Tenis - Semifinales, Nemesio Antunez (1979)

CC3501 Transformaciones (Parte 1)

Eduardo Graells-Garrido

Otoño 2023

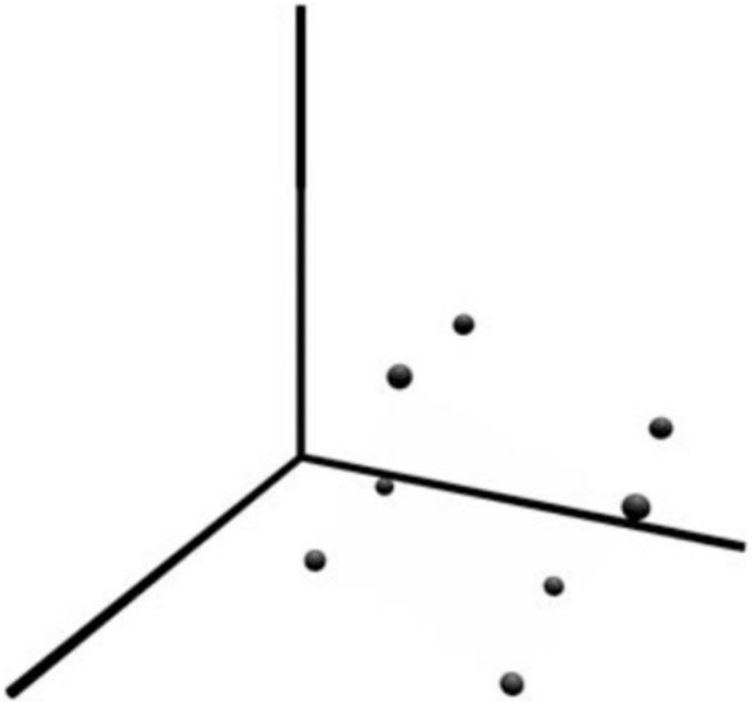
¿Qué es una transformación?

Cualquier función que asigne una posición nueva a un punto.

Nos enfocaremos en transformaciones comunes (rotación, escalamiento, etc.) expresadas como **mapas lineales**.

Y lo haremos en 2D y 3D.

Recordar: representamos los objetos como un conjunto de primitivas determinadas por sus vértices.



$$f : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

¿Hacia dónde apunta el eje Z?

Los ejes cartesianos son ortogonales.

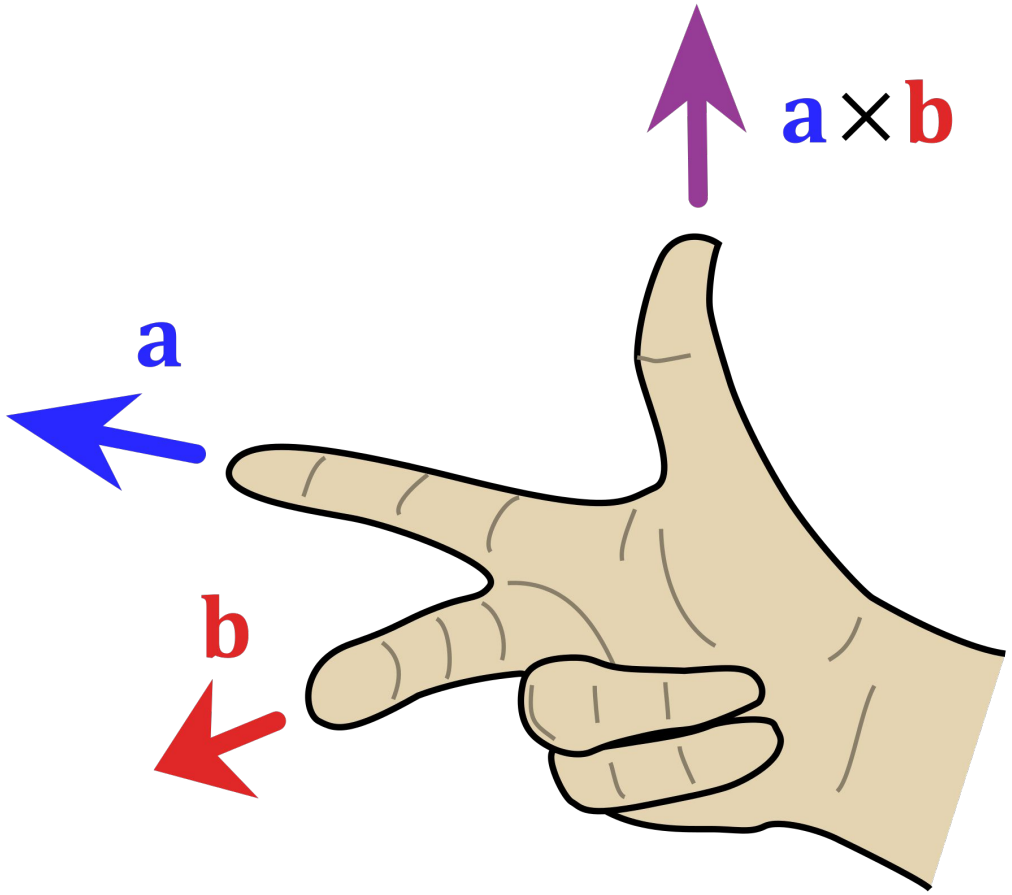
Estamos acostumbrados a trabajar con X e Y en la pantalla.

Entonces, ¿hacia dónde apunta el eje Z?

¿Afuera o adentro?

La respuesta la tenemos en la r**egla de la mano derecha**. El pulgar nos dice hacia dónde apunta el vector.

También existe la regla de la mano de izquierda.



¿Cuándo se usan?

¡Siempre!

Manipular objetos en el espacio

Posicionar la cámara

Animar objetos

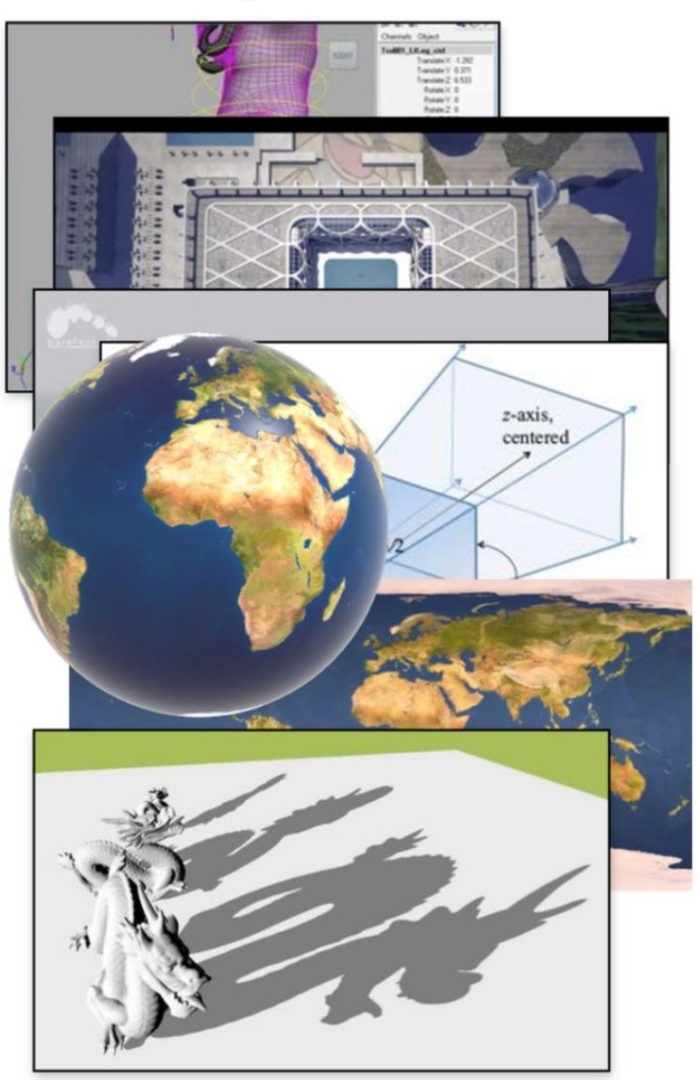
Proyectar objetos 3D en superficies 2D

Proyectar superficies 2D en objetos 3D

Proyectar sombras de objetos 3D en otros objetos 3D

Determinar la proyección de la imagen

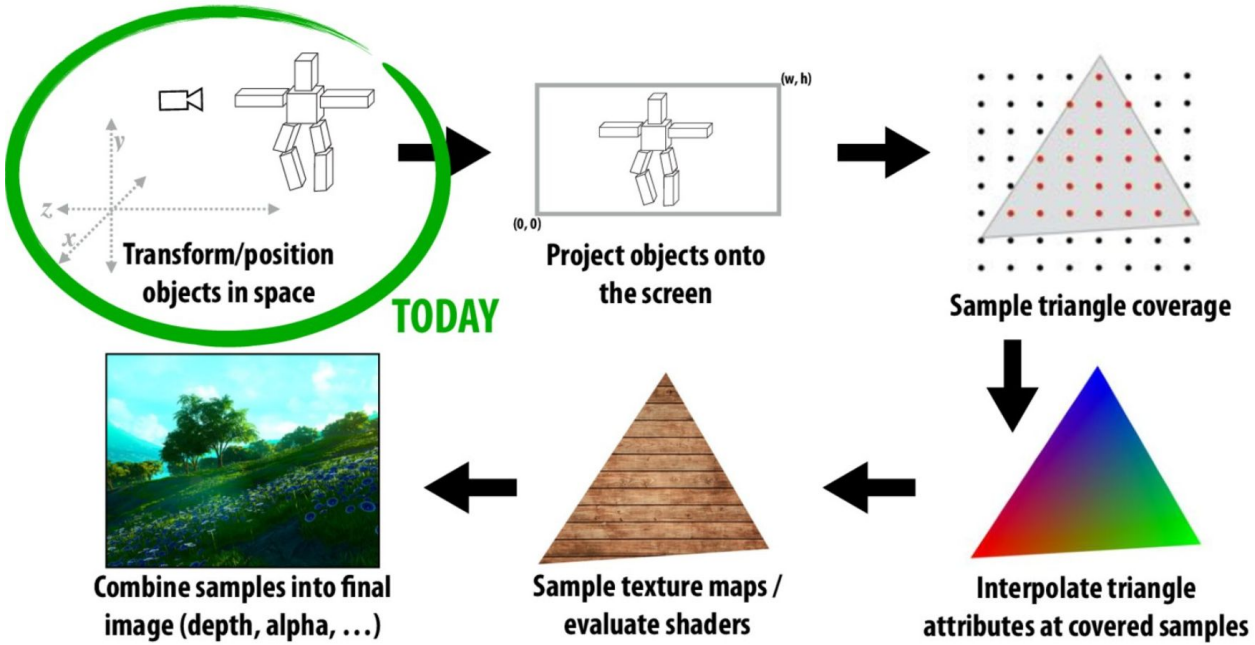
¡Y más!



¿Cuándo se usan?

En el contexto del rendering, las transformaciones son lo primero que se aplica.

Y ello independiente del paradigma que se utilice: sea rasterización o ray tracing (o similares), las transformaciones se aplican primero, y siempre.

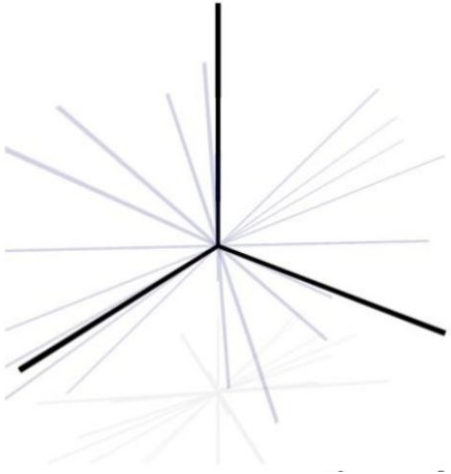


Mapas Lineales

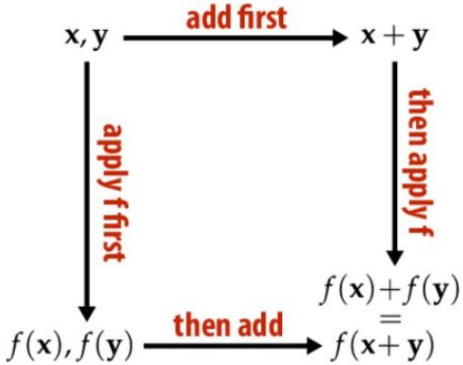
¿Qué significa que la función

$$f : \mathbb{R}^n \to \mathbb{R}^n$$

sea **lineal**?



Geométricamente: convierte líneas en líneas, y preserva el origen



Algebraicamente: preserva las operaciones dentro del espacio vectorial (suma y escalamiento)

Beneficios de la Linealidad

Barato de calcular. Los sistemas lineales son fáciles de resolver.

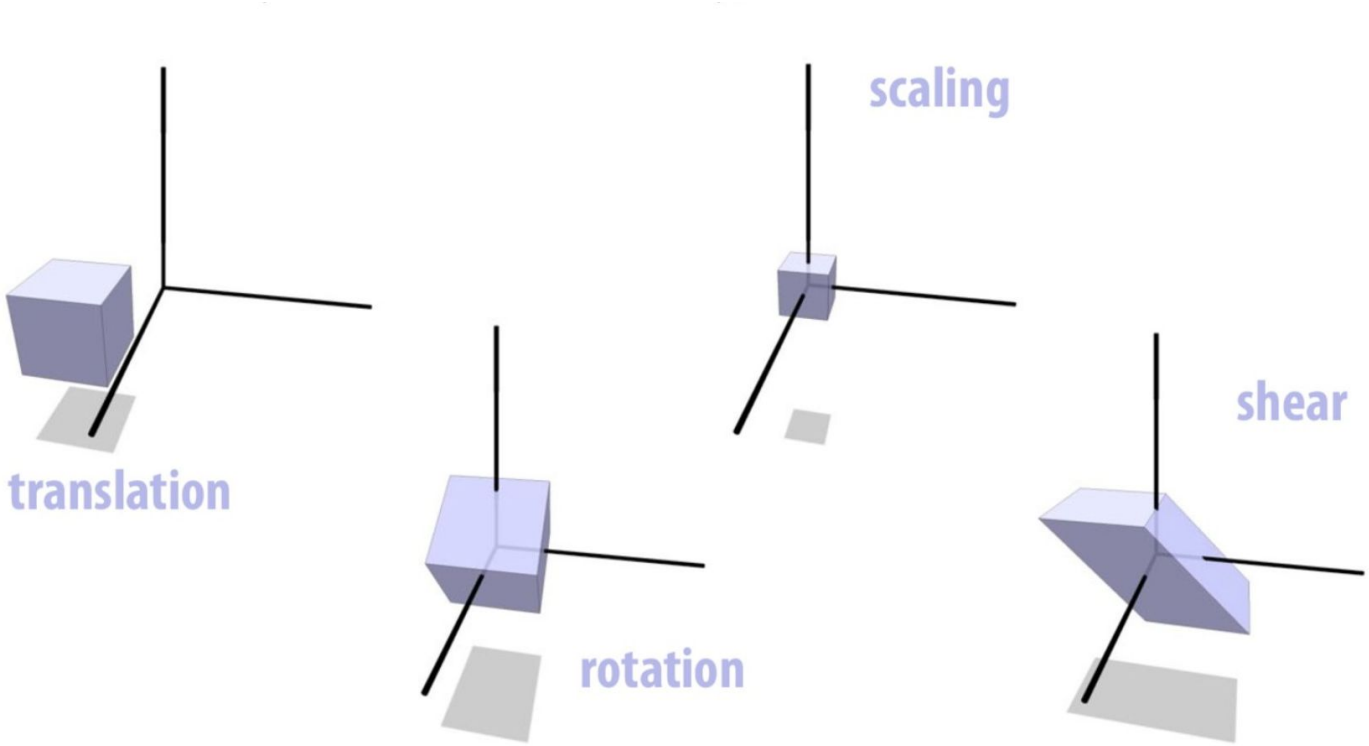
La composición de transformaciones lineales es lineal.

El producto de muchas matrices es una matriz. Nos da una representación uniforme. Simplifica algoritmos y sistemas (GPUs, APIs, etc.)

$$\underbrace{\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{rotation} \underbrace{\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}}_{scale} \underbrace{\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}}_{rotation} \cdots = \underbrace{\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}}_{\substack{composite \\ transformation}}$$

Tipos de Transformación

¿Qué tienen en común estas transformaciones?



Una transformación se define por por sus invariantes

lineal: mantiene líneas y origen

$$f(a\mathbf{x}+\mathbf{y}) = af(\mathbf{x}) + f(\mathbf{y}),$$
$$f(0) = 0$$

traslación: mantiene distancias entre puntos

$$f(\mathbf{x}-\mathbf{y}) = \mathbf{x}-\mathbf{y}$$

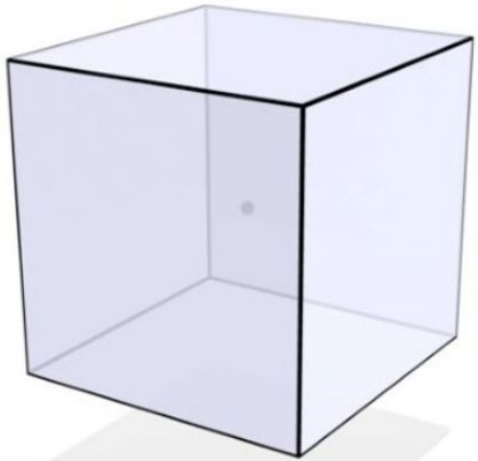
escalamiento: mantiene las líneas en el origen y la dirección de los vectores

$$f(\mathbf{x})/|f(\mathbf{x})| = \mathbf{x}/|\mathbf{x}|$$

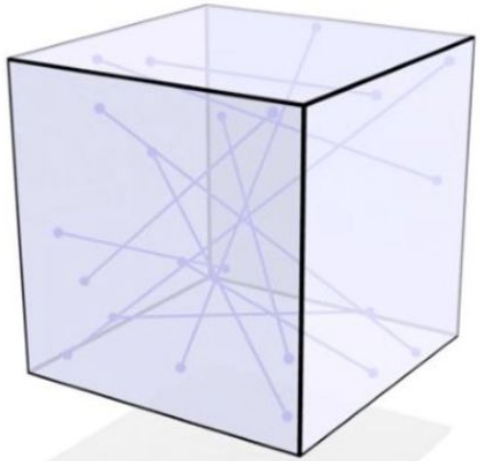
rotación: mantiene el origen, la distancia entre los puntos y su orientación

$$|f(\mathbf{x})-f(\mathbf{y})| = |\mathbf{x}-\mathbf{y}|,$$
$$\det(f) > 0$$

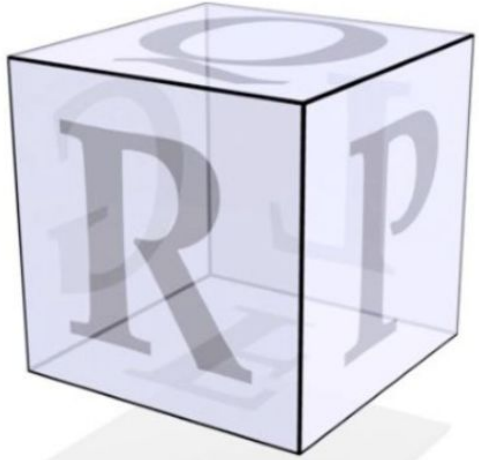
Una rotación tiene tres propiedades:



keeps origin fixed



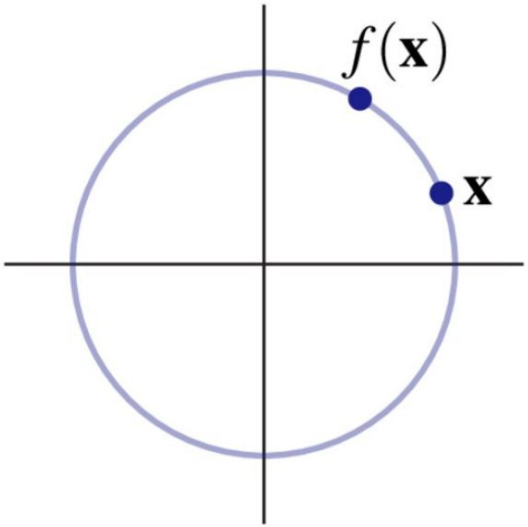
preserves distances

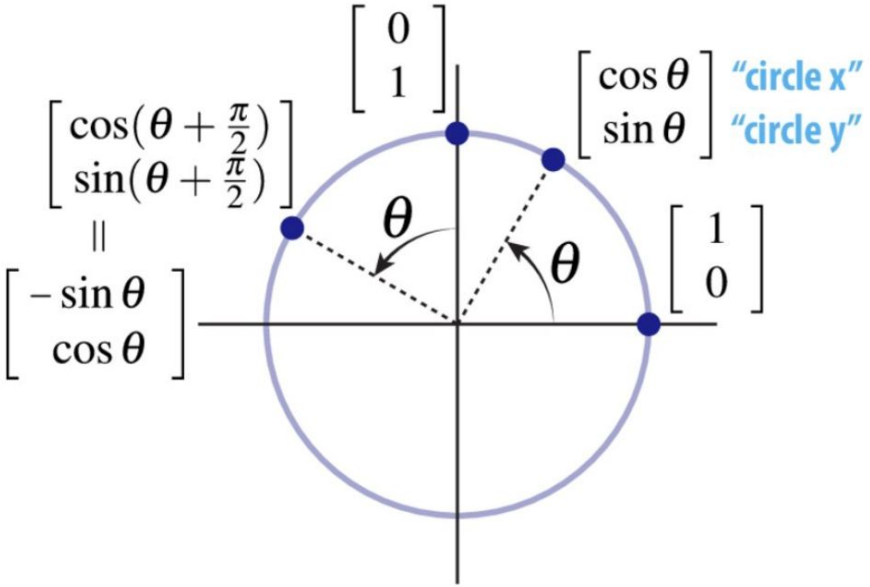


preserves orientation

Rotaciones en 2D

Como las rotaciones preservan las distancias y el origen, una **rotación por un ángulo theta convierte un punto x en otro punto dentro del círculo de radio |x|**.

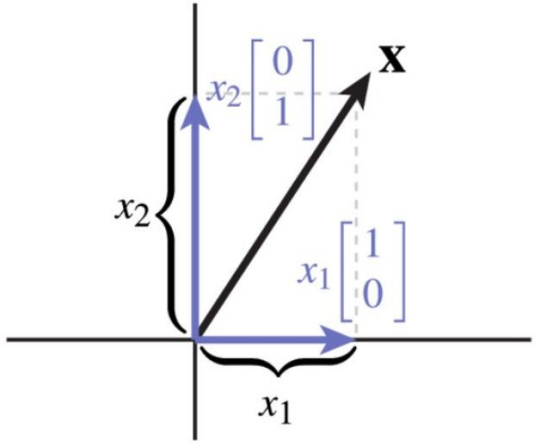


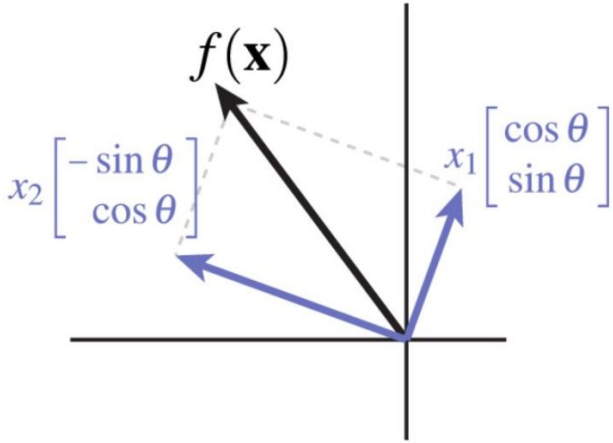


Rotaciones como matrices

Un punto arbitrario (x1, x2) se puede rotar aplicando las rotaciones que vimos anteriormente de manera simultánea.

¡Esta fórmula se puede expresar como una multiplicación matricial!





$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$f(\mathbf{x}) = x_1 \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + x_2 \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

$$f_\theta(\mathbf{x}) = \begin{bmatrix} \cos\theta & -\sin(\theta) \\ \sin\theta & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

¿Y en 3D?

Para rotar alrededor de cada eje aplicamos la misma idea: dejamos cada eje fijo y aplicamos la regla del círculo.

La dirección de los ángulos importa, aquí es importante la regla de la mano derecha.

rotate around x_1

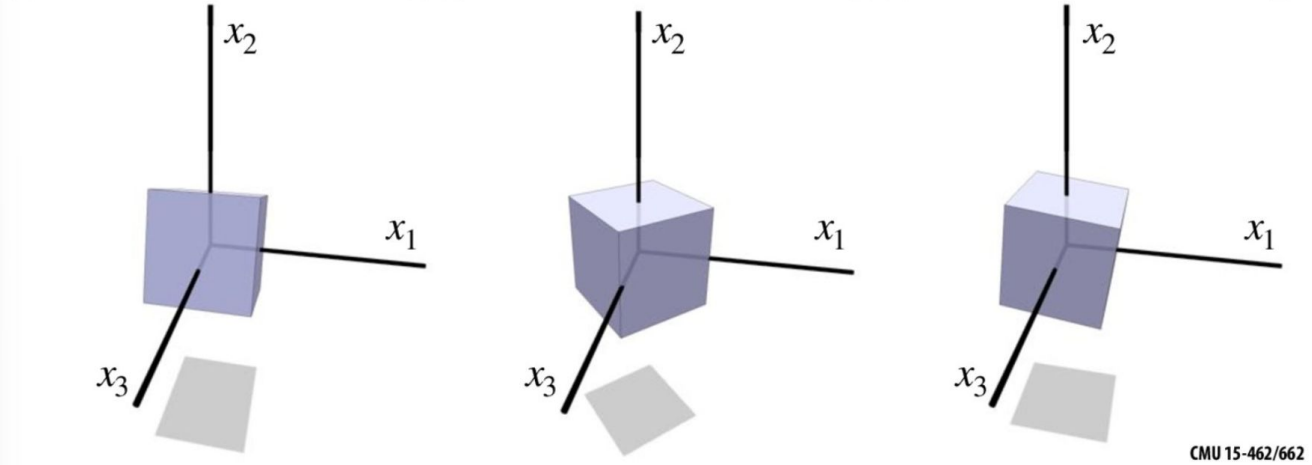
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin(\theta) \\ 0 & \sin\theta & \cos(\theta) \end{bmatrix}$$

rotate around x_2

$$\begin{bmatrix} \cos\theta & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos(\theta) \end{bmatrix}$$

rotate around x_3

$$\begin{bmatrix} \cos\theta & -\sin(\theta) & 0 \\ \sin\theta & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



¿Cómo se expresa la inversa de una rotación?

$$R^{\mathsf{T}} = R^{-1}$$

No todas las matrices cuya traspuesta sea su inversa son rotaciones.

En este ejemplo no estamos hablando de una rotación.

¿Qué es lo que hace esta matriz?

$$Q = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Q^{\mathsf{T}}Q = \begin{bmatrix} (-1)^2 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Transformaciones Ortogonales

Las matrices de la slide anterior son **reflexiones**.

Permiten crear reflejos de los elementos de nuestra escena. Ejemplo: **hacer un espejo**.

Esta transformación también preserva distancia y origen, al igual que las rotaciones. Cuando eso sucede, hablamos de **transformaciones ortogonales**.

La diferencia entre ambas es el signo del **determinante** de la matriz: rotación tiene determinante positivo, reflexión tiene determinante negativo.

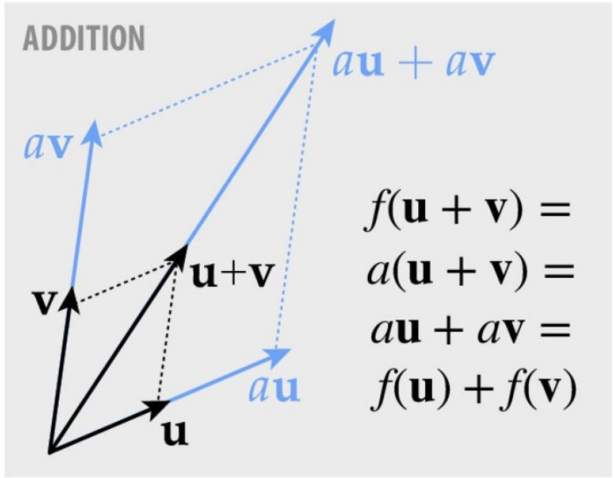
Escalamiento

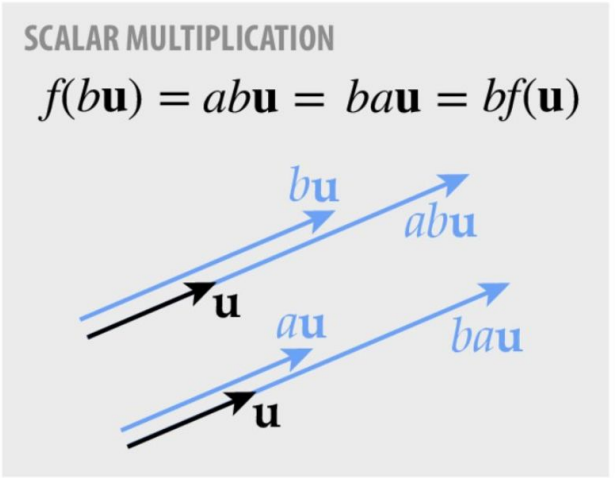
La definición de escalamiento es directa:

$$f(\mathbf{u}) = a\mathbf{u}, \quad a \in \mathbb{R}$$

Esta función preserva la dirección de los vectores:

$$\frac{\mathbf{u}}{|\mathbf{u}|} = \frac{a\mathbf{u}}{|a\mathbf{u}|}$$





Por tanto, es una transformación lineal.

Escalamiento como Matriz

¿Qué pasa si **a** es negativa?

Tenemos una secuencia de reflexiones.

$$\underbrace{\begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} au_1 \\ au_2 \\ au_3 \end{bmatrix}}_{a\mathbf{u}}$$

Escalamiento no-uniforme

¿Qué pasa si escalamos cada eje por valores distintos?

Esto sigue escalando respecto a los ejes base.

¿Se puede hacer un escalamiento en base a ejes arbitrarios?¿Cómo lograrlo?

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} au_1 \\ bu_2 \\ cu_3 \end{bmatrix}$$

Escalamiento no Uniforme

La combinación es la siguiente:

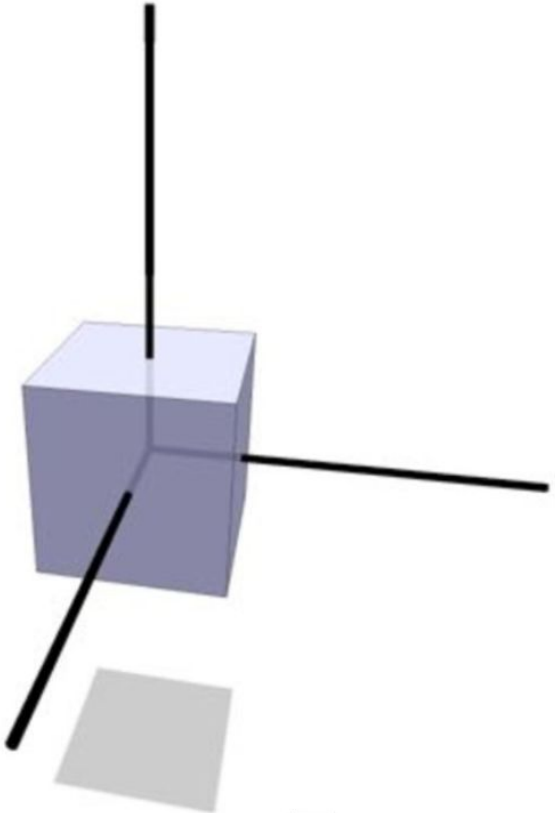
Rotar hacia el nuevo eje

Aplicar escalamiento en la diagonal

Rotar de vuelta hacia los ejes originales

Como la inversa de la rotación es la traspuesta, el resultado es una matriz simétrica:

$$A := R^{\mathsf{T}} D R$$



$$f(\mathbf{x}) = R^{\mathsf{T}} D R \mathbf{x}$$

¿Todas las matrices simétricas representan escalamiento no-uniforme para algún conjunto existente de ejes?

Teorema Espectral

¡Sí!

Como toda matriz simétrica tiene vectores propios ortonormales, y

$$Ae_i = \lambda_i e_i$$

De la relación

$$AR = RD$$

Podemos determinar que $R = \begin{bmatrix} e_1 & \cdots & e_n \end{bmatrix}$ $D = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$

Por tanto cada matriz simétrica es equivalente a un escalamiento no uniforme a lo largo de un conjunto de ejes ortogonales.

Deformación “Shear”

Esta transformación **desplaza cada punto x en una dirección u en función de su distancia a través de un vector v**:

$$f_{\mathbf{u},\mathbf{v}}(\mathbf{x}) = \mathbf{x} + \langle \mathbf{v}, \mathbf{x} \rangle \mathbf{u}$$

Es una transformación lineal y de hecho se puede representar matricialmente:

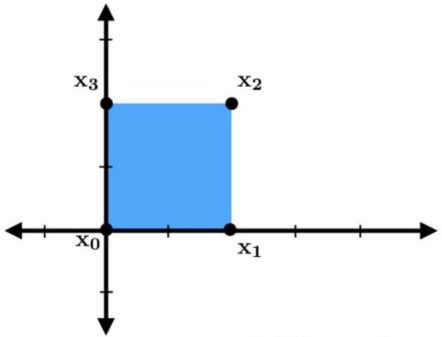
$$A_{\mathbf{u},\mathbf{v}} = I + \mathbf{u}\mathbf{v}^{\mathsf{T}}$$

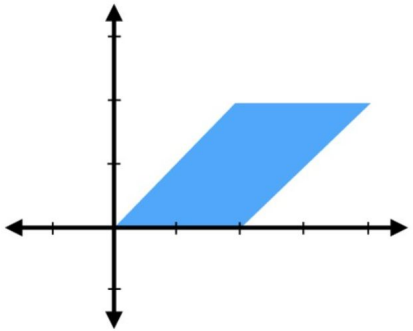
$\mathbf{u} = (\cos(t),0,0)$

$\mathbf{v} = (0,1,0)$

$$A_{\mathbf{u},\mathbf{v}} = \begin{bmatrix} 1 & \cos(t) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear



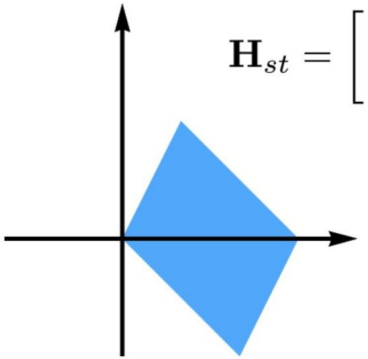


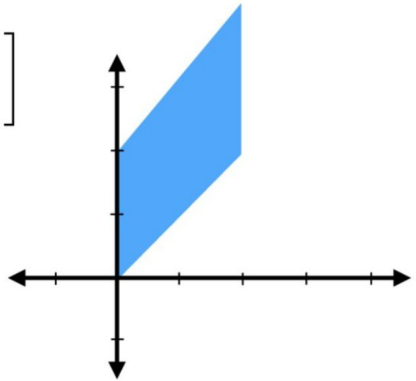
Shear in x:

$$\mathbf{H}_{xs} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$

Arbitrary shear:

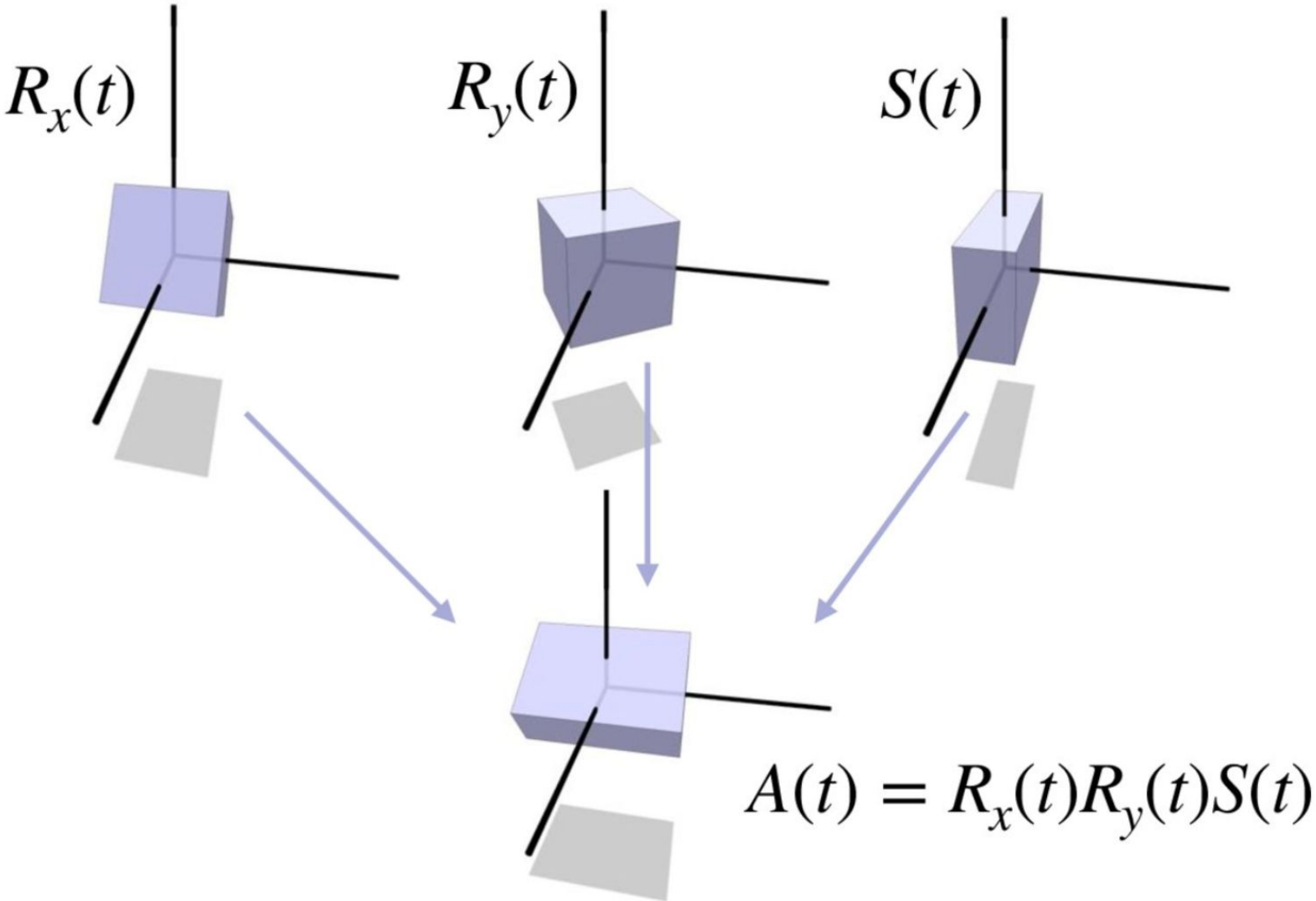
$$\mathbf{H}_{st} = \begin{bmatrix} 1 & s \\ t & 1 \end{bmatrix}$$





Shear in y:

$$\mathbf{H}_{ys} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$


$R_x(t)$
$R_y(t)$
$S(t)$
$A(t) = R_x(t)R_y(t)S(t)$

¿Cómo descomponer una transformación lineal en una secuencia de operaciones?

Descomposición de Transformaciones

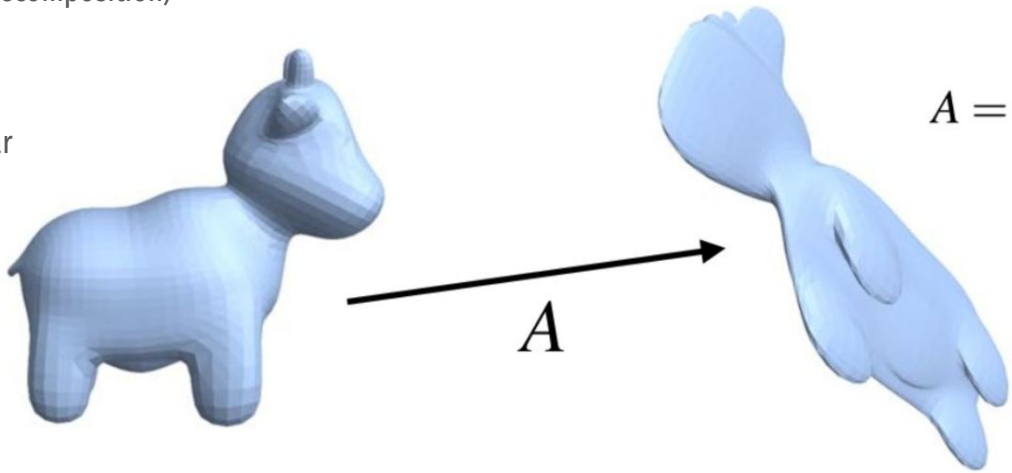
En general, no hay una manera **única** de descomponer una transformación.

Existen descomposiciones que podemos utilizar:

SVD (singular value decomposition)

Factorización LU

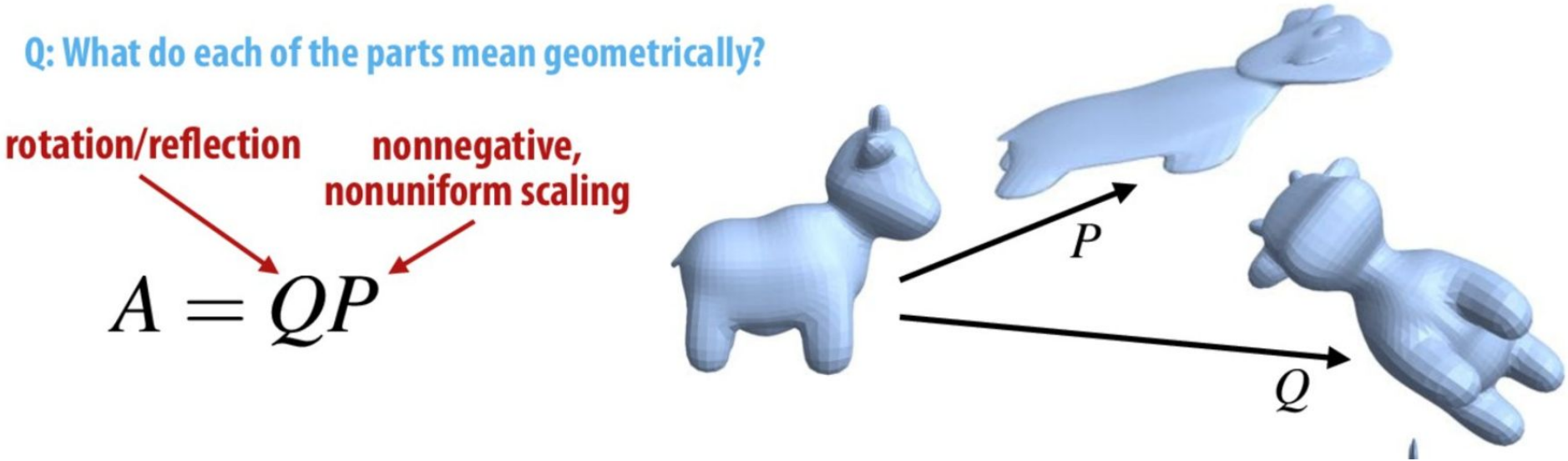
Descomposición Polar



$$A = \begin{bmatrix} .34 & -.11 & -.89 \\ -.65 & .52 & -.70 \\ .25 & .23 & -.69 \end{bmatrix}$$

Descomposición Polar

Esta operación descompone una matriz A en la multiplicación de una matriz ortogonal Q y una matriz P semidefinida positiva y simétrica:



Singular Value Decomposition

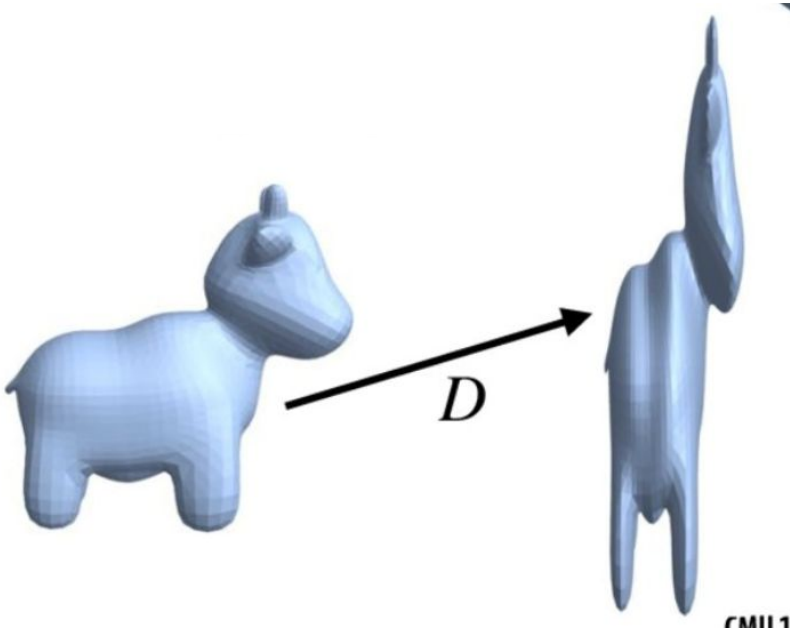
Ya que la matriz P es simétrica, podemos usar la descomposición espectral P = VDV^T (V es ortogonal, D es diagonal).
Si reemplazamos QV por U, nos queda una composición rotación x escalamiento x rotación. Esta es una descomposición SVD.

$$A = \underbrace{QV}_{U} DV^{\mathsf{T}} = UDV^{\mathsf{T}}$$

rotation

axis-aligned scaling

rotation



¿Por qué son útiles estas descomposiciones?

¡Interpolación!

Usualmente una animación define distintas etapas importantes llamadas **keyframes**.

Necesitamos interpolar entre **keyframes** para tener una animación suave.

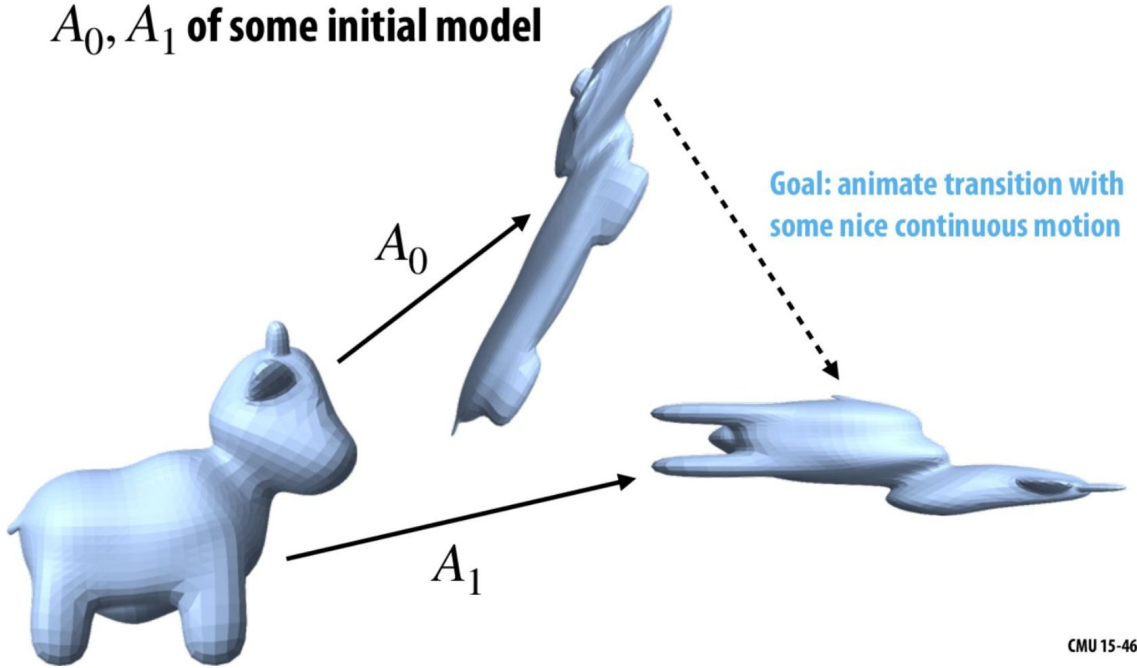
Figura 2.17: Keyframes de una animación para un modelo 3D. En este caso, se trata de la animación que representa a un personaje corriendo.

¿Cómo llegar desde A0 hasta A1?

Una primera aproximación ingenua es definir el intervalo de tiempo que dura la animación y realizar una interpolación lineal. Pero se ve **horrible**.

$$A(t) = (1 - t)A_0 + tA_1$$

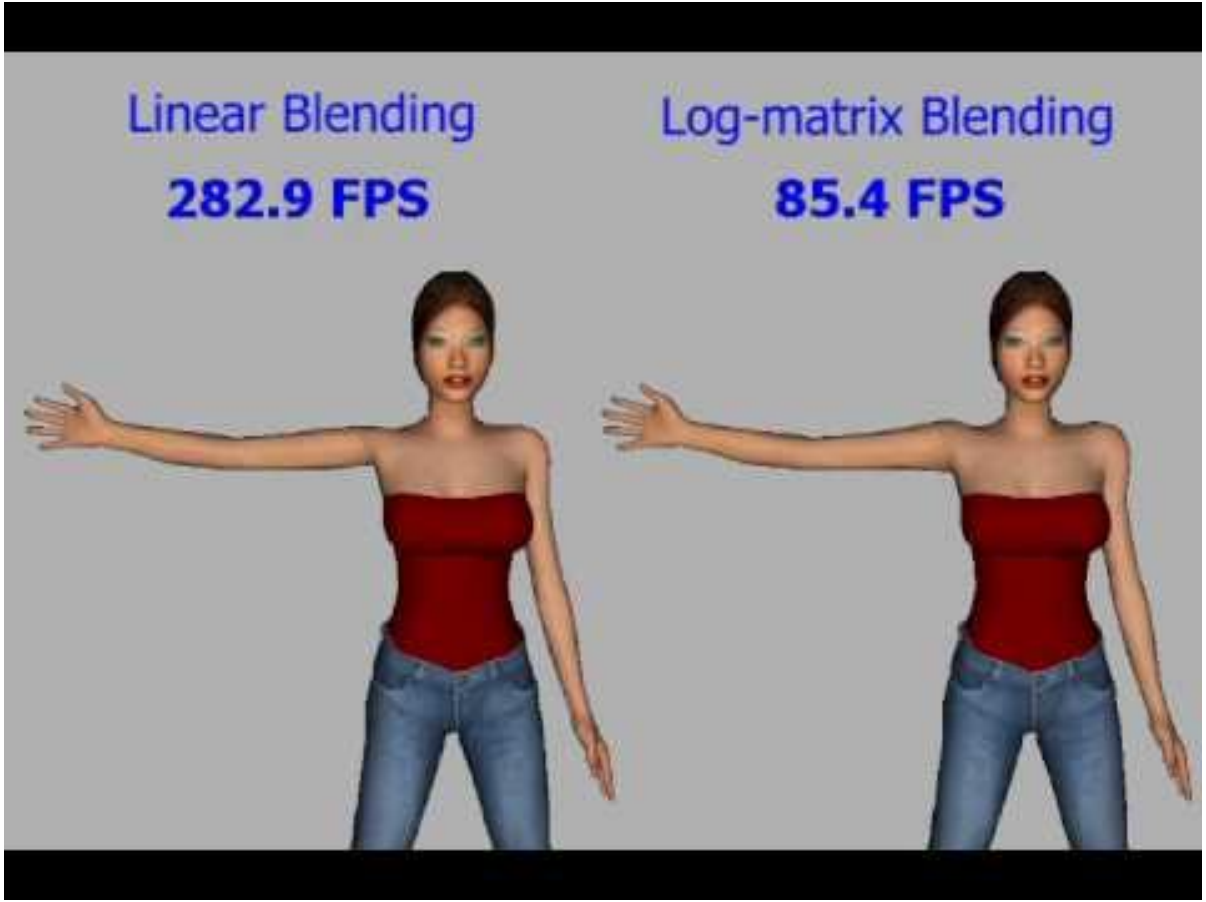
- **Consider interpolating between two linear transformations A_0, A_1 of some initial model**



Ejemplo

¿Por qué eso es un problema? Consideren este ejemplo de interpolación de animaciones en un modelo 3D de un cuerpo humano.

Más adelante en las clases de animación veremos como se estructuran y realizan las animaciones de este estilo.



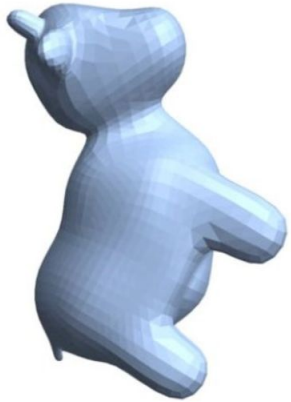
Idea: interpolar cada componente

$$A_0 = Q_0 P_0, \quad A_1 = Q_1 P_1$$

scaling

$$P(t) = (1-t)P_0 + tP_1$$

rotation



$$\widetilde{Q}(t) = (1-t)Q_0 + tQ_1$$

$$\widetilde{Q}(t) = Q(t)X(t)$$

final interpolation

$$A(t) = Q(t)P(t)$$

…looks better!

See: Shoemake & Duff, "Matrix Animation and Polar Decomposition"

Traslaciones

Hasta el momento hemos dejado de lado una transformación básica: la **traslación**.

Su definición es básica.

¿Es una transformación **lineal**?

$$f_{\mathbf{u}}(\mathbf{x}) = \mathbf{x} + \mathbf{u}$$

additivity

$$f_{\mathbf{u}}(\mathbf{x}+\mathbf{y}) = \mathbf{x}+\mathbf{y}+\mathbf{u}$$

$$f_{\mathbf{u}}(\mathbf{x}) + f_{\mathbf{u}}(\mathbf{y}) = \mathbf{x}+\mathbf{y}+2\mathbf{u}$$

homogeneity

$$f_{\mathbf{u}}(a\mathbf{x}) = a\mathbf{x}+\mathbf{u}$$

$$af_{\mathbf{u}}(\mathbf{x}) = a\mathbf{x}+a\mathbf{u}$$

¡No! Es una transformación afín (es invertible), no lineal.

Composición de Transformaciones

Para transformaciones lineales **multiplicamos** matrices:

$$A_3(A_2(A_1\mathbf{x})) = (A_3A_2A_1)\mathbf{x}$$

Para traslaciones **sumamos** vectores:

$$f_{\mathbf{u}_3}(f_{\mathbf{u}_2}(f_{\mathbf{u}_1}(\mathbf{x}))) = f_{\mathbf{u}_1+\mathbf{u}_2+\mathbf{u}_3}(\mathbf{x})$$

¿Y qué pasa si quiero componer transformaciones lineales con traslaciones?

Tendríamos que considerar casos especiales para sumar vectores y multiplicar matrices.

Coordenadas Homogéneas

Surgieron al querer estudiar perspectivas

Introducidas por Moebius como una manera natural de asignar coordenadas a líneas

Surgen de manera natural en muchos aspectos de computación gráfica:

Transformaciones 3D, proyección en perspectiva, mapas de sombras, geometría discreta conformal, recorte, luces direccionales...

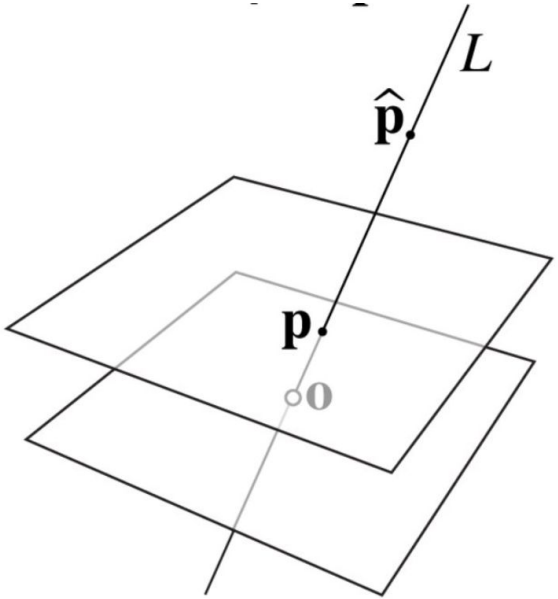
... es un concepto que veremos de manera recurrente durante el curso.

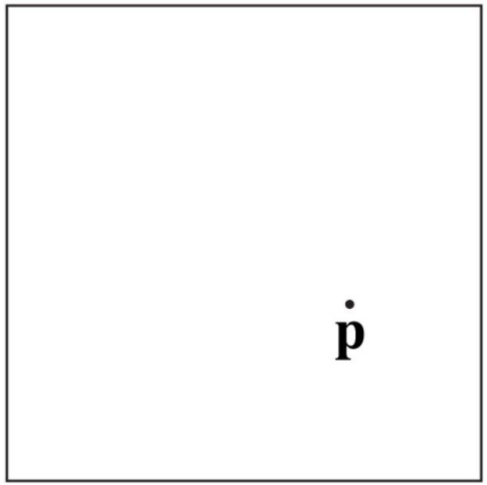
Consideren cualquier plano 2D que no contenga al origen en 3D.

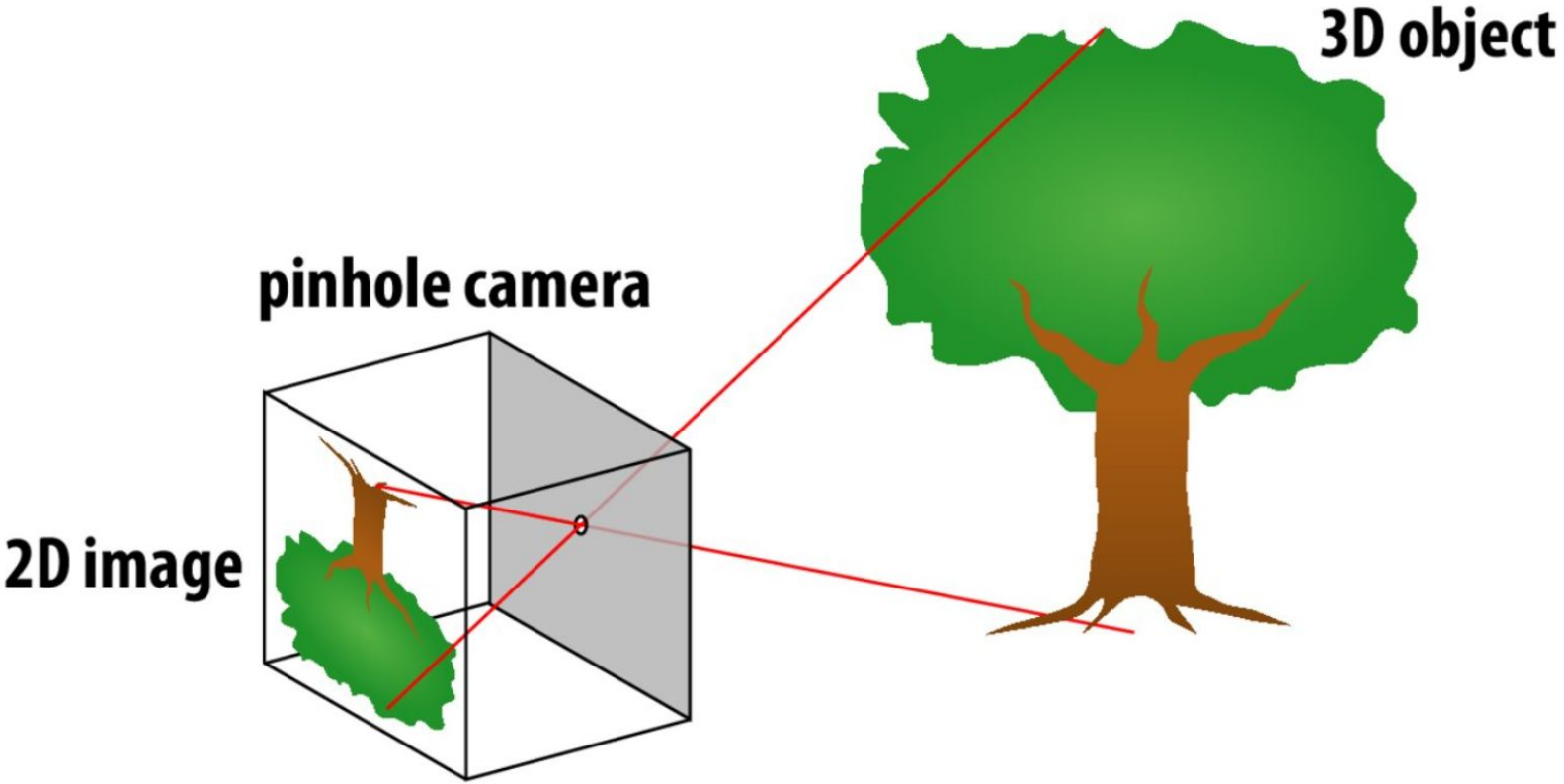
Cualquier línea L que pase a través del origen se intersecta con ese plano.

Para representar a la línea podemos usar el punto p en que intersecta al plano.

Dicho de esa manera, cualquier punto que esté en la línea L puede ser usado para representar al punto p.







Esto se asemeja a mirar el mundo a través del lente de una cámara.
Veremos más de esto en la clase de **vistas**.

Definición

p = (xi, yi)

El plano de proyección será el mismo de nuestra imagen, y se encontrará en un valor z = f (usualmente f = 1)

p^ = (xs, ys, zs) tal que (xs/zs, ys/zs) = (xi,yi)

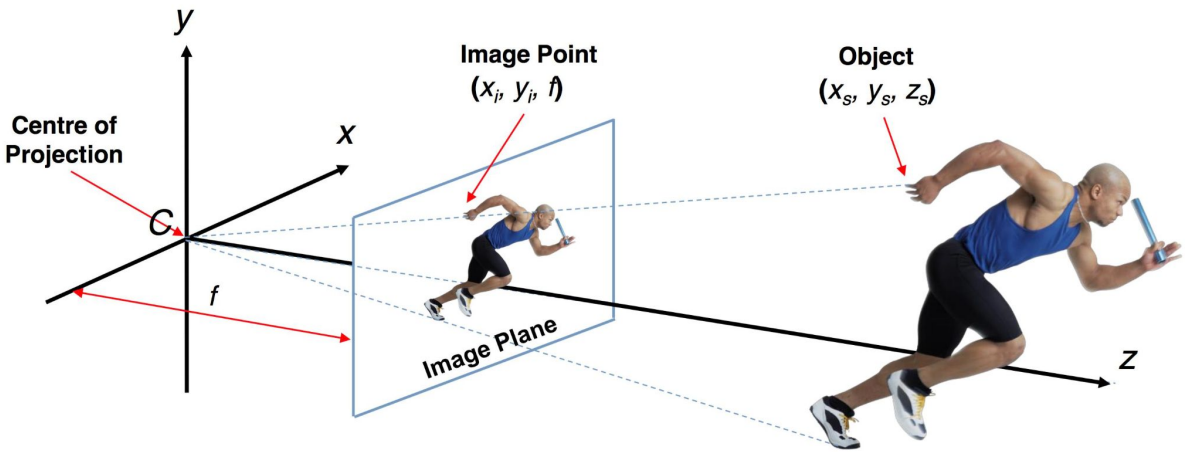
Se dice que **p^** es una coordenada homogenea de **p**

Ejemplo: (x, y, 1)

En general, (cx, cy, c) para c != 0

¿Por qué sirve esto para las transformaciones?

¿De dónde proviene? Hint: geometría proyectiva (ver link propuesto al final)



$$x_i = f\frac{x_s}{z_s}, \quad y_i = f\frac{y_s}{z_s}$$

Recordemos la definición de shear:

$$f_{\mathbf{u},\mathbf{v}}(\mathbf{x}) = \mathbf{x} + \langle \mathbf{v}, \mathbf{x} \rangle \mathbf{u}$$

Su expresión matricial es:

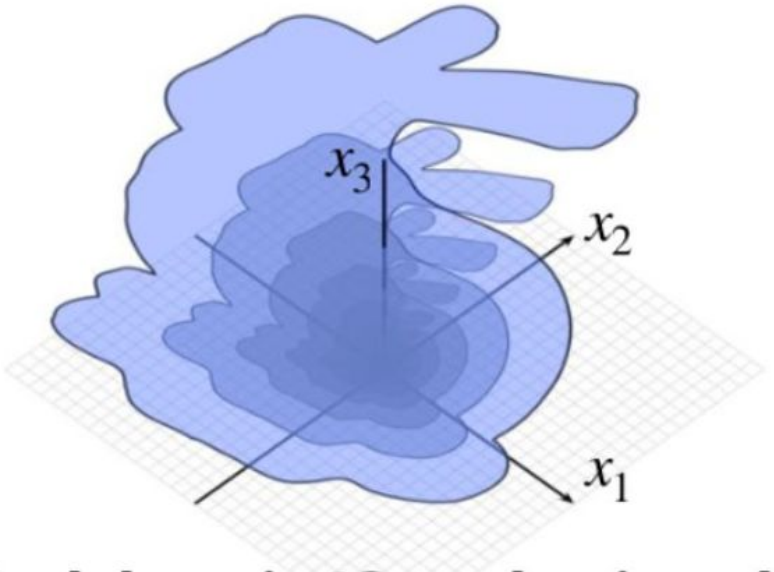
$$f_{\mathbf{u},\mathbf{v}}(\mathbf{x}) = \left(I + \mathbf{u}\mathbf{v}^{\mathsf{T}}\right) \mathbf{x}$$

Si v = (0, 0, 1), obtenemos

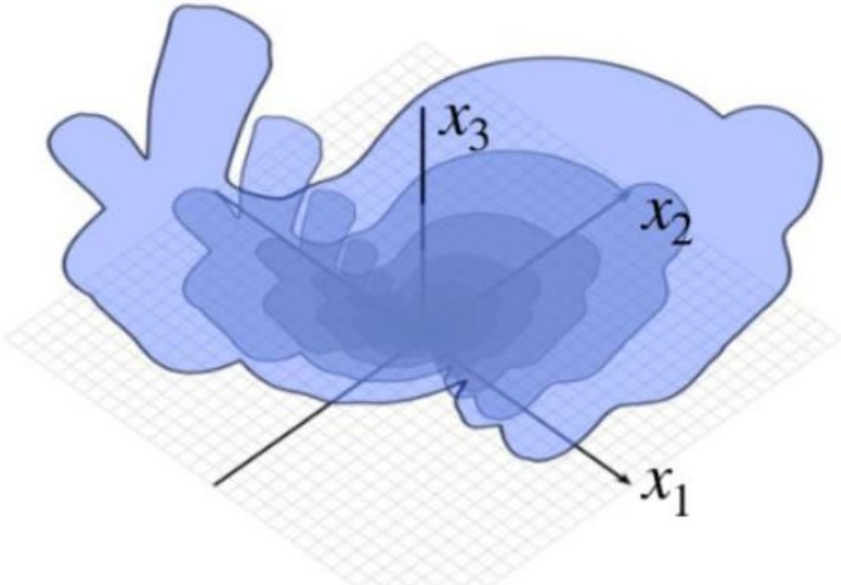
$$\begin{bmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cp_1 \\ cp_2 \\ c \end{bmatrix} = \begin{bmatrix} c(p_1 + u_1) \\ c(p_2 + u_2) \\ c \end{bmatrix} \xrightarrow{1/c} \begin{bmatrix} p_1 + u_1 \\ p_2 + u_2 \end{bmatrix}$$

En Coordenadas Homogéneas...

La forma original en 2D se puede interpretar como si tuviese múltiples copias, cada una escalada de manera uniforme por X3 (para distintos valores de c)

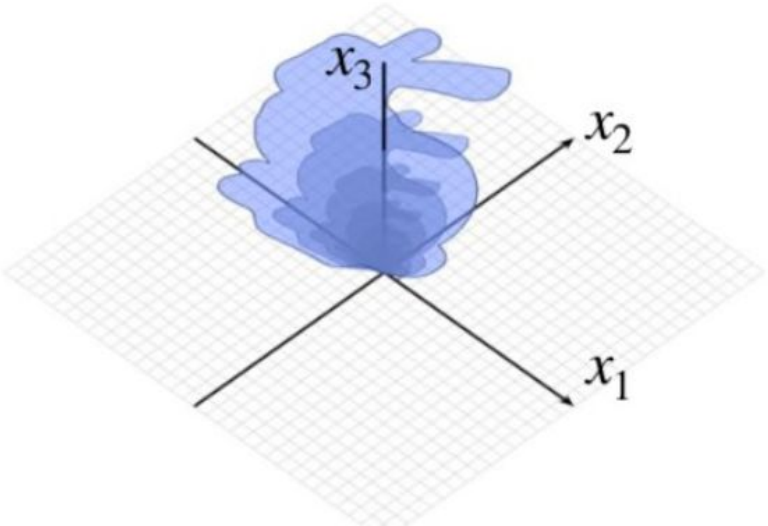


La rotación 2D es una rotación 3D alrededor del eje X3

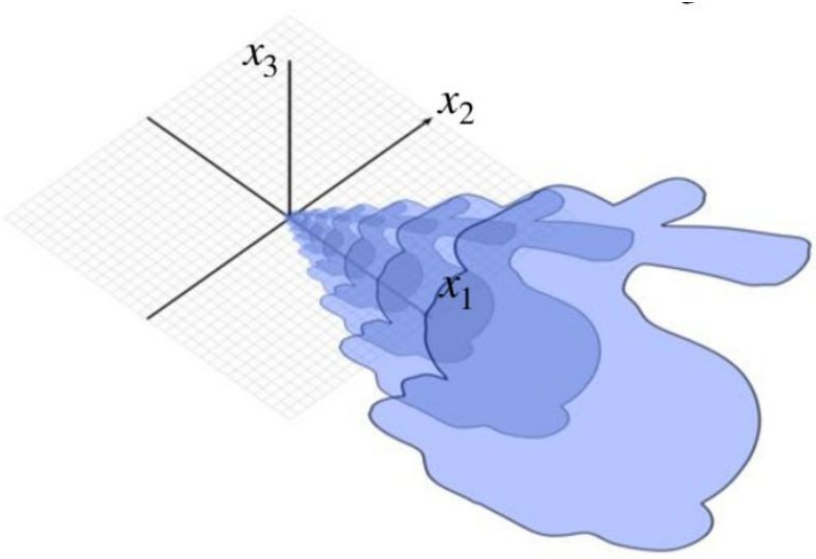


En Coordenadas Homogéneas...

La escala 2D es una escala no uniforme, en X1 y X2, preservando X3.



Traslación 2D es el shear 3D.



Transformaciones 3D en Coordenadas Homogéneas

Ahora es fácil componer transformaciones, porque todas se expresan como matrices 4D.

A las transformaciones lineales en 3D se les agrega una columna, y la traslación es shear en 4D.

point in 3D

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

rotate (x, y, z) around y by θ

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

shear (x, y) by z in (s, t) direction

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

scale x, y, z by a, b, c

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

translate (x, y, z) by (u, v, w)

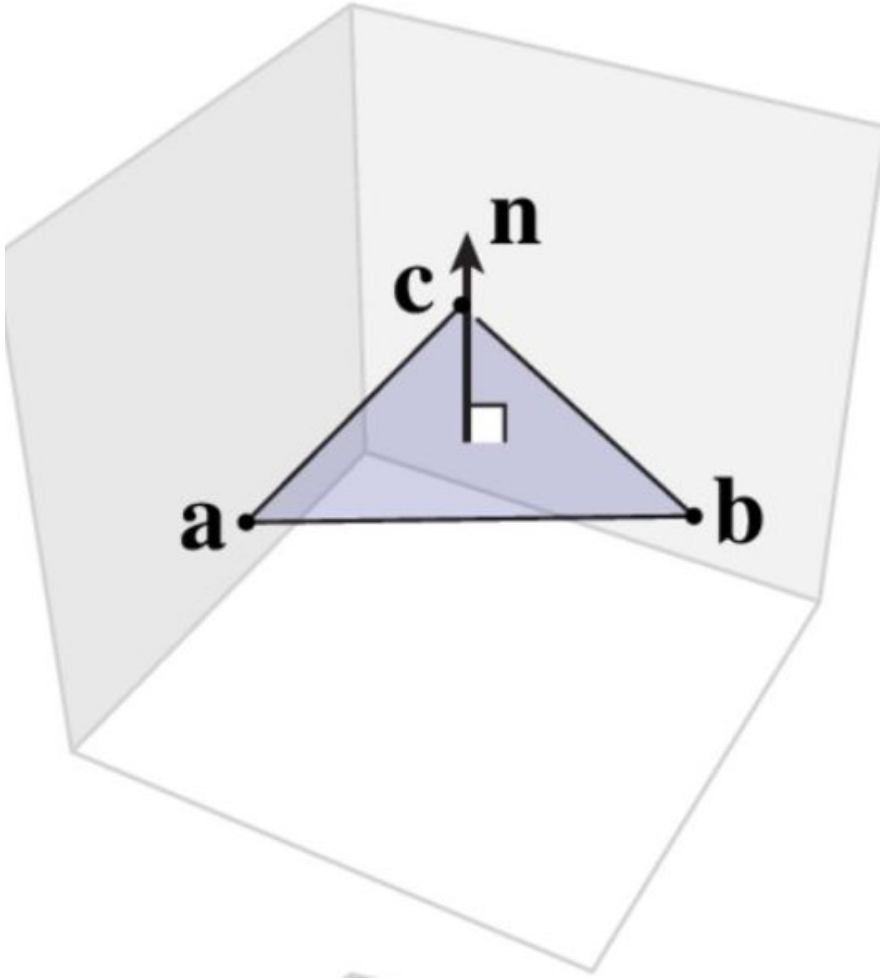
$$\begin{bmatrix} 1 & 0 & 0 & u \\ 0 & 1 & 0 & v \\ 0 & 0 & 1 & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Puntos vs Vectores

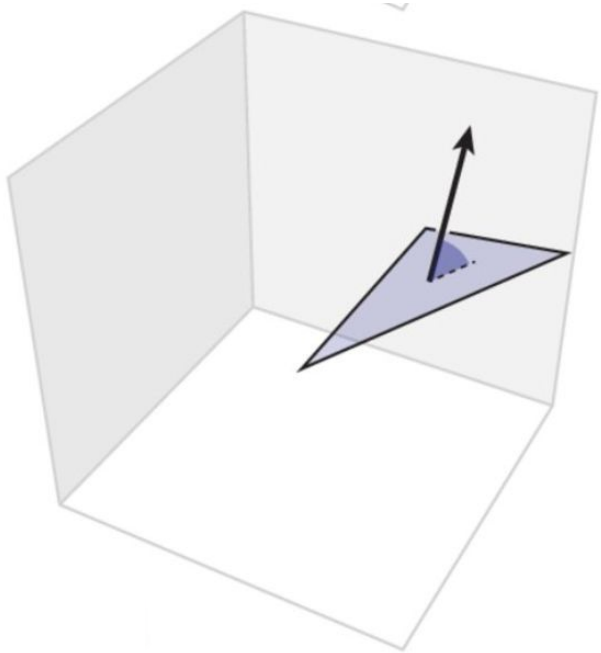
Otra utilidad de las coordenadas homogéneas es la distinción entre **puntos** y **vectores**.

¿Cuál es la diferencia?

¿Qué pasa si transformo este triángulo y su normal?



$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & u \\ 0 & 1 & 0 & v \\ -\sin\theta & 0 & \cos\theta & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Al aplicar la misma transformación a la normal, nos damos cuenta que su normal ya no es ortogonal a la superficie.

Cuando transformamos una normal (un vector), solo queremos que rote, no que se traslade.

La solución es usar la coordenada homogénea 0.

Eso logra que la traslación se ignore y por tanto solo queda la rotación.

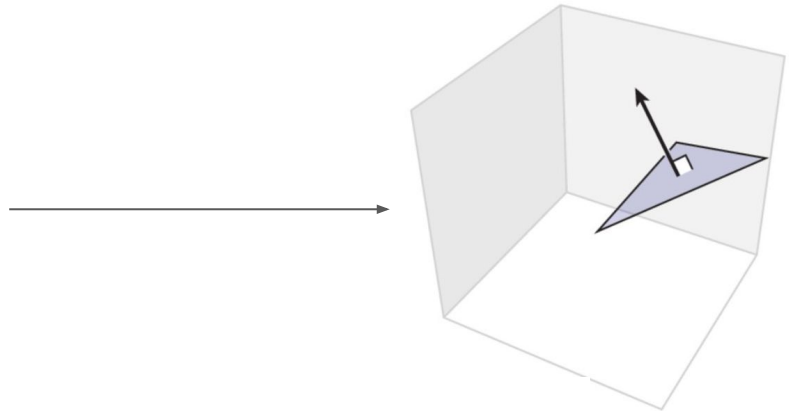
Propuesto: ¿cuál es la implicación matemática de usar c=0? Recordemos que se divide por cero en su definición.

rotate normal around y by θ

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & u \\ 0 & 1 & 0 & v \\ -\sin\theta & 0 & \cos\theta & w \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ 1 \end{bmatrix}$$

translate normal by (u, v, w)

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ 0 \end{bmatrix}$$



Próxima Clase

¡Veremos transformaciones en OpenGL con el Zorzalito!

¿Preguntas?

Propuesto: leer
https://math.stackexchange.com/questions/1570016/why-are-homogenous-coordinates-needed-in-image-projection