Informática Gráfica.
Sesión 3: Espacios y transformaciones afines.

Carlos Ureña, Sept 2025.
Dept. Lenguajes y Sistemas Informáticos.
Universidad de Granada.

Índice

Espacios afines y marcos de referencia 3
Transformaciones afines y matrices de transformación. 31
Transformaciones usuales en Informática Gráfica. 50
Transformaciones en Godot. 81

Sección **1.**

Espacios afines y marcos de referencia

1. Las estructuras de espacio afín y espacio vectorial.
2. Marcos afines y coordenadas homogéneas
3. Producto escalar y vectorial de vectores.

Subsección **1.1.**

Las estructuras de espacio afín y espacio vectorial.

Espacios afines

La noción abstracta de **Espacio Afín** es esencial en Informática Gráfica, ya que permite:

- Razonar sobre los puntos en el espacio,
- Diseñar e implementar representaciones y de esos puntos y estructuras de datos relacionadas en la memoria de los ordenadores, y
- Diseñar e implementar algoritmos que operan sobre esas representaciones y estructuras de datos.

Un espacio afín esencialmente es un modelo de los puntos de un espacio geométrico (de 1,2,3 o más dimensiones) y de las operaciones que podemos hacer con ellos. Está muy relacionado con el concepto de **Espacio Vectorial**

Estructura de *Espacio Vectorial*

Un **espacio vectorial** V_n (con n entero, > 0) es un conjunto de elementos llamados **vectores**, o **vectores libres**, y que notaremos con una flecha: $\vec{u}, \vec{v}, \vec{w}$, etc ..., los cuales interpretamos informalmente como flechas en el espacio, representando un **desplazamiento** desde el origen hasta el destino. El origen es indiferente, lo importante es la distancia y la dirección hasta el destino (por eso se llaman *vectores libres*). En V_n hay definidas estas **dos operaciones**:

Suma de vectores:

Para cualesquiera $\vec{u}, \vec{v} \in V_n$, existe un único $\vec{w} \in V_n$ tal que $\vec{w} = \vec{u} + \vec{v}$. El vector $\vec{w}$ representa un desplazamiento equivalente a aplicar primero el desplazamiento $\vec{u}$ y luego el desplazamiento $\vec{v}$ (o al revés).

Producto por real:

Para cualquier $\vec{u} \in V_n$ y cualquier $a \in \mathbb{R}$, existe un $\vec{v} \in V_n$ tal que $\vec{v} = a\vec{u}$. El vector $\vec{v}$ representa un desplazamiento en la misma dirección que $\vec{u}$, pero con una distancia multiplicada por a (si a es negativo, el sentido es el opuesto).

Axiomas del espacio vectorial

El espacio vectorial V_n cumple las siguientes propiedades, para cualquier $\vec{u}, \vec{v}, \vec{w} \in V_n$ y $a, b \in \mathbb{R}$:

1. Asociatividad de la suma: $(\vec{u} + \vec{v}) + \vec{w} = \vec{u} + (\vec{v} + \vec{w})$
2. Conmutatividad de la suma: $\vec{u} + \vec{v} = \vec{v} + \vec{u}$
3. Elemento identidad de la suma: existe un elemento $\vec{0} \in V_n$ (llamado *vector nulo*) tal que $\vec{u} + \vec{0} = \vec{u}$.
4. Elemento opuesto de la suma: para cada $\vec{u} \in V_n$, existe su *vector opuesto* $-\vec{u} \in V_n$ tal que $\vec{u} + (-\vec{u}) = 0$
5. Elemento identidad del producto: $1\vec{u} = \vec{u}$
6. Asociatividad del producto: $a(b\vec{u}) = (ab)\vec{u}$
7. Distributividad de la suma de vectores y producto: $a(\vec{u} + \vec{v}) = a\vec{u} + a\vec{v}$
8. Distributividad de la suma de reales y producto: $(a + b)\vec{u} = a\vec{u} + b\vec{u}$

(identificar vectores con desplazamientos permite entender estos axiomas).

Bases de un espacio vectorial

Una **base** de un espacio vectorial V_n es un conjunto ordenado de n vectores $\left\{\vec{b}_0, \vec{b}_1, ..., \vec{b}_{n-1}\right\}$, con $\vec{b}_i \in V_n$, que cumplen estas dos propiedades:

- Cualquier vector $\vec{v} \in V_n$ puede expresarse como *combinación lineal* de todos los vectores de la base, es decir, existe un única tupla $(a_0, a_1, ..., a_{n-1}) \in \mathbb{R}^n$ tal que:
$$\vec{v} \;=\; a_0\vec{b}_0 + a_1\vec{b}_1 + ... + a_{n-1}\vec{b}_{n-1}.$$
- Ningún vector de la base puede expresarse como combinación lineal del resto (son **linealmente independientes**).

Cualquier conjunto de n vectores linealmente independientes en V_n es una base del espacio. Se dice que n es la **dimensión** de V_n

La elección de una base permite representar cualquier vector $\vec{v} \in V_n$ mediante sus **coordenadas** $(a_1, a_2, ..., a_n)$, que siempre serán **relativas a dicha base**.

La estructura de *Espacio Afin*

Un **espacio afín** A_n (con n entero, > 0) sobre un espacio vectorial V_n es un conjunto de elementos llamados **puntos**, que notaremos con un punto sobre ellos $\dot{p}, \dot{q}, \dot{r},$ etc ..., los cuales interpretamos informalmente como posiciones en un espacio de n dimensiones. En A_n definimos esta operación:

Suma punto y vector:

Para cualquier punto $\dot{p} \in A_n$ y cualquier vector $\vec{v} \in V_n$, existe un punto $\dot{q} \in A_n$ tal que $\dot{q} = \dot{p} + \vec{v}$.

Intuitivamente, el punto $\dot{q}$ representa la posición a la que se llega al desplazarse desde la posición $\dot{p}$ siguiendo el desplazamiento representado por el vector $\vec{v}$.

Axiomas del espacio afín

El espacio afín A_n cumple las siguientes propiedades, para cualquier $\dot{p}, \dot{q}, \dot{r} \in A_n$ y cualquier $\vec{u}, \vec{v} \in V_n$:

1. Identidad de la suma: $\dot{p} + \vec{0} = \dot{p}$
2. Asociatividad: $(\dot{p} + \vec{v}) + \vec{w} = \dot{p} + (\vec{v} + \vec{w})$
3. Dados dos puntos $\dot{p}$ y $\dot{q}$ cualesquiera, existirá un único vector $\vec{v} \in V_n$ tal que $\dot{q} = \dot{p} + \vec{v}$.

Los axiomas implican que podemos definir la operación:

Resta de puntos: para cualquiera dos puntos $\dot{p}$ y $\dot{q}$ se define el vector $\vec{v} = \dot{q} - \dot{p}$ como el único vector tal que $\dot{p} = \dot{q} + \vec{v}$.

Lógicamente, se cumple $\dot{p} - \dot{p} = \vec{0}$.

Rectas en el espacio afín.

Una **recta** que pasa por $\dot{p}$ y $\dot{q}$ se define como el conjunto de puntos $\dot{r}$ tales que existe un $t \in \mathbb{R}$ tal que:

$$\dot{r} \quad = \quad \dot{p} + t(\dot{q} - \dot{p})$$

- A esta expresión se le denomina **ecuación paramétrica** de la recta.
- Al vector $\vec{v} = \dot{q} - \dot{p}$ se le denomina **vector director** de la recta.
- La ecuación paramétrica permite identificar puntos en un recta con valores reales.
- El valor t puede interpretarse (informalmente) como la *distancia* entre $\dot{r}$ y $\dot{p}$, medida en unidades de la longitud del vector $\dot{q} - \dot{p}$.
- Dos recta son paralelas cuando sus vectores directores son paralelos.

Combinaciones afínes de puntos

Los puntos **no pueden ser multiplicados por reales**, pero se puede definir la **combinación afín** $a\dot{p} + b\dot{q}$ de dos puntos $\dot{p}$ y $\dot{q}$ (usando dos pesos $a, b \in \mathbb{R}$ tales que $a + b$ vale 1 o 0):

- Cuando $a + b = 1$ la combinación afín se define como un punto en la recta que pasa por a y b, así:

$$a\dot{p} + b\dot{q} \quad := \quad \dot{p} + b(\dot{q} - \dot{p})$$

- Cuando $a + b = 0$ la combinación se define como un vector paralelo a la recta que pasa por a y b, así:

$$a\dot{p} + b\dot{q} \quad := \quad b(\dot{q} - \dot{p})$$

Esto permite definir la **multiplicación de un punto $\dot{p}$ por 1 o por 0**:

$$\begin{aligned} 1\dot{p} \quad &:= \quad 1\dot{p} + 0\dot{p} \quad = \quad \dot{p} \\ 0\dot{p} \quad &:= \quad 0\dot{p} + 0\dot{p} \quad = \quad \vec{0} \end{aligned}$$

Subsección **1.2.**

Marcos afines y coordenadas homogéneas

Marcos afines

Un **marco afín** (o *marco de referencia*, o *marco de coordenadas*, o *sistema de coordenadas*) $\mathcal{R}$ en un espacio afín A_n es una tupla compuesta por los n vectores de una **base** de V_n y un punto $\dot{o}$ del A_n (llamado **origen** del marco):

$$\mathcal{R} \;=\; (\vec{e}_0, \vec{e}_1, ..., \vec{e}_{n-1}, \dot{o})$$

Fijado el marco afín $\mathcal{R}$, cualquier punto $\dot{q} \in A_n$ puede expresarse como:

$$\dot{q} \;=\; \dot{p} + a_0\vec{e}_0 + a_1\vec{e}_1 + ... + a_{n-1}\vec{e}_{n-1}$$

e igualmente cualquier vector $\vec{v}$ puede expresarse como:

$$\vec{v} \;=\; a_0\vec{e}_0 + a_1\vec{e}_1 + ... + a_{n-1}\vec{e}_{n-1}$$

donde $a_0, a_2, ..., a_n$ son n valores reales **únicos**, denominados las **coordenadas** del punto $\dot{q}$ o del vector $\vec{v}$ en el marco $\mathcal{R}$.

Coordenadas homogéneas y su espacio afín.

Las **coordenadas homogéneas** de un punto o un vector (relativos a un marco $\mathcal{R}$) forman un **vector columna** con $n+1$ valores reales:

- Los n primeros valores son las coordenadas del punto o vector relativas a $\mathcal{R}$
- El último valor es 1 para puntos y 0 para vectores, se suele escribir w.

El conjunto de todas las posibles tuplas tiene una estructura de espacio afín:

- El subconjunto de las tuplas con $w=1$ es un **espacio afín**, que llamamos $\boldsymbol{A}_n$ ya que dos de estas tuplas pueden restarse y se obtiene otra tupla con $w=0$ (ya fuera de este subconjunto).
- El subconjunto de las tuplas con $w=0$ es un **espacio vectorial**, y en concreto es el espacio vectorial asociado al espacio afín anterior, ya que estas tuplas pueden sumarse entre ellas y multiplicarse por un real.

Fijado un marco afín $\mathcal{R}$ de A_n, hay una correspondencia biunívoca entre A_n y $\boldsymbol{A}_n$

Correspondencia entre tuplas y puntos y vectores

Decimos que los puntos y vectores se obtienen **interpretando** sus coordenadas homogéneas en un marco $\mathcal{R} = (\vec{e}_0, ..., \vec{e}_{n-1}, \dot{o})$, en el sentido siguiente:

- Si $\mathbf{u} = (a_0, ..., a_{n-1}, 1)^T$ son las coordenadas de $\dot{p}$ en $\mathcal{R}$, entonces podemos escribir la igualdad $\dot{p} = \mathcal{R}\mathbf{u}$, ya que

$$\dot{p} \;=\; 1\dot{o} + \sum_{0}^{n-1} a_i \vec{e}_i \;=\; (\vec{e}_0, \vec{e}_1, ..., \vec{e}_{n-1}, \dot{o}) \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \\ 1 \end{pmatrix} \;=\; \mathcal{R}\mathbf{u}.$$

- Si $\mathbf{v} = (a_0, ..., a_{n-1}, 0)^T$ son las coordenadas de $\vec{v}$ en $\mathcal{R}$, entonces podemos escribir la igualdad $\vec{v} = \mathcal{R}\mathbf{v}$, ya que:

$$\vec{v} \;=\; 0\dot{o} + \sum_{0}^{n-1} a_i \vec{e}_i \;=\; (\vec{e}_0, \vec{e}_1, ..., \vec{e}_{n-1}, \dot{o}) \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \\ 0 \end{pmatrix} \;=\; \mathcal{R}\mathbf{v}$$

Operaciones con las tuplas de coordenadas

Un marco afín $\mathcal{R}$ en un espacio afín A_n puede verse como una correspondencia 1 a 1 entre $\boldsymbol{A}_n$ hacia A_n, ya que permite, a partir de una tupla $\mathbf{c}$, obtener su correspondiente punto o vector $\mathcal{R}\mathbf{c}$ (y al revés).

- Esa aplicación es ***lineal*** en el sentido de que **conserva las operaciones de los espacios afines**, es decir para cualquiera tuplas, $\mathbf{u}, \mathbf{v}$ (en $\boldsymbol{V}_n$), y $\mathbf{p}$ (en $\boldsymbol{A}_n$), y real a se cumple:

$$\mathcal{R}(\mathbf{p}+\mathbf{u}) = \mathcal{R}\mathbf{p} + \mathcal{R}\mathbf{u}$$
$$\mathcal{R}(\mathbf{u}+\mathbf{v}) = \mathcal{R}\mathbf{u} + \mathcal{R}\mathbf{v}$$
$$\mathcal{R}(a\mathbf{u}) = a(\mathcal{R}\mathbf{u})$$

- Esto implica que podemos **usar las operaciones con las tuplas para hacer cálculos con puntos y vectores**, esto permite implementar algoritmos que operan en espacios afines o vectoriales.

Ventajas de las coordenadas homogéneas

Las coordenadas homogéneas en $\boldsymbol{A}_n$ se pueden usar para realizar cálculos con puntos y vectores en un ordenador, ya que las usamos como representaciones de los puntos y vectores abstractos de A_n.

El uso de las coordenadas homogéneas **simplifica muchísimo los cálculos con puntos y vectores en Informática Gráfica**:

- Permite representar de forma similar tanto los puntos como los vectores.
- Permite implementar la *transformaciones afines* con matrices.
- Simplifica los cálculos de proyección perspectiva.

Es importante recordar siempre que unas coordenadas no tienen sentido de forma independiente del marco de coordenadas al cual son relativas.

Marcos afines en IG

En Informática Gráfica se usan diversos marcos de coordenadas distintos:

- Marco de mundo: global a una escena.
- Marco de objeto: específico de un objeto concreto dentro de la escena.
- Marco de cámara: para coordenadas relativas a una cámara virtual posicionada y orientada de algna forma concreta dentro de una escena.
- Marco del dispositivo: para coordenadas en una ventana o una imagen, en unidades de pixels.
- Marco normalizado de dispositivo: para coordenadas en una ventana o una imagen, en unidades normalizadas entre -1 y $+1$.

El resultado anterior nos indica que **podemos usar matrices para convertir las coordenadas relativas a un marco en las coordenadas relativas a otro marco**, lo cual es esencial en Informática Gráfica.

Subsección **1.3.**

Producto escalar y vectorial de vectores.

La base especial W_n de un espacio afín A_n

En Informática Gráfica, si representamos objetos reales usando un espacio afín abstracto A_n, entonces necesitamos trasladar a A_n los conceptos de *distancia* y *perpendicularidad* que usamos en el espacio físico real. Para ello, designamos una base de V_n llamada la **base especial**

$$W_n \quad = \quad (\hat{e}_0, \hat{e}_1, ..., \hat{e}_{n-1})$$

cuyos vectores tienen **longitud unidad** y son perpendiculares dos a dos **por definición**. Usamos $\hat{}$ en lugar de $\vec{}$ para estos vectores, y les llama **versores**. En 2D escribiremos $W_2 = (\hat{x}, \hat{y})$, y en 3D $W_3 = (\hat{x}, \hat{y}, \hat{z})$. Una vez seleccionada W_n

- podemos definir la *distancia* en A_n, (ya que W_n define las unidades de distancia en todas las direcciones posibles),
- podemos decir si dos vectores son perpendiculares o no,
- definimos el *ángulo* entre dos vectores no nulos, y
- podemos decir cuando un marco afín es un *marco cartesiano* o no lo es.

Producto escalar de vectores

El **producto escalar** (***dot product***) es una función que se aplica dos vectores $\vec{u}, \vec{v} \in V_n$ y produce un valor real que se nota como $\vec{u} \cdot \vec{v}$.

El producto escalar cumple estas propiedades:

- Conmutativa: $\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u}$
- Distributiva respecto a la suma: $\vec{u} \cdot (\vec{v} + \vec{w}) = \vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w}$
- Asociativa respecto al producto por reales: $(a\vec{u}) \cdot \vec{v} = a(\vec{u} \cdot \vec{v})$
- Positivo definido: $\vec{u} \cdot \vec{u} \geq 0$

Muchas posibles definiciones distintas del producto escalar pueden cumplir estas propiedades, así que necesitamos también exigir que para dos vectores $\vec{e}_i$ y $\vec{e}_j$ de la base W_n se cumpla:

$$\hat{e}_i \cdot \hat{e}_j \quad = \quad \delta_{i,j} \quad := \quad \begin{cases} 1 \text{ si } i = j \\ 0 \text{ si } i \neq j \end{cases}$$

Longitud de un vectores. Versores.

La **norma** (o **módulo**, o **longitud**) de un vector $\vec{u} \in V_n$ es el número real no negativo que se define como:

$$\|\vec{u}\| \ := \ \sqrt{\vec{u} \cdot \vec{u}}$$

- La norma de un vector es siempre un valor real no negativo, y representa la longitud del desplazamiento que representa el vector.
- La **distancia** entre dos puntos $\dot{p}$ y $\dot{q}$ se define como el valor real $\|\dot{q} - \dot{p}\|$ (es decir, la longitud del vector que los une).
- Si sabemos que un vector tiene longitud 1, se le denomina **versor** (o **vector unitario**)., y se escribe con un gorro: $\hat{x}, \hat{y}, \hat{z}, \hat{e}, \hat{a}$ etc...
- Los vectores de la base W_n son unitarios, y por eso los hemos escrito con un gorro.

Perpendicularidad y ángulo entre vectores

Una vez bien definido el producto escalar, podemos definir la noción de **perpendicularidad** y el **ángulo** entre dos vectores $\vec{u}$ y $\vec{v}$ (ninguno nulo):

- Si se cumple que $\vec{u} \cdot \vec{v} = 0$, entonces se dice que los vectores $\vec{u}$ y $\vec{v}$ son **perpendiculares** (u **ortogonales**).
- El **ángulo** θ entre $\vec{u}$ y $\vec{v}$ es un valor real (en radianes), en el rango $[0, \pi]$, se define como:

$$\theta \quad := \quad \arccos\left(\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \ \|\vec{v}\|}\right),$$

  y por tanto, para dos versores $\hat{a}$ y $\hat{b}$ con ángulo θ entre ellos se cumple:

$$\cos(\theta) \quad = \quad \hat{a} \cdot \hat{b}.$$

- Los versores de W_n son **perpendiculares** dos a dos por la definición de W_n.

Producto vectorial en 3D

Consideramos ahora el espacio vectorial de las tuplas de 4 componentes con $w = 0$ (son coordenadas de vectores en 3D). En ese espacio se puede definir la operación de **producto vectorial** (***cross product***) entre dos vectores $\vec{u}, \vec{v}$, que produce un otro vector $\vec{w} = \vec{u} \times \vec{v}$. Esta operación cumple estas propiedades:

- **Anticonmutativa** : $\vec{u} \times \vec{v} = -(\vec{v} \times \vec{u})$
- **Distributiva** respecto a la suma: $\vec{u} \times (\vec{v} + \vec{w}) = \vec{u} \times \vec{v} + \vec{u} \times \vec{w}$
- **Asociativa** respecto al producto por reales: $(a\vec{u}) \times \vec{v} = a(\vec{u} \times \vec{v})$
- **Perpendicularidad**: se cumple $\vec{u} \cdot (\vec{u} \times \vec{v}) = 0$, implica que $\vec{u} \times \vec{v}$ es perpendicular a $\vec{u}$ y a $\vec{v}$.

Hay muchas posibles definiciones del producto vectorial que cumplen estas propiedades. Para definirlo correctamente se exige que si $W_3 = (\hat{x}, \hat{y}, \hat{z})$, entonces se cumpla:

$$\hat{x} \times \hat{y} = \hat{z} \qquad \hat{y} \times \hat{z} = \hat{x} \qquad \hat{z} \times \hat{x} = \hat{y}$$

Marcos ortogonales y ortonormales

Dado un marco $\mathcal{R} = \left(\vec{b}_0, ..., \vec{b}_n, \dot{o}\right)$, decimos que ese marco es:

Ortogonal: si los vectores de la base son perpendiculares dos a a dos, es decir:

$$i \neq j \qquad \Longrightarrow \qquad \vec{b}_i \cdot \vec{b}_j = 0$$

Ortonormal: si es ortogonal y además todos los vectores de la base son unitarios (son versores), es decir:

$$\vec{b}_i \cdot \vec{b}_i \;\; = \;\; 1$$

Definimos **la matriz de productos escalares** M asociada a $\mathcal{R}$ como la matriz con $a_{ij} := \vec{b}_i \cdot \hat{e}_j$, donde cada $\hat{e}_i$ es un versor de W_n. Entonces se cumple:

- Si $\mathcal{R}$ es ortogonal, M también lo es.
- Si $\mathcal{R}$ es ortonormal, M también lo es (tendrá determinante $+1$ o -1).

Orientación de un marco. Marcos cartesianos.

Un marco $\mathcal{R}$ cualquiera puede tener orientación a *derechas* o a *izquierdas*, la orientación depende de la base de $\mathcal{R}$, en concreto del signo del determinante de la matriz M de productos escalares asociada a $\mathcal{R}$

- Si es positivo, será a ***derechas*** (W_n es a derechas **por definición**).
- Si es negativo, será a ***izquierdas***.

Un marco $\mathcal{R}$ es **cartesiano** si y solo si es ortonormal y además tiene orientación a derechas (es decir: el determinante de M es $+1$).

- Eso implica que los vectores en la base de $\mathcal{R}$ se pueden hacer coincidir con los de W_n mediante una misma *rotación* aplicada a todos ellos.
- Con esta definición, un marco cuya base es W_n es **cartesiano** por definición, pero en un espacio afín hay otros muchos otros marcos también cartesianos.

Cálculo de producto escalar con coordenadas cartesianas

En el espacio vectorial de las tuplas con $w = 0$, el producto escalar de dos tuplas de coordenadas (de vectores) $\mathbf{u} = (a_0, ..., a_{n-1}, 0)^T$ y $\mathbf{v} = (b_0, ..., b_{n-1}, 0)^T$ se escribe como $\mathbf{u} \cdot \mathbf{v}$ y es igual a la suma del producto componente a componente:

$$\mathbf{u} \cdot \mathbf{v} \quad = \quad \sum_{0}^{n-1} a_i b_i$$

Sea $\mathcal{C}$ es un marco cartesiano, y $\mathbf{u}$ y $\mathbf{v}$ las coordenadas en $\mathcal{C}$ de dos vectores $\vec{u}$ y $\vec{v}$ entonces el producto escalar de los vectores se puede calcular fácilmente usando el producto escalar de sus coordenadas, es decir, se cumple:

$$\vec{u} \cdot \vec{v} \quad = \quad (\mathcal{C}\mathbf{u}) \cdot (\mathcal{C}\mathbf{v}) \quad = \quad \mathbf{u} \cdot \mathbf{v}$$

Esto se cumple para cualquier marco cartesiano, es decir, el producto escalar de coordenadas es invariante entre marcos cartesianos.

Cálculo de producto vectorial con coordenadas cartesianas

En el espacio vectorial de las tuplas con $w = 0$ y $n = 3$ (coordenadas de vectores en 3D), el producto vectorial de dos tuplas de coordenadas $\mathbf{u} = (x_0, y_0, z_0, 0)$ y $\mathbf{v} = (x_1, y_1, z_1, 0)$ se escribe como $\mathbf{u} \times \mathbf{v}$ y se define como:

$$\mathbf{u} \times \mathbf{v} \quad = \quad \begin{pmatrix} y_0 z_1 - z_0 y_1 \\ z_0 x_1 - x_0 z_1 \\ x_0 y_1 - y_0 x_1 \end{pmatrix}$$

Sea $\mathcal{C}$ es un marco cartesiano, y $\mathbf{u}$ y $\mathbf{v}$ las coordenadas en $\mathcal{C}$ de dos vectores $\vec{u}$ y $\vec{v}$: entonces el producto vectorial de los vectores se puede calcular fácilmente usando el producto escalar de sus coordenadas, es decir, se cumple:

$$\vec{u} \times \vec{v} \quad = \quad (\mathcal{C}\mathbf{u}) \times (\mathcal{C}\mathbf{v}) \quad = \quad \mathcal{C}(\mathbf{u} \times \mathbf{v})$$

Esto se cumple para cualquier marco cartesiano, es decir, el producto vectorial de coordenadas es invariante entre marcos cartesianos.

Problemas: cálculo del prod. escalar y vectorial.

Problema 3.1:

Demuestra que efectivamente el producto escalar de dos vectores se puede calcular (usando sus coordenadas en cualquier marco cartesiano) como la suma del producto componente a componente. Usa las propiedades que definen dicho producto escalar.

Problema 3.2:

Demuestra que el producto vectorial de dos vectores se puede calcular usando sus coordenadas en cualquier marco cartesiano según se ha indicado.

Problema 3.3:

Demuestra que el producto vectorial de dos vectores es perpendicular a cada uno de esos dos vectores.

Sección **2.**

Transformaciones afines y matrices de transformación.

1. Transformaciónes afines
2. Transformación afín de coordenadas. Matrices.
3. Tipos de transformaciones.

Transformaciones geométricas

Una **transformación geométrica** T es una aplicación desde un espacio afín A_n en otro B_n con la misma dimensión, donde A_n y B_n pueden ser distintos o **el mismo espacio**.

La transformación T se aplica a los puntos de A_n y produce puntos en B_n. Si $\dot{q}$ es la imagen por T de $\dot{p}$, escribimos:

$$\dot{q} \;=\; T(\dot{p})$$

En general, una transformación geométrica

- aplicada a una recta puede producir otra recta, o bien una curva,
- puede ser continua o no serlo,
- puede no tener inversa si mas de un punto de A_n tiene como imagen un mismo punto en B_n.

Como consecuencia de todo esto, una transformación geométrica en general **no puede aplicarse a los vectores libres**, únicamente a los puntos.

Función asociada a una transformación geométrica

Si fijamos un marco $\mathcal{R}$ del espacio A_n y una transformación geométrica T, entonces existirá una función F que asocia cada tupla $\mathbf{c}$ de coordenadas de un punto $\dot{p}$ con la correspondiente tupla $\mathbf{c}' = F(\mathbf{c})$ de coordenadas del punto transformado por T, es decir, se cumple:

$$T(\dot{p}) \;=\; T(\mathcal{R}\mathbf{c}) \;=\; \mathcal{R}F(\mathbf{c}) \;=\; \mathcal{R}\mathbf{c}'$$

- La llamamos la **funciones asociadas a la transformación T en el marco $\mathcal{R}$**.
- Esa función es en realidad una transformación geométrica en el espacio de las tuplas de coordenadas de puntos, es decir, en $\boldsymbol{A}_n$.
- La función F se puede usar para implementar la transformación en un programa.
- Por supuesto, esta función depende del marco $\mathcal{R}$ que hemos seleccionado, para otro marco sería distinta.

Subsección **2.1.**

Transformaciónes afines

Transformaciones afines

Una transformación geométrica T es una **transformación geométrica afín** si T **es continua** y **transforma cada paralelogramo en otro paralelogramo**. Un paralelogramo está definido por 4 puntos, con lados iguales dos a dos. Formalmente:

- Para cualquiera cuatro puntos $\dot{p}, \dot{q}, \dot{r}, \dot{s}$, si forman un paralelogramo, entonces sus imágenes forman otro, es decir:

$$\dot{q} - \dot{p} \;=\; \dot{s} - \dot{r} \qquad \Longrightarrow \qquad T(\dot{q}) - T(\dot{p}) \;=\; T(\dot{s}) - T(\dot{r})$$

- Esa propiedad permite **aplicar T a vectores y producir vectores**. Si $\vec{v} = \dot{q} - \dot{p}$, entonces se cumple que:

$$T(\vec{v}) \;=\; T(\dot{q} - \dot{p}) \;:=\; T(\dot{q}) - T(\dot{p})$$

  el vector resultado está bien definido pues no importa que par $\dot{p}$ y $\dot{q}$ seleccionemos: siempre que $\vec{v} = \dot{q} - \dot{p}$, se obtendrá el mismo vector $T(\vec{v})$.

Linealidad de las transformaciones afines

Se puede demostrar fácilmente que si T es afín, entonces para cualquiera dos vectores $\vec{u}$ y $\vec{v}$ se cumplirá que:

$$T(\vec{u}+\vec{v}) \;=\; T(\vec{u})+T(\vec{v}).$$

También sabemos que T es **lineal**: para un punto $\dot{p}$, un vector $\vec{v}$, y un real a:

$$T(\dot{p}+a\vec{v}) \;=\; T(\dot{p})+aT(\vec{v}).$$

Esto implica que T:

- **transforma líneas rectas en líneas rectas**, conservando la ecuación paramétrica de dichas rectas, y por tanto,
- **conserva las proporciones** entre las longitudes de vectores paralelos, y
- **conserva el paralelismo**: transforma dos líneas paralelas en otras dos líneas paralelas.

Transformaciones afines singulares y no singulares

Podemos distinguir las siguientes dos clases de transformaciones afines, en función de si son una biyección o no:

- Una transformación afín T será **no singular** si siempre aplica puntos distintos en puntos distintos, es decir, se cumple:

$$\dot{q} - \dot{p} \neq \vec{0} \qquad \Longrightarrow \qquad T(\dot{q}) \neq T(\dot{p})$$

  es lo mismo que decir que la imagen de cualquier vector no nulo es otro vector no nulo. Una transformación no singular es biyectiva y **tiene inversa**, T^{-1}.
- Una transformación afín será **singular** si hay al menos dos puntos con la misma imagen, o equivalentemente, hay al menos un vector no nulo que se transforma en el vector nulo. Una transformación singular **no tiene inversa**.
- Un ejemplo de transformación afín singular es una que pone a cero una componente de los vectores (por ejemplo la componente X): anula cualquier vector paralelo a ese eje.

Subsección **2.2.**

Transformación afín de coordenadas. Matrices.

La matriz asociada a una transformación

Fijado un marco $\mathcal{R} = \left(\vec{b}_0, ..., \vec{b}_{n-1}, \dot{o}\right)$ y una transformación T (con función F en $\mathcal{R}$) consideramos los vectores de la base y el origen pero transformados por T, es decir los vectores $T\left(\vec{b}_j\right)$ y el punto $T(\dot{o})$. Sus coordenadas en $\mathcal{R}$ son $\mathbf{b}_j$ y $\mathbf{o}$, respectivamente, es decir:

$$\vec{b}_j = \mathcal{R}\mathbf{b}_j \quad \text{y} \quad \dot{o} = \mathcal{R}\mathbf{o} \qquad \text{donde:} \quad \begin{cases} \mathbf{b}_j = \left(b_{0,j}, ..., b_{n-1,j}, 0\right)^T \\ \mathbf{o} \ = \left(o_0, ..., o_{n-1}, 1\right)^T \end{cases}$$

Entonces podemos construir la **matriz asociada a T**, con $n+1$ filas y columnas, y que tiene las tuplas $\mathbf{b}_j$ y $\mathbf{o}$ dispuestas **por columnas**, y que escribimos como M_T:

$$M_T \ := \begin{pmatrix} b_{0,0} & \cdots & b_{0,n-1} & o_0 \\ \vdots & & \vdots & \vdots \\ b_{n-1,0} & \cdots & b_{n-1,n-1} & o_{n-1} \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

La función asociada a una transformación afín

Consideramos la función F la asociada a una transformación afín T en el marco $\mathcal{R}$ anterior. Sean $\mathbf{c} = (c_0, ..., c_{n-1}, w)$ las coordenadas en (en $\mathcal{R}$) de un punto o vector $\mathcal{R}\mathbf{c}$ cualquiera:

Sabemos que F *implementa* T, es decir $T(\mathcal{R}\mathbf{c}) = \mathcal{R}F(\mathbf{c})$. Puesto que T y $\mathcal{R}$ son lineales, para un punto $\dot{p} = \mathcal{R}\mathbf{p}$ y un vector $\vec{v} = \mathcal{R}\mathbf{v}$, y un real a se cumplirá:

$$T(\dot{p} + a\vec{v}) \quad = \quad \mathcal{R}(F(\mathbf{p}) + aF(\mathbf{v}))$$

Como consecuencia, las coordenadas transformadas $F(\mathbf{c})$ se pueden escribir usando las coordenadas transformadas de $\mathcal{R}$ y la tupla $\mathbf{c}$:

$$R\mathbf{c} \quad = \quad w\dot{o} + \sum_{j=0}^{n-1} c_j \vec{b}_j \qquad \Longrightarrow \qquad F(\mathbf{c}) \quad = \quad w\mathbf{o} + \sum_{j=0}^{n-1} c_j \mathbf{b}_j$$

Lo cual implica que la función F se puede expresar usando la matriz M_T:

$$F(\mathbf{c}) \quad = \quad M_T\mathbf{c}$$

Implementación de transformaciones afines con matrices

El resultado anterior nos dice que **podemos implementar una transformación afín usando su matriz asociada**:

- Eso permite implementar transformaciones afines en un programa, simplemente multiplicando la matriz (a la izquierda) por las coordenadas homogéneas del punto o vector a transformar (a la derecha).
- Si T es no singular, entonces M_T será invertible, y podremos implementar la transformación inversa T^{-1} usando la inversa de la matriz M_T^{-1}.
- La matriz M_T se puede descomponer como el producto de otras matrices cuadradas

$$M_T \quad = \quad D_T R_T$$

  es decir, aplicar M_T equivale a aplicar primero R_T y luego D_T, donde:

  - R_T es la matriz que transforma los vectores, y
  - D_T es la matriz que desplaza el origen.

Descomposición de matrices en 3D.

A modo de ejemplo, en 3D, consideramos un marco $\mathcal{R} = \left(\vec{e}_x, \vec{e}_y, \vec{e}_z, \dot{o}\right)$ y una transformación afín T, consideramos las coordenadas (en $\mathcal{R}$) de los vectores de la base y el origen transformados:

$$\begin{aligned} T(\vec{e}_x) &= \mathcal{R}(x_0, y_0, z_0, 0)^T & \qquad T\left(\vec{e}_y\right) &= \mathcal{R}(x_1, y_1, z_1, 0)^T \\ T(\vec{e}_z) &= \mathcal{R}(x_2, y_2, z_2, 0)^T & \qquad T(\dot{o}) &= \mathcal{R}\left(d_x, d_y, d_z, 1\right)^T \end{aligned}$$

Entonces se cumple:

$$M_T = \begin{pmatrix} x_0 & x_1 & x_2 & d_x \\ y_0 & y_1 & y_2 & d_y \\ z_0 & z_1 & z_2 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 & x_1 & x_2 & 0 \\ y_0 & y_1 & y_2 & 0 \\ z_0 & z_1 & z_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = D_T R_T$$

Composición de transformaciones y matrices. Inversas.

Las transformaciones afines se pueden componer: si T_1 y T_2 son dos transformaciones afines:

- La composición de ambas es otra transformación afín S, que se escribe como $T_2 \circ T_1$ y que supone transformar primero por T_1 y luego por T_2, es decir, para un punto $\dot{p}$ se cumple:

$$S(\dot{p}) \quad = \quad T_2(T_1(\dot{p}))$$

- En un marco fijo, la matriz M_S asociada a $T_2 \circ T_1$ se obtiene multiplicando las matrices M_2 y M_1 correspondientes a T_2 y T_1, respectivamente:

$$M_S \quad = \quad M_2 M_1$$

- Puesto que $M_T = D_T R_T$, la inversa de M_T se puede escribir como:

$$M_T^{-1} \quad = \quad (R_T D_T)^{-1} \quad = \quad R_T^{-1} D_T^{-1}$$

Transformación de marcos afines

Dada una transformación T **no singular** y el marco afín $\mathcal{R} = \left(\vec{b}_0, ..., \vec{b}_{n-1}, \dot{o}\right)$ podemos definir el marco afín *transformado* $\mathcal{S}$ así:

$$\mathcal{S} := \left(T\left(\vec{b}_0\right), ..., T\left(\vec{b}_{n-1}\right), T(\dot{o})\right) \quad \text{donde:} \quad \begin{cases} T\left(\vec{b}_j\right) = \mathcal{R}\left(b_{0,j}, ..., b_{n-1,j}, 0\right)^T \\ T(\dot{o}) = \mathcal{R}(o_0, ..., o_{n-1}, 1)^T \end{cases}$$

Entonces se pueden escribir los vectores de la base y el origen transformados como combinaciones lineales de los vectores de la base y el origen originales:

$$T\left(\vec{b}_j\right) = 0\dot{o} + \sum_{i=0}^{n-1} b_{i,j}\vec{b}_i \qquad \text{y} \qquad T(\dot{o}) = 1\dot{o} + \sum_{i=0}^{n-1} o_i\vec{b}_i$$

Lo cual significa que podemos escribir el marco $\mathcal{S}$ **multiplicando el marco $\mathcal{R}$ por la matriz M_T** (por la derecha):

$$\mathcal{S} = \mathcal{R}M_T$$

donde M_T es la matriz asociada a T.

Cambio de coordenadas entre marcos afines

Dados dos marcos afines $\mathcal{A}$ y $\mathcal{B}$, siempre existe una transformación afín T que transforma $\mathcal{A}$ en $\mathcal{B}$. Dicha transformación tendrá asociada una matriz M (en $\mathcal{A}$) tal que $\mathcal{A}M = \mathcal{B}$. Podemos nombrar esa matriz como $M_{\mathcal{AB}}$, tiene las coordenadas de $\mathcal{B}$ en $\mathcal{A}$.

Si un punto o vector tiene coordenadas $\mathbf{c}_{\mathcal{A}}$ en $\mathcal{A}$ y $\mathbf{c}_{\mathcal{B}}$ en $\mathcal{B}$, entonces se cumple:

$$\mathcal{A}\mathbf{c}_{\mathcal{A}} = \mathcal{B}\mathbf{c}_{\mathcal{B}}$$

Pero podemos sustituir $\mathcal{B}$ por $\mathcal{A}M_{\mathcal{AB}}$ y usando la asociatividad, obtenemos:

$$\mathcal{A}c_{\mathcal{A}} = \mathcal{B}\mathbf{c}_{\mathcal{B}} = (\mathcal{A}M_{\mathcal{AB}})\mathbf{c}_{\mathcal{B}} = \mathcal{A}M_{\mathcal{AB}}\mathbf{c}_{\mathcal{B}} = \mathcal{A}(M_{\mathcal{AB}}\mathbf{c}_{\mathcal{B}})$$

Puesto que las coordenadas e n $\mathcal{A}$ son únicas, se deduce que:

$$\mathbf{c}_{\mathcal{A}} = M_{\mathcal{AB}}\mathbf{c}_{\mathcal{B}}$$

Es decir, **la matriz $M_{\mathcal{AB}}$ permite hacer el cambio de coordenadas desde las coordenadas en $\mathcal{B}$ a las coordenadas en $\mathcal{A}$.**

Interpretaciones de la acción de una matriz

Si tenemos una matriz M cualquiera (con determinante no nulo), y un marco afín $\mathcal{A}$, siempre existe una transformación afín T con matriz M que transforma $\mathcal{A}$ en otro marco $\mathcal{B}$. Esto implica que podemos ver la matriz M de varias formas distintas:

- Como algo que implementa la función F de T: dadas las coordenadas $\mathbf{c}$ de un punto o vector relativas a $\mathcal{A}$, nos da las coordenadas $F(\mathbf{c})$ del punto o vector transformado, también relativas a $\mathcal{A}$:

$$F(\mathbf{c}) \quad = \quad M\mathbf{c}$$

- Permite hacer el cambio de coordenadas desde las coordenadas en $\mathcal{B}$ a las coordenadas en $\mathcal{A}$. Si $\mathbf{c}_{\mathcal{B}}$ son coordenadas en $\mathcal{B}$ y $\mathbf{c}_{\mathcal{A}}$ las correspondientes en $\mathcal{A}$ (ambas de un mismo punto o vector), entonces:

$$\mathbf{c}_{\mathcal{A}} \quad = \quad M\mathbf{c}_{\mathcal{B}}$$

- Como algo que transforma el marco $\mathcal{A}$ en el marco $\mathcal{B}$ ya que

$$\mathcal{B} = \mathcal{A}M$$

Subsección **2.3.**

Tipos de transformaciones.

Transformaciones isométricas

Una transformación T es una transformación **isométrica** si conserva el valor absoluto del producto escalar, es decir, para dos vectores cualquiera $\vec{u}$ y $\vec{v}$ se cumple:

$$\| T(\vec{u}) \cdot T(\vec{v}) \| = \| \vec{u} \cdot \vec{v} \|$$

Como consecuencia T

- es una transformación afín, pero no conserva *la orientación*,
- conserva las distancias entre puntos, es decir conserva la longitud de los vectores: se cumple $\|T(\vec{v})\| = \|\vec{v}\|$,
- conserva el valor absoluto del ángulo entre dos líneas,
- conserva las áreas y volúmenes de los objetos, y
- tendrá asociada una matriz R_T (en un marco cartesiano cualquiera) que será ortonormal, y con determinante igual a $+1$ o a -1.

Las **traslaciones, las rotaciones y las reflexiones** son ejemplos de isometrías.

Transformaciones rígidas.

Una transformación T que conserva el producto escalar (incluyendo el signo) es una transformación **rígida** u **ortogonal**. Para dos vectores cualquiera $\vec{u}$ y $\vec{v}$ se cumple:

$$T(\vec{u}) \cdot T(\vec{v}) \quad = \quad \vec{u} \cdot \vec{v}$$

Por tanto T:

- es afín y conserva la *orientación*,
- es isométrica,
- conserva las distancias, áreas y volúmenes,
- conserva los ángulos (en magnitud y signo), y
- tendrá asociada una matriz R_T (en un marco cartesiano cualquiera) que será ortonormal, y con determinante igual a $+1$.

Las **traslaciones y rotaciones** son rígidas (no las reflexiones).

Sección **3.**

Transformaciones usuales en Informática Gráfica.

1. Traslaciones
2. Escalados y reflexiones.
3. Cizallas
4. Rotaciones

Introducción

En esta sección estudiaremos las transformaciones usuales en Informática Gráfica, tanto en el espacio 2D como 3D.

- Traslaciones.
- Escalados: uniformes, no uniformes, reflexiones.
- Cizallas (*shearings*).
- Rotaciones: entorno a al origen (en 2D), entorno a los ejes de coordenadas o entorno a un eje arbitrario (en 3D).

En cada caso vemos las propiedades y las correspondientes matrices, relativas siempre a un marco $\mathcal{R}$ en 2D o 3D (que en algunos casos será cartesiano). Para introducir las transformaciones se indica como se transforma el origen y los versores de dicho marco cartesiano, y eso nos da directamente la matriz.

Subsección **3.1.**

Traslaciones

Traslaciones en 2D y 3D

Una **traslación** T por un vector $\vec{d}$ es una transformación afín rígida que a cada punto $\dot{p}$ le asocia el punto $\dot{p} + \vec{d}$, y lógicamente no afecta a los vectores.

En el espacio 2D, para un marco $\mathcal{R} = (\vec{x}, \vec{y}, \dot{o})$, la traslación T por el vector $\vec{d}$ de coordenadas $\mathbf{d} = \left(d_x, d_y, 0\right)^T$ en $\mathcal{R}$ viene dada por:

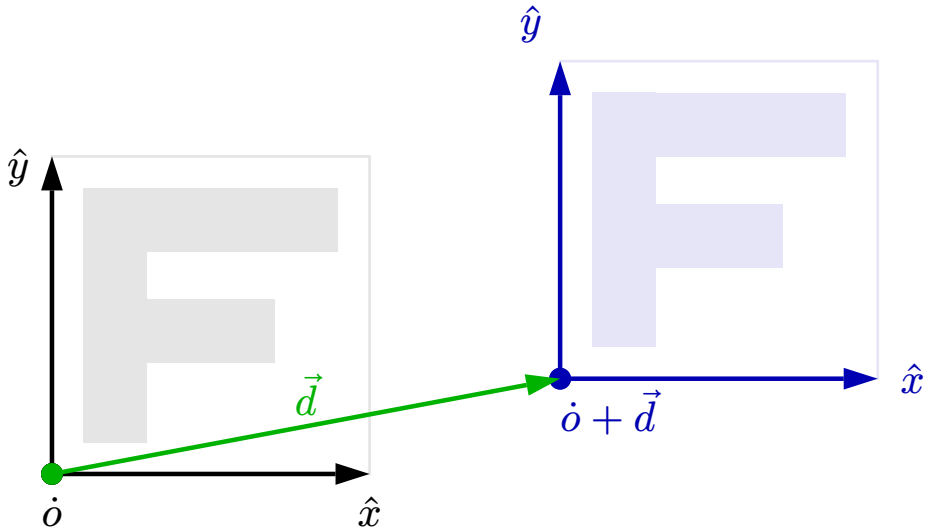
$$\begin{array}{rcl} T_{\mathbf{d}}(\vec{x}) & = & \vec{x} \\ T_{\mathbf{d}}(\vec{y}) & = & \vec{y} \\ T_{\mathbf{d}}(\dot{o}) & = & \dot{o} + \vec{d} \end{array} \qquad M_T = \begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

En 3D, para un marco $\mathcal{R} = (\vec{x}, \vec{y}, \vec{z}, \dot{o})$, la traslación T por el vector $\vec{d}$ de coordenadas $\mathbf{d} = \left(d_x, d_y, d_z, 0\right)^T$ en $\mathcal{R}$ viene dada por:

$$\begin{array}{rcl} T_{\mathbf{d}}(\vec{x}) & = & \vec{x} \\ T_{\mathbf{d}}(\vec{y}) & = & \vec{y} \\ T_{\mathbf{d}}(\vec{z}) & = & \vec{z} \\ T_{\mathbf{d}}(\dot{o}) & = & \dot{o} + \vec{d} \end{array} \qquad M_T = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Esquema de la traslación en 2D

Vemos una traslación en 2D por un vector $\mathbf{d} = (1.6, 0.3)$, en negro el marco original y en azul el desplazado. En verde vemos el vector de desplazamiento. Los ejes no cambian.



Subsección **3.2.**

Escalados y reflexiones.

Escalados.

Una **escalado** E_{s} es una transformación afín que multiplica las coordenadas de un punto o vector en un marco $\mathcal{R}$ cualquiera por n factores escalares que forman la tuplas $\mathrm{s} = (s_0, ..., s_{n-1})$. Las distancias al origen se ven por tanto multiplicadas por esos factores. El origen de $\mathcal{R}$ no cambia, y se llama **foco de escalado**.

En 2D, la tupla de los dos factores es: $\mathrm{s} = \left(s_x, s_y\right)$

$$\begin{aligned} E_{\mathrm{s}}(\vec{x}) &= s_x\vec{x} \\ E_{\mathrm{s}}(\vec{y}) &= s_y\vec{y} \\ E_{\mathrm{s}}(\dot{o}) &= \dot{o} \end{aligned} \qquad M_T = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

En 3D, la tupla es: $\mathrm{s} = \left(s_x, s_y, s_z\right)$

$$\begin{aligned} E_{\mathrm{s}}(\vec{x}) &= s_x\vec{x} \\ E_{\mathrm{s}}(\vec{y}) &= s_y\vec{y} \\ E_{\mathrm{s}}(\vec{z}) &= s_z\vec{z} \\ E_{\mathrm{s}}(\dot{o}) &= \dot{o} \end{aligned} \qquad M_T = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Tipos de escalados

Las transformaciones de escalado pueden clasificarse en base a los factores de escala, suponiendo que se define en un marco $\mathcal{C}$ cartesiano:

- Si todos los factores de escala son todos iguales a un valor s, se dice que el escalado es **uniforme**, conserva los ángulos y las proporciones de los vectores, ya que la longitud de un vector se multiplica por s al transformar.
- Si los factores son distintos, el escalado es **no uniforme**, y no conserva los ángulos ni las proporciones.

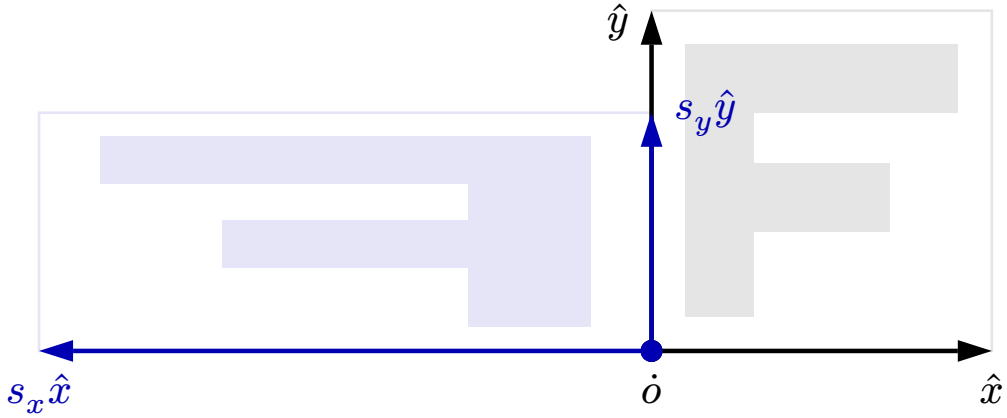
También en un marco cartesiano, podemos analizar el signo del producto de los factores (es el determinante de la matriz R_T)

- Si el producto es **positivo**, el escalado **conserva la orientación** del sistema de referencia.
- Si el producto es **negativo**, el escalado **invierte la orientación** del sistema de referencia.

Esquema de escalados en 2D

Vemos en escalado en 2D por un vector, en negro el marco original y en azul el escalado. Se usan los factores de escala $s_x = -1.8$ y $s_y = 0.7$.

El cuadrado unidad con la figura en gris se transforma en el rectángulo con la figura en azul, invirtiendo la orientación del marco (ya que el signo de $s_x s_y$ es negativo).



Reflexiones

Una **reflexión** S_i es un caso particular de escalado. Son escalados definidos en un marco cartesiano $\mathcal{C}$ que cumplen:

- Los factores de escala son todos 1, excepto uno de ellos, el correspondiente al versor $\hat{e}_i$ que es -1. Por eso, son su propia inversa, es decir $S_i^2 = I$.
- No conservan la orientación: en 2D y 3D convierten un marco a derechas en uno a izquierdas y al revés.
- Conservan las longitudes de los vectores, y la magnitud de los ángulos (no su signo). Así que son **isométricas**, pero no **rígidas**.
- La transformación invierte el signo de una de las coordenadas de cualquier punto o vector, con lo cual es una *reflexión* cuyo *eje o plano de reflexión* (una línea o plano invariante) es el la línea o el plano paralelo a $\hat{e}_i$ que pasa por el origen.
- Las distancias al eje o plano se conservan, pero el punto o vector pasa al otro lado del mismo.

Reflexiones con eje o planos arbitrarios

Una reflexión $S_{\hat{n}}$ puede hacerse teniendo como eje una línea cualquiera por el origen (en 2D) o con un plano de reflexión por el origen (en 3D). La línea o el plano no son necesariamente paralelos a los ejes de coordenadas.

La línea o el plano serán perpendiculares a un versor (unitario) $\hat{n}$ cuyas coordenadas serán $\mathbf{n} = \left(n_x, n_y, n_z\right)$ en 3D o $\mathbf{n} = \left(n_x, n_y\right)$ en 2D.

En estas condiciones, la matriz asociada es la *Matriz de Householder*:

$$M_T \;=\; I - 2(\mathbf{n}\mathbf{n}^T)$$

donde I es la matriz identidad y $\mathbf{n}\mathbf{n}^T$ es esta matriz (en 2D y 3D):

$$\mathbf{n}\mathbf{n}^T := \begin{pmatrix} n_x n_x & n_x n_y & 0 \\ n_x n_y & n_y n_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{n}\mathbf{n}^T := \begin{pmatrix} n_x n_x & n_x n_y & n_x n_z & 0 \\ n_x n_y & n_y n_y & n_y n_z & 0 \\ n_x n_z & n_y n_z & n_z n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Subsección **3.3.**

Cizallas

Cizallas.

Una **cizalla** (***shear***) C_{ij} es una transformación afín que, en un marco $\mathcal{R}$ cualquiera, incrementa la i-ésima coordenada de una tupla, usando un incremento proporcional a la j-ésima coordenada de esa tupla (con $i \neq j$).

- El valor transformado c_i' de la i-ésima coordenada será $c_i' = c_i + ac_j$, donde a es un parámetro real que define la cizalla, y c_i y c_j las coordenadas originales. Las otras coordenadas c_j (con $i \neq j$) no cambian.
- La matriz asociada es igual a la matriz identidad, excepto que el valor en la fila i y columna j es a en lugar de 0.
- El marco transformado por C_{ij} es igual al original, excepto que en el marco transformado el eje $T(\vec{e}_j)$ se hace igual a $\vec{e}_j + a\vec{e}_i$.
- Las cizallas no conservan las longitudes ni los ángulos en general. Por eso son transformaciones **no rígidas**.
- Las cizallas únicamente conservan longitudes en líneas perpendiculares a $\vec{e}_j$. En 3D conservan ángulos entre vectores perpendiculares a $\vec{e}_j$.

Matrices de las cizallas en 2D.

En 2D hay dos tipos de cizallas, las llamamos C_{01} y C_{10}. Ambas dependen del parámetro real a.

- La cizalla C_{01} incrementa la coordenada X según la Y por tanto el marco transformado y la matriz son:

$$\begin{aligned} C_{01,a}(\vec{x}) &= \vec{x} \\ C_{01,a}(\vec{y}) &= \vec{y} + a\vec{x} \\ C_{01,a}(\dot{o}) &= \dot{o} \end{aligned} \qquad M_T = \begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- La otra cizalla en 2D es C_{10}, que incrementa la coordenada Y según la X, así que ahora tenemos:

$$\begin{aligned} C_{10,a}(\vec{x}) &= \vec{x} + a\vec{y} \\ C_{10,a}(\vec{y}) &= \vec{y} \\ C_{10,a}(\dot{o}) &= \dot{o} \end{aligned} \qquad M_T = \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrices de las cizallas en 3D.

En 3D hay 6 tipos de cizallas, ya que hay 6 posibles pares i, j distintos.

Al igual que en 2D:

- La matriz es igual a la matriz identidad, excepto que en la fila i y columna j aparece el valor a en lugar de 0.
- El eje $\hat{e}_j$ se incrementa por $a\hat{e}_i$.

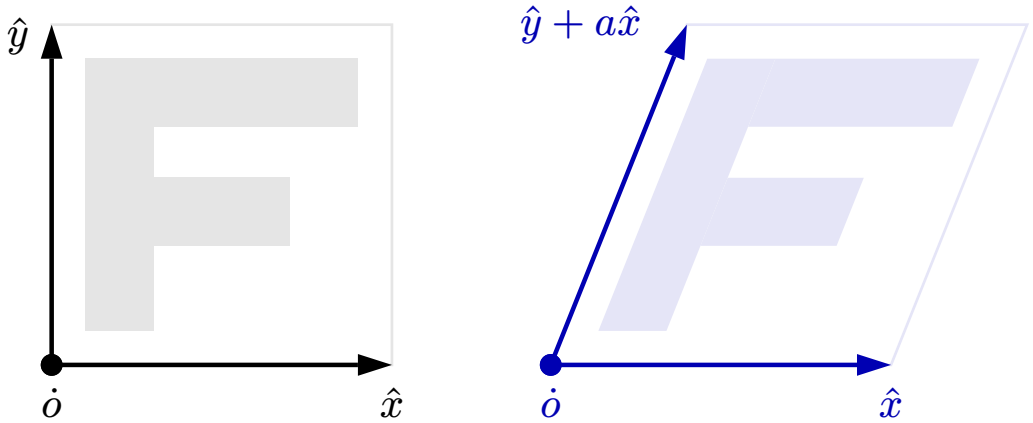
A modo de ejemplo, vemos la cizalla C_{21} que incrementa la coordenada Z ($i = 2$) en base a la Y ($j = 1$)

$$\begin{aligned} C_{21,a}(\vec{x}) &= \vec{x} \\ C_{21,a}(\vec{y}) &= \vec{y} + a\vec{z} \\ C_{21,a}(\vec{z}) &= \vec{z} \\ C_{21,a}(\dot{o}) &= \dot{o} \end{aligned} \qquad M_T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & a & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Esquema de una cizalla en 2D

Vemos la cizalla C_{01} en 2D con $a = 0.4$. Los puntos se desplazan en el eje X una distancia proporcional a su coordenada Y.

El cuadrado unidad con la figura en gris (a la izquierda) se transforma en el paralelogramo con la figura en azul a la derecha (la transformación no conlleva traslación, pero se han desplazado las figuras para apreciarlas mejor).



Subsección **3.4.**

Rotaciones

Rotaciones en 2D

Una transformación de rotación R en un marco cartesiano $\mathcal{C} = (\hat{x}, \hat{y}, \dot{o})$, es una transformación afín rígida que:

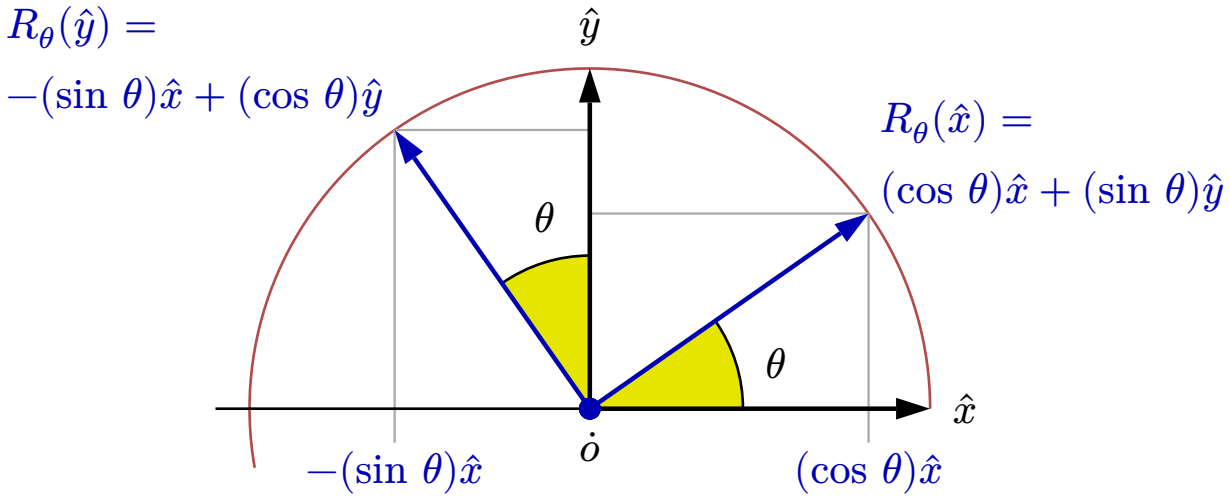
- lleva cada punto $\dot{p}$ a otro a la misma distancia de $\dot{o}$ que el original. Se dice que $\dot{o}$ es el **centro de rotación**.
- depende de un parámetro θ, llamado **ángulo de rotación**, que es el ángulo en radianes (con signo) entre $\dot{p}$ y $R(\dot{p})$. Si $\theta > 0$, la rotación es antihoraria, y si $\theta < 0$ es horaria.

Así que la transformación de los versores y el origen, y la matriz son:

$$\begin{aligned} R_\theta(\hat{x}) &= (\cos\theta)\hat{x} + (\sin\theta)\hat{y} \\ R_\theta(\hat{y}) &= -(\sin\theta)\hat{x} + (\cos\theta)\hat{y} \\ R_\theta(\dot{o}) &= \dot{o} \end{aligned} \qquad M_R = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Componentes de los ejes rotados en 2D

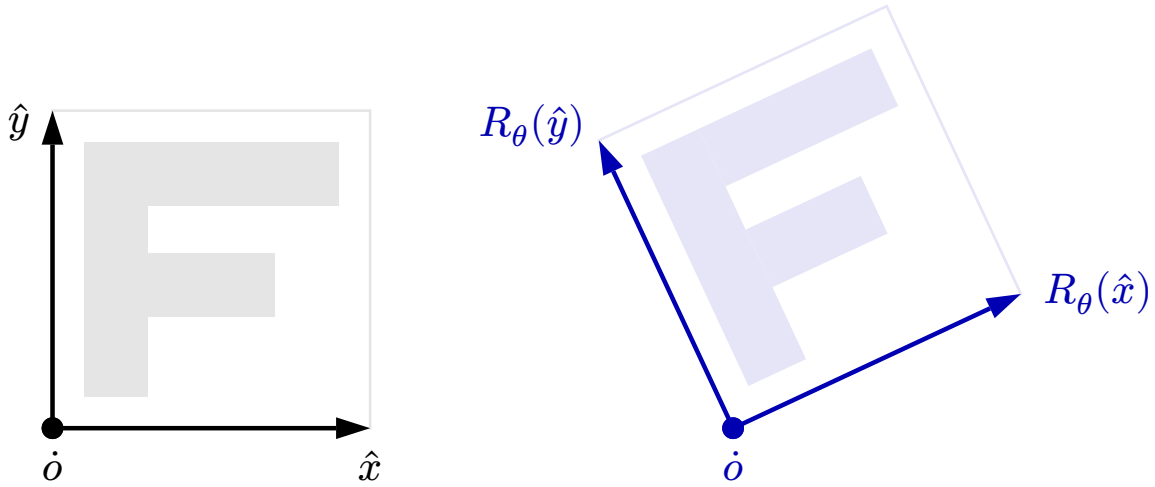
Vemos una rotación en 2D un ángulo $\theta = 35^{\circ}$, que es mayor que 0. En negro aparece un marco $\mathcal{R}$ y en azul el marco transformado. Ambos tienen el mismo origen.



Esquema de una rotación en 2D

Vemos una rotación en R_θ con $\theta = 25^\circ$. Los puntos se rotan entorno al origen.

El cuadrado de lado 1 con la figura en gris (a la izquierda) se transforma en el cuadrado con la figura en azul a la derecha (la transformación no conlleva traslación, pero se han desplazado las figuras para apreciarlas mejor).



Problemas: rotaciones 2D (1/3)

Problema 3.4:

Demuestra que el producto escalar de vectores en 2D es invariante por rotación, es decir, que para cualquier ángulo θ y vectores $\vec{u}$ y $\vec{v}$ se cumple:

$$R_\theta(\vec{u} \cdot \vec{v}) \quad = \quad R_\theta(\vec{u}) \cdot R_\theta(\vec{v})$$

(para demostrarlo usa las coordenadas de los vectores en un marco cartesiano cualquiera)

Problema 3.5:

Demuestra que en 2D las rotaciones no modifican la longitud de un vector, es decir, que para cualquier ángulo θ y vector $\vec{v}$, se cumple:

$$\| \, R_\theta(\vec{v}) \, \| \quad = \quad \| \, \vec{v} \, \|$$

Problemas: rotaciones 2D (2/3)

Problema 3.6:

Demuestra que si rotamos en 2D un vector +90 grados o −90 grados, obtenemos otro vector perpendicular al original, es decir, si $\|\theta\| = \pi/2$ entonces

$$\vec{v} \cdot R_\theta(\vec{v}) \quad = \quad 0.$$

Problema 3.7:

Demuestra que una matriz de rotación en 2D es siempre ortonormal, independientemente del ángulo. Eso quiere decir que sus filas son perpendiculares dos a dos, e igual ocurre con sus columnas, y además cada fila y columna tienen longitud 1.

Problemas: rotaciones 2D (3/3)

Problema 3.8:

Demuestra que, en 2D, el producto de una matriz de rotación y una de escalado no es conmutativo en general, excepto si el escalado es uniforme.

Problema 3.9:

Demuestra que en 2D, el producto de una matriz de rotación y otra de traslación (por un vector no nulo) no es conmutativo.

Rotaciones en 3D

Una transformación de rotación R_θ en un marco cartesiano $\mathcal{C} = (\hat{x}, \hat{y}, \hat{z}, \dot{o})$, es una transformación afín rígida que:

- lleva cada punto $\dot{p}$ a otro a la misma distancia que $\dot{p}$ de una línea que pasa por $\dot{o}$, y es paralela a un versor $\hat{e}$ (a ese versor o a esa línea se les llama **eje de rotación**).
- también depende de un **ángulo de rotación** θ en radianes entre $\dot{p}$ y $R(\dot{p})$.
- cuando $\theta > 0$, la rotación es antihoraria, vista desde un punto situado en el eje de rotación y mirando hacia el origen. Si $\theta < 0$, la rotación es horaria.
- el eje puede ser paralelo a uno de los tres versores de $\mathcal{C}$, o puede ser otro cualquiera.

Rotaciones en 3D entorno a los ejes cartesianos. Eje Z

En este caso el eje de rotación $\hat{e}$ es alguno de los versores del marco $\mathcal{C}$, es decir es $\hat{x}$, $\hat{y}$ o $\hat{z}$. Las coordenadas en el eje $\hat{e}$ no cambian, ya que las rotaciones ocurren en un plano perpendicular a $\hat{e}$. Las otras dos coordenadas cambian igual que en 2D.

Nombramos las rotaciones indicando ángulo y eje. Para una rotación entorno al eje Z tenemos las mismas expresiones que en 2D para X e Y:

$$\begin{aligned} R_{z,\theta}(\hat{x}) &= (\cos\theta)\hat{x} + (\sin\theta)\hat{y} \\ R_{z,\theta}(\hat{y}) &= -(\sin\theta)\hat{x} + (\cos\theta)\hat{y} \\ R_{z,\theta}(\hat{z}) &= \hat{z} \\ R_{z,\theta}(\dot{o}) &= \dot{o} \end{aligned}$$

$$M_R = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotaciones en 3D entorno a los ejes cartesianos. Ejes Y y X

Para rotaciones entorno al eje Y:

$$\begin{aligned} R_{\mathrm{y},\theta}(\hat{x}) &= (\cos\theta)\hat{x} - (\sin\theta)\hat{z} \\ R_{\mathrm{y},\theta}(\hat{y}) &= \hat{y} \\ R_{\mathrm{y},\theta}(\hat{z}) &= (\sin\theta)\hat{x} + (\cos\theta)\hat{z} \\ R_{\mathrm{y},\theta}(\dot{o}) &= \dot{o} \end{aligned}$$

$$M_R = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Para rotaciones entorno al eje X:

$$\begin{aligned} R_{\mathrm{x},\theta}(\hat{x}) &= \hat{x} \\ R_{\mathrm{x},\theta}(\hat{y}) &= (\cos\theta)\hat{x} + (\sin\theta)\hat{y} \\ R_{\mathrm{x},\theta}(\hat{z}) &= -(\sin\theta)\hat{x} + (\cos\theta)\hat{y} \\ R_{\mathrm{x},\theta}(\dot{o}) &= \dot{o} \end{aligned}$$

$$M_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Problemas: rotaciones 3D entorno a los ejes cartesianos(1/n)

Considera las rotaciones en 3D entorno a $\hat{e}$, que es uno de los ejes $\hat{x}$, $\hat{y}$ y $\hat{z}$ de un **marco cartesiano 3D**.

Problema 3.10:

Demuestra que el producto escalar de vectores en 3D es invariante por estas rotaciones. y que tampoco modifican la longitud de un vector. Te puedes basar en los problemas similares en 2D.

Problema 3.11:

Demuestra que el producto vectorial de dos vectores rota igual que lo hacen esos dos vectores, es decir, que para cualquiera dos vectores $\vec{u}$ y $\vec{v}$ y un ángulo θ, se cumple:

$$R_{\theta,\hat{e}}(\vec{u} \times \vec{v}) \quad = \quad R_{\theta,\hat{e}}(\vec{u}) \times R_{\theta,\hat{e}}(\vec{v})$$

Rotaciones en 3D de ejes arbitrarios

En Informática Gráfica se puede necesitar hacer rotaciones entorno a ejes que son distintos de los ejes cartesianos.

Suponiendo que el eje es un versor $\hat{e}$ (unitario), entonces se puede escribir el vector rotado usando la llamada **fórmula de Rodrigues**:

$$R_{\hat{e},\theta}(\vec{v}) \quad = \quad (\cos\theta)\vec{v} \quad + \quad (1-\cos\theta)(\vec{v}\cdot\hat{e})\hat{e} \quad + \quad (\sin\theta)\hat{e}\times\vec{v}$$

Si las coordenadas de $\hat{e}$ en $\mathcal{C}$ son $(e_0, e_1, e_2, 0)$, podemos escribir la matriz M_R como una suma:

$$M_R = \begin{pmatrix} a_{00} & a_{01} & a_{02} & 0 \\ a_{10} & a_{11} & a_{12} & 0 \\ a_{20} & a_{21} & a_{22} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} c & -se_2 & se_1 & 0 \\ se_2 & c & -se_0 & 0 \\ -se_1 & se_0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ con } \begin{cases} a_{ij} := (1-c)e_ie_j \\ c := \cos\theta \\ s := \sin\theta \end{cases}$$

Los ángulos de Euler para rotaciones en 3D

Otra forma alternativa de definir una rotación 3D de eje arbitrario es usando una tupla de tres valores reales (α, β, γ) llamados **ángulos de Euler**, cada uno de ellos en radianes.

Una rotación de eje arbitrario $R_{\hat{e},\theta}$ será equivalente a la composición de tres rotaciones entorno cada uno de los tres ejes $\hat{x}, \hat{y}, \hat{z}$ de un marco cartesiano, usando los ángulos de Euler:

$$R_{\hat{e},\theta} \quad = \quad R_{\mathrm{z},\gamma} \circ R_{\mathrm{y},\beta} \circ R_{\mathrm{x},\alpha}$$

Usar esto puede ser útil si conocemos los tres ángulos y queremos obtener la matriz (independientemente de $\hat{e}$ y θ, que no conoceremos), un ejemplo son las *cámaras orbitales*. Sin embargo:

- Si partimos del eje $\hat{e}$ y el ángulo θ no es sencillo calcular los ángulos de Euler que producen la misma rotación.
- Fijado $\hat{e}$ y θ, puede haber más de una tripleta de ángulos de Euler que den lugar a la misma rotación.

Cuaterniones para rotaciones en 3D

Otra alternativa es usar los **cuaterniones** (***quaternions***) que son objetos que se pueden ver como una extensión de los números complejos a 4 dimensiones. Un cuaternión q tiene la forma:

$$q = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$$

donde a, b, c y d son reales, y $\mathbf{i}, \mathbf{j}$ y $\mathbf{k}$ son tres unidades imaginarias distintas. Al igual que los complejos, los cuaterniones se pueden sumar, restar y multiplicar entre ellos. Las propiedades de los cuaterniones permiten:

- Representar rotaciones en 3D de eje y ángulo arbitrarios con un cuaternión (es decir, con una tupla de 4 reales en memoria).
- Multiplicar cuaterniones para componer rotaciones.
- Representar cualquier vector en 3D como un cuaternión.
- **Rotar un vector alrededor de un eje arbitrario de forma muy eficiente**, usando los dos cuaterniones que representan la rotación y el vector y obteniendo el cuaternión del vector rotado.

Rotaciones de ejes arbitrarios: alternativas

Si una rotación debe ser aplicada a muchos vectores en la GPU, entonces es mejor calcular una vez la matriz M_R y usarla para aplicarla a cada vector. Se puede hacer de dos formas:

- Calculando la matriz M_T indicada antes.
- Componiendo: (1) una matriz de rotación R que alinea $\hat{e}$ con $\hat{x}$, (2) la rotación por θ entorno a $\hat{x}$ y finalmente (3) la rotación inversa R^{-1} de la primera. Es más complejo que calcular la matriz directamente.

Para rotar pocos vectores en la CPU puede ser más eficiente no calcular la matriz, se puede hacer de dos formas:

- Usando la fórmula de Rodrigues directamente.
- Usando *cuaterniones*.

La fórmula de Rodrigues y los cuaterniones son equivalentes, y para pocos vectores ambas formas son mucho más eficientes que calcular la matriz. Los cuaterniones, sin embargo, son un poco más eficientes que la fórmula de Rodrigues.

Sección **4.**

Transformaciones en Godot.

1. Tuplas de valores reales.
2. Matrices de transformación.

Introducción

En esta sección estudiaremos como se representan las tuplas de coordenadas homogéneas en 2D y 3D en Godot (clases `Vector2` y `Vector3`), y las matrices de transformación (clases `Transform2D` y `Transform3D`).

También veremos como se implementan las transformaciones más comunes: traslaciones, rotaciones y escalados, mediante el atributo `transform` de las clases `Node2D` y `Node3D`.

Veremos como la transformación de un nodo en el árbol de escena afecta a ese nodo y a todos sus hijos.

Subsección **4.1.**

Tuplas de valores reales.

Tuplas 2D. Clase *Vector2*.

Un objeto de la clase `Vector2` representa una tupla de dos valores reales, que se puede usar para guardar coordenadas de puntos o vectores en 2D.

- Contienen dos valores reales representados en coma flotante con 32 bits de precisión (equivalente a un `float` de C/C++, pero no a un `float` de GDscript, que tiene 64 bits).
- Los elementos son accesibles como `v.x` y `v.y`, o como `v[`e`]` donde e es una expresión entera que se evalúa a 0 o 1.
- Estos objetos se pueden sumar, restar y multiplicar por un escalar (un `float` de GDscript), usando los operadores binarios infijos `+`, `-` y `*`.
- También se pueden usar diversos métodos para otras operacioes:
  - $\mathbf{v}$`.length()` devuelve la longitud del vector $\mathbf{v}$ (un `float`)
  - $\mathbf{v}$`.normalized()` devuelve el vector $\mathbf{v}/\|\mathbf{v}\|$ (`Vector2`), si $\|\mathbf{v}\| > 0$.
  - $\mathbf{v}$`.dot(`$\mathbf{u}$`)` devuelve el producto escalar de $\mathbf{v} \cdot \mathbf{u}$ (un `float`).

Hay otros muchos métodos, consultar:
docs.godotengine.org/en/stable/classes/class_vector2.html

Tuplas 3D. Clase *Vector3*.

Un objeto de la clase `Vector3` representa una tupla de tres valores reales, que se puede usar para guardar coordenadas de puntos o vectores en 3D.

- Contienen tres valores reales representados en coma flotante con 32 bits de precisión
- Los elementos son accesibles como `v.x`, `v.y`, `v.z`, o como `v[`e`]` donde e es una expresión entera que se evalúa a 0, 1 o 2.
- Estos objetos se pueden sumar, restar y multiplicar por un escalar (un `float` de GDscript), usando los operadores binarios infijos `+`, `-` y `*`.
- También se pueden usar diversos métodos para otras operaciones:
  - $\mathbf{v}$`.length()` devuelve la longitud del $\mathbf{v}$ (`float`)
  - $\mathbf{v}$`.normalized()` devuelve $\mathbf{v}/\ \|\mathbf{v}\|$ (`Vector3`), si $\|\mathbf{u}\| > 0$.
  - $\mathbf{v}$`.dot(`$\mathbf{u}$`)` devuelve el producto escalar de $\mathbf{v} \cdot \mathbf{u}$ (`float`).
  - $\mathbf{v}$`.cross(`$\mathbf{u}$`)` devuelve el producto vectorial $\mathbf{v} \times \mathbf{u}$ (otro `Vector3`).

Consultar: docs.godotengine.org/en/stable/classes/class_vector3.html

Otros clases relacionadas

Existen otras clases para tuplas:

- Las clases `Vector2i` y `Vector3i` representan tuplas de dos y tres valores enteros (32 bits con signo). Una tupla `Vector3i` se puede usar, por ejemplo, para guardar los tres índices de un triángulo en una malla.
- La clase `Vector4` representa tuplas de cuatro valores reales (32 bits con signo). Se puede usar, por ejemplo, para representar un cuaternión.

Así como otros tipos de clases:

- `Basis`: bases del espacio vectorial en 3D.
- `Line2D`: líneas en 2D.
- `Plane`: planos en 3D.

Ahora veremos las clases para matrices de transformación:

- `Transform2D`: matrices 3x3 en 2D.
- `Transform3D`: matrices 4x4 en 3D.

Subsección **4.2.**

Matrices de transformación.

Matrices de transformación 2D. Clase *Transform2D*

Un objeto de la clase `Transform2D` representa una matriz 3×3 que implementa una transformación afín en el espacio afín 2D A_2.

- Se guardan seis valores reales de 32 bits, las dos primeras filas de la misma. La tercera fila no se guarda y siempre es $(0, 0, 1)$.
- Se pueden multiplicar (componer) entre ellas con el operadores binarios infijo `*`.
- Se pueden multiplicar o dividir por un un `float` con el operador `*` o `/`.
- Se pueden aplicar a un `Vector2` con el operador `*` por la derecha, devuelve el `Vector2` resultado de aplicar la transformación (añadiendo $w = 1$ al `Vector2`, es decir, considerando el `Vector2` como las coordenadas de un punto).

Consultar: docs.godotengine.org/en/stable/classes/class_transform2d.html

Creación de objetos *Transform2D*. Propiedades

Hay varios métodos para crear objetos `Transform2D`. Supongamos que θ es un ángulo en radianes, a un real, $\mathbf{x}$, $\mathbf{y}$, $\mathbf{s}$ son `Vector2` (tuplas que representan vectores), y $\mathbf{o}$, $\mathbf{p}$ son `Vector2` que representan puntos. Entonces:

- `Transform2D()` crea la matriz identidad.
- `Transform2D(`θ`,`$\mathbf{o}$`)` crea la matriz que rota θ radianes entorno a $\mathbf{o}$.
- `Transform2D(`$\mathbf{x}$`,`$\mathbf{y}$`,`$\mathbf{o}$`)` crea una matriz con $\mathbf{x}$, $\mathbf{y}$ y $\mathbf{o}$ como 1a, 2a y 3a columnas, es decir, damos la base y el origen del marco transformado. Por tanto esto **permite construir cualquier matriz de transformación**.
- `Transform2D(`θ`,`$\mathbf{s}$`,`a`,`$\mathbf{o}$`)` crea la matriz componiendo una rotación por θ, un escalado por $\mathbf{s}$, una cizalla por a y una traslación por $\mathbf{o}$.

Se pueden usar las *propiedades* `x`, `y` y `origin` para consultar o actualizar los objetos `Vector2` con los valores en la 1a, 2a y 3a columnas, respectivamente. Permite leer o modificar directamente cualquier real de la matriz.

Métodos para componer una matriz 2D con otra

La clase `Transform2D` tiene varios métodos para crear una matriz B y componerla con otra matriz existente A, sin modificar A.

Hay varios métodos que devuelven BA (componen B por la **izquierda**: la acción es A seguida de B). Son:

- A`.rotated(`θ`)`: $B =$ rotación de θ radiantes entorno al origen.
- A`.scaled(`$\mathbf{s}$`)`: $B =$ matriz de escalado con factores $\mathbf{s}$.
- A`.translated(`$\boldsymbol{t}$`)`: $B =$ traslación por el vector $\boldsymbol{t}$.

Muchas veces querremos componer las matrices creadas por la **derecha** , es decir obtener AB (acción de B seguida de A). Los correspondientes métodos son:

- A`.rotated_local(`θ`)`
- A`.scaled_local(`$\mathbf{s}$`)`
- A`.translated_local(`$\boldsymbol{t}$`)`

Transformaciones en 3D. La clase *Transform3D*

La clase `Transform3D` es similar a la clase `Transform2D`, solo que se guardan 4 columnas de 3 valores reales cada una (12 valores reales en total)

Los constructores, métodos y propiedades son similares, solo que se usan tuplas de tipo `Vector3` en lugar de `Vector2`. Destacamos estos dos constructores:

- `Transform3D()`: crea la matriz identidad.
- `Transform3D(x,y,z,o)`: crea una matriz cuyas columnas son $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ y $\mathbf{o}$ (tipo `Vector3` todos), es decir, damos la base y el origen del marco transformado.

Al igual que en 2D:

- estas matrices se pueden multiplicar entre ellas con `*`, y
- se pueden aplicar a un `Vector3` con `*`, por la derecha (considerando el `Vector3` como las coordenadas de un punto, es decir, añadiendo $w = 1$).

Métodos para componer una matriz 3D con otra

La clase `Transform3D` tiene varios métodos para crear una matriz B y componerla con otra matriz existente A, sin modificar A.

Varios métodos que devuelven BA (componen B por la **izquierda**: la acción es A seguida de B). Son:

- A`.rotated(`$\mathbf{e}$`,`θ`)`: $B =$ rotación de θ radiantes entorno al eje $\mathbf{e}$ (un `Vector3` que debe de tener longitud unidad).
- A`.scaled(`$\mathbf{s}$`)`: $B =$ matriz de escalado con factores $\mathbf{s}$ (es un `Vector3`)
- A`.translated(`$\boldsymbol{t}$`)`: $B =$ traslación por el vector $\boldsymbol{t}$ (`Vector3`)

Muchas veces querremos componer las matrices creadas por la **derecha** , es decir obtener AB (acción de B seguida de A). Los correspondientes métodos son:

- A`.rotated_local(`$\mathbf{e}$`,`θ`)`
- A`.scaled_local(`$\mathbf{s}$`)`
- A`.translated_local(`$\boldsymbol{t}$`)`

La transformación de un nodo

Cada nodo de tipo `Node2D` o `Node3D` tiene una propiedad llamada `transform` que es un objeto de la clase `Transform2D` o `Transform3D`, respectivamente. Inicialmente es la matriz identidad.

- En los scripts podemos asignar valores a esa propiedad (en la función `_ready()` o `_init`), con lo cual podemos cambiar la matriz de transformación que se aplica los vértices de ese nodo en tiempo de ejecución.
- En el editor, se pueden cambiar los valores iniciales de `transform` (posición, rotación y escalado, entre otras) en el inspector de propiedades del nodo.

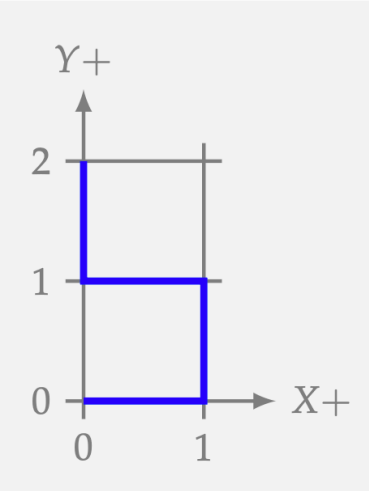
También existen métodos en esas clases que permiten modificar la matriz de transformación de un nodo, sin necesidad de construir una nueva matriz y asignarla a `transform`. Estos métodos se explican más adelante.

Problemas: implementación de transformaciones 2D en Godot

Para este problema y el siguiente, usa el mismo proyecto de Godot para visualización 2D que ya has usado en problemas previos.

Problema 3.12:

Crea un script global (*autoload*) con una función llamada `gancho` (sin parámetros) que crea y devuelve un objeto de la clase `Mesh` con una polilínea azul como la de la figura (los ejes se han dibujado por claridad).

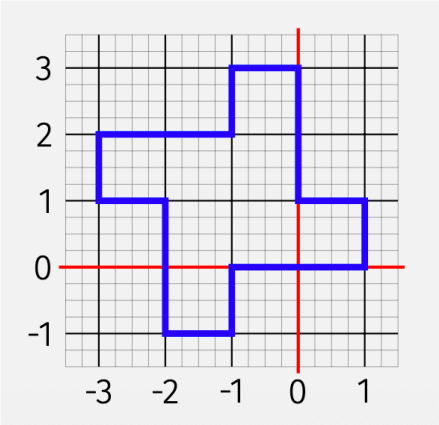


Crea en tu proyecto un nodo 2D de tipo `MeshInstance2D` y en `_ready` asígnale como malla (propiedad `mesh`) el objeto resultado de llamar a `gancho()`, ponle un color azul (propiedad `modulate`) y verifica que el gancho aparece en pantalla al ejecutar el proyecto.

Problemas: implementación de transformaciones 2D en Godot

Problema 3.13:

Crea un nodo 2D de tipo `Node2D` y llámalo `Gancho_x4`. En `_ready`, añádele cuatro nodos hijos de tipo `MeshInstance2D`, cada uno de ellos con un malla creada con la función `gancho` del problema anterior, pero con su `transform` modificada para que el objeto `Gancho_x4` se vea como en la figura (la rejilla y los ejes en rojo se han dibujado por claridad).



Fin de transparencias.