Mallas de polígonos

Dr. Ivan Sipiran

Modelando objetos complejos

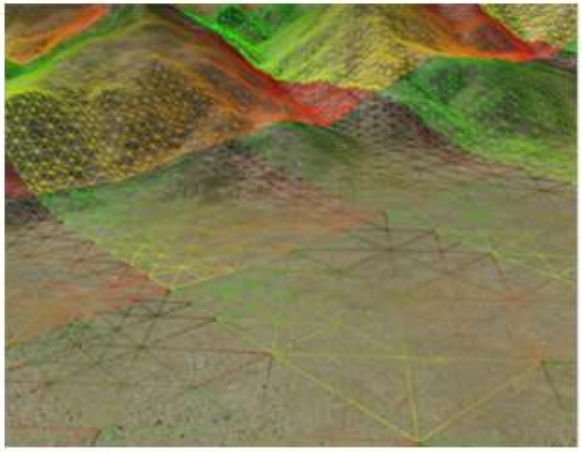
- Cuerpos geométricos simples pueden ser modelados vía ecuaciones
- Ejemplo: Esfera, cilindro, cubo, cono, etc.
- Cómo modelaríamos un objeto más complejo?
- Objetivos:
  - Modelar cualquier objeto en cualquier nivel de precisión
  - Debe ser fácil de construir y modificar
  - Cálculos deben ser simples
  - Debe ser fácil de implementar (de ser posible…)

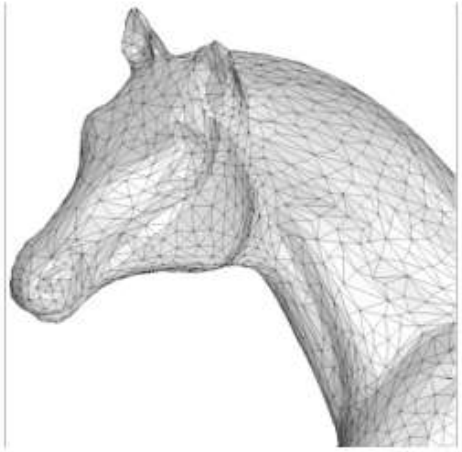
Modelando objetos complejos

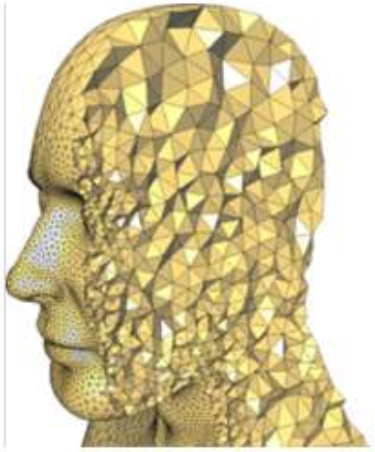
- Cuerpos geométricos simples pueden ser modelados vía ecuaciones
- Ejemplo: Esfera, cilindro, cubo, cono, etc.
- Cómo modelaríamos un objeto más complejo?
- Objetivos:
  - Modelar cualquier objeto en cualquier nivel de precisión
  - Debe ser fácil de construir y modificar
  - Cálculos deben ser simples
  - Debe ser fácil de implementar (de ser posible…)
- Logramos representar mayor complejidad utilizando piezas simples:
  - Mallas de polígonos, superficies parámetricas (Bézier)

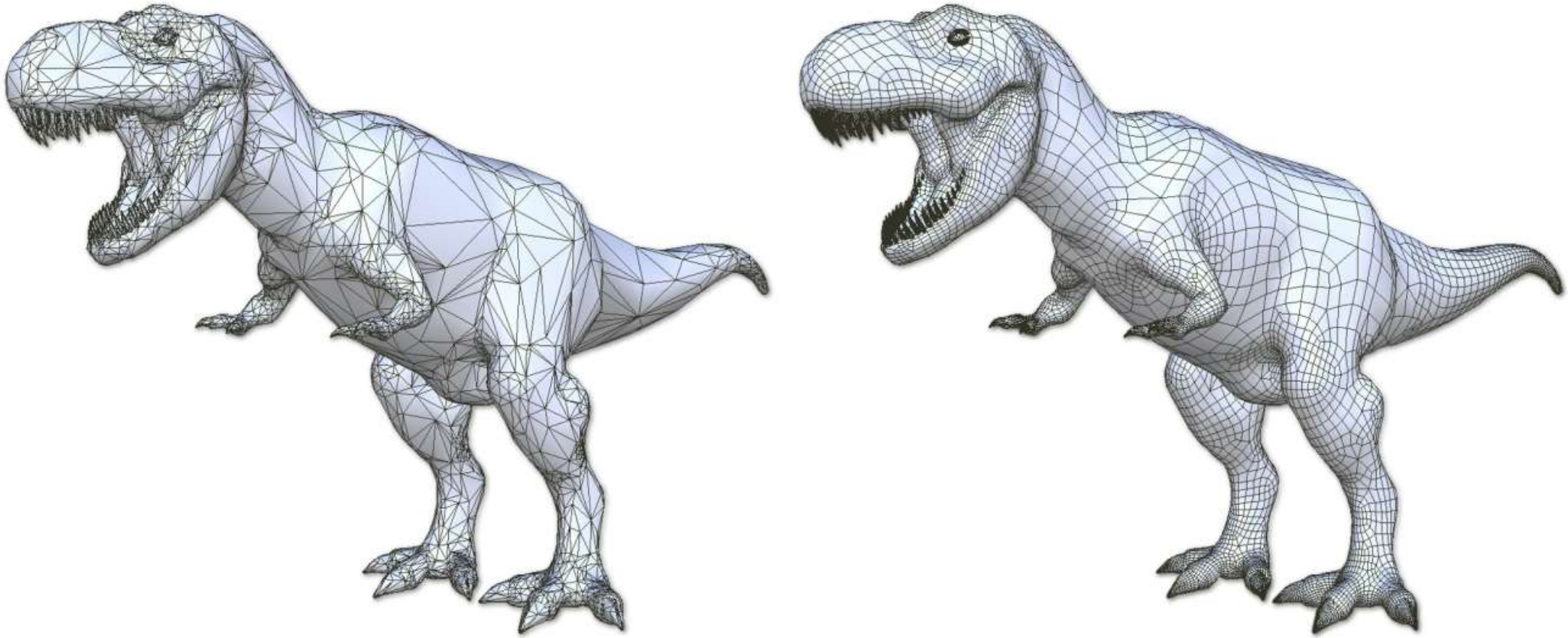
Mallas de polígonos

- Cualquier figura puede ser modelada por polígonos (si utilizamos suficientes)
- Qué tipos de polígonos?
  - Triángulos, cuadriláteros, pentágonos, etc
  - Mallas de triángulos son las más comunes
  - Si hay más de 3 lados, hay ambigüedades sobre lo que se quiere representar



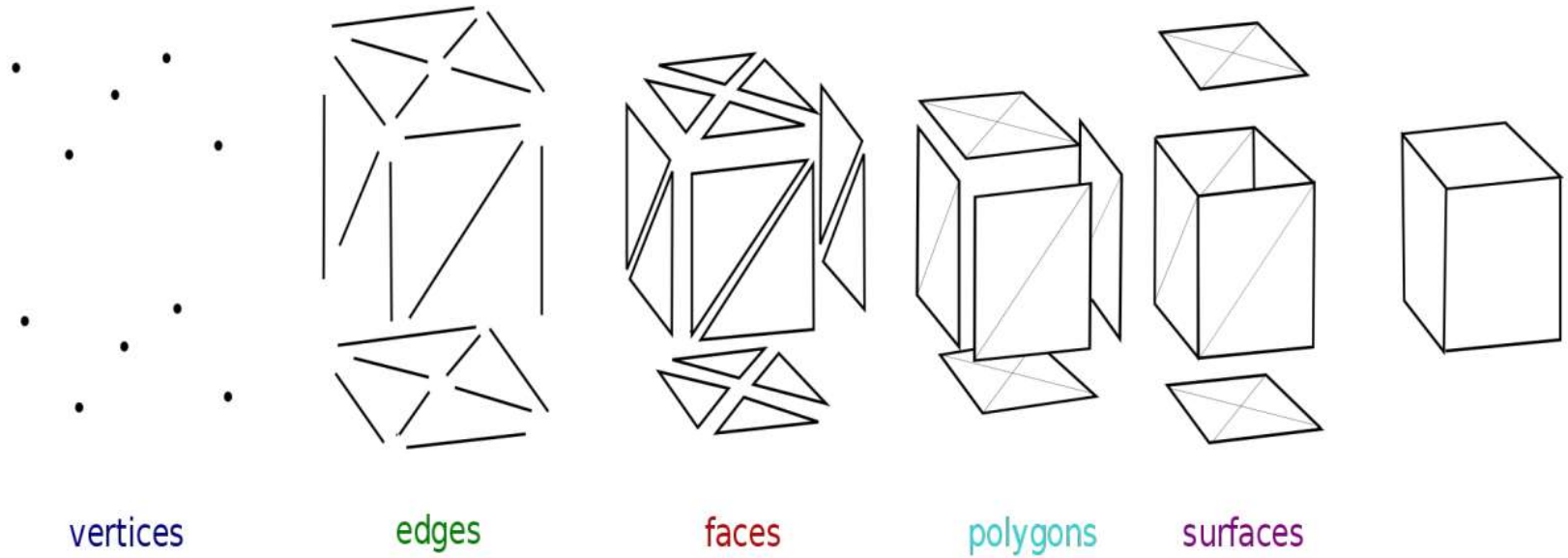






Créditos: Art Mesh

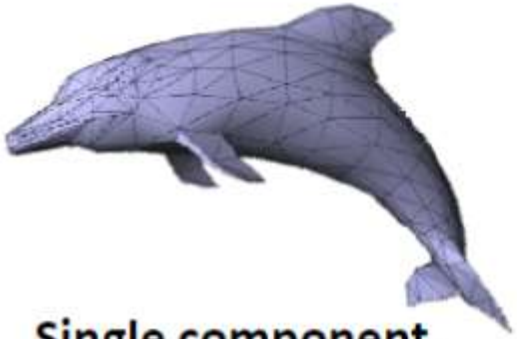
Conceptos



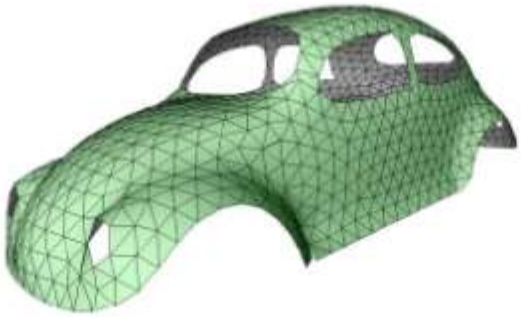
Conceptos

- Una malla de polígonos tiene una geometría y topología asociada
  - Elementos topológicos: arcos y caras
  - Elementos geométricos: puntos, segmentos y polígonos
  - La topología asociada la definen las relaciones de vecindad

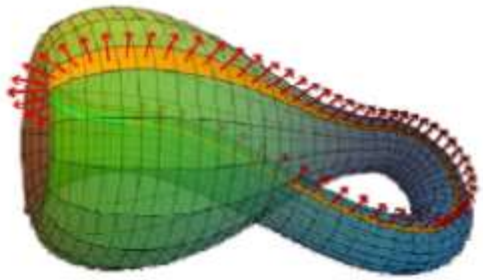
- La validación debe ser geométrica y topológica



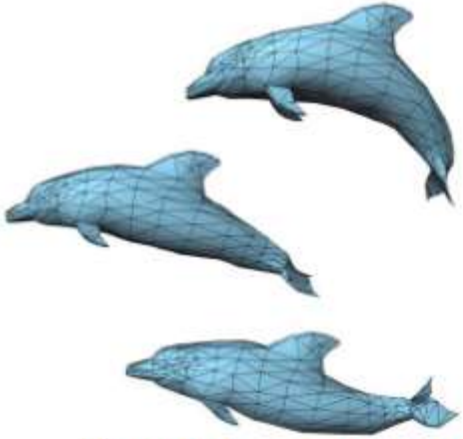
Single component, closed, triangular, orientable manifold



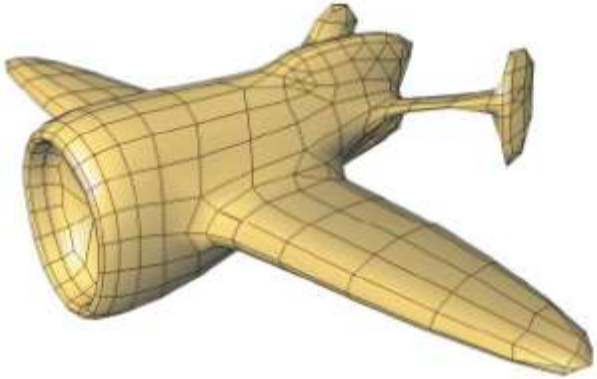
With boundaries



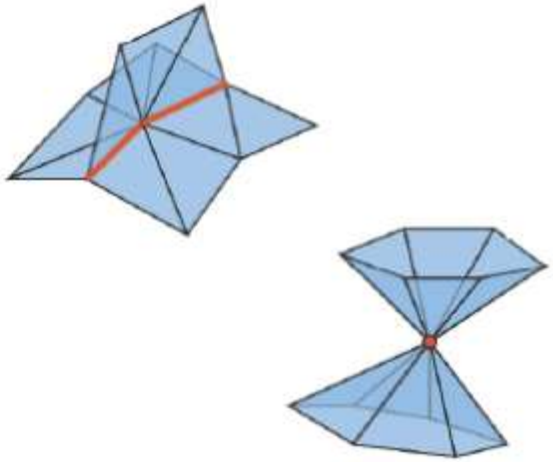
Not orientable



Multiple components



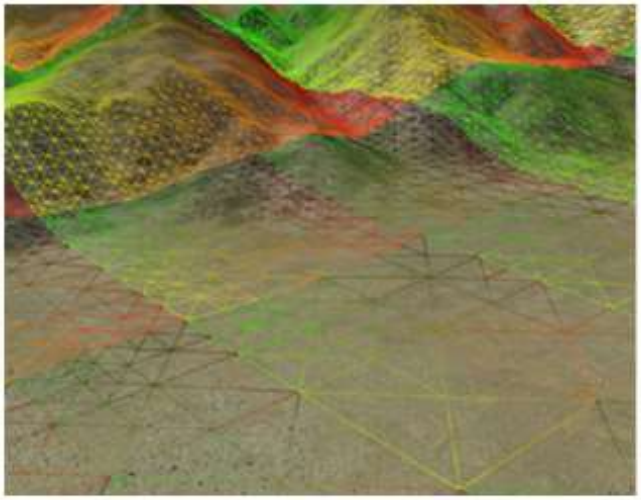
Not only triangles

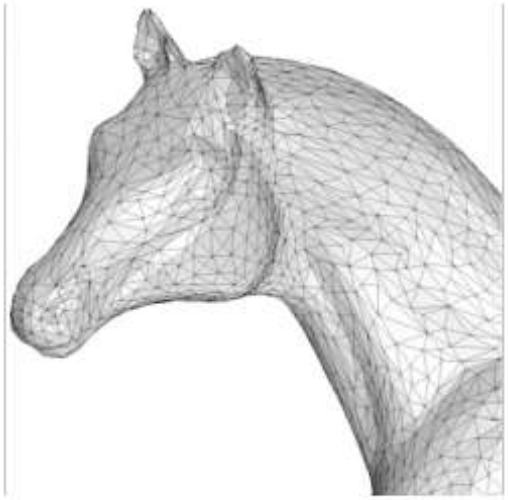


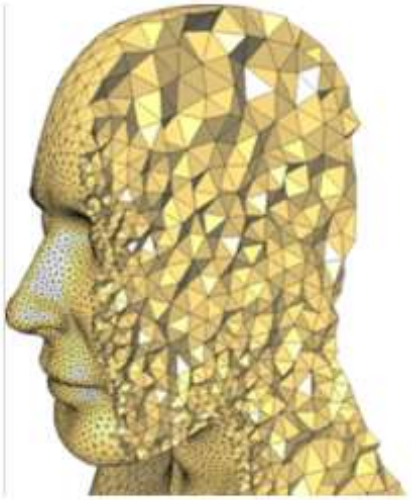
Non manifold

Aplicaciones

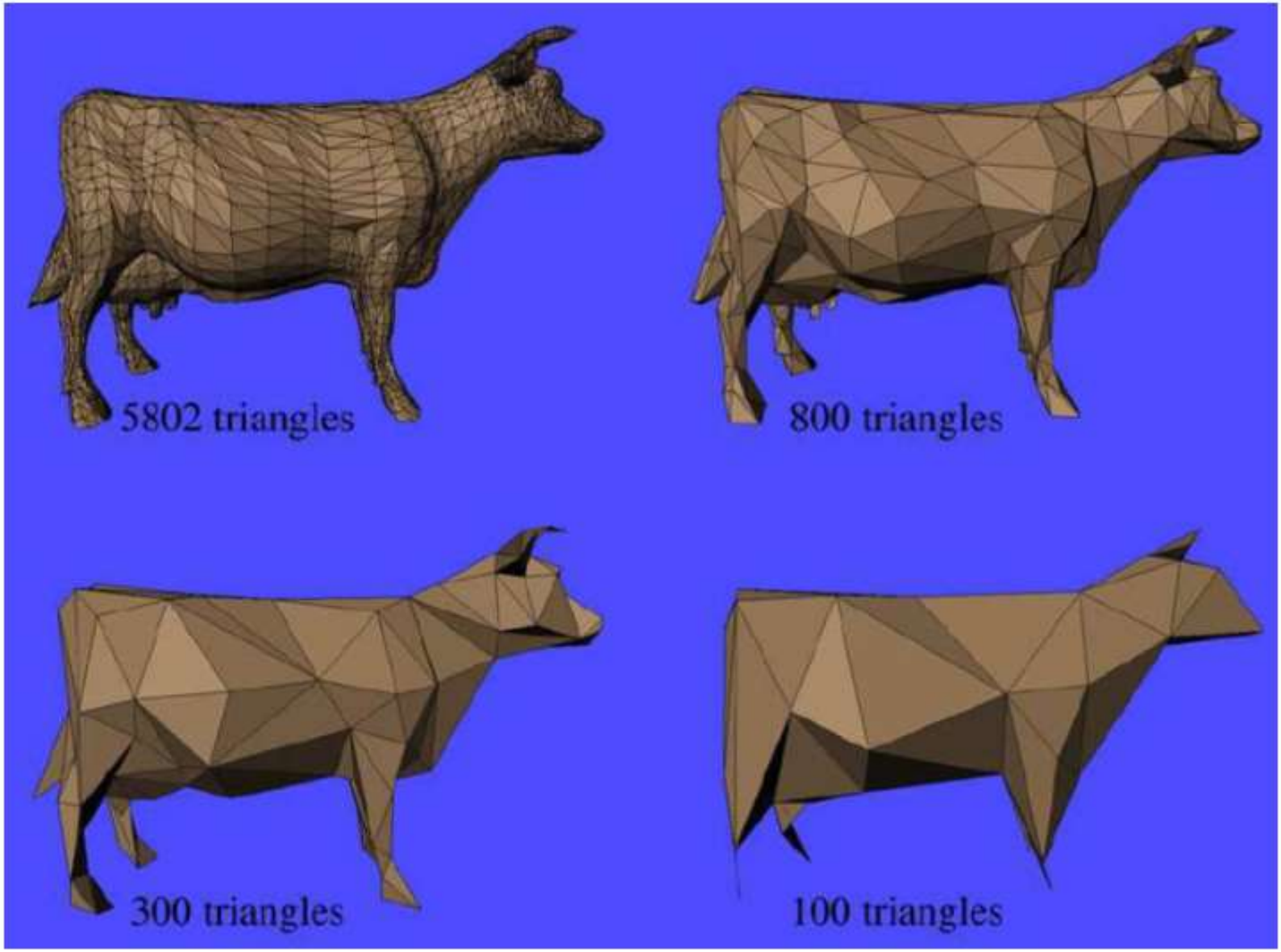
- Las mallas de polígonos se utilizan para representar superficies de objetos
- Aplicaciones en:
  - Generación de una malla de volumen (tetraedros, hexaedros, o mixta)
  - Visualización en juegos y entretenimiento
  - Modelación de terrenos
  - Simulación numérica en aplicaciones científicas y de ingeniería





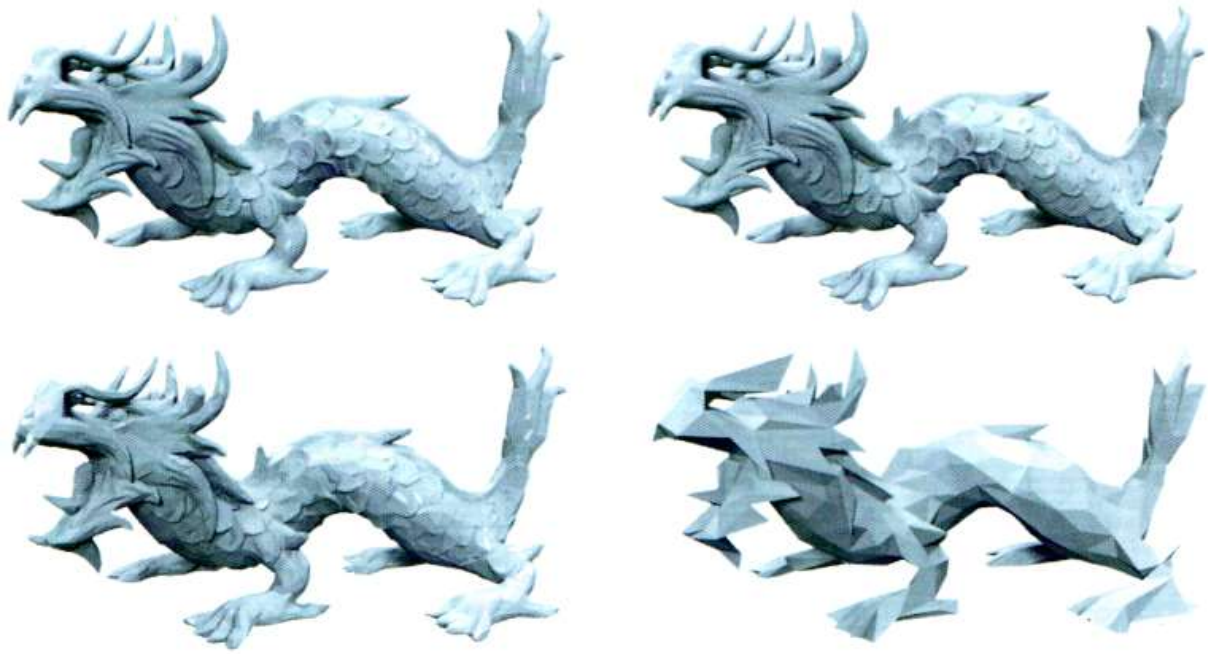


Cuántos polígonos utilizar?



Distintos niveles de detalle

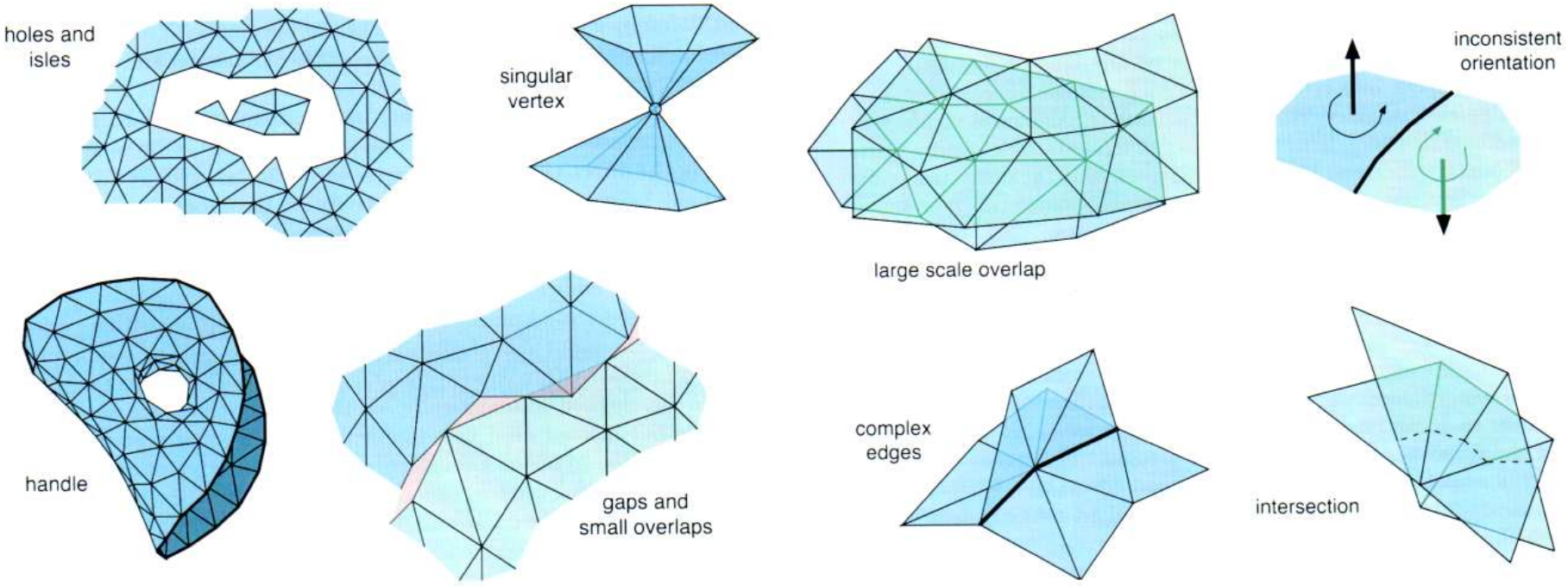
- Diferentes modelos para objetos cercanos y lejanos
- Diferentes modelos para renderizar y para detectar colisiones
- Compresión de datos obtenidos del mundo real (sensores raster)



Problemas de las mallas geométricas

- Se necesitan muchos polígonos para representar figuras suaves
- Se necesitan muchos polígonos para representar detalles
- Difíciles de editar
- Se necesita mover cada uno de los vértices
- Test de intersección o de dentro/fuera son difíciles de implementar
- Es fácil romper ciertas garantías geométricas

The freak show



Estructuras de Datos para Mallas

Porqué utilizar estructuras de datos?

- Permiten validar la geometría
- Usualmente las librerías gráficas asumen que las mallas son correctas y conexas
- Una malla es conexa cuando no hay vértices, arcos ni caras aisladas
- Consultas tipo:
  - Pertenece cada vértice extremo a al menos dos arcos?
  - Pertenece cada arco a una o dos caras?
  - Es cada cara cerrada?
  - Tiene cada cara al menos un arco compartido?
  - Cada arco debe referenciar al menos a una cara
- Una buena estructura de datos permite agilizar dichas consultas y validaciones

Operaciones frecuentes

- Acceder a vértices, arcos y caras individualmente
- Recorrer ordenadamente los arcos que definen cada cara
- Acceder a las caras incidentes de un arco
- Dado un arco, podemos acceder a sus dos vértices
- Dado un vértice, una cara o arco debe ser accesible

- Estas operaciones permiten recorrer local y globalmente la malla

Face-based data structures

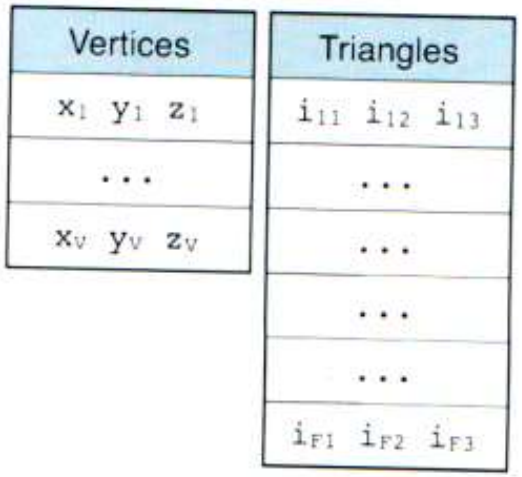
- La forma más simple de representar una malla poligonal corresponde a la especificación de caras por las posiciones de sus vértices
- OpenGL puede renderizar una malla en 3D utilizado solo esta información
- Formato conocido: STL

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
...	...	...
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

- No representa conectividad entre las distintas caras poligonales
- Pueden existir problemas de precisión numérica, el mismo vértice es anotado una y otra vez.
- Esta representación no es apropiada generalmente

Face-based data structures

- Podemos evitar repetir vértices una y otra vez definiendo triángulos con índices a una tabla de vértices



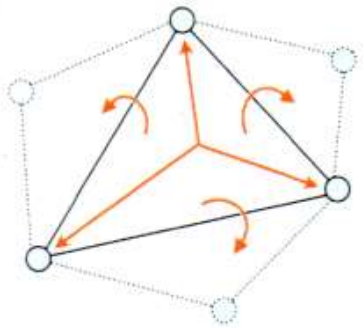
- Esta representación es usada por formatos como OFF, OBJ, VRML
- OpenGL puede trabajar con este tipo de estructura también
- Como no hay información de conectividad, las consultas son costosas

Face-based data structures

- Podemos añadir información de conectividad
- Para cada cara se debe almacenar:
  - Referencias a sus tres vértices
  - Referencias a sus triángulos vecinos
- Para cada vértice se debe almacenar
  - Referencia a una de sus caras incidentes
  - Sus tres coordenadas

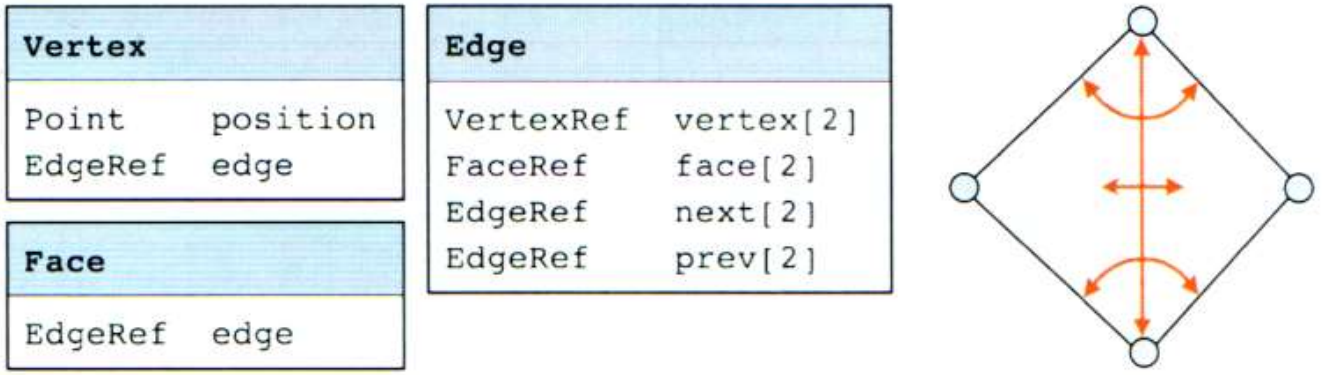
Vertex	
Point	position
FaceRef	face

Face	
VertexRef	vertex[3]
FaceRef	neighbor[3]



Edge-based data structures

- La conexión más básica ocurre en los arcos
- Estructuras bien conocidas de este tipo son: Winged-Edge y Quad-Edge



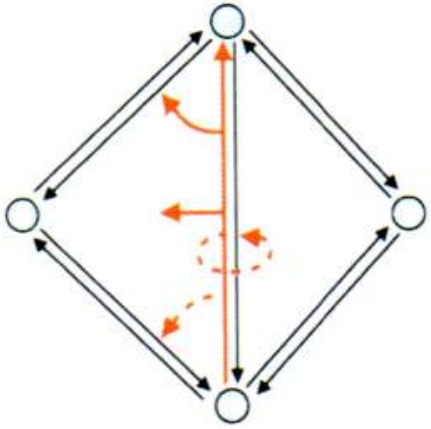
Halfedge-based data structures

- Cada arco, inicialmente sin orientación, es dividido en 2 sub-arcos orientados
- De ahí el nombre *halfedge*
- Puede representar mallas de polígonos arbitrariamente complejas que sean orientables y 2-manifold
- Los halfedges son orientados CCW (counter-clockwise) alrededor de cada cara y a lo largo de cada borde

Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
HalfedgeRef	prev
HalfedgeRef	opposite

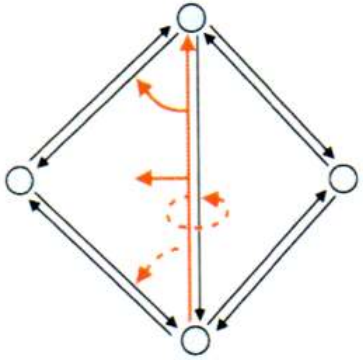


Halfedge-based data structures

Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

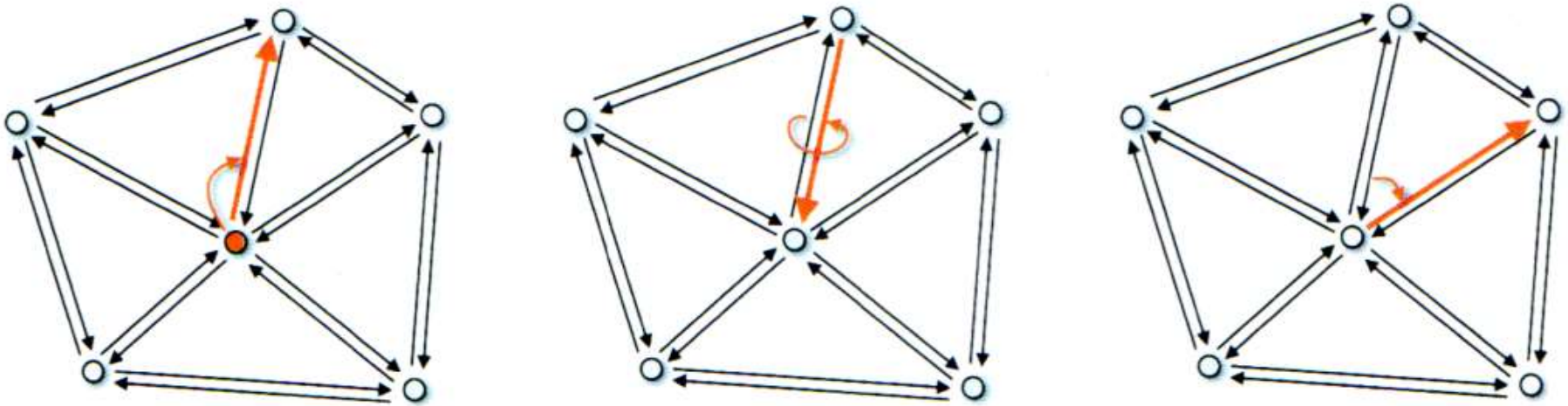
Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
HalfedgeRef	prev
HalfedgeRef	opposite



- Cada halfedge referencia una única esquina, un vértice no compartido en la cara
- Permite fácilmente añadir metadata a cada vértice como coordenadas de textura o normales
- Si los arcos opuestos están almacenados en pares, podemos omitir la referencia al halfedge opuesto.

Halfedge-based data structures

- Esta estructura permite enumerar el one-ring neighborhood de manera eficiente



Mallas orientables

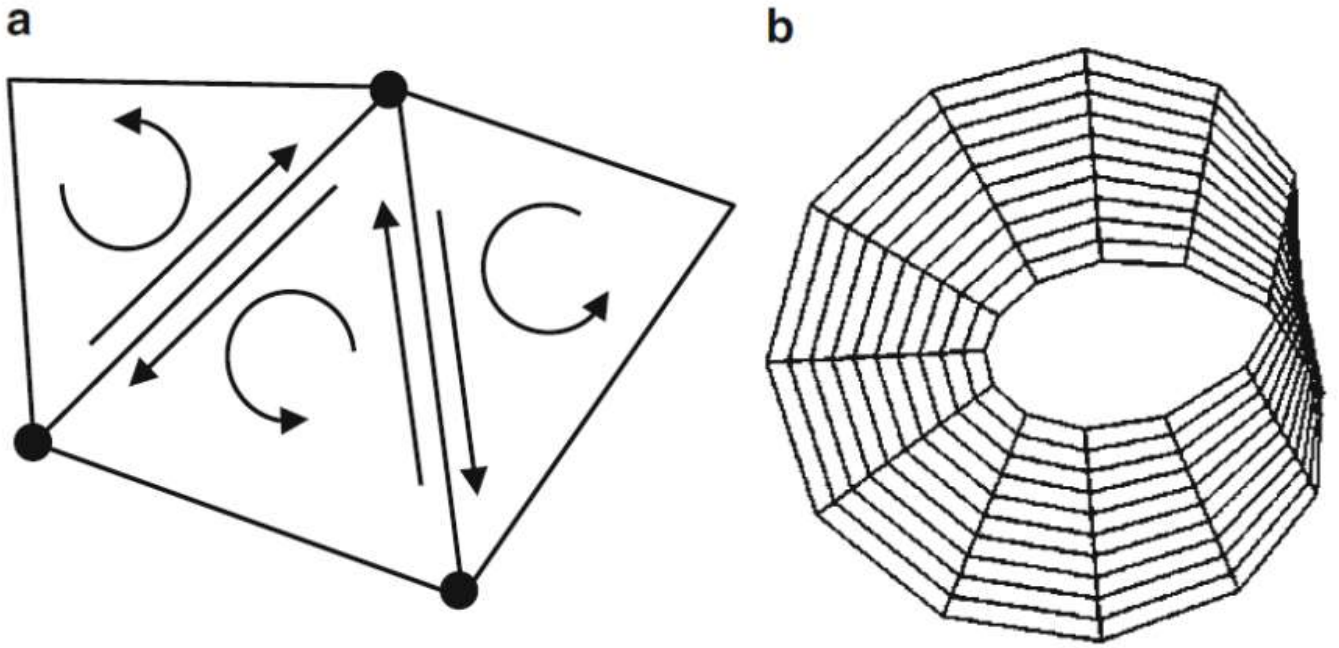


Fig. 8.7 (**a**) Compatible faces in an orientable mesh. (**b**) The Möbius strip is an example of a non-orientable mesh

Elección de una estructura de datos

- La eficiencia y consumo de memoria de los algoritmos geométricos dependen en gran medida de la estructura de datos subyacente
- La elección de la estructura de datos debe considerar:
  - Requerimientos topológicos: representaremos solamente superficies cerradas o modelos más complejos? Sólo triángulos o polígonos más complejos? La malla será regular, semi-regular o irregular? Modelaremos una jerarquía de objetos?
  - Requerimientos algorítmicos: Queremos solamente renderizar la malla o necesitamos acceso eficiente a la malla? Cambiará en el tiempo? Necesitamos asociar datos a cada vértice, cara?Qué limitaciones de memoria tenemos?
- Construir una estructura de datos requiere pre-procesamiento de información.

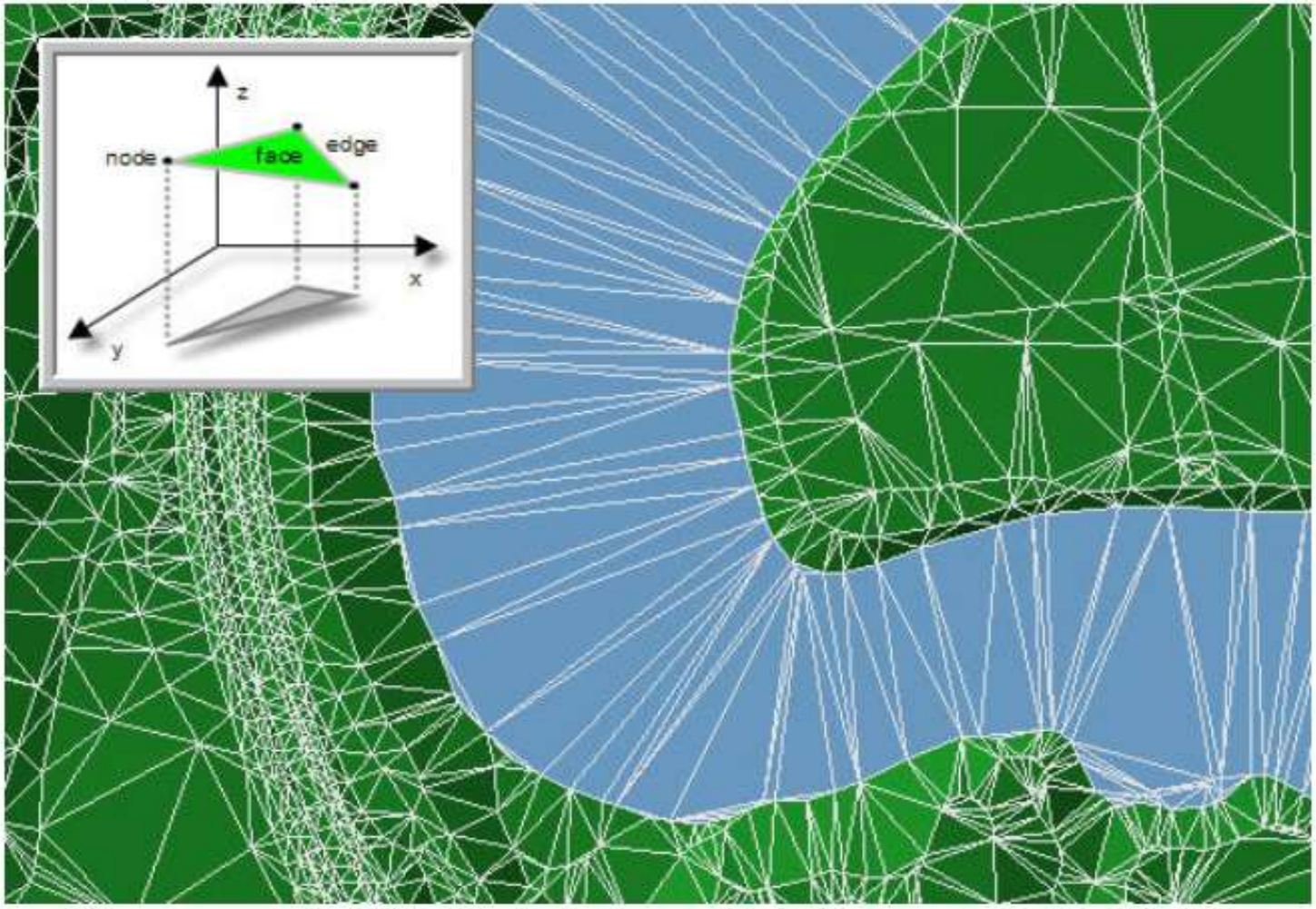
Elección de una estructura de datos

- Criterios a considerar:
  - Tiempo de construcción
  - Tiempo para ejecutar determinada consulta
  - Tiempo para ejecutar determinada operación
  - Consumo de memoria y redundancia

Modelación de Terrenos

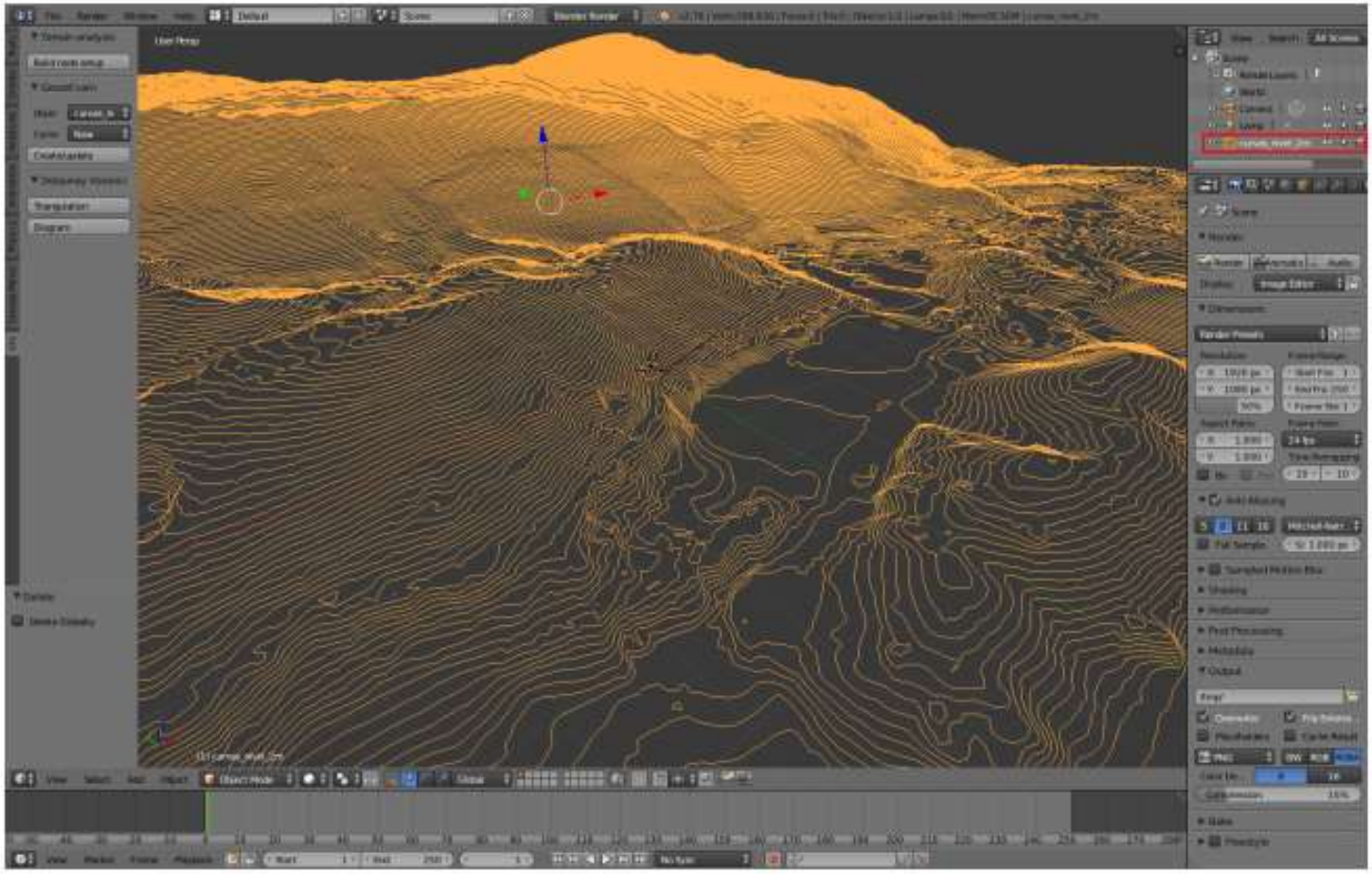
Link: Grass GIS

Modelación de Terrenos



Link: TIN en ArcGIS

Modelación de Terrenos



Link: Modelación de Terrenos en Blender

Modelación de Terrenos

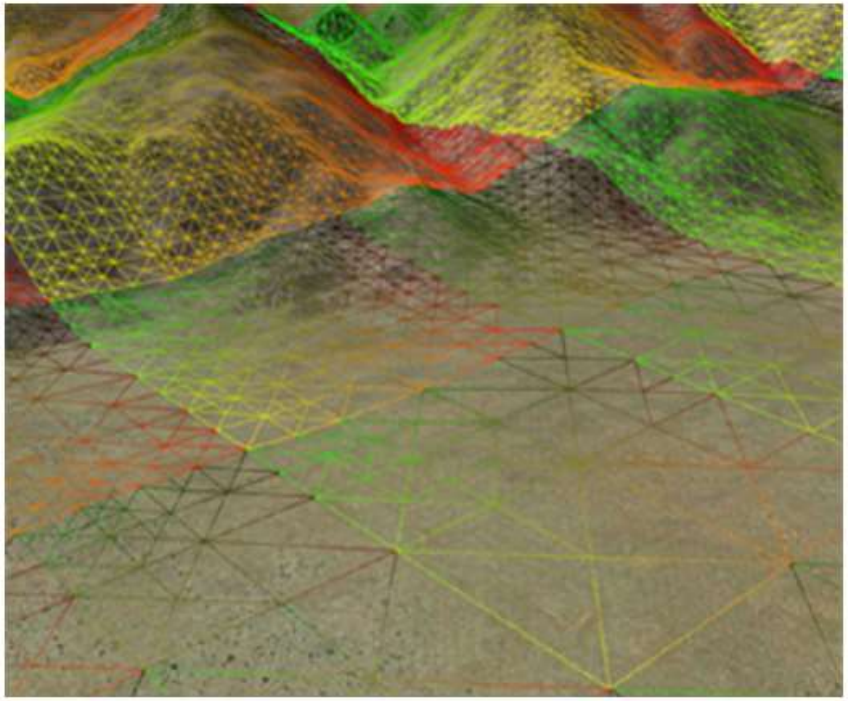
- Un terreno puede ser modelado como una función escalar en el plano
- El valor escalar se asocia a la elevación del punto (x, y)

$$z = h(x, y)$$

- Se dice que esta configuración es de $2\,{}^{1}/_{2}$ dimensiones
- Es posible modelar terrenos con mallas de distintos tipos de figuras
- Las elecciones más comunes son triángulos y cuadriláteros
- La malla puede ser:
  - Regular (uniforme): todos los triángulos son del mismo tamaño
  - No regular (no uniforme): triángulos son de distinto tamaño

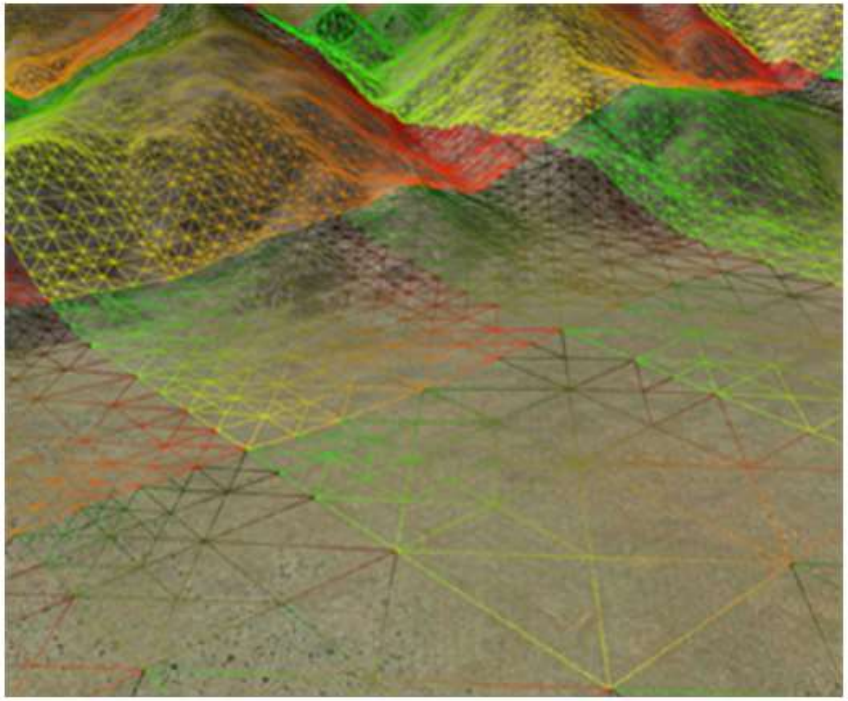
Modelando Terrenos con triangulaciones

- Algoritmo 1
  - Entrada: Datos equi-espaciados en x e y
  - Generar grilla rectangular
  - Dividir en triángulos
  - Calcular la elevación de los puntos dada por $h(x, y)$
  - Eliminar triángulos en zonas de altura parecida



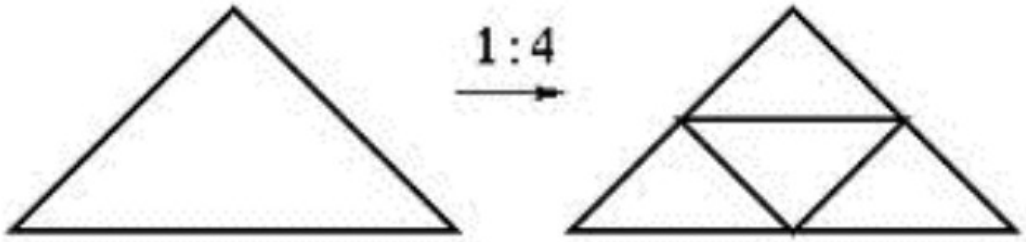
Modelando Terrenos con triangulaciones

- Algoritmo 2
  - Entrada: Datos equi-espaciados en x e y
  - Generar el rectángulo más grande
  - Dividir en dos triángulos
  - Dividir en triángulos mas pequeños si el terreno no posee similar elevación

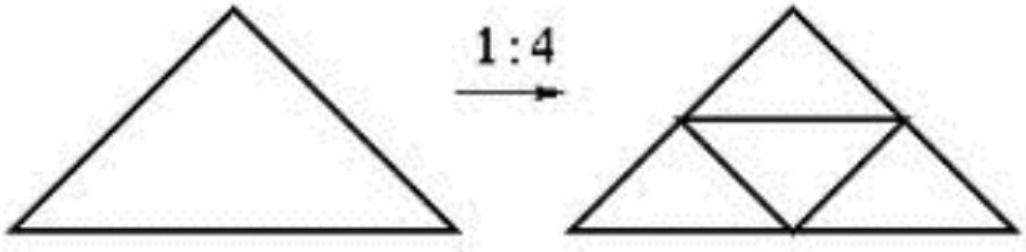


Fractales para modelar terrenos

- Método de síntesis artificial de terrenos
- Cada vértice del triángulo tiene asignada una elevación $h(x, y)$
- Cada triángulo puede subdividirse en 4 triángulos iguales usando los puntos medios



Fractales para modelar terrenos



- Podemos considerar como elevación de los nuevos vértices el promedio de las elevaciones de los vértices originales
- Se suma (o se resta) un valor aleatorio
- Se itera ejecutando este procedimiento para cada uno de los triángulos generados

Fractales para la modelación de terrenos

- Se obtienen terrenos de apariencia muy realista
- Es posible utilizar técnicas mixtas
- Ejemplo: Considerar una función de elevación $h(x, y)$ simple y luego añadir perturbaciones con la estrategia fractal descrita