Mallas y procesamiento

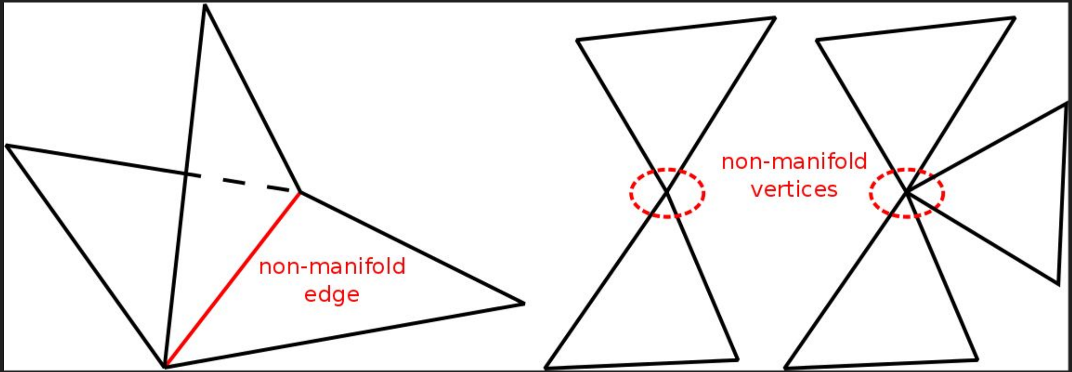
mleguen@gmail.com

Mesh topology

- Vertex : 0-simplex
- Arista : 1-simplex
- Triángulo : 2-simplex
- El grado de un vértice (degree/valence) representa la cantidad de aristas que lo comparten
- Un vértice es adyacente a las aristas que lo comparten
- El grado de una arista representa cuántas caras la comparten. (1 : arista de frontera, 2 : arista interior)

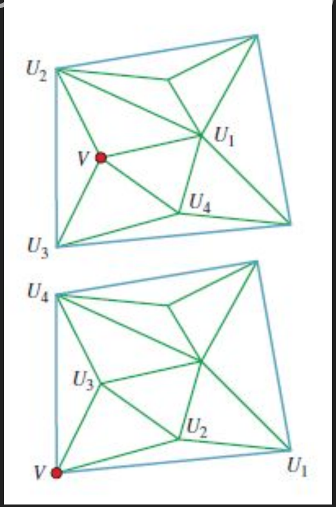
Manifold Mesh

- Una malla non-manifold es una malla que no puede existir en la realidad
- Por cada arista son una o dos caras que la comparten.
- Los vértices no puede ser la unión de dos triángulos que no comparten aristas.



Manifold Mesh

- Cada cara solo aparece una vez
- El grado de un vértice es superior a 2.
- v un vértice
  - los vértices que comparten una arista con v pueden ser ordenados, U1,U2,U3, ... , Un, de tal manera que (V,U1,U2) , (V,U2,U3), (V,Un-1,Un) son triángulos
  - Si (V,U1,Un) es un triángulo entonces V es interior
  - Si (V,U1,Un) no un triángulo V es frontera



Mallas - Estructura

- Estructura directa :
  - Lista de vértices
  - Lista de índices (triángulos)
- Esta estructura es práctica para mandar los datos de una malla a OpenGL.
- Cada vértice también tiene que tener una información sobre su normal y sus coordenadas de textura.

Cube.obj

```
v  0.0  0.0  0.0
v  0.0  0.0  1.0
v  0.0  1.0  0.0
v  0.0  1.0  1.0
v  1.0  0.0  0.0
v  1.0  0.0  1.0
v  1.0  1.0  0.0
v  1.0  1.0  1.0

vn  0.0  0.0  1.0
vn  0.0  0.0 -1.0
vn  0.0  1.0  0.0
vn  0.0 -1.0  0.0
vn  1.0  0.0  0.0
vn -1.0  0.0  0.0

vt 0.25 0.0
vt 0.5  0.0
vt 0    0.25
vt 0.25 0.25
vt 0.5  0.25
vt 0.75 0.25
```

```
vt 0.0  0.5
vt 0.25 0.5
vt 0.5  0.5
vt 0.75 0.5
vt 0.25 0.75
vt 0.5  0.75
vt 0.25 1.0
vt 0.5  1.0

f  1/11/2  7/14/2  5/12/2
f  1/11/2  3/13/2  7/14/2
f  1/7/6  4/4/6  3/3/6
f  1/7/6  2/8/6  4/4/6
f  3/1/3  8/5/3  7/2/3
f  3/1/3  4/4/3  8/5/3
f  5/10/5  7/6/5  8/5/5
f  5/10/5  8/5/5  6/9/5
f  1/11/4  5/12/4  6/9/4
f  1/11/4  6/9/4  2/8/4
f  2/8/1  6/9/1  8/5/1
f  2/8/1  8/5/1  4/4/1
```

Mallas - Estructura

- Problema : Con esta estructura de datos procesar la malla implica realizar algoritmos con complejidad algorítmica muy alta.
- Ejemplo : Queremos buscar los triángulos que comparten un vértice *v*.

```
//tengo un vertice v
for(int i=0; i< nfaces;++i)
     for(int j=0;j<3;++j)
          if(v.id == faces[i*3+j])
          //procesar el triangulo
          //faces[i*3]
```

- Para buscar los triángulos de vecindario de cada vértice tenemos que realizar un algoritmo en O(n*m)
- n cantidad de vértice
- m cantidad de triángulos*3

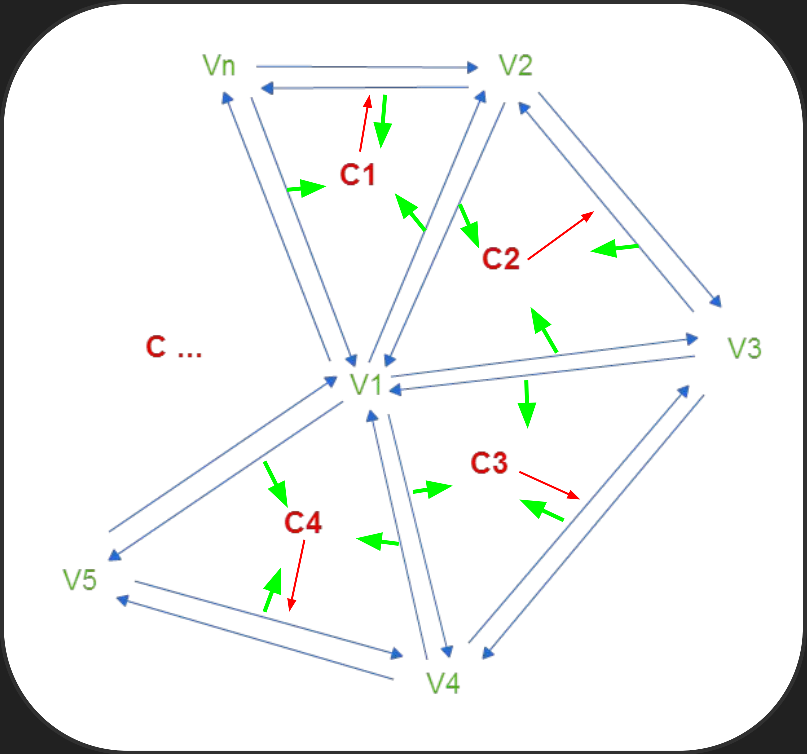
- Para buscar los triángulos que comparten un vértice v tenemos que realizar un algoritmo en O(n)
- n cantidad de triángulos*3

Mallas - Estructura

- Agregar un vértice o triángulo es rápido.
- Suprimir un vértice es lento porque por el vértice a suprimir necesitamos encontrar y suprimir todos los triángulos de la lista de índices.
- Una posibilidad es almacenar por cada vértice, una lista de triángulos vecino (índices que apuntan hacia la lista de triángulos)
  - Facilita procesar las mallas.
  - Construcción de esta estructura es muy simple

Mallas - Half-Edge Data Structure

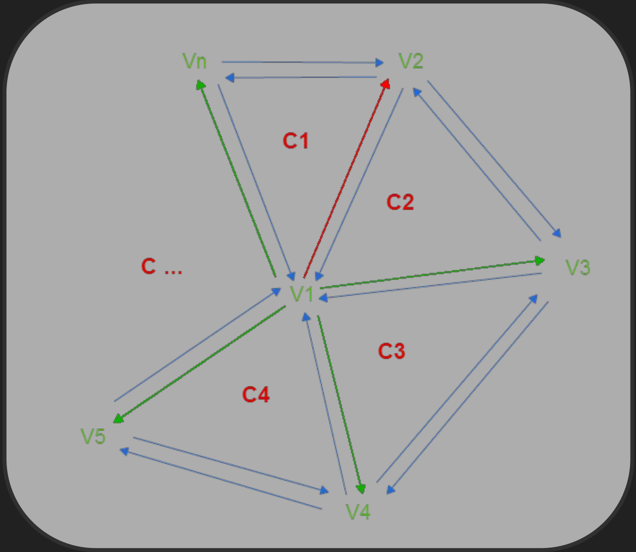
- Vértice (Vn) (*Vertex)*
  - Puntero hacia una de sus aristas salientes
  - Posición
  - Color/Normal/etc.
- Cara (*face)*
  - Puntero hacia una de sus aristas ( )
  - Normal/etc.
- Media arista ( ) (*Edge)*
  - Puntero hacia su vértice cabeza
  - Puntero hacia su arista anterior/siguiente
  - Puntero hacia su arista opuesta (twin)
  - Puntero hacia su cara ( )



Mallas - Half-Edge Data Structure

```
//v actual vértice

Face* f;

//arista saliendo del actual vértice v

Edge* e=v->edge;
```

```
do
{
    f=e->face;
    //tratar la cara f
    // ...

    /*Cambiamos la arista por la siguiente de
la   arista opuesta al actual arista*/
    e=e->twin->next;
}while(e != v->edge);
```

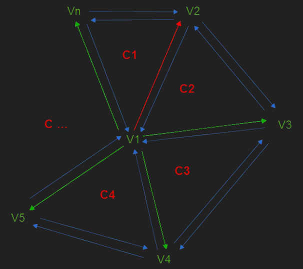



```
//tengo un vertice v
for(int i=0; i< nfaces;++i)
    for(int j=0;j<3;++j)
        if(v.id == faces[i*3+j])
        //tratar el triangulo
        //faces[i*3]
```

Mallas - Half-Edge Data Structure

```
    //v actual vértice

    Face* f;

    //arista saliendo del actual vértice v

    do
    {
        f=e->face;
        //tratar la cara f
        // ...

        /*Cambiamos la arista por la siguiente de
la   arista opuesta al actual arista*/
        e=e->twin->next;
    }while(e != v->edge);
```

- La estructura de datos permite realizar el algoritmo anterior con una complejidad algorítmica de O(n), con n = cantidad de caras que comparten el vértice v. En general son 6 triángulos, esta cantidad en práctica varía.
- La enorme ventaja aquí es que este algoritmo no depende del tamaño del objeto.
- Este algoritmo para todos los vértices de objeto se realiza en O(n) n=cantidad de vértices. :)



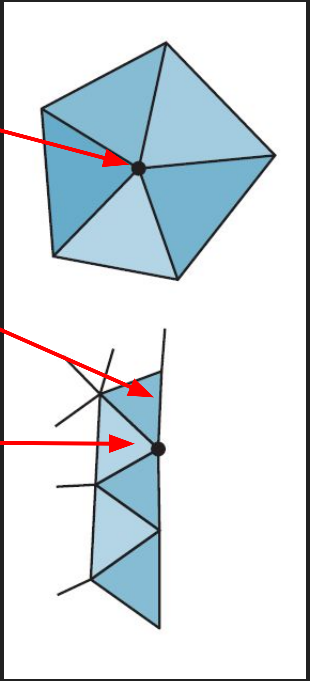
Mallas - Half-Edge Data Structure

- Rendimiento alto
- Algoritmos más prácticos
- Construcción rápida de la estructura

- Construcción de la estructura a la lectura de los datos
- Algoritmos más difíciles de leer

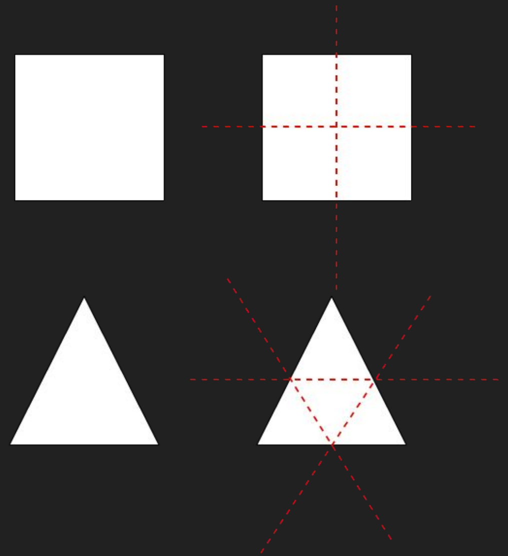
Malla - Frontera

- Vértice interior (Interior vertex)
- Arista frontera (boundary edge)
    - En la estructura de datos esta arista tiene un puntero hacia una cara = NULL
- Vértice frontera (boundary vertex)
    - En la estructura de datos este vértice tiene al menos dos aristas que tienen un puntero hacia una cara = NULL



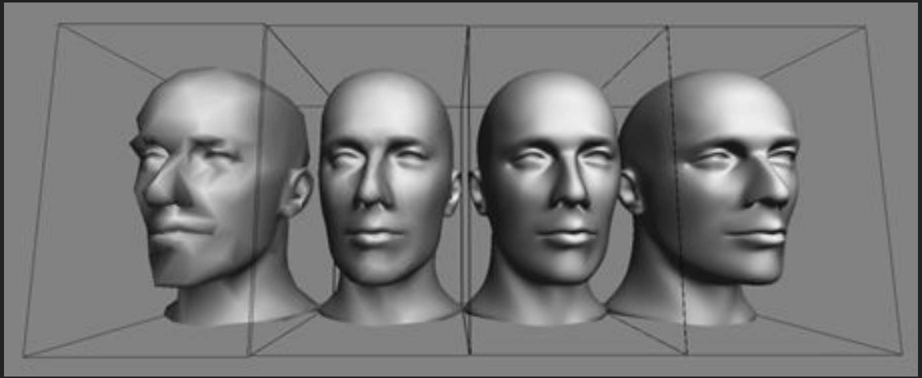
Subdivisión de superficie

- Caras rectangulares
  - Agregar 5 vértices
  - 4 -> 12 aristas
- Caras triangulares
  - Agregar 3 vértices
  - 3 -> 9 aristas



Subdivisión de superficie

- Transformar un modelo tosco en un modelo fino
  - Aproximación de una superficie lisa
    - Respecto del C2 vértice regular
    - Respecto del C1 vértice irregular
    - Convergencia aparente rápida
- Método bastante usual
  - Diseñadores
  - Videojuegos
  - Modelos 3D escaneados

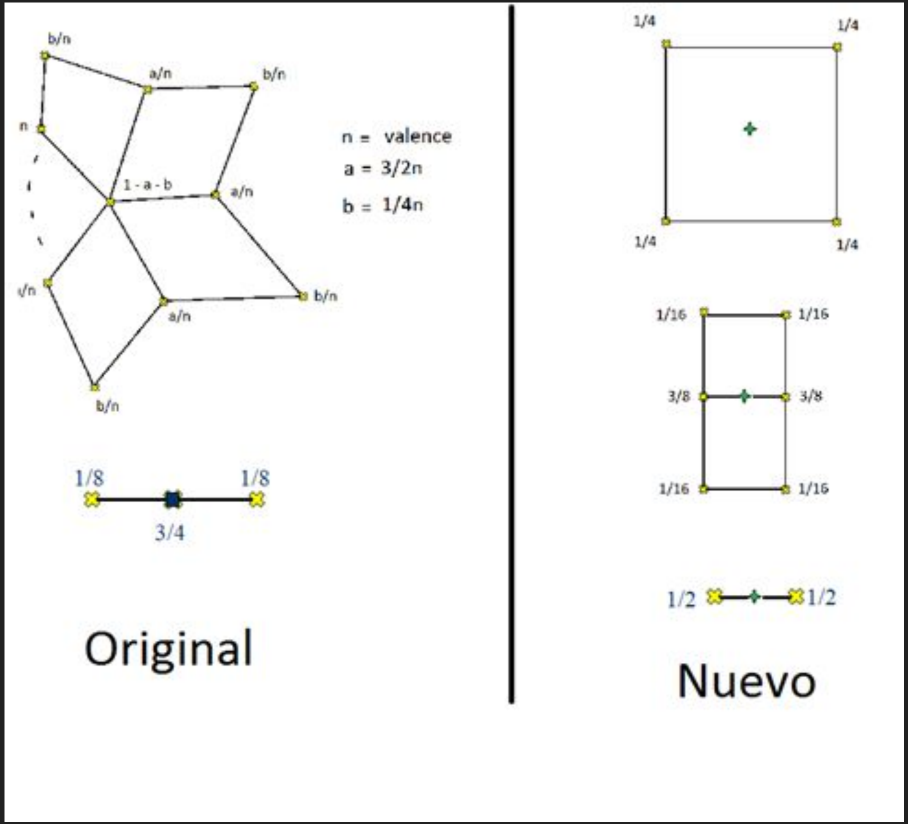


Subdivisión de Catmull - Clark

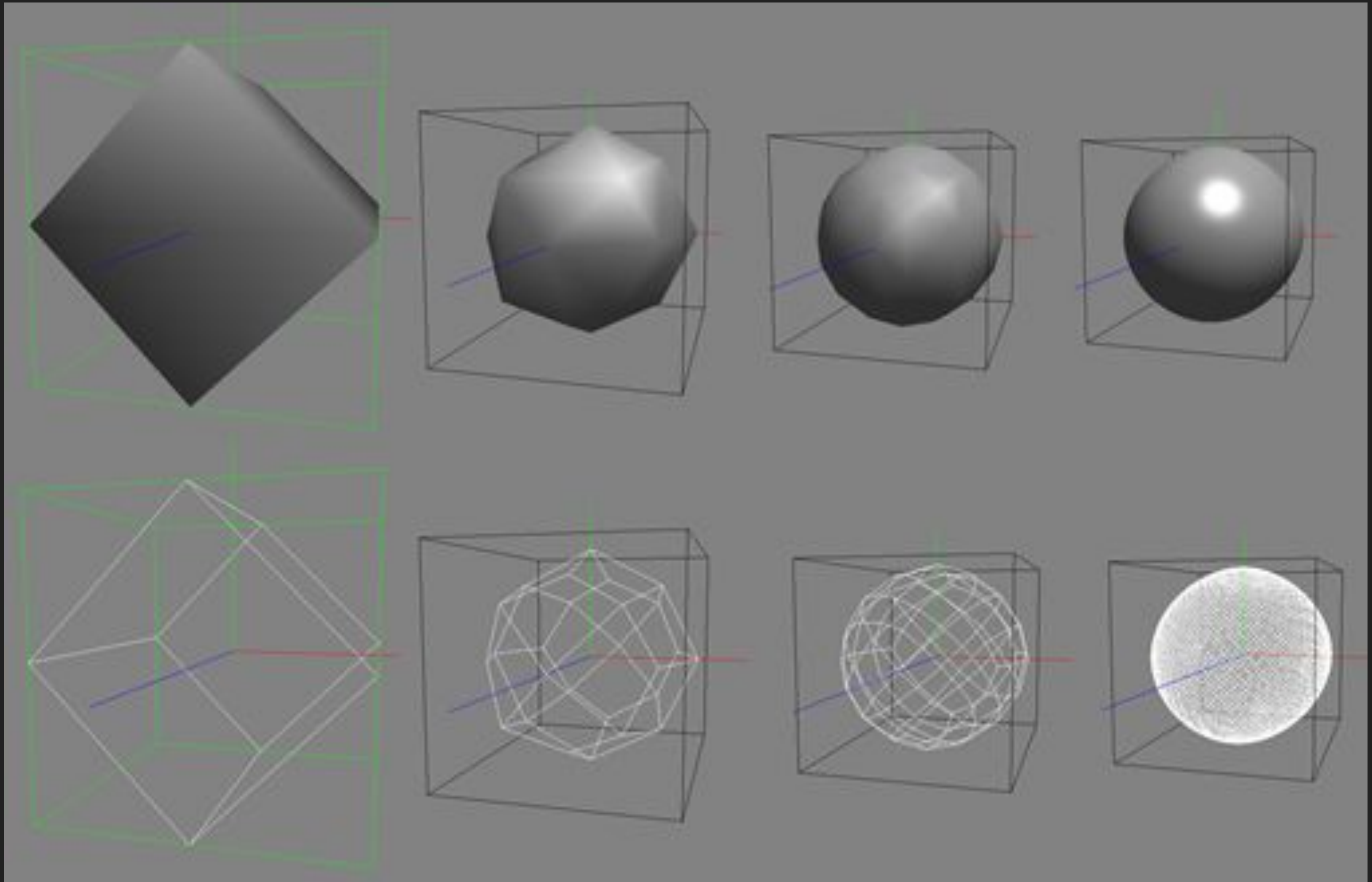
- 1978
- Edwin Catmull
- Presidente de Disney y Pixar
- Jim Clark
- Silicon Graphics Industry
- Caras cuadradas
- Vértice regular 4 vecinos directos
- Geri's Game 1997

Subdivisión de Catmull - Clark

- Crear nuevos vértices
- Nuevas posiciones
  - suma ponderada de los vértices
  - originales del vecindario del vértice a mover
  - Punto interior original
    - P=(1-a-b).P+∑(a.Pd)+∑(b.Pi)
- Contracción de la malla



Subdivisión de Catmull - Clark

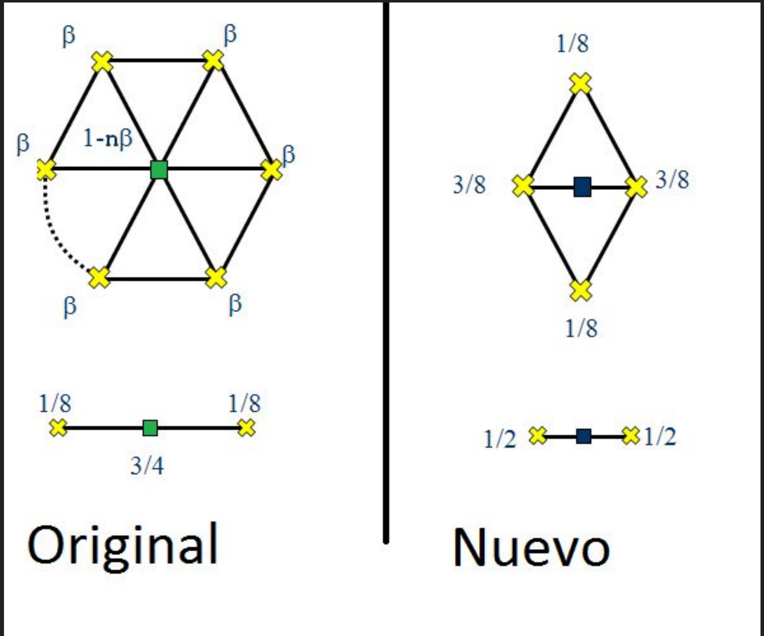


Subdivisión de Loop

- 1987
- Charles Loop
  - Microsoft Research
- Caras triangulares
- Vértice regular 6 vecinos directos

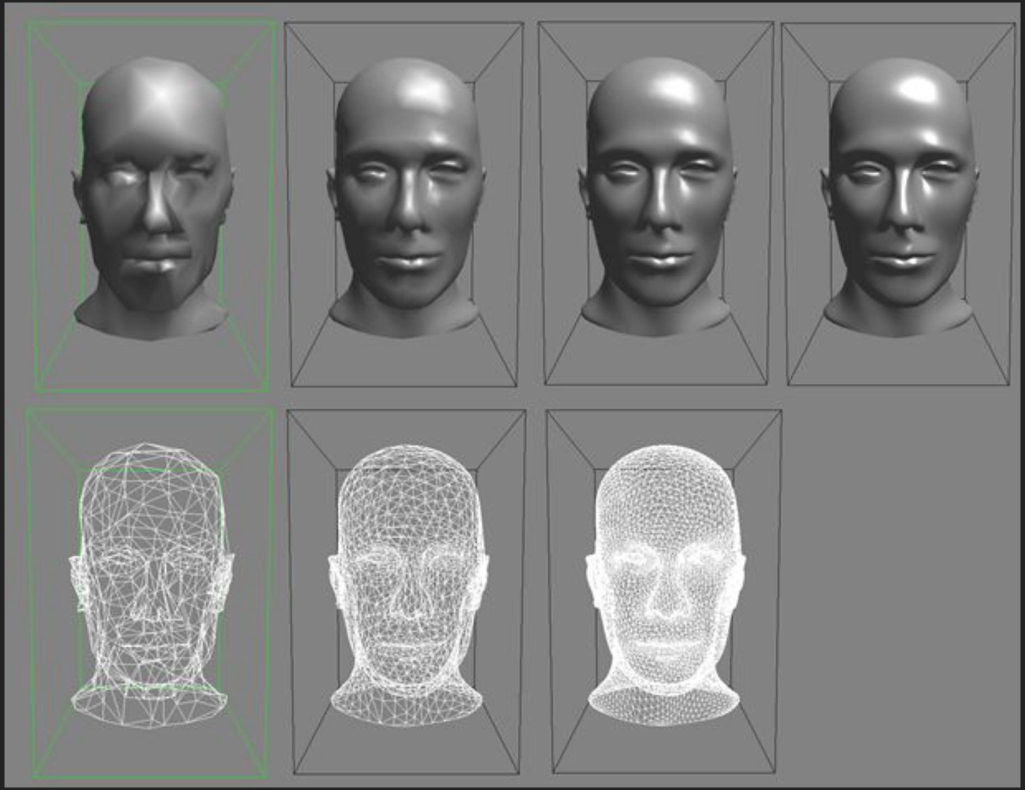
Subdivisión de Loop

- Crear nuevos vértices
- Nuevas posiciones
  - Suma ponderada de los vértices originales del vecindario del vértice a mover
  - Punto interior original
    - P =(1-nβ)∗P+ ∑ (β∗Pk)
- Contracción de la malla
- Pre-calcular β por diferentes valores de n
  - n ∈ [3, 10] por ejemplo



$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left[\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right]^2\right)$$

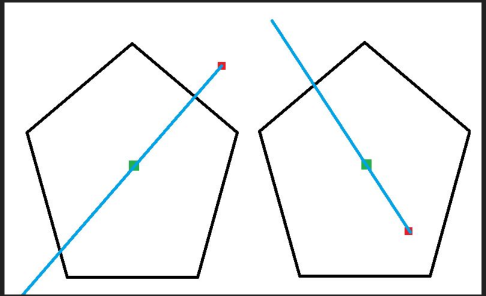
Subdivisión de Loop

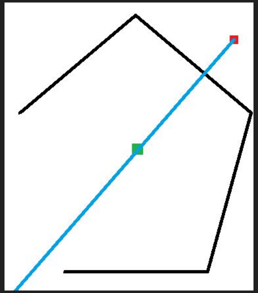


Fusión

- Creación de una escena
  - Fusionar dos objetos 3D (unión booleana)
- Mallas triangulares
- Etapas
  - Suprimir los vértices inútiles
  - Calcular las intersecciones
  - Construcción de los polígonos
  - Triangulación de los polígonos

Supresión de vértices



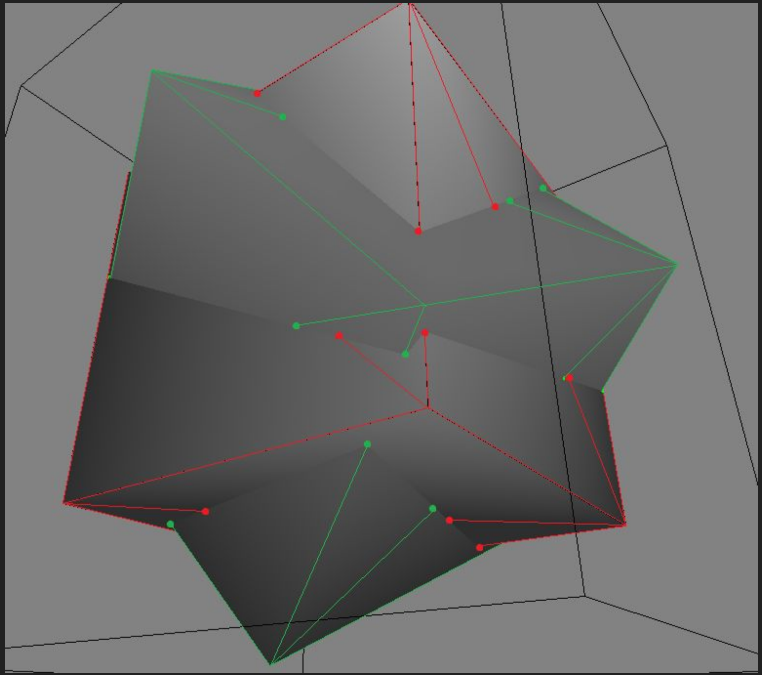


- Guardar los puntos afuera de la unión
- Lanzamiento de rayos
  - Par : afuera
  - Impar : adentro
- Problema con las mallas abiertas
  - Solución ?
    - Tapar los huecos (objeto no plano)

Intersecciones

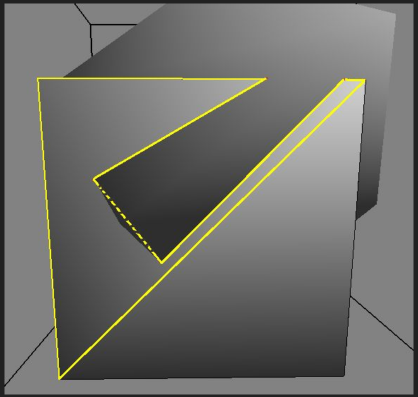
- Por cada arista del objeto 1
  - Calculamos las intersecciones con las caras de objeto 2
- Ídem por las aristas del objeto 2

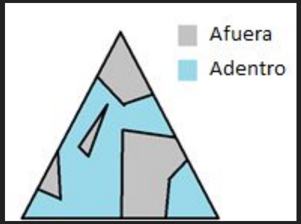
- Intersecciones
  - Nuevos vértices
- Estructura de datos
  - Lista ordenada de intersecciones por arista
  - Lista de intersecciones por cara



Construcción de polígonos

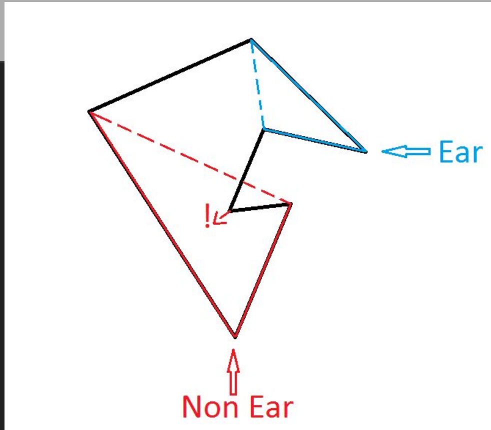
- Construir por cada cara los polígonos
- Polígonos cóncavos
- Huecos





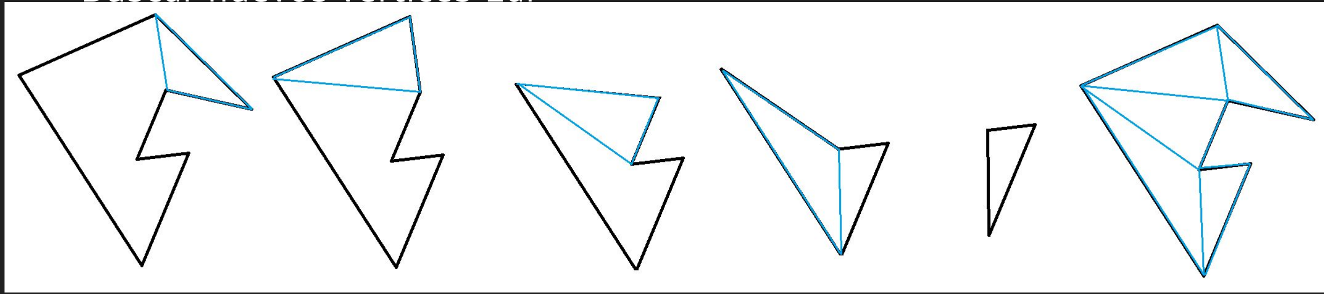
Triangulación por Ear clipping

- Vértice Ear
- V_i Ear : un vértice convexo para el cual el triangulo $V_{i-1}\ \ V_i\ \ V_{i+1}$ no contiene ningún otro vértice del polígono



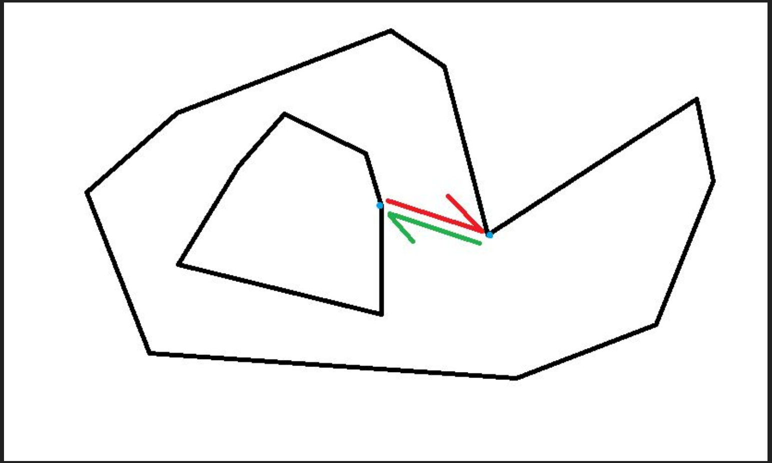
Triangulación por Ear clipping

- Algoritmo recursivo
  - Supresión de los vértices « Ear »
  - Creación de los triángulos
  - Buscar nuevos vértices Ear

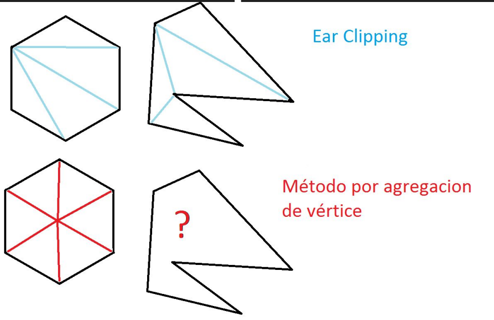


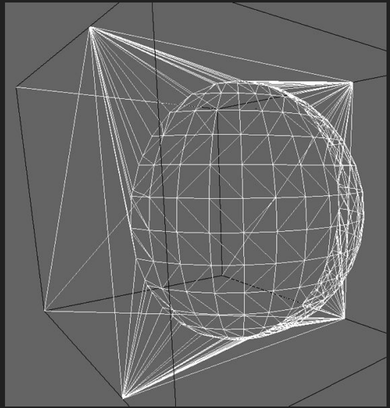
Polígonos con huecos

- Suprimir el hueco
- Cortar el polígono
  - Punto más cerca visible



Triangulación por Ear clipping





- Problemas
  - Deformación al subdividir

Video

[Video](): Software para subdivisión de superficies