# 2 Representación poligonal

Introducción
Modelo poligonal
Teselación
Simplificación

# Introducción

- Modelado geométrico
  - Creación del modelo 3D en el ordenador
  - Técnica de representación
    - Superficies (polígonos, bicúbicas, …)
    - Sólidos (CSG, Octrees, …)
  - Manipulación de la representación (Edición)
    - Difícil para polígonos
    - Fácil para superficies y Sólidos

# Introducción

- Revisión de modelos

**POLÍGONOS**
Son el presente
Buenos para el Hw
Fácil creación
Modelo aproximado
Muchos polígonos (uso de texturas)
Difícil manipulación y animación

**SUP. BICÚBICAS**
El futuro
Control del LOD
Bajo coste espacial
Fácil edición
Problemas de continuidad

**IMPLÍCITAS**
Cálculo de intersecciones
Blobs
Difícil manipulación

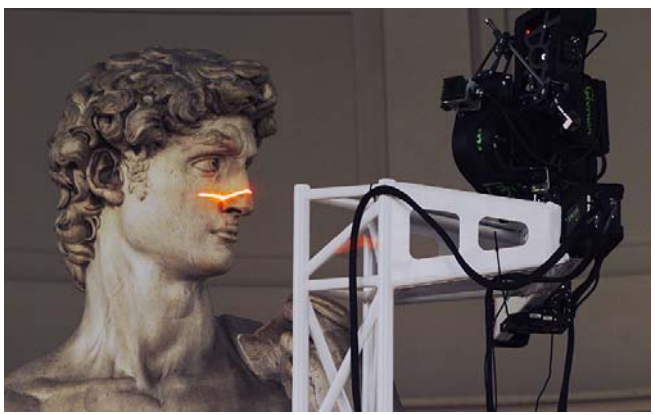
**CSG**
Bueno para el usuario
Modelado restringido

**SUBDIVISIÓN ESPACIAL**
Útil en algunas aplicaciones
Alto coste

Modelado contextual vs general

# Modelo poligonal

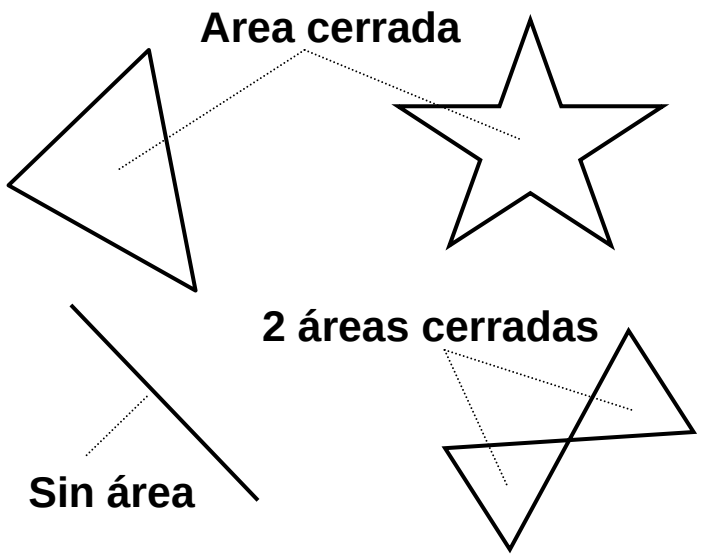
- Formas de adquisición
  - Definida por el usuario
    - Utilizando software de modelado geométrico (Blender, 3DStudio, Maya, …)
    - Escribiendo un programa
  - Escaneando un objeto real
  - Reconstrucción basada en imágenes

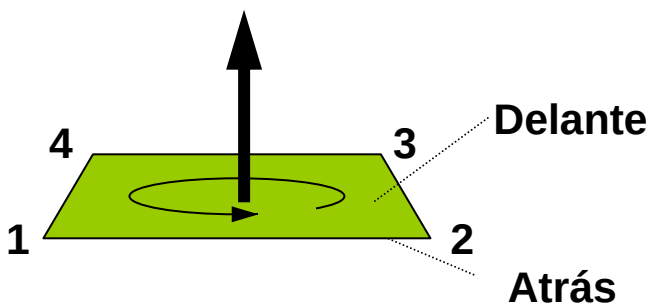


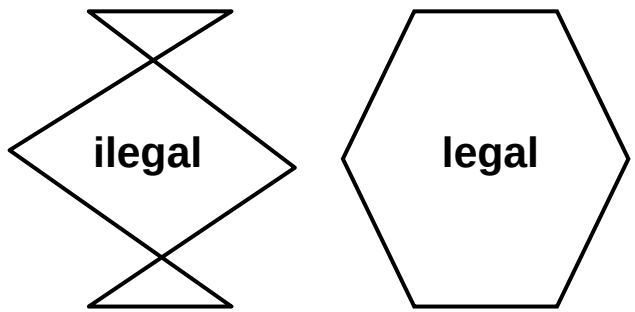
56 millones de polígonos

# Modelo poligonal

- Polígono
  - Conjunto de líneas rectas (arcos) que no se cruzan y que unen un conjunto coplanar de puntos (vértices) definiendo un área simple (habitualmente convexa y sin agujeros)



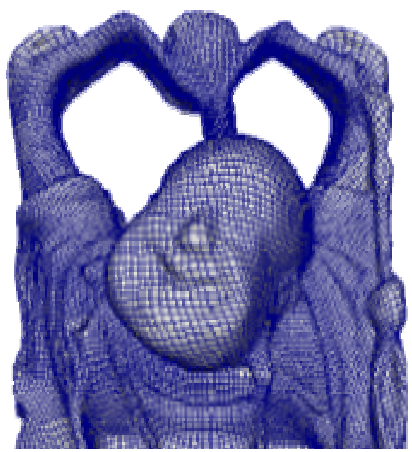



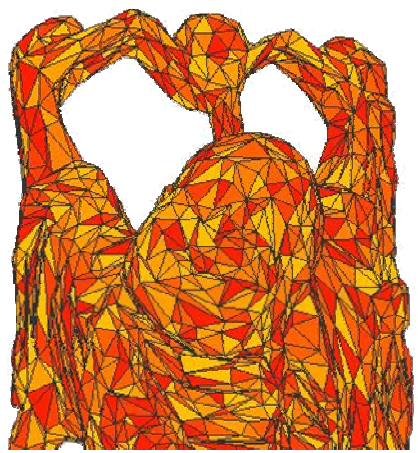
# Modelo poligonal

- Malla de polígonos
  - colección de vértices, aristas y polígonos conectados de forma que cada arista es compartida como máximo por dos polígonos (manifold)
    - vértice: punto de coordenadas (x,y,z)
    - arista: segmento de línea que une dos vértices
    - polígono: secuencia cerrada de aristas
  - Ley de Euler

$$F - E + V - L = 2(B - G)$$

- Tipos de representaciones
  - Explícita
  - Lista de vértices
  - Lista de aristas
- Tipos de mallas
  - Cuadriláteros
  - Triángulos



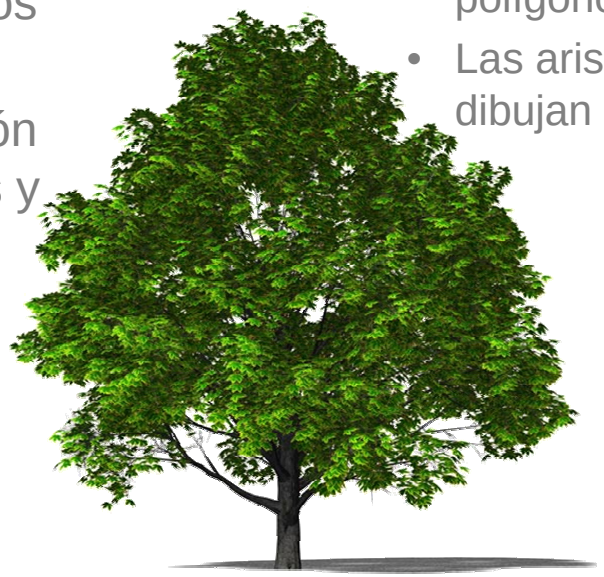


# Modelo poligonal

- Explícita
  - Polígono = lista de coordenadas de vértices

$$P = ((x_1, y_1, z_1), \ldots, (x_n, y_n, z_n))$$

  - Los vértices se almacenan en orden (horario o antihorario)
  - Los vértices compartidos están duplicados
  - No existe representación explícita de los vértices y aristas compartidas

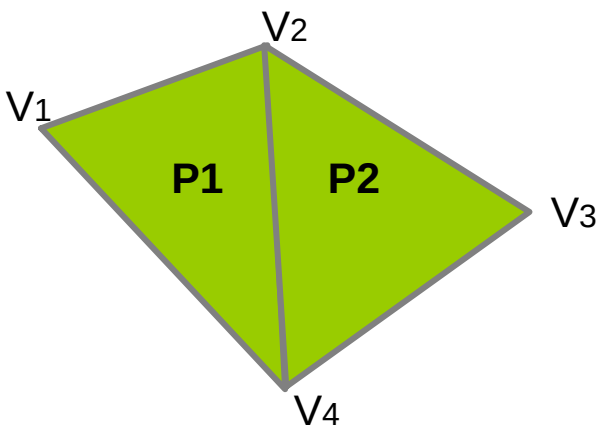


  - Ventajas:
    - Representación eficiente para polígonos individuales
  - Problemas:
    - Alto coste de almacenamiento
    - Para mover un vértice es necesario recorrer todos los polígonos
    - Las aristas compartidas se dibujan dos veces

# Modelo poligonal

- Lista de vértices
  - Polígono = lista de índices a la lista de vértices
  - Cada vértice se almacena una sola vez en una lista

$$V = ((x_1, y_1, z_1), ..., (x_n, y_n, z_n))$$



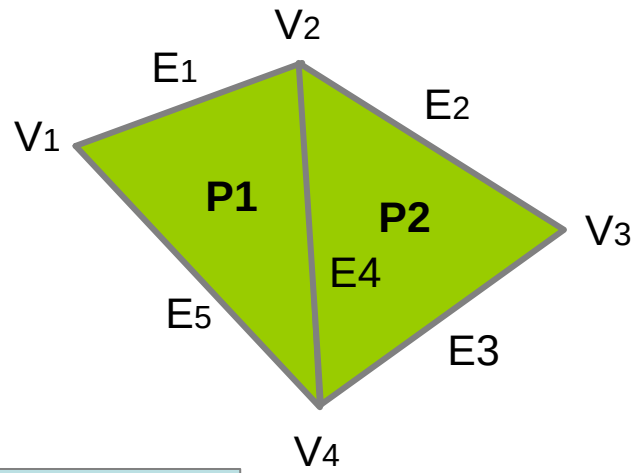

  - Ventajas
    - Cada vértice se almacena una sola vez
    - Las coordenadas de los vértices pueden cambiarse fácilmente
  - Problemas
    - Difícil encontrar polígonos que compartan una arista
    - Las aristas compartidas se siguen dibujando dos veces

$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), ..., (x_4, y_4, z_4))$$

$$P_1 = (1,2,4) \qquad P_2 = (4,2,3)$$

# Modelo poligonal

- Lista de aristas
  - Polígono = lista de índices a una lista de aristas
  - Se mantiene la lista de vértices
  - Cada arista apunta a dos vértices y a los polígonos a los que pertenece
  - Ventajas
    - Cada vértice se almacena una sola vez
    - Las aristas compartidas se dibujan una sola vez
  - Problema
    - Difícil determinar que aristas comparten un vértice (en todas las representaciones)




$$E_1 = (V_1, V_2, P_1, \lambda)$$

$$E_2 = (V_2, V_3, P_2, \lambda)$$

$$E_3 = (V_3, V_4, P_2, \lambda)$$

$$E_4 = (V_4, V_2, P_1, P_2)$$

$$E_5 = (V_4, V_1, P_1, \lambda)$$

$$P_1 = (E_1, E_4, E_5)$$

$$P_2 = (E_2, E_3, E_4)$$

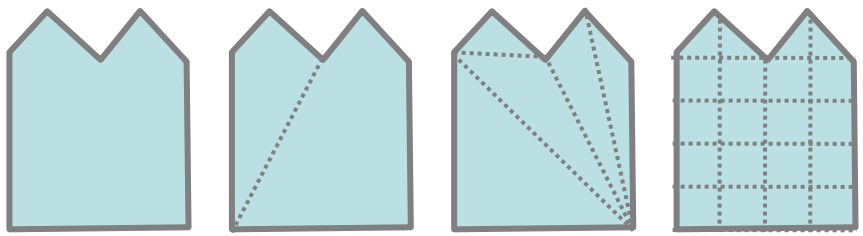
$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4))$$

# Modelo poligonal

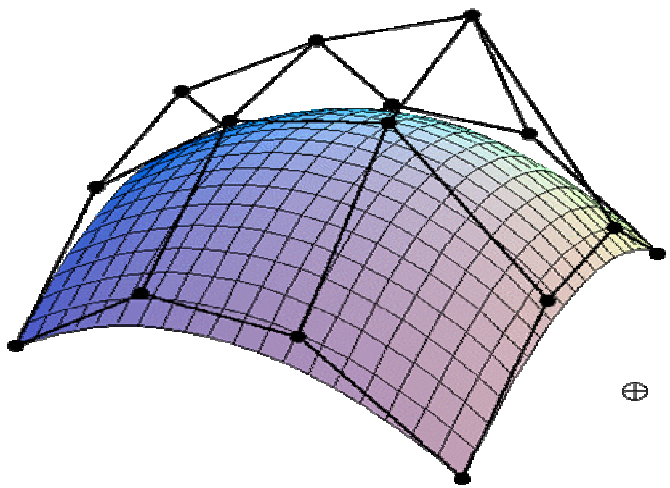
- Otros atributos
  - Normales
    - Por cara (flat shading). Se almacena una normal por cada cara
    - Por vértice (smooth shading). Una lista de normales similar a la lista de vértices
  - Texturas
    - Se asignan coordenadas de textura por cada vértice
  - Coeficientes iluminación
    - Pueden ser por objeto, por cara o por vértice
  - Mapas de iluminación
  - Etc.

# Teselación

- Definición
  - Proceso de dividir una superficie en un conjunto de polígonos
    - Teselación de polígonos (conversión a triángulos)

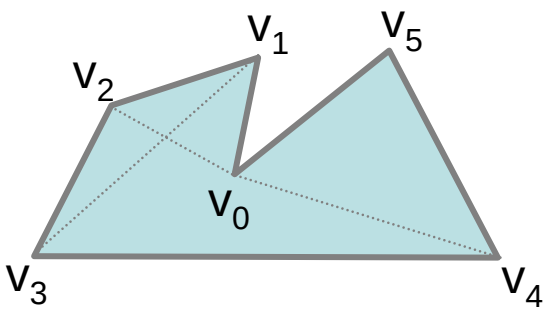


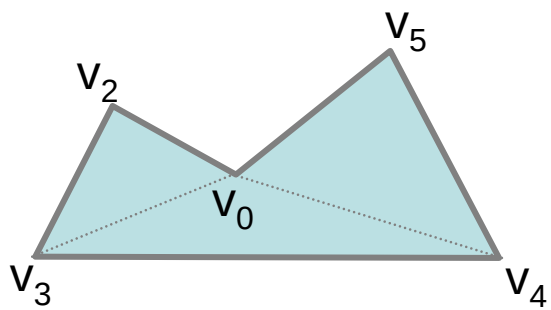
    - Teselación de superficies curvas



# Teselación

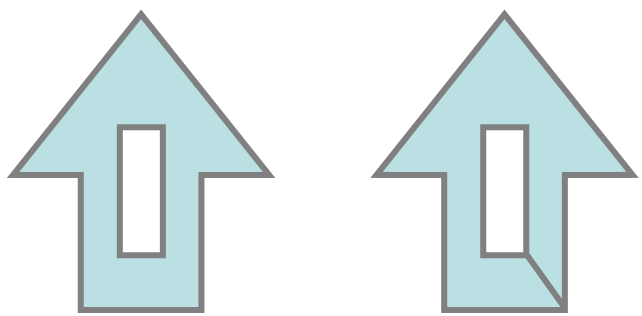
- Teselación de polígonos
  - Fuerza bruta $O(n^3)$
    - Examinar cada segmento que une dos vértices
    - Si el segmento intersecta algún otro del polígono no se puede usar
    - Sino se divide el polígono y se repite el proceso con los dos triángulos resultantes
  - *Ear clipping* $O(n^2)$
    - Primer paso. Encontrar todos los triángulos con vértices (i, i+1, i+2) y detectar si el segmento (i, i+2) corta algún arco del polígono
    - Segundo paso. Eliminar todos los triángulos encontrados y recalcular si necesario
  - Métodos más complicados en O(n log n)



# Teselación

- Tipos de polígonos
  - Convexos
  - Cóncavos
    - Dividir un polígono en zonas convexas suele ser más eficiente (en términos de visualización y almacenamiento) que dividirlo en triángulos
    - Esto es debido a que un polígono convexo puede representarse fácilmente con *fans* o *strips*
  - Polígonos con varias fronteras
    - Se pueden convertir a polígonos con una frontera incluyendo arcos de unión



# Teselación

- Teselación de superficies curvas
  - Uniforme. Subdivisión regular del espacio paramétrico (u, v)
    - *Forward differencing*
      - Evalúa rápidamente funciones polinómicas en intervalos uniformes
      - Problemas numéricos. Todos los valores se calculan con sumas.
      - Ideal para futuras implementaciones HW (Moreton)

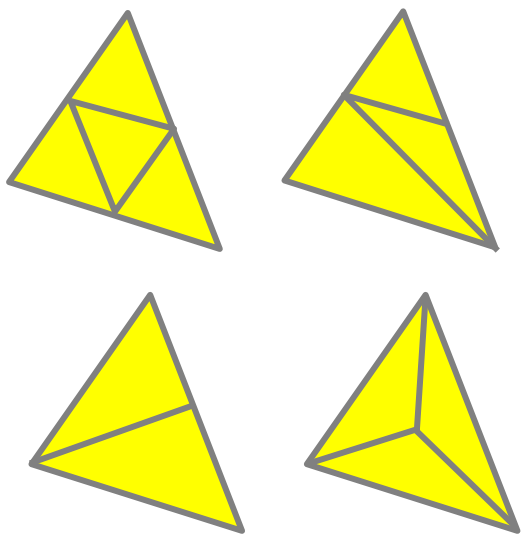
$$P(u) = \sum_{i=0}^{n} c_i u^i = c_0 + c_1 u + c_2 u^2 + c_3 u^3$$

$$\delta_i^1 = P_{i+1} - P_i$$
$$\delta_i^2 = \delta_{i+1}^1 - \delta_i^1$$
$$\delta_i^3 = \delta_{i+1}^2 - \delta_i^2$$

```
Calcular P,d¹,d²,d³
Para i = 1 to k
  P=P+d¹
  d¹=d¹+d²
  d²=d²+d³
```

# Teselación

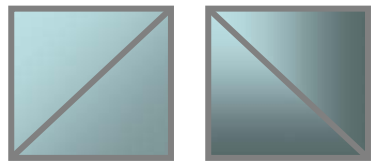
- Adaptativa
  - Subdividen según la curvatura, la longitud de los arcos o el tamaño del pixel
  - Problemas: agujeros entre distintos niveles de subdivisión
  - Ejemplos:



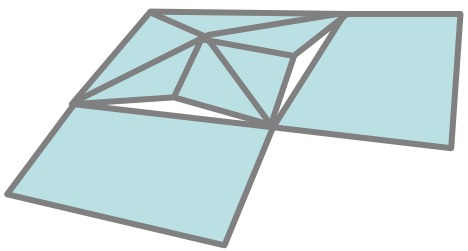
```
teselar(p,q,r)
  PQ=divide(p,q)
  QR=divide(q,r)
  RP=divide(r,p)
  if(PQ and QR and RP)
    teselar(p,(p+q)/2,(p+r)/2)
    teselar(q,(q+r)/2,(q+p)/2)
    teselar(r,(r+p)/2,(r+q)/2)
    teselar((p+q)/2,(q+r)/2,(r+p)2)
  else if(PQ and QR)
    teselar(p,(p+q)/2,r)
    teselar((p+q)/2,(q+r)/2,r)
    teselar((p+q)/2,q,(q+r)/2)
  else if(PQ)
    teselar(p,(p+q)/2,r)
    teselar(q,r,(p+q)/2)
  else if(divideTri(p,q,r))
    teselar((p+q+r)/3,p,q)
    teselar((p+q+r)/3,q,r)
    teselar((p+q+r)/3,r,p)
//no están todos los casos
```

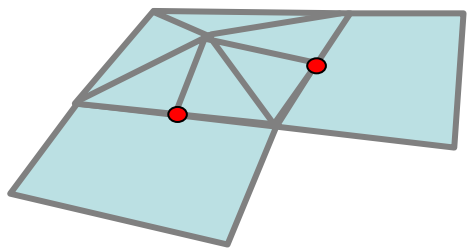
# Teselación

- Problemas
  - Orientación
  - División de cuadriláteros
    - Sombreado. Dividir por los vértices con colores similares
    - Texturas. Problemas con la interpolación
  - Superficies curvas
    - Agujeros y T-vértices. Debido a distintas subdivisiones entre retazos



Agujeros





T-vértices

# Simplificación

- El problema
  - Modelos de gran complejidad geométrica
    - Bases de datos CAD
    - Escáner de gran precisión
  - Los recursos siempre son limitados
    - CPU, espacio, velocidad del HW gráfico, ancho de banda de la red
  - Se necesitan modelos simplificados
    - Simplificación manual
    - Simplificación automática

50.761 triángulos

10.000 triángulos

# Simplificación

- Objetivo
  - Dada una malla a alta resolución encontrar una aproximación con menos polígonos preservando su apariencia
  - Aplicaciones escalables (diferentes arquitecturas y plataformas)
  - Adecuar la representación a la importancia del objeto en la escena

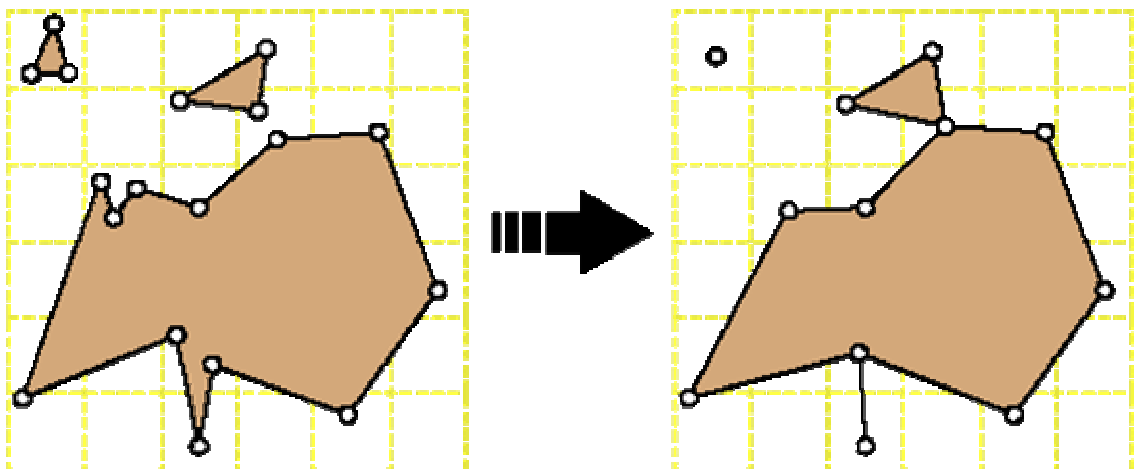
**543.644 vertices** **43.644 vertices** **5.437 vertices**

# Simplificación

- Propiedades deseables de un algoritmo de simplificación
  - Rapidez (no en tiempo real)
  - Calidad de la aproximación (en algún sentido)
  - Manejo de distintos tipos de mallas de entrada
- Clasificación
  - Tipo de operación de simplificación
    - ¿Puede cambiar la topología?
    - ¿Funciona para todo tipo de superficies?
  - Métrica utilizada en el proceso
    - Geométrica (más sencilla de calcular)
    - Basada en imágenes (apariencia visual)

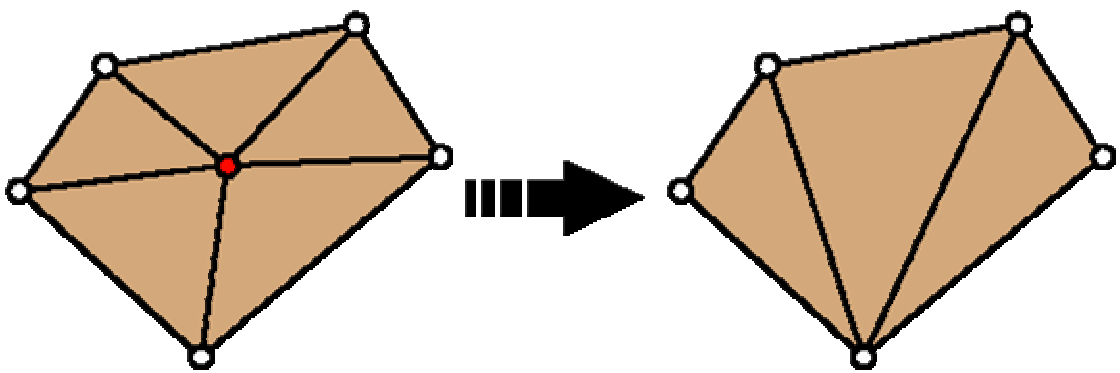
# Simplificación

- Operaciones
  - Agrupación de vértices (vertex clustering)
    - Unir vértices próximos
    - Partición del espacio en celdas (rejillas, esferas, cajas, …). Unir todos los vértices de la misma celda
    - Triángulos con varios vértices en el grupo quedan degenerados
    - Método general y robusto
    - No muy atractivo



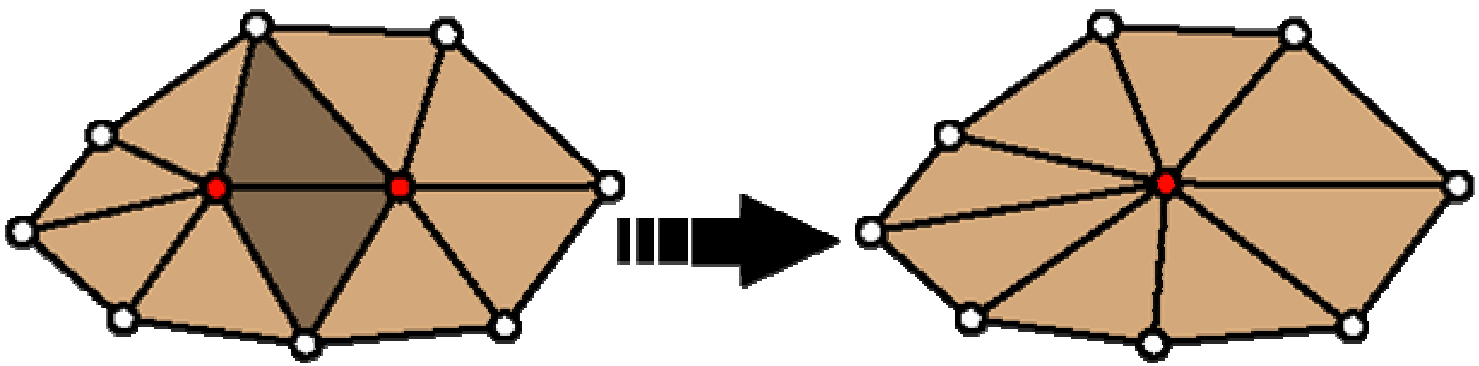
# Simplificación

- Eliminación de vértices
  - Ordena los vértices según su importancia
  - En el proceso de simplificación
    - Se elimina el vértice menos importante y sus triángulos
    - Aparecen nuevos triángulos (teselación)
  - Superficies manifold
  - Preserva la estructura topológica local
  - Método muy utilizado



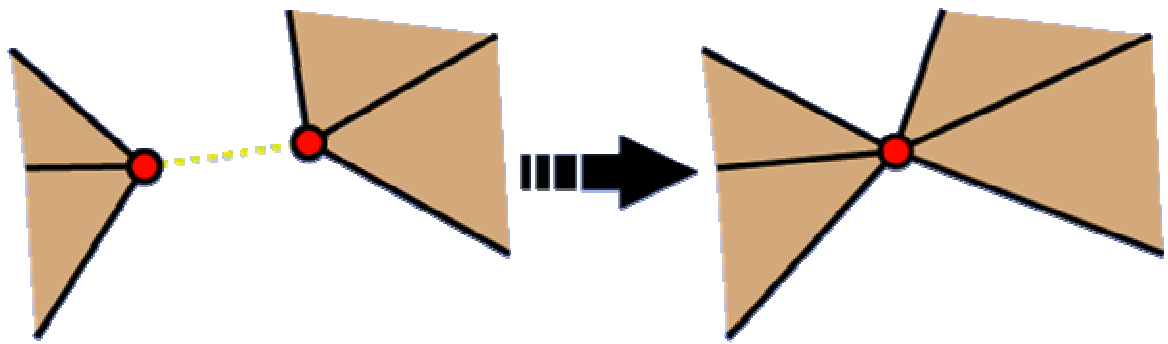
# Simplificación

- Colapso de aristas
  - Une los dos vértices de una arista
    - Puede cambiar la posición y los atributos del vértice
  - Proceso
    - Ordena todos los arcos según su coste
    - Contrae el arco con menor coste
    - Actualiza el coste de los arcos involucrados
    - Elimina los 2 triángulos degenerados en cada paso
  - Permite transiciones suaves de la simplificación (geomorphing)



# Simplificación

- Contracción de pares de vértices
  - Unir cualquier par de vértices
    - Basándose en la geometría, la topología, etc.
  - Más flexible que el colapso de aristas
  - Mas control local que el agrupamiento de vértices



# Simplificación

- Métricas
  - Geométricas
    - Heurísticas simples
      - Longitud de las aristas, ángulos, área, …
    - Distancias a la superficie original
    - Ej: Medida del error con planos
      - Error = suma de las distancias al cuadrado de los planos del conjunto

$$Error(v) = \sum_i (n_i^T v + d_i)^2$$

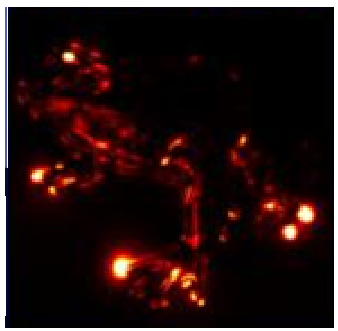
      - Inicialización con planos de las caras incidentes (Inicialmente el error es 0)
      - Cuando se elimina una arista, se utiliza la unión de planos de los dos vértices

$$planos(v') = planos(v_1) \cup planos(v_2)$$

# Simplificación

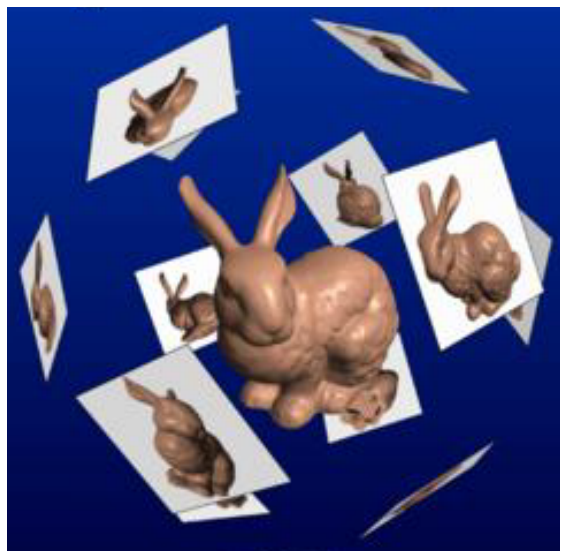
- Basadas en imágenes

Original



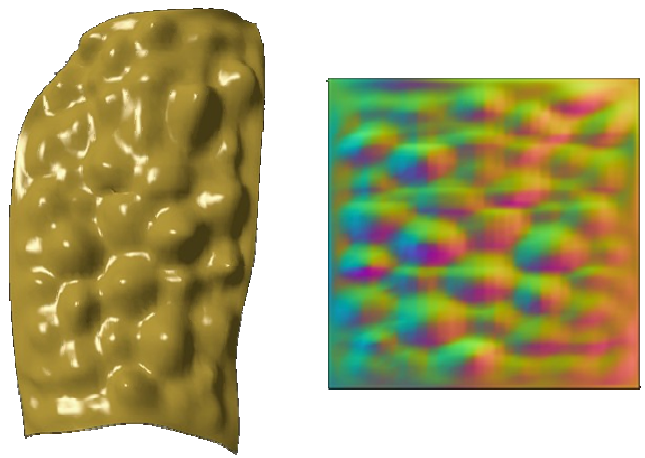
Diferencia

Simplificado



# Simplificación

- Preservando la apariencia
  - Utilización de mapas de normales
    - Normales en el espacio del objeto
    - Constante a medida que el objeto se simplifica (misma textura para todos los niveles de detalle)
    - Mejor que utilizar Bump-Mapping
      - Diferentes texturas para cada nivel de detalle
  -



# Simplificación

– Ejemplo. Diferentes niveles de detalle con y sin mapas de normales