaders

```
uniform float time;  
const float PI = 3.141592;
```

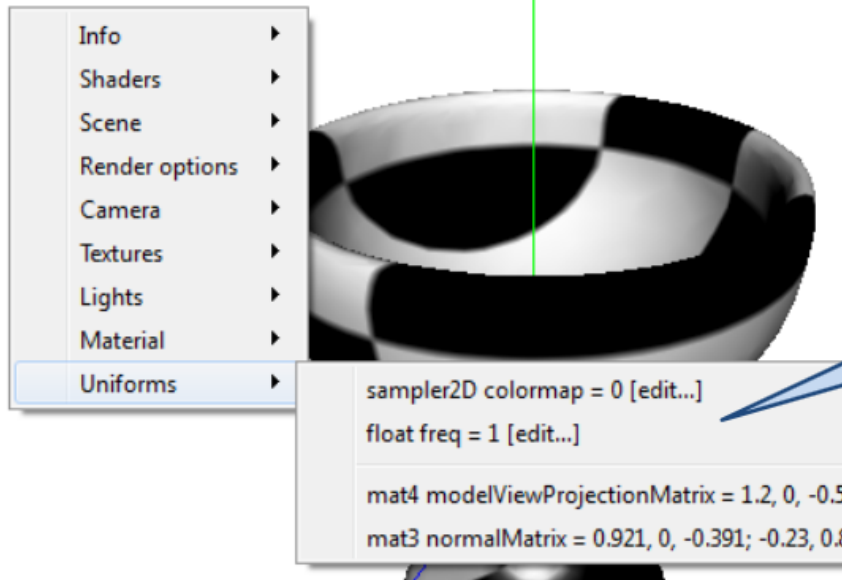
Temps (segons) des de la darrera compilació.

```
void main()  
{  
    fragColor = vec4(0.5*(sin(2*PI*time)+1.0));  
}
```

User-defined uniforms

```
uniform float freq=2.0; // frecuencia en Hz
void main()
{
    fragColor=vec4(.5*(sin(2*PI*freq*time)+1.0));
}
```

Uniform definit per l'usuari; convé donar-li un valor.



Uniforms definit per l'usuari: el viewer permet editar-los (actualment limitat a bool, int, float)

Uniforms definit pel viewer (el menu no en permet l'edició)