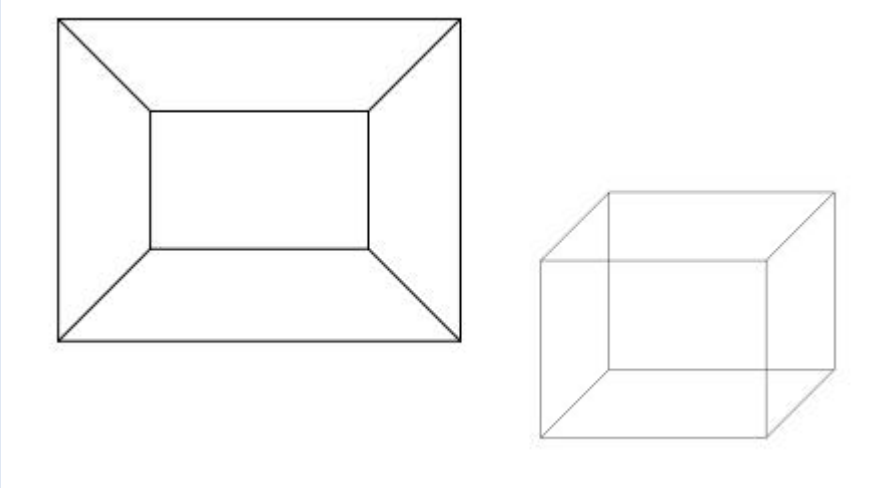
Modelación de objetos 3D
Parte II:
Modelación de sólidos

Contenido

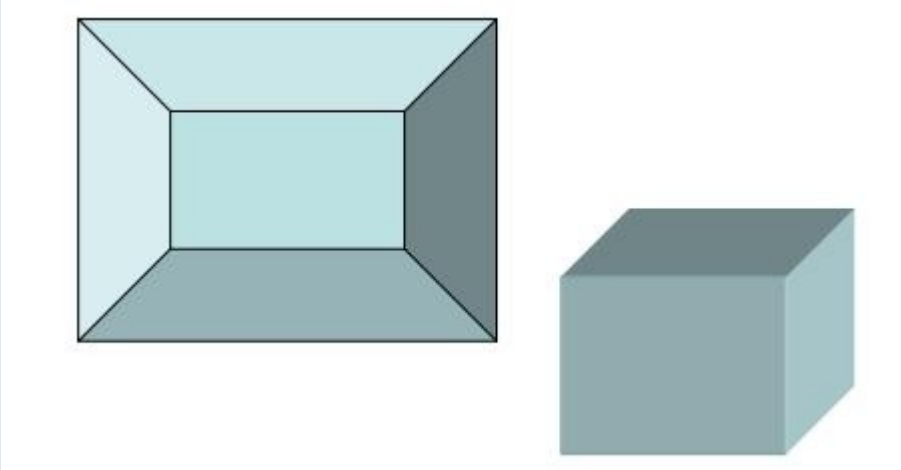
- Conceptos: Qué es un sólido?
- Propiedades de los modelos de sólidos
- Geometría sólida constructiva (Constructive solid modeling)
- Representación por el borde (B-rep)
- Representación de barrido (Sweep-rep)
- Descomposición espacial en celdas

Conceptos: qué es un sólido?

- Sólido es un objeto en 3D en el que está claramente definido su interior y exterior
  - Clasificar si punto (x,y,z) está en el interior, borde o exterior del objeto
  - Calcular propiedades tales como volumen y momentos de inercia
  - Calcular interferencias o detectar colisiones



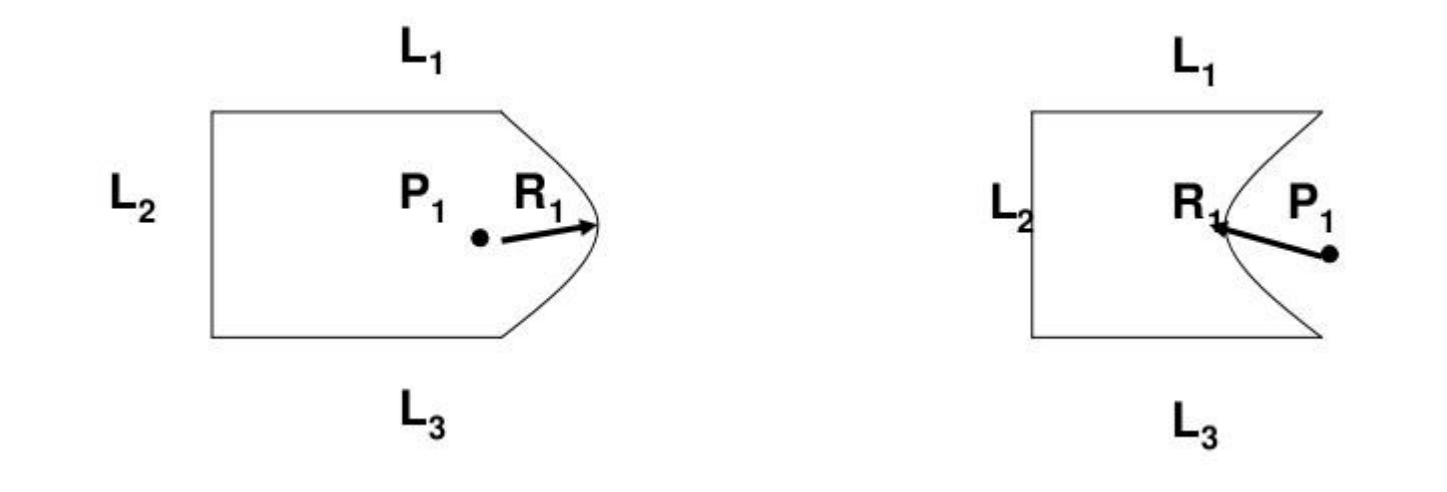
Modelo wireframe (solo arcos y vértices)



Modelo sólido

Conceptos: geometría versus topología

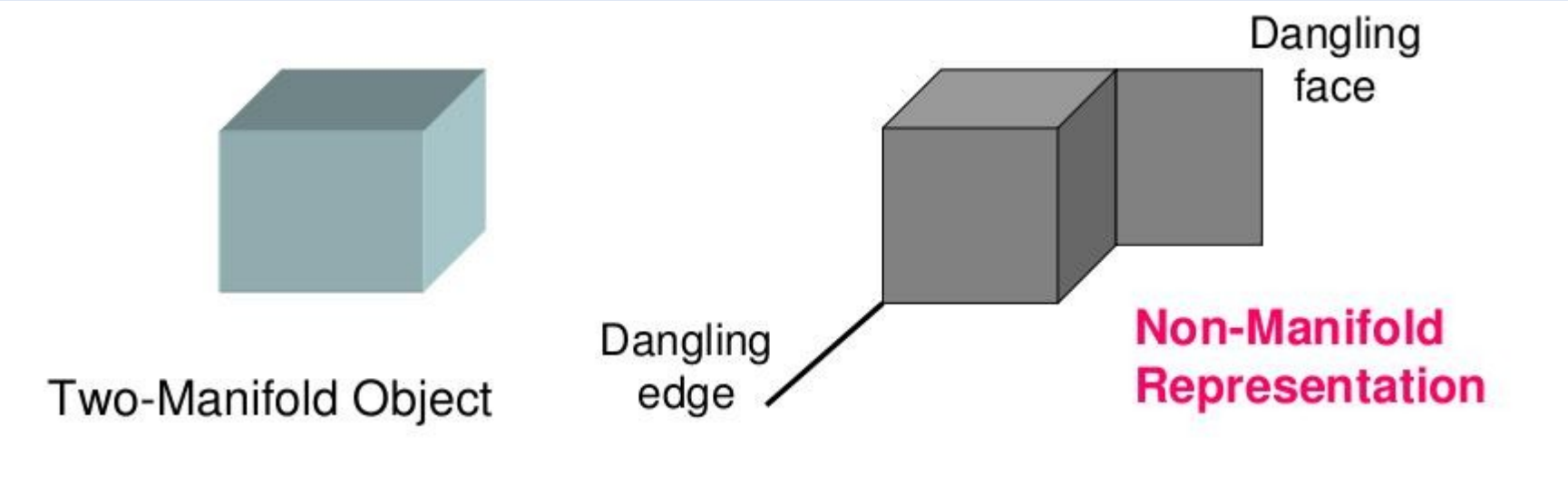
- Geometría: métricas, dimensiones del objeto y su posición en el sistema de coordenadas usado
- Topología: información combinatoria como conectividad, asociatividad y vecindad. Información invisible



Misma topología, distinta geometría

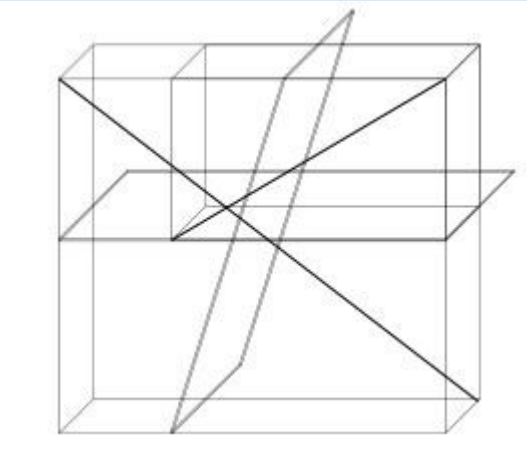
Conceptos: manifold versus non-manifold

- Two-manifold: objetos bien definidos, cerrados y homomórficos a una bola topológica
- Non-manifold: objetos descritos por partes wireframe, no cerradas y sólidos



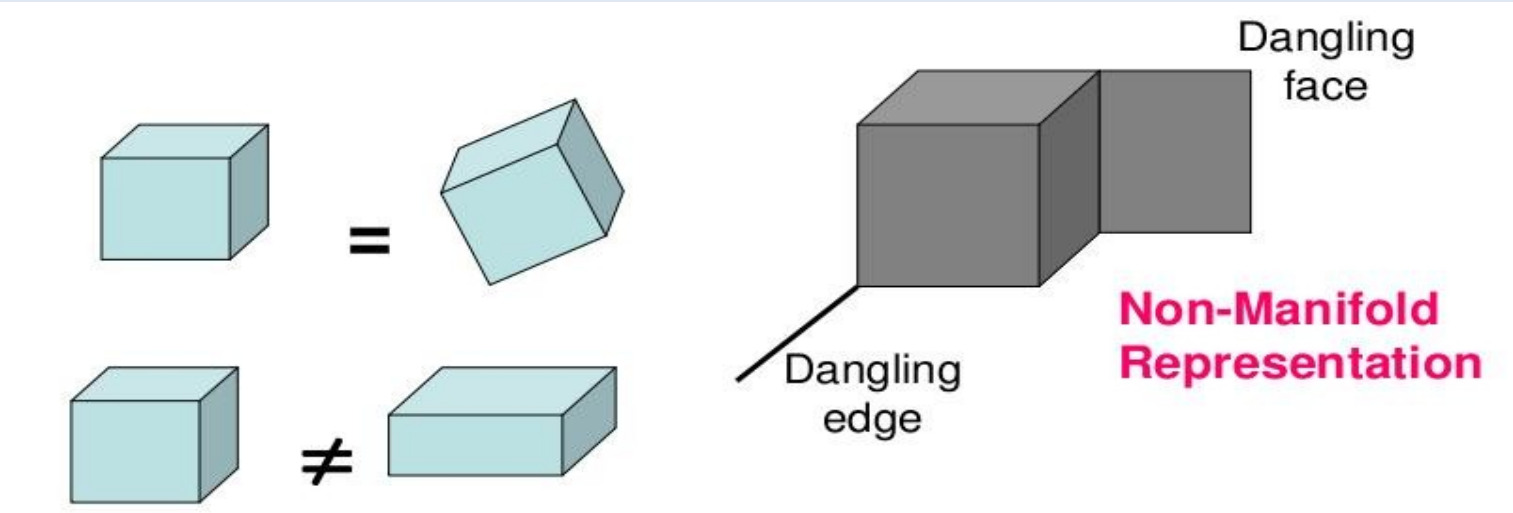
Modelos wireframe

- Desventajas:
  - Ambiguos
  - Subjetivos a la interpretación humana
  - Objetos complejos con muchos arcos son confusos
  - No permiten calcular propiedades como masa, detectar colisiones, etc



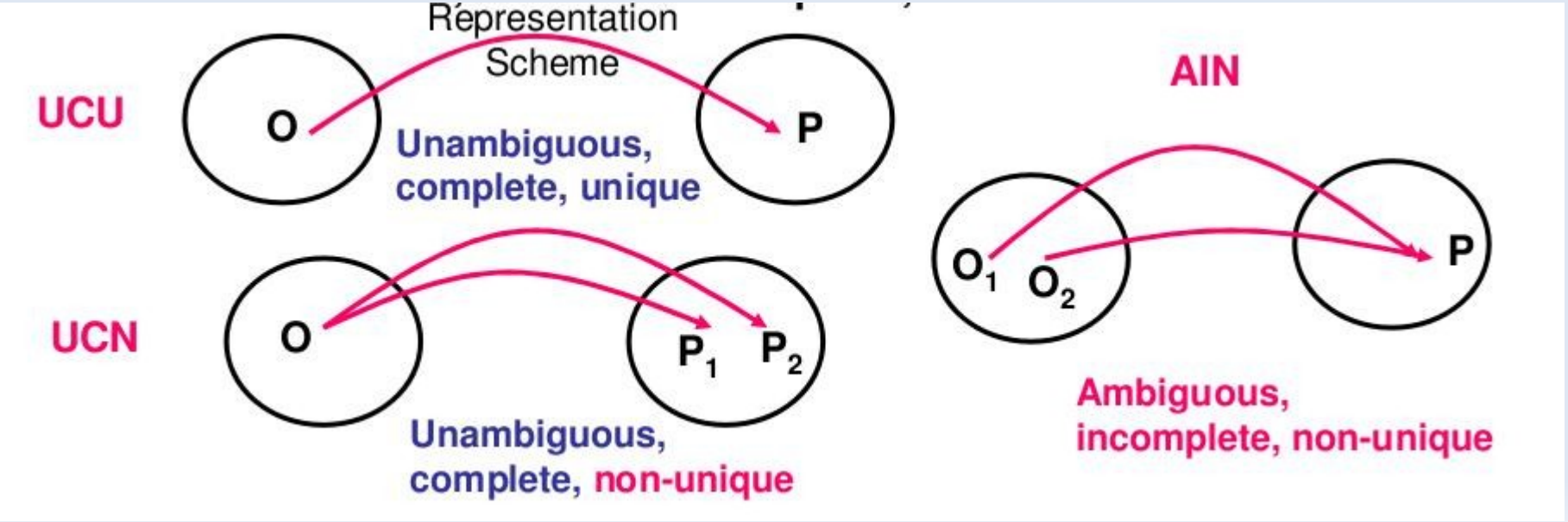
Propiedades de los modelos de sólidos

- Rigidez: la forma del sólido es invariante con respecto a cambios de posicion y orientación
- Homogéneos tridimensionalmente: el borde del sólido debe estar en contacto con el interior y exterior. No se permiten arcos “dangling” ni caras “dangling”
- Finitos y descripción finita: el sólido es acotado y se describe con una cantidad finita de información
- Cerradura bajo movimientos rígidos y operaciones booleanas regularizadas: movimientos sobre los objetos y operaciones booleanas deben producir objetos válidos



Propiedades de los modelos de sólidos

- Cada modelo debe establecer:
  - Dominio: el tipo de objetos que puede ser representado (alcance)
  - Validez: debe ser un sólido bien definido
  - Completitud: datos suficientes para la descripción
  - Unicidad: importante para comparar si dos objetos son iguales
  - No ambiguedad: representar sólo un objeto

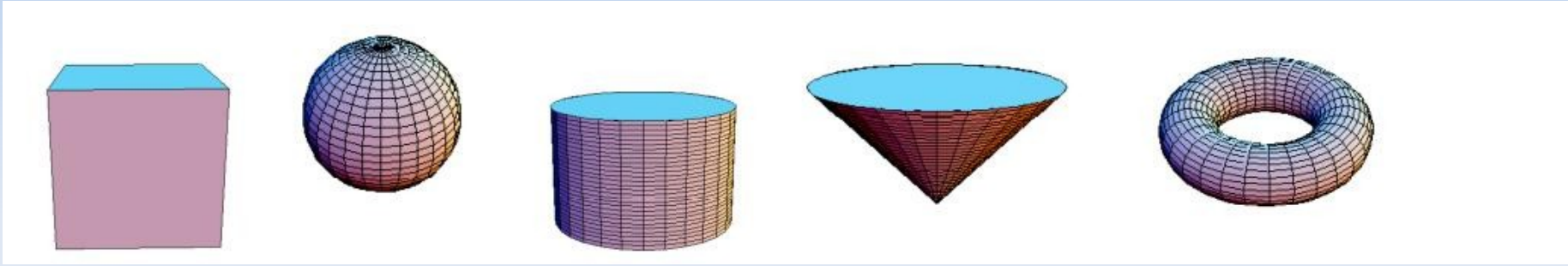


Modelos de sólidos

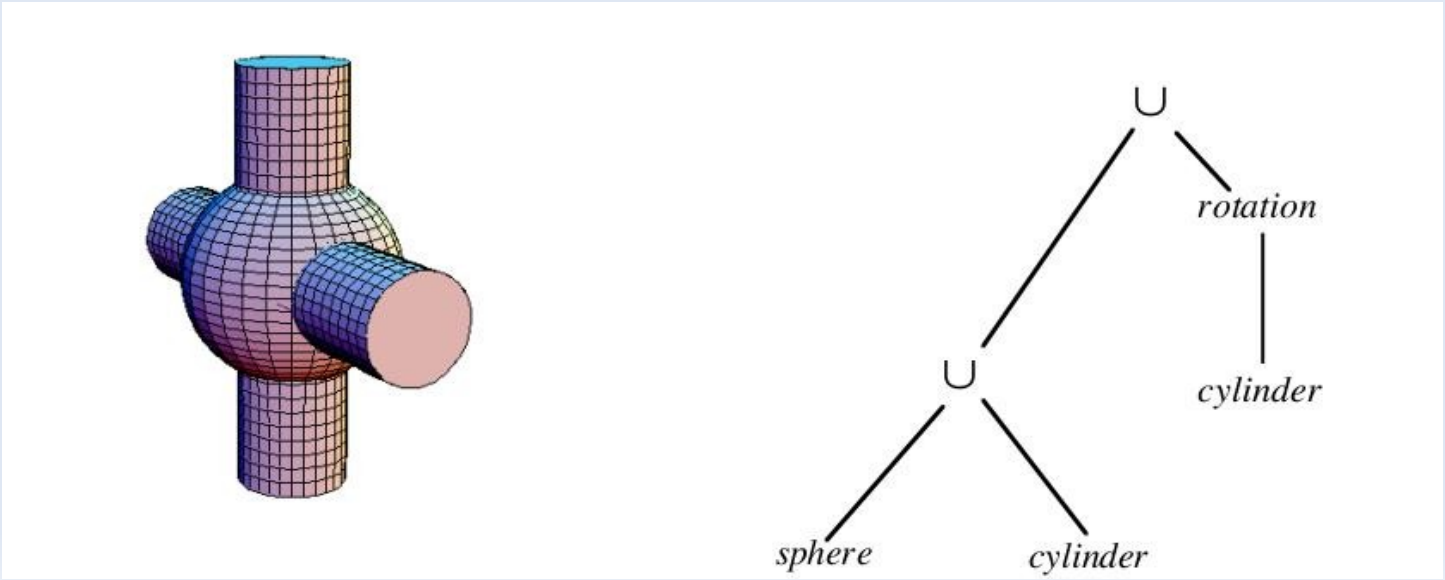
- Geometría sólida constructiva (Constructive solid modeling CSG)
- Representación del borde (b-rep boundary representation)
- Representación de barrido (sweep representation)
- Descomposición espacial en celdas (cell decomposition)
- ...

Geometría sólida constructiva

- Sólidos básicos: cajas, esferas y cilindros, entre otros

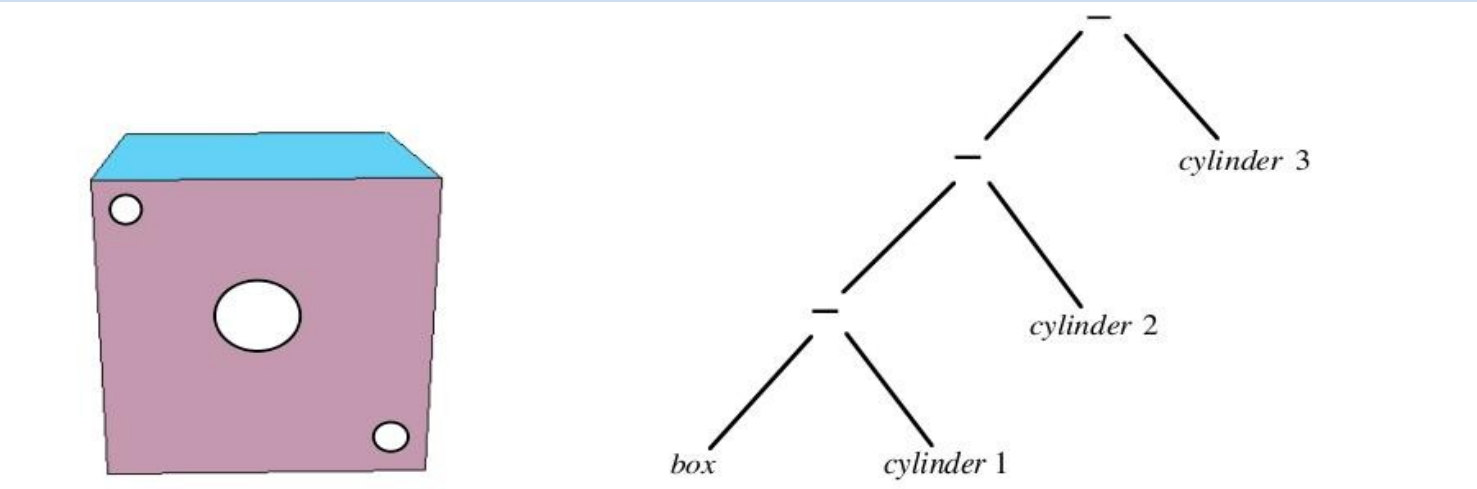


- Construir sólidos más complejos a partir de sólidos básicos aplicando operadores booleanos (unión, intersección y diferencia)



Geometría sólida constructiva

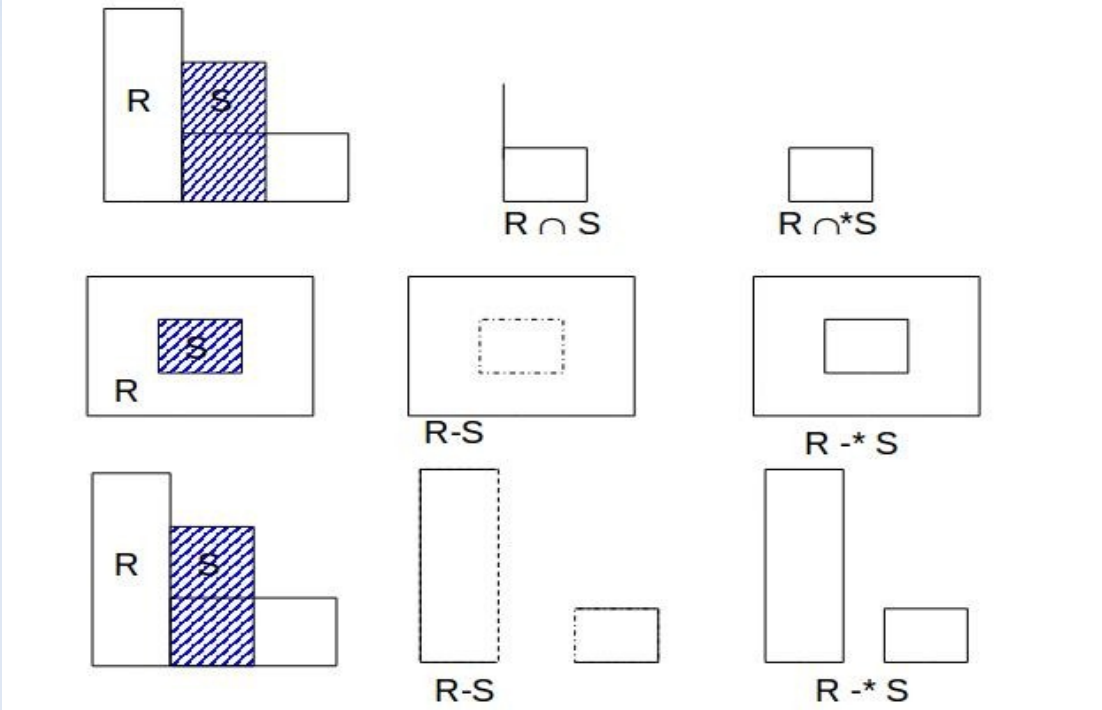
- Creando túneles a través de un bloque usando el operador resta:



- El objeto se almacena en árboles binarios llamado CGS trees: nodos internos almacenan operadores o transformaciones y nodos hoja los objetos primitivos

Geometría sólida constructiva

- Uso de operadores regularizados para mantener objetos homogéneos en la dimensión que se modelan
  - En 2D

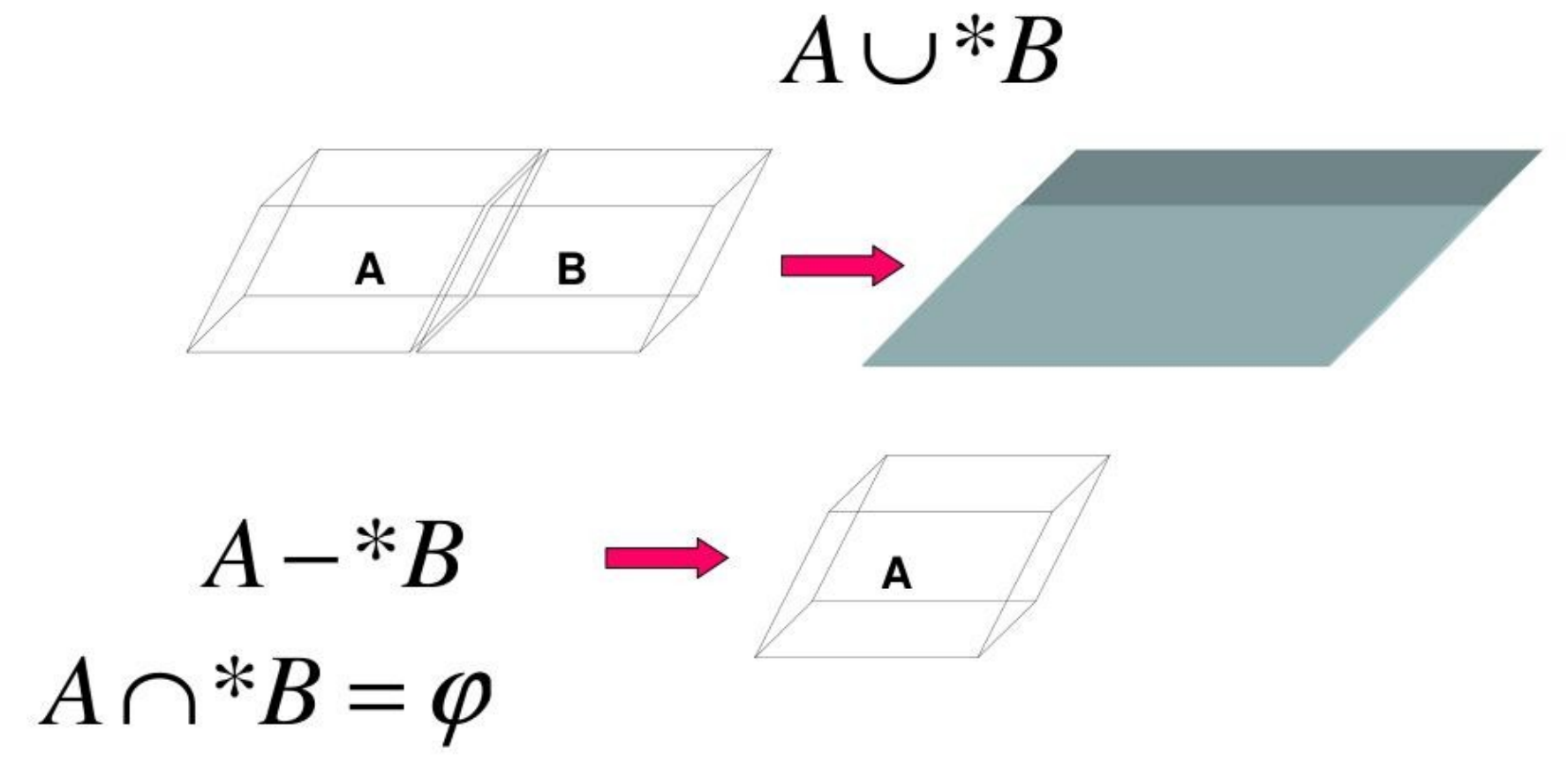


Operadores normales

Operadores regularizados

Geometría sólida constructiva

- En 3D

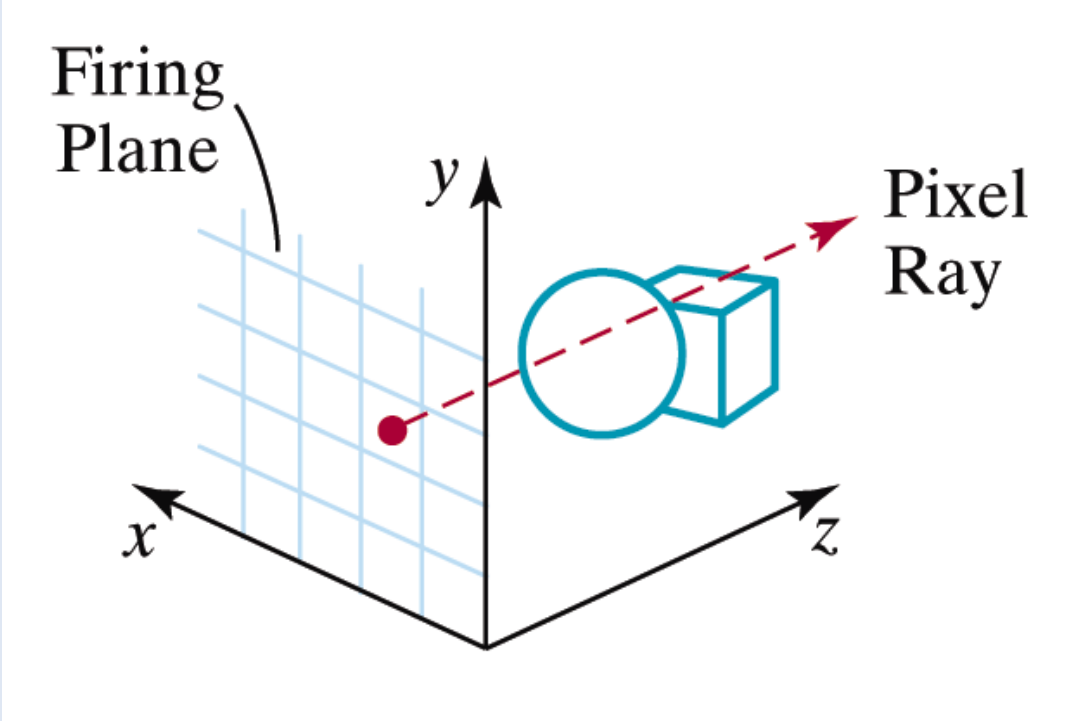


Geometría sólida constructiva

- Recordemos que el objeto se almacena en árboles binarios llamado CGS trees: nodos internos almacenan operadores o transformaciones y nodos hoja los objetos primitivos
- Si no tenemos la representación explícita de la superficie, cómo podemos calcular su volumen? Cómo podemos saber que partes serán visibles?
  - Se calcula la intersección de un rayo (línea) con el sólido
    - Notar que si un rayo intersecta una primitiva no necesariamente intersectará el sólido resultante pues ésta puede ser restada del sólido
  - Para saber qué punto es visible, de los segmentos interiores se obtiene el punto más cercano al punto de vista (más detalles cuando veamos algoritmos de ray tracing)

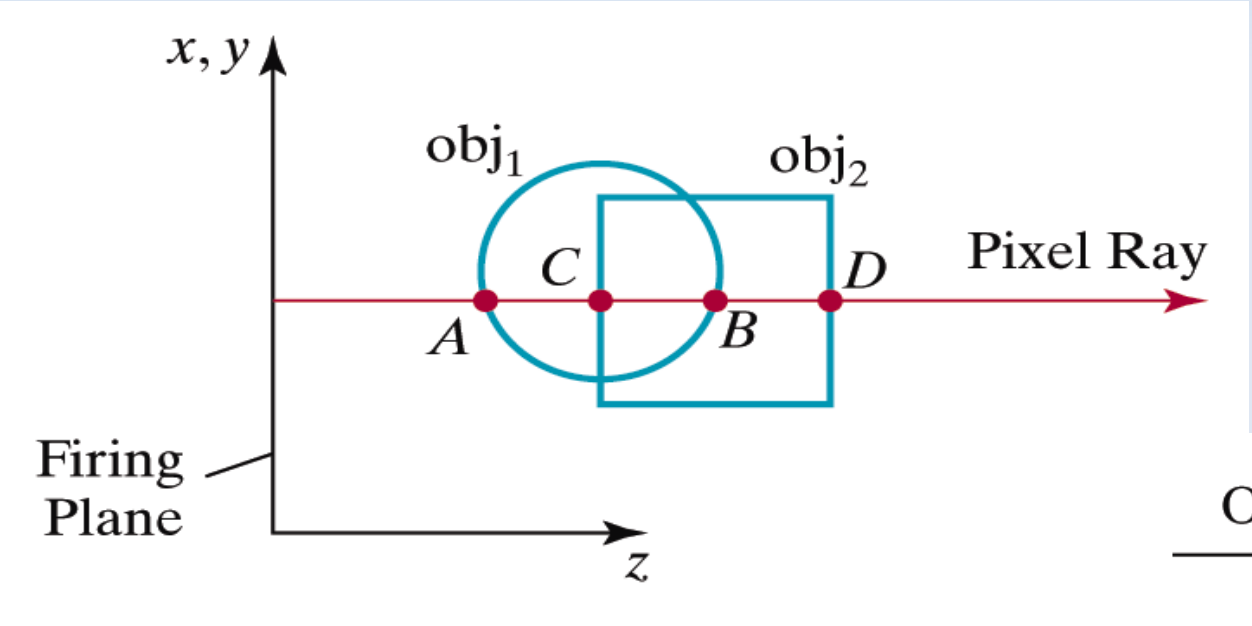
Geometría sólida constructiva

- Cómo calcular el volumen?
  - Técnica de ray casting: se disparan rayos equi-espaciados, emanando por ejemplo del plano xy (firing plane)en la dirección z



Geometría sólida constructiva

- Calcular las intersecciones con cada primitiva y ordenarlas con respecto al firing plane
- Los límites de la superficie de cada objeto compuesto es determinado por el conjunto de operaciones especificado



Operation	Surface Limits
Union	A, D
Intersection	C, B
Difference ($obj_2 - obj_1$)	B, D

Geometría sólida constructiva

- En qué sistema de coordenadas se calculan las intersecciones?
  - Recordar que cada primitiva está definida en sus propias coordenadas (coordenadas de modelamiento)
  - Su posición en WC es determinada por las transformaciones de modeling definidas para lograr que los objetos primitivos se superpongan
  - La transformación inversa es aplicada al rayo para llevarlo a las coordenadas de modelamiento y allí se calcula la intersección con los objetos primitivos individualmente

Geometría sólida constructiva

- Finalmente, cómo calcular el volumen?
  - Aproximar área de cada pixel en el firing plane por un cuadrado pequeño A_{ij}
  - El volumen a lo largo del camino del rayo originado en el pixel (i,j) es

$$V_{ij} \approx A_{ij} \Delta z_{ij}$$

    donde Δz_{ij} es la profundidad del objeto a lo largo del rayo
  - El volumen aproximado total es:

$$V \approx \sum V_{ij}$$

Geometría sólida constructiva

- Otro método para calcular el volumen? El método de Monte Carlo
  - Método estocástico basado en técnicas probabilísticas
    - Encerrar el sólido en un cuboide (bounding box más pequeña que lo rodea)
    - Generar puntos aleatorios, uniformente distribuidos dentro del cuboide: algunos caeran dentro del sólido y otros fuera
    - La probabilidad que un punto caiga dentro del sólido es igual a la razón entre el volumen del sólido y el volumen del cuboide

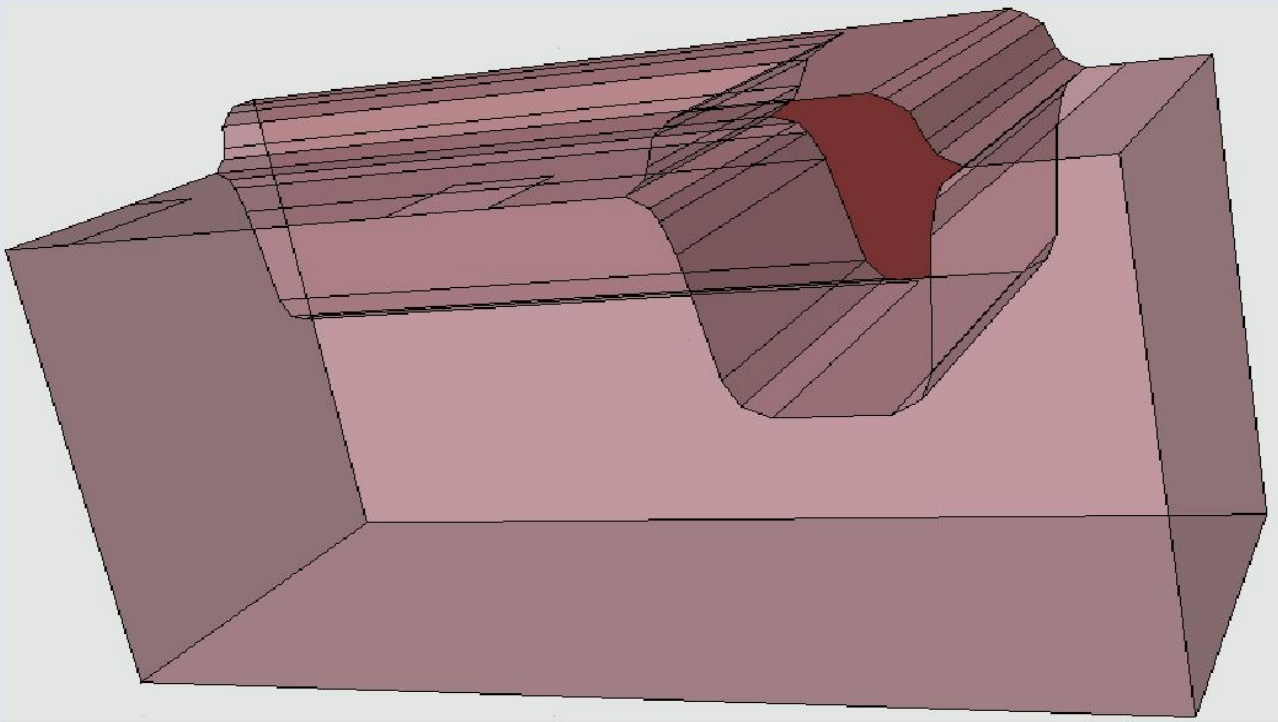
$$\frac{Número\ \ puntos\ \ dentro\ \ del\ \ sólido}{Número\ \ puntos\ \ generados} \approx \frac{volumen(sólido)}{volumen(cuboide)}$$

Geometría sólida constructiva

- Ventajas:
  - Representación compacta
  - Estructura de datos es robusta
  - La forma de crear los objetos basados en los operadores booleanos intersección, resta y unión es simple
- Desventajas:
  - Relaciones de adyacencia no están explícitas
  - No permite acceso directo a los vértices y arcos del sólido
  - Su representación no es única (no es fácil determinar si dos representaciones representan el mismo sólido)

Representación por el borde (B-rep)

- Almacena la geometría y topología del sólido
  - Geometría modela posición, tamaño y orientación
  - Topología modela adjacencia y conectividad



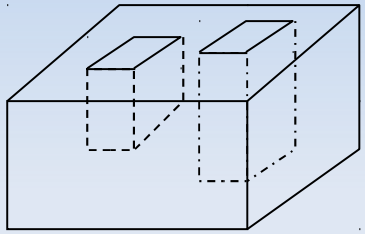
  - CGS: qué almacena?

Representación por el borde (B-rep)

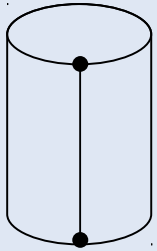
- Interior del objeto encerrado por un conjunto de caras orientables
  - Orientación (vector normal) permite distinguir dos lados de la superficie
- Ley de Euler: chequea validez topológica de la B-rep
  - V-E+F = 2 (para poliedro simple)
  - V-E+F -H = 2(C-G)
    - V = número de vértices
    - E = número de arcos
    - F = número de caras
    - H = número de cavidades en las caras
    - G = número de túneles
    - C = número de cuerpos distintos
  - Ref: O'Rourke, Computational geometry in C. pp105. 1998.

Representación por el borde (B-rep)

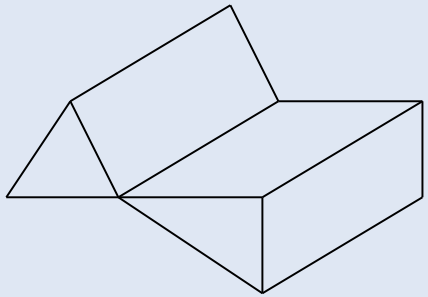
- Aplicando la formula de Euler:
  - Cuboide con un tunel y una cavidad (no atraviesa el cuboide)



  - Cilindro



  - Objeto Non-manifold



Representación por el borde (B-rep)

- Cómo obtener la representación del borde?
  - Especificación por el usuario de cada vértice y cara (es muy lento, es fácil equivocarse, …)
  - Generar el borde a partir de una especificación CSG (esto lo proveen los modeladores de sólidos)
  - Generar el borde a partir de una secuencia de imágenes 2D (algoritmo: marching cubes .. más adelante)
- Qué información y cuánta almacenar?
  - Mínimo: vértices y caras
    - Muy ineficiente para encontrar relaciones adyacencia

Representación por el borde (B-rep)

- Estructuras de datos para un poliedro basado especificado por un conjunto de polígonos (pseudo-java). Qué queremos consultar?
  - Encontrar todos los arcos que rodean una cara
  - Encontrar todas las caras adyacentes a un cara
  - Encontrar todos los vértices de una cara
  - Encontrar todos los arcos que pasan por un vértice
- Qué permite consultar la siguiente estructura de datos en O(1)?

```
class Point{
    float x,y,z;
    …
}
```

```
class Edge{
    int endpoints[2];
    int faces[2];
    …
}
```

```
class Face{
    int  vertices[];
    int  edges[];
    int  n_vertices;
    ...
}
```

```
class Polyhedron{
    int faces[];
    int n_faces;
    …
}
```

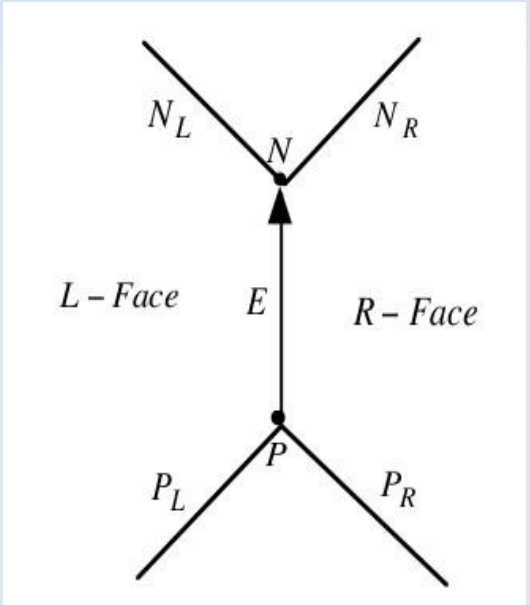
```
class Points{
    Point points[];
    int n_points;
    ...
```

```
class Edges{
    Edge edges[];
    Int n_edges;
    ....
}
```

```
class Faces{
    Face faces[];
    int n_faces;
    …
}
```

Representación por el borde (B-rep)

- Estructura de datos clásica: Winged-edge. Es la estándar para representar la topología con un compromiso razonable entre espacio y tiempo
  - Cada arco "apunta a":
    - Vértices: Previo (P) y Next (N) (orientación)
    - Caras: Left (L) y Right(R)
    - Arcos: P_R, N_R y P_L, N_L
  - Cada cara y vértice apunta a un arco que lo contiene




```
class Point{
    float x,y,z;
    int edge;
    …
}
```

```
class Edge{
    int previo,next;
    int left, right;
    int pr,nr,pl,nl;
    …
}
```

```
class Face{
    int  edge;
    …
}
```

```
class Polyhedron{
    int faces[];
    int n_faces;
    …
}
```

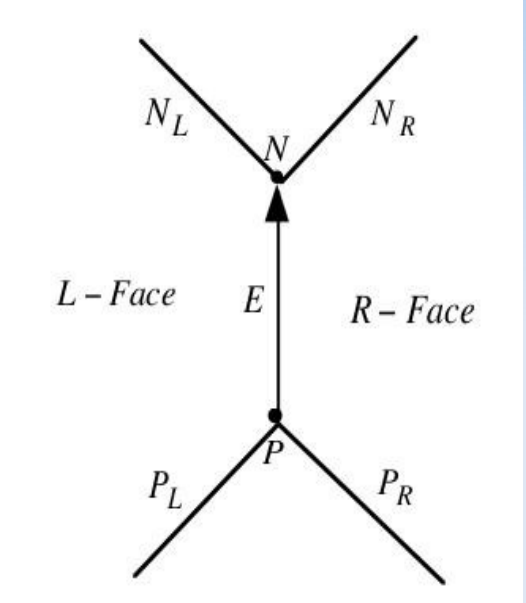
```
class Points{
    Point points[];
    int n_points;
    ...
```

```
class Edges{
    Edge edges[];
    Int n_edges;
    ....
}
```

```
class Faces{
    Face faces[];
    int n_faces;
    …
}
```

Representación por el borde (B-rep)

- Usando Winged-edge, se puede saber qué arcos comparten un mismo vértice sin recorrer todos los arcos?
  - Input: índice al vértice




```
class Point{
    float x,y,z;
    int edge;
    …
}
```

```
class Edge{
    int previo,next;
    int left, right;
    int pr,nr,pl,nl;
    …
}
```

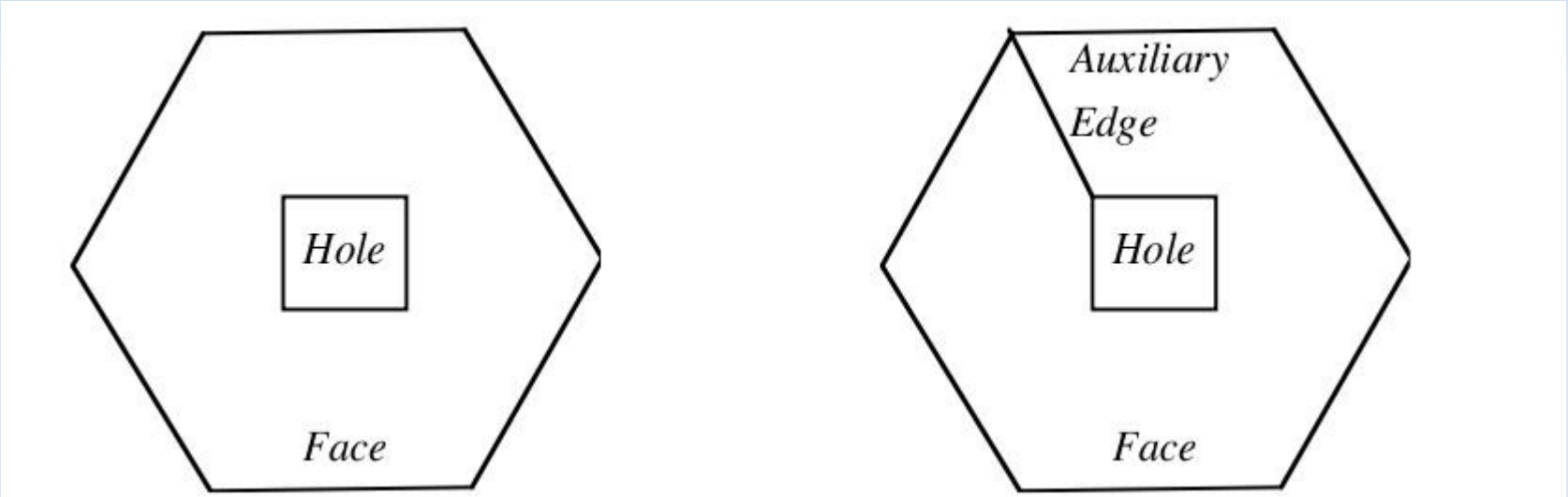
```
class Face{
    int  edge;
    …
}
```

  - Algoritmo:

```
Obtener arco e que contiene el vértice
f = left, f_final = right
while(  f != f_final ) {
    Buscar en pl,nl, otro arco e de f que contiene al vértice
    Buscar la cara vecina de e distinta de f y almacenarla en f
}
```

Representación por el borde (B-rep)

- Se puede usar winged-edge con geometrías con hoyos? No
  - Se inserta un arco auxiliar. Este arco auxiliar tiene la misma cara por ambos lados



Representación por el borde (B-rep)

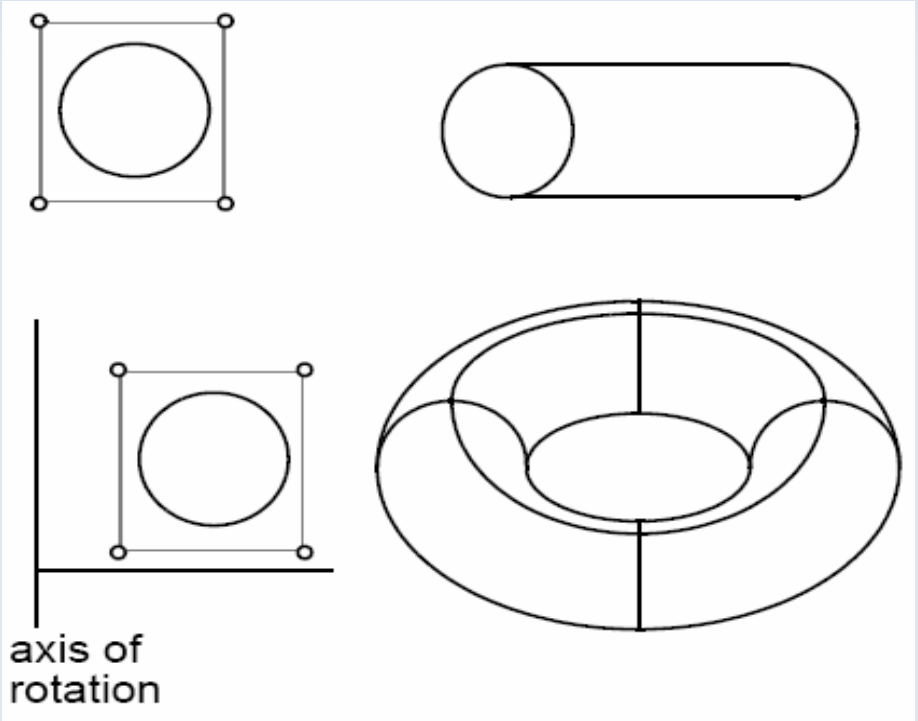
- Cómo es el cálculo de volumen en una B-rep?
  - Usando ray casting? si
  - Usando Montecarlo? si
  - De manera exacta?
    - Triangular los polígonos de la superficie
    - Orientar caras de tal manera que todas las normales esten hacia afuera
    - Elegir un punto P cualquiera del espacio fuera del poliedro
    - volumen = 0;
    - For each cara f del poliedro
      - volumen += Determinante (f0,f1,f2,P)/6
  - Se basa en la suma del volumen de tetraedros. La cuarta columna de la matriz contiene 1s.

Representación por el borde (B-rep)

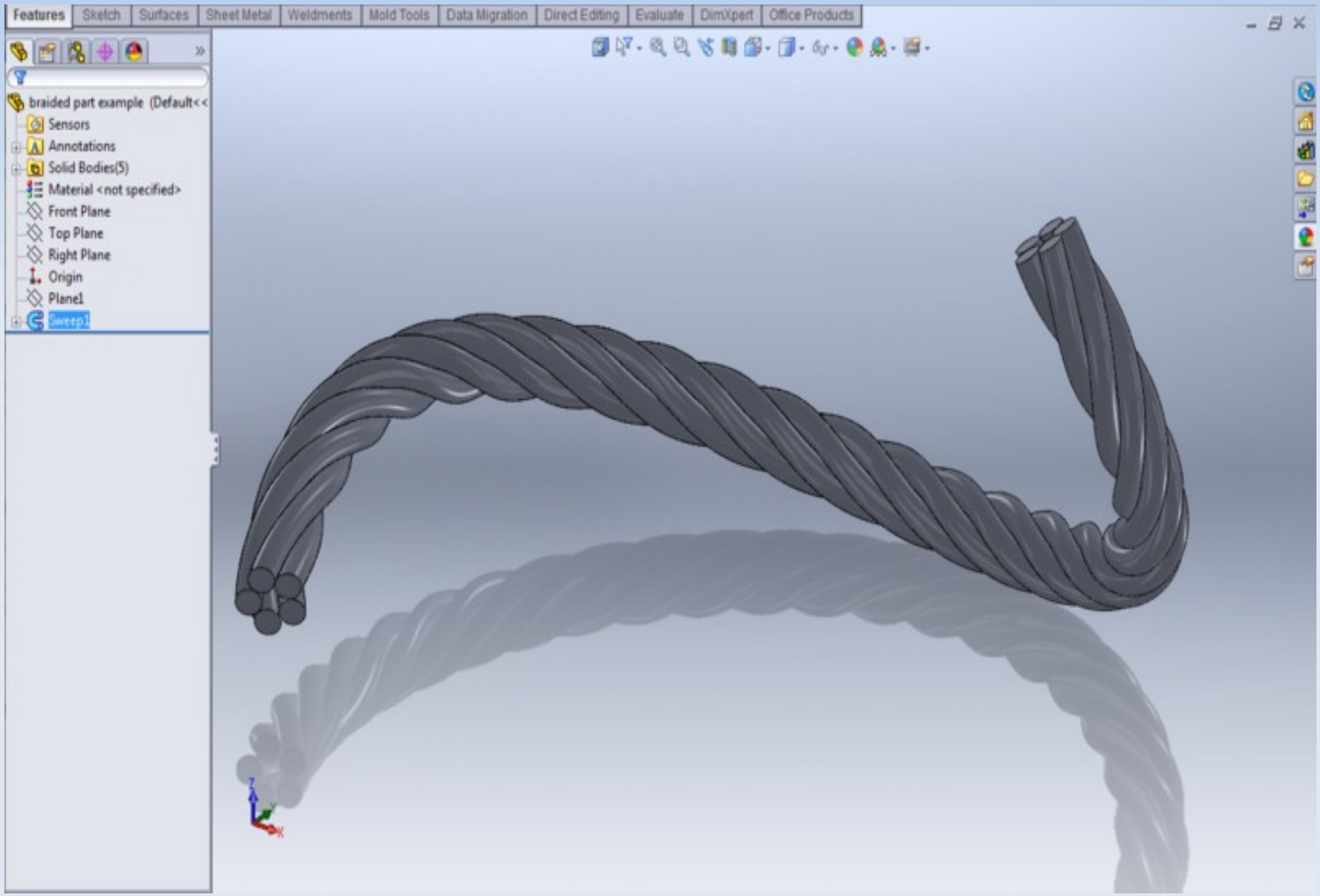
- Ventajas:
  - La información sobre adyacancia (vecindades) está explícita en la representación
  - Representación única
- Desventajas:
  - Operadores booleanos son difíciles de implementar
    - Se necesita calcular intersecciones de manera explícita y actualizar la topología en la estructura de datos
    - Problemas de robustez en el cálculos de operaciones matemáticas usando punto flotante

Representación de barrido (sweep rep)

- Objeto 3D se crea a partir de hacer un barrido con un objeto 2D
  - Rotar con respecto a un eje (volumen de revolución)
  - Desplazar en una cierta trayectoria o curva

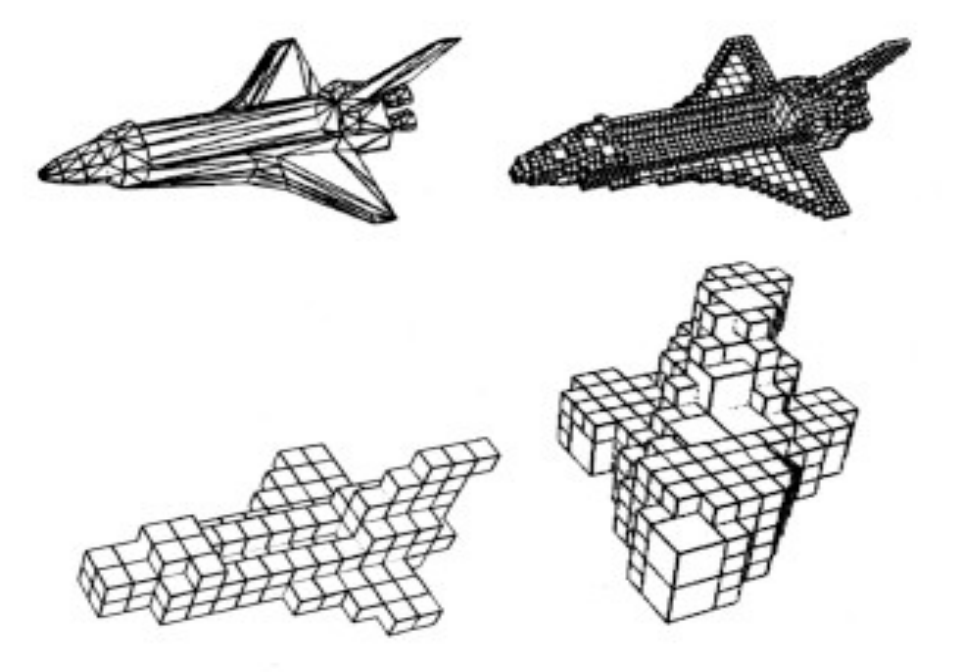


Representación de barrido (sweep rep)



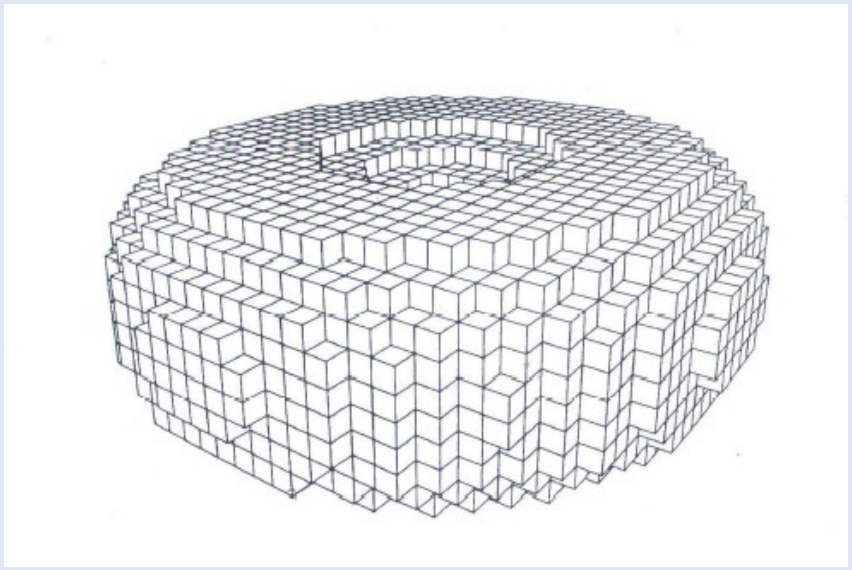
Descomposición espacial en celdas (cell decomposition)

- Objeto se descompone en celdas básicas que se intersectan solo en los bordes
  - Celdas básicas 2D: triángulos y cuadriláteros
  - Celdas básicas 3D: tetraedros, hexaedros, pirámides y prismas, entre otros



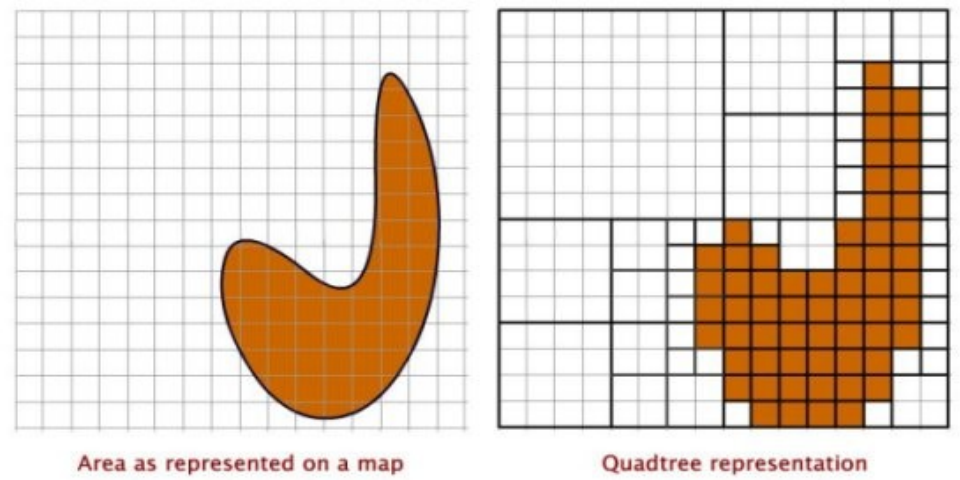
Descomposición espacial en celdas (cell decomposition)

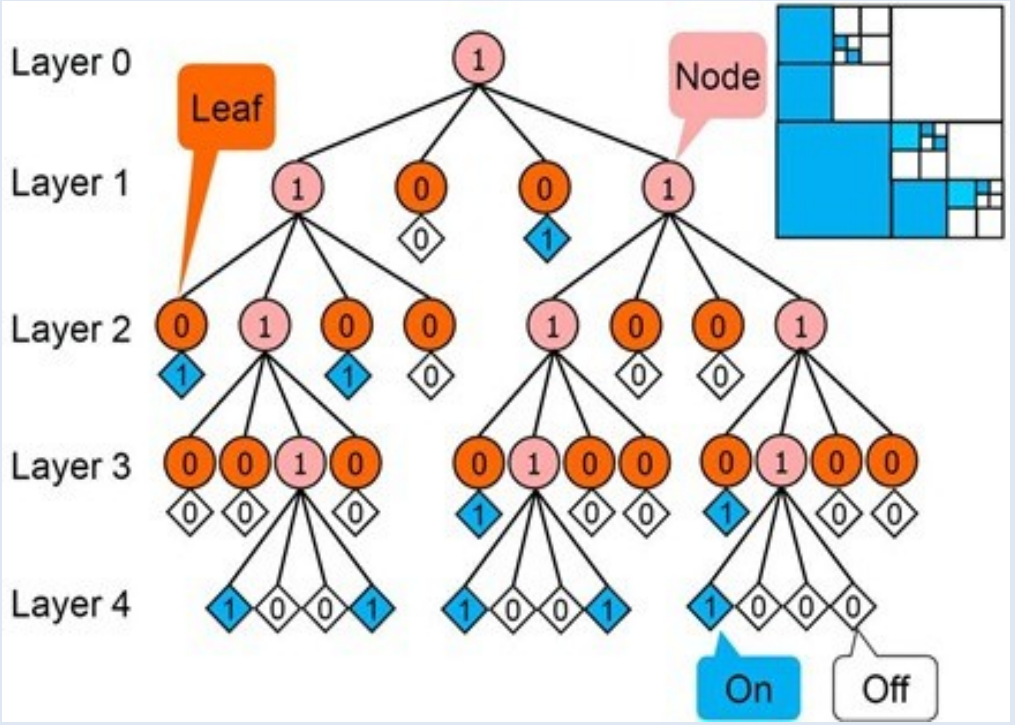
- Enumeración espacial simple (malla de voxels): partición generada por cubos (voxels) de igual tamaño, generalmente de arcos paralelos a los ejes principales
  - Sólido se representa por los cubos que están dentro o intersectan el borde del sólido



Descomposición espacial en celdas (cell decomposition)

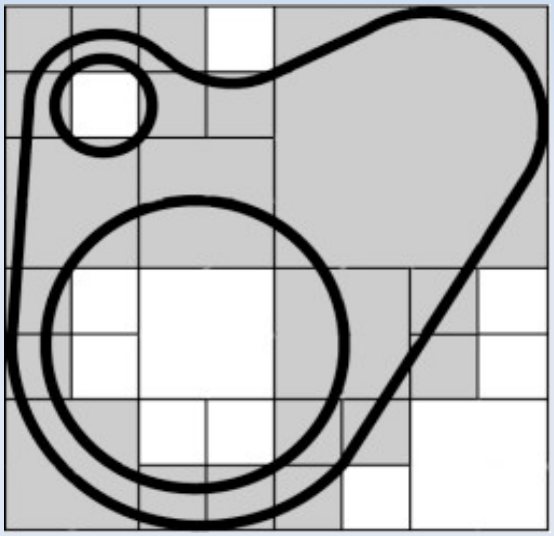
- Quadtrees (representar objetos 2D) variando el tamaño de los cuadrados
  - Rodear objeto por cuadrado más pequeño
  - Dividir en cuatro hasta lograr precisión deseada
  - Estructura de datos: árbol

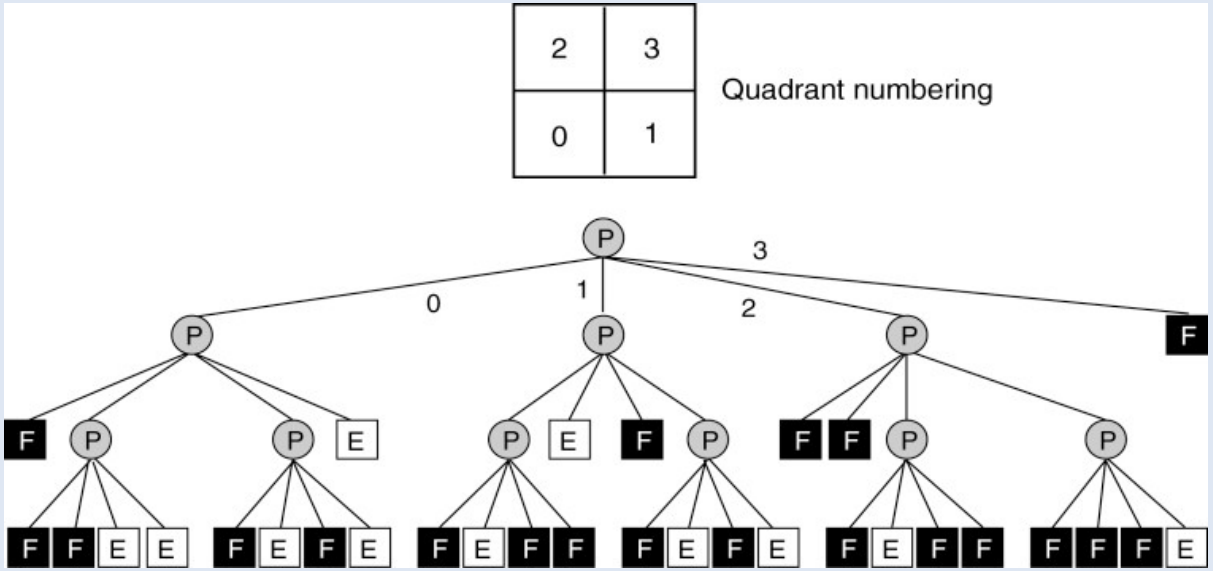




Descomposición espacial en celdas (cell decomposition)

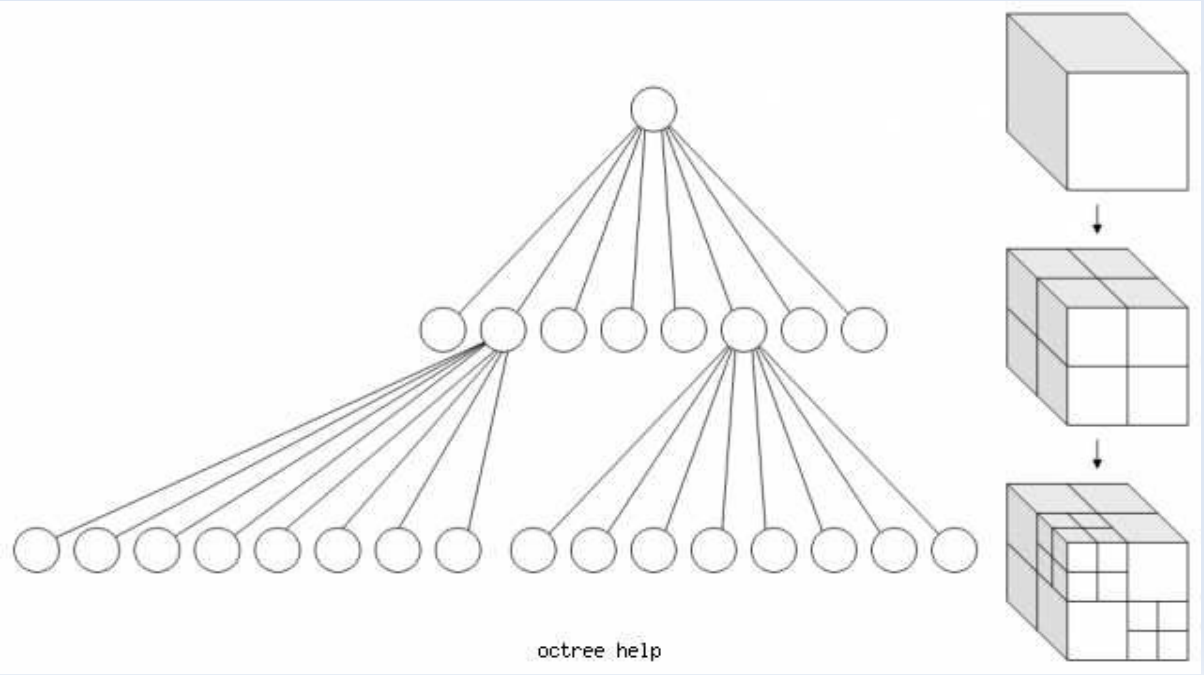
- Quadtrees (representar objetos 2D): otro ejemplo





Descomposición espacial en celdas (cell decomposition)

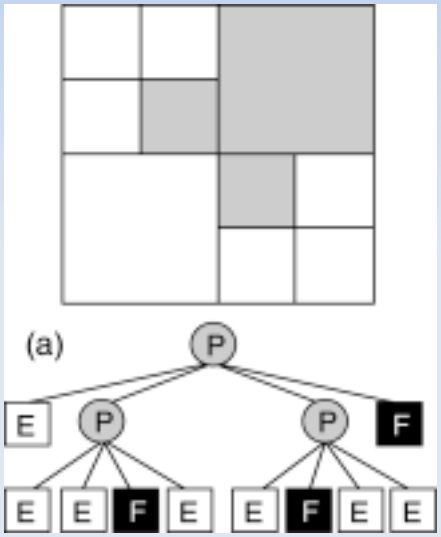
- Octrees (representar objetos 3D variando el tamaño de los cuadrados
  - Rodear objeto porcubo más pequeño
  - Dividir en ocho hasta lograr precisión deseada
  - Estructura de datos: árbol

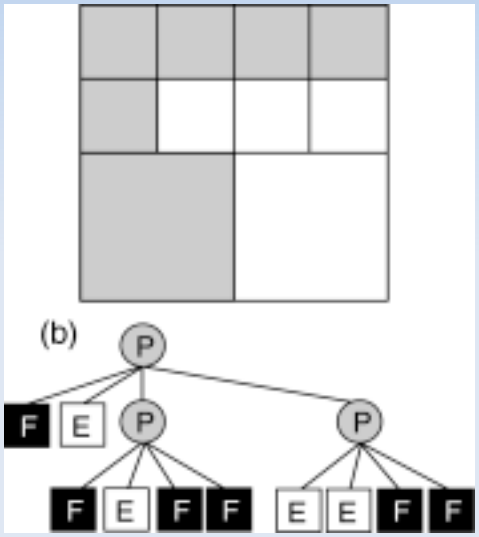


Descomposición espacial en celdas (cell decomposition)

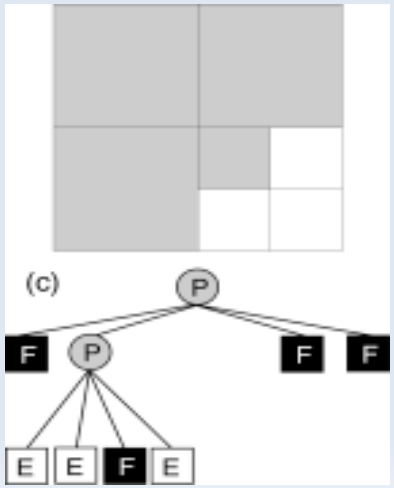
- Octrees: operadores unión e intersección (sobre un quadtree)

S

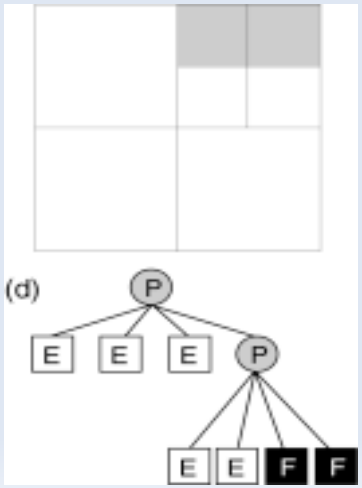




T



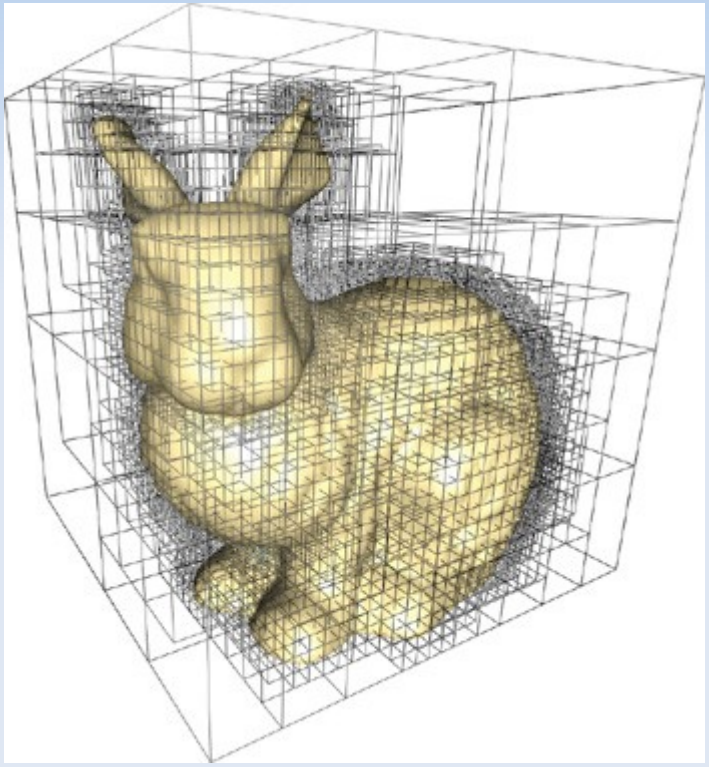
S U T

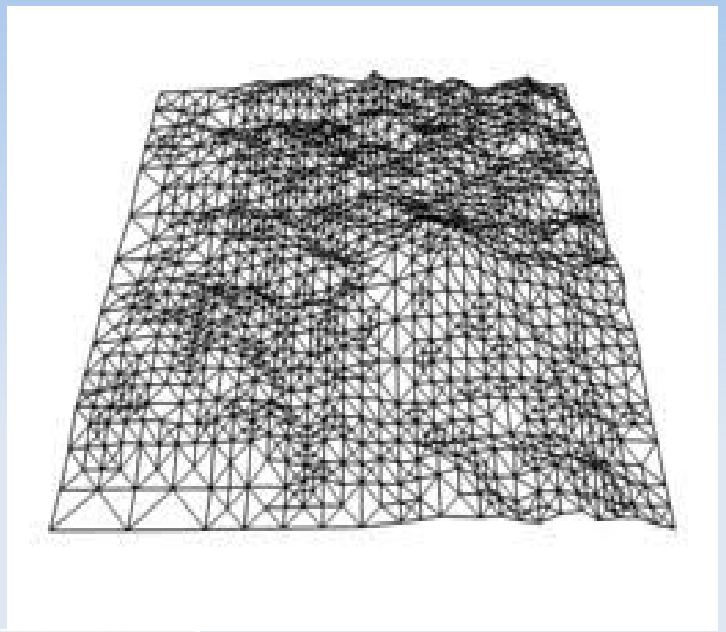


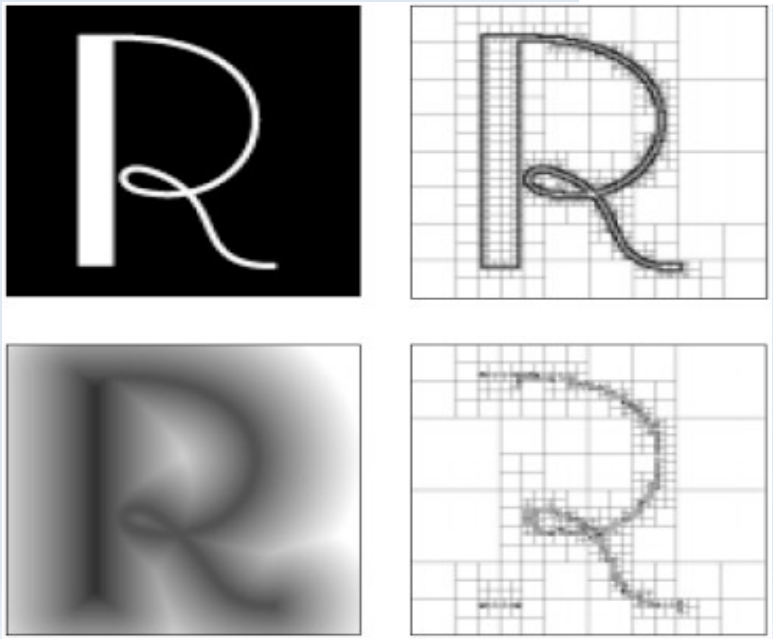
S∩T

Descomposición espacial en celdas (cell decomposition)

- Aplicaciones con octrees o quadtrees:







Descomposición espacial en celdas (cell decomposition)

- Cálculo de volumen? Sumar volumen de cubos
- Operaciones como unión, intersección y resta son simples
  - Propuesto: definir todos los casos
- Desventaja: se necesitan muchas celdas cuando hay que representar superficies curvas y detalles pequeños
- Extensiones: generación de mallas de tetreadros y mixtas

Comparación de los modelos

	CSG	B-rep	Octree
Precisión	buena	buena	Más o menos
Dominio	grande	grande	grande
unicidad	no	si	si
Validación (chequear que el modelo es válido)	fácil	difícil	fácil
Cerradura (bajo operadores booleanos)	Si (simple)	Si (difícil de implementar)	Si (simple)
Compacto	Si	No (en comparación de CGS)	Más o menos