

Modelos Geométricos

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática
Curso 2016-2017

Contenidos

1 Objetivos

2 Introducción

3 Modelado de Sólidos

- Definición de sólido
- Modelo de sólido

4 Representación B-Rep

5 Modelos de alta resolución

- Modelos multiresolución

6 Descomposición espacial

- Grid
- Octree
- Octrees extendidos
- Indexación espacial de la escena

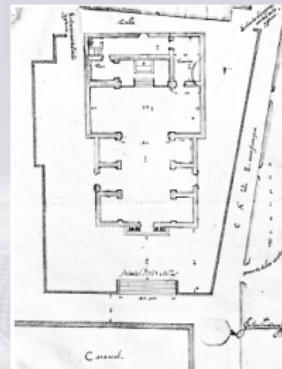
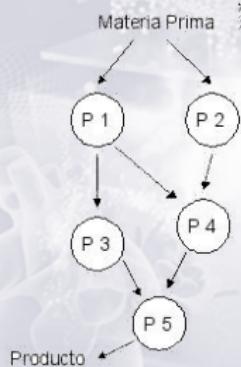
Objetivos

- Concepto de modelo
- Diferencias entre modelos geométricos y volumétricos
- Concepto de sólido y cómo modelarlo
- Representación de sólidos mediante su frontera
 - ▶ Mallas de triángulos
- Representación de sólidos por descomposición espacial
- Indexación espacial de la escena

Introducción

- ¿Qué es un modelo?

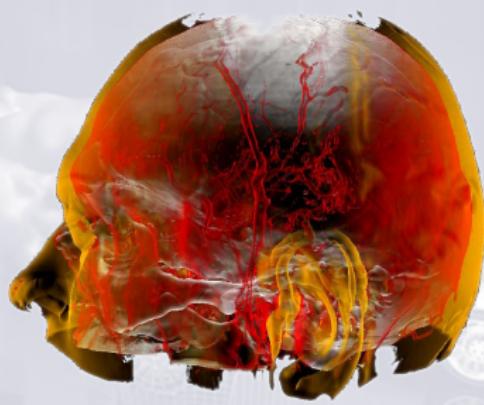
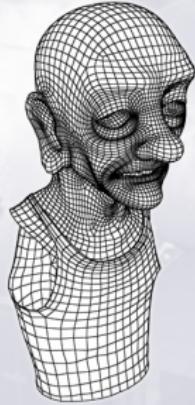
- ▶ Representación de características relevantes de una entidad
- ▶ La entidad puede ser real o imaginaria
- ▶ Permite *trabajar* con el modelo en vez de hacerlo con la entidad



Modelos geométricos y volumétricos

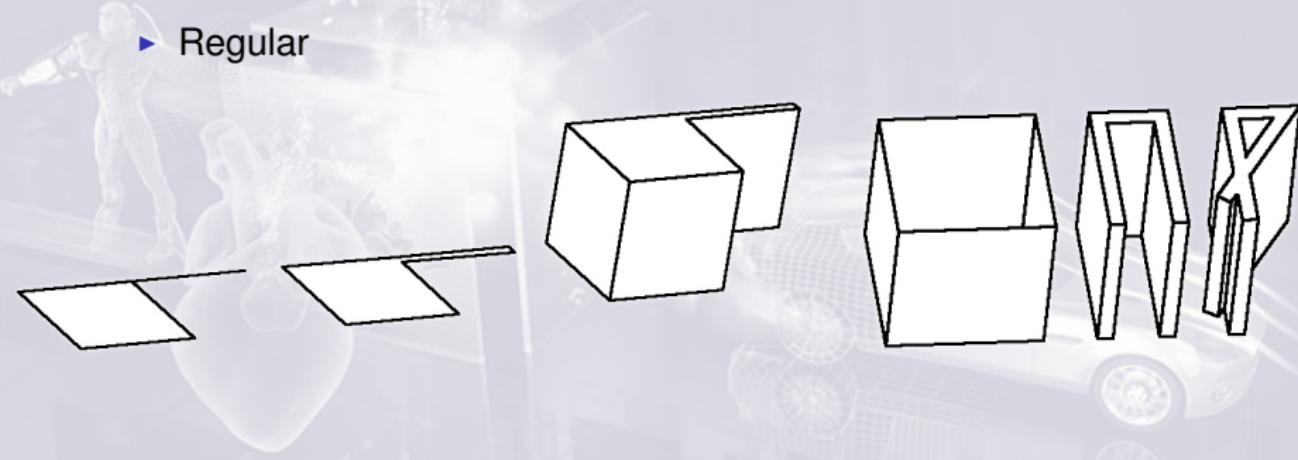
- Modelos geométricos
 - ▶ Representan la geometría de un objeto, su frontera
 - ▶ Se considera que el objeto es homogéneo en su interior

- Modelos volumétricos
 - ▶ Representan propiedades espacialmente localizadas
 - ▶ El interior es heterogéneo en cuanto a su material o propiedades



Modelado de sólidos

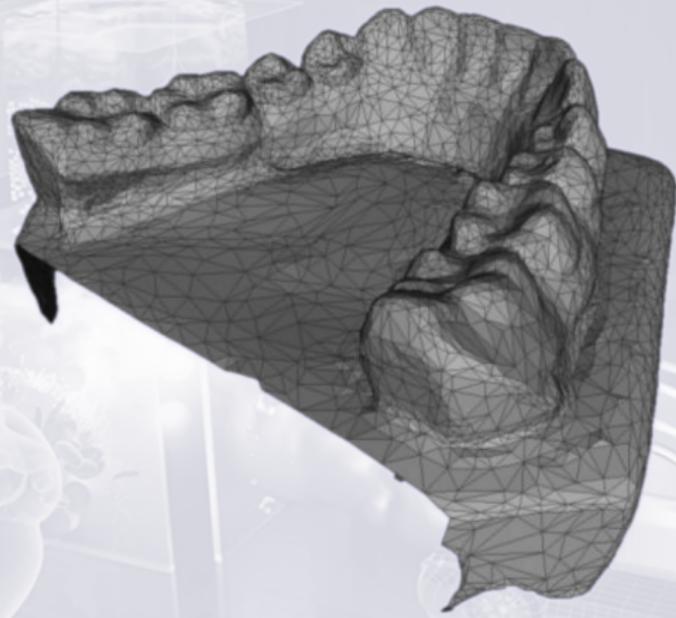
- ¿Qué es un sólido?
 - ▶ Una *porción* del espacio. $S \subset \mathbb{R}^3$
 - ▶ Acotado
 - ▶ Cerrado
 - ▶ Rígido
 - ▶ Regular



Modelado de Sólidos

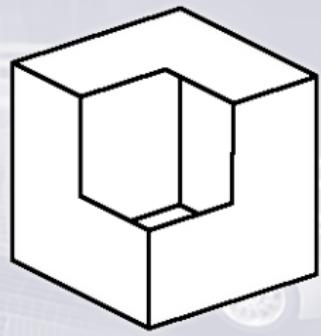
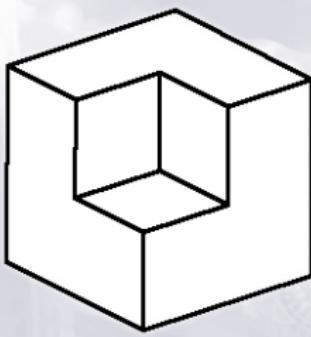
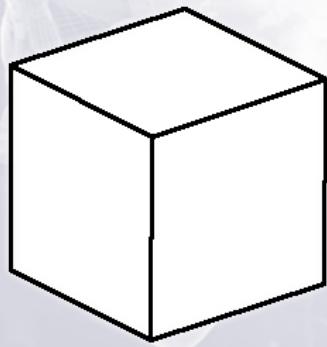
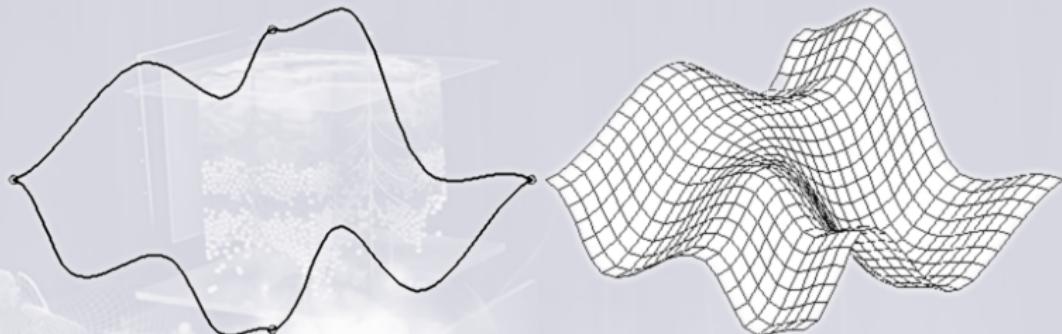
Sólidos vs. Superficies

- El modelo siguiente, ¿representa un sólido?, ¿por qué?



Sólidos vs. Superficies

- ¿Qué es una superficie?



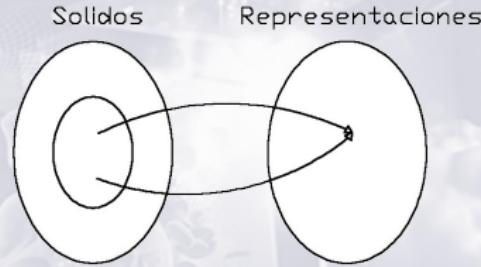
Modelado de Sólidos

- **Representación de un Sólido:**

- ▶ Una colección finita de símbolos (de un alfabeto finito) que describen un sólido.
- ▶ Ejemplo: $(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 \leq r^2$

- **Esquema de Representación:**

- ▶ La relación entre los sólidos y las descripciones de estos.

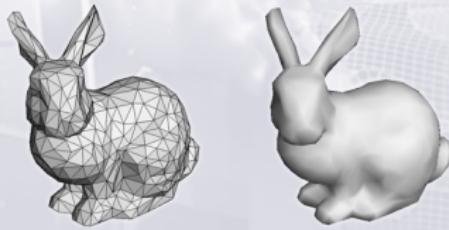


- Propiedades deseables: **dominio**, **no-ambigüedad**, poca ocupación de memoria, facilidad de creación y edición, etc.

Modelado de Sólidos

Representación B-Rep

- Un sólido es representado describiendo la frontera que lo delimita
 - ▶ Boundary Representation, B-Rep
- La frontera es representada mediante un malla de triángulos
- Características de la representación
 - ▶ Es exacta para sólidos poliédricos
 - ▶ Es aproximada en las fronteras curvas
 - ★ Usando las técnicas de rendering adecuadas se suple *visualmente* esa desventaja
 - ▶ Es robusta e independiente de la complejidad del sólido

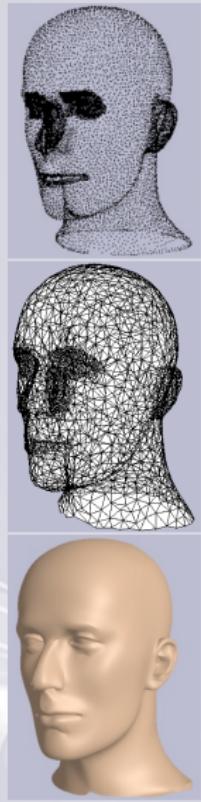


Mallas de triángulos

- ¿Cómo representarlas?
- Depende:
 - ▶ ¿Quiero representar superficies? ¿Sólidos?
 - ▶ ¿Represento solo triángulos? ¿Se admiten polígonos arbitrarios?
 - ▶ ¿Me interesa una representación jerárquica?
 - ▶ ¿Qué algoritmos se van a usar?
 - ▶ ¿Sólo se va a dibujar?
 - ▶ ¿Qué tipo de operaciones se van a hacer?
 - ▶ ¿Qué información extra debo almacenar?
 - ▶ ¿Qué requisitos de espacio de almacenamiento y tiempo de procesamiento voy a tener? ¿Dispongo de recursos suficientes?

Representación de mallas de triángulos

- Geometría
 - ▶ Elementos básicos que componen la malla.
Los vértices.
- Topología
 - ▶ Conexiones y relaciones entre dichos elementos.
Los triángulos
- Información extra
 - ▶ Coordenadas de textura
 - ▶ Color
 - ▶ Transparencia, etc.



Representación de mallas de triángulos

Listado de triángulos

- Representación:

- ▶ Malla = { Triángulo }
- ▶ Triángulo = ($x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$)

- Otra representación:

- ▶ Malla = { coordenada }

Es decir, Malla = [$x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, \dots, x_n, y_n, z_n$]

- ¿Qué ventajas y desventajas?

Ejemplo usando OpenGL

```
glBegin(GL_TRIANGLES);           // OJO !
    glVertex3f(0.0f, 0.0f, 0.0f); // Esta forma de hacerlo
    glVertex3f(1.0f, 1.0f, 0.0f); //     está deprecated
    glVertex3f(2.0f, 0.0f, 0.0f);
    .
    .
glEnd();
```

Representación de mallas de triángulos

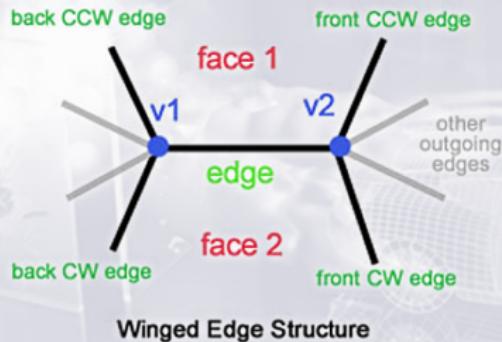
Lista de vértices y de triángulos

- Representación:
 - ▶ Malla = ({ Vértice } , { Triángulo })
 - ▶ Vértice_i = (x_i, y_i, z_i)
 - ▶ Triángulo_j = (v_k, v_l, v_m)
- O también:
 - ▶ Malla = ({ Coordenada } , { Índice })
 - Es decir, Vértices = [$x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n$]
 - Triángulos = [$v_1, v_2, v_3, \dots, v_2, v_1, v_7$]
- ¿Ventajas y desventajas?
- Los Vertex Buffer Objects se basan en esta representación

Representación de mallas de triángulos

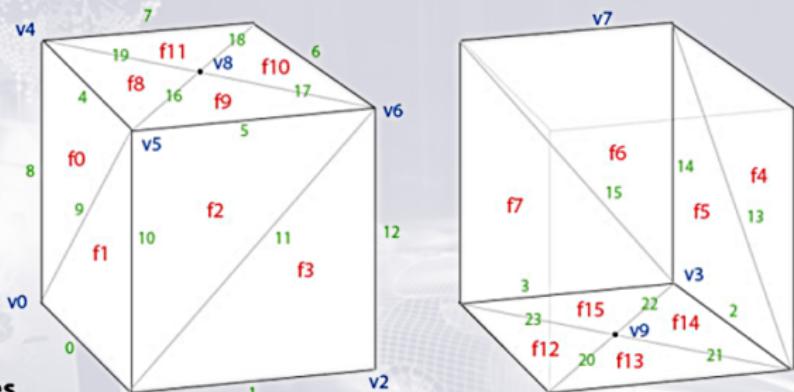
Aristas aladas

- Esta representación se centra en la arista
- Se almacena un **lista de aristas**. Cada arista guarda:
 - ▶ Referencias a sus 2 vértices
 - ▶ Referencias a sus 2 caras
 - ▶ Referencias a las aristas anterior y siguiente en cada cara



Aristas aladas

- Además, se almacena:
 - Una **lista de vértices**
 - Cada vértice almacena referencias a las aristas que inciden en él
 - Una **lista de caras**
 - Cada cara almacena referencias a las aristas que la componen



Winged-Edge Meshes

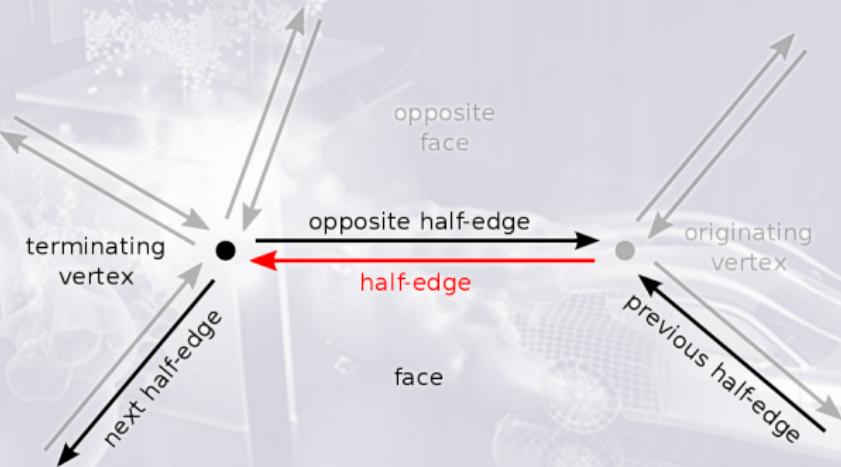
Face List
f0 4 8 9
f1 0 10 9

Edge List
e0 v0 v1 f1 f12 9 23 10 20
e1 v1 v2 f3 f13 11 20 12 21

Vertex List
v0 0,0 8 9 0 23 3
v1 1,0 10 11 1 20 0

Semi-aristas aladas

- Cada arista se almacena 2 veces. Dos semi-aristas
 - ▶ Una vez por cada cara
 - ▶ Con la información relativa a dicha cara
 - ▶ Siempre se recorre en sentido antihorario según su cara



Semi-aristas aladas

Estructura

- **Lista de Semi-Aristas.** Se almacenan las siguientes referencias:
 - ▶ Al vértice destino
 - ▶ A su cara
 - ▶ A la semi-arista previa
 - ▶ A la semi-arista siguiente
 - ▶ A la semi-arista opuesta
- **Lista de Vértices.** Para cada uno se almacena:
 - ▶ Sus coordenadas
 - ▶ Referencia a una de sus semi-aristas
 - ▶ Otros datos asociados: normal, coordenadas de textura, etc.
- **Lista de Caras.** Para cada una se almacena:
 - ▶ Referencia a una de sus semi-aristas

Bibliotecas existentes

Computational Geometry Algorithms Library (CGAL)

- www.cgal.org
- Biblioteca Open Source en C++
- Proporciona implementaciones eficientes y fiables de algoritmos sobre geometría
- Permite usar coordenadas tanto reales como enteras
- Usa semiaristas aladas
 - ▶ La usan empresas como Leica, Toshiba o France Telecom
 - ▶ Tiene 500.000 líneas de código y 3.500 páginas de manual
 - ▶ Su equipo lo integran 20 desarrolladores
 - ▶ Recibe unas 10.000 descargas anuales



OpenMesh + OpenFlipper

OpenMesh OpenFlipper



RWTHAACHEN
UNIVERSITY

- www.openmesh.org www.openflipper.org
- Desarrollado por el grupo del profesor Kobbelt
- OpenMesh es una biblioteca para la gestión de mallas poligonales
- OpenFlipper es un framework para desarrollar aplicaciones geométricas completas
- Se basa en QT para la interfaz de usuario
 - ▶ Usa semiaristas aladas
 - ▶ Implementa numerosos algoritmos básicos
 - ▶ Permite adaptar fácilmente el Vértice para que contenga más o menos información

VCGLib

Visual Computing Lab



- vcg.isti.cnr.it
- Desarrollada por el Visual Computing Lab de Pisa
Perteneciente al Consiglio Nazionale delle Ricerche (ISTI CNR)
- Usa listas de vértices, caras y aristas
- Proporciona clases para calcular propiedades sobre las mallas
- Es la base de MeshLab
 - ▶ Un potente software open source de procesamiento de mallas

Modelos de alta resolución

Introducción

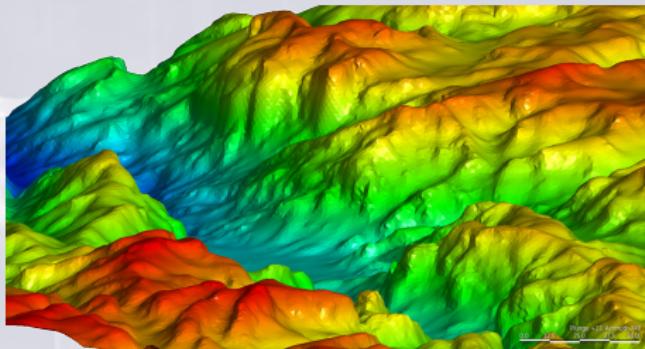
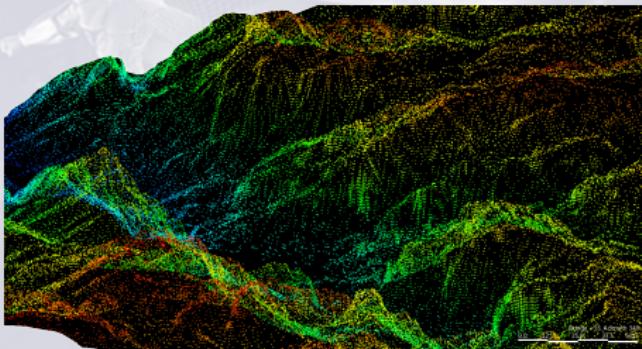
- El realismo a nivel visual se puede conseguir mediante rendering
 - ▶ Texturas, shading, etc.
- Para obtener precisión geométrica en el modelo es necesario usar mallas con un alto número de polígonos



Modelos de alta resolución

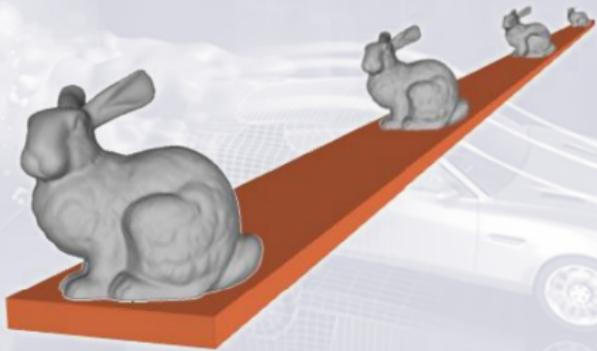
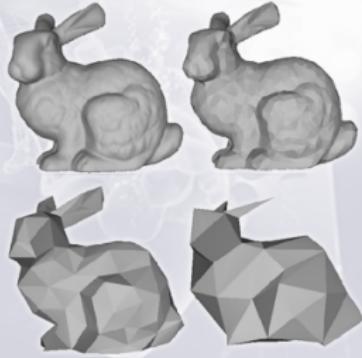
Captura

- Un escáner 3D puede capturar decenas de millones de puntos
 - ▶ ¿Cómo se representan mallas de tal tamaño?
 - ▶ ¿Cómo se manipulan?
 - ▶ ¿Cómo se visualizan?



Modelos multiresolución

- El modelo incluye (o permite obtener) representaciones del objeto a diferente resolución
 - ▶ También se usan los términos *Nivel de Detalle* y *Level of Detail* (LOD)
- Las representaciones geométricamente más simples se usan en:
 - ▶ Elementos que están distantes
 - ▶ Ocupan pocos píxeles al visualizarse
 - ▶ Son menos importantes



Multiresolución discreta

- Es el esquema más básico y antiguo
- El más utilizado por su simplicidad
- Se crean y almacenan varias versiones del modelo completo a distinta resolución
- Se usa una u otra según alguna métrica de calidad



Multiresolución discreta

Problemas

- Presenta algunos problemas
 - ▶ Hay que tener las instancias calculadas y almacenadas, puede que incluso en memoria
 - ▶ Si el umbral no se ajusta bien, pueden notarse saltos de un modelo a otro
 - ▶ Si no se hace el cambio a tiempo, pueden usarse más o menos resolución que la que se debería



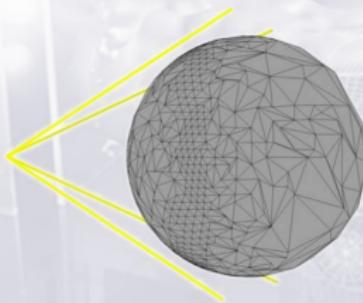
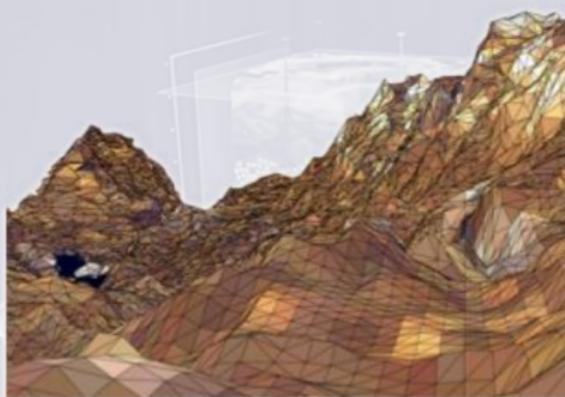
Multiresolución continua

- Se va refinando o simplificando la malla completa según algún criterio
 - ▶ El tamaño en píxeles de cada triángulo
 - ▶ La distancia al observador
 - ▶ La zona de la pantalla, etc.



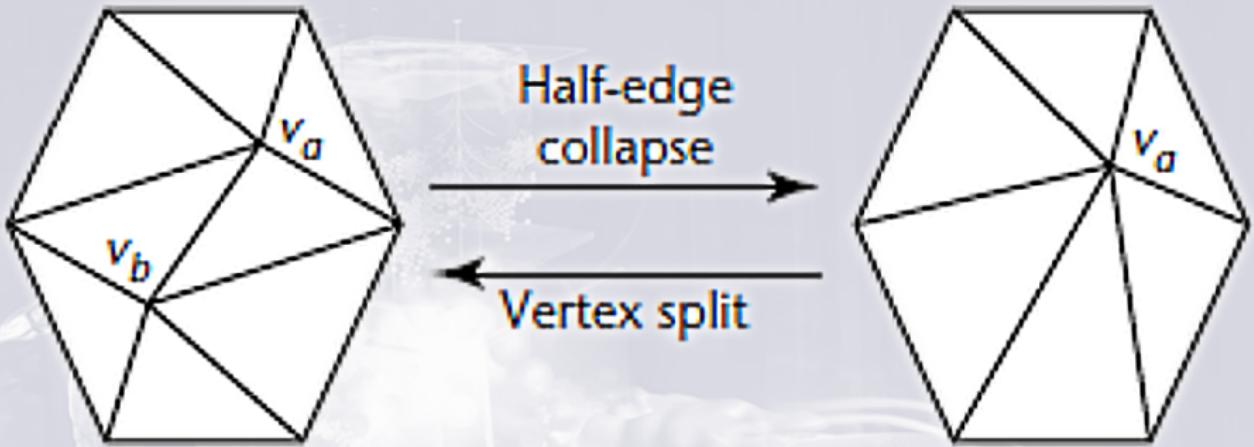
Multiresolución dependiente del observador

- Un modelo puede tener distintas zonas a distinta resolución



Operadores de simplificación

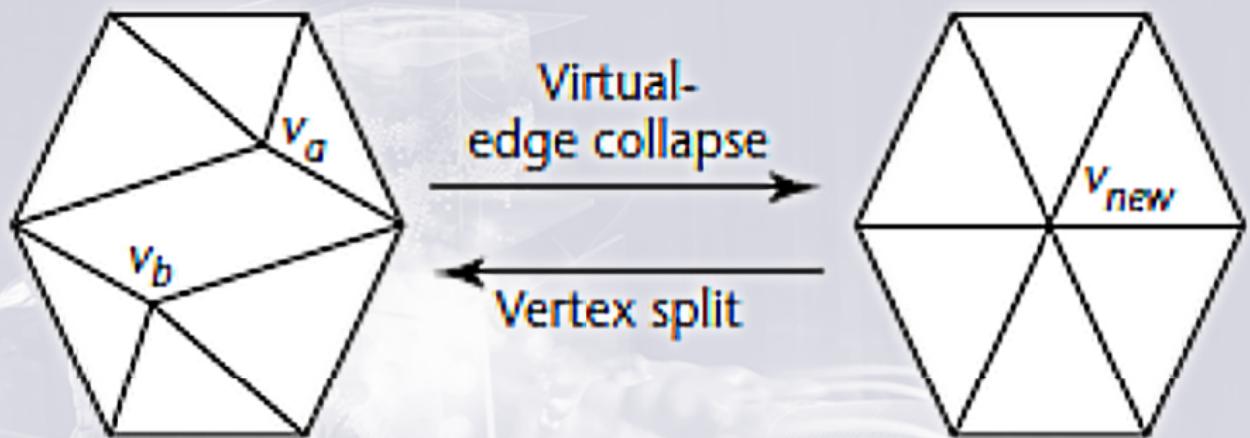
Colapsado de arista / expansión de vértice



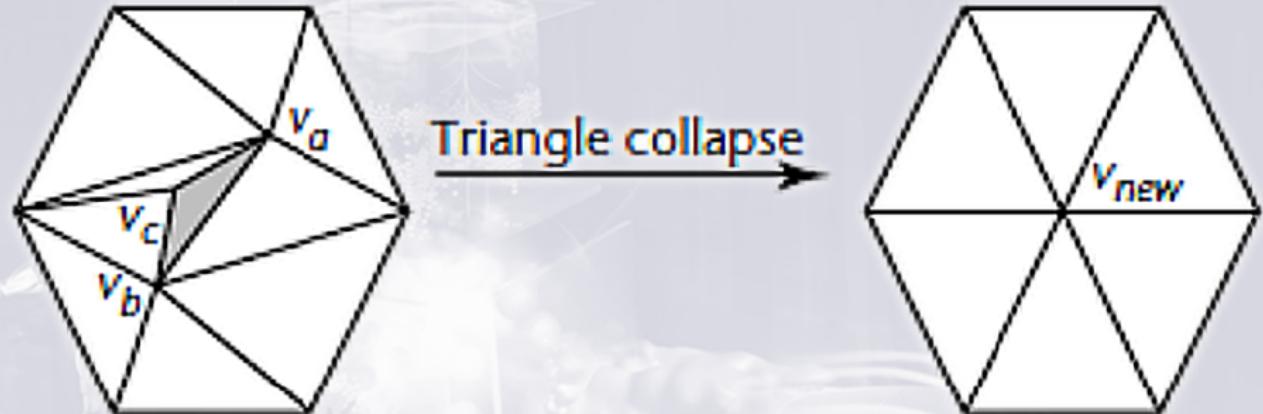
Imágenes de operadores extraídas de:

Luebke D. et al; *Level of detail for 3D graphics*; Morgan Kaufmann Publixhers; 2003

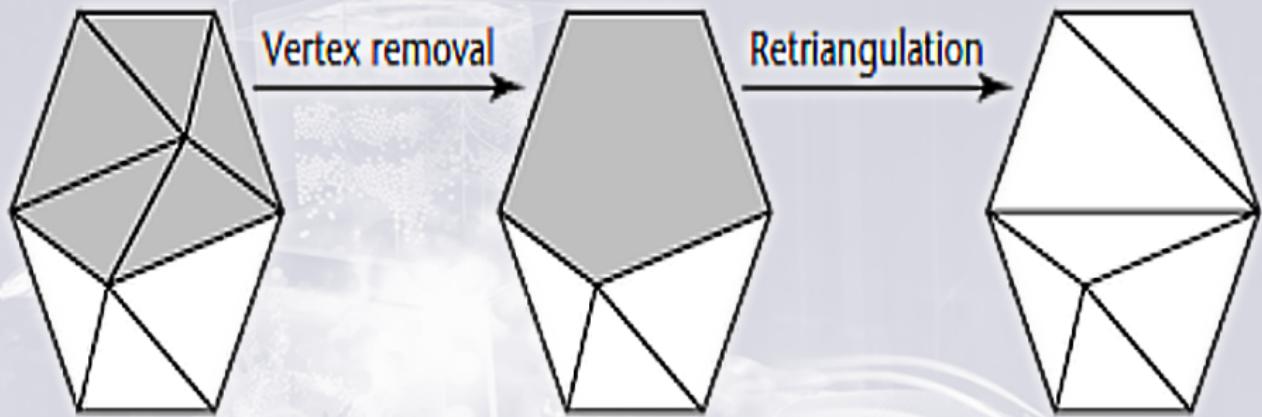
Colapsado de vértices / expansión de vértice



Colapsado de triángulo



Eliminación de vértice



Bibliotecas existentes

Proland



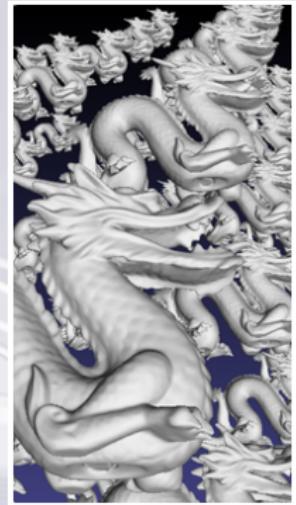
- proland.inrialpes.fr
- Biblioteca en C++ bajo licencia GPL v3
- Permite la visualización de modelos de terrenos, bosques, etc.
- Incluye la visualización de entornos atmosféricos complejos
- Dispone de buena documentación on-line y tutoriales



Bibliotecas existentes

Nexus

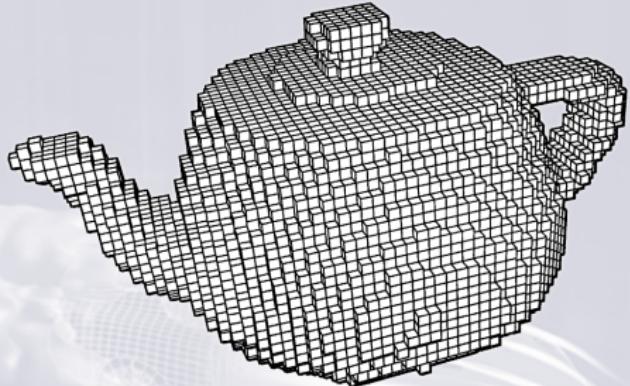
- vcg.isti.cnr.it/nexus/
- Biblioteca en C++
- Desarrollada por el Visual Computing Lab de Pisa
- Permite la creación y visualización de mallas multiresolución
- Características
 - ▶ Visualización interactiva de grandes modelos
 - ▶ Compresión
 - ▶ Color por vértice
 - ▶ Transmisión progresiva vía HTTP



Modelado de Sólidos

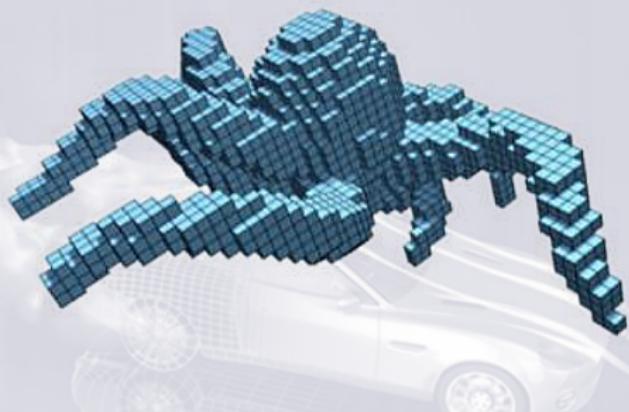
Representación por Descomposición espacial

- Se basan en representar el interior de los sólidos
- En algunos casos, la representación de la frontera es aproximada



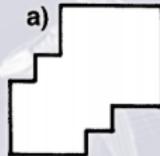
Grid

- El espacio es descompuesto en un grid de cubos de igual tamaño
- Se indica qué vóxeles ocupa el objeto
 - ▶ A vóxeles más pequeños, se tiene una representación más precisa
 - ▶ Pero que requiere más espacio de almacenamiento



Octree

- Permite reducir el espacio requerido al agrupar vóxeles
- Permite realizar ciertas operaciones muy eficientemente
 - ▶ Op. booleanas, cálculo del centro de masas, del volumen, etc.

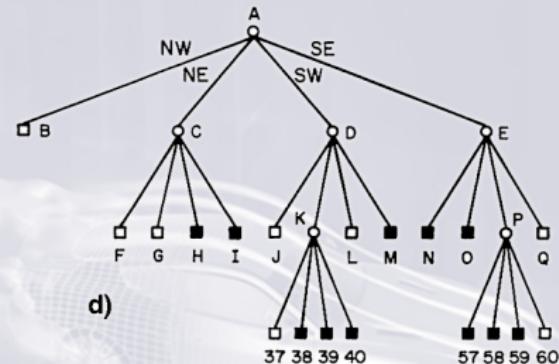


b)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0

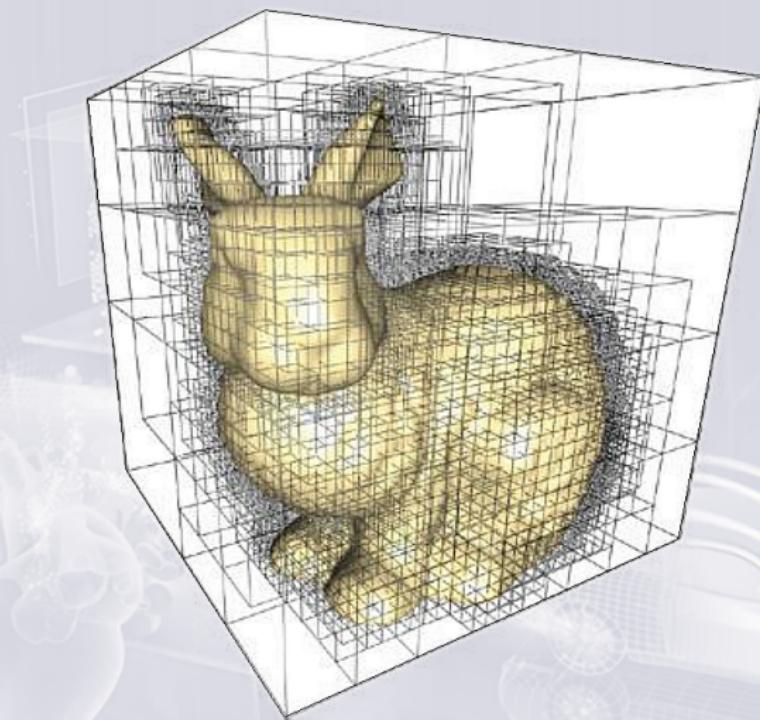
c)

		F	G
B		H	I
J	37 38 39 40	N	O
L	M 57 58 59 60	Q	



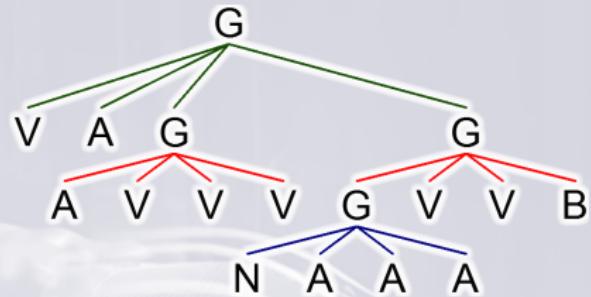
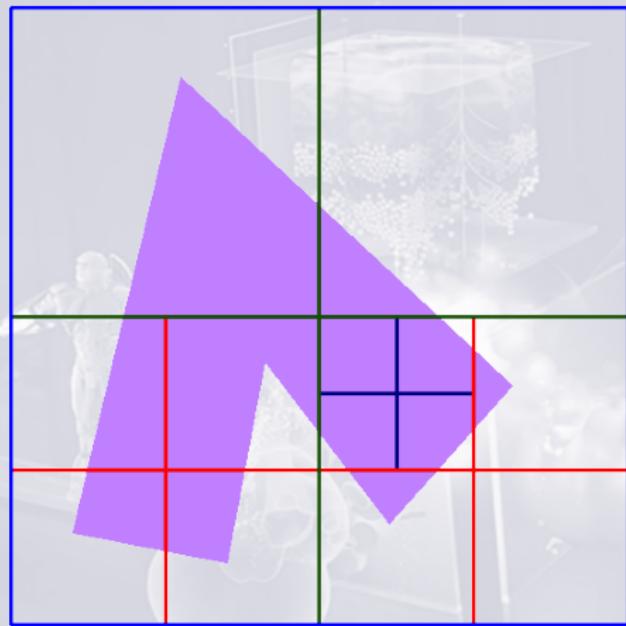
Octree

Ejemplo



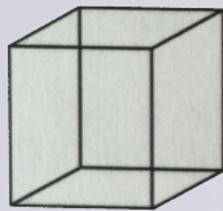
Octrees extendidos

Concepto en 2D

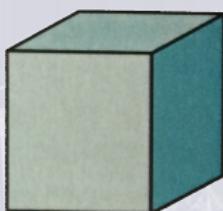


Octrees extendidos

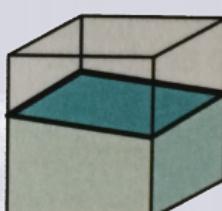
Tipos de nodos en 3D



Nodo BLANCO



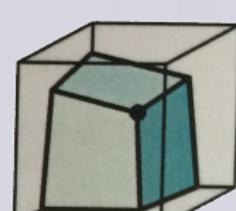
Nodo NEGRO



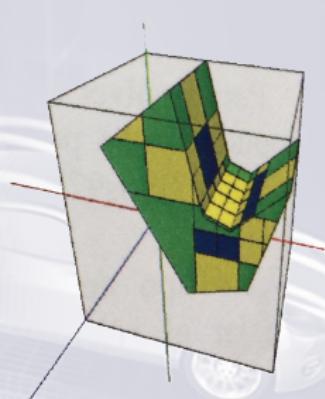
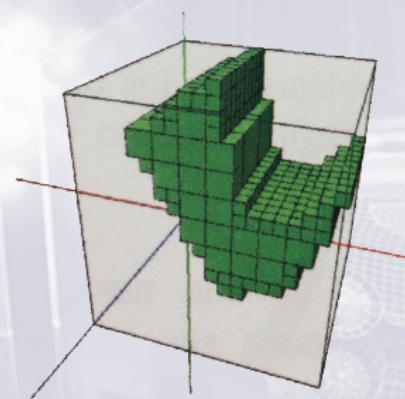
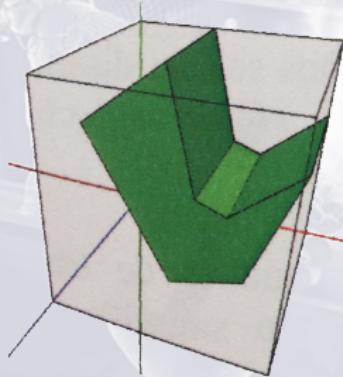
Nodo CARA



Nodo ARISTA

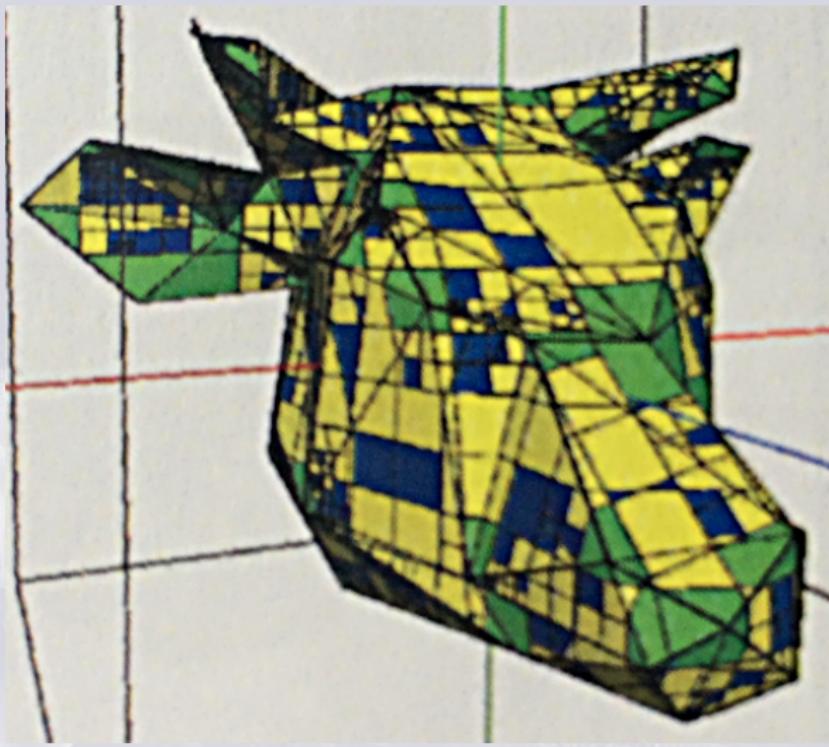


Nodo VERTICE



Octrees extendidos

Ejemplo



Indexación espacial de la escena

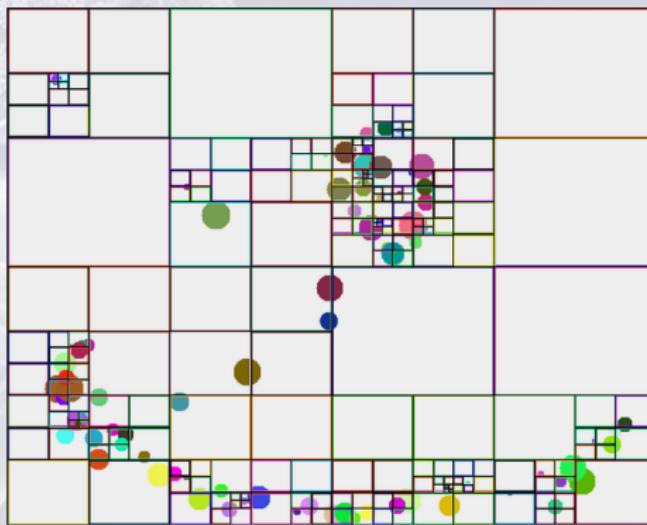
- Cuando se tienen muchos objetos en la escena es importante saber identificarlos con rapidez
 - ▶ Cuando se lanzan rayos Ray Tracing para rendering
 - ▶ Cuando se buscan colisiones entre objetos
- Las estructuras de descomposición espacial como las vistas son útiles para este fin



Estructuras para indexación espacial

Octrees

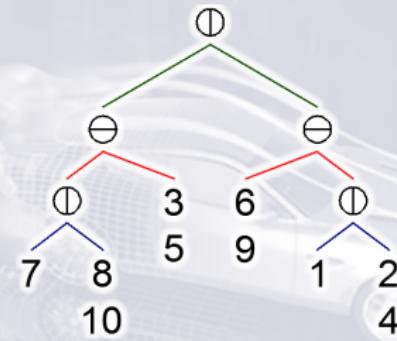
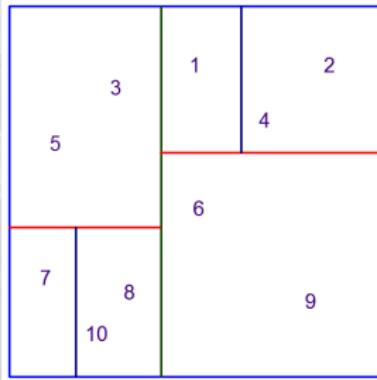
- El espacio se subdivide jerárquicamente en octantes
- Un octante solo se subdivide si contiene más objetos que el límite establecido



Estructuras para indexación espacial

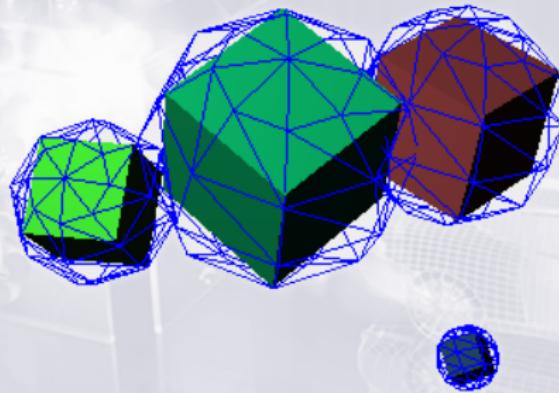
KD-Trees

- El espacio se subdivide en 2 semiespacios por un plano
- El plano está alineado con los ejes
- Se realiza la subdivisión alternativamente en cada dimensión
- Un semiespacio solo se subdivide si contiene más objetos que el límite establecido



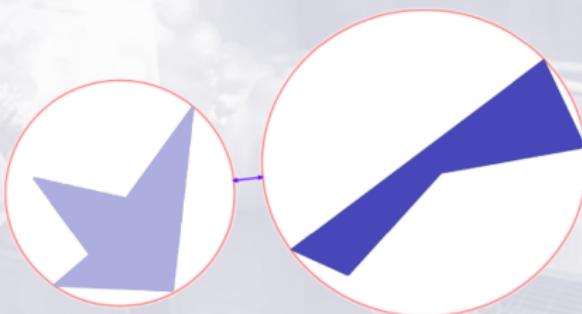
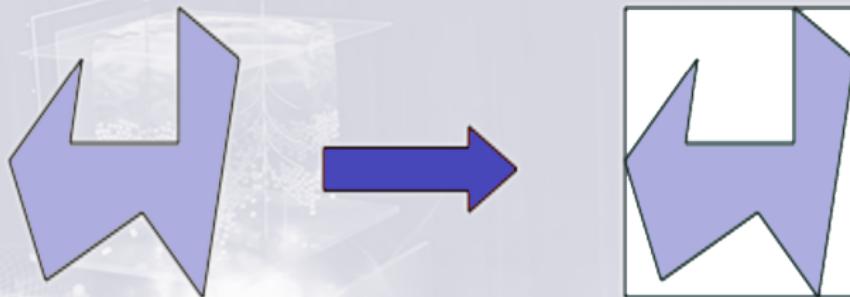
Detección de colisiones

- La detección de colisiones se realiza en dos fases
 - ▶ Fase gruesa:
Se descartan rápidamente los elementos que no colisionan
 - ▶ Fase fina:
Se determina con exactitud si 2 elementos sí colisionan
- En la fase gruesa se usan cajas o esferas englobantes



Cajas y esferas englobantes

Ejercicio: Diseña sendos algoritmos para calcular la caja y la esfera englobante de un objeto cualquiera a partir de su lista de vértices



Modelos Geométricos

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática
Curso 2016-2017

Parte de este material ha sido realizado en colaboración con Francisco Javier Melero Rus