Tema VIII

Esquemas de Modelado Sólido

Ricardo Ramos
Colaboradores: Juan Manuel García Fernández, Guillermo Rapado Pelayo, José Isidro Fernández Álvarez, Mª Agustina Bernardo García, José Manuel Martínez García, Juan Daniel de Alaiz Rojo.

En este tema se ven los principales esquemas de representación del Modelado Sólido, es decir, el esquema de representación por fronteras (B.rep) y el esquema booleano (CSG).

1. Introducción

Los modelos comunes suelen ser lo suficientemente complejos como para no poder ser definidos mediante una única ecuación. En otras palabras, no es posible definir los conjuntos de puntos que modelan una silla o un zapato, por ejemplo, mediante una ley que determine si un punto dado pertenece o no al conjunto.

Para poder modelar los objetos complejos no queda más remedio que recurrir a la combinación de objetos simples, conocidos normalmente como **primitivas**. Dependiendo de las primitivas que se utilicen, y del planteamiento que se adopte para combinarlas, nos encontraremos ante diferentes *esquemas de modelado*.

Dentro del Modelado Sólido existen varios esquemas, aunque hay dos que destacan sobre el resto:

Modelado de fronteras, también conocido como **B-Rep** (*Boundary Representation* - Representación de Fronteras), utiliza primitivas bidimensionales. Se describen los sólidos mediante estructuras de datos, donde se registran y organizan las caras (orientadas), las aristas y los vértices, más la información topológica correspondiente.

Modelado booleano, conocido también como **CSG** (*Constructive Solid Geometry* - Geometría Constructiva de Sólidos) combina primitivas tridimensionales. Los sólidos se representan mediante un conjunto de expresiones lógicas booleanas que relacionan las primitivas. Tanto las superficies como el interior del objeto representado quedan definidos implícitamente.

Para poder combinar las primitivas en un esquema de modelado, éstas han de ser definidas previamente. Una forma frecuente de crearlas es *estableciendo los parámetros de las ecuaciones que las definen*. Entre las primi-

tivas más comunes se encuentran los polígonos (exclusivos del modelado de fronteras), esferas, cilindros, conos y toros. Los parámetros suelen ser longitudes, como radios, diámetros, alturas, etc. Además, *es posible definir primitivas 3D mediante la combinación de elementos bidimensionales*, es decir, se puede utilizar un sistema de modelado de fronteras básico para definir las primitivas.

Veamos con más detalle cada uno de los planteamientos anteriores.

2. Modelado de fronteras

Al estudiar un esquema de modelado, una vez hecha la introducción sobre los *conceptos básicos* aplicados en el esquema, se han de analizar las *estructuras de datos* utilizadas, los *problemas de representación* que se presentan, y la *validez de los modelos* creados. Veamos cada uno de estos puntos cuando se modela con B-rep.

2.1 Conceptos básicos.

Los modelos de frontera surgieron a partir de los modelos alámbricos, superando sus deficiencias mediante la incorporación de una descripción completa de las superficies de los objetos sólidos. Un objeto sólido se representa efectuando la división de la superficie en una serie de caras de tamaño y forma convenientes. El criterio que normalmente se aplica para determinar la forma de las caras es el de que éstas *han de quedar definidas matemáticamente de forma completa y sencilla.* Además, será necesario que los trozos de superficies satisfagan ciertos criterios topológicos, para garantizar que la subdivisión se corresponde con un modelo válido.

Como lo más usual es que los objetos modelados sean eulerianos, las caras deberán ser discos topológicos, es decir, sin agujeros. Sin embargo, algunos modeladores pueden trabajar con caras agujereadas, siendo éste uno de los principales criterios de diferenciación entre los modeladores de fronteras. La figura 1 muestra diversos ejemplos de superficies que pueden ser consideradas como caras válidas.

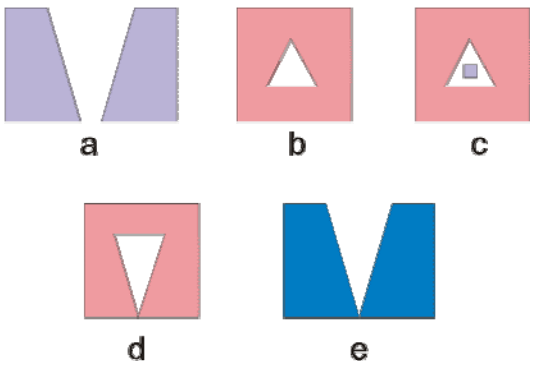


figura 1: definición de las caras

Los casos *a* y *c* representan conjuntos inconexos bidimensionales y se han de registrar como dos caras. En *d* y *e* las fronteras de las caras se tocan; *d* se puede considerar como una cara, mientras que *e* debe registrarse como dos. La cara *b* no sería válida en un esquema de modelado que trabajase ex-

clusivamente con discos topológicos.

Las caras se definen mediante curvas cerradas trazadas sobre la superficie del modelo. A su vez, las curvas cerradas son troceadas convenientemente en aristas, de forma que puedan ser definidas mediante una ecuación paramétrica. Finalmente, las aristas quedarán limitadas por los vértices.

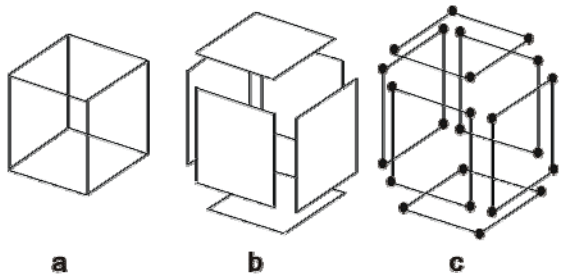


figura 2: componentes básicos del modelo de fronteras.

La figura 2 representa los *componentes básicos* de un modelo de fronteras: la superficie del objeto se divide en un conjunto cerrado de caras (a). Cada una de las caras se representa en términos de polígonos cerrados (b), que a su vez quedan definidos mediante aristas y vértices (c).

Normalmente, los trozos de superficie y sus fronteras (aristas) quedan definidos mediante ecuaciones paramétricas, aunque, si la curvatura de las caras es 0 (K = 0), el modelado de fronteras se puede reducir a trabajar con simples polígonos planos de aristas rectas, lo que simplifica mucho los problemas de modelado y de visualización.

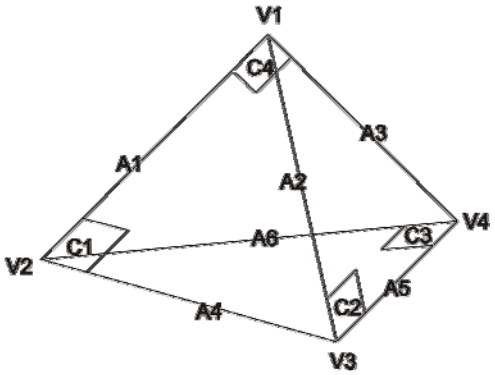
Las caras, aristas y vértices, más la información geométrica asociada (ecuaciones de aristas y superficies, coordenadas de los vértices, etc.), constituyen los componentes básicos de los modelos de fronteras. Éstos, además de la *información geométrica*, han de indicar cómo se relacionan las caras, aristas y vértices (*información topológica).*

Según esto, las estructuras de datos de los modelos de fronteras han de manejar dos tipos diferentes de información: *punteros* para establecer las relaciones topológicas y *los datos numéricos* que determinan la información geométrica. La necesidad de registrar y trabajar con punteros hace que *las estructuras basadas en grafos sean las más idóneas para representar los modelos de fronteras*.

Por lo tanto, un objeto sólido se puede representar como una lista de las caras del objeto y sus respectivas ecuaciones de superficie. Las aristas, definidas mediante ecuaciones, tendrán punteros a los vértices y caras contiguas. Por último, los vértices se representarán como listas de coordenadas, con punteros a las aristas de cada vértice.

En la figura 3 se muestra un ejemplo general de un modelo basado en grafos. Como se ve, el modelo contiene *información redundante*, la cual puede llegar a ser útil para mejorar la velocidad de búsqueda en algunos algoritmos. Aparte de la posibilidad de poder trabajar o no con caras agujereadas, una de las principales diferencias entre los modeladores basados en grafos se encuentra en el **grado de redundancia** que contiene la estructu-

ra utilizada, como pronto veremos.



Vértices	Aristas	Caras
V_1	A_1	C_1
V_2,V_3,V_4 A_1,A_2,A_3 C_1,C_2,C_4	V_1,V_2 A_2,A_3,A_4,A_6 C_1,C_4	V_1,V_2,V_3 A_1,A_4,A_2 C_2,C_3,C_4
V_2	A_2	C_2
V_1,V_3,V_4 A_1,A_4,A_6 C_1,C_3,C_4	V_1,V_3 A_1,A_3,A_4,A_5 C_1,C_2	V_1,V_3,V_4 A_2,A_5,A_3 C_1,C_3,C_4
V_3	A_3	C_3
V_1,V_2,V_4 A_2...	V_1,V_4 A_1, A2...	V_4,V_3,V_2 A_4...

figura 3: modelo basado en grafos.

Según se ha comentado, cuando los sólidos se representan como poliedros de caras planas, se simplifica mucho la información que se ha de registrar, y por lo tanto también las estructuras de datos. Dado que las aristas son líneas rectas, pueden ser registradas como pares de vértices, por lo que las caras quedarán definidas como listas ordenadas de vértices. Los poliedros también pueden quedar registrados mediante un grafo simple, guardando en una tabla las coordenadas de los vértices, y en otra, llamada **matriz de conectividad o matriz de adyacencia**, sus conexiones (relaciones topológicas).

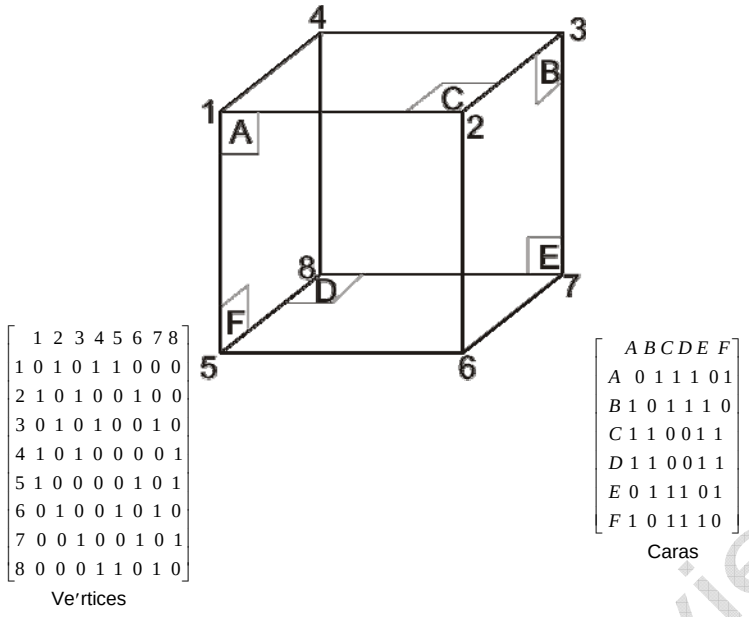


figura 4: matrices de conectividad

En la figura 4 tenemos un ejemplo de las matrices de conectividad de los vértices y las caras de un cubo. Un "1" en la matriz indica que las caras o vértices están conectados.

2.2 Estructuras de datos

Todos los esquemas de modelado de fronteras representan las caras de modo explícito en sus estructuras de datos. El resto de la información geométrica y topológica puede quedar registrada de muchas formas diferentes, con mayor o menor grado de redundancia. Veamos algunas de las principales estructuras de datos utilizadas en la representación de los modelos poliédricos. Para poder comparar mejor los resultados, en todos los casos utilizaremos como referencia el mismo modelo.

2.2.1 Estructuras basadas en polígonos

En esta representación los objetos quedan registrados como una colección de polígonos, siendo cada polígono una secuencia de coordenadas (x, y, z). Esta secuencia puede almacenarse tanto en una tabla, donde cada polígono tendrá un descriptor que lo identifique, como en una lista encadenada. Aunque estas estructuras son muy sencillas, sin embargo generan bastante información redundante, ya que las coordenadas de un mismo vértice se han de registrar tantas veces como caras convergen en él. Por lo tanto, en los modelos con muchos polígonos (caso muy frecuente), no es la solución más idónea.

2.2.2 Estructuras basadas en vértices.

La redundancia en la estructura anterior puede ser eliminada si se considera a los vértices como elementos independientes de los objetos, aunque han de quedar asociados con las caras. Este planteamiento permite la posibilidad de diseñar varias estructuras diferentes.

La lista de vértices de cada cara *viene dada en un orden consistente* (en este caso, en el sentido de giro de las agujas del reloj, visto desde fuera del cubo). De este modo es posible averiguar la orientación (normal) de las caras, información que será muy útil a la hora de eliminar aristas y caras ocultas. Veamos un ejemplo:

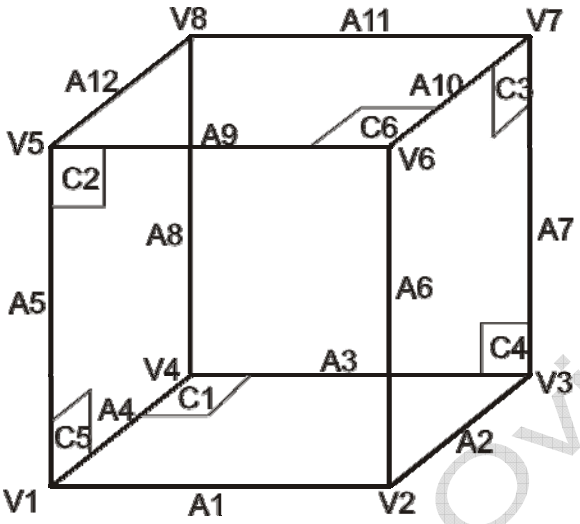


figura 5

Tabla de vértices	
Vértices	Coordenadas
V_1	$X_1\ Y_1\ Z_1$
V_2	$X_2\ Y_2\ Z_2$
V_3	$X_3\ Y_3\ Z_3$
V_4	$X_4\ Y_4\ Z_4$
V_5	$X_5\ Y_5\ Z_5$
V_6	$X_6\ Y_6\ Z_6$
V^7	$X_7\ Y_7\ Z_7$
V_8	$X_8\ Y_8\ Z_8$

Tabla de caras	
Caras	Vértices
C_1	$V_1\ V_2\ V_3\ V_4$
C_2	$V_6\ V_2\ V_1\ V_5$
C_3	$V_7\ V_3\ V_2\ V_6$
C_4	$V_8\ V_4\ V_3\ V_7$
C_5	$V_5\ V_1\ V_4\ V_8$
C_6	$V_7\ V_6\ V_5\ V_8$

tabla 1: estructura de vértices

En el ejemplo, gracias a la orientación implícita de los polígonos, un algoritmo de eliminación de caras ocultas descartaría fácilmente las caras C1, C4 y C5. La estructura no incluye información alguna sobre las superficies. Como las caras son planas, su geometría queda completamente definida a través de las coordenadas de los vértices.

Ver que la información que se deja implícita *se ha de calcular cuando se necesite, lo que supone consumo de tiempo*. Por el contrario, si se registra toda la información de forma explícita, *se produce un consumo mayor de espacio*. Por tanto, se ha de buscar el equilibrio "espacio/tiempo", en función de la velocidad de cálculo y la memoria disponible.

El ocupar espacio extra con información redundante, normalmente permite acelerar su búsqueda, aunque pueden surgir problemas inesperados. Por ejemplo, por falta de precisión, se pueden originar pequeñas diferencias entre los valores calculados y los registrados de las coordenadas de los vértices, lo que dificulta averiguar cuál es el valor correcto.

2.2.3 Estructuras basadas en aristas

Con este método, las caras de los objetos se representan como secuencias de aristas que forman un camino cerrado. La tabla siguiente está basada en el modelo de la figura 5.

Tabla de aristas

Aristas	Vértices
A_1	V_1 V_2
A_2	V_2 V_3
A_3	V_3 V_4
A_4	V_4 V_1
A_5	V_1 V_5
A_6	V_2 V_6
A_7	V_3 V_7
A_8	V_4 V_8
A_9	V_5 V_6
A_{10}	V_6 V_7
A_{11}	V_7 V_8
A_{12}	V_8 V_5

Tabla de vértices

Vértices	Coordenadas
V_1	X_1 Y_1 Z_1
V_2	X_2 Y_2 Z_2
V_3	X_3 Y_3 Z_3
V_4	X_4 Y_4 Z_4
V_5	X_5 Y_5 Z_5
V_6	X_6 Y_6 Z_6
V_7	X_7 Y_7 Z_7
V_8	X_8 Y_8 Z_8

Tabla de caras

Caras	Aristas
C_1	A_1 A_2 A_3 A_4
C_2	A_9 A_6 A_1 A_5
C_3	A_{10} A_7 A_2 A_6
C_4	A_{11} A_8 A_3 A_7
C_5	A_{12} A_5 A_4 A_8
C_6	A_{12} A_{11} A_{10} A_9

tabla 2: estructura de aristas

En la tabla de aristas se indica la orientación de cada arista. Por ejemplo, la arista A_1 se considera positivamente orientada desde el vértice V_1 al V_2. Además, las caras también se encuentran orientadas, ya que las aristas se listan en sentido consistente (en este caso, el de las agujas del reloj, visto desde fuera del sólido). Ver que cada arista aparece en dos caras, una con orientación positiva (p. e., sentido del reloj) y la otra con orientación negativa.

Esta sería una estructura de aristas básica. Se puede incluir más o menos información en una estructura, dependiendo de las necesidades y las ventajas que se obtengan durante la búsqueda (visualización). Por ejemplo, se puede asociar a cada arista el identificador del par de caras que separa; esta información podría ser útil para los algoritmos de eliminación de caras ocultas y/o en los de aplicación de intensidad.

A) Estructura winged-edge.

Las estructuras simples obligan a que los cálculos sean bastante caros. Por ejemplo, descubrir si dos caras están unidas por una arista para verificar la integridad del modelo, requiere obtener la lista de aristas de todas las caras.

Las estructuras sofisticadas tratan de incluir la mayor cantidad de información posible dentro de un volumen de datos razonable y rápidamente accesible, para disminuir el costo de los cálculos. Entre las estructuras más elaboradas, una buena representante es conocida como *winged-edge*, desarrollada por Baumgart en 1972.

Además de registrar información sobre las aristas y las caras, en esta estructura *también quedan indicados los bucles (caminos cerrados simples) que forman las aristas al ser recorridas en el sentido positivo y el negativo.*

i.- Creación de la estructura winged-edge

Al ser los poliedros objetos eulerianos, en las estructuras basadas en aristas ha de tener en cuenta que *cada arista separa exactamente a 2 caras*.

Entonces, dada una arista *a* y siendo *C'*, C" las caras que separa, si recorremos la arista en el sentido de orientación positivo (+), al llegar al vértice nos encontraremos con la arista *a',* que pertenece a la cara *C';* si se recorre *a* en el sentido negativo (–), nos encontraremos con *a"*, que pertenece a *C". El sentido de recorrido,0 y las aristas localizadas al llegar a los vértices, constituyen la información sobre los bucles que se registra en la winged-edge.*

Según esto, para crear la tabla de aristas en la winged-edge se opera como sigue:

Se establece el sentido de recorrido consistente de los bucles que forman las aristas de las caras. En nuestro caso hemos optado por el sentido de giro de las agujas del reloj, considerando los objetos desde el exterior.

Por cada arista a_i, se busca el par de caras C_a, C_b que separa y se establece el sentido de recorrido positivo de la arista. Es indiferente el sentido elegido, aunque, una vez establecido, no se puede variar. En nuestro caso, el sentido de recorrido positivo (+) vendrá dado por (V_a, V_b), que indica que la arista se recorre en sentido positivo cuando se va del vértice V_a al V_b.

Se recorre la arista a_i en el sentido +, hasta llegar al vértice. Entonces, se elige una arista de la cara C_a o una de la cara C_b, de forma que el recorrido por la nueva arista *sea en el sentido de recorrido consistente de la cara* (agujas del reloj). Sólo una de las aristas cumple esta condición, que será la que se registre en la estructura (puntero R+).

Se vuelve a recorrer la arista a_i, pero esta vez en el sentido –. Cuando se llaga al vértice opuesto, se aplica de nuevo el criterio del punto anterior, y la arista seleccionada quedará indicada por el puntero R–.

Un ejemplo de esta representación se muestra en la tabla siguiente, que corresponde a la figura 6 (que es idéntica a la figura 5).

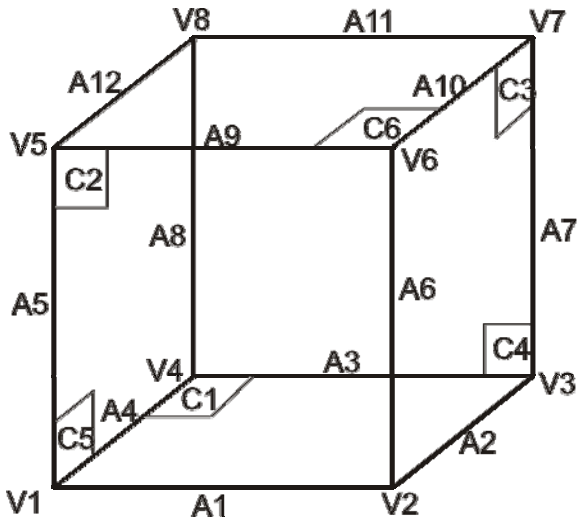


figura 6

Tabla de aristas

Aristas	Vértices	R+	R–
A_1	(V_1,V_2)	A_2	A_5
A_2	(V_2,V_3)	A_3	A_6
A_3	(V_3,V_4)	A_4	A_7
A_4	(V_4,V_1)	A_1	A_8
A_5	(V_1,V_5)	A_9	A_4
A_6	(V_2,V_6)	A_{10}	A_1
A_7	(V_3,V_7)	A_{11}	A_2
A_8	(V_4,V_8)	A_{12}	A_3
A_9	(V_5,V_6)	A_6	A_{12}
A_{10}	(V_6,V_7)	A_7	A_9
A_{11}	(V_7,V_8)	A_8	A_{10}
A_{12}	(V_8,V_5)	A_5	A_{11}

Tabla de vértices

Vértices	Coordenadas
V_1	X_1 Y_1 Z_1
V_2	X_2 Y_2 Z_2
V_3	X_3 Y_3 Z_3
V_4	X_4 Y_4 Z_4
V_5	X_5 Y_5 Z_5
V_6	X_6 Y_6 Z_6
V_7	X_7 Y_7 Z_7
V_8	X_8 Y_8 Z_8

Tabla de caras

Caras	Aristas de inicio	Sentido
C_1	A_1	+
C_2	A_9	+
C_3	A_6	+
C_4	A_7	+
C_5	A_{12}	+
C_6	A_9	–

tabla 3: estructura winged-edge

En la construcción de la *tabla de caras*, por cada cara sólo se ha de incluir el identificador de cualquiera de sus aristas, más un bit que indique la orientación de la arista elegida (en la tabla de aristas), cuando se recorre la cara en el sentido consistente. Por ejemplo, para la primera cara (C_1), se ha tomado como representante la arista A_1. Al recorrer la cara C_1 en el sentido consistente (agujas del reloj), la arista A_1 se recorre de V_1 a V_2. Si nos fijamos en la tabla de aristas, vemos que de V_1 a V_2 (V_1, V_2) ha sido establecido como el sentido positivo de recorrido de la arista A_1. Por lo tanto, se ha incluido un + en la *tabla de caras*.

ii.- Búsqueda de la información

Para buscar la información sobre los poliedros en una estructura winged-edge se procede como sigue:

En la *tabla de caras*, se localiza la arista y el sentido de recorrido de la cara buscada. Por ejemplo, si la cara buscada es la C_5, entonces la información inicial es (A_{12}, +).

En la *tabla de aristas* se averigua cuál es el vértice y la próxima arista a la que llega, cuando se recorre según el sentido indicado. Para la arista (A_{12}, +), el recorrido finaliza V_5, y la siguiente arista es la A_5 (*recorrido*(A_{12}, +)→(V_5, A_5)).

Finalmente, se recorre la nueva arista partiendo desde el vértice encontrado hasta el vértice opuesto. En el ejemplo, yendo desde V_5 a V_1 en la arista A_5, el sentido de recorrido es negativo. Por tanto, según la *tabla de aristas*, se llegará al vértice V_1 de la arista A_4 (*recorrido*(A_5, –)→(V_1, A_4)). El proceso se repetirá hasta llegar a la arista de partida (A_{12}).

B) Estructura winged-edge completa

En la versión general de la winged-edge, en la *tabla de aristas* se incluyen los identificadores CR+ y CR–, que son punteros hacia las caras que separa una arista dada, y cuyo sentido consistente de giro coincide con el sentido de recorrido positivo y negativo de la arista. Además, de manera análoga a los punteros R+ y R–, se incluyen el PR+ y el PR–, que apuntan a las aristas

previas, cuando se recorre la arista en sentido positivo y negativo.

La figura 7 muestra hacia donde apuntan los diferentes punteros de la estructura, cuando la arista se recorre de V_a a V_b (sentido positivo) y de V_b a V_a (sentido negativo).

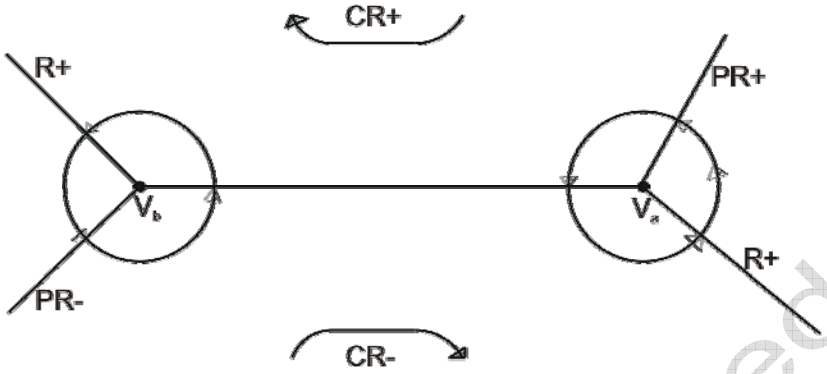


figura 7: winged-edge general

Basándonos de nuevo en la figura 6, veamos cómo quedan las diferentes tablas para la estructura completa de winged-edge:

Tabla de aristas

Aristas	Vértices	CR+	CR–	R+	PR+	R–	PR–
A_1	(V_1,V_2)	C_1	C_2	A_2	A_4	A_5	A_6
A_2	(V_2,V_3)	C_1	C_3	A_3	A_1	A_6	A_7
A_3	(V_3,V_4)	C_1	C_4	A_4	A_2	A_7	A_8
A_4	(V_4,V_1)	C_1	C_5	A_1	A_3	A_8	A_5
A_5	(V_1,V_5)	C_2	C_5	A_9	A_1	A_4	A_{12}
A_6	(V_2,V_6)	C_3	C_2	A_{10}	A_2	A_1	A_9
A_7	(V_3,V_7)	C_4	C_3	A_{11}	A_3	A_2	A_{10}
A_8	(V_4,V_8)	C_5	C_4	A_{12}	A_4	A_3	A_{11}
A_9	(V_5,V_6)	C_2	C_6	A_6	A_5	A_{12}	A_{10}
A_{10}	(V_6,V_7)	C_3	C_6	A_7	A_6	A_9	A_{11}
A_{11}	(V_7,V_8)	C_4	C_6	A_8	A_7	A_{10}	A_{12}
A_{12}	(V_8,V_5)	C_5	C_6	A_5	A_8	A_{11}	A_9

tabla 4: tabla de aristas en la winged-edge completa

Tabla de vértices

Vértices	Arista de inicio	Coordenadas
v_1	A_1	x_1 y_1 z_1
v_2	A_2	x_2 y_2 z_2
v_3	A_3	x_3 y_3 z_3
v_4	A_4	x_4 y_4 z_4
v_5	A_9	x_5 y_5 z_5
v_6	A_{10}	x_6 y_6 z_6
v_7	A_{11}	x_7 y_7 z_7
v_8	A_{12}	x_8 y_8 z_8

Tabla de caras

Caras	Arista de inicio
C_1	A_1
C_2	A_9
C_3	A_6
C_4	A_7
C_5	A_{12}
C_6	A_9

tabla 5: tablas de vértices y caras de la winged-edge completa

Ver que en la *tabla de caras* ya no es preciso incluir los signos de orientación, ya que resultan redundantes. En la *tabla de vértices*, aunque no sea estrictamente necesario, a cada vértice se le asocia la arista en la que dicho

vértice es punto de partida en el sentido positivo. De este modo se facilita la búsqueda, o lo que es igual, se simplifican los cálculos. Dado que un mismo vértice puede ser punto de partida en más de una arista, se elige una de ellas. Por ejemplo, V_1 es punto de partida para las aristas A_1 y A_5.

El uso de la estructura winged-edge se ha extendido notablemente. Muchos de los sistemas de Modelado Sólido la adoptaron como representación de fronteras. Sin embargo, no ha tenido mucho éxito en el modelado de sólidos con agujeros en las caras (no eulerianos).

2.3 Caras con varias fronteras (caras con agujeros)

Las estructuras que se han mostrado hasta el momento solamente son válidas para representar los poliedros eulerianos de superficies planas, es decir, sin agujeros en los polígonos. Sin embargo, a veces es interesante que el modelador pueda trabajar con superficies con varias fronteras, o lo que es igual, superficies perforadas.

Entre los posibles planteamientos que permiten trabajar con caras agujereadas, dos son los más obvios.

El primero, visto en el tema anterior, consiste en transformar los modelos no eulerianos en objetos de Euler, *subdividiendo las superficies agujereadas en discos topológicos, mediante la introducción de aristas auxiliares.* La figura 8 muestra nuevamente este procedimiento.

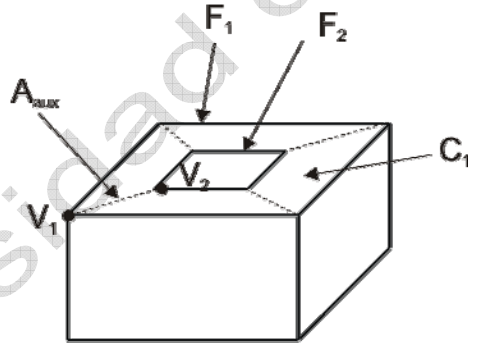


figura 8: Transformación en objeto de Euler

Este método tiene el problema de que las aristas auxiliares deben ser marcadas de algún modo, para que no aparezcan durante el trazado de las aristas, p. e., en la visualización del modelo en formato alámbrico. Sin embargo, con la winged-edge completa no es necesario diferenciar las aristas auxiliares, ya que éstas, recorridas en sentido positivo y negativo pertenecen a la misma cara[1], por lo que son fácilmente distinguibles. Así, en la figura 8, al recorrer la arista A_{aux} de V_1 a V_2, por las normas establecidas pertenece a la cara C_1. Si se recorre de V_2 a V_1, por las mismas reglas también pertenece a C_1.

[1] Ver que con la introducción de las aristas auxiliares las nuevas caras que se forman no llegan a tomar identidad propia. Su finalidad principal es la de crear bucles cerrados y discos topológicos que permitan mantener la generalidad de los algoritmos de búsqueda, eludiendo las excepciones.

El otro planteamiento, consiste en crear tantos bucles cerrados como fronteras tenga la cara, y asociar a cada cara una lista con los bucles (fronteras) que posee. En el ejemplo anterior (figura 8), $C_1(F_1, F_2)$ sería la lista de bucles de la cara 1. Como vemos, es una manera explícita de asociar la información a cada cara, fácilmente aplicable a todas las estructuras de datos que se hemos visto.

2.4 Problemas en la representación de fronteras

El modelado de fronteras no está exento de problemas, sobre todo si se desea operar booleanamente con los objetos B-rep.

Por ejemplo, supongamos que se quiere efectuar la unión entre dos objetos en forma de L, tal como muestra la figura 9.

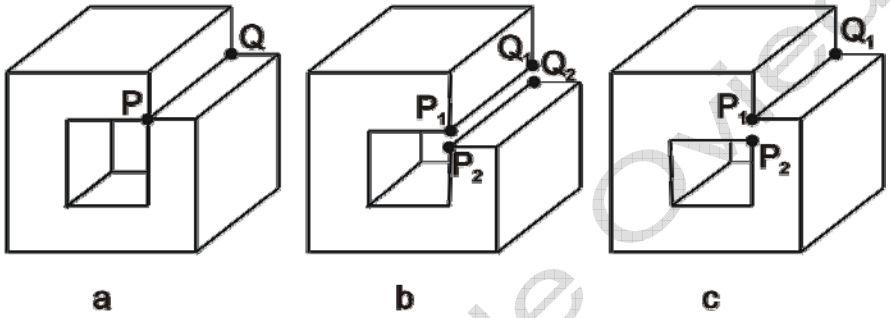


figura 9: diferentes interpretaciones de un mismo resultado

Vemos que el resultado inicial (a) no es un objeto euleriano, dado que la arista (P, Q) pertenece a 4 caras.

Para resolver este tipo de problemas, los sistemas de modelado pueden adoptar una de estas tres posturas:

Todos los objetos deben ser eulerianos, por lo que los resultados no eulerianos (p. e., el "a" en la figura 9) serán considerados como error. Este planteamiento radical, aunque puede dar buenos resultados, impone severas restricciones a los modeladores, sobre todo los que efectúan operaciones booleanas. Además, dependiendo de cómo se haya implantado internamente el sistema, incluso pueden llegar a rechazar los resultados intermedios resultantes al aplicar los operadores de Euler.

Los objetos son todos eulerianos, pero se permiten las coincidencias geométricas en el espacio 3D, aunque sus estructuras topológicas estén separadas, es decir, sean independientes. Este planteamiento obliga a que se tenga que dar una interpretación euleriana, de las estructuras no eulerianas.

Volviendo al ejemplo de la figura 9, la arista (P, Q) se ha de convertir en dos aristas, aunque sus coordenadas geométricas coincidan. El problema está en que hay dos posibles interpretaciones topológicas: la "b", que como vemos se trata de un objeto de grado 0 (familia de las esferas), y la "c", que es un objeto euleriano de grado 1 (familia del toro). ¿Cuál de las dos es la apropiada ?. Normalmente se opta por la solución más simple, es decir, *los resultados de menor grado*, en este caso el modelo "b".

3. *Se permite la existencia de objetos no eulerianos*. No cabe duda que esta es la solución más cómoda, ya que simplifica mucho los algoritmos al no tener que verificar la integridad de los modelos. En contrapartida, deberá ser el usuario quien ponga cuidado en que sus modelos sean más o menos coherentes. Por definición, este planteamiento queda descartado del Modelado Sólido estricto, aunque puede ser válido en otros muchos casos, especialmente si los modelos son para realizar síntesis de imágenes.

Además de los problemas topológicos, el modelado de fronteras también presenta algunos problemas prácticos. Una de las causas se debe a que los modelos poliédricos sólo se aproximan a las superficies de los objetos que no son poliedros, requiriendo gran cantidad de datos para aproximar decentemente los objetos con superficies curvas. Por ejemplo, consideremos el problema de representar un objeto cilíndrico introducido en un agujero, mediante b-reps poliédricos (ver la figura 10). Si las fronteras de los dos objetos se tocan, suponiendo que inicialmente ambas coinciden, éstas intersecarán si uno de los objetos es girado ligeramente, independientemente de la cantidad de polígonos se usen en la aproximación.

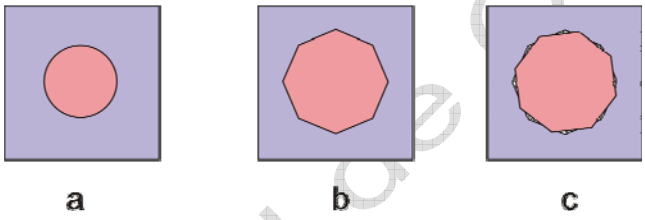


figura 10: Problemas en las aproximaciones poliédricas

2.5 Validez de los modelos de fronteras

En principio, *un modelo de fronteras es válido si se define la frontera de un objeto de forma coherente*. Sin embargo, dado que el Modelado Sólido se interesa sobre todo por los objetos 3D cerrados con superficies orientadas (eulerianos o no), los criterios que los modelos de fronteras han de respetar para que puedan considerarse válidos son los siguientes:

1. El conjunto de caras del modelo es cerrado, es decir, forma completamente el sólido sin que falte ninguna parte
2. Las caras no se intersecan unas con otras, excepto en los vértices y aristas comunes.
3. Las fronteras de las caras son polígonos simples que no se intersecan a sí mismos.

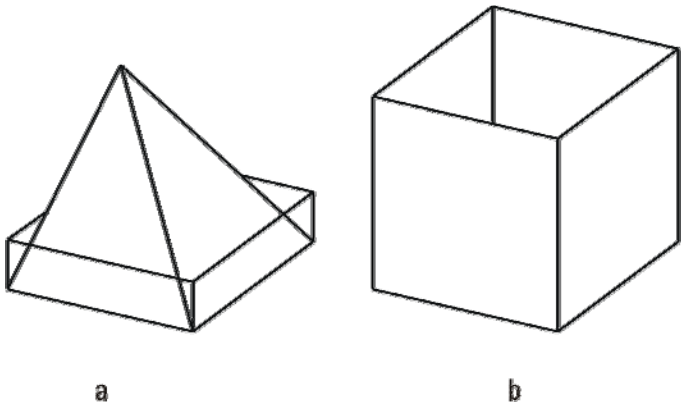


figura 11: Modelos de frontera no válidos.

La 2ª y 3ª condición excluyen objetos que se cortan entre ellos mismos como el objeto de la figura 11-a. Ver que la superficie del fondo de la pirámide interseca con la superficie principal.

La 1ª condición prohíbe objetos como la caja sin tapadera de la figura 11-b. Para representar objetos abiertos con un modelo de frontera, el mejor enfoque es modelar las zonas vacías con caras especialmente marcadas (p. e., transparentes), en lugar de omitirlas.

La 1ª condición también está directamente relacionada con la exigencia de que cada arista deba pertenecer a dos caras, para que los modelos de fronteras posean integridad topológica: *si en algún lado falta un trozo de superficie, necesariamente habrá una o más aristas que no cumplan esta condición*. Ver que la estructura winged-edge automáticamente obliga a que la 1ª condición sea cierta, ya que, por construcción, esta estructura no admite aristas que pertenezcan a una sola cara. Además, también fuerza a que las caras queden orientadas[2].

A diferencia de lo que ocurre con la 1ª condición, *no se puede garantizar que las condiciones 2ª y 3ª sean ciertas, por el mero hecho de aplicar estrictamente las normas de construcción de las estructuras*. Asignando valores geométricos inapropiados es posible crear modelos erróneos, aunque se cumplan las condiciones anteriores y las características topológicas sean perfectas.

Por lo tanto, para garantizar la *integridad geométrica*, o bien se recurre a pruebas computacionalmente costosas, basadas en la comparación de cada par de caras del sólido, o bien se limita la libertad de modelado del usuario, proporcionándole mecanismos de descripción que necesariamente obliguen a un diseño válido.

2.6 Descripción de los modelos de fronteras

Uno de los principales problemas de los modelos de fronteras reside en la complejidad de su descripción. Las habilidades de una persona normal dis-

[2] Los objetos no orientables están prohibidos también por la segunda condición, debido a que siempre se cortan a sí mismos en el espacio tridimensional.

tan mucho de poder construir un modelo, más o menos complejo, proporcionando directamente la información geométrica y topológica de los vértices, aristas y caras. Por tanto, el diseñador de modelos de fronteras ha de disponer de una serie de herramientas que faciliten la descripción de los modelos.

Estas herramientas, a grosso modo, pueden clasificarse en dos grupos:

Conversores, que traducen a B-rep los modelos creados mediante otros esquemas de modelado, más fáciles e intuitivos de manejar para el usuario (CSG, barrido, NURBS, etc.). Dependiendo de cuál sea el esquema de modelado elegido para el diseño, las técnicas de conversión varían mucho. Por ejemplo, si se ha utilizado CSG se utilizan los *conversores a fronteras*. En cambio, si el modelo ha sido descrito mediante NURBS (o superficies curvas, en general), se aplican procesos de *teselación*, para convertir las superficies curvas en un mosaico de polígonos planos.

Modificadores locales, que, como su nombre indica, son capaces de realizar automáticamente pequeñas modificaciones locales, como el redondeado de esquinas y bordes. Estas modificaciones se efectúan mediante los operadores de Euler, sustituyendo una o más caras por un conjunto de caras más pequeñas, convenientemente dispuestas. En la figura 12 se pueden ver tres modificaciones locales, de menor a mayor dificultad.

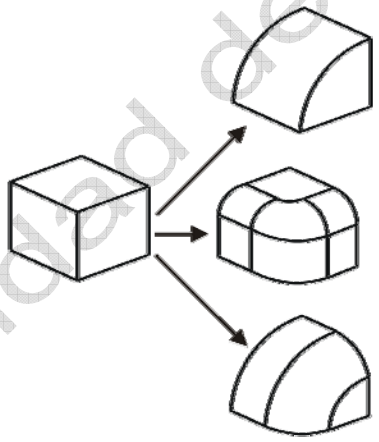


figura 12: Modificaciones locales en los modelos B-rep

3. Modelado Booleano (CSG)

Al conjunto de procesos que permiten el diseño de sólidos complejos mediante la combinación booleana de objetos simples (*primitivas*) se conoce como *modelado booleano*, o también como *Geometría Constructiva de Sólidos* (CSG).

En el modelado booleano, después de trasladar, girar y escalar convenientemente las primitivas, se aplican los *operadores booleanos regularizados* ($\cup^*$, $\cap^*$ y $-^*$) para combinarlas. Además de los procesos anteriores, también son necesarios métodos de clasificación de vértices y aristas, así como de *evaluación de fronteras*, según veremos más adelante.

Dado que los operadores booleanos son regularizados, todos los sólidos con los que se va a operar, así como los objetos resultantes, han de tener la

misma dimensión espacial.

Un sólido construido mediante un modelador booleano queda descrito mediante un árbol binario en el que:

- El nodo raíz es el sólido resultante.

- Los nodos internos son los operadores booleanos.

- Los nodos hoja son las primitivas.

En los nodos terminales del árbol no sólo se representan las primitivas, sino que también se indican las transformaciones lineales que se han de efectuar sobre ellas. En el árbol, tanto las operaciones como los nodos hoja deberán encontrarse ordenadas, debido a que las operaciones booleanas no son, en general, conmutativas. El proceso de seguimiento (recorrido) del árbol se hará partiendo de los nodos terminales.

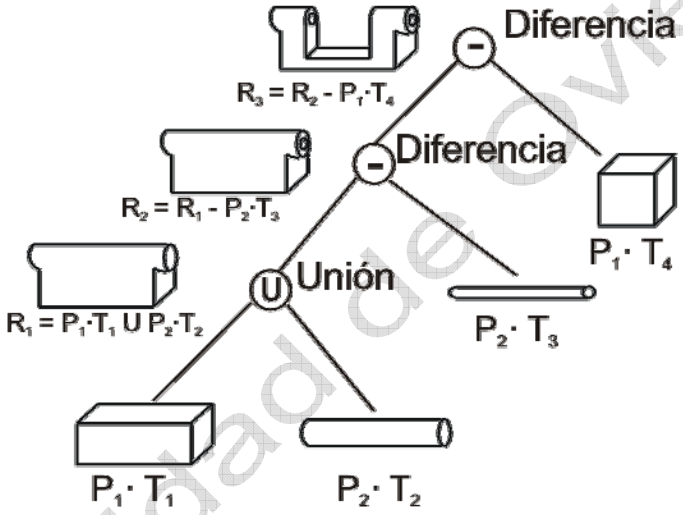


figura 13: árbol binario de un modelo booleano

En el ejemplo de la figura 13 se puede ver la descripción de un modelo booleano, llamado $\mathbf{R_3}$. Partiendo de las primitivas $\mathbf{P_1}$ (cubo) y $\mathbf{P_2}$ (cilindro), se obtiene el objeto final (R_3) después de tres niveles de operaciones booleanas.

En primer lugar se efectúa la unión de P_1 y P_2, después de haber transformado y posicionado adecuadamente las primitivas en el espacio, mediante las matrices netas T_1 y T_2. Por lo general, el escalado de las primitivas *no es uniforme*, es decir, los factores de escala S_x, S_y, y S_z son diferentes. De este modo, una sola primitiva proporciona una infinita variedad de copias diferentes.

A continuación, al resultado intermedio R_1 se le quita $\mathbf{P_2}$, después de haber sido escalada y posicionada la primitiva por T_3. Por la figura vemos que los factores de escala de T_3 son diferentes a los de T_2, ya que el cilindro conserva su altura, pero no el diámetro.

Finalmente, a R_2 se le resta la primitiva $\mathbf{P_1}$, trasformada esta vez por T_4, obteniéndose el objeto raíz del árbol R_3.

Ver que el modelado booleano *es un proceso descriptivo*, es decir, *sólo se*

limita a especificar qué primitivas se utilizan y cómo se combinan. En otras palabras, *en los modelos booleanos sólo se dispone de la información geométrica y topológica de las primitivas, pero no la de los objetos modelados*. Por tal motivo, también se les denomina *modelos descriptivos* y/o *no evaluados*, ya que para conocer sus características geométricas y topológicas es preciso efectuar un proceso de *evaluación*, conocido como **evaluador de fronteras**, que se encarga de hallar las intersecciones entre las superficies de las primitivas, para luego poder buscar los vértices y aristas. Además, ha de analizar la conectividad de los elementos encontrados, para determinar la topología del modelo.

3.1 Definición de las primitivas

Como sabemos, las primitivas son los únicos elementos de los modelos booleanos (sin evaluar) que disponen de la información geométrica y topológica. Por tal motivo, para su definición se ha de utilizar un método que permita el registro de dicha información.

Muchos modeladores utilizan un esquema de fronteras (basado en grafos) para definir las primitivas. Aunque en principio pueda parecer una solución exagerada, no es un planteamiento desacertado ya que la mayoría de los sistemas de modelado han de trabajar conjuntamente con modelos de fronteras y booleanos (*modeladores duales)*. Por lo tanto, se pueden aprovechar todos los procesos y herramientas del modelador de fronteras para definir las primitivas del modelador booleano.

Otros modeladores utilizan un esquema de *barrido*, que consiste en trasladar o girar una línea o superficie generadora, para definir las primitivas. Los conceptos básicos de este método de modelado serán expuestos en el tema siguiente.

De cualquier modo, el método más ortodoxo de definición se basa en la *descripción algebraica de las primitivas.*

3.1.1 Definición algebraica de las primitivas

A cada conjunto de puntos $\mathcal{R}$, en principio se le puede asignar una *función característica* $g_{\mathcal{R}}(X) : X \rightarrow \{0,1\}$, que indica si un punto cualquiera X pertenece o no a $\mathcal{R}$. En otras palabras, si

Ec. 1

$$g_{\mathcal{R}}(X) = 1 \Rightarrow X \in \mathcal{R}$$

$$g_{\mathcal{R}}(X) = 0 \Rightarrow X \notin \mathcal{R}$$

Según se comentó al inicio del tema, esta función característica no es de mucha ayuda cuando se trata de conjuntos de puntos generales (sillas, zapatos), ya que su definición sería bastante ardua. Sin embargo, existe una clase interesante de conjuntos de puntos, cuya función característica puede quedar representada por una ecuación algebraica sencilla, del tipo $F(x, y, z) = 0$, siendo x, y, z variables reales. En estos conjuntos se cumple que:

$\forall\ X = (x, y, z)$, si $F(X) \geq 0 \Rightarrow X \in \mathcal{R}$, si $F(X) < 0 \Rightarrow X \notin \mathcal{R}$, es decir que pertenece al complementario de $\mathcal{R}$.

A) Semiespacios

La función $F(X) = 0$ define una superficie que divide el espacio cartesiano en dos regiones o conjuntos de puntos. *Estas dos regiones se llaman semiespacios y están representados por las funciones* $F(X) \geq 0$ y $F(X) \leq 0$. Por lo tanto, la función $F(x, y, z)$ vale 0 en la superficie, y toma valores distintos de 0 en el interior del semiespacio que contiene los puntos que van a formar parte del sólido. Los semiespacios quedan representados matemáticamente mediante la función $F(x, y, z)$, y mediante la ecuación $f(x, y, z) \geq 0$ o $f(x, y, z) \leq 0$. Ver que en cualquier caso, los puntos de las superficies pertenecen a los semiespacios.

Como representante típico de los semiespacios tenemos el *semiespacio planar*, cuya función característica es $Ax + By + Cz + D \geq 0$. Este semiespacio se define como la unión de los puntos del plano definido por $Ax + By + Cz + D = 0$ (conjunto de puntos de frontera *fR*), más los puntos de uno de los subespacios separados por el plano, en este caso el positivo (puntos de interior *iR*).

Otro ejemplo lo tenemos en el *semiespacio cilíndrico*, formado por todos los puntos que se encuentran en el interior y en la superficie de un cilindro infinito, con el eje en Z. En este caso, su ecuación característica es $x^2 + y^2 - r^2 \leq 0$.

La figura 14 muestra los semiespacios anteriores. Las flechas señalan hacia los puntos que pertenecen al semiespacio.

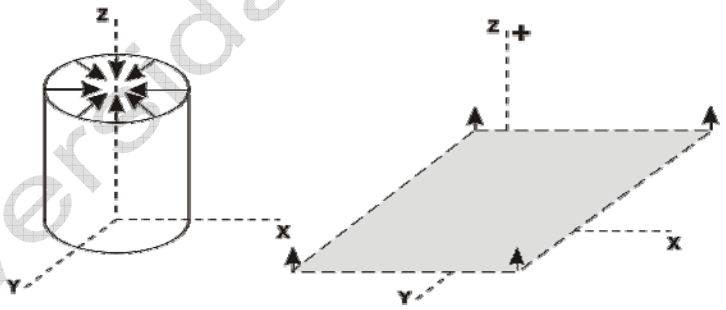


figura 14: ejemplos de semiespacios abiertos

Los ejemplos anteriores representan superficies ilimitadas que dividen el espacio cartesiano en dos regiones ilimitadas, generando semiespacios abiertos. Sin embargo, también puede haber semiespacios cerrados.

Así, la ecuación $x^2 + y^2 + z^2 - r^2 = 0$ define una superficie cerrada que divide el espacio cartesiano en una región ilimitada y otra limitada. La superficie representa una esfera de radio *r*, y el *semiespacio esférico* que genera queda definido por $x^2 + y^2 + z^2 - r^2 \leq 0$, o sea, por todos los puntos que están dentro (*iR)* y en la superficie de la esfera (*fR*).

Otros semiespacios interesantes para modelado, definidos por ecuaciones características de grado 2 o superior, son los generados por las superfi-

cies cónicas (*semiespacios cónicos* abiertos, $x^2 + y^2 - z^2 \leq 0$), y por los toros (*semiespacios tóricos* cerrados, $(x^2 + y^2 + z^2 - (a^2 + b^2))^2 \leq 4a^2(b^2 - z^2)$). La figura 15 muestra estos semiespacios, con los parámetros que intervienen en sus ecuaciones características.

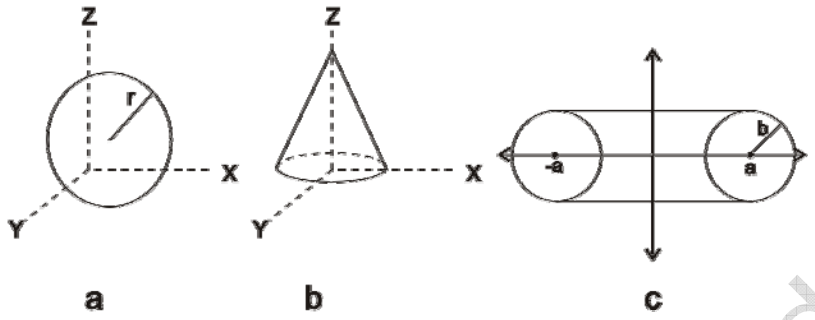


figura 15: semiespacio esférico, cónico y tórico

En la figura 15-c se muestra una sección transversal del toro, que consiste en dos círculos, de radio ***b***, centrados en $x = \pm a$, $z = 0$.

B) Primitivas de semiespacios abiertos

Por definición, el modelado booleano requiere que las primitivas sean sólidos cerrados. Por tanto, si un modelador desea utilizar semiespacios abiertos (planares, cilíndricos, cónicos, etc.) antes ha de efectuar una *composición de semiespacios*, de modo que el semiespacio resultante sea un espacio cerrado.

Como los semiespacios son conjuntos de puntos, los procesos naturales para crear primitivas cerradas a partir de semiespacios abiertos, no son otros que los operadores booleanos regularizados de unión, intersección y diferencia, aunque normalmente, *con la intersección es suficiente.* En definitiva, *las primitivas que requieren semiespacios abiertos en su definición, se construyen combinando copias de semiespacios, mediante operaciones booleanas*.

Un ejemplo lo tenemos en la construcción de un cubo de longitud de arista $l = 1$, posicionado en el origen de coordenadas (figura 16-a). En este caso, el cubo se construye mediante la intersección de 6 semiespacios planares, convenientemente orientados.

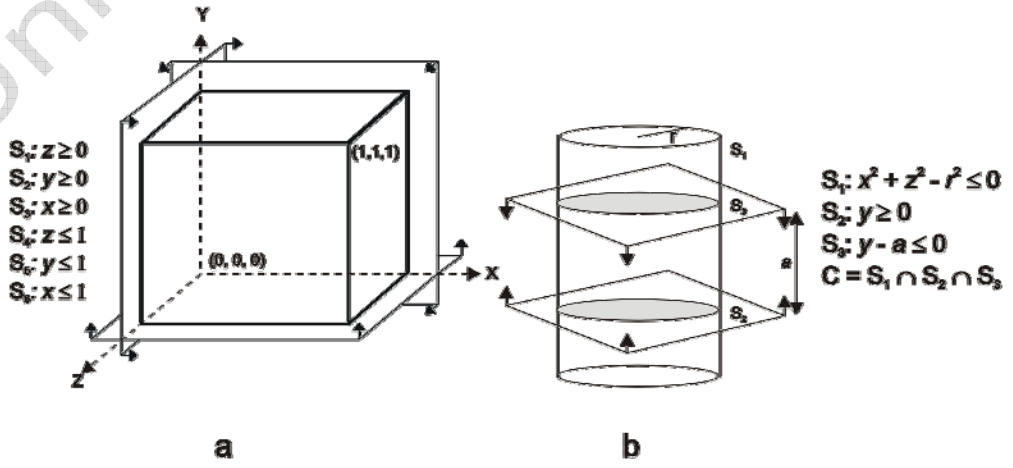


figura 16: obtención primitivas a partir de semiespacios.

En el ejemplo de la figura 16-a, los seis semiespacios planares vienen dados por:

$$S_1: Z \geq a,\ S_4: Z \leq b;\ S_2: Y \geq c,\ S_5: Y \leq d,\ S_3: X \geq e,\ S_6: X \leq f.$$

En este caso, se ha decidido que $a = c = e = 0$ y que $b = d = f = 1$.

Combinando los semiespacios anteriores, el espacio cerrado (cubo) quedaría definido mediante la intersección de los semiespacios anteriores:

$$C = S_1 \cap S_2 \cap S_3 \cap S_4 \cap S_5 \cap S_6$$

Otro ejemplo, en el que se utilizan semiespacios abiertos diferentes para crear una primitiva, lo encontramos en la figura 16-b.

En este caso, se trata de crear un cilindro cerrado. Para ello, se utiliza un semiespacio cilíndrico y dos planares, orientados según muestra la figura. Las ecuaciones características de estos semiespacios son:

S_1: $x^2 + z^2 - r^2 \leq 0$
S_2: $y \geq 0$
S_3: $y - a \leq 0$, siendo $\boldsymbol{a}$ la altura del cilindro.

El cilindro queda definido por:

$$C = S_1 \cap S_2 \cap S_3$$

C) Registro de las primitivas de semiespacios

Acabamos de ver que el proceso de construcción de las primitivas definidas mediante semiespacios abiertos es similar al utilizado para definir los modelos booleanos, aunque más simple. Por lo tanto, puede decirse que una primitiva de estas características, no es otra cosa que *un modelo booleano básico.*

Como tal, para representar una primitiva se ha de:

Registrar la información geométrica de los semiespacios

Registrar las combinaciones booleanas de los semiespacios

El registro de la información geométrica se reduce a la representación de las ecuaciones características (más los parámetros correspondientes) de los semiespacios que definen la primitiva.

Una forma común de representar semiespacios consiste en hacer copias de *semiespacios base* (equivalentes a las primitivas en los modelos booleanos) los cuales se identifican mediante un código. Cada semiespacio pertenecerá a un determinado tipo y tendrá asociada una lista de parámetros de escalado, que determinarán la forma de la copia del semiespacio base. Además, una matriz de transformación será la encargada de calcular la orientación y localización actual del semiespacio en el sistema de referencia.

Otro planteamiento interesante para el registro algebraico de los semiespacios consiste en utilizar la expresión matricial común a todas de las funciones de orden 2:

Ec. 2

$$f(x,y,z) = |xyz1| \begin{vmatrix} A_{11}A_{12}A_{13}A_{14} \\ A_{21}A_{22}A_{23}A_{24} \\ A_{31}A_{32}A_{33}A_{34} \\ A_{41}A_{42}A_{43}A_{44} \end{vmatrix} \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix}$$

De este modo, todos los semiespacios cuadráticos quedan representados de modo similar, lo que permite la homogeneización de los procesos. Para cada semiespacio se ha de almacenar su matriz de 16 coeficientes A_{ij}.

Ninguno de los planteamientos anteriores es netamente superior al otro, pues ambos tienen sus ventajas y sus inconvenientes, aunque es más común la adopción del primero.

En cuanto al registro de las *combinaciones booleanas,* una posibilidad viene dada por el hecho de que *cualquier expresión booleana se puede expresar como una "suma de productos", que en este caso sería unión de intersecciones*.

Por lo tanto, una primitiva S puede quedar expresada por:

Ec. 3

$$S = \bigcup_i \bigcap_j S_{ij}$$

donde los S_{ij} son los semiespacios individuales.

3.2 Conjuntos de primitivas

Una cuestión vital a la hora de crear un modelador booleano es la de *establecer el conjunto de primitivas de que va a disponer el modelador*.

Se llama *dominio* o *poder expresivo* de un modelador, *a la capacidad que posee para modelar los diferentes objetos.* Cuanto mayor sea el poder expresivo de un modelador, mayores son las posibilidades de modelado.

El dominio de un sistema de modelado no depende del número de primitivas disponibles, sino que viene dado por *la variedad de semiespacios utilizados al construir las primitivas*, del *conjunto de operadores booleanos desarrollados,* y del *total de operaciones de transformación disponibles*.

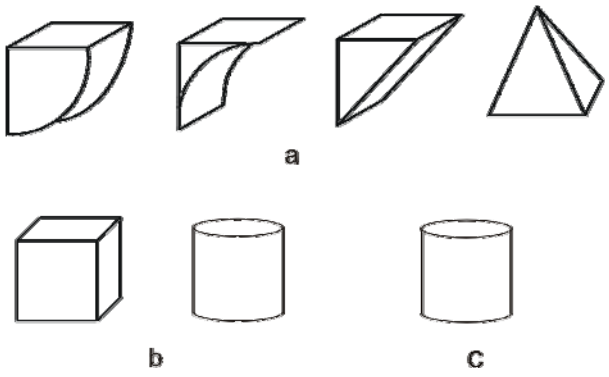


figura 17: tres colecciones de primitivas con el mismo poder expresivo.

En la figura anterior se muestran tres conjuntos de primitivas con el mismo poder expresivo, a pesar de que el número y la variedad de primitivas sean distintos. Esto se debe a que en su construcción se han utilizado exactamente los mismos semiespacios, el planar y el cilíndrico.

De cualquier modo, a igual poder expresivo, un modelador que disponga de una colección mayor de primitivas puede facilitar mucho más el modelado que otro que disponga de pocas primitivas. En otras palabras, aunque dos conjuntos de primitivas tengan el mismo poder expresivo, no necesariamente disponen de la misma *potencia de modelado.* Ésta, se evalúa en función del *número de primitivas diferentes del conjunto,* además del total de operaciones de transformación disponibles.

Desde el punto de vista del modelador, cuanto mayor sea el dominio del conjunto de primitivas utilizado, y más grande sea la potencia de modelado, mayores son los *recursos de modelado* del sistema. En la evaluación de los recursos de modelado, también se han de tener en cuenta las transformaciones lineales y operadores disponibles.

En un dominio dado, el trabajar con conjuntos de primitivas de mayor potencia de modelado normalmente implica la utilización de menos primitivas en la construcción de un modelo dado. Sin embargo, *esto no siempre es cierto*, ya que el total de primitivas a utilizar depende de la geometría del modelo que se crea. Por ejemplo, el conjunto de primitivas de la figura 17-a tiene mayor potencia de modelado que el de la figura 17-c. Sin embargo, si se desea modelar un cilindro, con el conjunto de la figura 17-c se utilizarían menos primitivas que con el de la figura 17-a.

En principio, un sistema de modelado podría permitir al usuario aumentar el dominio y/o la potencia del modelador, dejándole definir sus propias primitivas. Esto implicaría que habría que comprobar, por parte del usuario mismo o del sistema, la validez de estas primitivas, creándose una incertidumbre sobre la integridad de los modelos. La mejor solución es que el sistema de modelado proporcione un conjunto completo de primitivas, de modo que no exista la necesidad de crear otras por parte del usuario. Éste, sólo se encargará de proporcionar los parámetros necesarios para manipular las primitivas, como p. e., la posición, orientación, forma y tamaño.

En la figura 18 se muestra un conjunto operativo de primitivas, más los parámetros de control que cada una requiere (excluidos los de posición y orientación).

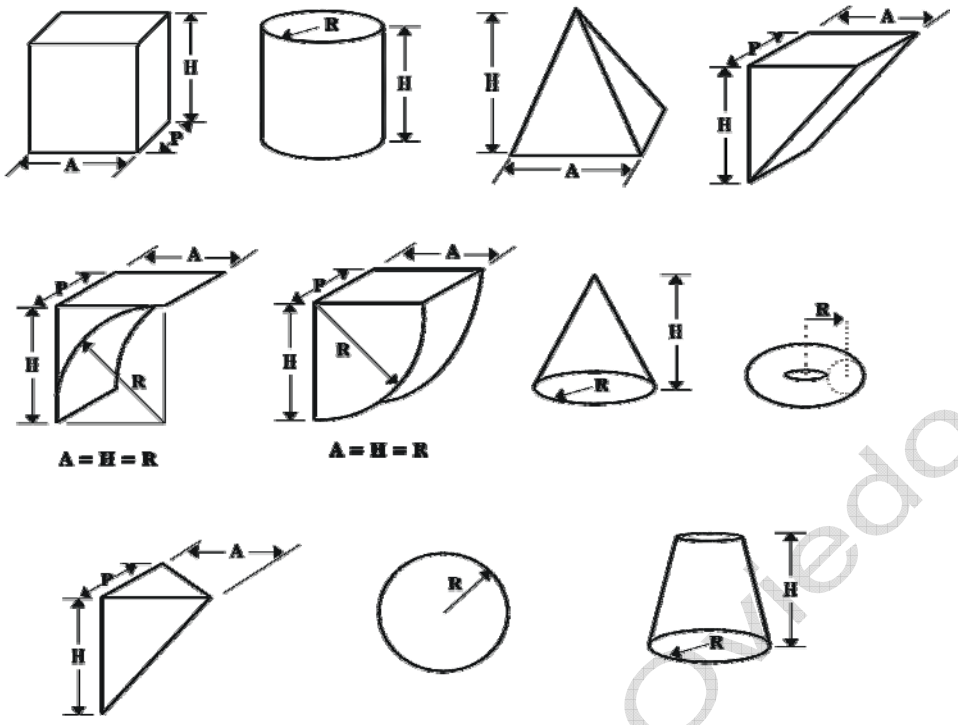


figura 18: Conjunto operativo de primitivas

El dominio del conjunto anterior es de 5 semiespacios diferentes (planar, cilíndrico, cónico, esférico, y tórico), y su potencia de modelado es de 11, dado que dispone de 11 primitivas diferentes.

3.3 Visualización de los modelos booleanos

La visualización de los modelos booleanos varía según sea el método utilizado para buscar la información visual, es decir, los procesos aplicados con el método directo (ray casting o ray tracing) son diferentes a los utilizados en el de rendering standard.

3.3.1 Visualización con ray casting

Para entender mejor cómo se efectúa la visualización de los modelos booleanos mediante ray casting, recordemos brevemente cuál es la información disponible para decidir qué superficies son visibles y cuales no.

Por un lado, en el espacio de referencia se encuentran ***N*** sólidos (primitivas), compartiendo o no un mismo volumen de espacio. Por otro lado, se dispone de un árbol binario CSG, que proporciona una descripción precisa de cómo se combinan booleanamente dichas primitivas.

Como las primitivas presentes pueden ser muchas y las combinaciones entre ellas muy complejas, un planteamiento de "divide y vencerás" parece el más apropiado para resolver el arduo problema de averiguar qué primitivas y puntos de la superficie son visibles, y cuales no.

Para ello se aplica el proceso siguiente:

Se inicia el recorrido del árbol binario *comenzando por el nivel inferior,* localizando el primer par de nodos terminales, la primitiva y la matriz de transformación asociada a cada nodo; en el nodo padre de los nodos ter-

minales se encuentra el operador booleano que se ha de aplicar al par de primitivas.

Una vez que las primitivas localizadas están convenientemente escaladas y situadas en el sistema de referencia, *se efectúa la clasificación del rayo con respecto a cada una de las dos primitivas* (clasificación de E_1 en E_3), o lo que es igual, se averigua qué puntos del rayo se encuentran en las fronteras de las primitivas y cuáles en su interior. Siendo t_e y t_s los valores que toma $\boldsymbol{t}$ (en la ecuación paramétrica del rayo) en los puntos de intersección de entrada y salida (fronteras), el rayo queda clasificado, con respecto a cada primitiva, en el intervalo $[t_e, t_s]$ (un intervalo por primitiva).

Una vez que se dispone de los intervalos $[t_e, t_s]_1$ y $[t_e, t_s]_2$, *se procede a clasificar el rayo, con respecto al modelo resultante de operar booleanamente ambas primitivas*, según sea el operador booleano asignado al nodo padre. En otros términos, si P_1 y P_2 son las primitivas y $<op>$ el operador booleano, siendo $R = P_1 <op> P_2$, se ha de clasificar el rayo con respecto a R. Como pronto veremos, esta clasificación no se realiza directamente sobre el objeto resultante (R), sino indirectamente sobre los intervalos de clasificación del rayo.

Con los resultados obtenidos, se continúa recorriendo el árbol recursivamente, hasta que se llegue a su raíz, momento en el que se sabrá los puntos del modelo final que son intersecados y atravesados por el rayo.

Volviendo al punto tercero, es importante ver que si se crease un nuevo objeto operando booleanamente con las primitivas (proceso bastante costoso), y se procediese a la clasificación del rayo con el objeto resultante (R), *el intervalo o intervalos* $[t_e, t_s]_R$ *que se obtuviesen, serían los mismos que los obtenidos al efectuar directamente la operación* $[t_e, t_s]_1 <op> [t_e, t_s]_2$. *Esta propiedad es la clave que permite que la visualización de los modelos booleanos, mediante ray casting o ray tracing, sea muy sencilla.*

A) Diagramas de Roth

Para facilitar la comprensión del proceso de clasificación de los rayos con respecto a las primitivas y modelos resultantes es de gran ayuda (al menos visualmente) la utilización de los llamados *diagramas de Roth.* Éstos no son otra cosa que la representación recursiva en la recta real $[-\infty, +\infty]$ de los intervalos de clasificación $[t_e, t_s]$.

Para ver cómo son los diagramas de Roth y entender mejor la visualización de los modelos booleanos en general, nada mejor que mostrar algunos ejemplos. Comenzaremos viendo un caso sencillo, en el que sólo intervienen dos primitivas.

Sea $R = P_1 \cdot T_1 \cap P_2 \cdot T_2$ el objeto resultante de la intersección de dos primitivas, tal como muestra la figura 19

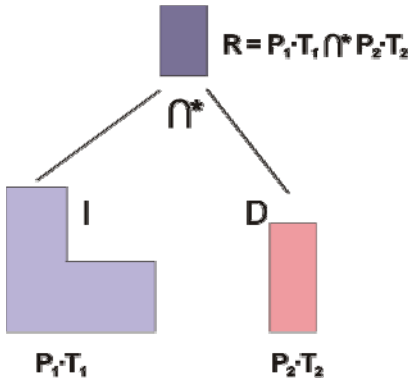


figura 19: modelo booleano

A la hora de visualizar el modelo booleano, el algoritmo de ray casting se va a encontrar, en este ejemplo, con dos sólidos posicionados en el SUR. Suponiendo que la dirección del rayo trazado sea la que se muestra en la figura 20, la clasificación del rayo, con respecto a las primitivas, queda reflejada en un diagrama de Roth como sigue:

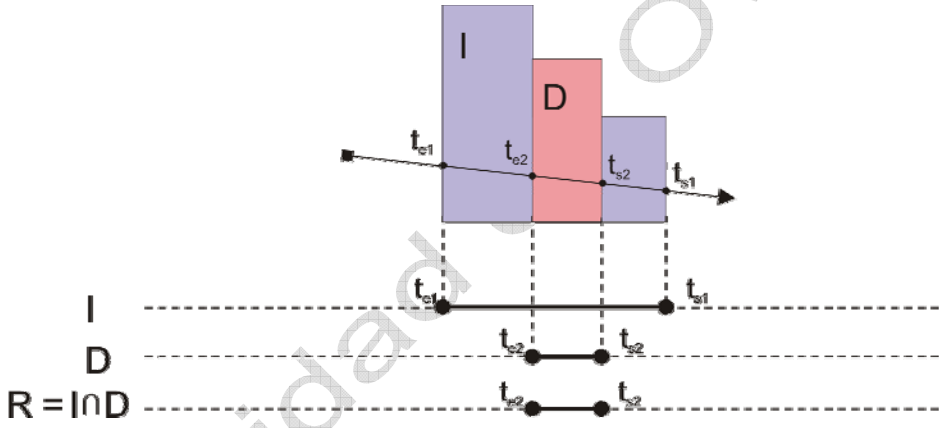


figura 20: diagrama de Roth para un modelo de dos niveles

De los 4 puntos de intersección posibles (t_{e1}, t_{s1}, t_{e2}, y t_{s2}), solamente serán válidos t_{e2} y t_{s2}en la visualización del objeto resultante. Ver que $[t_{e2}, t_{s2}] = [t_{e1}, t_{s1}] \cap [t_{e2}, t_{s2}]$. Por lo tanto, no ha sido preciso calcular el objeto resultante (R en la figura 19), para saber qué puntos serán visibles y las propiedades óptico-geométricas en dichos puntos. En el mismo modelo, si el rayo trazado fuese el que se muestra en la figura 21, la clasificación del rayo quedaría:

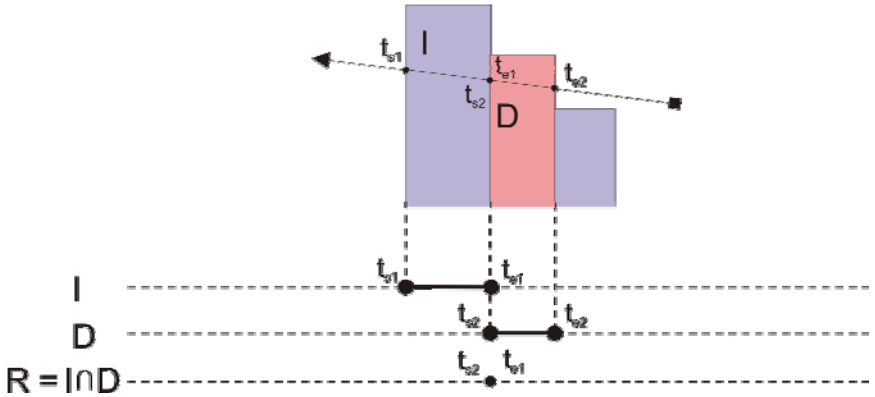


figura 21: diagrama de Roth

En este caso, al calcular la intersección entre los intervalos de clasificación del rayo, el resultado es $[t_{e1}, t_{s2}] = [t_{e1}, t_{s1}] \cap [t_{e2}, t_{s2}]$. Vemos que se produce una singularidad, la cual es fácilmente detectable dado que $t_{e1} \cong t_{s2}$, considerando el posible error de precisión. La detección de una singularidad puede considerarse como un test de validez alternativo en los operadores booleanos regularizados que, como en el ejemplo, implica que el rayo no interseca con el objeto resultante R.

Veamos otro ejemplo (también en 2D), en el que el árbol binario tenga más de un nivel.

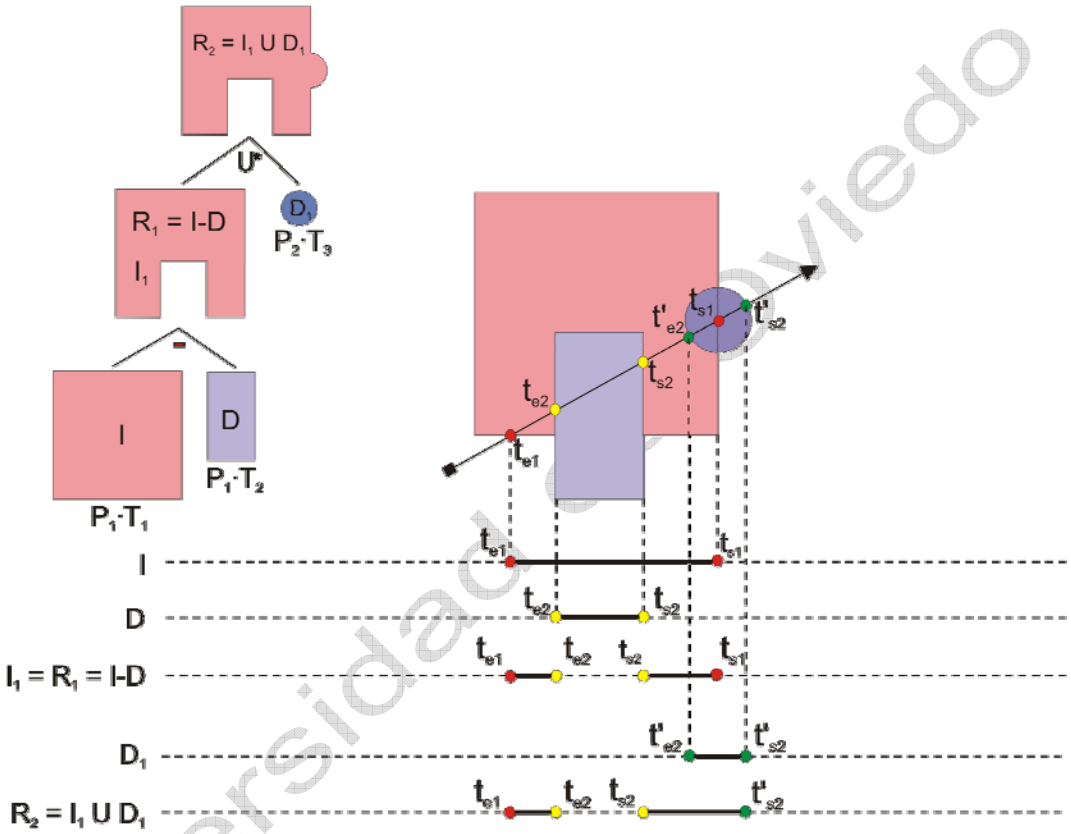


figura 22: diagrama de Roth para un modelo de tres niveles

Como puede verse en el diagrama de Roth de la figura 22, primero se efectúa la clasificación del rayo con las primitivas del nivel inferior ($P_1 \cdot T_1$ y $P_1 \cdot T_2$), y a continuación, se hace otro tanto con R_1, que es igual a ($P_1 \cdot T_1 - P_1 \cdot T_2$). El resultado, pasa a ser la primitiva izquierda del nivel superior. El proceso se repite con R_1 y $P_2 \cdot T_3$, obteniéndose finalmente los segmentos de clasificación del rayo con R_2. Conociendo los valores de ***t*** en los puntos de intersección, y las primitivas donde se localizan, el algoritmo de ray casting puede completar su ejecución. Si el objeto es opaco, el valor de ***t*** más pequeño será el único válido, ya que se trata del punto de intersección más cercano al observador. Si fuese transmisor de la luz, los restantes valores de ***t*** serían necesarios para calcular correctamente la intensidad y color del píxel.

3.3.2 Visualización por el método standard

La visualización standard de los modelos booleanos es bastante más compleja que la visualización mediante el método directo.

Como acabamos de ver, en el segundo caso sólo es necesario efectuar la clasificación rayo-sólido, es decir, averiguar qué puntos del rayo se encuentran fuera, en la frontera y en el interior de los sólidos. Sin embargo, en la visualización standard, no sólo se han de clasificar las aristas (que además pueden ser curvas), sino que también se ha de realizar la clasificación cara-sólido (E_2 en E_3).

A pesar de las dificultades, son comunes los modeladores que efectúan estos procesos, ya que, además de necesarios para la visualización standard, *también se requieren para determinar las características geométricas y físicas de los modelos*, como áreas, volúmenes, densidades, pesos, etc. Recordemos que los modelos booleanos solamente son descriptivos, y no proporcionan directamente este tipo de información. La búsqueda de toda la información implícita se conoce como *evaluación del modelo*.

Los procesos encargados de evaluar los modelos booleanos son los llamados *evaluadores de fronteras,* que incluyen, además de otros procesos, los algoritmos de clasificación. Los evaluadores de fronteras localizan las intersecciones de las superficies de las primitivas, y determinan cuáles son las caras, aristas y vértices válidos para crear el modelo de fronteras. Se ha de tener en cuenta que el total de caras, vértices, y aristas que se generan al intersecar las primitivas, es muy superior al número de caras, vértices, y aristas que pasarán a formar parte del modelo B-rep.

3.4 Evaluadores de fronteras

Dado que la evaluación de fronteras en 2D es más sencilla, veremos en esta dimensión los conceptos y algoritmos básicos utilizados por los *evaluadores de fronteras*.

Localizar las intersecciones de las fronteras y determinar cuáles son las superficies o segmentos activos, es decir, pertenecientes al modelo de fronteras, son las cuestiones clave en la evaluación de las fronteras de los modelos booleanos. Veamos cómo se pueden solucionar estos problemas cuando las primitivas son simples polígonos.

Supongamos que los polígonos son los mostrados en la figura 23-a. Las flechas indican el sentido de recorrido consistente, *que ha de ser igual en ambos polígonos*. Veamos cómo se evalúa dicho modelo, cuando los operadores booleanos son los de unión, intersección y diferencia.

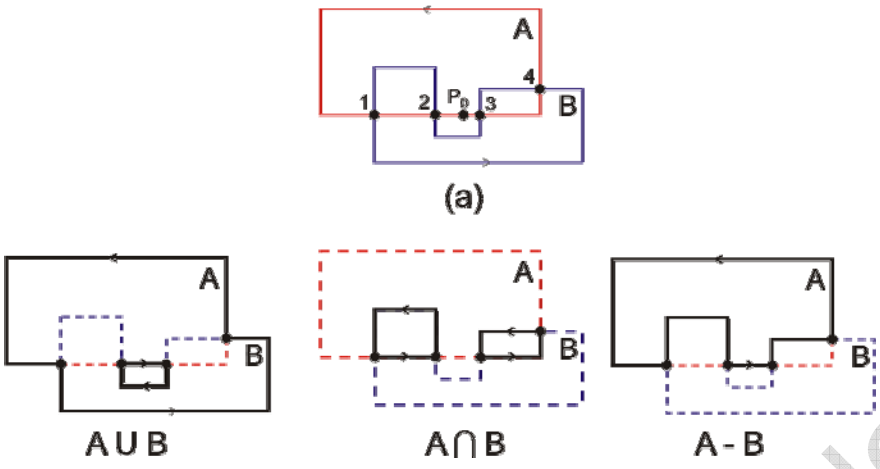


figura 23: ejemplos de evaluación de las fronteras en 2D

Evaluación de A ⋃ B

Localizar los puntos de intersección de las aristas de A y B. En la figura 23-a estos puntos vienen indicados por {1, 2, 3, 4}.

Segmentar las aristas de A y B, tomando como vértices los puntos anteriores. Según el sentido de recorrido, los segmentos formados son: {[1, 2], [2, 3], [3, 4], [4, 1]} en A, y {[1, 4], [4, 3], [3, 2], [2, 1]} en B.

Comenzando por el primer segmento de la lista de segmentos de A (o B), localizar un punto del segmento y ver si pertenece a B. Si dicho punto también es de B, se selecciona el siguiente segmento de A. En el segmento [2, 3] del ejemplo, se localiza el primer punto (P_0) de A que no pertenece a B.

Dibujar el segmento de A, según el sentido de recorrido. En el ejemplo, se pinta el segmento [2, 3] desde el punto 2 al 3.

Cuando se llegue al final del segmento (punto 3), se localiza el segmento de B con el que interseca, dibujando éste en el sentido de recorrido de B. En nuestro caso, el segmento de B encontrado es el [3, 2], que se traza desde el punto 3 al 2.

Finalizado el trazado del segmento de B, en un nuevo segmento de A se localiza un punto que no pertenezca a B, es decir, se repite el proceso a partir del paso tercero.

Continuar el proceso hasta que se acaben el segmento de A.

En el ejemplo, después de trazar los segmento activos [2, 3] y [3, 2] de A y B, respectivamente, de los restantes segmentos de A, el único que no posee puntos en común con B es el [4, 1]. Después de trazarlo, al llegar al punto 1 nos encontramos con el segmento [1, 4] de B[3], por lo que igualmente lo trazamos.

El resultado de la unión de A y B se muestra en la misma figura 23. Ver que se ha generado una frontera interna (agujero). Éstas son fáciles de

[3] El segmento [2, 1] de B no es válido, ya que se debería trazar en el sentido de recorrido inverso.

detectar, ya que en ellas el sentido de recorrido es opuesto al de las fronteras externas.

Evaluación de A ∩ B

El algoritmo de intersección es similar al de la unión, excepto en que sólo se han de trazar los segmentos de A que *tengan puntos en común con B*. El criterio para el trazado de los segmentos de B no varía.

Evaluación de A – B

Finalmente, la evaluación de la diferencia entre los polígonos también se efectúa de forma similar a como se hace la unión, aunque los segmentos de B se han de trazar en sentido contrario al de recorrido. En el ejemplo, primero se traza el segmento [4, 3] (del punto 3 al 4), y después el segmento [2, 1], del punto 1 al 2 (ver la figura 23).

Los evaluadores de fronteras[4] que acabamos de ver, requieren el soporte de varios algoritmos, como p. e. un *localizador de los puntos de intersección,* un *clasificador punto-superficie* que indique si un punto está dentro, en la frontera, o fuera de un polígono dado, y un *trazador de segmentos* que forme bucles y los reorganice.

En 3D, la evaluación de los modelos booleanos es bastante más compleja de lo que acabamos de ver. Además de aristas y puntos, los evaluadores también tienen que clasificar las superficies, determinar dónde se intersecan y dónde se crean o eliminan aristas o vértices. Las nuevas aristas aparecen como consecuencia de la intersección de superficies y los nuevos vértices donde concluyen las nuevas aristas.

Para finalizar, veamos algunos de los principios que facilitan la evaluación de los modelos 3D y la validación de los resultados. Algunos resultarán familiares ya que se basan el los conceptos vistos en el tema anterior. Por ejemplo:

a) Si dos curvas cerradas en el plano se intersecan, *lo harán en un número par de puntos*.

b) Si dos curvas cerradas coplanares *A* y *B* no se intersecan entonces, *si un punto de B está dentro de A, la curva B está dentro de A*.

c) Si una curva cerrada interseca la superficie de una figura tridimensional *lo hará en un número par de puntos*.

d) Si un plano infinito *P* interseca con una superficie cerrada tridimensional *S* entonces la sección del plano ha de consistir en una o más curvas cerradas que no intersecan.

La figura 24 muestra un ejemplo de cada uno de los puntos anteriores.

[4] Ver que a estos mismos algoritmos, si A y B fuesen modelos B-rep, se les llamaría *operadores booleanos*.

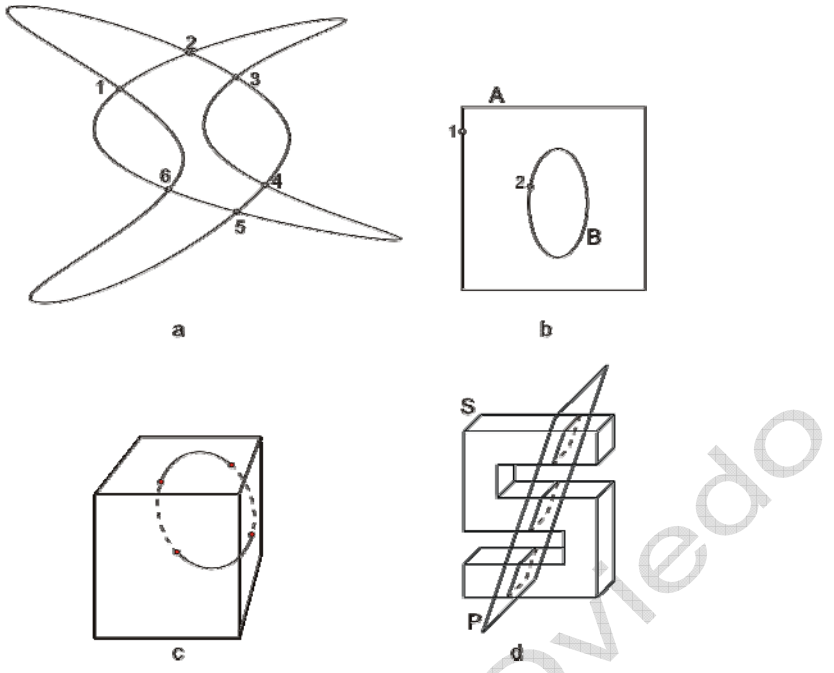


figura 24: conceptos aplicados en la evaluación de las fronteras

e) L*as caras del nuevo sólido (superficies activas) han de ser un subconjunto de las caras de los sólidos que se combinan*[5]. Las caras se pueden modificar, pero no crear. Sin embargo, se pueden crear nuevas aristas y vértices y se puede borrar cualquier tipo de elemento.

3.4.1 Casos problemáticos en la evaluación

Al evaluar las fronteras, existen situaciones que pueden dar lugar a errores si no se tienen en consideración. Por sencillez, vamos a ver varios ejemplos en 2D, aunque siempre es posible encontrar el equivalente tridimensional. Un ejemplo de cada caso puede verse en la figura 25

a) La unión de dos primitivas disjuntas (que no intersecan).

b) La diferencia de dos primitivas disjuntas.

c) La unión de dos primitivas donde una contiene completamente a la otra. Esta operación va a dar como resultado la primitiva que contiene a la otra.

d) La diferencia de dos primitivas donde la primitiva positiva contiene completamente a la primitiva negativa (A – B). Se produce un agujero.

e) La diferencia de dos primitivas donde la primitiva negativa contiene completamente la positiva (B – A).

f) La diferencia de dos primitivas que crean dos o más objetos nuevos.

g) La unión de dos primitivas que son tangentes.

h) La unión de dos primitivas que crean bucles o cavidades interiores.

---

[5] Ver que esta afirmación es una consecuencia directa de que los puntos de frontera puedan llegar a ser puntos de interior, pero no a la viceversa

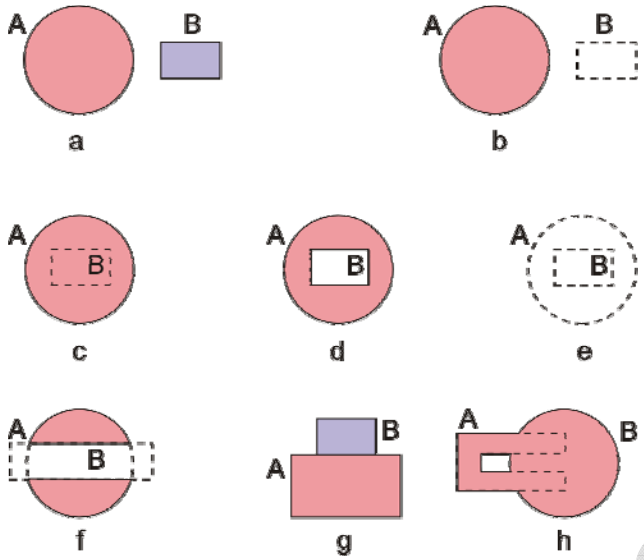


figura 25: casos problemáticos en la evaluación de las fronteras

Aunque no se muestran en la figura anterior, las operaciones $A \cup A$ y $A - A$ también pueden causar problemas.

4. Modelado de fronteras versus modelado booleano

En la siguiente tabla se resumen las ventajas e inconvenientes de estos dos esquemas:

Esquema	Ventajas	Inconvenientes
CSG	La estructura de datos de un modelo booleano es simple y la cantidad de datos reducida. La gestión interna de la estructura de datos es por lo tanto fácil.	Las operaciones disponibles para la creación y modificación de los sólidos están limitadas. Generalmente no es fácil implementar otras operaciones distintas a las operaciones booleanas.
	Si las primitivas son válidas, queda garantizada la integridad de los modelos.	La visualización por el método standard es costosa.
	Las modificaciones globales de los modelos booleanos son fáciles. Basta con cambiar los parámetros asociados a las primitivas (tamaño, forma, situación y orientación).	Las transformaciones locales son difíciles de implantar en los entornos interactivos.
	Mayor facilidad de descripción que en los modelos B-rep, aunque se requiere experiencia ya que normalmente las superficies buscadas están implícitas en las primitivas.	El poder expresivo de los esquemas de modelado booleano es inferior al de los B-rep.
	Cualquier modelador que utilice *CSG* como esquema de representación interno, puede suministrar un modelo *B-rep*. El sistema puede soportar por tanto una amplia variedad de aplicaciones.	Al ser modelos no evaluados, se precisan los evaluadores de fronteras, que requieren mucho tiempo de cálculo. Si embargo, como los pasos básicos son muy simples, son buenos candidatos para ser implantados en VLSI

B-Rep	Mayor poder expresivo que el CSG, pues el rango de primitivas posibles es más amplio.	Estructura compleja y gran cantidad de información.
	Fáciles de visualizar mediante el método directo y el standard	La comprobación de la integridad de los modelos es complicada.
	Facilidad en la ejecución de modificaciones locales. El usuario dispone de más operaciones que en el CSG.	Algunos operadores, como los booleanos, pueden generar modelos abiertos (no íntegros), al operar sobre modelos cerrados.
	Al ser modelos evaluados, se tiene fácil acceso a la información geométrica y topológica en los B-rep poliédricos. Los algoritmos se complican mucho cuando se trabaja con superficies curvas	La descripción directa de los modelos es muy tediosa. Se han de utilizar herramientas de ayuda a la descripción o bien conversores desde otros esquemas a B-rep
	La visualización en modelo alámbrico es inmediata.	Es posible la pérdida de modelos originales, al operar incorrectamente con ellos.

tabla 6: Comparación de las ventajas e inconvenientes de los esquemas de representación CSG y B-Rep.

Se advierte fácilmente que lo que resulta fácil en uno es complicado en el otro. Por tanto, como veremos en el próximo tema, es lógico combinar ambos esquemas para aprovechar sus ventajas y paliar sus inconvenientes.