Tema 2 - Arquitectura y modelos para entornos virtuales.

2.2 Métodos básicos de representación.

Germán Arroyo, Juan Carlos Torres

5 de febrero de 2021

Contenido del tema

Tema 2: Arquitectura y modelos para entornos virtuales.

2.1 Grafos de escena y modelos jerárquicos.

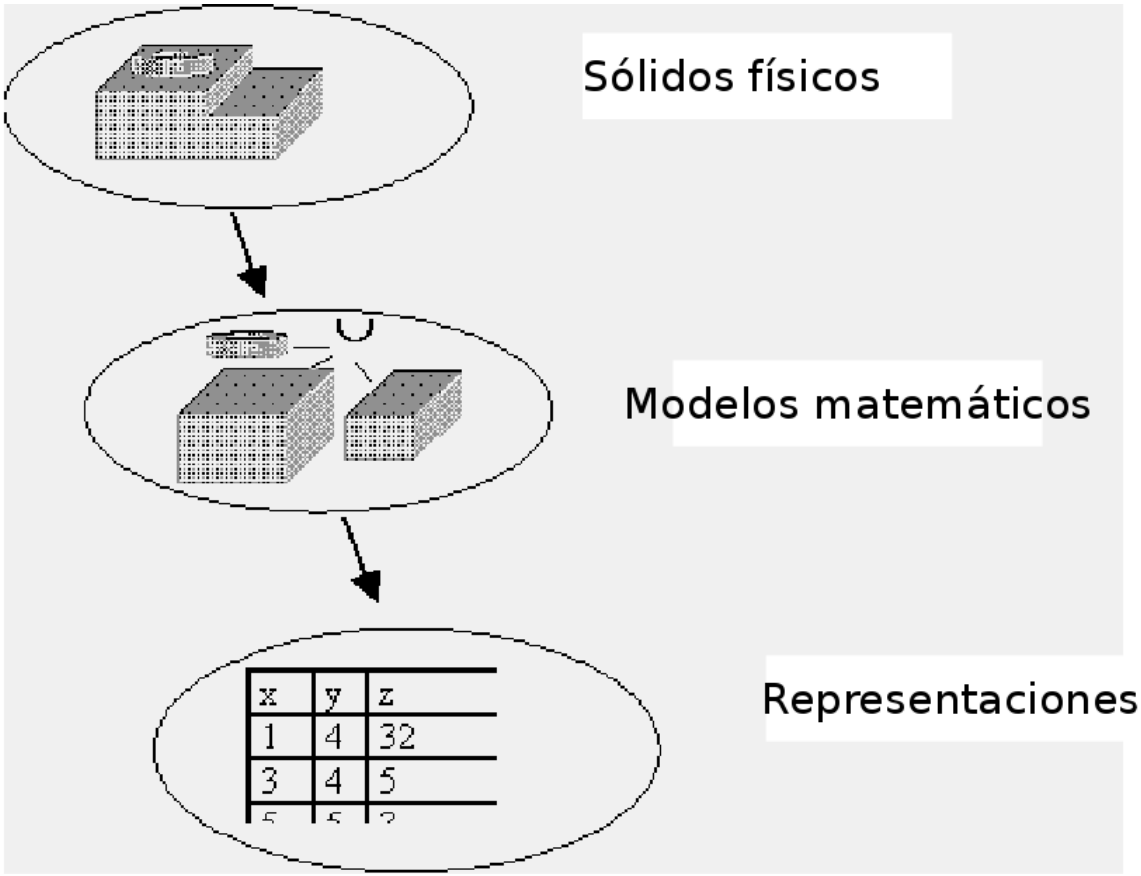
2.2 Métodos básicos de representación.

2.3 Sistemas básicos de iluminación y cámaras.

2.4 Modelos de generación procesal.

2.1 Métodos básicos de representación

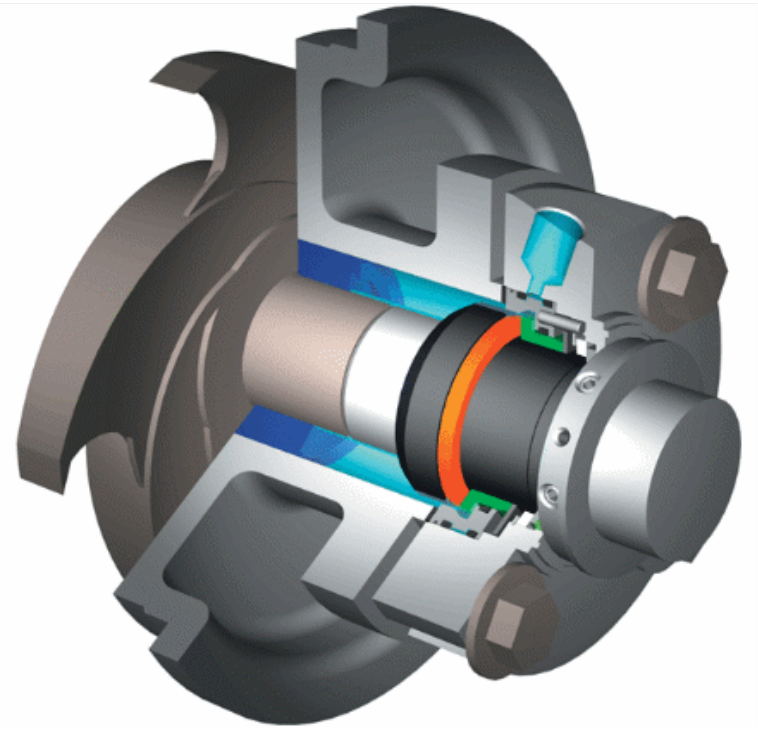
Modelos de malla que alguna vez se tratan como sólidos.



Sólidos y mallas están relacionados. ¡Pero no son lo mismo!

Sólidos

Con un sólido se pueden calcular propiedades que no se pueden calcular en una malla.



Cara	Vértices					
1	(0,0,0)	(0,2,-1)	(0,3,2)	(0,1,2)	NULL	
2	(0,2,-1)	(2,3,.-1)	(2,4,1)	(0,3,2)	NULL	

Visualizar

Editar

Realizar cálculos y simulaciones

Sólidos: Modelo topológico

- Definiciones:
- **Sólido:** es un subconjunto cerrado y acotado de $\mathbb{E}^3$.
- **Rígido:** dos sólidos que se diferencian tan solo en una transformación rígida (sin deformaciones) son el mismo sólido.
- **Homogeneidad tridimensional:** En el espacio 3D solamente puede haber transformaciones homogéneas (dadas por matrices 4x4).

Transformaciones homogéneas (I)

- Dada una matriz H que representa una transformación homogénea, y un vector u:
  - $H = \begin{pmatrix} a_x & b_x & c_x & p_x \\ a_y & b_y & c_y & p_y \\ a_z & b_z & c_z & p_z \\ d_1 & d_2 & d_3 & 1 \end{pmatrix}; u = \begin{pmatrix} u_x \\ u_y \\ u_z \\ 1 \end{pmatrix}$
- La transformación homogénea de u, es representada por v:
  - $v = H \cdot u$
- Es común trasponer las operaciones para calcularlas en el computador, lo que invierte el orden de cómputo:
  - $v^T = u^T \cdot H^T$

Transformaciones homogéneas (II)

- El vector $\vec{p} = (p_x, p_y, p_z, 1)$ representa una traslación.
  - $T = \begin{pmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}; u = \begin{pmatrix} u_x \\ u_y \\ u_z \\ 1 \end{pmatrix}$

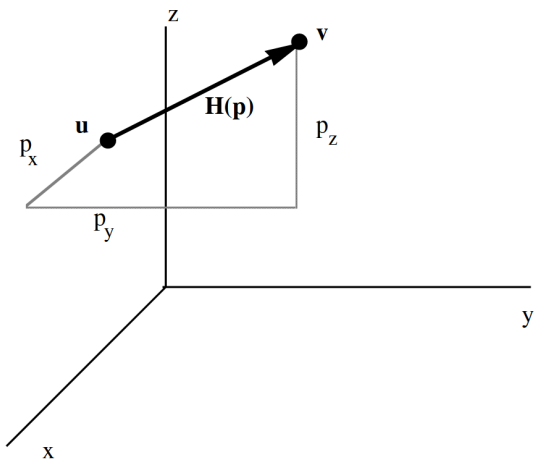


Figura 1: Traslación del punto u a v.

Transformaciones homogéneas (III)

- El vector $\vec{s} = (s_x, s_y, s_z, 1)$ representa un escalado en los tres ejes ortogonales.
  - $S = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; u = \begin{pmatrix} u_x \\ u_y \\ u_z \\ 1 \end{pmatrix}$

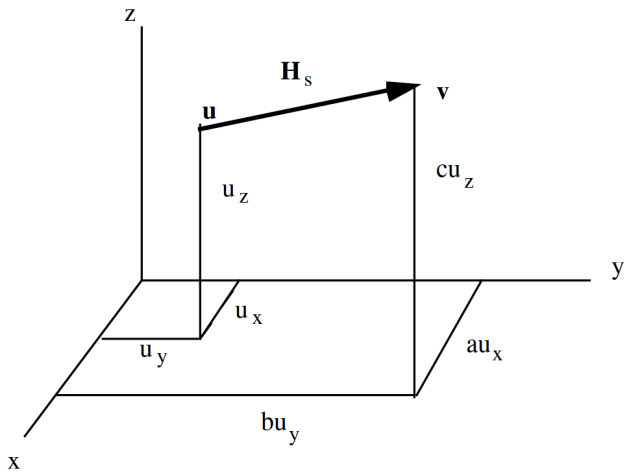
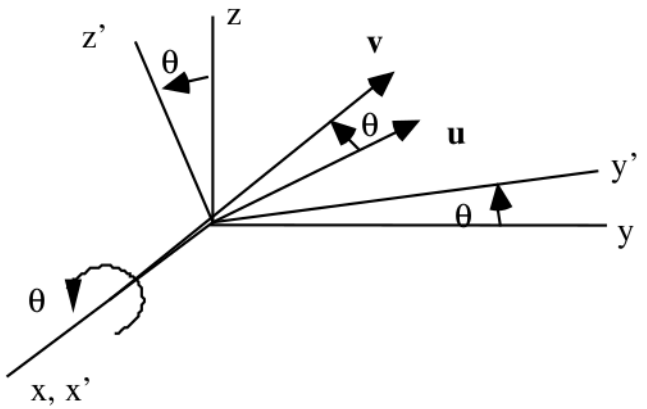


Figura 2: Escalado u a v.

Transformaciones homogéneas (VI)

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta_\alpha & -\sin\Theta_\alpha & 0 \\ 0 & \sin\Theta_\alpha & \cos\Theta_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y = \begin{pmatrix} \cos\Theta_\beta & 0 & \sin\Theta_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta_\beta & 0 & \cos\Theta_\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos\Theta_\gamma & 0 & -\sin\Theta_\gamma & 0 \\ 0 & 1 & 0 & 0 \\ \sin\Theta_\gamma & 0 & \cos\Theta_\gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Transformaciones homogéneas (V)

- ¡Cuidado! Las operaciones no son conmutativas, ¡tampoco las rotaciones!
  - $R_y(\pi/2) \cdot R_z(\pi/2) \neq R_z(\pi/2) \cdot R_y(\pi/2)$
- Se puede calcular la transformación inversa de cualquiera de estas matrices homogéneas (4x4):

> Si: $v = R \cdot u, R^-1 \cdot v = R^-1 \cdot R \cdot u = I \cdot u = u$, **Tenemos que:** $u = R^-1 \cdot v$

  - En el caso de una translación y rotación, basta con invertir el signo de las magnitudes/ángulos: $t_{inv} = -t$, $\Theta_{inv} = -\Theta$.
  - En el caso del escalado, hay que invertir las cantidades: $s_{inv} = s^{-1}$.

Problema con sólidos

Aunque podamos pensar que la operación entre dos elementos sólidos devuelve siempre un sólido esto no es siempre así.

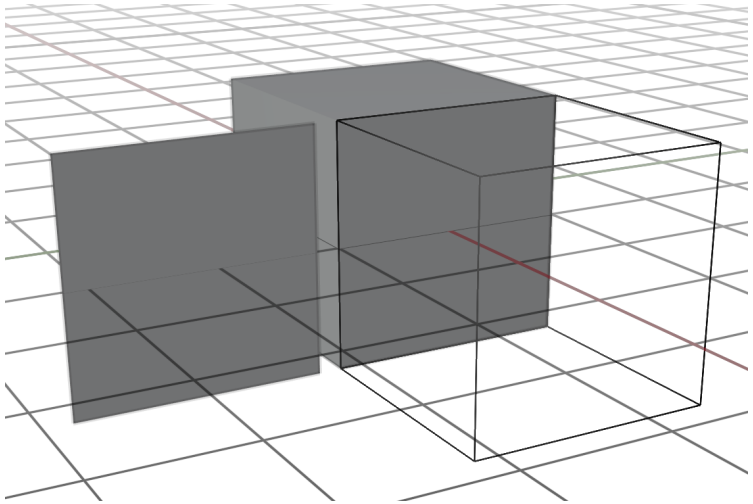


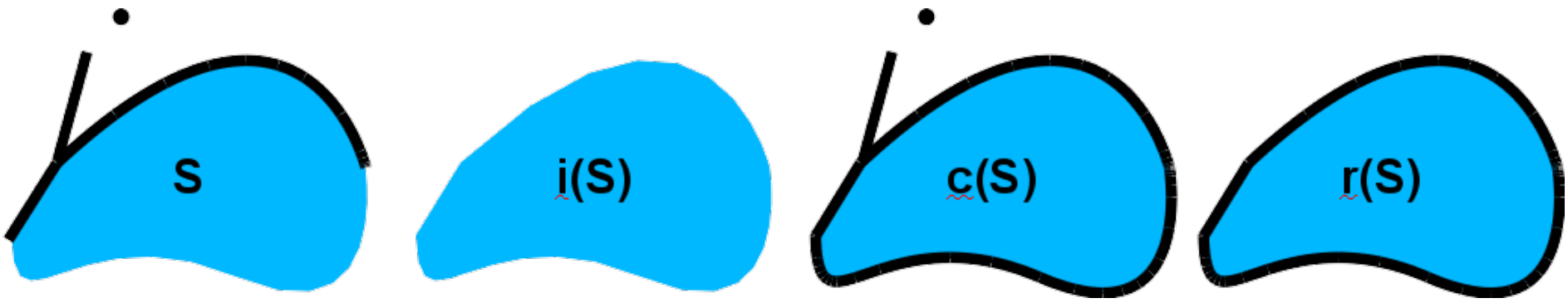
Figura 3: Ejemplo de resta entre de dos cubos.

¡Necesitamos un marco teórico para definir un sólido y evitar estos problemas!

Regularización del sólido (I)

La idea para regularizar el sólido es simple:

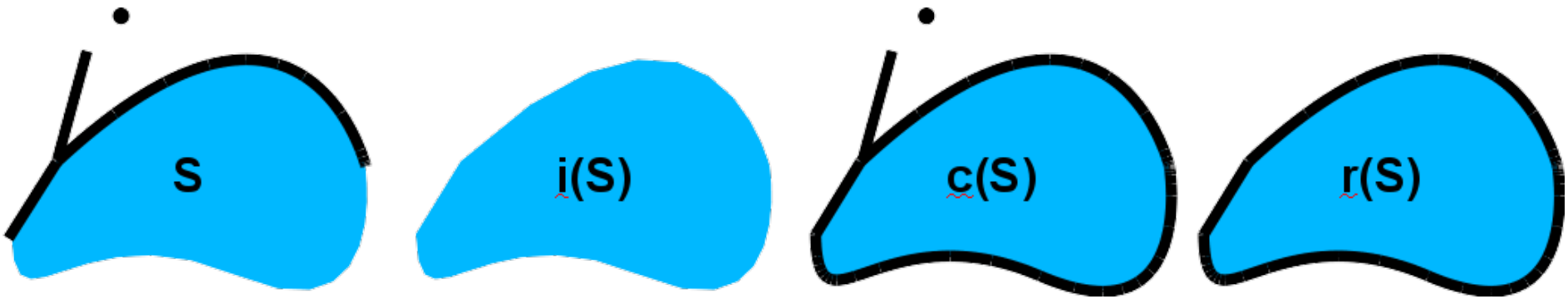
1. Calcular el resultado como siempre y bajar de dimensión una vez los componentes han sido generados.
2. Calcular el interior del resultado. El resultado es un sólido sin bordes (en azul).
3. Calcular la clausura del resultado anterior, lo que añade los bordes (en negro).



Regularización del sólido (II)

Formalmente:

- La regularización de un conjunto se define como la clausura (c) de su interior (i): $r(S) = c(i(S))$
- Un **conjunto** es **regular** si es igual a su regularización: $S = r(S)$



Operaciones de sólido

Las operaciones booleanas de sólidos se definen en base a la clausura (c) del interior (i):

- **Unión:** $A \oplus B = c(i(A \cup B))$
- **Diferencia:** $A \ominus B = c(i(A - B))$
- **Intersección:** $A \odot B = c(i(A \cap B))$

$\cup$, $-$ y $\cap$ son operadores en el conjunto.

El resultado de la resta de los cubos anteriores sería, por tanto, vacía.

Propiedades del sólido

Propiedades del sólido:

- Cerrado.
- Orientable.

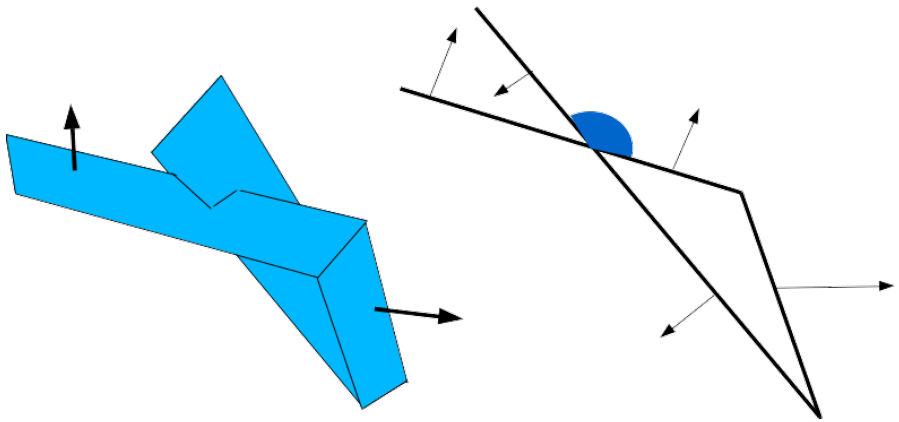


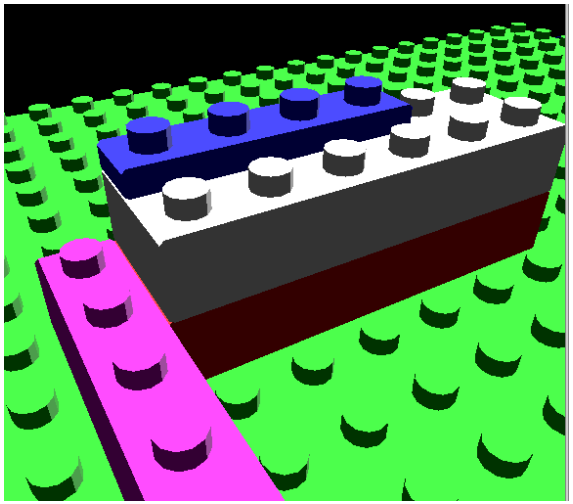
Figura 4: Podemos saber la orientación de forma automática, mirando hacia el interior del sólido.

Construcción de un sólido

Un sólido puede construirse de varias formas (difíciles de construir):

- Instanciación de primitivas (ej. cubos).
- Descomposición espacial (ej. celdas, octrees, …).
- Geometría constructiva de sólidos (CSG).
- Barrido (ej. solevados, …).
- Fronteras (Brep).

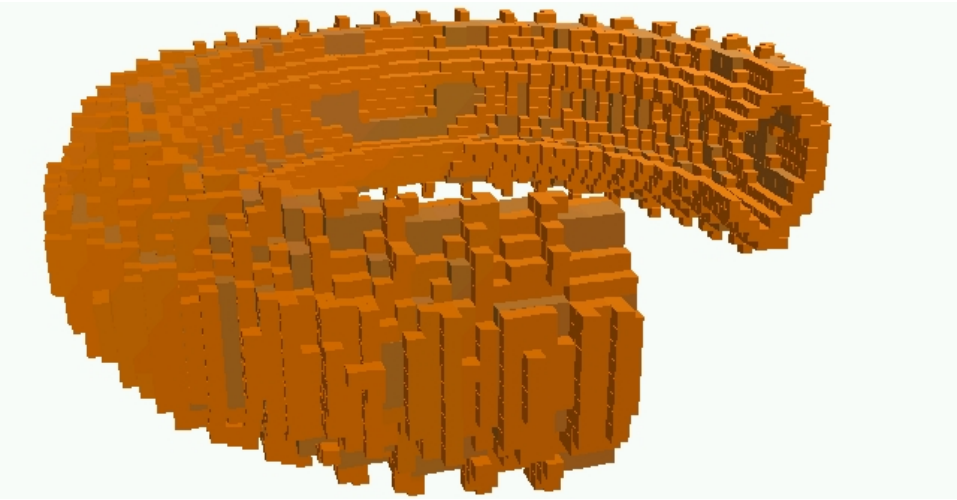
Instanciación de primitivas



El objeto se representa como un conjunto de primitivas que se instancian en el escenario:

- Las primitivas se pueden intersectar.
- No se pueden calcular propiedades.

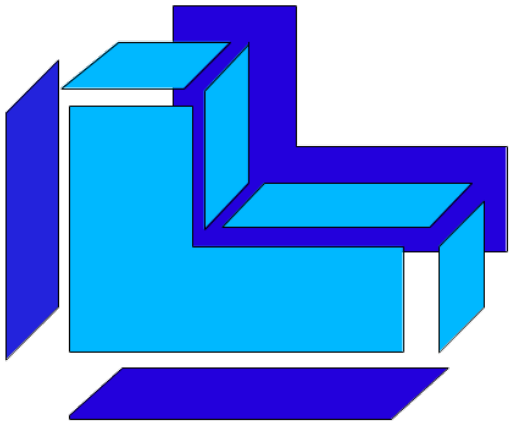
Descomposición espacial



El sólido se describe en base a una descomposición del espacio en una colección de celdas (jerárquica o no):

- Las celdas son elementos simples disjuntos.
- El forma mas común de celda es la cúbica.
- La representación de un sólido es la enumeración de las celdas que ocupa.

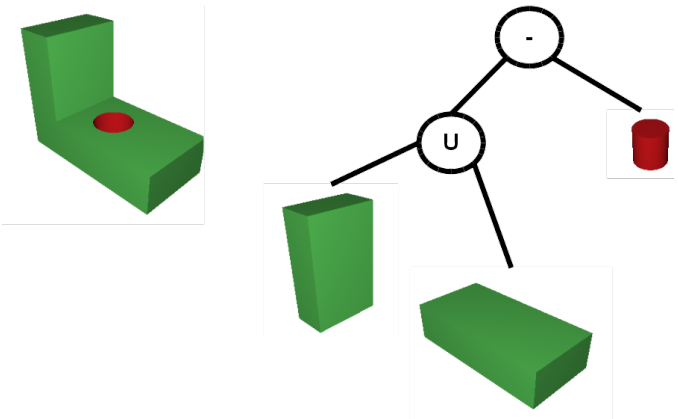
Representación de fronteras B-rep



El objeto se representa mediante un conjunto de caras que describen su frontera:

- Cuando se representa un sólido la frontera debe ser cerrada y orientable.
- La mayor parte de los sistemas utilizan representaciones de fronteras, con caras poligonales.

Geometría Constructiva de Sólidos (CSG)



El sólido se representa como una expresión booleana que indica como construirloo a partir de primitivas simples:

- La representación es la propia expresión.
- Las primitivas son sólidos parámetricos.
- La representación no es única.

Muy usada en motores de realidad virtual y juegos por su versatilidad.

Mallas de polígonos

Las mallas de polígonos se basan en tres primitivas: vértice (V), arista (E) y cara (F).

La **malla** es una terna (V, E, F) donde:

- $V = \{p_i \in \mathbb{E}^3 \mid 1 \leq i \leq n\}$
- $E = \{(i, j) \in V \times V\}$
- $F = \{(i_1, i_2, ..., i_n) \mid i_k \in V, (i_k, i_k + 1) \in E\}$

Donde:

- Cada arista E pertenece al menos a una cara F.
- Cada vértice V pertenece al menos a una arista E.

Mallas de polígonos

- Una arista es frontera si pertenece a una sola cara.
- Una malla es manifold (variedad) si:
  - Cada arista pertenece a lo sumo a dos caras
  - Para cada vértice sus polígonos adyacentes forman un disco

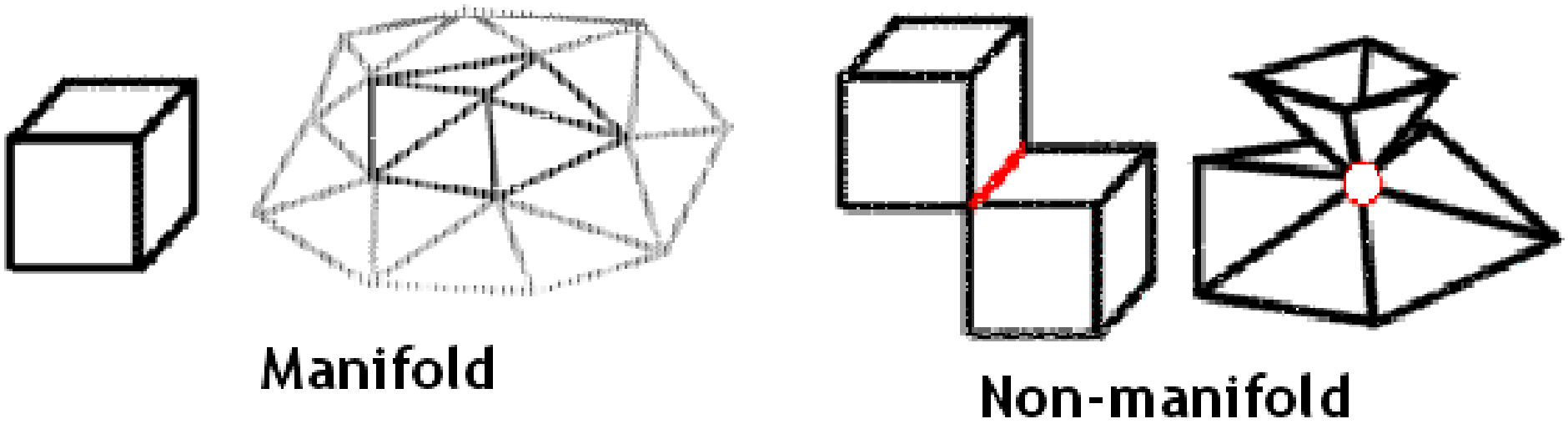


Figura 5: Diferencias entre mallas manifold y no-manifold.

Poliedros

Una malla es un **poliedro** si:

- Es manifold.
- Es cerrado.
- No hay intersecciones entre caras (salvo en aristas y vértices compartidos).

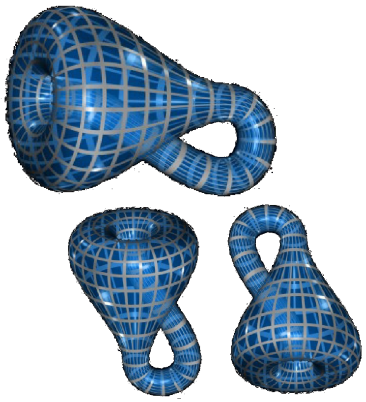
A cada cara se le puede asignar una **orientación**:

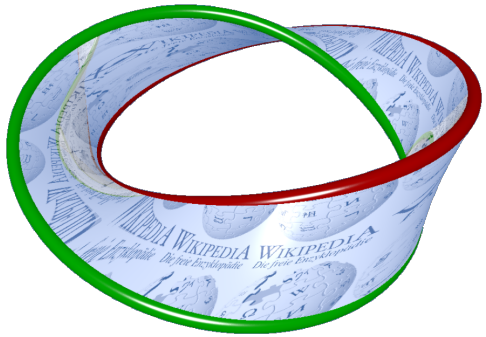
- La orientación se determina por el recorrido de sus vértices.
- Puede ser horario (CW) o antihorario (CCW).
- El lado exterior (o frontal) de la cara se determina por convenio (suele ser CCW).
  - Aunque la orientación de una cara se puede forzar.

Relación entre mallas y sólidos

Una malla es **orientable** si todas sus caras se pueden orientar de forma consistente.

- Todas con orientación CW o CCW.
- Cada arista se recorre en sentido contrario en sus dos caras.





Una **malla cerrada orientable** representa un sólido.

Mallas: estructuras de programación (I)

```
var vertex_array = [] # Vértices
var index_array = [] # Caras
```

```
vertex_array[0] = Vector3(-1, -1, 0)
vertex_array[1] = Vector3(-1, 1, 0)
vertex_array[2] = Vector3(1, 1, 0)
vertex_array[3] = Vector3(1, -1, 0)
```

```
index_array[0] = 0
index_array[1] = 1
index_array[2] = 2
index_array[3] = 3
index_array[4] = 0
```

Mallas: estructuras de programación (II)

```
vertex_array[0] = Vector3(-1, -1, 0)
vertex_array[1] = Vector3(-1, 1, 0)
vertex_array[2] = Vector3(1, 1, 0)
vertex_array[3] = Vector3(1, -1, 0)
```

Si no le damos un índice a las caras se asumen que son triángulos (CCW).

En Godot hay cuatro formas de hacer geometría mediante código:

https://docs.godotengine.org/en/3.2/tutorials/content/procedural_geometry/index.html?highlight=geometry

Ejercicio Teórico-Práctico:

Creación de un cubo proceduralmente.

> Crea los arrays necesarios para dibujar las caras de un cubo. Revisa el tutorial de Godot para crear geometría, ignora las normales (*normals*) y las coordenadas de textura (*UV*). Introduce el código en un nodo de Godot del tipo que consideres más apropiado.